EPFL

Thèse n° 9260

Loss landscape and symmetries in Neural Networks

Présentée le 8 décembre 2021

Faculté des sciences de base
Laboratoire de physique des systèmes complexes
Programme doctoral en physique

pour l'obtention du grade de Docteur ès Sciences

par

Mario GEIGER

Acceptée sur proposition du jury

Prof. R. Houdré, président du jury
Prof. M. Wyart, directeur de thèse
Prof. M. Mézard, rapporteur
Dr J. Bruna, rapporteur
Prof. L. Chizat, rapporteur

École
polytechnique
fédérale
de Lausanne

2021

Acknowledgements

For this dissertation: For accepting to read and report my work, I warmly thank Lenaïc Chizat, Joan Bruna, Houdre Romuald and Marc Mézard. And Matthieu for having reread it hundreds of times.

For those years: I thank Matthieu Wyart for his supervision during my doctoral studies in the PCSL (Physics of Complex Systems Laboratory) group. I particularly appreciate to have learn his way of doing physics. I thank his constant support and guidance during the time I spent here. I thank Taco Cohen for the great collaboration he proposed me, his mentorship in the topic of equivariant neural networks and the fun I had in Amsterdam. I would like to thank Tess Smidt for the great collaboration and all her support in many topics. I thank all the members of the PCSL group for all the fun discussions and the outdoor activities. Special thanks to Jonas for his hiking plans, Corinne for the positive spirit and fun stories, Levent, Marko, Tom, Stefano, Riccardo, Jonas for the friendship. I want to thank Stefano for his constant help during the first year of my PhD. And both Stefano and Riccardo for the moral support. I thank the people I met during the summer schools: Florent, Lenka, Giulio, Stéphane, Marylou, Antoine and all the others. I would like to thank the members of the e3nn collaboration: Alby, Simon, Ben, Bradley, Josh, Kostiantyn, Martin, Thomas, and the ones I forgot to mention, for their contribution to the library. I thank the people with whom I spent most of my time during my PhD: Camilo, Thibault, Quentin, Ivan, Darja, Anne-Sophie, Robin, Ariane, Danilo (online), Yseult, Nicolas, Carmen, Quentin, Velia, Axelle, Youssef. I finally thank my family. My dad for never doubting that everything would work out. My mom to have shown me the periodic table of elements when I was a child.

Lausanne, November 19, 2021 M. G.

Abstract

Neural networks (NNs) have been very successful in a variety of tasks ranging from machine translation to image classification. Despite their success, the reasons for their performance are still not well-understood. This thesis explores two main themes: loss landscapes and symmetries present in data.

Machine learning consists of training models on data by optimizing the model parameters. This optimization is done by minimizing a loss function. NNs, a family of machine learning models, are created by composing functions, called layers. Informally, they can be visualized as a set of interconnected neurons.

Ten years ago, NNs became the most popular models of machine learning. With their success come many open questions. For example, neural networks and glassy systems both have many degrees of freedom and highly non-convex objective or energy functions, respectively. However, glassy systems get stuck in local minima near where they are initialized, whereas neural networks avoid this even when they 100s of times more parameters than the number of data use to train them? (i) What drives this difference in behavior? (ii) How is it then that NNs do not become too specialized to the training data (overfitting)?

In the first part of this thesis, we show that in classification tasks, NNs undergo a jamming transition dependent on the number of parameters, N. This answers (i): With a sufficiently high N above a critical number N^*, local minima are avoided. Then, we establish a "double-descent" behavior in the test error of classification tasks: It decreases twice as a function of N, before N^* but also after, until infinity, where it converges to its minimum. We answer (ii) by explaining the origins of this double-descent. Finally, we introduce a phase diagram that describes the landscape of the loss function and unifies the two limits in which a neural network can converge when sending N to infinity.

In the second part of this thesis, we explore the issue of the curse of dimensionality (CD): Sampling a d-dimensional space requires an exponential number of points P. However, NNs perform well even for $P \ll \exp(d)$. Symmetries in the data play a role in this conundrum. For example, to process images we use convolutional NNs (CNNs) which have the property of being locally connected and equivariant with respect to translations, i.e., a translation in the input leads to a corresponding translation in the output. Although empirical experience suggests that locality and equivariance contribute to the success of CNNs, it is difficult to understand how. Indeed, equivariance reduces the dimensionality of the data only slightly. Stability toward diffeomorphisms however might be the key to CD. We studied how NNs are affected by images distorted by diffeomorphisms. Our results suggest that locality and

equivariance allow, during learning, to develop stability towards diffeomorphisms *relative* to other generic transformations.
Following this intuition, we have created new architectures by extending CNNs properties to 3D rotations.
Our work contributes to the current understanding of the behavior of neural networks empirically observed by machine learning practitioners. Moreover, the architectures developed for 3D rotation problems are currently being applied to a wide range of domains.

Résumé

Les réseaux de neurones (NNs) ont connu un grand succès dans une variété de tâches qui vont de la traduction automatique à la classification d'images. Malgré leur succès, leur fonctionnement est encore mal compris. Cette thèse explore deux thèmes : leur dynamique d'apprentissage ainsi que les symétries présentes dans les données.

L'apprentissage automatique consiste à entraîner des modèles sur des données en optimisant les paramètres du modèle. Cette optimisation est faite en minimisant une fonction de coût. Les NNs, une famille de modèles utilisables dans l'apprentissage automatique, sont créés en composant des fonctions, appelées couches. Informellement, ils peuvent être visualisés comme un ensemble de neurones interconnectés.

Il y a dix ans, les NNs sont devenus les modèles favoris de l'apprentissage automatique. Avec leur succès viennent des énigmes. Parmi elles, la fonction de coût est similaire à la fonction d'énergie d'un système physique vitreux : non-convexe avec beaucoup de degrés de libertés. (i) Comment se fait-il donc que les NNs ne soient pas, comme les systèmes vitreux, piégés dans un minimum local proche de l'initialisation ? Autre énigme, les meilleurs NNs peuvent avoir jusqu'à cent fois plus de paramètres que le nombre de données utilisées pour les entraîner. (ii) Comment se fait-il donc que les NNs ne se spécialisent pas trop aux données d'apprentissage (surapprentissage) ?

Dans la première partie de cette thèse, nous montrons que dans les tâches de classification, les NNs subissent une transition de jamming contrôlée par le nombre de paramètres N. Ce qui répond à (i) : Avec un nombre suffisant de paramètres N supérieurs à un nombre critique N^*, les minima locaux sont évités. Ensuite, nous découvrons la "double-descente" dans l'erreur de test des tâches de classification : Elle décroît deux fois en fonction de N, avant N^* mais également après, jusqu'à l'infini où elle converge vers sont minimum. Nous répondons à (ii) en expliquant les origines de la double-descente. Finalement, nous introduisons un diagramme de phase qui décrit le paysage de la fonction de coût et unifie les deux limites dans lesquelles un réseau de neurones peut converger lorsqu'on envoie N à l'infini.

Dans la deuxième partie de cette thèse, nous explorons la question de la malédiction de la dimensionnalité (CD) : Échantillonner un espace de dimension d, nécessite un nombre exponentiel de points P. Cependant, les NNs ont de bonnes performances, même pour $P \ll \exp(d)$. Les symétries contenues dans les données jouent un rôle dans cette énigme. Par exemple, pour traiter des images on utilise des NNs convolutionnels (CNN) qui ont la propriété d'être équivariants par rapport aux translations : Les translations commutent avec toutes leurs couches. Bien que l'expérience empirique suggère que l'équivariance contribue au succès

des CNN, on comprend mal comment. En effet, elle ne réduit que peu la dimensionnalité des données. Nous avons étudié comment les NNs sont affectés par des images déformées par des difféomorphismes. Nos résultats suggèrent que l'équivariance permet, au cours de l'apprentissage, de développer une stabilité envers les difféomorphismes.
Suivant cette intuition, nous avons créé de nouvelles architectures en étendant cette propriété aux rotations en 3D.
Nos résultats proposent des réponses aux questions que se posent les praticiens des réseaux de neurones. Nos architectures pour les problèmes liés aux rotations 3D sont maintenant fréquemment utilisées.

Contents

Introduction

Machine learning is the class of algorithms which train models on data by tuning parameters. **Supervised learning** uses machine learning and annotated datasets to solve regression and classification tasks. Both of these tasks consist of finding a predicting function that given some input, predicts the correct output. Here is an example of a regression task: predict tomorrow's temperature, given meteorologic data from the past days. And here is one of a classification tasks: classify bird songs into their corresponding bird specie.

Neural networks are a family of models made of an alternate composition of parametrized linear functions and non-linear functions. Each repetition of a linear function followed by a non-linear function is called a **layer**. **Deep** neural networks have many layers LeCun et al. (2015). The simplest **architecture** (kind) of neural network is the **Fully Connected network** (FC).

0.1 Fully-connected Neural Networks

Neural networks are parametrized functions:

$$f : W \times X \to Y \tag{1}$$

$W = \mathbb{R}^N$ is the space of parameters, $X = \mathbb{R}^d$ is the input space and Y is the output space. The main task of the neural network practitioner is to chose an architecture that has the suited input and output format to the problem he has to solve.

The most versatile and simple architecture is the Fully-Connected (FC) network. It is however not the most efficient. Its operations can be suggestively visualized as a network of interconnected neurons. In this neuronal analogy, the FC is made of successive layers of neurons where neurons of each layer are *fully-connected* to the neurons of the next layer. Figure 1 shows a fully-connected neural network of $L = 3$ hidden layers of fixed width (same h for all hidden layers) of $h = 9$ neurons.

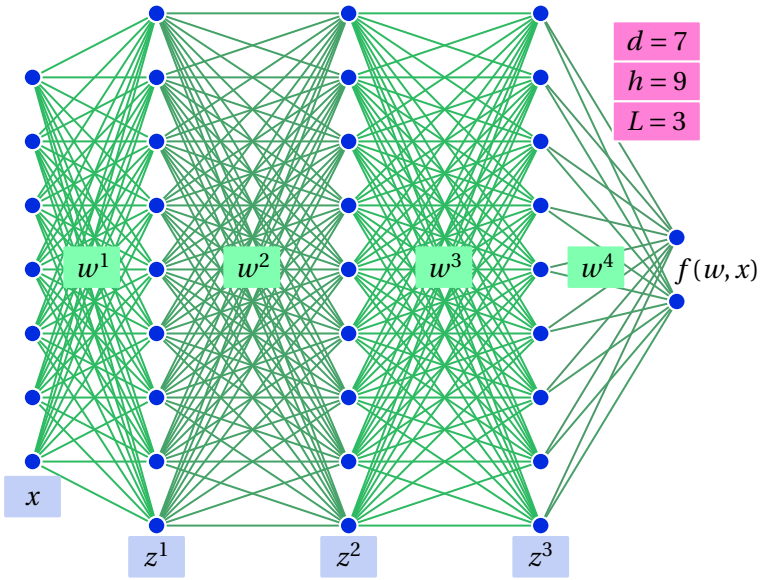


Figure 1 – A fully-connected neural network with $L = 3$ hidden layers of $h = 9$ neurons and two readout neurons. Information propagates from left-to-right. This network contains $N = 7 \times 9 + 2 \times 9^2 + 9 \times 2 = 243$ parameters.

For an input $x \in X = \mathbb{R}^d$, the output of the network ($Y = \mathbb{R}^c$) is given by the recurrence formula,

$$\begin{aligned} f(w,x) &= h^{-a_{L+1}} w^{L+1} z^L(x) + b^{L+1} & z^L(x) &= \phi(\tilde{z}^L(x)) \\ \tilde{z}^{l+1}(x) &= h^{-a_{l+1}} w^{l+1} z^l(x) + b^{l+1} & z^l(x) &= \phi(\tilde{z}^l(x)) \\ \tilde{z}^1(x) &= d^{-a_1} w^1 x + b^1 \end{aligned} \tag{2}$$

where $w = \{w^1, \ldots, w^{L+1}, b^1, \ldots, b^{L+1}\}$ are the parameters, $w^1 \in \mathbb{R}^{h \times d}$, $w^2 \ldots w^L \in \mathbb{R}^{h \times h}$ and $w^{L+1} \in \mathbb{R}^{c \times h}$ are called the weights and $b^1 \ldots b^L \in \mathbb{R}^h$ and $b^{L+1} \in \mathbb{R}^c$ are the biases. $\tilde{z}, z$ are the pre- and post-activations, $\phi : \mathbb{R} \to \mathbb{R}$ is the activation function, we will often use the `relu`: $\phi(z) = \max(0, \sqrt{2}z)$.[1] The total number of parameters is $N = dh + (L-1)h^2 + hc \approx Lh^2$.

The choice for the constants $\{a_l\}$ and the probability distributions used to initialize the weights and biases are part of the **initialization**. Many different choices of initialization exist: Xavier is balancing forward and backward passes Glorot and Bengio (2010), He adapts Xavier for `relu` He et al. (2015), sparse initialization Martens (2010), orthogonal initialization Hu et al. (2020). The two initializations we use **NTP** and **MFP** are listed in Table 1. They are inspired from Jacot et al. (2018); Yang and Hu (2020). In both of them, the matrices are drawn from the normal distribution $w^l_{\alpha\beta} \sim \mathcal{N}(0,1)$[2] and the biases are initialized to zero.

[1] Assuming the distribution of the pre-activation to be symmetric, adding the factor $\sqrt{2}$ is a good practice to ensure that the second moment of the pre- and post-activations remain equal.

[2] Comparing to the notation in Yang and Hu (2020), we only consider parametrizations with $b_l = 0$.

	Definition	NTP	MFP[3]
a_l	{previous width}$^{-a_l} w^l$	1/2	$\begin{cases} 1/2 & l \le L \\ 1 & l = L+1 \end{cases}$

Table 1 – Neural Tangent (NTP) and Mean Field (MFP) parametrizations. Notation taken from Yang and Hu (2020).

0.2 Algorithms

Neural networks are trained by minimizing a **loss** function that measures how poorly the network fits the **trainset** (subset of the available data used for the training). The trainset, $\mathscr{R}$, contains P pairs of input/output data point.

In the case of classification we need to distinguish the output of the predictor and the labels of the dataset. Therefore Y will denote the set of outputs and $\bar{Y}$ will denote the set of labels. For the classification among $c > 2$ classes: $Y = \mathbb{R}^c$ and $\bar{Y} = \{1, \dots, c\}$. For the case of binary classification: $Y = \mathbb{R}$ and $\bar{Y} = \{\pm 1\}$.

Gradient based algorithms use a loss function defined as

$$\mathscr{L}(\mathscr{T}, f) = \frac{1}{|\mathscr{T}|} \sum_{x,y \in \mathscr{T}} \ell(f(x), y) \tag{3}$$

where a predictor $f : X \to Y$ is evaluated on a set of inputs/labels $\mathscr{T} \subset X \times \bar{Y}$ using an loss per sample $\ell : Y \times \bar{Y} \to \mathbb{R}$.

The mathematically simplest optimization algorithm is **gradientflow**, it is defined as the following partial differential equation

$$\dot{w}(t) = -\frac{d}{dw} \mathscr{L}(\mathscr{T}, f(w(t))) \tag{4}$$

however its implementation is non trivial.

Instead, the most common gradient based algorithm is **Stochastic Gradient Descent** (SGD) Robbins and Monro (1951). At each **epoch**, the trainset $\mathscr{R}$ is randomly cut in batches of size b:

$$\mathscr{R} = \mathscr{T}_1 \cup \mathscr{T}_2 \cup \mathscr{T}_3 \cup \dots \qquad |\mathscr{T}_i| = b \tag{5}$$

Using a **learning rate** η, each step of SGD updates the parameters

$$w_{i+1} = w_i - \eta \nabla \mathscr{L}(\mathscr{T}_i, f(w_i)) \tag{6}$$

Once all the batches has been used, we sample new batches and start the next epoch. The learn-

[3] A generalized version of the MFP of Yang and Hu (2020) for L hidden layers, it is equivalent to μP up to the symmetry of the abc-Parametrizations.

ing rate is typically decreased during the training. SGD is computationally efficient because $b \ll P$. However it is not the only motivation to use it, SGD improves the performance Keskar et al. (2017); Smith and Le (2018). It is not clear why: Some proposed a Fokker-Planck formalism with Ito calculus Chaudhari and Soatto (2018a), while others proposed that SGD regularize the variance of gradients Roberts (2021a); Smith et al. (2021b).

For classification the most common loss is the cross entropy defined as

$$\ell(p, y) = -\log \frac{\exp(p_y)}{\sum_{i=1}^{c} \exp(p_i)} \tag{7}$$

where $p \in Y$ are the prediction of the model and are called in this case **logits**.

For binary classification the cross entropy becomes

$$\ell(p, y) = -\log \frac{\exp(yp/2)}{\exp(p/2) + \exp(-p/2)} = \log(1 + \exp(-yp)) \tag{8}$$

With the cross entropy, the loss never reaches zero. The user has to stop the training at an arbitrary step. Usually the step to stop is determined by evaluating the loss on a subset of the dataset independent from the trainset called the **validationset**. The algorithm is stepped when the loss on the validationset stops decreasing.

0.3 Convolutional Neural Networks

A decade ago, neural networks became the favorite model in machine learning. Starting with the problem of image classification Marr and Poggio (1979). Every year since 2010, happened the largest contest of object recognition: ImageNet Large Scale Visual Recognition Challenge. For this challenge, a trainset of many images is provided to the participants who has to come up with an algorithm that is then tested on the testset by the organizers. In 2012, for the first time, a neural network with an architecture specialized for images, called a **convolutional neural network** (CNN) named AlexNet Krizhevsky et al. (2012), outperformed the competitors by more than 10%, the top-5 error drop from 26.1% to 15.3%. Since that time neural network kept improving and reach now a top-5 error of 2.1% Zhai et al. (2021). [4]

[5] Convolutional neural networks, inspired from the primate brain, perform much better than FCs for a variety of tasks including image classification. Each hidden layer is composed of a number of channels, each representing the data as a feature map – itself an image. CNNs perform the same operations at distinct locations in the image, and enforce that these operations are local. CNNs are thus more constrained than FCs, and, as illustrated in Figure 2 for one dimension, they can be obtained from the latter by imposing that some weights are

[4] https://paperswithcode.com/sota/image-classification-on-imagenet reports all the best architectures for ImageNet.

[5] Paragraph adapted from Geiger et al. (2021)

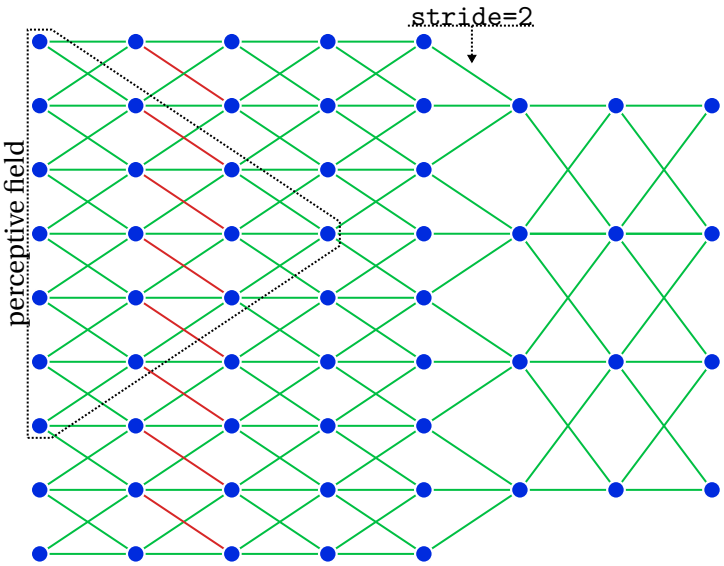


Figure 2 – A 1-dimensional CNN. Neurons are connected to their neighbors of the previous layer. Weights are shared: for instance all the red weights have the same value. At some layer, the feature map resolution is reduced, here we show how it is done with the method called stride. It requires many layers for the information to propagate from one side of the feature map to the other side, the perceptive field of a neuron are the neurons of the previous layers connected to it.

identical (to enforce translational equivariance) and other zero (to enforce locality).

0.4 Questions

Since the success of AlexNet in 2012 Krizhevsky et al. (2012), neural networks shined in wide range of tasks including speech Amodei et al. (2016) and text recognition Shi et al. (2016), self-driving cars Huval et al. (2015) and beating the best humans at Go Silver et al. (2017) or video games Mnih et al. (2013). Neural networks are also used to create artificial images Karras et al. (2020) and songs Topirceanu et al. (2014). Strikingly, the technological success of neural networks is not understood. Among the open questions: Why the non convexity of the loss does not leads to a dynamic stuck into local minima of high loss? Why the overparameterization does not hurt performance by overfitting? How neural networks beat the curse of dimensionality? These questions are developed now.

0.4.1 Non convex loss

[6] A problem which made deep learning less popular in the 90's, concerns learning. The loss function has no guarantees to be convex, and the parameters space is high-dimensional. What

[6] Section partially adapted from Geiger et al. (2021)

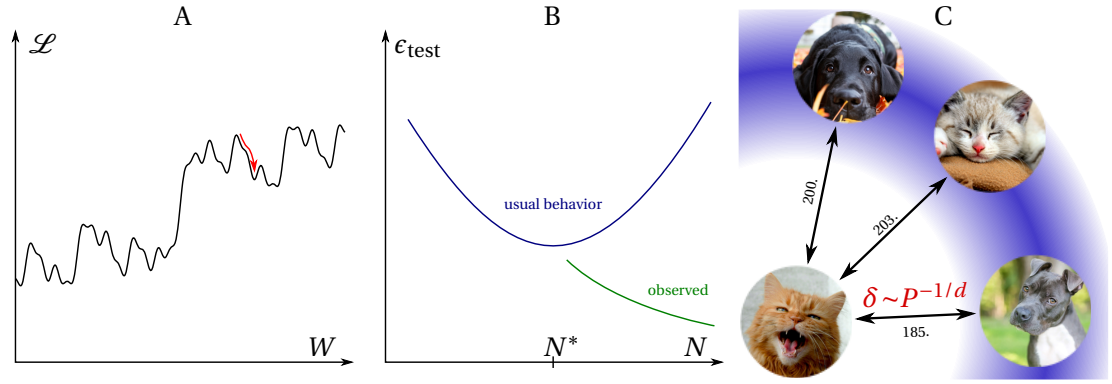


Figure 3 – A: Sketch of the loss landscape. Starting from a random initialization, the dynamics could get stuck in a bad minimum of the loss. B: Usual behaviour (blue line) of the test error as a function of the number of parameters, expected to be minimal when N is of order of the smallest value N^* where data can be fitted. Green: observations indicate that for deep learning, increasing the number of parameters generally improves behaviour, even in a regime where data are perfectly fitted. C: Curse of dimensionality. In high dimensions, every point lies far away from its neighbours, forbidding classification based on distances alone.

prevents then the learning dynamics to get stuck in poorly performing minima with high loss, as sketched in Figure 3.A? In other words, under which conditions can one guarantee that training data are well fitted? Is the loss landscape similar to the energy landscape of glassy systems in physics, where the number of minima is exponential in the number of degrees of freedom Berthier and Biroli (2011); Choromanska et al. (2015)?

The picture seems different than the glassy system. According to Lipton (2016), the loss landscape of a small CNN trained on MNIST[7] is locally non-convex, however weights do not converge to critical points, instead traveling large distances through flat basins in weight space.

Several works support that the loss function is characterized by a connected level set: any two points in parameter space with identical loss are connected by a path in which the loss is constant: (i) The quadratic loss of `relu` network with a single hidden layer is asymptotically connected Freeman and Bruna (2017). (ii) The training error is zero at every differentiable local minimum of the quadratic loss of a FC Soudry and Carmon (2016). (iii) For regression, the zero loss set is a manifold of dimension $N - P$ Cooper (2018). (iv) For over-parametrized networks, the hessian has a sharp concentration of eigenvalues at zero Sagun et al. (2017a,b).

Other works argue that the dynamics act as an implicit regularizer that brings the weights toward specific places of the landscape: (i) Gradient based algorithm with an alternative geometry for the weight space provide beneficial implicit regularization Neyshabur et al. (2017). (ii) In the same spirit Ji et al. (2021) proposed to use as metric the Fisher information

[7] MNIST is a dataset of 70000 handwritten digits. Each digits is a 28 pixels square image.

matrix.

All these studies concerns over-parametrized networks. Under-parametrized neural networks are however glassy Baity-Jesi et al. (2018). It raises the question whether there is a transition between the under and over parametrized phases? And what is the nature of this transition? Does it has consequences on the performance?

0.4.2 Overparameterization

ImageNet trainset contains $P = 10^6$ images. A surprising fact is that the State-Of-The-Art networks for image classification have around $N = 10^8$ parameters.[8] It is 100 times more than P the number of images in the trainset. How is it that neural network don't overfit while they are typically overparametrized i.e. $P \ll N$?

[9] In that regime, their capacity is very large: they can fit the trainset even if labels are randomized Zhang et al. (2017). Statistical learning theory then gives no guarantees that over-fitting does not occur, and that the model learnt has any predictive power. Indeed from statistics text books one expects a U-shape learning curve relating the test error to the number of parameters of the model, as depicted in Figure 3.B. In this view, at small N the hypothesis class of the learning algorithm is too small, leading to a bias in the learnt predictor. At large N, it presents a large variance due to over-fitting noise in the data. A very puzzling aspect of deep learning is that increasing N passed the point where all data are perfectly fitted does not destroy predictive power, but instead *improves* it Neyshabur et al. (2017, 2018); Bansal et al. (2018); Advani et al. (2020) as sketched in Figure 3.B.

0.4.3 Curse of dimensionality

[10] Once learning is done, generalization performance can be estimated from a **testset** (data not used to train, but whose distribution is identical to the trainset) by computing the probability to misclassify one new datum, the so-called test error ϵ. How many data are needed to learn a task is characterized by the learning curve $\epsilon(P)$.[11] Generally, it is observed that the test error is well described by a power law decay $P^{-\beta}$ in the range of trainset size P available[12] with an exponent β that depends jointly on the data and the algorithm chosen. In Hestness et al. (2017), β is reported for SOTA architecture for several tasks: see Table 2.

General arguments suggest that β should be extremely small — and learning thus essentially

[8] The architecture `VOLO-D5` is the current SOTA with top-1 error of 12.9% Yuan et al. (2021). It contains $N = 2.96\,10^8$ parameters.

[9] Paragraph adapted from Geiger et al. (2021)

[10] Section adapted from Geiger et al. (2021)

[11] A different network is trained for every P.

[12] For datasets where a finite fraction of the trainset is mislabeled, one expects the test error to eventually plateau to a finite value. However, such a saturation of the test error is not seen in practice for various benchmarks, suggesting that such a noise is small in magnitude.

Task	$\beta \approx$	$d \approx$
translation	0.3-0.36	NA
language modeling	0.06-0.09	10^4
speech recognition	0.3	NA
image classification	0.3-0.5	10^5

Table 2 – Data from Hestness et al. (2017)

impossible — when the dimension d of the data is large, which is generally the case in practice. For example in a regression task, if the only assumption on the target function is that it is Lipschitz continuous, then the test error cannot be guaranteed to decay faster than with an exponent $\beta \sim 1/d$ Luxburg and Bousquet (2004). This *curse of dimensionality* Bach (2017) stems from the geometrical fact that the distance δ among nearest-neighbour data points decays extremely slowly in large d as $\delta \sim P^{-1/d}$, as depicted in the Figure 3.C. Thus, interpolation or classification methods based on distances are expected to be very imprecise for generic data.

0.5 Summary of results

This thesis is split into two main parts and a third small one: A study of the loss landscape where we give answers to the two first questions and depict a phase diagram for deep learning. A second part on symmetries where we seek insight on how CNNs manage to beat the curse of dimensionality and where we propose new architectures that could extend the success of CNNs to other data types. Finally, two extra works, less directly related to the main story, are reported at the end.

0.5.1 Part I: Landscape vs Parametrization

We showed a connection with the jamming phenomena, that we now introduce.

A Jamming Transition in Deep Learning

[13] Describing the energy landscape of physical systems is a much studied problem, especially in the context of glasses. Progress was made on this topic in the last fifteen years by considering finite range interactions, for which bringing the energy to zero is equivalent to satisfying a set of constraints. In that case, the landscape is controlled by a critical point O'Hern et al. (2003); Wyart (2005); J Liu et al. (2010), the so-called jamming transition. It occurs as the particle density ϕ increases. For $\phi > \phi_c$, a gradient descent from a random initial positions of the particles gets stuck in one – out of many – meta-stable states, corresponding to a glassy solid as depicted in Fig.4(B,D). For lower densities, the gradient descent reaches zero energy, as

[13] Section adapted from Geiger et al. (2021)

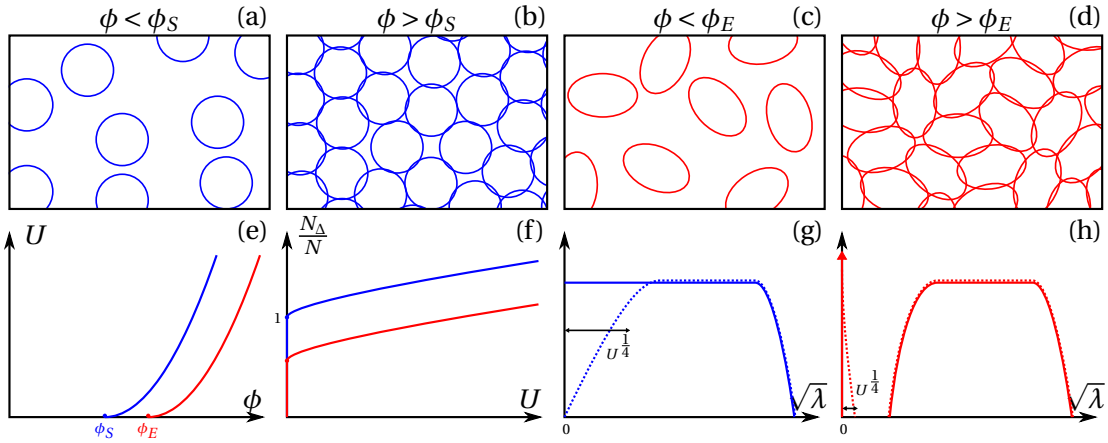


Figure 4 – Sketch of the jamming transition for repulsive spheres and ellipses. (a,b,c,d) Both systems transition from a fluid to a solid as the density passes some threshold, noted ϕ_S for spheres and ϕ_E for ellipses. (e) For denser packings, the potential energy U becomes finite. (f) The ratio N_Δ/N between the number of particles in contact N_Δ (corresponding to unsatisfied constraints) and the number of degrees of freedom N jumps discontinuously to a finite value, which is unity for spheres but smaller for ellipses. (g,h) This difference has dramatic consequence on the energy landscape, in particular on the spectrum of the Hessian. In both cases, the spectrum becomes non-zero at jamming, but it displays a delta function with finite weight for ellipses (indicating strictly flat directions), followed by a gap with no eigenvalues, followed by a continuous spectrum (h, full line). For spheres, there is no delta function nor gap (g, full line). As one enters the jammed phase, in both cases a characteristic scale $\lambda \sim \sqrt{U}$ appears in the spectrum (g and h, dotted lines). (Figure and caption taken from Geiger et al. (2019).)

sketched in Fig.4(E) : particles can freely move without restoring forces, and the landscape has many flat directions. It corresponds to the situations depicted in Fig.4(A,C).

There are two universality classes for jamming, leading to distinct properties for the curvature of the landscape (i.e. the spectrum of the Hessian) Wyart et al. (2005); DeGiuli et al. (2014); Franz et al. (2015); Brito et al. (2018); Mailman et al. (2009); Zeravcic et al. (2009), for the structure of the packing obtained Wyart (2012); Lerner et al. (2013); Charbonneau et al. (2014b,a) and for the dynamical response to a perturbation Lerner et al. (2012); DeGiuli et al. (2015). Spheres and ellipses fall in distinct classes as illustrated in Fig.4. More generally, the jamming transition occurs generically in satisfiability problems with continuous degrees of freedom (it can be defined with discrete degrees of freedom Krzakala and Kurchan (2007), but then differs qualitatively; in particular, the discussion below does not apply to the discrete case).

In the work presented in Chapter 1, we show that deep neural networks also undergoes a jamming transition: In the same spirit as introducing finite range interactions in glasses, by choosing the appropriate loss function for classification (a change without consequences on the performance) the structure of the loss function becomes very similar to the energy function of glasses. As a consequence, (i) we predict a sharp jamming transition that (ii) fall

into the universality class of ellipses. (iii) Close to the transition, the Hessian of the loss has a delta in zero and a gap followed by a bulk. We confirmed all those predictions empirically. Thus, we establish a sharp transition separating over and under parametrization.

Overparameterization & Double-Descent

Regression problems display a peak of the test error at the point where the train error becomes zero Advani et al. (2020). In Chapter 2, we look how in classification tasks, the test error is affected by the jamming transition. We observe the phenomenon that has later been coined **double-descent** Belkin et al. (2019). This phenomena is visible in Figure 6 as a blue dashed curve: The test error has a U-shape before jamming, (i) a peak at jamming, and (ii) a decrease above jamming.

We show that (i) is due to the divergence of the model as N become close to the jamming transition. We argue that the norm of the network scales as $\|f\| \sim 1/(N-N^*)$.

To think about (ii), the second descent, we need to introduce the infinite width limits in which a neural network can eventually converge: the Mean-Field limit or the Neural Tangent Kernel limit (NTK).

The Mean-Field Limit [14] Let's consider the infinite width limit of a FC initialized with MFP. In this limit, the weights must change significantly for the output to become $\mathcal{O}(1)$ and fit the data. This recently discovered limit is called "mean field", "rich" but also "feature learning regime" in the literature, because the neurons learn how to respond to different aspects of the input data – whereas, in the NTK limit, the neuron's response evolves infinitesimally. This regime has been studied in several works focusing mostly on one-hidden layer networks Mei et al. (2018); Rotskoff and Vanden-Eijnden (2018); Chizat and Bach (2018); Sirignano and Spiliopoulos (2020b); Mei et al. (2019); Nguyen (2019); Sirignano and Spiliopoulos (2020a), with recent development for deeper nets, see e.g. Nguyen and Pham (2020). In this setting the output function for a one hidden layer reads:

$$\tilde{f}(w,x) = \frac{1}{h}\sum_{i=1}^{h} W_i^{(2)} \phi\left(\frac{1}{\sqrt{d}} W_i^{(1)} \cdot x + B_i\right), \tag{9}$$

The law of large number can then be invoked to replace Eq.9 by an integral:

$$\tilde{f}(w,x) \to \int \mathrm{d}W^{(2)} \mathrm{d}W^{(1)} \mathrm{d}B\, \rho(W^{(2)}, W^{(1)}, B)\; W^{(2)} \phi\left(\frac{1}{\sqrt{d}} W^{(1)} \cdot x + B\right) \tag{10}$$

where ρ is the density of parameters. It is then straightforward to show that gradient flow leads to a dynamics on ρ of the usual type for conserved quantities: it is the divergence of a flux $\partial\rho/\partial t = -\nabla \cdot J$. Here the divergence is computed with respect to the set of weights associated

[14] Section adapted from Geiger et al. (2021)

with a single neuron, and the flux follows $J = \rho\Psi(W^{(2)}, W^{(1)}, B; \rho_t)$ where Ψ is some function that can be expressed in terms of the loss Mei et al. (2018). This formulation is equivalent to the hydrodynamics description of interacting particles in some external potential. When the number of particles (or neurons) is large, ρ is a more appropriate description than keeping track of all the particle positions (the weights in that case).

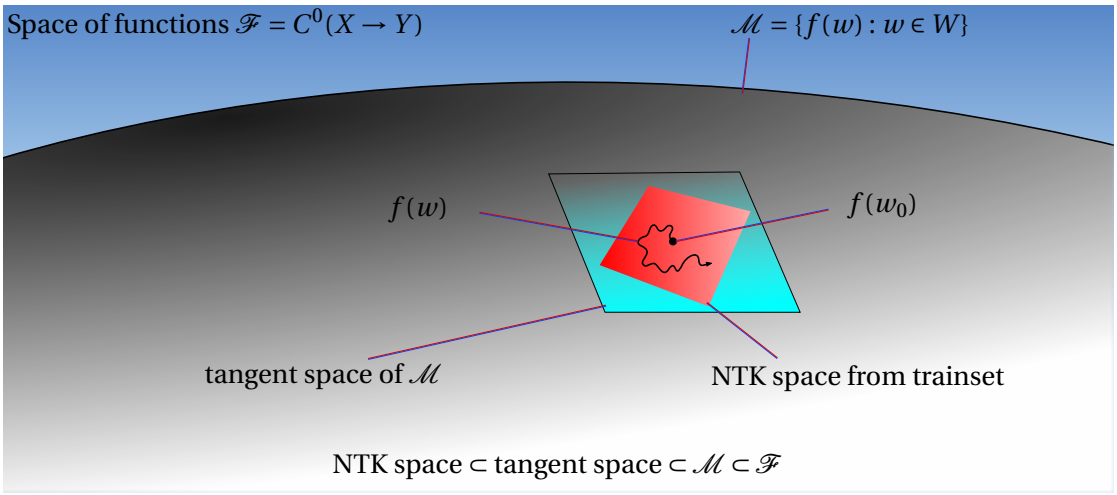


Figure 5 – In the space $\mathscr{F}$ of all continuous functions $X \to Y$, the neural network f can be seen as a manifold $\mathscr{M}$ of dimension N parametrized by its parameters W. The square represents the tangent space of $f(w_0)$, also of dimension N. During the training in the NTK limit, the trajectory is embedded into the space spanned by NTK evaluated on the trainset $span(\{\Theta(\cdot, x) : x, \_ \in \mathscr{R}\})$, of dimension $\min(N, P)$.

The Kernel Limit FCs with NTP initialization converge to Gaussian processes as $h \to \infty$ with a kernel that can be computed recursively Neal (1996); Williams (1997); Lee et al. (2018); de G. Matthews et al. (2018); Novak et al. (2019); Yang (2019a). Recently, FCs with NTP initialization has been shown to learn following another kernel called the Neural Tangent Kernel (NTK) Jacot et al. (2018); Du et al. (2019); Allen-Zhu et al. (2019); Lee et al. (2019); Arora et al. (2019); Park et al. (2019). It has been then recently generalized to any architectures Yang (2019b, 2020a,b).

In this limit, the weights move only by a little. Therefore the model can be linearized which implies that learning is a kernel method. Indeed, since the changes of the model, δf, are described by its gradients, $\delta f = \nabla f \, \delta w$, and since the dynamics moves the weights in the direction of the gradients, $\delta w \propto \nabla f$, the learning is entirely described by the NTK:

$$\Theta(w, x_1, x_2) = \nabla f(w, x_1) \cdot \nabla f(w, x_2) \tag{11}$$

Figure 5 shows an intuitive picture of the NTK limit. In the space of continuous functions from space X to space Y, the neural network can be seen as a manifold of dimension N

parametrized by the weights and biases of the network, shown as the gray area on the figure. When the parameters only change a little, the network remains in the tangent space of the manifold, around the initialization, shown as the cyan square lying on top of the gray manifold. Since the dynamics is controlled by the gradients evaluated at the point of the trainset, the evolution of the network is restricted in an even smaller subspace, depicted as the red square. Its dimension is limited by the number of points in the trainset, P.

It is shown in Jacot et al. (2018) that when $h \to \infty$, the NTK (therefore the NTK space) is independent of the initialization and it remains the same during the training.

The second descent We show that the second descent is the result of a noisy convergence toward the well-defined infinite model (NTK or Mean-Field). The noise comes from the initialization: For finite N, the neural network is affected by the random initialization in the form of random fluctuations to its outputs. These fluctuations jeopardize the performance. In Chapter 2, we give arguments for the scaling of the fluctuations and the scaling of the test error, that we confirm empirically.

There is a simple way to remove the fluctuation at finite N: **Ensemble averaging**. Ensemble averaging consists of training several networks with different random initialization and then averaging their outputs into a single predictor. Plain blue lines of Figure 6 show the test error of ensemble average predictors which is almost flat after the jamming transition. It suggests that it may be optimal to train many neural networks just above jamming, at distributable reasonable computational costs, instead of training a single big network at high computational cost, as confirmed in Lee et al. (2020). These results were confirmed for deeper architectures by Nakkiran et al. (2019) (like ResNet18 while we used shallow FCs and CNNs with up to 5 hidden layers).

A Phase Diagram for Learning Regimes

In Chapter 3, we present a phase diagram for deep learning that unifies jamming and different learning regimes. In the large width limit, the NTP initialization leads to the NTK limit while the MFP initialization leads to the Mean-Field limit. However the only difference between the two initializations stands for a different prefactor multiplying the weights of the last layer. This prefactor can be pushed outside of the model. Like Chizat et al. (2019), we used the predictor

$$F(w) = \alpha(f(w) - f(w_0)) \tag{12}$$

where $\alpha \in \mathbb{R}$ is a constant and f is initialized with NTP.

We give arguments to extend the predictions of Chizat et al. (2019) to deep FCs. That are, (i) the two infinite limits NTK and Mean-Field are mapped to two regimes at finite h: lazy regime and feature regime, respectively. (ii) The crossover between the two regimes at some α^* and (iii) it scales as $\alpha^* \sim 1/\sqrt{h}$.

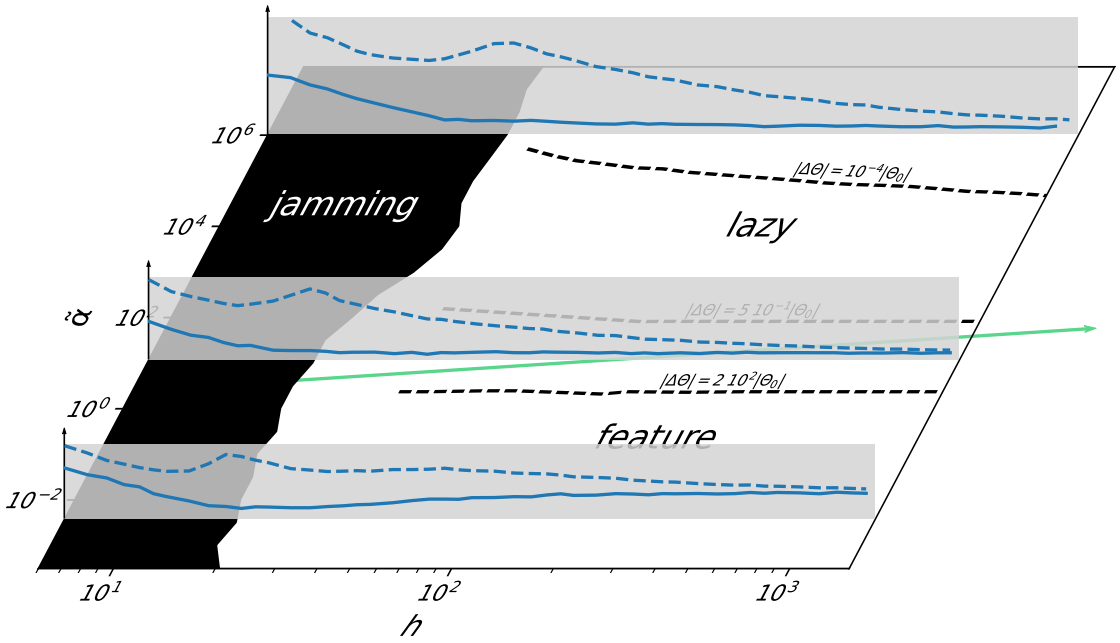


Figure 6 – Phase space of the landscape of a FC ($L = 2$) trained on MNIST binary (odd/even) with gradientflow. The under-parametrized phase, filled in black, corresponds to the region where the neural network is trapped in local minima of the loss. The dashed blue curves shows the test error for three different values of $\tilde{\alpha}$. The plain blue curves shows the ensemble average test error i.e. the test error of the average of 20 independent trainings. The double-descent phenomena is visible on the dashed blue curves. The black dashed lines are level sets of constant relative evolution of the NTK. The green arrow indicates the direction of the NTK limit.

We observe that the performance of deep neural networks depends on the regime and the architecture. Using gradientflow, FCs tends to perform better in the lazy regime while CNNs tends to perform better in the feature regime. This difference has been recently confirmed by Lee et al. (2020).

This phase diagram puts together the jamming transition (that shows a little dependency on α), the feature and lazy regimes and the NTK and Mean-Field limits. The phase diagram is conveniently shown as a $(h, \tilde{\alpha})$ plane where $\tilde{\alpha} = \sqrt{h}\alpha$, since then the cross over happens for $\tilde{\alpha}^* \sim 1$.

Figure 6 shows horizontal black dashed lines that corresponds to level set of the relative evolution of the neural tangent kernel. When the kernel evolution is small, the model is in the lazy regime. When it is large (compare to its value at initialization), it shows that the model is not linear anymore, indicating that features as been learned. In the $(h, \tilde{\alpha})$ plane, the Mean-field limit corresponds to the right border, pushed to infinity. While the NTK limit corresponds to a limit in diagonal ($\tilde{\alpha} \sim \sqrt{h}$) shown as a green arrow in Figure 6.

These results are presented in Chapter 3.

0.5.2 Part II: Data Symmetries

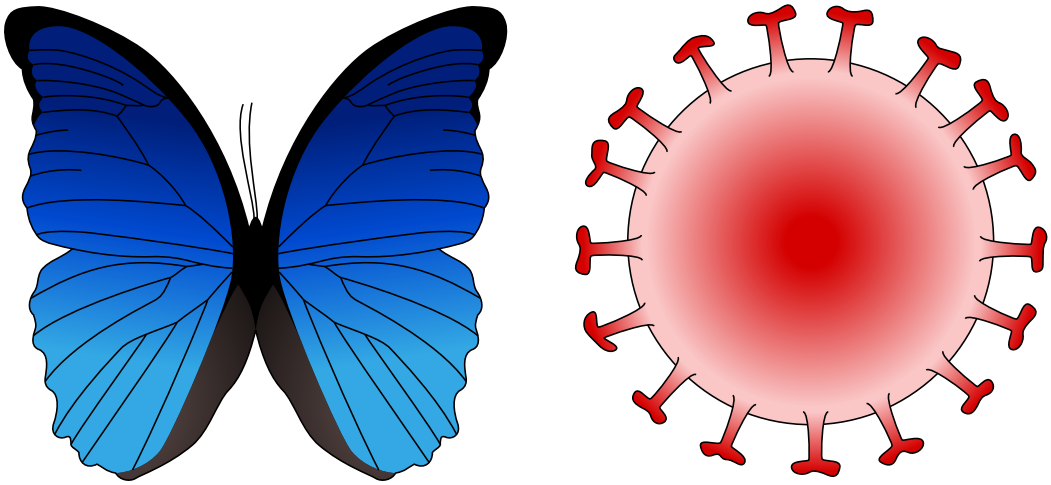


Figure 7 – (Left) A butterfly has an approximate mirror symmetry. (Right) The spikes of a virus are similar to each other and appears in many orientations.

Data contain symmetries. As exemplified in Figure 7, the two wings of a butterfly are symmetrical and the spikes at the surface of a virus are self similar. Natural images contains a lot of textures with self-similar patterns. Natural images contain in particular the translation symmetry: The absolute position of an object on an image does not affect its nature – the image contains a cat regardless of whether the cat is in the top left or bottom right corner. Are these symmetries exploited by deep neural networks?

If a model is stable to a set of transformations for which the target function is invariant, Bietti et al. (2021) showed, for kernel methods, improvements in sample complexity by a factor equal to the size of the set of transformations compared to the corresponding non-invariant model.

In fact, the convolution layers of CNNs are **equivariant** to translations. If the input z of a convolution F is translated by t, $T_t[z](s) = z(s-t)$, the output of the convolution $F[z]$ is equal to the translated output of the original input.

$$T_t[F[z]] = F[T_t[z]] \tag{13}$$

Making abstraction of the border effects and the discretization in partial pooling layers, [15] the output of a CNN is invariant to translation: $f(T[x]) = f(x)$. According to Bietti et al. (2021), since the set of translations of images of d pixels is at most of cardinality d, invariance to translations should effectively augment the trainset size by $\sim d$.

[15] For instance, when using a stride of 2, a translation by an odd number of pixels will have a non-trivial effect on the output.

However, a factor d is not enough to beat the curse of dimensionality, requiring $P \sim (1/\epsilon)^d$. It has been suggested that invariance toward diffeomorphisms is important Azulay and Weiss (2018); Zhang (2019); Dieleman et al. (2016).

Diffeomorphisms of 2D Images: relative stability matters

Figure 8 – (left) Original image (right) Image deformed by a random diffeomorphism sampled from a maximum entropy distribution.

Diffeomorphisms are local translations, see an example in Figure 8. The composition of two diffeomorphisms is a diffeomorphism. So, diffeomorphisms form a group much bigger than the translations. Again, according to Bietti et al. (2021), having a model invariant to the group of diffeomorphisms would effectively increase the size of the trainset much more than the translations do.

However, image classification tasks are only stable to small diffeomorphisms. If the image are too much scrambled, the information might be lost. To benefit from an effective trainset augmentation, the model should be quasi-invariant to small diffeomorphisms but not to large ones that might affect the label of the image. What is the best quantity that would measure it?

Non-parametric models have been created by hand to be stable to diffeomorphisms Bruna and Mallat (2013); Mallat (2016). Those models are stable in the sense that $|f(\tau x) - f(x)| < C\|x\| \sup_u |\nabla \tau(u)|$.

Modern architectures are often not constructed to be stable to diffeomorphisms if average pooling operations are not present, still they perform better than those hand made models. Could CNNs be able to learn to become stable to diffeomorphisms?

In Chapter 4, we test these ideas empirically. To do so, we sampled random diffeomorphisms using maximum entropy distribution. We maximize the entropy with a temperature T fixing

the expectation value of $|\nabla\tau(u)|^2$. By controlling the temperature we can probe different amplitude of diffeomorphisms, see an example in Figure 8.

We show that CNNs do learn to become stable to diffeomorphisms *relatively to* the stability to generic noise, unlike what was looked at in the past where only stability was considered. In our experiments using CIFAR10[16], we observed that stability to diffeomorphisms D_f is not well correlated to the test error. However, we observe a remarkable correlation between the test error and the ratio

$$R_f = \frac{D_f}{G_f} \tag{14}$$

where G_f is the stability toward additive Gaussian noise. It suggests that obtaining a small R_f is key to achieve good performance.

Equivariant Architectures

Some datasets like molecules, 3d shapes, point clouds, 3d graphs, proteins, etc. contain the additional symmetries of rotations and mirror.

Cohen and Welling (2016) proposed to extend the architectures of CNNs to equivariant to these additional symmetries. As a proof of concept, they implemented a CNNs equivariant to the dihedral group (in addition to the translations). The dihedral group contains 8 elements. It is generated by the 90 degree rotations and horizontal/vertical mirrors.

During his master thesis, the candidate independently developed a CNN equivariant to the dihedral group. It allowed him to start a collaboration with Taco Cohen, author of Cohen and Welling (2016) and later Tess Smidt, author of Thomas et al. (2018).

In Chapter 5, we present the work of these collaborations. We present three equivariant architectures for 3D rotations. (i) SphericalCNN allows to treat signals on the 3D sphere, it is the first neural network architecture equivariant with respect to a continuous group. (ii) 3D Steerable CNN, an architecture for 3D data equivariant to 3D translations and rotations. Similar to Thomas et al. (2018) that is defined on point clouds. (iii) Euclidean Neural Networks (`e3nn`), an extension of (ii) with the mirror symmetry and a more efficient and modular code. e3nn converged into an open source library now used by many research groups.

Finally, we give a mathematical formalism that generalize many equivariant architectures. It allows one to create new architecture and gives an uniform formalism to study them.

[16] CIFAR10 is a dataset of 60000 squared images of 32 pixels wide, labelled among 10 classes.

0.5.3 Part III: Miscellaneous

Training curve: asymptotic behavior of kernel regression

So far we looked at the loss landscape as a function of the number of parameters.

Generalization bounds for kernel ridge regression has been obtained in the past Scholkopf and Smola (2001); Cucker and Smale (2002); Vapnik (1999); Györfi et al. (2002). In Chapter 6, we derive how the test error change asymptotically with P for *ridgeless* kernel regression.

We use a Teacher-Student setup. As a dataset, we assume that the trainset has its points x on a regular lattice and the Teacher generate the y's by sampling from a Gaussian field with covariance equal to the kernel. The Student learns those data via kernel regression. For translations invariant kernels, we derive analytically β, the exponent of the training curve: $\epsilon \sim P^{-\beta} + o(P^{-\beta})$. We found good agreements with experiments on CIFAR10 and MNIST.

A new direction: landscape in reinforcement learning

At the end of the candidate thesis, we started to study another problem in which the landscape is non-convex: the reinforcement learning. We explored the landscape of the gain and optimal solutions of a simple problem of hypothesis testing in which we can understand the optimal strategies and study the glassiness of the gain landscape.

We consider the two arm bandit problem with an agent with finite memory. The problem is defined as following: There is two arms (left and right) that the agent can pull to gain rewards. The agent knows that, with equal probabilities, he is in one of the tow possible scenarios: (i) the probability to get a reward is $1/2+\mu/2$ for the left arm and $1/2-\mu/2$ for the right arm, or (ii) the contrary. At each play, the agent has only access to the last event (last reward and last arm pulled) and the current state of its memory. From that he has to chose which arm play and update its memory state.

We studied two types of memories for whose we found the optimal strategies and we showed their dependency to the time horizon. The article is available in Chapter 7.

Loss Landscape vs Parametrization Part I

1 A Jamming Transition in Deep Learning

The following paper is the preprint version of Geiger et al. (2019). The regulation of the doctoral school requires that candidate to detail its contributions to the papers.

Candidate contributions The candidate contributed to this paper by participating to the discussions, suggesting ideas and by doing the numerical experiments.

The jamming transition as a paradigm to understand the loss landscape of deep neural networks

Mario Geiger,[1,*] Stefano Spigler,[1,*] Stéphane d'Ascoli,[2,3] Levent Sagun,[2,1] Marco Baity-Jesi,[4] Giulio Biroli,[2,3] and Matthieu Wyart[1]

[1] *Institute of Physics, EPFL, CH-1015 Lausanne, Switzerland*
[2] *Institut de Physique Théorique, Université Paris-Saclay, CEA, CNRS, F-91191 Gif-sur-Yvette, France*
[3] *Laboratoire de Physique Statistique, École Normale Supérieure, PSL Research University, F-75005 Paris, France*
[4] *Department of Chemistry, Columbia University, 10027 New York, USA*

(Dated: June 18, 2019)

Deep learning has been immensely successful at a variety of tasks, ranging from classification to artificial intelligence. Learning corresponds to fitting training data, which is implemented by descending a very high-dimensional loss function. Understanding under which conditions neural networks do not get stuck in poor minima of the loss, and how the landscape of that loss evolves as depth is increased remains a challenge. Here we predict, and test empirically, an analogy between this landscape and the energy landscape of repulsive ellipses. We argue that in fully-connected deep networks a phase transition delimits the over- and under-parametrized regimes where fitting can or cannot be achieved. In the vicinity of this transition, properties of the curvature of the minima of the loss (the spectrum of the hessian) are critical. This transition shares direct similarities with the jamming transition by which particles form a disordered solid as the density is increased, which also occurs in certain classes of computational optimization and learning problems such as the perceptron. Our analysis gives a simple explanation as to why poor minima of the loss cannot be encountered in the overparametrized regime. Interestingly, we observe that the ability of fully-connected networks to fit random data is independent of their depth, an independence that appears to also hold for real data. We also study a quantity Δ which characterizes how well ($\Delta < 0$) or badly ($\Delta > 0$) a datum is learned. At the critical point it is power-law distributed on several decades, $P_+(\Delta) \sim \Delta^\theta$ for $\Delta > 0$ and $P_-(\Delta) \sim (-\Delta)^{-\gamma}$ for $\Delta < 0$, with exponents that depend on the choice of activation function. This observation suggests that near the transition the loss landscape has a hierarchical structure and that the learning dynamics is prone to avalanche-like dynamics, with abrupt changes in the set of patterns that are learned.

PACS numbers: 64.70.Pf,65.20.+w.77.22.-d

I. INTRODUCTION

Deep neural networks are now central tools for a variety of tasks including image classification [1, 2], speech recognition [3] and the development of artificial intelligence that can for example master the game of Go beyond human level [4, 5]. A neural network represents a (very high-dimensional) function f that depends on a large number of parameters N [2]. These parameters are learned so as to correctly classify P training data by minimizing some loss function $\mathcal{L}$, generally via stochastic gradient descent (a kind of noisy version of gradient descent). There is great flexibility in the network architecture, loss function and minimization protocol one can use. These features are ultimately selected to optimize the classification of previously unseen data, or *generalization*. Although the current progress in designing [6, 7] and training [8] networks that generalize well is undeniable, it remains mostly empirical. A general theory explaining and fostering this success is lacking, and central questions remain to be clarified. First, since the loss function is generally not convex, why doesn't the learning dynamics get stuck in poorly performing minima with high loss? In other words, under which conditions can one guarantee that training data are well fitted? Second, what are the benefits of deeper networks? On the one hand it is often argued, and proved in some cases, that the advantage of deep networks stems from their enhanced expressive power, i.e. their ability to build complex functions with a much smaller number of parameters than needed for shallow networks [9–13]. Indeed if deep networks are able to fit data with less parameters, then they are likely to generalize better. On the other hand, one can handcraft neural networks that fit even structure-less, random data with a rather small number of parameters $N \sim P$ [14–17]. These results for the static capacity of networks appear to be independent of depth [16, 17]. Yet, it is unclear whether such parsimonious solutions can be found dynamically in practice simply by descending the loss function, and whether depth can help finding them. More generally, how is the loss landscape affected by depth?

Complex physical systems with non-convex energy landscapes featuring an exponentially large number of local minima are called glassy [18]. Does the landscape of deep learning fall into a known class of glassy systems?

[*] These two authors contributed equally.

Along this line, an analogy between deep networks and mean-field glasses (p-spins) has been proposed [19], in which the learning dynamics is expected to get stuck in the highest minima of the loss, which are the most abundant. Yet, several numerical and rigorous works [20–23] (the latter focusing on shallow and very overparametrized networks) suggest a different landscape geometry where the loss function is characterized by a connected level set. Furthermore, studies of the Hessian of the loss function [24–26] and of the learning dynamics [27, 28] support that the landscape is characterized by an abundance of flat directions, even near its bottom, at odds with traditional glassy systems.

In the last decade several works have unveiled an analogy between the physical phenomenon of jamming [29, 30] and phase transitions taking place in certain classes of computational optimization and learning problems [31–33], in particular the perceptron [33, 34] — the simplest neural network performing linear classification. In this work we push this analogy further and show that the geometry of the training loss landscape and the training dynamics of fully connected deep neural networks is affected by a jamming transition similar to that of repulsive ellipses [30]. As illustrated in Fig. 2, jamming occurs in packings of particles interacting through a finite-range potential $\mathcal{U}$, when the particle density ϕ reaches some critical value ϕ_c. At that point, particles can no longer be accommodated without touching each other and the system becomes a solid with singular landscape properties, embodied for example in the spectrum of the Hessian of $\mathcal{U}$ [35, 36], that at the transition displays many (almost) flat directions. Particles of different shapes, such as spheres and ellipses, can lead to different jamming scenarios [37–40].

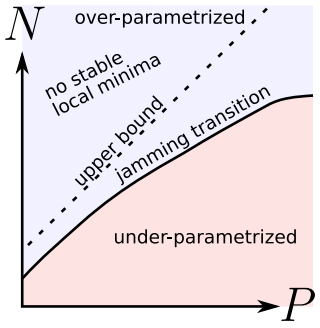


FIG. 1. N: degrees of freedom, P: training examples.

Here we show that for two commonly used loss functions (cross-entropy and quadratic hinge), fully-connected deep networks undergo a jamming transition too, below which all data are correctly fitted and above which they are not, both for real data (images) and random data [41]. In both cases the transition appears to be solely controlled by the number of parameters of the network N, independently of depth. For random data, the transition takes place as the quantity P/N increases toward some critical value P/N^*. For the hinge loss, using results from the jamming literature we argue that $P/N^* \geq C_0$ where C_0 is a constant that we can measure *a posteriori* once learning took place. To hold, this result requires the network output to remain sensitive to all its weights during training, as we observe empirically in the examples we study. This view supports that the dynamics cannot get stuck in poor minima in the over-parametrized regime where networks tend to operate, because there are not enough constraints to form minima in that regime. We also find that the jamming transition is sharp and the landscape appears to fall in the same universality class independently of depth (as long as at least one hidden layer is present). Differently from the (non-convex) perceptron, that was proven to lie in the same universality class as spherical particles [33, 34], we show that deep networks instead jam in a manner similar to ellipses. From our analysis we deduce the singular properties of the spectrum of the Hessian of the loss, which indeed must display many flat directions. We find empirically that other key quantities (the fraction of data which are almost correctly or almost incorrectly classified) display power-law behaviours on several decades, with new exponents. In glassy systems, such power-laws reveal properties that cannot be reached by studying the Hessian, in particular the fact that the dynamics occurs via broadly distributed avalanches [42–44], indicative of a hierarchical organization of the landscape [45]. This observation thus suggests that these properties also characterize deep networks near the transition. Note that in this work we focus on training and the ability of deep neural networks to fit a dataset. The implications and the relations with generalization between jamming and generalization are investigated in [46, 47].

II. ANALOGY BETWEEN JAMMING AND DEEP LEARNING

A. Jamming

Understanding the energy landscape — in particular the properties of the Hessian, referred to as vibrational properties in this context — in disordered systems of interacting particles is a long-standing and practically important problem [48]. It was realized that for purely repulsive, finite-range particles, such properties are singular near the jamming transition where the system becomes a solid [35, 36], allowing one to develop and test theories for the vibrations of glasses, that turn out to apply in a broader class of systems where the interactions do not necessarily have finite range [29]. Here we shall follow the same strategy for deep networks, where the role of the "interaction potential" is played by the choice of loss function. Finite-range interactions are mimicked by the *hinge loss*, for which we predict a sharp transition when

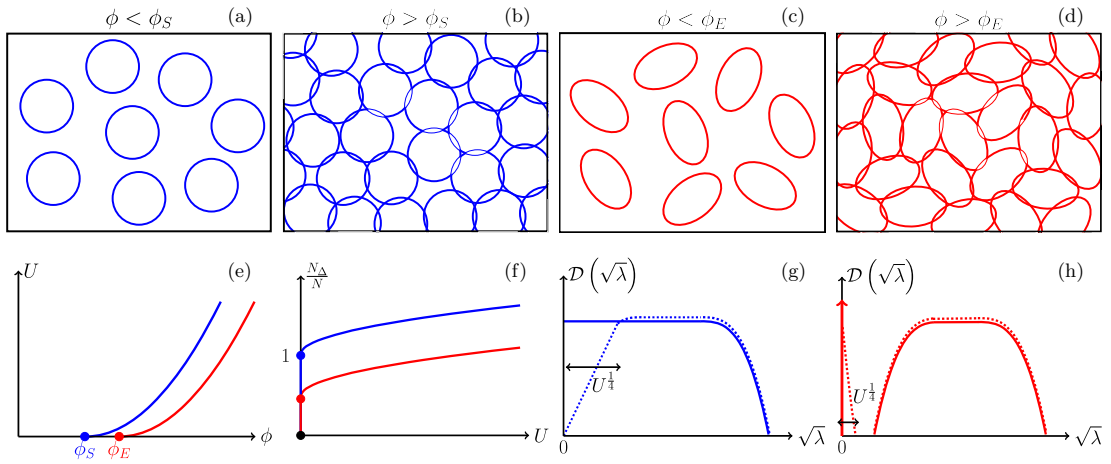


FIG. 2. Sketch of the jamming transition for repulsive spheres and ellipses. (a,b,c,d) Both systems transition from a fluid to a solid as the density passes some threshold, noted ϕ_S for spheres and ϕ_E for ellipses. (e) For denser packings, the potential energy $\mathcal{U}$ becomes finite. (f) The ratio N_Δ/N between the number of particles in contact N_Δ (corresponding to unsatisfied constraints) and the number of degrees of freedom N jumps discontinuously to a finite value, which is unity for spheres but smaller for ellipses. (g,h) This difference has dramatic consequence on the energy landscape, in particular on the spectrum of the Hessian. In both cases, the spectrum becomes non-zero at jamming, but it displays a delta function with finite weight for ellipses (indicating strictly flat directions), followed by a gap with no eigenvalues, followed by a continuous spectrum (h, full line). For spheres, there is no delta function nor gap (g, full line). As one enters the jammed phase, in both cases a characteristic scale $\lambda \sim \sqrt{\mathcal{U}}$ appears in the spectrum (g and h, dotted lines).

going from an overparametrized to an underparametrized regime. At the transition, the Hessian is singular and displays an abundance of low-energy modes. For other types of losses — such as for the commonly used cross-entropy loss defined below — the transition exists but its effects on the spectrum are expected to be less sharp (see discussion below).

We start by recalling some results on the jamming transition. We will first discuss the case of spherical particles, since it has been studied thoroughly and is easier to formalize. The behavior of elliptical particles will be discussed later on. Consider spheres of radius R at positions $\{\mathbf{r}_i\}$, corresponding to a total number of degrees of freedom $\tilde{N}$. We denote by $r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$ the distance between particles i and j, and define their overlap $\Delta_{ij} = 2R - r_{ij}$. Two particles are said to be in contact if $\Delta_{ij} > 0$, and N_Δ denotes the number of such contacts. We label by μ all the possible pairs of particles (ij) and by m the sets of contacts. We consider the following potential energy:

$$\mathcal{U} = \sum_{\mu \in m} \frac{1}{2} \Delta_\mu^2 . \tag{1}$$

We denote by N the effective number of degrees of freedom which affect the variables Δ_μ. It is in general smaller than $\tilde{N}$ because of (i) global translations or rotations of the system and (ii) "rattlers", i.e. particles which make no contact with the others, whose degrees of freedom are irrelevant as far as the solid phase is concerned.

As the jamming transition is approached from above (large density ϕ), $\mathcal{U} \to 0$ as sketched in Fig. 2, implying that $\Delta_\mu \to 0 \ \forall \mu \in m$. As argued in [49], for each $\mu \in m$ the constraint $\Delta_\mu = 0$ defines a manifold of dimension $N - 1$. Satisfying N_Δ such equations thus generically leads to a manifold of solutions of dimension $N - N_\Delta$. Imposing that solutions exist thus implies that, at jamming, one has

$$N_\Delta \leq N \, . \tag{2}$$

Note that this argument implicitly assumes that the N_Δ constraints are independent. In disordered systems this assumption is generally correct in practice, but it may break down if symmetries are present, which is the case e.g. in crystals where Eq. (2) can be violated.

An opposite bound can be obtained for spheres by considerations of stability, by imposing that in a stable minimum the Hessian must be positive definite [35]. The Hessian is an $N \times N$ matrix which can be written as [50]

$$\mathcal{H}_U = \sum_{\mu \in m} \nabla \Delta_\mu \otimes \nabla \Delta_\mu + \sum_{\mu \in m} \Delta_\mu \nabla \otimes \nabla \Delta_\mu \equiv \mathcal{H}_0 + \mathcal{H}_p \, , \tag{3}$$

where $\mathcal{H}_0$ and $\mathcal{H}_p$ correspond to the first and second sum, respectively. $\mathcal{H}_0$ is positive semi-definite, since it is the sum of N_Δ matrices of rank unity; thus $\mathrm{rk}(\mathcal{H}_0) \leq N_\Delta$, implying that the kernel of $\mathcal{H}_0$ is at least of dimension $N - N_\Delta$. On the other hand for *spheres* — but not for ellipses, and this will have major consequences

— $\mathcal{H}_p$ is negative definite, which simply stems from the fact that the second-order contribution of the displacement to the distance between two points is always positive - a straightforward application of the Pythagoras theorem. It is easy to show [35] that any non-zero vector $|u\rangle$ belonging to the kernel of $\mathcal{H}_0$ must satisfy $\langle u|\mathcal{H}_U|u\rangle = \langle u|\mathcal{H}_p|u\rangle < 0$ [51]. Thus stability requires that $\mathrm{rk}(\mathcal{H}_0) = N$, implying that $N_\Delta \geq N$. Together with Eq. (2) that leads to $N_\Delta = N$: as spheres jam the number of degrees of freedom and the number of constraints (stemming from contacts) are equal, as empirically observed [52]. This property is often called *isostaticity*: when it holds, mean-field arguments [53–55] predict that the density of vibrational modes $D(\sqrt{\lambda})$ displays a plateau up to vanishingly small λ, as observed numerically [35, 36] and sketched in Fig. 2G.

However, for *ellipses* [37] (and as we shall see, for deep networks), this argument breaks down because $\mathcal{H}_p$ is not negative definite. Whether such a matrix has positive eigenvalues or not plays a role of utmost importance in the selection of the universality class of the jamming transition, and it has major consequences on the singularity of the landscape, as it can be evinced from the spectrum of the Hessian matrix. Indeed, for ellipses stability and jamming can, and generically do, occur at:

$$N_\Delta/N < 1, \tag{4}$$

a situation that is referred to as *hypostatic*. The density of vibrational modes at jamming must then display a delta function in zero of magnitude $1 - N_\Delta/N$, corresponding to the kernel of $\mathcal{H}_0$ ($\mathcal{H}_p$ vanishes at jamming since $\Delta_\mu \to 0\ \forall \mu \in m$). Mean-field arguments applied to hypostatic materials [40, 56] predict that at larger λ, the spectrum presents a gap before becoming continuous again, as sketched in Fig. 2H. Away from jamming the effects of $\mathcal{H}_p$ kick in and broaden the delta function by an amount proportional to the typical value of the overlap $\Delta \sim \sqrt{\mathcal{U}}$, as sketched in Fig. 2H.

We now show that even in the hypostatic case, stability can be constraining. Let us denote by E_- the vector space spanned by the negative eigenvalues of $\mathcal{H}_p$, whose dimension very close to jamming is denoted N_-. Stability then imposes that the intersection of the kernel of $\mathcal{H}_0$ and E_- is zero, which is possible only if

$$N_\Delta \geq N_-. \tag{5}$$

Finally, another key structural property of the jamming transition is contained in the distribution $P_+(\Delta)$ of *positive overlaps*, sometimes referred to as *forces* (the force between two particles is Δ when $\Delta > 0$), and the distribution $P_-(\Delta)$ of *gaps* ($\Delta < 0$) between particles. It was shown that even if a packing of spherical particles is linearly stable, paths in the phase space that lower the energy are easily found unless both distributions are critical, with $P_+(\Delta) \sim \Delta^\theta$ and $P_-(\Delta) \sim (-\Delta)^{-\gamma}$, with $\gamma \geq (1-\theta)/2$ [42, 57], as numerically confirmed in [58, 59]. For a broad class of dynamics, this bound must be saturated [43], a scenario referred to as *marginal stability* which implies that the dynamics proceeds via power-law distributed events (called avalanches) in which the set of constraints change. Calculations in infinite dimensions [45, 60] showed that marginal stability is associated with a hierarchical organization of minima of the energy (a phenomenon referred to as *replica symmetry breaking* [61]), and exponents were found to follow $\gamma = 0.41269\ldots$ and $\theta = 0.42311\ldots$ which appear accurate even in finite dimensions [57, 62].

B. Deep Learning

Set-up: We consider a binary classification problem, with a set of P distinct training data denoted as $\{\mathbf{x}_\mu, y_\mu\}_{\mu=1,\ldots,P}$. The vector x_μ is the datum itself, which lives in dimension d (e.g. it could be an image), and $y_\mu = \pm 1$ is its label. A network architecture corresponds to a function $f(\mathbf{x};\mathbf{W})$, where $\mathbf{W}$ denotes the vector of parameters and $f(\mathbf{x};\mathbf{W})$ corresponds to the output of the network shown in Fig. 3. In this scheme, each neuron sums the activity of all the neurons in the previous layer with some weights, sketched as connections in Fig. 3 (each connection thus corresponds to one parameter $W^{(i)}_{\alpha,\beta}$). Next, a bias $B^{(i)}_\alpha$ is added to this sum (one additional parameter per neuron) to obtain the so-called pre-activation ($a^{(i)}_\alpha$ in the picture and in the equations). The neuron activity is then a non-linear function ρ of that pre-activation: in what follows we will deal mainly with $\rho(a) = a\theta(a)$ — the so-called rectified linear unit — but we will also present some results with $\rho(a) = \tanh(a)$. The computation is done iteratively from the first layer (close to the input $\mathbf{x}$) to the last one (the output $f(\mathbf{x};\mathbf{W})$):

$$f(\mathbf{x};\mathbf{W}) \equiv a^{(L+1)}, \tag{6}$$

$$a^{(i)}_\beta = \sum_\alpha W^{(i)}_{\alpha,\beta}\, \rho\left(a^{(i-1)}_\alpha\right) - B^{(i)}_\beta, \tag{7}$$

$$a^{(1)}_\beta = \sum_\alpha W^{(1)}_{\alpha,\beta}\, x_\alpha - B^{(1)}_\beta. \tag{8}$$

In our notation the vector $\mathbf{W}$ contains all the parameters, including the biases. $\mathbf{W}$ is learned by minimizing a cost function, which can generically be written $\mathcal{L}(\mathbf{W}) = \frac{1}{P}\sum_{\mu=1}^{P} \ell\left(y_\mu, f(\mathbf{x}_\mu;\mathbf{W})\right)$. A widely chosen kind of loss is the cross entropy, $\ell(y,f) = \log\left(1 + e^{-yf}\right)$. Another common choice is the hinge loss, defined as $\ell(y,f) = \frac{1}{2}\Delta(y,f)^2\theta(\Delta(y,f)) = \frac{1}{2}\max(0,\Delta(y,f))^2$, where we have introduced the data overlap

$$\Delta(y,f) \equiv \epsilon - yf, \tag{9}$$

with $\epsilon > 0$ being a constant. In what follows we choose $\epsilon = 1/2$ without loss of generality [63]. These loss functions take such a simple form only for a binary classification task, with labels $y = \pm 1$, where $\ell(y,f) \equiv \ell(y \cdot f)$; the two loss functions are compared in Fig. 4. In the

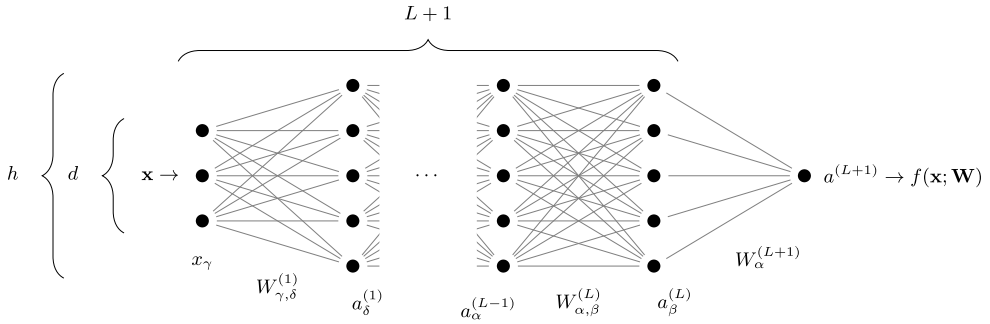


FIG. 3. Architecture of a fully-connected network with L hidden layers of constant size h. Points indicate neurons, connections between them are characterized by a weight. Biases are not represented here.

Particles	vs	Neural networks
positions of particles (N degrees of freedom)	$\leftrightarrow$	parameters of the network (N degrees of freedom)
pairs of particles (ij)	$\leftrightarrow$	patterns μ
energy $\mathcal{U}$	$\leftrightarrow$	loss $\mathcal{L}$
long range interaction	$\leftrightarrow$	(for instance) cross-entropy
finite range interaction	$\leftrightarrow$	hinge loss
particle density ϕ	$\leftrightarrow$	number of data divided by the number of parameters P/N
separate two particles	$\leftrightarrow$	fit a datum
force distribution	$\leftrightarrow$	density of unsatisfied patterns $P_+(\Delta)$
gap distribution	$\leftrightarrow$	density of satisfied patterns $P_-(\Delta)$

TABLE I. Correspondence between the jargon of particle systems and that of neural networks.

hinge loss, the condition $\Delta_\mu = \Delta\left(y_\mu, f(\mathbf{x}_\mu;\mathbf{W})\right) < 0$ ensures that the datum μ is *satisfied* — that is, correctly classified by a margin ϵ. The data which do not respect this margin will be referred to as *unsatisfied* (not to be confused with *misclassified* data, for which $y_\mu f(\mathbf{x}_\mu) < 0$) — the number of such data will be denoted as N_Δ. With this definition, $\mathcal{L}$ is formally identical to $\mathcal{U}$ in Eq. (1) as already noted for the perceptron [55], and it can be written as $\mathcal{L}(\mathbf{W}) = \frac{1}{P}\sum_{\mu\in m}\frac{1}{2}\Delta_\mu^2$, where m is the set of unsatisfied patterns. The correspondence between interacting particles and neural networks is summarized in Table I.

Performance of the hinge loss and its extension to multi-class problems: This section can be skipped at a first reading. We tested in the context of image classification that the hinge loss performs as well as the cross entropy on a state-of-the-art architecture [64]: we ran the implementation [65] for CIFAR-10 and we retrained it by replacing the cross entropy by the hinge loss. To compare the two losses in a standard setting we adapted the hinge loss for multiple classes, although in what follows we only study binary classification. To predict the label of an input $\mathbf{x}_\mu$ among 10 possible labels $c = 0, \ldots, 9$, the network's last layer returns as output a list of 10 values $f_{\mu,c}$: each $f_{\mu,c}$ can be interpreted as the probability that c is the predicted label. Let $t_{\mu,c}$ be the true target labels: for each μ, $t_{\mu,c}$ is equal to 1 if c is equal to the label of $\mathbf{x}_\mu$ and -1 otherwise. Multiclass hinge loss can then be written as

$$\mathcal{L} = \frac{1}{10P}\sum_{\mu,c}(\epsilon - t_{\mu,c}f_{\mu,c})^2\theta(\epsilon - t_{\mu,c}f_{\mu,c}). \qquad (10)$$

We obtained an error of 3.72% by running their original code (they report on github an error of 3.68%) and 3.61%, 3.65%, 3.82% in three runs with the hinge loss.

Effective number of parameters: Following the argument developed after Eq. (1), we expect that at the transition point where the loss becomes non-zero, Eq. (2) will hold true and $N_\Delta \leq N$. (Related arguments were recently made for a quadratic loss [23]. In this case, we expect that the landscape will be related to that of floppy spring networks, whose spectra were predicted in [56]). Just as is the case for the jamming of particles, here we must pay attention to the effective number of degrees of freedom that do affect the output, $N_{\text{eff}}(\mathbf{W})$. in the space of functions going from the neighborhoods of the training set to real numbers, we consider the manifold of functions $f(\mathbf{x};\mathbf{W})$ obtained by varying $\mathbf{W}$. We denote by $N_{\text{eff}}(\mathbf{W})$ the dimension of the tangent space of this manifold at $\mathbf{W}$. We discuss in Appendix A how $N_{\text{eff}}(\mathbf{W})$ can be measured. In general we have $N_{\text{eff}}(\mathbf{W}) \leq N$. Sev-

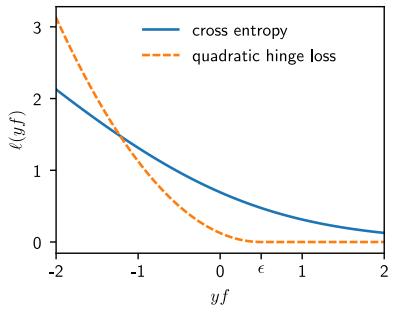


FIG. 4. Cross entropy and hinge loss functions. If the network classifies two classes with labels $y = \pm 1$ then the loss can be written as $\ell(y, f) = \ell(yf)$. The plot shows the two cases studied in this work, namely the cross-entropy and the hinge loss; for the latter, a parameter $\epsilon = \frac{1}{2}$ has been used.

eral reasons can make $N_{\text{eff}}(\mathbf{W})$ strictly smaller than N, including:

- The signal does not propagate in the network, i.e. $f(\mathbf{x}; \mathbf{W}) = C_1$ where C_1 is a constant for all $\mathbf{x}$ in the neighborhood of the training points $\mathbf{x}_\mu$. In that case, the manifold is of dimension unity and $N_{\text{eff}}(\mathbf{W}) = 1$. This situation will occur for a poor initialization of the weights as discussed in [66], or for example if all biases are too negative on the neurons of one layer for ReLU activation function. It can also occur if the data $\mathbf{x}_\mu$ are chosen in an adversarial manner for a given choice of initial weights. For example, one can choose input patterns so as to not activate the first layer of neurons (which is possible if the number of such neurons is not too large). Poor transmission will be enhanced (and adversarial choices of data will be made simpler) if the architecture presents some bottlenecks. In the situation where $N_{\text{eff}}(\mathbf{W}) = 1$, it is very simple to obtain local minima of the loss at finite loss values, even when the model has many parameters.

- The activation function is linear, then the output function is an affine function of the input, leading to $N_{\text{eff}} \leq d + 1$. Dimension-dependent bounds will also exist if the activation function is polynomial (because the output function then is also restricted to be polynomial).

- Symmetries are present in the network, e.g. the scale symmetry in ReLU networks. It will reduce one degrees of freedom per node.

- Some neurons are never active e.g. in the ReLU case, their associated weights do not contribute to N_{eff}.

Thus there are $N - N_{\text{eff}}$ directions in parameters space that do not affect the function. These directions will lead to zero modes in the Hessian at any minima of the loss. In what follows we consider stability with respect to the N_{eff} directions orthogonal to those, which thus affect the output function. Our results on the impossibility to get stuck in bad minima are expressed in terms of N_{eff}. However, as reported in Appendix A, we find empirically that for a proper initialization of the weights and rectangular fully connected networks, $N_{\text{eff}} \approx N$ (the difference is small and equal to the number of hidden neurons, and only results from the symmetry associated with each ReLU neuron). Henceforth to simplify notations we will use the symbol N to represent the number of effective parameters. In the following sections, the Hessians are computed with respect to all the N parameters.

Constraints on the stability of minima: Let us suppose (and justify later) that for a fixed number of data P, if N is sufficiently large then gradient descent with proper weights initialization leads to $\mathcal{L} = 0$, whereas if N is very small after training $\mathcal{L} > 0$. Consider that N is increased from a small value. At some value N^* the loss obtained after training will approach zero [67], i.e. $\lim_{N \to N^*} \mathcal{L} = 0$. In analogy with the behavior of packings of particles, we refer to this point as the jamming transition. At the transition the stability constraint developed in Eq. (5) above also applies if the derivative of $f(\mathbf{x}; \mathbf{W})$ is continuous, which holds true if the non-linear function ρ is smooth. Thus we have:

$$P \geq N_\Delta \geq N_- \,, \tag{11}$$

since $P \geq N_\Delta$ (the number of unsatisfied patterns is obviously smaller than the total number of patterns).

We shall assume that the fraction $N_-/N \equiv C_0$ of negative eigenvalues of $\mathcal{H}_p$ does not vanish in the large N limit. In Appendix B we provide an argument supporting this result in the case of a specific non-linear function (ReLU) and random data, that yields $C_0 = 1/2$ independently of depth. It implies that unlike for spheres, but just like ellipses, $\mathcal{H}_p$ is not negative definite: we are therefore in the hypostatic scenario where one expects $N_\Delta < N^*$ at jamming, a point at which the spectrum must display a fraction of flat directions, as well as stiff ones, as described in Fig. 2H.

Moreover from this assumption and Eq.11, we obtain that stability cannot be obtained for $N \geq P/C_0$. For larger N, the dynamics cannot get stuck in a bad minimum, because in this over-parametrized regime there are not enough constraints to form them. It implies for the jamming transition that:

$$N^* \leq P/C_0. \tag{12}$$

Notice that this bound is expected to be valid for any monotonic cost function, as for instance the cross entropy (the Hessian can always be decomposed as in Eq. (3)). However, the spectrum of the Hessian would be different [68].

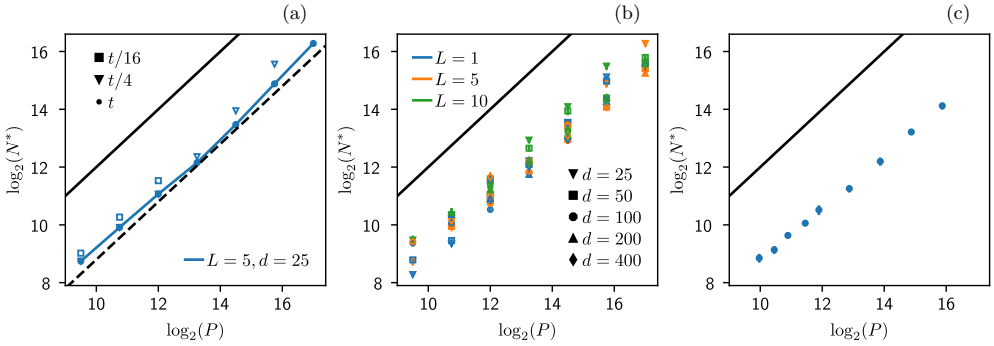


FIG. 5. Jamming transition with random data. (a) N^* vs number of data P for different learning times as indicated in legend, where $t = 10^6$ steps and a cross-entropy loss function is used. The curves at small times (orange and green) are shown as diverging to indicate the absence of the transition. The dotted black line toward which the dynamics appear to converge has slope 1, supporting $N^* \sim P$ at long times. Here $L = 5$ and $d = 25$. (b) N^* vs number of data P after $t = 10^6$ for various depths L and input dimensions d as indicated in legend, using the same loss function. The transition shows little dependency on L and d. (c) Same plot as (b) for a network with hinge loss, with $d = h$ and $t = 2 \cdot 10^6$. In the three plots (a,b,c), the black line indicates the theoretical upper bound: $P/N^* = 1/2 - N_c/N$ derived for the hinge loss.

Smooth *vs* non-smooth output function: In our numerical study below, we consider the most common choice for the non-linear function ρ, namely the rectified linear unit (ReLU): $\rho(a) = a\,\Theta(a) = \max(0, a)$. In that case, as stated above we expect for random data the spectrum of $\mathcal{H}_p$ to be symmetric (a fact that appears to also hold true for the image dataset we use, see below), thus $N_-/N = 1/2$. Yet, with the ReLU, $f(\mathbf{x}; \mathbf{W})$ is not continuous and presents cusps, so that the Hessian needs not be positive definite for stability and Eq. (11) needs to be modified. Introducing the number of directions N_c presenting cusps, stability implies $N_\Delta \geq N_- - N_c$ leading to $C_0 N^* \leq P + N_c$. Empirically we find that $N_c/N^* \in [0.21, 0.25]$ both for random data and images as reported in Appendix C, implying that:

$$N^* \leq 4P. \tag{13}$$

For comparison, below we also present results for networks with tanh activation functions. In that case the landscape is smooth and the system ends up in minima without negative eigenvalues. For such networks the spectrum of $\mathcal{H}_p$ is not exactly symmetric, and we observe $C_0 = N_-/N \approx 0.43$.

Main results: Overall, our analysis supports that

1. In the case of hinge loss there is a sharp transition for $N^* \leq P/C_0$ ($N^* < 4P$ with the ReLU), below which the loss converges to some non-zero value (under-parametrized phase) and above which it becomes null (over-parametrized phase).

2. At that point the fraction N_Δ/N of unsatisfied constraints per degree of freedom jumps to a finite value, see Fig. 6 (a,b).

3. Unlike for spheres or the perceptron, isostaticity $N_\Delta/N = 1$ cannot be guaranteed. Instead one expects generically $N_\Delta/N < 1$ as for ellipses.

4. We are thus in the hypostatic universality class, where the scaling properties of the spectrum of the Hessian near jamming are prescribed in Fig.2.

In the next sections, we confirm these predictions in numerical experiments and observe the generalization properties at and beyond the transition point.

III. FOR RANDOM DATA THE TRANSITION OCCURS FOR $N \sim P$

We begin the numerical study of the transition between the overparametrized and underparametrized regime in the case of random data, taken to lie on the d-dimensional hyper-sphere of radius $\sqrt{d}$, $\mathbf{x}_\mu \in \mathcal{S}^d$ with random label $y_\mu = \pm 1$. The source code used to generate the simulations described in this section and the following ones is available at `https://github.com/mariogeiger/nn_jamming`. We proceed as follows: we build a network with a number of weights N large enough for it to be able to fit the whole dataset without errors. Next, we reduce the number of weights by decreasing the width h while keeping the depth L fixed, until the network cannot correctly classify all the data anymore within the chosen learning time. We denote this transition point N^*.

We have noticed (data not shown) that the precise location of the transition point P/N^* has a mild dependence on the dynamics (ADAM versus regular SGD, choice of batch size, learning rate schedule, etc...): the same holds true for the jamming of repulsive particles, where the choice of the dynamics affects the precise value of the critical density ϕ_c, but not the critical behaviour close to this point.

Cross-entropy loss: We first consider the cross-entropy loss — the results are qualitatively similar to those with the hinge loss. As initial condition for the dynamics we use the default initialization of `pytorch` [69]. The system then evolves according to a stochastic gradient descent (SGD) with a learning rate of 10^{-2} for $5 \cdot 10^5$ steps and 10^{-3} for $5 \cdot 10^5$ steps; the batch size is set to $\min(P/2, 1024)$; only in this case, with the cross-entropy loss, batch normalization is also used. In Fig. 5 (a) we show how N^* depends on the total learning time: the larger is the learning time the more the asymptotic relationship N^* vs P is consistent with an asymptotic linear behaviour. Note that for large P and small times, errors are always present and the transition cannot be found.

In Fig. 5 (b) we show N^* versus the number of data P after $t = 10^6$ steps for several depths L and input dimensions d (we checked that $t = 10^6$ is enough to get convergence to the conjectured asymptotic linear behaviour for all depths investigated). It is noteworthy that (i) the points always lie below the theoretical upper bound $P/N^* = 1/2 - N_c/N$, and (ii) the transition does not appear to depend on L and d. Surprisingly, this result indicates that in the present setup the ability of fully connected networks to fit random data is independent of the depth. As we shall see, we observe the same independence on depth for the image data studied below.

Hinge loss: In order to test the dependence of our results on the specific choice of the loss function, we performed the same experiment using the hinge loss. In this case we used an orthogonal initialization [70], no batch normalization and $t = 2 \cdot 10^6$ steps of ADAM [71] with batch size $= P$ and a learning rate starting at 10^{-4}, progressively divided by 10 every 250k steps. The location of the transition is shown in Fig. 5 (c): results are very similar to that of the cross-entropy loss.

Hinge *v.s.* cross-entropy loss from a conceptual perspective: As shown above, both losses appears to lead to similar performances. As shown in this section, both of them also displays a transition where all data are fitted. Yet, the nature of this transition is harder to investigate for the cross-entropy. Indeed in that case the total loss is never zero, except if the output and therefore the weights diverge. Thus in the overparametrized phase, the learning dynamic never settles, and the weights slowly drift to infinity. In practice, users stop learning at finite times (which is not needed for the hinge loss where the dynamics really stops in the overparametrized regime when the loss vanishes). Working at finite time however blurs true critical behavior near jamming, as discussed in [47].

IV. THE TRANSITION IS HYPOSTATIC

From the analysis of Section II, the number of constraints per parameter N_Δ/N is expected to jump discontinuously at the transition. To test this prediction we consider several architectures, both with $N \approx 8000$ and $d = h$ but with different depths $L = 2$, $L = 3$ and $L = 5$. The vicinity of the transition is studied by varying P around the transition value. We used the hinge loss with the same gradient descent dynamics as described above, for a duration of 10^7 steps. Fig. 6 (a) reports the ratio N_Δ/N as a function of the ratio P/N and of the learning time, as detailed in caption. It is clear that in the range where N_Δ/N has reached a stationary value (i.e. for $P/N < 2.8$ and $P/N > 2.9$), a jump has occurred from 0 to $N_\Delta/N \approx 0.75$, a result consistent with the bound of Eq. (5) implying $N_\Delta/N \geq (N_- - N_c)/N \gtrapprox 0.25$. For $P/N \in [2.8, 2.9]$, the dynamics has not yet converged and the data are somewhat scattered. This observation is presumably the signature of the usual slowing down that occurs near critical points.

Fig. 6 (b) shows the same quantity N_Δ/N, now plotted as a function of the loss $\mathcal{L}$. Strikingly, all the scatter is gone, and one observes a clear discontinuous behaviour for $\mathcal{L} \to 0$. Interestingly, this state of affairs is very similar to the jamming transition of particles, for which the noise in the data due to finite size effects is quite strong when quantities are expressed in terms of the density ϕ (analogous to P/N) but very small when quantities are expressed in terms of potential energy $\mathcal{U}$ (analogous to $\mathcal{L}$) [52].

For the sake of completeness we also show the number of misclassified data as a function of the loss in Fig. 6 (c). The number of misclassified data increases monotonically — and initially very slowly — with the loss. Indeed, close to the jamming threshold in the underparametrized phase, if $0 < \Delta_\mu < \epsilon$ the pattern μ is well classified but the corresponding gap Δ_μ is positive: unsatisfied constraints do not lead to misclassification right away.

In Fig.6 (d) we show that N_Δ/N vs the loss $\mathcal{L}$ exhibits a sharp transition also for networks with tanh activation functions.

V. SPECTRUM OF THE HESSIAN OF THE LOSS NEAR JAMMING

The Hessian is a key feature of landscapes, as it characterizes its curvature, and it is also a central aspect of the theoretical description above. In this section we systematically analyze the spectra of $\mathcal{H}$, $\mathcal{H}_0$ and $\mathcal{H}_p$. To test the predictions on the singularity of the Hessian matrix, we need to focus on the underparametrized data points near the transition. These points are contained in the black rectangle on the left side of Fig. 6 (b). The networks that we use are relatively small, but, for reference, it would be possible to compute the spectrum of the Hessian also for large networks, as discussed e.g. in [72, 73]. The setting is as above: the network uses the hinge loss and is trained with ADAM with full batches (batch size $= P$), orthogonal initialization and no batch normalization.

Relu networks: At the end of each run, we compute the hessian $\mathcal{H}$ of the loss $\mathcal{L}$, as well as the two terms $\mathcal{H}_0$ and $\mathcal{H}_p$ contributing to it, as defined in Eq. (3). Fig. 7

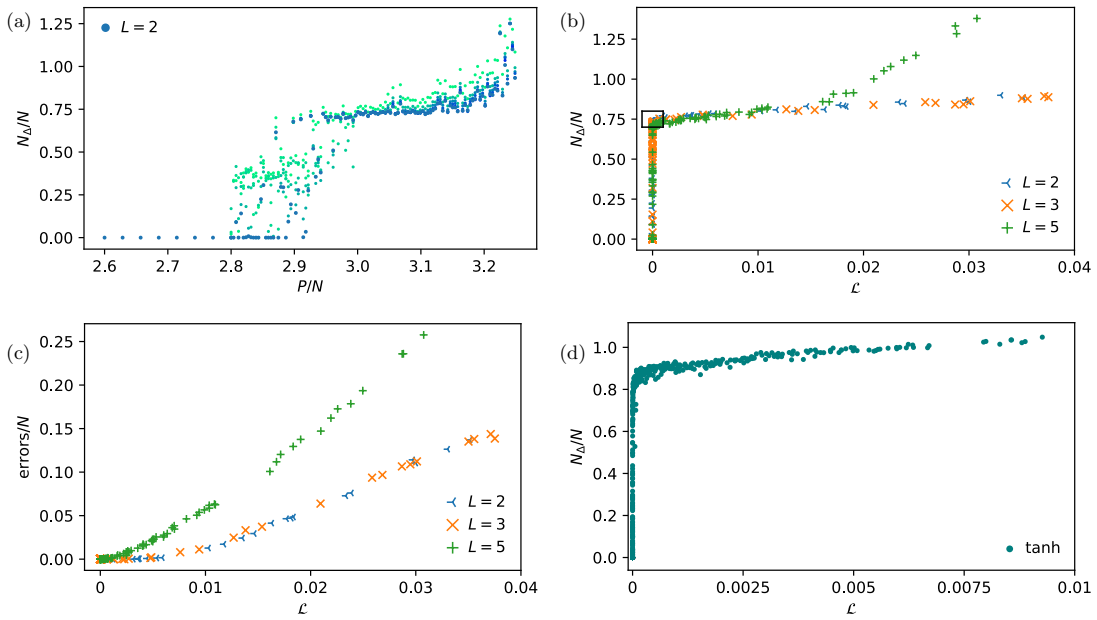


FIG. 6. Behaviour near the transition for random data. (a) Number of unsatisfied constraints N_Δ per parameter N as a function of P/N. Collections of vertical points correspond to the same run, but with different learning times from green (short time, starting at $3 \cdot 10^5$ steps) to blue (10^7 steps). The data support a discontinuous jump in this quantity at some $P/N \in [2.8, 2.9]$ at asymptotically long times. Indeed, outside that range the learning dynamics appear to have converged to zero for $r < 2.8$, and to some value > 0.7 for $P/N > 2.9$. In the interval $P/N \in [2.8, 2.9]$, data are still evolving in time. (b) N_Δ/N vs $\mathcal{L}$ follows a curve with almost no scatter for all $L = 2, 3, 5$. This is similar to the jamming transition where finite size noise is eliminated when quantities are plotted against the potential energy, rather than the packing fraction [30]. The black rectangle on the left side of the plot (small loss $\mathcal{L}$ and finite ratio N_Δ/N) marks the points in the underparametrized phase that are close to the transition. (c) Relationship between the number of misclassified data (data points with negative $y_\mu f(\mathbf{x}_\mu)$) and $\mathcal{L}$, displaying a smooth behavior. (d) N_Δ/N vs $\mathcal{L}$ for a network with $L = 3$, but with tanh activation functions rather than ReLUs.

(a) shows the positive part of the spectrum of $\mathcal{H}_p$ for different values of the loss, illustrating that the dependence on the latter is very significant. In Fig. 7 (b) we confirm that the spectrum of $\mathcal{H}_p$ collapses when the eigenvalues are re-scaled by $\mathcal{L}^{1/2}$, as expected from Section II. The key observation is that these spectra are symmetric, as argued in Appendix B. We also don't observe any accumulation of eigenvalues at $\lambda = 0$, except for the trivial zero modes stemming from the scaling symmetry of ReLU neurons (whose number is the total number of hidden neurons, much smaller than the number of weights). Fig. 7 (c) shows the spectrum of $\mathcal{H}_0$ at the end of training for runs close to the jamming transition. As expected it is semi-positive definite, with a delta peak at $\lambda = 0$ corresponding to $N - N_\Delta$ modes. It is followed by a gap and a continuous spectrum, as predicted near the jamming transition of particles if $N_\Delta < N$ [56] (which occurs for elliptic particles [40]). As the loss increases, N_Δ increases and the gap is reduced. Finally in Fig. 7 (d), the spectrum of $\mathcal{H}$ is shown. Interestingly the spectrum of the Hessian is not positive definite, but present some unstable modes. This phenomenon stems from our choice of ReLU activation function, which leads to cusps in the landscape as quantified in the C. Such cusps can stabilize directions that would be unstable according to the Hessian.

Tanh networks: On the contrary, networks with tanh activation functions exhibit a smooth landscape, and in principle the loss is able to reach minima without any negative eigenvalues, since there are no cusps that could possibly stabilize them. Indeed, when minimizing a tanh-network with $P = 11000$ random patterns and $N = 2232$ parameters, we observe that after 10 million ADAM steps there remain only 11 negative eigenvalues (between $-5 \cdot 10^{-5}$ and $-2 \cdot 10^{-8}$), and after 100 million ADAM steps only 6 were left (between $-4 \cdot 10^{-6}$ and $-2 \cdot 10^{-8}$). For comparison, in ReLU networks the number of negative eigenvalues at the end of training is about $10\%N$.

In Fig. 7 (e-g) we show the spectrum of the matrices $\mathcal{H}_0, \mathcal{H}_p$ and $\mathcal{H} = \mathcal{H}_0 + \mathcal{H}_p$, for a tanh-network at jam-

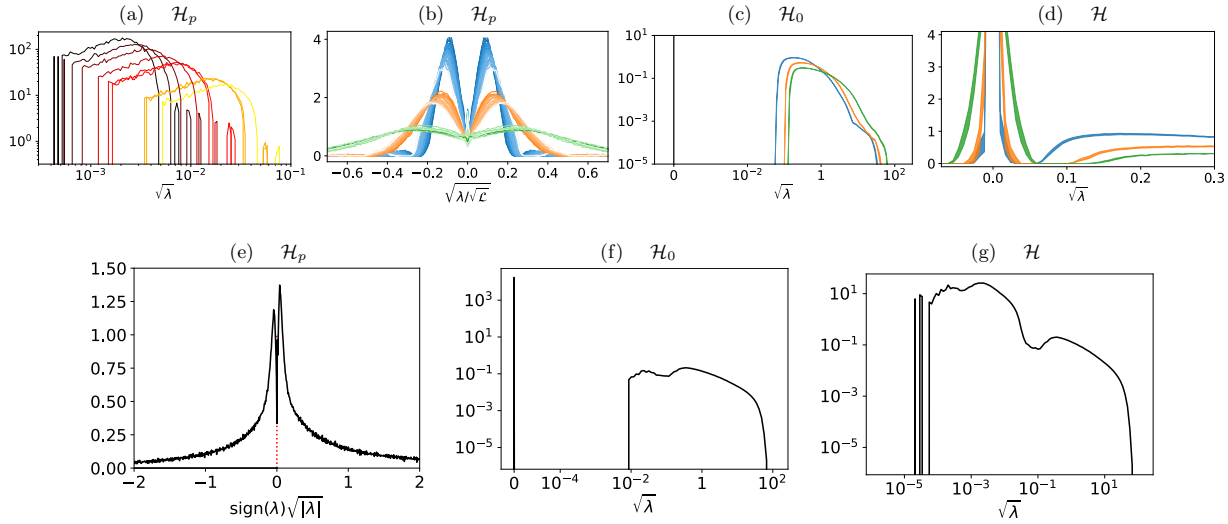


FIG. 7. The data shown in this figure concern the underparametrized points close to the transition for random data, which in Fig. 6 (b) are enclosed in a black rectangle. (a) Positive part of the spectrum of $\mathcal{H}_p$ for ten distinct runs in the underparametrized phase close to the transition. The associated loss value grows from black (low) to yellow (high). (b) These spectra collapse when plotted in terms of $\lambda/\sqrt{\mathcal{L}}$ as expected. Lighter colors correspond to higher losses. Note that they appear symmetric, in agreement with our hypothesis estimating the number of negative modes (an argument that explains this fact can be found in Appendix B). Colors are as in (d): $L = 2$ (blue), $L = 3$ (red) and $L = 5$ (green). (c) The spectrum of $\mathcal{H}_0$ contains a delta function in zero of weight $N - N_\Delta$, followed by a gap, followed by a continuous spectrum, as expected for hypostatic systems. (d) The spectrum of the total Hessian $\mathcal{H}$ has a similar shape, excepted that the delta function is blurred. Note that $\mathcal{H}$ has negative eigenvalues. These directions may in fact be stabilized by the N_c cusps of the linear rectifier, or alternatively may indicate that the learning dynamics did not converge to a local minimum yet. The thickness of each line correspond to the standard deviation. (e-g) Spectrum of the matrices $\mathcal{H}_p, \mathcal{H}_0$ and $\mathcal{H} = \mathcal{H}_0 + \mathcal{H}_p$ for tanh networks, respectively. Notice that the spectrum of $\mathcal{H}_p$ is no longer symmetric, compared to the ReLU case: the faction of negative eigenvalues is $C_0 \approx 0.43$.

ming. The matrix $\mathcal{H}_0$ is qualitatively similar to what was observed in ReLU-networks: it presents a delta peak in 0 and a gapped bulk of positive eigenvalues. The matrix $\mathcal{H}_p$ appears quite different, since it is no longer symmetric: the number N_- of negative eigenvalues is approximately $0.43N$ — therefore $C_0 = 0.43$ instead of $C_0 = 0.5$. The total Hessian $\mathcal{H}$ is not gapped in the present case, even though it displays two peaks. In order to clearly have a gap we would have to sample points closer to jamming (with a smaller loss, since $\mathcal{H}_p$ is proportional to $\sqrt{\mathcal{L}}$ close to jamming).

Overall, as we move from the under-parametrized phase to the over-parametrized one the situation is as follows:

1. N below N^*: There are many constraints with respect to the number of variable N, $\mathcal{H}_0$ is almost full rank and can easily compensate the negative eigenvalues of $\mathcal{H}_p$. The spectrum of $\mathcal{H}_p$ is symmetric.

2. N approaching N^* from below: The rank of $\mathcal{H}_0$ decreases but it does not go below $C_0 N$ since it has to compensate the vanishingly small negative eigenvalues of $\mathcal{H}_p$.

3. As N is large enough, the dynamics finds a global minimum at $\mathcal{L} = 0$ and $\mathcal{H}_p$ vanishes.

VI. DISTRIBUTION OF GAPS REVEALS NEW SINGULAR BEHAVIOUR

We now study the distribution of *gaps* $\Delta < 0$ and *overlaps* $\Delta > 0$, which play an important role near jamming. Positive Δ's are associated with unsatisfied patterns — which increase the loss of the system — whereas negative Δ's correspond to satisfied patterns — which are correctly classified with a margin ϵ and do not contribute to the loss. The latter offer an important measure not only at the jamming transition, but also in the over-parametrized regime, where they signal how much room is left around a minimum of the loss to fit additional patterns. In Fig. 8 (a,b) we show the two distributions for different depths $L = 2, 3, 5$ (positive Δ's have been rescaled by $\mathcal{L}^{1/2}$). Remarkably, they behave as power laws for about two decades, $P_+(\Delta/\sqrt{\mathcal{L}}) \sim (\Delta/\sqrt{\mathcal{L}})^\theta$ and $P_-(\Delta) \sim |\Delta|^{-\gamma}$, with novel exponents $\theta \approx 0.3$ and $\gamma \approx 0.2$ that appear to differ from those found for the jamming of particles (which are $\theta \approx 0.42311\ldots$ and $\gamma \approx 0.41269\ldots$). For comparison, in Fig. 8 (c,d) we show

the distribution of the same variables for tanh-networks, that also display power-law behaviors but with different exponents $\theta \approx 0.2$ and $\gamma \approx 0.16$.

In the case of spheres, the two exponents are related by an inequality that happens to be saturated [42, 57]. The inequality comes from arguments on the stability of jammed packings, and the fact that it is saturated (which can be proven for certain dynamics [43]) implies that such systems are *marginally stable*: they display an abundance of low-energy excitations and are prone to avalanche dynamics and crackling response when perturbed [43], a property associated with a hierarchical organization of the loss landscape [44, 45]. The presence of such power laws for deep networks thus suggests they are marginally stable as well, and that the learning dynamics may occur by avalanches where the unsatisfied constraints change by bursts. This will be subject of detailed studies in a future paper.

VII. IMAGE DATA: MNIST

We now consider a dataset called MNIST, which consists of a collection of black and white pictures of 28×28 pixels depicting handwritten digits from 0 to 9. The labels y_μ in principle would be the digits themselves ($y_\mu \in \{0, \dots, 9\}$), but to compare more directly with our previous experiments we gathered all the digits into two groups (even and odd numbers) with labels $y_\mu = \pm 1$. The architecture of the network is as in the previous sections: the d inputs are fed to a cascade of L fully-connected layers with h neurons, that in the end result in a single scalar output. The loss function used is the hinge loss.

If we kept the original input size of $28 \times 28 = 784$ then the majority of the network's weights would be necessarily concentrated in the first layer (the width h cannot be too large in order to be able to compute the Hessian). To avoid this issue, we opted for a reduction of the input size. We performed a principal component analysis (PCA) on the whole dataset and we identified the 10 dimensions that carry the most variance; then we used the components of each image along these directions as a new input of dimension $d = 10$. This projection hardly diminishes the performance of the network (we find the generalization accuracy to be larger than 90% at the jamming transition in Fig. 10 for $P \geq 10^4$).

In Fig. 9 we show that a jamming transition is also found for real data with a discontinuous behavior of N_Δ/N. Fig. 9 (a) shows the number of unsatisfied patterns per parameter N_Δ/N increasing P at fixed N, and in Fig. 9 (b) the same quantity is plotted against the loss. As for random data, the latter is less noisy. In Fig. 9 (c) we show that the number of misclassified data (i.e. the number of patterns with $y_\mu f(\mathbf{x}_\mu) < 0$) grows smoothly with the loss. These plots depict the same scenario as we found for random data, namely the one presented in Fig. 6 (a-c), except for the magnitude of the density of constraints at the transition with $N_\Delta/N \approx 0.5$ rather than $N_\Delta/N \approx 0.7$ as observed before. Hence, the number of unsatisfied patterns at the transition is not universal.

Also the spectrum of the Hessian matrix is similar to that of random data. In Fig. 9 (e-h) we show the positive part of the spectrum of $\mathcal{H}_p$, the total spectrum of $\mathcal{H}_p$, the spectrum of $\mathcal{H}_0$ and the spectrum of the total Hessian $\mathcal{H}$, respectively. As with random data: the matrix $\mathcal{H}_p$ has a symmetric spectrum and the matrix $\mathcal{H}_0$ has a finite number of zero modes and a gapped continuous distribution of modes at high energy. The spectrum of the total Hessian is again similar to that of $\mathcal{H}_0$, where the delta function in zero has been smeared.

The distribution of gaps (negative Δ's) is plotted in Fig. 9 (d), suggesting a power law with an exponent $\gamma = 0.25$ that is slightly larger than the value found for random data, $\gamma \approx 0.2$. It is unclear whether this difference is significant. We observed that the distribution of overlaps (positive Δ's) has large sample to sample variations (not shown), and the acquisition of enough statistics to measure it extensively will be done elsewhere.

A key difference between random and structured data however is the location N^* of the transition, shown in Fig. 10 versus the number P of patterns. For a fixed number P of MNIST pictures we ran several simulations with networks of different sizes, and found in this way the lowest value N^* for which all patterns could still be classified correctly. In the figure we present the results for two network architectures of different depths $L = 1, 3, 5$ (the width h was varied in order to control the network size). Key results are that (i) N^* is essentially independent of depth, especially at larger P and (ii) the minimum number of parameters N^* to fit the data is significantly smaller than for random data, a difference that seems to increase with P. The behavior of N^* in the (hypothetical) limit $P \to \infty$ could be indeed different from the linear scaling of random data: a sub-linear scaling or even a finite asymptotic value are possible alternatives. More generally, how the data structure affects the location of the transition $N^*(P)$ is an important question for the future.

VIII. CONCLUSION

By slightly changing the loss function — i.e. by considering the hinge loss rather than the commonly used cross entropy, a change that does not degrade performance — we could recast the problem of minimizing the loss function of deep networks into a constraint satisfaction problem with continuous degrees of freedom. This kind of problem has been abundantly studied in physics, in particular in the context of the jamming of particles, and some theoretical tools developed in that field readily apply to deep networks. In particular from this analogy one predicts a sharp transition as the number of parameters is reduced, separating a region where all constraints can be satisfied (that is, all the data are perfectly fitted) and the loss is zero after learning, and a region where the ratio

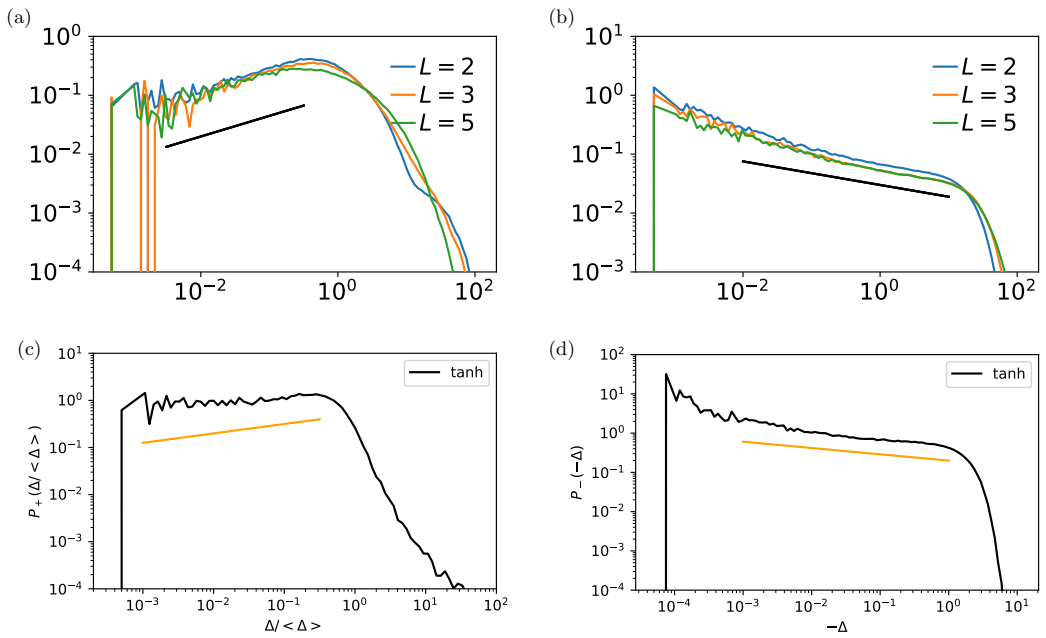


FIG. 8. (a) Distribution of re-scaled overlaps $z \equiv \Delta/\sqrt{\mathcal{L}} > 0$ near threshold, supporting that $P_+(z) \sim z^\theta$ with an exponent $\theta \approx 0.3$ that does not vary with L in the range probed. (b) The distribution of gaps $P_-(\Delta) \sim |\Delta|^{-\gamma}$ for $\Delta < 0$, with $\gamma \approx 0.2$, which again does not vary with L. (c-d) Distribution of overlaps and gaps for tanh-networks. The exponents in this case are different: $\theta \approx 0.2$, $\gamma \approx 0.16$.

of the number of unsatisfied constraints to the number of parameters is of order one. This ratio jumps discontinuously at the transition, where it attains a value smaller than one. Near that point, the spectrum of the Hessian is singular, reminiscent of a critical behavior. One key finding is that deep learning falls into the hypostatic universality class, similar to that of ellispes. We also observe a scaling behavior and new exponents characterizing how well constraints are satisfied or not (through the distributions $P_-(\Delta)$ and $P_+(\Delta)$, respectively). This bears comparison with the known behavior of packings of particles — where such singularities signal marginality and avalanche-type response — and of the perceptron (the simplest, shallow, neural network), that lie in the same universality class. Yet there is no theory so far to explain these exponents for deep networks. These results also shed light on some aspects of deep learning:

Not getting stuck in poor minima of the loss: Our analysis supports that in the overparametrized regime, the dynamics does not get stuck in poor minima because the number of constraints to satisfy (data to fit) P is too small to hamper minimization: the system is in an easy satisfiable phase. In particular assuming that a certain operator (namely the matrix $\mathcal{H}_p$) has a fraction of negative eigenvalues (which we could show in the case of the ReLU activation function and random data, and confirm numerically) implies that no poor minima exist if $P/N < P/N^* = \mathcal{O}(1)$. Here N is the number of effective degrees of freedom of the network, which in all the cases we studied is essentially equal to the number of parameters. This argument does not rule out the possibility that, with a very poor choice of initial condition, a poor minimum of the loss can be found. This is the case in particular if the network does not propagate the signal (then $N = 1$ in our formalism, independently of the number of parameters). Presumably usual tricks used to train deep networks (batch normalization, residual links, proper weight initialization, ...) ensure that the sensitivity of the network to its parameters is preserved during training so that N is indeed similar to the number of parameters, a hypothesis that would be useful to test in a broader setting.

In the under-parametrized phase the network gets stuck at a positive loss, either because the ground state is no longer at zero loss or because the system is trapped in an excited local minimum. The fact that the jamming transition itself depends on the dynamics (as is the case for the jamming of particles) suggests that in the under-parametrized case the network is in a local minimum.

Role of depth: We observed that depth is not helpful

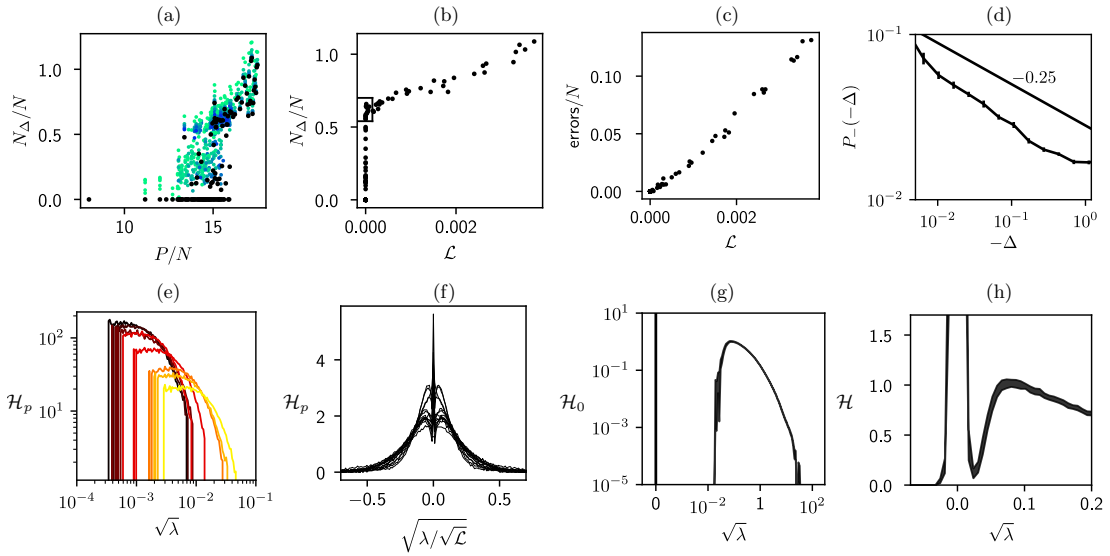


FIG. 9. Results with the MNIST dataset, keeping the first 10 PCA components. $d = 10$, $h = 30$ and $L = 5$ ($N = 3900$), varying $P = 1, \dots, 70k$. (a) The number of unsatisfied patterns N_Δ/N jumps discontinuously when $r = P/N$ is increased. (b) The same quantity is less noisy when plotted against the loss. (c) The number of misclassified data is a smooth function of the loss. (d) Distribution of the negative gaps ($\Delta < 0$), with a tentative exponent $\gamma = 0.25$. In the second row (e-h), the Hessian of the runs contained in the rectangle of plot (b) are shown: (e) positive part of the spectrum of $\mathcal{H}_p$, in logarithmic scale; (f) the total spectrum of $\mathcal{H}_p$ appears to be symmetric; (g) the spectrum of $\mathcal{H}_0$ presents a delta function in zero and a gapped continuous spectrum at high frequencies; (h) the spectrum of the total Hessian $\mathcal{H}$ resembles that for random data: the delta function in the spectrum of $\mathcal{H}_0$ is smeared.

to fit random data in fully connected networks: increasing depth and reducing width so that the total number of weights is fixed does not allow to fit the data with less parameters. We have also observed that this finding continues to hold in a realistic case based on MNIST. This may seem to clash with mathematical results, such as [9–11, 13], which establish that depth enhances expressivity. However, we tackle the question of expressivity for *realistic* data and learning protocols, which is quite different. Our results, that need confirmation by further studies on a broader range of data, point toward a negative answer for fully connected networks. It may be that the added expressive power of deep networks is only useful for architectures exploiting the symmetry and hierarchy in the data (e.g. as in convolutional networks). Alternatively, depth may play a role in accelerating the learning dynamics [74].

Reference point for network architectures: key properties of deep networks, including the learning dynamics and the generalization power, are believed to be affected by the landscape geometry. We have argued that there exists a critical line $N^*(P)$ where the landscape is singular (with both flat and stiff directions), suggesting that it will be a useful reference point to study dynamics and generalization. Concerning the former, our observations suggest that learning near threshold may occur by avalanches, that is, by abrupt changes in the set of data that are correctly classified. In practice, networks are generally trained in the overparametrized regime $N \gg N^*$. It would be interesting to investigate whether the learning dynamics, at intermediate times where many data are not fitted yet, resembles the dynamics near threshold and displays bursts of changes in the constraints. Concerning the latter, we have studied the effect of jamming on generalization since this article was first written, as appears in [46, 47].

ACKNOWLEDGMENTS

We thank C. Brito, C. Cammarota, T.S. Cohen, S. Franz, Y. LeCun, F. Krzakala, R. Ravasio, P. Urbani and L. Zdeborova for helpful discussions. This work was partially supported by the grant from the Simons Foundation (#454935 Giulio Biroli, #454953 Matthieu Wyart). M.W. thanks the Swiss National Science Foundation for support under Grant No. 200021-165509.

The manuscript [75], which appeared at the same time than ours, shows that the critical properties of the jamming transition found for the non-convex perceptron [34] hold more generally in some shallow networks. This universality is an intriguing result. Understanding the con-

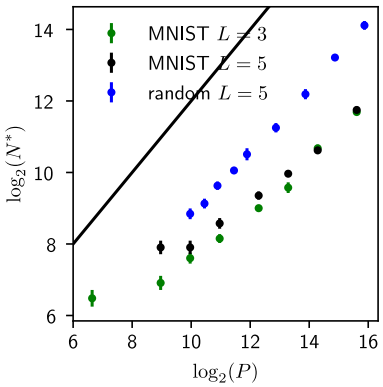


FIG. 10. Results with the MNIST dataset, keeping the first 10 PCA components (see main text), with $d = 10$ and varying P and h. The plot shows the number of parameters N^* at the jamming transition. For comparison, we also show the theoretical upper bound (solid curve) and the results found with random data (black points). The maximum number of steps is $2 \cdot 10^6$.

nection with our findings, which show instead a jamming transition similar to that of ellipses, is certainly worth future studies.

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems* (2012) pp. 1097–1105.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," Nature **521**, 436 (2015).

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," IEEE Signal processing magazine **29**, 82–97 (2012).

[4] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," Nature **529**, 484 (2016).

[5] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, *et al.*, "Mastering the game of go without human knowledge," Nature **550**, 354 (2017).

[6] Yann LeCun, Yoshua Bengio, *et al.*, "Convolutional networks for images, speech, and time series," The handbook of brain theory and neural networks **3361**, 1995 (1995).

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016) pp. 770–778.

[8] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning* (2015) pp. 448–456.

[9] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio, "On the number of linear regions of deep neural networks," in *Advances in neural information processing systems* (2014) pp. 2924–2932.

[10] Monica Bianchini and Franco Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," IEEE transactions on neural networks and learning systems **25**, 1553–1565 (2014).

[11] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein, "On the expressive power of deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 70, edited by Doina Precup and Yee Whye Teh (PMLR, International Convention Centre, Sydney, Australia, 2017) pp. 2847–2854.

[12] Ronen Eldan and Ohad Shamir, "The power of depth for feedforward neural networks," in *Conference on Learning Theory* (2016) pp. 907–940.

[13] Holden Lee, Rong Ge, Tengyu Ma, Andrej Risteski, and Sanjeev Arora, "On the ability of neural nets to express distributions," in *Proceedings of the 2017 Conference on Learning Theory*, Proceedings of Machine Learning Research, Vol. 65, edited by Satyen Kale and Ohad Shamir (PMLR, Amsterdam, Netherlands, 2017) pp. 1271–1296.

[14] Elizabeth Gardner, "The space of interactions in neural network models," Journal of physics A: Mathematical and general **21**, 257 (1988).

[15] Rémi Monasson and Riccardo Zecchina, "Weight space structure and internal representations: a direct approach to learning and generalization in multilayer neural networks," Physical review letters **75**, 2432 (1995).

[16] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, "Understanding deep learning requires rethinking generalization," International Conference on Learning Representations (2017).

[17] Eric B Baum, "On the capabilities of multilayer perceptrons," Journal of complexity **4**, 193–215 (1988).

[18] Ludovic Berthier and Giulio Biroli, "Theoretical perspective on the glass transition and amorphous materials," Reviews of Modern Physics **83**, 587 (2011).

[19] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun, "The loss surfaces of multilayer networks," in *Artificial Intelligence and Statistics* (2015) pp. 192–204.

[20] C Daniel Freeman and Joan Bruna, "Topology and geometry of deep rectified network optimization landscapes," International Conference on Learning Representations (2017).

[21] Elad Hoffer, Itay Hubara, and Daniel Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *Advances in Neural Information Processing Systems* (2017) pp. 1729–1739.

[22] Daniel Soudry and Yair Carmon, "No bad local minima: Data independent training error guarantees for multilayer neural networks," arXiv preprint arXiv:1605.08361 (2016).

[23] Yaim Cooper, "The loss landscape of overparameterized neural networks," arXiv preprint arXiv:1804.10200 (2018).

[24] Levent Sagun, Léon Bottou, and Yann LeCun, "Singularity of the hessian in deep learning," International Conference on Learning Representations (2017).

[25] Levent Sagun, Utku Evci, V. Uğur Güney, Yann Dauphin, and Léon Bottou, "Empirical analysis of the hessian of over-parametrized neural networks," ICLR 2018 Workshop Contribution, arXiv:1706.04454 (2017).

[26] Andrew J Ballard, Ritankar Das, Stefano Martiniani, Dhagash Mehta, Levent Sagun, Jacob D Stevenson, and David J Wales, "Energy landscapes for machine learning," Physical Chemistry Chemical Physics (2017).

[27] Zachary C Lipton, "Stuck in a what? adventures in weight space," International Conference on Learning Representations (2016).

[28] Marco Baity-Jesi, Levent Sagun, Mario Geiger, Stefano Spigler, Gerard Ben Arous, Chiara Cammarota, Yann LeCun, Matthieu Wyart, and Giulio Biroli, "Comparing dynamics: Deep neural networks versus glassy systems," in *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80, edited by Jennifer Dy and Andreas Krause (PMLR, Stockholmsmässan, Stockholm Sweden, 2018) pp. 314–323.

[29] M. Wyart, "On the rigidity of amorphous solids," Annales de Phys **30**, 1–113 (2005).

[30] Andrea J. Liu, Sidney R. Nagel, W Saarloos, and Matthieu Wyart, *Dynamical Heterogeneities in Glasses, Colloids, and Granular Media* (2010).

[31] Florent Krzakala and Jorge Kurchan, "Landscape analysis of constraint satisfaction problems," Physical Review E **76**, 021122 (2007).

[32] Lenka Zdeborová and Florent Krzakala, "Phase transitions in the coloring of random graphs," Physical Review E **76**, 031131 (2007).

[33] Silvio Franz, Giorgio Parisi, Maxime Sevelev, Pierfrancesco Urbani, and Francesco Zamponi, "Universality of the sat-unsat (jamming) threshold in non-convex continuous constraint satisfaction problems," SciPost Physics **2**, 019 (2017).

[34] Silvio Franz and Giorgio Parisi, "The simplest model of jamming," Journal of Physics A: Mathematical and Theoretical **49**, 145001 (2016).

[35] Matthieu Wyart, Leonardo E Silbert, Sidney R Nagel, and Thomas A Witten, "Effects of compression on the vibrational modes of marginally jammed solids," Physical Review E **72**, 051306 (2005).

[36] L. E. Silbert, A. J. Liu, and S. R. Nagel, "Vibrations and diverging length scales near the unjamming transition," Phys. Rev. Lett. **95**, 098301 (2005).

[37] Aleksandar Donev, Ibrahim Cisse, David Sachs, Evan A. Variano, Frank H. Stillinger, Robert Connelly, Salvatore Torquato, and P. M. Chaikin, "Improving the density of jammed disordered packings using ellipsoids," Science **303**, 990–993 (2004).

[38] Mitch Mailman, Carl F. Schreck, Corey S. O'Hern, and Bulbul Chakraborty, "Jamming in systems composed of frictionless ellipse-shaped particles," Phys. Rev. Lett. **102**, 255501 (2009).

[39] Z. Zeravcic, N. Xu, A. J. Liu, S. R. Nagel, and W. van Saarloos, "Excitations of ellipsoid packings near jamming," Europhys. Lett. **87**, 26001 (2009).

[40] Carolina Brito, Harukuni Ikeda, Pierfrancesco Urbani, Matthieu Wyart, and Francesco Zamponi, "Universality of jamming of non-spherical particles," arXiv preprint arXiv:1807.01975 (2018).

[41] This transition influences the generalization properties of deep networks, too. This has been observed, for instance, in [46, 76], and studied by the authors in [47] (preprint).

[42] Matthieu Wyart, "Marginal stability constrains force and pair distributions at random close packing," Phys. Rev. Lett. **109**, 125502 (2012).

[43] Markus Müller and Matthieu Wyart, "Marginal stability in structural, spin, and electron glasses," Annual Review of Condensed Matter Physics **6**, 177–200 (2015).

[44] Silvio Franz and Stefano Spigler, "Mean-field avalanches in jammed spheres," Physical Review E **95**, 022139 (2017).

[45] Patrick Charbonneau, Jorge Kurchan, Giorgio Parisi, Pierfrancesco Urbani, and Francesco Zamponi, "Fractal free energy landscapes in structural glasses," Nature Communications **5** (2014).

[46] Stefano Spigler, Mario Geiger, Stéphane d'Ascoli, Levent Sagun, Giulio Biroli, and Matthieu Wyart, "A jamming transition from under-to over-parametrization affects loss landscape and generalization," NIPS Workshop "Integration of Deep Learning Theories" contribution, arXiv preprint arXiv:1810.09665 (2018).

[47] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart, "Scaling description of generalization with number of parameters in deep learning," arXiv preprint arXiv:1901.01608 (2019).

[48] A.C. Anderson, *Amorphous Solids: Low Temperature Properties*, edited by W. A. Phillips, Topics in Current Physics, Vol. 24 (Springer, Berlin, 1981).

[49] Alexei V. Tkachenko and Thomas A. Witten, "Stress propagation through frictionless granular material," Phys. Rev. E **60**, 687–696 (1999).

[50] A similar decomposition has been used in [25, 77, 78].

[51] Once again, this statement is true except for global translation or rotation of the systems, whose number however is fixed in the large N limit and disappears when the ratio N_Δ/N is considered.
[52] Corey S. O'Hern, Leonardo E. Silbert, Andrea J. Liu, and Sidney R. Nagel, "Jamming at zero temperature and zero applied stress: The epitome of disorder," Phys. Rev. E **68**, 011306–011324 (2003).
[53] Eric DeGiuli, Adrien Laversanne-Finot, Gustavo Alberto Düring, Edan Lerner, and Matthieu Wyart, "Effects of coordination and pressure on sound attenuation, boson peak and elasticity in amorphous solids," Soft Matter **10**, 5628–5644 (2014).
[54] Le Yan, Eric DeGiuli, and Matthieu Wyart, "On variational arguments for vibrational modes near jamming," EPL (Europhysics Letters) **114**, 26003 (2016).
[55] Silvio Franz, Giorgio Parisi, Pierfrancesco Urbani, and Francesco Zamponi, "Universal spectrum of normal modes in low-temperature glasses," Proceedings of the National Academy of Sciences **112**, 14539–14544 (2015).
[56] Gustavo Düring, Edan Lerner, and Matthieu Wyart, "Phonon gap and localization lengths in floppy materials," Soft Matter **9**, 146–154 (2013).
[57] Edan Lerner, Gustavo During, and Matthieu Wyart, "Low-energy non-linear excitations in sphere packings," Soft Matter **9**, 8252–8263 (2013).
[58] E. Lerner, G. Düring, and M. Wyart, "Toward a microscopic description of flow near the jamming threshold," EPL (Europhysics Letters) **99**, 58003 (2012).
[59] Patrick Charbonneau, Eric I. Corwin, Giorgio Parisi, and Francesco Zamponi, "Universal microstructure and mechanical stability of jammed packings," Physical Review Letters **109**, 205501– (2012).
[60] Patrick Charbonneau, Jorge Kurchan, Giorgio Parisi, Pierfrancesco Urbani, and Francesco Zamponi, "Exact theory of dense amorphous hard spheres in high dimension. iii. the full replica symmetry breaking solution," Journal of Statistical Mechanics: Theory and Experiment **2014**, 10009 (2014).
[61] Marc Mézard, Giorgio Parisi, and Miguel Virasoro, *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*, Vol. 9 (World Scientific Publishing Company, 1987).
[62] Patrick Charbonneau, Eric I Corwin, Giorgio Parisi, and Francesco Zamponi, "Jamming criticality revealed by removing localized buckling excitations," Physical Review Letters **114**, 125504 (2015).
[63] The parameter ϵ fixes the scale of the loss, in the sense that, if one rescales both ϵ and the weights of the last layer by the same factor α, then all the observables are equivariant with respect to this transformation ($\Delta \to \alpha\Delta$, $\mathcal{L} \to \alpha^2\mathcal{L}$, $N_\Delta \to N_\Delta$, ...). Consequently, since any positive value of ϵ leads to the same behavior of the system, we have arbitrarily fixed its value to $\epsilon = \frac{1}{2}$. If ϵ were 0, the network would try to enforce $f(\mathbf{x};\mathbf{W}) \equiv 0$ regardless of the specific pattern.
[64] Xavier Gastaldi, "Shake-shake regularization of 3-branch residual networks," International Conference on Learning Representations (2017).
[65] `https://github.com/mariogeiger/pytorch_shake_shake`.
[66] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein, "Deep information propagation," arXiv preprint arXiv:1611.01232 (2016).
[67] For finite P, N^* will present fluctuations induced by differences of initial conditions. The fluctuations of P/N^* are however expected to vanish in the limit where P and N^* become large. This phenomenon is well-known for the jamming of particles, and is referred to as finite size effects.
[68] For the cross entropy, when the data become separable true minima exists only for diverging weights, a complication that does not occur with the hinge loss.
[69] Weights and biases are initialized with a uniform distribution on $[-\sigma, \sigma]$, where $\sigma^2 = 1/f_{in}$ and f_{in} is the number of incoming connections.
[70] Andrew M Saxe, James L McClelland, and Surya Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," International Conference on Learning Representations (2014).
[71] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," International Conference on Learning Representations (2015).
[72] Ryan P Adams, Jeffrey Pennington, Matthew J Johnson, Jamie Smith, Yaniv Ovadia, Brian Patton, and James Saunderson, "Estimating the spectral density of large implicit matrices," arXiv preprint arXiv:1802.03451 (2018).
[73] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao, "An investigation into neural net optimization via hessian eigenvalue density," arXiv preprint arXiv:1901.10159 (2019).
[74] Ravid Shwartz-Ziv and Naftali Tishby, "Opening the black box of deep neural networks via information," arXiv preprint arXiv:1703.00810 (2017).
[75] P. Urbani S. Franz, S. Hwang, "Jamming in multilayer supervised learning models," arXiv preprint arXiv:1809.09945 (2018).
[76] Madhu S Advani and Andrew M Saxe, "High-dimensional dynamics of generalization error in neural networks," arXiv preprint arXiv:1710.03667 (2017).
[77] Jeffrey Pennington and Yasaman Bahri, "Geometry of neural network loss surfaces via random matrix theory," in *International Conference on Machine Learning* (2017) pp. 2798–2806.
[78] James Martens, "New insights and perspectives on the natural gradient method," arXiv preprint arXiv:1412.1193 (2014).

Appendix A: Effective number of degrees of freedom

Due to several effects discussed in the main text, the function $f(\mathbf{x};\mathbf{W})$ can effectively depend on less variables that the number of parameters, and thus reduce the dimension of the space spanned by the gradients $\nabla_{\mathbf{W}} f(\mathbf{x};\mathbf{W})$ that enters in the theory. For instance, there could be symmetries that reduce the number of effective degrees of freedom (e.g. each ReLU activation function has one of such symmetries, since one can rescale inputs and outputs in such a way that the post-activation is left invariant); another reason could be that a neuron might never activate for all the training data, thus effectively reducing the number of neurons in the network; furthermore, we expect that the network's true dimension would also be reduced if its architecture presents some bottlenecks, is poorly designed or poorly initialized. For

example if all biases are too negative on the neurons of one layer in the Relu case, the network does not transmit any signals, leading to $N = 1$ and to the possible absence of unstable directions even if the number of parameters is very large.

It is tempting to define the effective dimension by considering the dimension of the space spanned by $\nabla_{\mathbf{W}} f(\mathbf{x}_\mu; \mathbf{W})$ as μ varies. This definition is not practical for small number of samples P however, because this dimension would be bounded by P. We can overcome such a problem by considering a neighborhood of each point $\mathbf{x}_\mu$, where the network's function and its gradient can be expanded in the pattern space:

$$f(\mathbf{x}) \approx f(\mathbf{x}_\mu) + (\mathbf{x} - \mathbf{x}_\mu) \cdot \nabla_{\mathbf{x}} f(\mathbf{x}_\mu), \tag{A1}$$

$$\nabla_{\mathbf{W}} f(\mathbf{x}) \approx \nabla_{\mathbf{W}} f(\mathbf{x}_\mu) + (\mathbf{x} - \mathbf{x}_\mu) \cdot \nabla_{\mathbf{x}} \nabla_{\mathbf{W}} f(\mathbf{x}_\mu). \tag{A2}$$

Varying the pattern μ and the point $\mathbf{x}$ in the neighborhood of $\mathbf{x}_\mu$, we can build a family M of vectors:

$$M = \{\nabla_{\mathbf{W}} f(\mathbf{x}_\mu) + (\mathbf{x} - \mathbf{x}_\mu) \cdot \nabla_{\mathbf{x}} \nabla_{\mathbf{W}} f(\mathbf{x}_\mu)\}_{\mu,\mathbf{x}}. \tag{A3}$$

We then define the effective dimension N as the dimension of M. Because of the linear structure of M, it is sufficient to consider, for each μ, only $d+1$ values for x, e.g. $x - x_\mu = 0, \hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_d$, where $\hat{\mathbf{e}}_n$ is the unit vector along the direction n. The effective dimension is therefore

$$N = \mathrm{rk}(G), \tag{A4}$$

where the elements of the matrix G are defined as

$$G_{i,\alpha} \equiv \partial_{W_i} f(\mathbf{x}_\mu) + \hat{\mathbf{e}}_n \cdot \nabla_{\hat{\mathbf{e}}_n} \partial_{W_i} f(\mathbf{x}_\mu), \tag{A5}$$

with $\alpha \equiv (\mu, n)$. The index n ranges from 0 to d, and $\hat{\mathbf{e}}_0 \equiv 0$.

In Fig. 11 we show the effective number of parameters N versus the total number of parameters $\tilde{N}$, in the case of a network with $L = 3$ layers trained on the first 10 PCA components of the MNIST dataset. There is no noticeable difference between the two quantities: the only reduction is due to the symmetries induced by the ReLU functions (there is one such symmetry per neuron. Indeed the ReLU function $\rho(z) = z\Theta(z)$ satisfies $\Lambda\rho(z/\Lambda) \equiv \rho(z)$.) We observed the same results for random data.

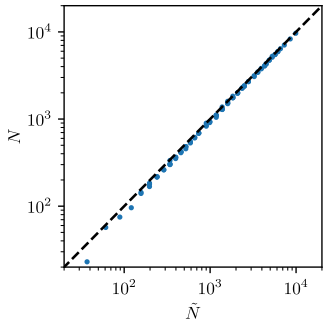


FIG. 11. Results with the MNIST dataset, keeping the first 10 PCA components. $d = 10$ and $L = 3$, varying P and h. Effective N vs total number of parameters $\tilde{N}$. N is always smaller than $\tilde{N}$ because there is a symmetry per each ReLU-neuron in the network.

Appendix B: $\mathrm{sp}(H_p)$ is symmetric for ReLu activation function and random data

We consider $\mathcal{H}_p = -\sum_\mu y_\mu \rho(\Delta_\mu) \hat{\mathcal{H}}_\mu$, where $\hat{\mathcal{H}}_\mu$ is the Hessian of the network function $f(\mathbf{x}_\mu; \mathbf{W})$ and ρ is the Relu function. We want to argue that the spectrum of $\mathcal{H}_p$ is symmetric in the limit of large N.

We do two main hypothesis: First, the trace of any finite power of $\mathcal{H}_p$ is self-averaging (concentrates) with respect to the average over the random data:

$$\frac{1}{N}\mathrm{tr}(\hat{\mathcal{H}_p}^n) = \frac{1}{N}\overline{\mathrm{tr}(\hat{\mathcal{H}_p}^n)}.$$

Second,

$$\frac{1}{N}\sum_{\mu_1,\cdots,\mu_n} \overline{y_{\mu_1}\rho(\Delta_{\mu_1})\cdots y_{\mu_n}\rho(\Delta_{\mu_n})\mathrm{tr}(\hat{\mathcal{H}}_{\mu_1}\cdots\hat{\mathcal{H}}_{\mu_n})} =$$

$$\frac{1}{N}\sum_{\mu_1,\cdots,\mu_n} \overline{y_{\mu_1}\rho(\Delta_{\mu_1})\cdots y_{\mu_n}\rho(\Delta_{\mu_n})}\,\overline{\mathrm{tr}(\hat{\mathcal{H}}_{\mu_1}\cdots\hat{\mathcal{H}}_{\mu_n})}$$

The first hypothesis is natural since $\hat{\mathcal{H}}_p$ is a very large random matrix, for which the density of eigenvalues is expected to become a non-fluctuating quantity. The second hypothesis is more tricky: it is natural to assume that the trace concentrates, however one also need to show that the sub-leading corrections to the self-averaging of the trace can be neglected.

Using these two hypothesis and the result, showed below, that

$$\overline{\mathrm{tr}(\hat{\mathcal{H}}_{\mu_1}\cdots\hat{\mathcal{H}}_{\mu_n})} = 0 \tag{B1}$$

for all n odds, one can conclude that all odds traces of $\hat{\mathcal{H}}_p$ are zero. This implies that the spectrum of $\hat{\mathcal{H}}_p$ is symmetric, more precisely that the fractions of negative and positive eigenvalues are equal.

In order to show that the statement (B1) above holds, let us argue first that $\overline{\mathrm{tr}(\hat{\mathcal{H}}_\mu^n)} = 0$ for any *odd* n.

$$\mathrm{tr}(\hat{\mathcal{H}}_\mu^n) = \sum_{i_1,i_2,\ldots,i_n} \hat{\mathcal{H}}^\mu_{i_1,i_2}\hat{\mathcal{H}}^\mu_{i_2,i_3}\cdots\hat{\mathcal{H}}^\mu_{i_n,i_1}, \tag{B2}$$

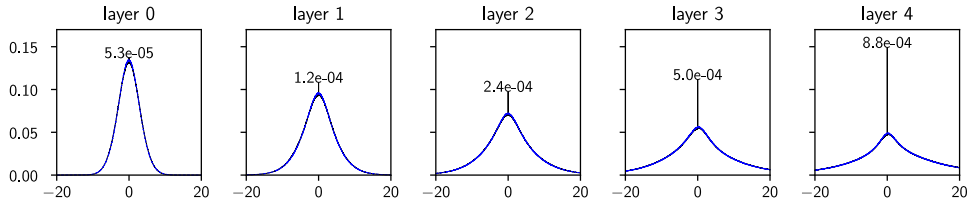


FIG. 12. Density of the pre-activations for each layers with $L = 5$ and random data, averaged over all the runs just above the jamming transition with that architecture. Black: distribution obtained over the training set. Blue: previously unseen random data (the two curves are on top of each other except for the delta in zero). The values indicate the mass of the peak in zero, which is only present when the training set is considered.

where the indices $i_1, \dots, i_n$ stand for synapses connecting a pair of neurons (i.e. each index is associated with a synaptic weight $W^{(j)}_{\alpha,\beta}$: we are not writing all the explicit indexes for the sake of clarity). The term of the hessian obtained when differentiating with respect to weights $W^{(j)}_{\alpha,\beta}$ and $W^{(k)}_{\gamma,\delta}$ reads

$$\hat{\mathcal{H}}^{\mu;(jk)}_{\alpha\beta;\gamma\delta} = \sum_{\pi_0,\dots,\pi_L} \theta(a^\mu_{L,\pi_L}) \cdots \theta(a^\mu_{1,\pi_1}) x^\mu_{\pi_0} \cdot \cdot \partial_{W^{(j)}_{\alpha,\beta}} \partial_{W^{(k)}_{\gamma,\delta}} \left[W^{(L+1)}_{\pi_L} W^{(L)}_{\pi_L,\pi_{L-1}} \cdots W^{(1)}_{\pi_1\pi_0} \right]. \quad \text{(B3)}$$

where we denoted with a the inputs in the nodes of the network. Our argument is based on a symmetry of the problem with random data: changing the sign of the weight of the last layer $W^{(L+1)} \longrightarrow -W^{(L+1)}$ and changing the labels $y_\mu \longrightarrow -y_\mu$ leaves the loss unchanged. We will show that this symmetry implies that $\mathrm{tr}(\hat{\mathcal{H}}^n_\mu)$ averaged over the random labels is zero for odd n.

In fact, note that the sum in Eq.B3 contains a weight per each layer in the network, with the exception of the two layers j, k with respect to which we are deriving. This implies that any element of the hessian matrix where we have not differentiated with respect to the last layer ($j, k < L+1$) is an odd function of the last layer $W^{(L+1)}$, meaning that if $W^{(L+1)} \longrightarrow -W^{(L+1)}$, then the sign of all these Hessian elements is inverted as well.

If in the argument of the sum in Eq. (B2) there is no index belonging to the last layer, then the whole term changes sign under the transformation $W^{(L+1)} \longrightarrow -W^{(L+1)}$. Suppose now that, on the contrary, there are m terms with one index belonging to the last layer (we need not consider the case of two indices both belonging to the last layer because the corresponding term in the Hessian would be 0, as one can see in Eq. (B3)). For each index equal to $L + 1$ (the last layer), there are exactly two terms: $\hat{\mathcal{H}}^\mu_{j,L+1} \hat{\mathcal{H}}^\mu_{L+1,k}$ (for some indexes j, k). Since j, k cannot be $L + 1$ too, this implies that the number m of terms with an index belonging to the last layer is always even. Consequently, when the sign of $W^{(L+1)}$ is reversed, the argument of the sum in Eq. (B2) is multiplied by $(-1)^{n-m}$ (once for each term *without* an index belonging to the last layer), which is equal to -1 if n is odd. The same symmetry can be used to show that a matrix made of an odd product of matrices $\hat{\mathcal{H}}_\mu$, such as $\hat{\mathcal{H}}_\mu \hat{\mathcal{H}}_{\mu'} \hat{\mathcal{H}}_{\mu''}$, must also have a symmetric spectrum, concluding our argument.

Appendix C: Density of pre-activations for ReLU activation functions

The densities of pre-activation (i.e. the value of the neurons before applying the activation function) is shown in Fig. 12 for random data. It contains a delta distribution in zero. The number N_c of pre-activations equal to zero when feeding a network $L = 5$ all its random dataset is $N_c \approx 0.21N$, corresponding to the number of directions in phase space where cusps are present in the loss function. For MNIST data we find $N_c \approx 0.19N$. By taking $L = 2$ and random data we find $N_c \approx 0.25N$. In these directions, stability can be achieved even if the hessian would indicate an instability. For this reason, instead of N_- in Equation 11 one should use $N/2 - N_c \approx 0.25N$.

2 Overparameterization & Double-Descent

The following paper is the preprint version of Spigler et al. (2019). It is a short version of Geiger et al. (2019) with added observation of double-descent.

Candidate contributions The candidate discovered the double-descent phenomenon in classification with neural networks.

A jamming transition from under- to over-parametrization affects generalization in deep learning

S Spigler[1,⋆], M Geiger[1,⋆], S d'Ascoli[2], L Sagun[1], G Biroli[2] and M Wyart[1]

[1] Institute of Physics, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland,

[2] Laboratoire de Physique Statistique, École Normale Supérieure, PSL Research University, 75005 Paris, France

⋆ These two authors contributed equally.

June 2019

Abstract. In this paper we first recall the recent result that in deep networks a phase transition, analogous to the jamming transition of granular media, delimits the over- and under-parametrized regimes where fitting can or cannot be achieved. The analysis leading to this result support that for proper initialization and architectures, in the whole over-parametrized regime poor minima of the loss are not encountered during training, because the number of constraints that hinders the dynamics is insufficient to allow for the emergence of stable minima. Next, we study systematically how this transition affects generalization properties of the network (i.e. its predictive power). As we increase the number of parameters of a given model, starting from an under-parametrized network, we observe for gradient descent that the generalization error displays three phases: *(i)* initial decay, *(ii)* increase until the transition point — where it displays a cusp — and *(iii)* slow decay toward an asymptote as the network width diverges. However if early stopping is used, the cusp signaling the jamming transition disappears. Thereby we identify the region where the classical phenomenon of over-fitting takes place as the vicinity of the jamming transition, and the region where the model keeps improving with increasing the number of parameters, thus organizing previous empirical observations made in modern neural networks.

1. Introduction

Despite the remarkable progress in designing [1, 2] and training [3] neural networks, there is still no general theory explaining their success, and their understanding remains mostly empirical. Central questions need to be clarified, such as what conditions need to be met in order to fit data properly, why the dynamics does not get stuck in spurious local minima, and how the depth of the network affects its loss landscape.

Complex physical systems with non-convex energy landscapes featuring an exponentially large number of local minima are called glasses [4]. An analogy between deep networks and glasses has been proposed [5, 6], in which the learning dynamics is expected to slow down and to get stuck in the highest minima of the loss. Yet, in the regime where the number of parameters is large (often considered in practice), several

numerical and rigorous works [7, 8, 9, 10, 11] suggest a different landscape geometry where the loss function is characterized by a connected level set. Furthermore, studies of the Hessian of the loss function [12, 13, 14] and of the learning dynamics [15, 16] support that the landscape is characterized by an abundance of flat directions, even near its bottom, at odds with traditional glasses.

1.1. Jamming transition and supervised learning

In a previous article [17] we have introduced an analogy between supervised learning with deep neural networks and a class of glassy systems, namely random dense packings of repulsive particles. It generalized a previous seminal analogy established between the loss landscape of the perceptron (the simplest network without hidden neurons) and the energy landscape of spherical particles [18], and specified the universality class to which deep learning corresponds to. The critical behavior of these granular systems, although very general, is of easier understanding when we consider particles that interact only within a finite range: upon increasing their density, such systems undergo a critical *jamming transition* [19, 20] when there is no longer space to accommodate all the particles without them touching one another. Before the transition the energy is zero, and after it increases with the density. The inclusion of longer-range interactions blurs the transition but its effects are still affecting the energy landscape [19]. Deep networks behave similarly when we look at the training loss, and, again, a clear criticality emerges when considering a "finite-range" loss function — the hinge loss: when the number P of training points is small enough, the network is able to learn the whole training set and reaches zero training loss, and upon increasing the dataset size we find a critical "jamming" point where perfect training does not occur and learning gets stuck in a positive minimum of the the training loss.

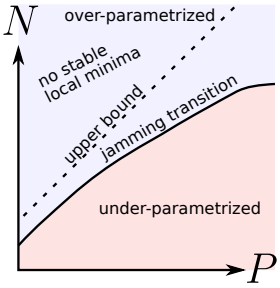


Figure 1. N: degrees of freedom, P: training examples.

For the full analogy we point to the aforementioned paper [17]. In the present work we first review the arguments that show that the existence of the jamming transition, studied in the (N, P) plane where N is the number of degrees of freedom of the network (informally, its size) and P is the size of the training set. As it turns out, there is a critical line $N^\star(P)$ (whose exact location can depend on the chosen dynamics) delimiting two phases, one where the learning reaches zero training loss, and one where it gets stuck in a minimum with finite loss — see Fig. 1. We present some numerical results that characterize the different phases, both for random data and for

the MNIST dataset, using fully-connected networks with ReLU activation functions. Then, we show novel data that illustrate that this transition affects the most crucial aspect of learning, namely the generalization error. We observe that generalization properties are strongly affected by the proximity to the jamming transition: for a gradient descent dynamics, in the under-parameterized phase before jamming (large P or small N) the generalization error is increasing; at the transition it displays a cusp; after jamming, in the over-parametrized phase, the error decreases monotonically. If early stopping is used, the cusp disappears, implying that the jamming transition is precisely the point where over-fitting is very strong.

1.2. Generalization versus over-fitting

The puzzle regarding the good generalization properties of neural networks despite their large size has been the topic of study for several other works. Some of them focus on the effects of various ways of regularizing the network, thereby effectively reducing the dimension [21, 22, 23, 24]. Yet another body of works focus on the effects of sheer size of a neural network [25, 26, 27].

Among previous studies, [28] stands out as a natural predecessor of our work. In [28] the authors present one of the first empirical observations of the cusp in test error of a non-linear model, a behaviour that is reminiscent of the perceptron. There, the training dynamics is run on a two-layer student network whose training data is provided by a teacher network with a similar architecture. The authors have observed (i) a cusp in generalization error, and (ii) monotonic decay in the test error when early stopping is used.

In this work, we show that the cusp in generalization corresponds to a phase transition where the number of unsatisfied constraints suddenly drops to zero as N increases. We quantify how the location of the transition $N^*(P)$ depends on P for both random data and natural images, and find that the data structure significantly affects $N^*(P)$. Our analysis makes it clear that $N^* \neq P$, an assumption sometimes made in previous studies. Overall, it relates the cusp in generalization to a well-known body of literature in physics associated with the "jamming" transition.

Since the initial preparation of the present work, the field has progressed quickly within a matter of months. The described cusp in the generalization error has been observed empirically in [29] for random forest models and simple neural networks. Further theoretical studies on regression showed a precise mathematical description of the cusp behaviour in [30, 31, 32], albeit on models that are practically somewhat further away from modern neural networks. Finally, in [33], our subsequent work, we develop a quantitative theory for (i) the cusp which is associated to the divergence of the norm of the output function at the critical point of the phase transition and (ii) the asymptotic behaviour of the generalization error as $N \to \infty$ which is associated with the reduced fluctuations of the output function. That work also shows that after ensemble averaging several networks, performance is optimal near the jamming the threshold, emphasizing the practical importance of this transition.

2. Theoretical framework

In this section we recall in detail the analogy between jamming and supervised learning for deep neural networks [17, 18]. This will set the stage for the following thorough analysis of the phase transition and its role on generalization.

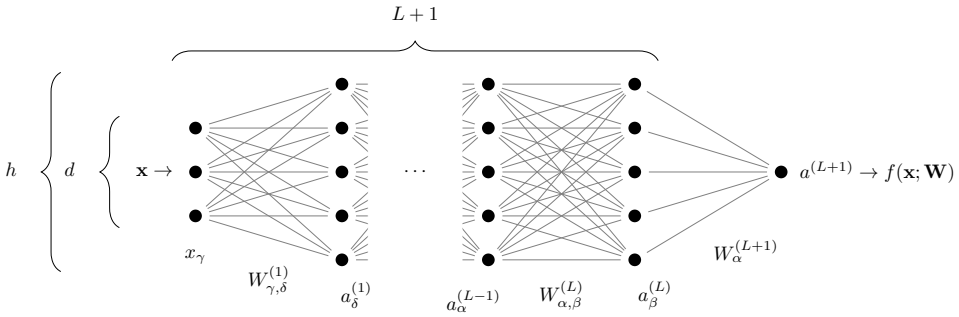


Figure 2. Architecture of a fully-connected network with L hidden layers of constant size h. Points indicate neurons, connections between them are characterized by a weight. Biases are not represented here.

2.1. Set-up

We consider a binary classification problem, with a set of P distinct training data denoted $\{(\mathbf{x}_\mu, y_\mu)\}_{\mu=1}^P$. The vector $\mathbf{x}_\mu$ is the input, which lives in a d-dimensional space, and $y_\mu = \pm 1$ is its label. We denote by $f(\mathbf{x};\mathbf{W})$ the output of a fully-connected network corresponding to an input $\mathbf{x}$, parametrized by $\mathbf{W}$. We represent the network as in Fig. 2, and the output function is written recursively as

$$f(\mathbf{x};\mathbf{W}) \equiv a^{(L+1)}, \tag{1}$$

$$a^{(i)}_\beta = \sum_\alpha W^{(i)}_{\alpha,\beta}\, \rho\left(a^{(i-1)}_\alpha\right) - B^{(i)}_\beta, \tag{2}$$

$$a^{(1)}_\beta = \sum_\alpha W^{(1)}_{\alpha,\beta}\, x_\alpha - B^{(1)}_\beta, \tag{3}$$

where $a^{(i)}_\alpha$ are the preactivations. In our notation the set of parameters $\mathbf{W}$ includes, with a slight abuse of notation, both the weights $W^{(i)}_{\alpha,\beta}$ and the biases $B^{(i)}_\alpha$. $\rho(z)$ is the non-linear activation function, e.g. the ReLU $\rho(z) = z\theta(z)$ or the hyperbolic tangent $\rho(z) = \tanh(z)$. The parameters are learned by minimizing the quadratic hinge loss:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{P}\sum_{\mu=1}^{P} \frac{1}{2}\max\left(0, \Delta_\mu\right)^2 \equiv \frac{1}{P}\sum_{\mu\in m} \frac{1}{2}\Delta_\mu^2, \tag{4}$$

where $\Delta_\mu \equiv 1 - y_\mu f(\mathbf{x}_\mu;\mathbf{W})$ and m is the set of patterns with $\Delta_\mu > 0$ and contains N_Δ elements. These patterns describe *unsatisfied constraints*: they are either incorrectly classified or classified with an insufficient margin (whereas patterns with $\Delta_\mu < 0$ are learned with margin 1). We adopt this loss function since it makes the jamming transition simpler to analyze‡, but this choice does not influence the performance of the network, as we have reported in [17].

‡ The often used cross-entropy loss function also displays a transition where all data are well-fitted. However, in the over-parametrized regime the dynamic never stops, as the total loss vanishes only if the output and therefore the weights diverge. Imposing a time cut-off is done in practice, but it blurs the criticality near jamming, as exemplified below with the early stopping procedure.

We are interested in the transition between an over-parametrized phase where the network can satisfy all the constraints ($\mathcal{L} = 0$) and an under-parametrized phase where some constraints remain unsatisfied ($\mathcal{L} > 0$).

2.2. A note on the effective number of parameters

In our discussion the notion of effective number of degrees of freedom is important. In the space of functions going from the neighborhoods of the training set to real numbers, consider the manifold of functions $f(\mathbf{x}; \mathbf{W})$ obtained by varying $\mathbf{W}$. We denote by $N_{\text{eff}}(\mathbf{W})$ the dimension of the tangent space of this manifold at $\mathbf{W}$. In general we have $N_{\text{eff}}(\mathbf{W}) \leq N$. Several reasons can make $N_{\text{eff}}(\mathbf{W})$ strictly smaller than N, including:

- The signal does not propagate in the network, i.e. $f(\mathbf{x}; \mathbf{W}) = C_1$ for all $\mathbf{x}$ in the neighborhood of the training points $\mathbf{x}_\mu$. In that case, the manifold is of dimension unity and $N_{\text{eff}}(\mathbf{W}) = 1$. This situation will occur for a poor initialization of the weights, for example if all biases are too negative on the neurons of one layer for ReLU activation function (see for instance [34]). It can also occur if the data $\mathbf{x}_\mu$ are chosen in an adversarial manner for a given choice of initial weights. For example, one can choose input patterns so as to not activate the first layer of neurons (which is possible if the number of such neurons is not too large). Poor transmission will be enhanced (and adversarial choices of data will be made simpler) if the architecture presents some bottlenecks. In the situation where $N_{\text{eff}}(\mathbf{W}) = 1$, it is very simple to obtain local minima of the loss at finite loss values, even when the model has many parameters.
- The activation function is linear. Then, the output is an affine function of the input, leading to $N_{\text{eff}} \leq d + 1$. Dimension-dependent bounds will also exist if the activation function is polynomial (because the output function then is also restricted to be polynomial).
- Symmetries are present in the network, e.g. the scale symmetry in ReLU networks: since the ReLU function is homogeneous, multiplying the weights of a layer by some factor and dividing the weights in the next layer by the same factor leaves the output function invariant. It will reduce one degrees of freedom per node.
- Some neurons are never active e.g. in the ReLU case, their associated weights do not contribute to N_{eff}.

Thus there are $N - N_{\text{eff}}$ directions in parameter space that do not affect the function. These directions will lead to zero modes in the Hessian at any minimum of the loss. In what follows we consider stability with respect to the relevant N_{eff} directions, which do affect the output function. Our results on the impossibility to get stuck in bad minima are expressed in terms of N_{eff}. However, as reported in Appendix A.1, we find empirically that for a proper initialization of the weights and constant-width fully connected networks, $N_{\text{eff}} \approx N$ (the difference is small and equal to the number of hidden neurons, and only results from the symmetry associated with each ReLU neuron). Henceforth to simplify notations we will use the symbol N to represent the number of effective parameters.

2.3. Constraints on the stability of minima

In this section we show that the existence of a minimum at a vanishingly small training loss (i.e. approaching jamming) is enough to derive an upper bound for the transition

in the (N, P) plane.

Let us suppose (and justify later) that, for a fixed number of data P and with proper initialization of weights, if N is large enough then gradient descent leads to $\mathcal{L} = 0$, whereas if N is small after training $\mathcal{L} > 0$. Imagine increasing N starting from a small value: at some N^* the loss obtained after training approaches zero §, i.e. $\lim_{N \to N^*} \mathcal{L} = 0$. We refer to this point as the jamming transition. A vanishing training loss implies that $\Delta_\mu \to 0$ for each pattern $\mu = 1, \dots, P$. As argued in [35], for each $\mu \in m$ the constraint $\Delta_\mu \approx 0$ defines a manifold of dimension $N - 1$‖. Satisfying N_Δ such equations thus generically leads to a manifold of solutions of dimension $N - N_\Delta$¶. Imposing that a solution exists implies that at jamming:

$$N^* \geq N_\Delta. \tag{5}$$

Smooth activation function: An opposite bound can be obtained by considerations of stability (as was done for the jamming of repulsive spheres in [37]), by imposing that in a stable minimum the Hessian must be positive definite if the output function is smooth, as it must be the case if the activation function is smooth (see below for the situation where the function displays cusps, as occurs for ReLU neurons). The Hessian matrix, that is the matrix of second derivatives, is

$$\begin{aligned} \mathcal{H}_\mathcal{L} &= \frac{1}{P} \sum_{\mu \in m} \nabla \Delta_\mu \otimes \nabla \Delta_\mu + \frac{1}{P} \sum_{\mu \in m} \Delta_\mu \nabla \otimes \nabla \Delta_\mu \\ &\equiv \mathcal{H}_0 + \mathcal{H}_p. \end{aligned} \tag{6}$$

(Here ∇ is the gradient operator and $\otimes$ stands for tensor product). The first term $\mathcal{H}_0$ is positive semi-definite: it is the sum of N_Δ rank-one matrices, thus $\mathrm{rk}(\mathcal{H}_0) \leq N_\Delta$, implying that the kernel of $\mathcal{H}_0$ is at least of dimension $N - N_\Delta$.

Let us denote by E_- the negative eigenspace[+] of $\mathcal{H}_p$ and call N_- its dimension. Stability then imposes that $\ker(\mathcal{H}_0) \cap E_- = \{0\}$, which is only possible if $N_\Delta \geq N_-$. Hence, minima with positive training loss (and therefore the minimum found right above jamming) can only occur for:

$$P \geq N_\Delta \geq N_-. \tag{7}$$

(The first inequality trivially follows from the fact that the N_Δ patterns belong to the training set of size P). As reported in [17], we observe empirically that the spectrum of $\mathcal{H}_p$ is statistically symmetric in the cases that we consider in the present work, i.e. for ReLU activation function, both for MNIST and random data, both at initialization and at the end of training. In Appendix A.2 we provide a non-rigorous argument supporting that in the case of ReLU activation functions and random data the spectrum of $\mathcal{H}_p$ is indeed symmetric with $\lim_{N \to \infty} N_-/N_+ = 1$ independently of depth, where N_+ is the number of positive eigenvalue. We conjecture that in general the limiting spectrum of H_p as $N, P \to \infty$ (for any fixed ratio P/N) has a finite fraction $C_0 = N_-/N$ of negative eigenvalues for generic architectures and datasets. In [17] we observed $C_0 = 1/2$ for

§ For finite P, N^* will present fluctuations induced by differences of initial conditions. The fluctuations of P/N^* are however expected to vanish in the limit where P and N^* become large. This phenomenon is well-known for the jamming of particles, and is an instance of finite size effects.

‖ Related arguments were recently made for a quadratic loss [11]. In that case, we expect the landscape to be related to that of floppy spring networks, whose spectra are predicted in [36].

¶ Note that this argument implicitly assumes that the N_Δ constraints are independent. In disordered systems this assumption is generally correct, but it may break down if symmetries are present.

[+] The negative eigenspace is the subspace spanned by the eigenvectors associated with negative eigenvalues.

the ReLU activation function as expected, for tanh activation functions at jamming and at the end of training we found $C_0 \approx 0.43$. Thus, C_0 is not universal.

Finally we assume that the spectrum of H_p does not display a finite density of zero eigenvalues (once restricted to the space of parameters that affect the output, of dimension N_{eff}, supposed here to be equal to N). Note that this assumption breaks down if data can be identical with different labels, a case we exclude here *. Under this assumption we obtain from Equation (7) that local minima with positive training loss cannot be encountered if $N > P/C_0$, implying in particular that:

$$N^* \leq P/C_0 \tag{8}$$

Non-smooth activation functions: With ReLU activation functions, the output function $f(\mathbf{x}; \mathbf{W})$ is not smooth and presents cusps, so that the Hessian needs not be positive definite for stability. A minimum can lie on a point where the second derivative is not defined along some directions (because of the cusp), and we say that the cusp stabilizes those directions. Equation (7) needs to be modified accordingly: introducing the number of directions $N_c \equiv \beta N$ presenting cusps near jamming, stability implies $N_\Delta > N_- - N_c$ and:

$$N_\Delta \geq N(C_0 - \beta) \tag{9}$$

implying in turn that:

$$N^*(P) \leq \frac{P}{C_0 - \beta} \tag{10}$$

Numerically, we find that at jamming the fraction of directions along which there is a cusp is $N_c/N^* \equiv \beta \in (0.21, 0.25)$ both for random data and images as reported in the Appendix A.3. Using $C_0 = 1/2$ for Relu, we obtain the bounds:

$$N_\Delta \geq N/4 \tag{11}$$

$$N^* \leq 4P. \tag{12}$$

Main results: Overall, our analysis supports that for smooth activation functions there exists a constant C_0 such that:

- there is a transition for $N^*(P) \leq P/C_0$ below which the training loss converges to some non-zero value (under-parametrized phase) and above which it becomes null (over-parametrized phase).
- At the transition, the fraction N_Δ/N of unsatisfied constraints per degree of freedom jumps discontinuously to a finite value satisfying $C_0 \leq N_\Delta/N \leq 1$.

The complete list of results, including consequences of this analysis on the Hessian, is included in [17].

For *Relu* activation function, $C_0 = 1/2$ but the analysis is complicated by the presence of cusps. The jamming transition is still sharp, i.e. characterized by a discontinuous jump in constraints as specified by Eq.11.

In the next sections, we confirm these predictions for *Relu* in numerical experiments and observe the generalization properties at and beyond the transition point.

* Indeed even in the over-parametrized case, if $\mathbf{x}_i = \mathbf{x}_j$ but $y_i \neq y_j$ then an exact cancellation of terms occurs in the sum defining H_p, which can then be zero while $\mathcal{L} > 0$.

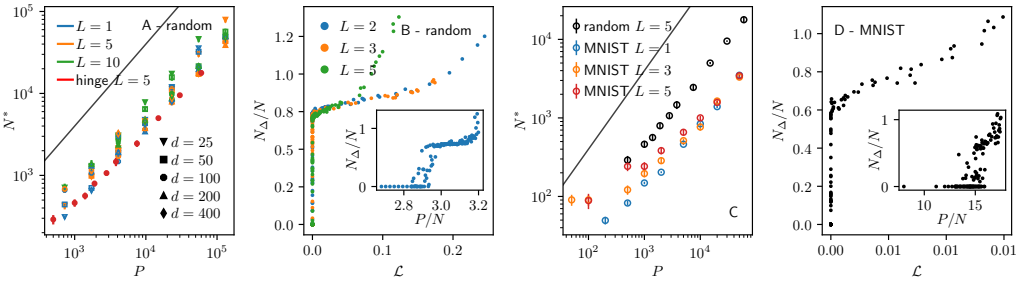


Figure 3. (A, B) Random data and (C, D) MNIST dataset. (A) and (C) depict the location N^* of the transition as a function of the number P, for networks with different cost functions or sizes. (B) and (D) show that in the $\mathcal{L}$-N_Δ/N and P/N-N_Δ/N planes the transition displays a discontinuous jump.

3. Location of the jamming transition

Here we present the numerical results on random data (uniformly distributed on a hypersphere and with random labels $y_\mu = \pm 1$) and on the MNIST dataset (partitioned into two groups according to the parity of the digits and with labels $y_\mu = \pm 1$). With MNIST, in order not to have most of the weights in the first layer, we reduce the actual input size by retaining only the first $d = 10$ principal components that carry the most variance (this hardly diminishes the performance for such a task). Further description of the protocols is in Appendix B.

In Fig. 3A,C we show the location of boundary N^* versus the number of samples P. N^* is estimated numerically for each P by starting from a large value of N and progressively decreasing it until $L > 0$ at the end of training. Varying input dimension, depth and loss function (cross entropy or hinge) has little effect on the transition. This result indicates that in the present setup the ability of fully-connected networks to fit random data does not depend crucially on depth. Fig. 3C shows also a comparison of random data with MNIST. A difference between random data and images is that the minimum number of parameters N^* needed to fit the real data is significantly smaller and grows less fast as P increases — for $P \gg 1$, $N^*(P)$ could be sub-linear or even tend to a finite asymptote: how the data structure affects $N^*(P)$ is an important questions for future studies.

From the analysis of Section 2, the number of constraints per parameter N_Δ/N is expected to jump discontinuously at the transition. This is shown in the insets of Fig. 3B,D. The scatter in these plots presumably reflects finite size effects known to occur near the jamming transition of particles [20]. All this scatter is however gone when plotting N_Δ/N as a function of the loss itself, as shown in the main panels of Fig. 3B,D.

4. Generalization at and beyond jamming

In Fig. 4A we show the evolution of the generalization error for networks at four different locations in the (N, P) plane. The networks are trained on MNIST at fixed $P = 50k$, and at different values N, both above, at and below jamming. Training is run for a fixed number of steps of vanilla gradient descent (the simulation details are in Appendix B). The profile of these curves is typical of most learning problems (if one does not recur to early stopping): notice that the point of minimum generalization error happens before the end of training. The increase of test error at late times is referred to "over-fitting" in the field. Very interestingly, it is clear from this figure that at small and large N, over-fitting is a weak effect, which however becomes very significant at intermediate N.

To study this effect, we systematically vary N at fixed P. In Fig. 4B the solid curve shows the generalization error against the network size N for three different values of P (we sampled subsets of MNIST). The dashed curve represents the value of the smallest error obtained during training, at prior time-steps (extracted from the profiles shown in Fig. 4A). The former displays a cusp at the transition point, as one can see clearly after rescaling the N-axis of each curve by the corresponding value of $N^\star(P)$. Strong over-fitting, corresponding to the difference between the solid and dashed lines, takes place only in the vicinity of the critical jamming transition (Fig. 4B-C). We thus posit that at fixed P, the benefit of early stopping [22] should diminish in the large-size limit. Beyond the jamming point, the accuracy keeps steadily improving as the number of parameters increases [25, 26, 27], although it does so quite slowly. We have provided a quantitative explanation for this phenomenon in [33]. In

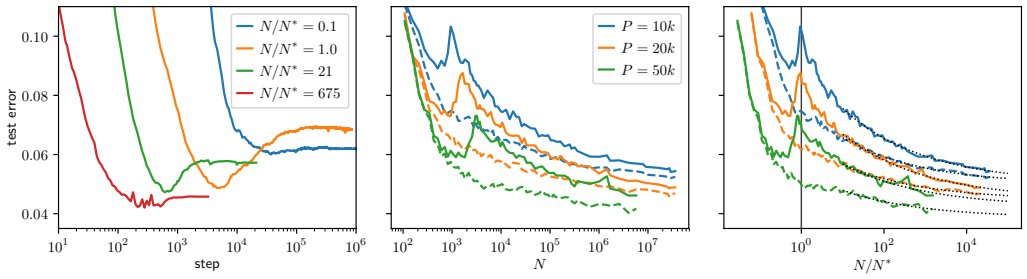


Figure 4. We trained a 5 hidden layer fully-connected network on MNIST. (A) Typical evolution of the generalization error over training time, for systems located at different points relatively to the jamming transition (for $P = 50k$): over-fitting is marked by the gap between the value at the end of training and the minimum at prior times. Notice that training of over-parametrized systems halts sooner because the networks have achieved zero loss over the training set. (B) Test error at the final point of training (solid line) and minimum error achieved during training (dashed line) vs. system size. (C) When N is scaled by $N^*(P)$ it is clear that over-fitting occurs at the jamming transition.

.

Appendix B.2 we have verified that the overall trends showed in Fig. 4 qualitatively hold also for other depths.

Notice that although the cusp has been found also in shallow networks (in particular the perceptron [38, 39]), their behavior is at odds with what we observe: for the perceptron, test error asymptotically *increases* with N.

5. Conclusions

Understanding the effect of over-parametrization on the behavior of deep neural networks is a central problem in machine learning. In this work, by focusing on the hinge loss, we recast the minimization of the loss function as a constraint-satisfaction problem with continuous degrees of freedom. A similar approach was used in the field of interacting particles, which display a sharp jamming transition affecting the landscape if the interaction is chosen to be finite range [20]. Following the analogy we were able to predict a sharp transition as the number of network parameters is varied, separating a region in the (P, N) plane where a global minimum can be found ($\mathcal{L} = 0$) from a region where the number of unsatisfied constraints is a fraction of the number of parameters, so $\mathcal{L} > 0$. These results also shed light on several aspects of deep learning:

Not getting stuck in local minima: In the over-parametrized regime, the dynamics does not get stuck in local minima at finite loss value because the number of constraints to satisfy is too small to hamper minimization. It follows from our assumptions on the negative eigenspace of the matrix $\mathcal{H}_p$ that in this regime the landscape is flat and local minima do not exist (assuming that the number of effective parameters that affect the output function is N). For a smooth activation function we predict that one cannot get stuck in a bad minimum for $N \geq P/C_0$, implying in particular that $N^* \leq P/C_0$ where C_0 is a constant. We obtain a less demanding bound for ReLU activation functions due to the presence of cusps in the landscape, a situation for which we expect $C_0 = 1/2$. In practice, for random data $N^*(P)$ scale linearly with P (in this sense, the bound is tight). By contrast, for structured data $N^*(P)$ appears to scale sub-linearly with P. Predicting the curve $N^*(P)$ remains a challenge for the future.

Reference point for fitting and generalization: There exists a critical curve $N^*(P)$ on the N-P plane above which the global minima of the landscape become accessible. The curve also appears to be linked to the generalization potential of the model. We show that in the cases that we considered, *(i)* the generalization error decreases when $N \ll N^*$; then *(ii)* it increases and culminates in a cusp at $N \approx N^*$ that is erased by early stopping, most useful in this region; finally, *(iii)* in the over-parametrized phase, it monotonically decreases, although very slowly.

Acknowledgments We thank Marco Baity-Jesi, Carolina Brito, Chiara Cammarota, Taco S. Cohen, Silvio Franz, Yann LeCun, Florent Krzakala, Riccardo Ravasio, Andrew Saxe, Pierfrancesco Urbani and Lenka Zdeborova for helpful discussions. This work was partially supported by the grant from the Simons Foundation (#454935 Giulio Biroli, #454953 Matthieu Wyart). M.W. thanks the Swiss National Science Foundation for support under Grant No. 200021-165509. The manuscript [40], which appeared at the same time as ours, shows that the critical properties of the jamming transition found for the non-convex perceptron [41] hold more generally in some shallow networks. This universality is an intriguing result. Understanding the connection with our findingsis certainly worth future studies.

[1] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.

[4] Ludovic Berthier and Giulio Biroli. Theoretical perspective on the glass transition and amorphous materials. *Reviews of Modern Physics*, 83(2):587, 2011.

[5] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 2933–2941, 2014.

[6] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[7] C Daniel Freeman and Joan Bruna. Topology and geometry of deep rectified network optimization landscapes. *International Conference on Learning Representations*, 2017.

[8] Luca Venturi, Afonso Bandeira, and Joan Bruna. Neural networks with finite intrinsic dimension have no spurious valleys. *arXiv preprint arXiv:1802.06384*, 2018.

[9] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pages 1729–1739, 2017.

[10] Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

[11] Yaim Cooper. The loss landscape of overparameterized neural networks. *arXiv preprint arXiv:1804.10200*, 2018.

[12] Levent Sagun, Léon Bottou, and Yann LeCun. Singularity of the hessian in deep learning. *International Conference on Learning Representations*, 2017.

[13] Levent Sagun, Utku Evci, V. Uğur Güney, Yann Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *ICLR 2018 Workshop Contribution, arXiv:1706.04454*, 2017.

[14] Andrew J Ballard, Ritankar Das, Stefano Martiniani, Dhagash Mehta, Levent Sagun, Jacob D Stevenson, and David J Wales. Energy landscapes for machine learning. *Physical Chemistry Chemical Physics*, 2017.

[15] Zachary C Lipton. Stuck in a what? adventures in weight space. *International Conference on Learning Representations*, 2016.

[16] Marco Baity-Jesi, Levent Sagun, Mario Geiger, Stefano Spigler, Gerard Ben Arous, Chiara Cammarota, Yann LeCun, Matthieu Wyart, and Giulio Biroli. Comparing dynamics: Deep neural networks versus glassy systems. In *Proceedings of the 35th International Conference on Machine Learning*, pages 314–323, 2018.

[17] Mario Geiger, Stefano Spigler, Stéphane d'Ascoli, Levent Sagun, Marco Baity-Jesi, Giulio Biroli, and Matthieu Wyart. The jamming transition as a paradigm to understand the loss landscape of deep neural networks. *arXiv preprint arXiv:1809.09349*, 2018.

[18] Silvio Franz, Giorgio Parisi, Pierfrancesco Urbani, and Francesco Zamponi. Universal spectrum of normal modes in low-temperature glasses. *Proceedings of the National Academy of Sciences*, 112(47):14539–14544, 2015.

[19] M. Wyart. On the rigidity of amorphous solids. *Annales de Phys*, 30(3):1–113, 2005.

[20] Andrea J Liu, Sidney R Nagel, W Saarloos, and Matthieu Wyart. *The jamming scenario - an introduction and outlook*. OUP Oxford, 06 2010.

[21] Rich Caruana, Steve Lawrence, and C Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408, 2001.

[22] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[24] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[25] Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of optimization and implicit regularization in deep learning. *arXiv preprint arXiv:1705.03071*, 2017.
[26] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.
[27] Yamini Bansal, Madhu Advani, David D Cox, and Andrew M Saxe. Minnorm training: an algorithm for training over-parameterized deep neural networks. *CoRR*, 2018.
[28] Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.
[29] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118*, 2018.
[30] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.
[31] Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *arXiv preprint arXiv:1903.07571*, 2019.
[32] Zhenyu Liao and Romain Couillet. The dynamics of learning: A random matrix approach. *arXiv preprint arXiv:1805.11917*, 2018.
[33] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *arXiv preprint arXiv:1901.01608*, 2019.
[34] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.
[35] Alexei V. Tkachenko and Thomas A. Witten. Stress propagation through frictionless granular material. *Phys. Rev. E*, 60(1):687–696, Jul 1999.
[36] Gustavo Düring, Edan Lerner, and Matthieu Wyart. Phonon gap and localization lengths in floppy materials. *Soft Matter*, 9(1):146–154, 2013.
[37] Matthieu Wyart, Leonardo E Silbert, Sidney R Nagel, and Thomas A Witten. Effects of compression on the vibrational modes of marginally jammed solids. *Physical Review E*, 72(5):051306, 2005.
[38] David Saad and Sara A Solla. On-line learning in soft committee machines. *Physical Review E*, 52(4):4225, 1995.
[39] Andreas Engel and Christian Van den Broeck. *Statistical mechanics of learning*. Cambridge University Press, 2001.
[40] P. Urbani S. Franz, S. Hwang. Jamming in multilayer supervised learning models. *arXiv preprint arXiv:1809.09945*, 2018.
[41] Silvio Franz and Giorgio Parisi. The simplest model of jamming. *Journal of Physics A: Mathematical and Theoretical*, 49(14):145001, 2016.
[42] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations*, 2014.
[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Appendix A. Network properties

In the following, we analyze numerically the networks properties that were used in the previous analysis. This provides a numerical confirmation of our arguments, and an in depth characterization of the networks.

Appendix A.1. Effective number of degrees of freedom

Due to several effects discussed above, the function $f(\mathbf{x};\mathbf{W})$ can effectively depend on less variables that the number of parameters, and thus reduce the dimension of the space spanned by the gradients $\nabla_{\mathbf{W}} f(\mathbf{x};\mathbf{W})$ that enters in the theory. For instance, there could be symmetries that reduce the number of effective degrees of freedom (e.g. each ReLU activation function has one of such symmetries, since one can rescale inputs and outputs in such a way that the post-activation is left invariant); another

reason could be that a neuron might never activate for all the training data, thus effectively reducing the number of neurons in the network; furthermore, we expect that the network's true dimension would also be reduced if its architecture presents some bottlenecks, is poorly designed or poorly initialized. For example if all biases are too negative on the neurons of one layer in the Relu case, the network does not transmit any signals, leading to $N = 1$ and to the possible absence of unstable directions even if the number of parameters is very large.

It is tempting to define the effective dimension by considering the dimension of the space spanned by $\nabla_{\mathbf{W}} f(\mathbf{x}_\mu; \mathbf{W})$ as μ varies. This definition is not practical for small number of samples P however, because this dimension would be bounded by P. We can overcome such a problem by considering a neighborhood of each point $\mathbf{x}_\mu$, where the network's function and its gradient can be expanded in the pattern space:

$$f(\mathbf{x}) \approx f(\mathbf{x}_\mu) + (\mathbf{x} - \mathbf{x}_\mu) \cdot \nabla_{\mathbf{x}} f(\mathbf{x}_\mu), \tag{A.1}$$

$$\nabla_{\mathbf{W}} f(\mathbf{x}) \approx \nabla_{\mathbf{W}} f(\mathbf{x}_\mu) + (\mathbf{x} - \mathbf{x}_\mu) \cdot \nabla_{\mathbf{x}} \nabla_{\mathbf{W}} f(\mathbf{x}_\mu). \tag{A.2}$$

Varying the pattern μ and the point $\mathbf{x}$ in the neighborhood of $\mathbf{x}_\mu$, we can build a family M of vectors:

$$M = \left\{ \nabla_{\mathbf{W}} f(\mathbf{x}_\mu) + (\mathbf{x} - \mathbf{x}_\mu) \cdot \nabla_{\mathbf{x}} \nabla_{\mathbf{W}} f(\mathbf{x}_\mu) \right\}_{\mu, \mathbf{x}}. \tag{A.3}$$

We then define the effective dimension N_{eff} as the dimension of M. Because of the linear structure of M, it is sufficient to consider, for each μ, only $d+1$ values for x, e.g. $x - x_\mu = 0, \hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_d$, where $\hat{\mathbf{e}}_n$ is the unit vector along the direction n. The effective dimension is therefore

$$N_{\text{eff}} = \text{rk}(G), \tag{A.4}$$

where the elements of the matrix G are defined as

$$G_{i,\alpha} \equiv \partial_{W_i} f(\mathbf{x}_\mu) + \hat{\mathbf{e}}_n \cdot \nabla_{\hat{\mathbf{e}}_n} \partial_{W_i} f(\mathbf{x}_\mu), \tag{A.5}$$

with $\alpha \equiv (\mu, n)$. The index n ranges from 0 to d, and $\hat{\mathbf{e}}_0 \equiv 0$.

In Fig. A1 we show the effective number of parameters N_{eff} versus the total number of parameters N, in the case of a network with $L = 3$ layers trained on the first 10 PCA components of the MNIST dataset. There is no noticeable difference between the two quantities: the only reduction is due to the symmetries induced by the ReLU functions (there is one such symmetry per neuron. Indeed the ReLU function $\rho(z) = z\Theta(z)$ satisfies $\Lambda\rho(z/\Lambda) \equiv \rho(z)$.) We observed the same results for random data.

Appendix A.2. $\text{sp}(H_p)$ *is symmetric for ReLu activation functions and random data*

We consider $\mathcal{H}_p = -\sum_\mu y_\mu \rho\left(\Delta_\mu\right) \hat{\mathcal{H}}_\mu$, where $\hat{\mathcal{H}}_\mu$ is the Hessian of the network function $f(\mathbf{x}_\mu; \mathbf{W})$ and ρ is the Relu function. We want to argue that the spectrum of $\mathcal{H}_p$ is symmetric in the limit of large N.

We do two main hypothesis: First, the trace of any finite power of $\mathcal{H}_p$ is self-averaging (concentrates) with respect to the average over the random data:

$$\frac{1}{N}\text{tr}(\hat{\mathcal{H}_p}^n) = \frac{1}{N}\overline{\text{tr}(\hat{\mathcal{H}_p}^n)}. \tag{A.6}$$

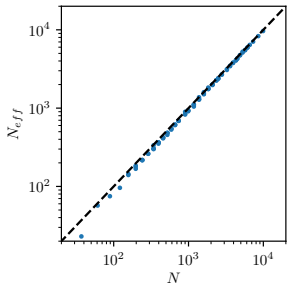


Figure A1. Results with the MNIST dataset, keeping the first 10 PCA components. $d = 10$ and $L = 3$, varying P and h. Effective N_{eff} vs total number of parameters N. N_{eff} is always smaller than N because there is a symmetry per each ReLU-neuron in the network.

Second,

$$\begin{aligned}&\frac{1}{N}\sum_{\mu_1,\cdots,\mu_n}\overline{y_{\mu_1}\rho(\Delta_{\mu_1})\cdots y_{\mu_n}\rho(\Delta_{\mu_n})\text{tr}(\hat{\mathcal{H}}_{\mu_1}\cdots\hat{\mathcal{H}}_{\mu_n})}=\\&=\frac{1}{N}\sum_{\mu_1,\cdots,\mu_n}\overline{y_{\mu_1}\rho(\Delta_{\mu_1})\cdots y_{\mu_n}\rho(\Delta_{\mu_n})}\cdot\overline{\text{tr}(\hat{\mathcal{H}}_{\mu_1}\cdots\hat{\mathcal{H}}_{\mu_n})}\end{aligned}\tag{A.7}$$

The first hypothesis is natural since $\hat{\mathcal{H}}_p$ is a very large random matrix, for which the density of eigenvalues is expected to become a non-fluctuating quantity. The second hypothesis is more tricky: it is natural to assume that the trace concentrates, however one also need to show that the sub-leading corrections to the self-averaging of the trace can be neglected.

Using these two hypothesis and the result, showed below, that

$$\overline{\text{tr}(\hat{\mathcal{H}}_{\mu_1}\cdots\hat{\mathcal{H}}_{\mu_n})}=0\tag{A.8}$$

for all n odds, one can conclude that all odds traces of $\hat{\mathcal{H}}_p$ are zero. This implies that the spectrum of $\hat{\mathcal{H}}_p$ is symmetric, more precisely that the fractions of negative and positive eigenvalues are equal.

In order to show that the statement (A.8) above holds, let us argue first that $\overline{\text{tr}(\hat{\mathcal{H}}_\mu^n)}=0$ for any *odd* n.

$$\text{tr}(\hat{\mathcal{H}}_\mu^n)=\sum_{i_1,i_2,\ldots,i_n}\hat{\mathcal{H}}^\mu_{i_1,i_2}\hat{\mathcal{H}}^\mu_{i_2,i_3}\cdots\hat{\mathcal{H}}^\mu_{i_n,i_1},\tag{A.9}$$

where the indices $i_1,\ldots,i_n$ stand for synapses connecting a pair of neurons (i.e. each index is associated with a synaptic weight $W^{(j)}_{\alpha,\beta}$: we are not writing all the explicit indexes for the sake of clarity). The term of the hessian obtained when differentiating

with respect to weights $W^{(j)}_{\alpha,\beta}$ and $W^{(k)}_{\gamma,\delta}$ reads

$$\hat{\mathcal{H}}^{\mu;(jk)}_{\alpha\beta;\gamma\delta} = \sum_{\pi_0,\dots,\pi_L} \theta(a^{\mu}_{L,\pi_L})\cdots\theta(a^{\mu}_{1,\pi_1})x^{\mu}_{\pi_0}\cdot \\ \cdot\partial_{W^{(j)}_{\alpha,\beta}}\partial_{W^{(k)}_{\gamma,\delta}}\left[W^{(L+1)}_{\pi_L}W^{(L)}_{\pi_L,\pi_{L-1}}\cdots W^{(1)}_{\pi_1\pi_0}\right]. \tag{A.10}$$

where we denoted with a the inputs in the nodes of the network. Our argument is based on a symmetry of the problem with random data: changing the sign of the weight of the last layer $W^{(L+1)} \longrightarrow -W^{(L+1)}$ and changing the labels $y_\mu \longrightarrow -y_\mu$ leaves the loss unchanged. We will show that this symmetry implies that $\mathrm{tr}(\hat{\mathcal{H}}^n_\mu)$ averaged over the random labels is zero for odd n.

In fact, note that the sum in Equation (A.10) contains a weight per each layer in the network, with the exception of the two layers j,k with respect to which we are deriving. This implies that any element of the hessian matrix where we have not differentiated with respect to the last layer ($j,k < L+1$) is an odd function of the last layer $W^{(L+1)}$, meaning that if $W^{(L+1)} \longrightarrow -W^{(L+1)}$, then the sign of all these Hessian elements is inverted as well.

If in the argument of the sum in Equation (A.9) there is no index belonging to the last layer, then the whole term changes sign under the transformation $W^{(L+1)} \longrightarrow -W^{(L+1)}$. Suppose now that, on the contrary, there are m terms with one index belonging to the last layer (we need not consider the case of two indices both belonging to the last layer because the corresponding term in the Hessian would be 0, as one can see in Equation (A.10)). For each index equal to $L+1$ (the last layer), there are exactly two terms: $\hat{\mathcal{H}}^{\mu}_{j,L+1}\hat{\mathcal{H}}^{\mu}_{L+1,k}$ (for some indexes j,k). Since j,k cannot be $L+1$ too, this implies that the number m of terms with an index belonging to the last layer is always even. Consequently, when the sign of $W^{(L+1)}$ is reversed, the argument of the sum in Equation (A.9) is multiplied by $(-1)^{n-m}$ (once for each term *without* an index belonging to the last layer), which is equal to -1 if n is odd. The same symmetry can be used to show that a matrix made of an odd product of matrices $\hat{\mathcal{H}}_\mu$, such as $\hat{\mathcal{H}}_\mu\hat{\mathcal{H}}_{\mu'}\hat{\mathcal{H}}_{\mu''}$, must also have a symmetric spectrum, concluding our argument.

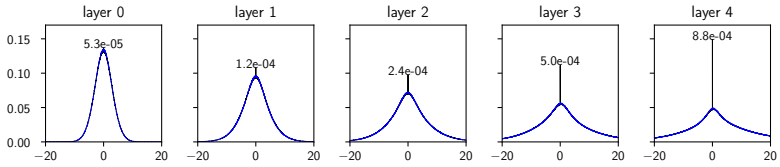


Figure A2. Density of the pre-activations for each layers with $L = 5$ and random data, averaged over all the runs just above the jamming transition with that architecture. Black: distribution obtained over the training set. Blue: previously unseen random data (the two curves are on top of each other except for the delta in zero). The values indicate the mass of the peak in zero, which is only present when the training set is considered.

Appendix A.3. Density of pre-activations for ReLU activation functions

The densities of pre-activation (i.e. the value of the neurons before applying the activation function) is shown in Fig. A2 for random data. It contains a delta distribution in zero. The number N_c of pre-activations equal to zero when feeding a network $L = 5$ all its random dataset is $N_c \approx 0.21N$, corresponding to the number of directions in phase space where cusps are present in the loss function. For MNIST data we find $N_c \approx 0.19N$. By taking $L = 2$ and random data we find $N_c \approx 0.25N$. In these directions, stability can be achieved even if the hessian would indicate an instability. For this reason, instead of N_- in Equation (7) one should use $N/2 - N_c \approx 0.25N$.

Appendix B. Parameters used in simulations

Appendix B.1. Random data

The dataset is composed of P points taken to lie on the d-dimensional hyper-sphere of radius $\sqrt{d}$, $\mathbf{x}_\mu \in \mathcal{S}^d$, with random label $y_\mu = \pm 1$. The networks are fully connected, and have an input layer of size d and L layers with h neurons each, culminating in a final layer of size 1. To find the transition we proceed as follows: we build a network with a number of parameters N large enough for it to be able to fit the whole dataset without errors. Next, we decrease the width h while keeping the depth L fixed, until the network cannot correctly classify all the data anymore within the chosen learning time. We denote this transition point N^*. As initial conditions for the dynamics we use the default initialization of `pytorch`: weights and biases are initialized with a uniform distribution on $[-\sigma, \sigma]$, where $\sigma^2 = 1/f_{in}$ and f_{in} is the number of incoming connections.

When using the cross entropy, the system evolves according to a stochastic gradient descent (SGD) with a learning rate of 10^{-2} for $5 \cdot 10^5$ steps and 10^{-3} for $5 \cdot 10^5$ steps (10^6 steps in total); the batch size is set to $\min(P/2, 1024)$, and batch normalization is used. We do not use any explicit regularization in training the networks. In Fig. B1 we check that $t = 10^6$ is enough to converge.

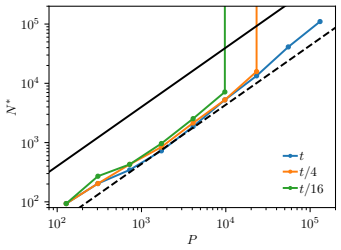


Figure B1. Convergence of the critical line for networks trained with cross entropy on random data.

When using the hinge loss, we use an orthogonal initialization [42], no batch normalization and $t = 2 \cdot 10^6$ steps of ADAM [43] with batch size P and a learning rate starting at 10^{-4}. In the experiments of section 3 (not for the experiments of section 4),

we progressively divided the learning rate by 10 every 250k steps. Also in this case we do not use any explicit regularization in training the networks.

To observe the discontinuous jump in the number N_Δ of unsatisfied constraints at the transition (Fig. 3B and inset), we consider three architectures, both with $N \approx 8000$ and $d = h$ but with different depths $L = 2$, $L = 3$ and $L = 5$. The vicinity of the transition is studied by varying P around the transition value and minimizing for 10^7 steps (a better minimization is needed to improve the precision close to the transition).

Details about Fig 3A hinge We took $d = h$ and trained for 2M steps. For some values of $P \in (500, 60\text{k})$, start at large h where we reach $N_\Delta = 0$ and decrease h until $N_\Delta > 0.1N$.

Details about Fig 3B We trained networks of depth 2,3,5 with $d = h = 62$, 51, 40 respectively for 10M steps. For $L = 3$ ($d = 51$, $h = 51$) we ran 128 training varying P from 21991 to 25918. For the value of N we take 7854 that correspond to the number of parameters minus the number of neurons, per neuron there is a degree of freedom lost in a symmetry induced by the homogeneity of the ReLU function. 37 of the runs have $N_\Delta = 0$, 74 have $N_\Delta > 0.4N$. Among the 19 remaining ones, 14 of them have N_Δ between 1 and 4, we think that these runs encounter numerical precision issues, we observed that using 32 bit precision accentuate this issue. We think that the 5 left with $4 < N_\Delta < 0.4N$ has been stoped too early. The same observation apply for the other depths.

Appendix B.2. Real data

The images in the MNIST dataset are gathered into two groups, with even and odd numbers and with labels $y_\mu = \pm 1$. The architecture of the network is as in the previous sections: the d inputs are fed to a cascade of L fully-connected layers with h neurons each, that in the end result in a single scalar output. The loss function used is always the hinge loss.

If we kept the original input size of $28 \times 28 = 784$ (each picture is 28×28 pixels) then the majority of the network's weights would be necessarily concentrated in the first layer (the width h cannot be too large in order to be able to compute the Hessian). To avoid this issue, we opt for a reduction of the input size. We perform a principal component analysis (PCA) on the whole dataset and we identify the 10 dimensions that carry the most variance on the whole dataset; then we use the components of each image along these directions as a new input of dimension $d = 10$. This projection hardly diminishes the performance of the network (which we find to be larger than 90% when using all the data and large N).

Details about Fig 3C We trained networks of depth 1,3,5 for 2M steps. For some values of $P \in (100, 50\text{k})$, start at large h where we reach $N_\Delta = 0$ and decrease h until $N_\Delta > 0.1N$.

Details about Fig 3D We trained a network of $L = 5$, $d = 10$, $h = 30$ for 3M steps. With P varying from 31k to 68k (using trainset and testset of MNIST).

Details about Fig 4 We trained a network of $L = 5$ and $d = 10$ for 500k steps. where $P \in \{10\text{k}, 20\text{k}, 50\text{k}\}$ and h varies from 1 to 3k. Fig B2 shows a comparison between $L = 5$ and $L = 2$.

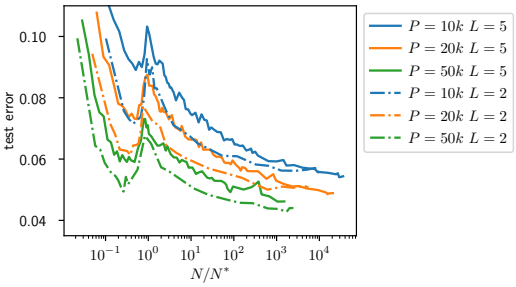


Figure B2. Generalization on MNIST 10 PCA. Comparison between two depth $L = 2$ and $L = 5$.

The following paper is the preprint version of Geiger et al. (2020a).

Candidate contributions The candidate participated in the scientific discussions and performed the numerical experiments (except the data of figure 8). The candidate developed tools to manage storage and analysis of large number of experiments.

Scaling description of generalization with number of parameters in deep learning

Mario Geiger[a,1], Arthur Jacot[b,1], Stefano Spigler[a], Franck Gabriel[b], Levent Sagun[a], Stéphane d'Ascoli[c], Giulio Biroli[c], Clément Hongler[b,2], and Matthieu Wyart[a,2]

[a]Institute of Physics, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland
[b]Institute of Mathematics, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland
[c]Laboratoire de Physique Statistique, École Normale Supérieure, PSL Research University, 75005 Paris, France

October 9, 2019

Abstract

Supervised deep learning involves the training of neural networks with a large number N of parameters. For large enough N, in the so-called over-parametrized regime, one can essentially fit the training data points. Sparsity-based arguments would suggest that the generalization error increases as N grows past a certain threshold N^*. Instead, empirical studies have shown that in the over-parametrized regime, generalization error keeps decreasing with N. We resolve this paradox through a new framework. We rely on the so-called Neural Tangent Kernel, which connects large neural nets to kernel methods, to show that the initialization causes finite-size random fluctuations $\|f_N - \bar{f}_N\| \sim N^{-1/4}$ of the neural net output function f_N around its expectation $\bar{f}_N$. These affect the generalization error ϵ_N for classification: under natural assumptions, it decays to a plateau value ϵ_∞ in a power-law fashion $\sim N^{-1/2}$. This description breaks down at a so-called jamming transition $N = N^*$. At this threshold, we argue that $\|f_N\|$ diverges. This result leads to a plausible explanation for the cusp in test error known to occur at N^*. Our results are confirmed by extensive empirical observations on the MNIST and CIFAR image datasets. Our analysis finally suggests that, given a computational envelope, the smallest generalization error is obtained using several networks of intermediate sizes, just beyond N^*, and averaging their outputs.

Introduction

Deep neural networks (DNNs) have proven to be very successful at a very wide range of tasks. In particular, for supervised learning tasks, they have yielded breakthroughs in various contexts, in particular for image classification [1, 2], speech recognition [3], and automatic translation [4]. Yet, a theoretical framework to understand the remarkable successes of DNNs remains to be constructed, and central questions need to be clarified.

First, supervised learning for a DNN corresponds to adjusting N parameters which describe an *output function* $f_N : \mathbb{R}^{n_{\text{in}}} \to \mathbb{R}^{n_{\text{out}}}$ to fit P training data points $(x_i, y_i)_{i=1,\dots,P}$ with $x_i \in \mathbb{R}^{n_{\text{in}}}, y_i \in \mathbb{R}^{n_{\text{out}}}$. In practice, it is done by initializing the parameters randomly and minimizing a (non-convex) loss function using a first-order method (e.g. gradient descent). The dynamics of the training of DNNs, and the question of whether a global minimum is attained are thus a priori delicate, involving the understanding of a complex loss landscape.

Second, DNNs are in practice trained in the so-called over-parametrized regime, where the number of parameters N is much larger than the number of data points P. Thus, DNNs are used in a regime where their capacity is very large (they can still classify the data even if all their labels are randomized). Surprisingly from the point of view of traditional statistical learning theory [5] DNNs generalize very well in practice, even without an explicit regularization. This thus raises the question of an appropriate framework to understand generalizations of DNNs.

Recent works suggest that the two questions above are closely connected. Numerical and theoretical studies [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17] show that in the over-parametrized regime, the loss landscape of DNNs is not rough with isolated minima as initially thought [18, 19], but instead has connected level sets and presents many flat directions, even near its global minimum. In particular, recent works on the over-parametrized regime of DNNs [20, 21, 22, 23] have shown that the landscape around a typical initialization point becomes essentially convex, allowing for convergence to a global minimum during training.

In [16, 17], it has been observed that when optimizing DNNs (using the so-called hinge loss), there is a sharp phase transition — whose location can depend on the chosen dynamics — at some $N^*(P)$ such that for $N \geq N^*$ the dynamic process reaches a global minimum of the loss. In particular whenever $N > N^*$, the *training error* (i.e. the total of the loss on the training set) reaches its global minimum. A counter-intuitive aspect of deep learning is that increasing N above N^* does not destroy the predictive power by over-fitting the data, but instead appears to improve the *generalization performance* (i.e. the probability that a data point outside of the training set

[1]M.G. and A.J. contributed equally to this work.
[2]E-mail: clement.hongler@epfl.ch, matthieu.wyart@epfl.ch

is correctly classified) [24, 25, 26, 27]. Indeed the test error (the probability of an incorrect classification for an unseen data point) has been observed to decrease as $N \to \infty$ in a slow power-law fashion [17]. In contrast, as $N \to N^*$, the test error blows up [27, 28, 17] (a phenomenon shown by the blue curve in Fig. 2). In the context of least-squares regression, the improvement of performance with N has been linked to the observed diminishing fluctuations of the DNN function after training [29], a result consistent with the notion of stronger implicit regularization with increasing N [30, 31]. This raises the question of understanding what controls these fluctuations and how they affect the test error in a classification task.

In this work, we address these questions in the context of classification tasks for fully-connected DNNs with a fixed number of layers $L \geq 2$, with wide hidden layers. We develop a framework based on a new connection between the $N \to \infty$ limit of DNNs and kernel methods [20]. More precisely, the training of DNNs can be recast as a kernel gradient descent associated with the so-called Neural Tangent Kernel (NTK). In the $N \to \infty$ limit, the NTK becomes deterministic and constant in time. This result explains why the generalization performance converges as $N \to \infty$, a result previously obtained for single hidden layer neural networks using a different approach [32, 33, 34, 35].

We consider a binary classification task; the DNN output function $f_N : \mathbb{R}^{n_{in}} \to \mathbb{R}$ is used to predict whether a data point belongs to the class ± 1 depending on the sign of f_N.

First, we introduce an NTK-based framework to study the random fluctuations of the output function f_N at the end of training due to the random initialization of the parameters. We find that (in the over-parametrized regime) the key finite-N effect is that the NTK at initialization has random fluctuations around its mean of order $N^{-1/4}$, leading to similar fluctuations for f_N.

Second, we consider the fluctuations of the decision boundary (the level set $\{f_N(x) = 0\}$): we argue that a variation δf_N of f_N yields an increase $\delta\epsilon \sim (\delta f_N)^2$ to the test error. We use this asymptotic result to predict the increase in generalization performance yielded by an ensemble averaging on n samples of the function f_N (each trained on the data separately) as n becomes large, as well as the increase in generalization performance as N grows.

Finally, this description breaks down at the transition point N^*, where the random fluctuations of f_N appear to diverge as a power law. We study this divergence through a simple argument on non-linear networks, suggesting that $\|f_N\| \sim (N - N^*)^{-1}$.

Overall, our work introduces a conceptual framework to describe how generalization error in deep learning evolves with the number of parameters. A practical consequence of our analysis is that performing an ensemble average of (both fully-connected and convolutional) DNNs with independent initializations can improve performance significantly: for a given computational envelope, it appears to be best to use several nets of intermediate sizes $N > N^*$ and to average their outputs.

Related works

After the electronic submission of the present work, and following on [17, 36], other articles have been written on the nature of the "double descent" curve in the generalization error (Fig. 2) [37, 38, 39] and on the asymptotic behavior of wide networks [40, 41, 42, 43]. Very recently in [38], a rigorous derivation of the double descent curve was obtained for the mean square regression of simple functions using random features models. Although the scaling arguments proposed here are not mathematical proofs, they provide a quantitative explanation of the double descent curve in a more general setting, including the regression and classification of empirical data by fully connected deep networks. Our predictions are tested empirically in that setting. Finally, our analysis is based on a scaling estimate of the fluctuations of the NTK at initialization, recently supported by more detailed analysis based on Feynman diagrams and path numbering [40, 41].

1 Setting

1.1 DNN Model and Training

We consider DNNs defining a real-valued output function $f_N(x;\theta)$ for $x \in \mathbb{R}^{n_{\text{in}}}$, where we aggregate the parameters into $\theta \in \mathbb{R}^N$. We first consider fully-connected DNNs of L layers, where each layer is made of h neurons, as in Fig. 1. The output function f_N is constructed recursively as

$$\begin{aligned} f_N(x;\theta) &\equiv a^{(L+1)}, \\ a_\beta^{(i)} &= \sum_\alpha W_{\alpha,\beta}^{(i)}\, \rho\left(a_\alpha^{(i-1)}\right) - B_\beta^{(i)}, \\ a_\beta^{(1)} &= \sum_\alpha W_{\alpha,\beta}^{(1)}\, x_\alpha - B_\beta^{(1)}. \end{aligned}$$

$W_{\alpha,\beta}^{(i)}$ is the weight of the synapse from neuron α in layer $(i-1)$ to neuron β in layer (i), and $B_\beta^{(i)}$ is the bias of neuron β in layer (i), as depicted in Fig. 1. The vector θ contains all weights and biases. $\rho : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function. Empirically we will use the standard ReLU $\rho(a) = \max(a, 0)$, but any other common nonlinear functions can be used (e.g. the softplus function). Polynomial functions must be avoided, as they do not lead to positive definite kernels, see discussion in [20].

The DNN function is used for binary classification: we aim to find θ such that for a data point x_μ, $\text{sign} f(x_\mu;\theta)$ correctly predicts the label $y_\mu \in \{\pm 1\}$. To do so, we minimize on a dataset $(x_\mu, y_\mu)_{\mu=1,\dots,P}$ the square-hinge cost function

$$C = \frac{1}{P}\sum_{\mu=1}^{P} \frac{1}{2}\max(0, \Delta_\mu)^2, \tag{1}$$

where $\Delta_\mu \equiv \epsilon_m - y_\mu f(x_\mu;\theta)$ and ϵ_m is the so-called margin, fixed to 1 in our numerical tests.

The network is then trained using a first-order method, such as gradient descent, for a maximum running time of t^*, and is stopped as soon as the training loss hits its lowest possible value (typically 0, unless two identical data points have different labels). The jamming transition point is defined

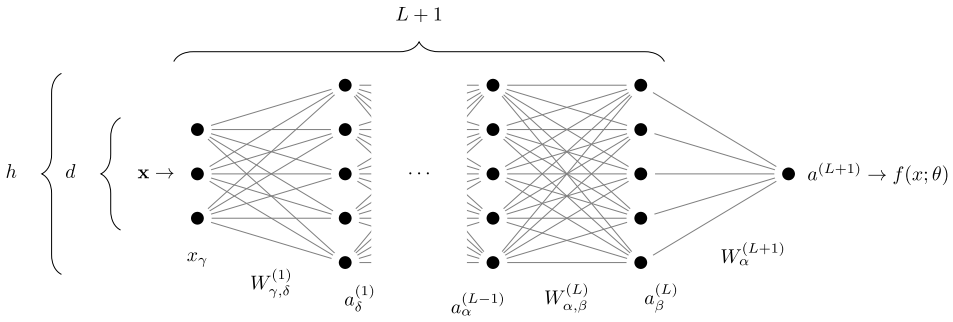


Figure 1: Architecture of a fully-connected network with L hidden layers of constant size h. Points indicate neurons, connections between them are weighted (biases are not represented here).

as the smallest value of N for which we reach the lowest possible loss at the end of training.

Note that the hinge loss leads to results that are very similar to the ones relying on the more commonly used cross-entropy loss [17]. It has the advantage however to stop in finite time in the over-parametrized regime $N > N^*$.

1.2 Numerical Setting

We first consider the task of classifying the parity of digits on the MNIST database [44]. For this architecture we consider only the first ten PCA components of the images. We then test our findings with a CNN architecture on the full images in the CIFAR10 dataset.

The DNNs are trained using a full-batch procedure (as opposed to stochastic gradient) described in $S.I$, for a maximum running time $t^* = 2 \cdot 10^6$ steps.

2 Numerical Results on MNIST

Fig. 2 demonstrates the performance of the above setup for the MNIST dataset: we find that at the end of training, the test error (i.e. the empirical generalization error) reaches a local maximum in a cusp-like fashion near the jamming transition N^* and then slowly decreases as N becomes larger. We denote by $\bar{f}^n_N$ the average of n samples of the function f_N taken with independent initial conditions. Remarkably, in our experiments, ensemble-averaging with $n = 20$ leads to a nearly flat test error for $N > N^*$; this supports the hypothesis that the improvement of generalization performance with N originates from reduced variance of f_N when N gets large, as recently observed for mean-square regression [29]. In addition to this leading finite-size effect, an interesting sub-leading finite-size effect can be observed, as discussed in Section 7.

3 Relationship Between Variance and Generalization in Classification Tasks [1]

3.1 Regression task

For mean square regression of some target function f_{true}, the increase of the mean square test error implied by the fluctuations of the output function is readily computed. Let us write $f_N = \bar{f}_N + \delta f_N$, where $\bar{f}_N = \lim_{n\to\infty} \bar{f}^n_N$ is the output of the learnt function, averaged over runs with different initial condition. δf_N is the relative distance between a single output and this average. Then

$$\Delta\epsilon = \overline{\|\bar{f}_N + \delta f_N - f_{\text{true}}\|^2_\mu} - \overline{\|\bar{f}_N - f_{\text{true}}\|^2_\mu} = \|\delta f_N\|^2 \quad (2)$$

is the contribution to the generalization error due to the fluctuations of the output function. The bar represents averages over different runs or initial conditions. For a measure μ on $\mathbb{R}^{n_{\text{in}}}$, we set $\|f\|^2_\mu = \int d\mu(x) f(x)^2$. The measure could be for instance the empirical measure on the training set or on the test set.

Our results below apply directly to mean square regression. In the next paragraphs we will argue that a similar quadratic relationship between test error and fluctuations also holds for classification under mild assumptions on the data; so that our results extend to that case as well.

3.2 Classification task

We now provide a heuristic argument relating fluctuations of the output function f_N to generalization performance. For a random function f (e.g. a DNN function with random initialization), we denote by $\langle\cdot\rangle = \langle\cdot\rangle_f$ the expectation with respect to f.

Consider a random smooth function f with expectation $\bar{f}$, and set $\delta f \equiv f - \bar{f}$. Let $B, \bar{B}$ denote the decision boundaries $B = \{f(x) = 0\}, \bar{B} = \{\bar{f}(x) = 0\}$, and consider a point x_0 that is being classified differently by f and $\bar{f}$, i.e. $f(x_0)\bar{f}(x_0) < 0$, as

[1] In spirit, this section shares some similarity with the bias variance decomposition developed in [45], except that we consider averaging on initial conditions instead of training set, and that we use the average output function as predictor, rather than applying the majority rule on a set of predictions.

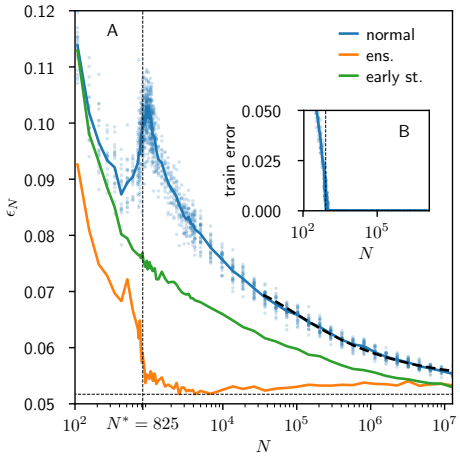


Figure 2: (A) Empirical test error *v.s.* number of parameters: average curve (blue, averaged over 20 runs); early stopping (green); ensemble average $\bar{f}_N^n$ (orange) over $n = 20$ independent runs. In all the simulations, we used fully-connected networks with depth $L = 5$ and input dimension $n_{\text{in}} = 10$, trained for $t = 2{\cdot}10^6$ epochs to classify $P = 10k$ MNIST images depending on their parity, using their first 10 PCA components. The test set consists of $50k$ images. The vertical dashed line corresponds to the jamming transition: at that point the test error displays a cusp-like local maximum. Ensemble averaging leads to an essentially constant behavior when N becomes larger than N^*. Black dashed line: asymptotic prediction of the form $\epsilon_N - \epsilon_\infty = B_0 N^{-1/2} + B_1 N^{-3/4}$, with $\epsilon_\infty = 0.054$, $B_0 = 6.4$ and $B_1 = -49$. (B) Training error *v.s.* number of parameters.

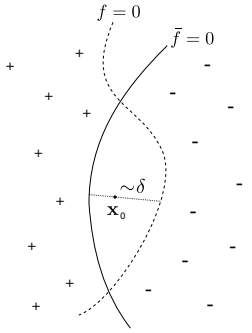


Figure 3: $f(x)$ and the expected function $\bar{f}(x)$ (see Section 3) classify points according to their sign. They agree on the classification everywhere (±'s in the figure are examples where the functions are respectively both positive or both negative) except for the points that lie in between the two boundaries $f = 0$ and $\bar{f} = 0$. In the figure, let x be one such point, and δ is the typical distance from the boundary $f = 0$. In the limit where f and $\bar{f}$ are close to each other, δ is of the same order of the distance between the two boundaries.

illustrated in Figure 3. Imagine drawing the shortest segment passing through x_0 that starts from a point in $\bar{B}$ and ends in B. If its length $\delta(x_0)$ is small, then the signed distance $\delta(x_0)$ between B and $\bar{B}$ is $\delta(x_0) = \delta f(x_0)/\|\nabla f(x_0)\| + o(\delta f(x_0))$. Note that for smooth activation functions, the smoothness of DNN output function is guaranteed and for ReLU-based DNNs, the output function is smooth outside of the training points (see S.I.). We show direct measurements of $\delta(x)$ in Section A of S.I., supporting that this estimate still holds and becomes more and more accurate as $N \to \infty$.

Next, we introduce the typical distance δ along the boundary:

$$\delta \equiv \langle |\delta f(x_0)|/\|\nabla f(x_0)\| \rangle_{x_0} \tag{3}$$

where the average is taken over all the test data x_0 classified differently by f and $\bar{f}$. As numerically shown in S.I., δ is very well estimated by $\|\delta f\|_\mu / \|\nabla f\|_\mu$ where μ is the uniform measure on all the test set.

We then denote by $\Delta\epsilon$ the difference between the true test error of f and that of $\bar{f}$. Under reasonable assumptions [2] it can be expanded by considering a small perturbation of the decision boundary $\bar{B}$ of $\bar{f}$ (that can consist of unconnected parts):

$$\Delta\epsilon = \int_B dx^{n_{\text{in}}-1} \left[\frac{\partial \epsilon}{\partial \delta(x)} \delta(x) + \frac{1}{2} \frac{\partial^2 \epsilon}{\partial^2 \delta(x)} \delta^2(x) + \mathcal{O}(\delta^3(x)) \right]. \tag{4}$$

The fact that $\langle \delta f(x) \rangle = 0$, suggests that $\langle \delta(x) \rangle = \mathcal{O}(\delta f(x)^2)$. This suggests in turn that in average the true test error increases quadratically with the norm of fluctuations δf:

$$\langle \Delta\epsilon \rangle \sim \langle \delta^2 \rangle \sim \left\langle \frac{\|\delta f\|_\mu^2}{\|\nabla f\|_\mu^2} \right\rangle. \tag{5}$$

Note that if $\bar{f}$ displays a minimal true test error, the decision boundary is optimal: $\partial\epsilon/\partial\delta(x) = 0$ and $\partial^2\epsilon/\partial^2\delta(x) \geq 0$ for all $x \in B$, implying that the prefactor in Eq. (5) must be positive [3]. If the true test error is small, the decision boundary will tend to be close to the ideal one, so that the prefactor in Eq. (5) will still be positive. [4]

Eq. (5) is a result on the ensemble average of the true test error. Yet, our data in Fig. 2 supports that the test error is a self-averaging quantity: the test error of a given output function (blue points) lies close to its average (blue line).

4 Asymptotic generalization as $n \to \infty$

Using the tools of the previous section, we can now study how an ensemble average f_N^n of n networks behaves in the $n \to \infty$ limit. The central limit theorem and the law of large numbers imply that $\delta f_N^n \sim 1/\sqrt{n}$ while $\|\nabla f_N^n\|_\mu$ converges to a constant. Thus $\delta \sim 1/\sqrt{n}$ and for the true test errors ϵ_N^n and

[2] We assume that the true test error is a smooth function of the decision boundary. This holds true if the probability distributions to find data of different labels are themselves smooth functions of the input (this is the case, for instance, if the input data have Gaussian noise).

[3] The pre-factor could be zero if the optimal boundary is degenerate, a situation that will not occur generically if the data have e.g. Gaussian noise.

[4] We expect this to be the case for the MNIST model we consider for which the test error is a few percents.

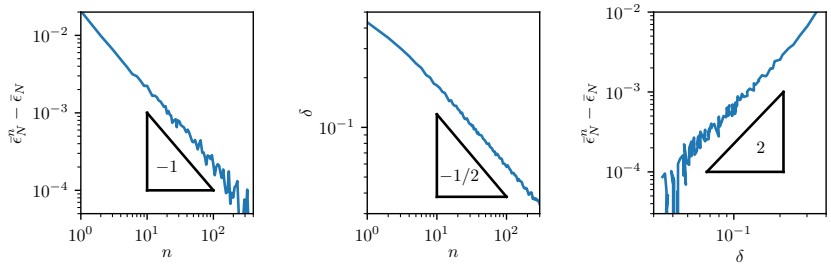


Figure 4: Left: increment of test error $\bar{\epsilon}_N^n - \bar{\epsilon}_N$ *v.s.* n, supporting $\bar{\epsilon}_N^n - \bar{\epsilon}_N \sim 1/n$. Center: δ as defined in Eq. (3) *v.s.* number of average n, supporting $\delta \sim 1/\sqrt{n}$. Right: increase of test error $\bar{\epsilon}_N^n - \bar{\epsilon}_N$ as a function of the variation of the boundary decision δ, supporting the prediction $\bar{\epsilon}_N^n - \bar{\epsilon}_N \sim \delta^2$. Here $n_{\text{in}} = 30$, $h = 60$, $L = 5$, $N = 16k$ and $P = 10k$. The value $\bar{\epsilon}_N = 2.148\%$ is extracted from the fit.

$\bar{\epsilon}_N^n$ of f_N^n and $\bar{f}_N^n$, we have $\epsilon_N^n - \bar{\epsilon}_N^n \sim 1/n$. These predictions are confirmed in Fig. 4.

5 Asymptotic Generalization as $N \to \infty$

We now study the fluctuations of $f_{N,t}$ throughout training for large networks using the NTK [20]. At initialization $t = 0$, $f_{N,t=0}$ is a random function whose limiting distribution as $N \to \infty$ is an explicit Gaussian [46, 47, 48]. These types of fluctuations do not vanish as $N \to \infty$: the variance of $f_{N,t=0}$ at initialization is essentially constant in N [5].

However, during the DNN training, the fluctuations of $f_{N,t}$ will shrink around the training points [20]. At the end of training, outside of the training points, the fluctuations due to the random initialization of the parameters manifest themselves in two ways: from the randomness of the initialization point in function space $f_{N,t=0}$ and from the randomness of the learning dynamics. The first one is essentially independent of N. Hence, to understand the way the fluctuations of the function at convergence $t \to \infty$ decrease with N, we must thus study the random fluctuations of the training process. The gradient descent dynamics of $f_{N,t}$ is described by the NTK $\Theta_{N,t}$:

$$\Theta_{N,t}(x, x') = \sum_{k=1}^{N} \frac{d}{d\theta_k} f_{N,t}(x) \frac{d}{d\theta_k} f_{N,t}(x') \tag{6}$$

where $\frac{d}{d\theta_k} f_{N,t}$ is the derivative of the output of the network with respect to one parameter θ_k and the sum is over all the network's parameters. For a general cost $C(f) = \frac{1}{P} \sum_i c_i(f(x_i))$, the function follows the kernel gradient $\nabla_{\Theta_{N,t}} C_{|f_{N,t}}$ of the cost during training

$$\begin{aligned} \partial_t f_{N,t}(x) &= -\nabla_{\Theta_{N,t}} C_{|f_{N,t}}(x) \\ &= -\frac{1}{P} \sum_i \Theta_{N,t}(x, x_i) c_i'(f_{N,t}(x_i)). \end{aligned} \tag{7}$$

[5] In our setup, the output variance at initialization is smaller than one. It is possible to suppress the randomness of $f_{N,t=0}$ at initialization by training $f_t' = f_t - f_{t=0}$. We have observed that it does not qualitatively affects our results.

The NTK is random at initialization and varies during training. However as the number h of neurons in each hidden layer goes to infinity, the NTK converges to a deterministic limit $\Theta_N^t \to \Theta_\infty$ which stays constant throughout training [20]. In this limit, the training corresponds to that of a kernel method (i.e. the output evolves along the vector space spanned by the functions $\Theta_\infty(x, x_i)$). The random fluctuations of the training process have now themselves two sources: the random fluctuations of the NTK at initialization, and the evolution of the NTK during training. On the one hand, we have that the variation of the NTK during training is of order $1/\sqrt{N}$, as is suggested by [49]:

$$\left\| \Theta_N^{t=0} - \Theta_N^{t=T} \right\|_F = \mathcal{O}\left(\frac{1}{h}\right) = \mathcal{O}\left(N^{-1/2}\right).$$

($\|\Theta\|_F = \sum_{ij} \Theta(x_i, x_j)^2$ is the Frobenius norm of the Gram matrix computed over the training set). On the other hand, the random fluctuations of the NTK at initialization are of order $N^{-1/4}$

$$\left\| \Theta_N^{t=0} - \Theta_\infty \right\|_F = \mathcal{O}\left(\frac{1}{\sqrt{h}}\right) = \mathcal{O}\left(N^{-1/4}\right). \tag{8}$$

Eq. (8) can be readily obtained by re-writing Eq. (6) as a sum on neurons and using the central limit theorem, as sketched in S.I. and tested empirically in [49]. From the above, we see that dominant source of random fluctuations during training is due to the randomness of the NTK at initialization and is of order $N^{-1/4}$.

Because the NTK describes the behaviour of the function $f_{N,t}$ during training, and because the time to converge to a minimum of the loss converges to a constant as $N \to \infty$, from Eq. (7) we expect the variance of the NTK to induce some variance of the same order to the function at the end of training: this is proven in the case of the mean square loss in the S.I. Hence, the random fluctuations of the kernel leads to fluctuations of $f_N^{t=\infty}$ of order $\mathcal{N}^{-1/4}$, and we predict:

$$\|f_{N,t=\infty} - \bar{f}_{N,t=\infty}\|_\mu - \left\langle \|f_\infty - \bar{f}_\infty\|_\mu \right\rangle \sim N^{-1/4}, \tag{9}$$

where the residual variance $\left\langle \|f_\infty - \bar{f}_\infty\|_\mu \right\rangle$ is due to the fact that we consider a finite dataset. In our setting, since our

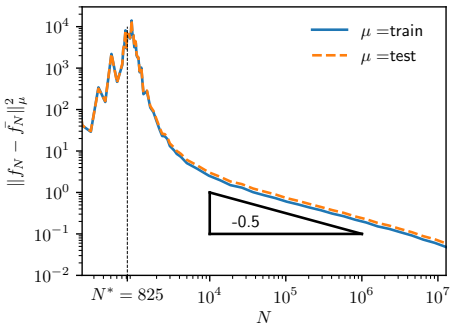


Figure 5: Variance of the output (averaged over $n = 20$ networks) *v.s.* number of parameters for different measures indicated in legend, showing a peak at jamming followed by a decay as N grows. Here $L = 5$, $n_{\text{in}} = 10$, $P = 10k$.

dataset is large, this residual term is negligible, leading one to:

$$\|f_{N,t=\infty} - \bar{f}_{N,t=\infty}\|_\mu \sim N^{-1/4}. \tag{10}$$

as checked in Fig.5.

We expect the fluctuations of ∇f_N to be of the size as those of f_N, leading to $\|\nabla f_N\|_\mu = C_0 + C_1 N^{-1/4} + o(N^{-1/4})$. This result is consistent with our observations, as shown in Fig. 6.A, in which we find empirically that C_1 is much larger than C_0. For the true test errors $\epsilon_N, \bar{\epsilon}_N$ of $f_N, \bar{f}_N$, from the decision boundary discussion, we get

$$\langle \epsilon_N \rangle - \bar{\epsilon}_N \sim \langle \delta_N^2 \rangle,$$

where δ_N indicates the typical distance between the decision boundaries $\bar{f}_N = 0$ and $f_N = 0$, as supported by Fig. 6.B. The fluctuations of the decision boundary δ_N can be approximated by $\|f_N - \bar{f}_N\|/\|_\mu \nabla f_N\|_\mu$, as supported by Fig. 6.C, leading to $\delta_N = A_0 N^{-1/4} + A_1 N^{-1/2} + o(N^{-1/2})$. We then obtain the key prediction

$$\epsilon_N - \bar{\epsilon}_N = B_0 N^{-1/2} + B_1 N^{-3/4} + o(N^{-3/4}). \tag{11}$$

Since we measure both ϵ_N and $\bar{\epsilon}_N$ independently, we can test the prediction for the leading exponent without any fitting parameters, and indeed confirm that asymptotically $\epsilon_N - \bar{\epsilon}_N$ is of order $N^{-1/2}$ as shown in Fig. 6.D.

Finally we estimate the evolution of test error with N. We have:

$$\epsilon_N - \langle \epsilon_\infty \rangle = (\epsilon_N - \bar{\epsilon}_N) + (\bar{\epsilon}_N - \bar{\epsilon}_\infty) + (\bar{\epsilon}_\infty - \langle \epsilon_\infty \rangle), \tag{12}$$

where ϵ_∞ denotes the true test error of f_N as $N \to \infty$ (notice that ϵ_∞ is still random, due to the random initialization and the fact that we have a finite dataset). The first term was estimated above, and turns out to be the dominant one for large datasets. The last term is independent of N, and cancels the first term for asymptotically large N (unaccessible in our numerics).

We provide a scaling argument to estimate the size of the second term. For large N, we expect the difference between $\bar{f}_N$ and $\bar{f}_\infty$ to stem from (i) the evolution of the kernel with time (which corresponds to learning features) and (ii) the fact that the relationship between the kernel and the function at infinite time is not linear, as described for the mean square loss in Eq. (17) of the S.I. Both effects are $\mathcal{O}(N^{-1/2})$, i.e. much smaller than the $\mathcal{O}(N^{-1/4})$ fluctuations of f_N around its mean. The typical distance $\delta_{N,\infty}$ between the interfaces $\bar{f}_N = 0$ and $\bar{f}_\infty = 0$ is thus small and $\mathcal{O}(N^{-1/2})$. According to Eq. (4) we get:

$$\bar{\epsilon}_N - \bar{\epsilon}_\infty = \int_B dx^{n_{\text{in}}-1} \left[\frac{\partial \epsilon}{\partial \delta(x)} \delta_{N,\infty}(x) + \mathcal{O}(\delta_{N,\infty}^2(x)) \right] \tag{13}$$

Thus $\bar{\epsilon}_N - \bar{\epsilon}_\infty = \mathcal{O}(N^{-1/2})$ cannot be neglected a priori. Overall, we get:

$$\epsilon_N - \epsilon_\infty = B_0 N^{-1/2} + B_1 N^{-3/4} \tag{14}$$

a form indeed consistent with observation as shown in Fig. 2.

For MNIST, both for FC and CNN (below), we always find $B_0 > 0$, consistent with the notion that the dominant effect of finite N is the increase in fluctuations of the output.

Note that a direct fit of the test error *vs* N gives an apparent exponent smaller than 1/2 [17], reflecting that (i) power-law fits are less precise when the value for the asymptote (here the value of ϵ_∞) is a fitting parameter and (ii) that correction to scaling needs to be incorporated for a good comparison with the theory (a fact that ultimately stems from the large correction to scaling of $\|\nabla f_N\|_\mu$ shown in Fig. 6.A).

6 Vicinity of the jamming transition

The asymptotic description for generalization in the large N limit is not qualitatively useful for $N \leq N^*$, where a cusp in test error is found. In the perceptron, the simplest network without hidden layers, the cusp in the test error at the jamming point is also observed and predicted analytically [50, 51, 52, 53, 54, 55]. Here instead, we argue that this cusp is induced by a divergence of $\|f_N\|_\mu$ at N^* when no regularization is used, as apparent in Fig. 7.A (no such divergence happens in the perceptron where $\|f_N\|_\mu$ is generally imposed). Indeed following our argument of Section 3, this effect must lead to singular fluctuations of the decision boundary at N^*, suggesting a singular behavior for the true test error. This phenomenon shares some similarity with the norm divergence that occurs in linear networks with mean square loss for which $\|f_N\|_\mu \sim |N - P|^{-2}$ [27, 28]. Yet, for losses better suited for classification such as the hinge loss, we argue that this explosion occurs at a different location with a different exponent.

Consider the hinge loss defined in Eq. (1). For $N \geq N^*$, the DNN is able to reach the global minimum of the loss, therefore all Δ_μ must be negative, i.e. all patterns must satisfy $y_\mu f(x_\mu) > \epsilon_m$. The parameter ϵ_m plays the role of a margin above which we are confident about the network's prediction. Because we do not use regularization on the norm $\|f\|_\mu$, the precise choice of ϵ_m does not affect N^*. Indeed the weights can always adjust during learning so as to multiply f by any scalar λ, effectively reducing the margin by a factor $1/\lambda$, making the data easier to fit. By contrast, if a regularization is imposed to fix $\|f\|_\mu = \lambda$ (which may be hard to implement in practice),

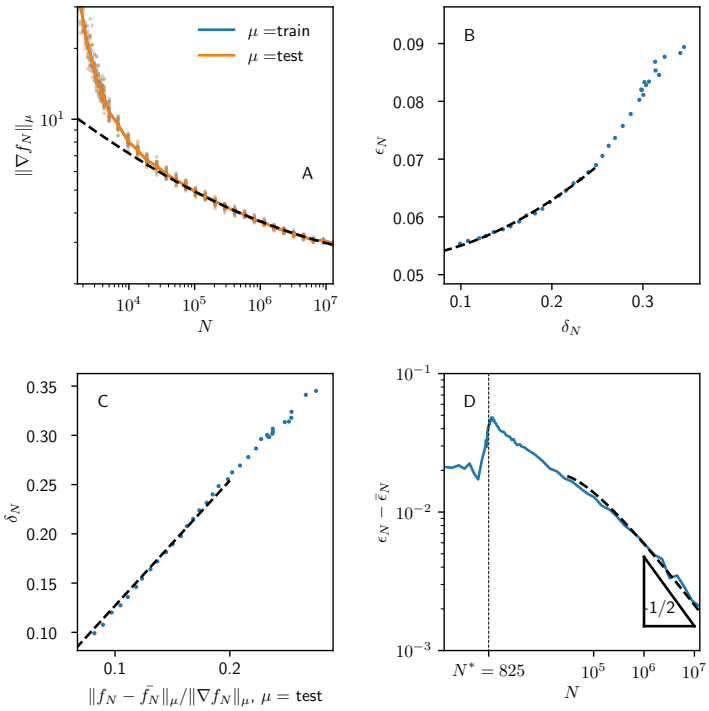


Figure 6: Here $L = 5$, $n_{\text{in}} = 10$, $P = 10k$. (A) The median of $\|\nabla f_N\|_\mu = \sqrt{\int d\mu(x) \|\nabla f_N(x)\|^2}$ over 20 runs (each appearing as a dot) is indicated as a full line. The dashed line correspond to our asymptotic prediction $\|\nabla f_N\| = C_0 + C_1 N^{-1/4}$ with $C_0 = 2.1$ and $C_1 = 51$. (B) Test error *v.s.* variation of the boundary, together with fit of the form $\epsilon_N = \epsilon_\infty + D_0 \delta_N^2$. (C) Variation of the boundary δ_N *v.s.* its estimate $\|f_N - \bar{f}_N\|/\|\nabla f_N\|$, well fitted by a linear relationship. (D) $\epsilon_N - \bar{\epsilon}_N$ *v.s.* N, with a fit of the form $\epsilon_N - \bar{\epsilon}_N = E_0 N^{-1/2} + E_1 N^{-3/4}$ with $E_0 = 7.6$ and $E_1 = -59$. If exponents in the fits are not imposed, we find for reasonable fitting ranges -0.28 instead of $-1/4$ in (A), 2.5 instead of 2 in (B), 1.1 instead of 1 in (C) and -0.42 instead of $-1/2$ in (D). Extracting exponents while also fitting for the location of the singularity, as is the case here for (A) and (B), leads to rather sloppy fits.

then N^* must be an increasing function of $\tilde{\epsilon}_m \equiv \epsilon_m/\lambda$. We assume that this function is differentiable in its argument around zero, a fact know to be true for the perceptron [56, 57], thus $N^*(\tilde{\epsilon}_m) = N^*(0) + B_0 \tilde{\epsilon}_m + o(\tilde{\epsilon}_m)$. Now consider our learning scheme (no regularization) for a network with $0 < N/N^*(0) - 1 \ll 1$, with initial conditions such that before learning $\|f_{N,t=0}\| = 1$. Initially, the effective margin is large with $\tilde{\epsilon}_m = 1$. Yet, all data can be fitted and the loss brought to zero if the norm increases so that $\tilde{\epsilon}_m \approx (N - N^*(0))/B_0$, corresponding to $\|f_N^t\| \sim (N - N^*)^{-1}$ where $N^* = N^*(0)$. At later times, the loss is zero and the dynamics stops.

This predicted inverse relation is tested in Fig. 7.B. It is important to note that, as it is the case for any critical points, working at finite times cuts off a true singularity: as illustrated in Fig. 7.B $\|f_{N,t}\|$ becomes more and more singular as t grows. This effect also causes a shift of the transition N^* where the loss vanishes, that converges asymptotically to a well-defined value in the limit $t \to \infty$ as documented in [16]. N^* is therefore defined when $\|f_{N,t}\|$ displays a power law as function of $N/N^* - 1$.

Note that for other losses like the cross-entropy, the dynamics never stops completely but becomes extremely slow [15]. In such cases, we expect that asymptotically $\|f_{N,t}\| = \infty$ as soon as $N > N^*$, although this singularity should build up logarithmically slowly in time. For finite learning times we expect that a singularity will occur near N^*, but will be blurred as for the hinge loss if $t < \infty$.

7 Subleading Finite-Size Effect

For a given computational envelope, it appears be more efficient to take a value of N slightly bigger than N^*, and to perform ensemble-averaging to reduce the variance. Quite remarkably, as shown in Figure 2, an additional effect appears to take place after ensemble-averaging: taking N only slightly bigger than N^* is not only more efficient from a computational point of view, but it also yields to a slightly better generalization performance than $N \gg N^*$. This corresponds to the middle term in Equation 13.

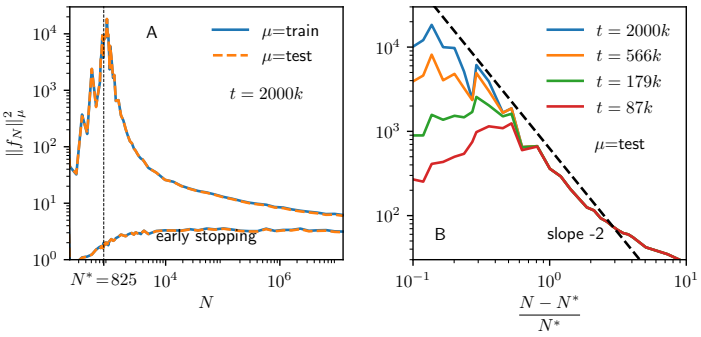


Figure 7: Here $L = 5$, $n_{\text{in}} = 10$, $P = 10k$. (A) $\|f\|_\mu^2 = \int d\mu(x) f(x)^2$ where for μ we took the uniform measure on the training and test set. We show the mean over the different realizations. Right after the jamming transition, the norm of the network diverges. (B) Same quantity computed after different learning times t as indicated in the legend, as a function of the distance from the transition. One observes that finite times cut off the divergence in the norm. The black line indicates a power-law with slope -2, that appears to fit the data satisfyingly. N^* has been fine tuned to obtain straight curves (power law behavior).

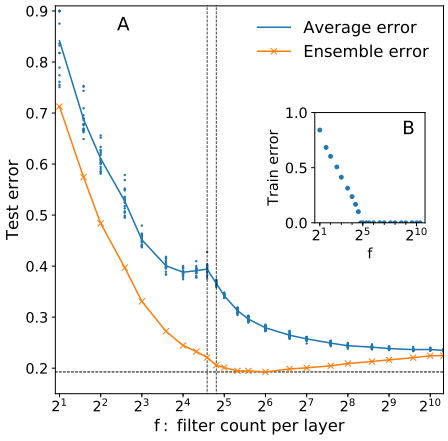


Figure 8: Empirical test (**A**) and train (**B**) error *v.s.* f, the number of filters at each convolutional layer: average curve (blue dots, averaged over 20 runs); ensemble average $\bar{f}_N^n$ (orange dots) over $n = 20$ independent weight initializations. The architecture is a three convolutional and 1 fully-connected layer and the model is trained on the standard CIFAR10 using stochastic gradient descent with a fixed learning rate during training. The jamming transition occurs at $\mathsf{f} \in \{24, \dots, 28\}$.

This could be viewed as supporting the classical intuition that keeping the models sparse by controlling the number of parameters is useful, when one averages over differently initialized networks and once the network is large enough. This effect appears stronger for CNN architecture, as confirmed in Section 8.

This effect could be explained by an evolution of the NTK during training. It suggests the possibility that (with ensembling) DNNs at finite N perform better than their kernel method counterparts. It hence appears to be both a very promising direction for future theoretical research and to be of practical interest.

8 Extension to Convolutional Networks

In this section, we test the generality of our findings for Convolutional Networks (CNNs) used for classification. We train the CNN on the CIFAR10 dataset which consists of 50,000 training and 10,000 test images of 32 by 32 resolution. Each image is labeled by one of the ten possible classes. The architecture is a vanilla model with 3 convolutional and 1 fully-connected layers. Each convolutional layer has f channels and the output of the CNN is a 10-dimensional vector (see S.I. for more details). The loss function is linear-hinge $C = \frac{1}{P}\sum_{\mu=1}^{P} \max(0, \Delta_\mu)$. We vary f from 2^1 to 2^{11}. For each value of f, we train $n = 20$ models with independent random initial conditions. For each f, the learning rate throughout is fixed at $1/\mathsf{f}$. The jamming transition occurs just before $\mathsf{f} \sim 28$. Soon after the transition, at $\mathsf{f} \sim 40, 48, 64$, the mean performances are between $\sim \%67 - 72$. The performance of the ensemble averaging is $\sim \%80.5 - 80.7$, and the average accuracy of the widest models is a little bit less than $\sim \%77.5$. Peak performance is achieved by ensembling with $\mathsf{f} = 64$, yielding a value of $\sim 80.7\%$, while the average performance without ensembling is lowest at $\mathsf{f} = 1280$ with a value of $\sim 77.5\%$.

9 Conclusion

We have provided a description for the evolution of the generalization performance of fixed-depth fully-connected deep neural networks, as a function of their number of parameters N. In the asymptotic regime of very large N, we find empirically that the network output displays reduced fluctuations with $\|f_N - \bar{f}_N\|_\mu \sim N^{-1/4}$. We have argued that this scaling behavior is expected from the finite N fluctuations of the Neural Tangent Kernel known to control the dynamics at $N = \infty$. Next we have provided a general argument relating fluctuations of the network output function to decreasing generalization performance, from which we predicted for the test error $\epsilon_N - \epsilon_\infty = C_0 N^{-1/2} + C_1 N^{-3/4} + \mathcal{O}(N^{-1})$, consistent with our observation on MNIST. Overall this approach explains the surprising finding that generalization keeps improving with the number of parameters.

We have then argued that this description breaks down at $N = N^*$ below which the training set is not fitted. For the hinge loss where this jamming transition is akin to a critical point, and in the case where no regularization (such as early stopping) is used, we observe the apparent divergence $\|f_N\| \sim (N - N^*)^{-\alpha}$. We have argued, based on reasonable assumptions, that $\alpha = 1$, consistent with our observations. This predicted blow up of the norm of f_N explains the spike in the error observed at N^*.

Our analysis furthermore suggests that optimal generalization does not require to take N much larger than N^*: since improvement of generalization with N stems from reduced variance in the output function, near-optimal generalization is readily obtained by performing an ensemble average of networks with N fixed, e.g. taken to be a few times N^*. The usefulness of averaging breaks down near N^*, where the variance of f_N is too large. This suggests that given a computational envelope, it is best from a generalization performance point of view to ensemble slightly beyond the jamming transition point. This is a result of practical importance which needs to be tested in a wide range of architectures and datasets.

Acknowledgements

We thank Marco Baity-Jesi, Carolina Brito, Chiara Cammarota, Taco S. Cohen, Silvio Franz, Yann LeCun, Florent Krzakala, Riccardo Ravasio, Andrew Saxe, Pierfrancesco Urbani and Lenka Zdeborova for helpful discussions.

This work was partially supported by the grant from the Simons Foundation (#454935 Giulio Biroli, #454953 Matthieu Wyart). M.W. thanks the Swiss National Science Foundation for support under Grant No. 200021-165509. C.H. acknowledges support from the ERC SG Constamis, the NCCR SwissMAP, the Blavatnik Family Foundation and the Latsis Foundation. We thank the KITP and the National Science Foundation under Grant No. NSF PHY-1748958 for hosting us while this manuscript was written.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[5] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2017.

[6] C Daniel Freeman and Joan Bruna. Topology and geometry of deep rectified network optimization landscapes. *International Conference on Learning Representations*, 2017.

[7] Luca Venturi, Afonso Bandeira, and Joan Bruna. Neural networks with finite intrinsic dimension have no spurious valleys. *arXiv preprint arXiv:1802.06384*, 2018.

[8] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pages 1729–1739, 2017.

[9] Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

[10] Yaim Cooper. The loss landscape of overparameterized neural networks. *arXiv preprint arXiv:1804.10200*, 2018.

[11] Levent Sagun, Léon Bottou, and Yann LeCun. Singularity of the hessian in deep learning. *International Conference on Learning Representations*, 2017.

[12] Levent Sagun, Utku Evci, V. Uğur Güney, Yann Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *ICLR 2018 Workshop Contribution, arXiv:1706.04454*, 2017.

[13] Andrew J Ballard, Ritankar Das, Stefano Martiniani, Dhagash Mehta, Levent Sagun, Jacob D Stevenson, and David J Wales. Energy landscapes for machine learning. *Physical Chemistry Chemical Physics*, 2017.

[14] Zachary C Lipton. Stuck in a what? adventures in weight space. *International Conference on Learning Representations*, 2016.

[15] Marco Baity-Jesi, Levent Sagun, Mario Geiger, Stefano Spigler, Gerard Ben Arous, Chiara Cammarota, Yann LeCun, Matthieu Wyart, and Giulio Biroli. Comparing dynamics: Deep neural networks versus glassy systems. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 314–323, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[16] Mario Geiger, Stefano Spigler, Stéphane d'Ascoli, Levent Sagun, Marco Baity-Jesi, Giulio Biroli, and Matthieu Wyart. The jamming transition as a paradigm to understand the loss landscape of deep neural networks. *arXiv preprint arXiv:1809.09349*, 2018.

[17] Stefano Spigler, Mario Geiger, Stéphane d'Ascoli, Levent Sagun, Giulio Biroli, and Matthieu Wyart. A jamming transition from under-to over-parametrization affects loss landscape and generalization. *arXiv preprint arXiv:1810.09665*, 2018.

[18] Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 2933–2941, 2014.

[19] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[20] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems 31*, pages 8580–8589. 2018.

[21] Simon S. Du, Xiyu Zhai, Barnabs Pczos, and Aarti Singh. Gradient Descent Provably Optimizes Over-parameterized Neural Networks. 2019.

[22] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A Convergence Theory for Deep Learning via Over-Parameterization. *CoRR*, abs/1811.03962, 2018.

[23] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

[24] Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of optimization and implicit regularization in deep learning. *arXiv preprint arXiv:1705.03071*, 2017.

[25] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

[26] Yamini Bansal, Madhu Advani, David D Cox, and Andrew M Saxe. Minnorm training: an algorithm for training over-parameterized deep neural networks. *CoRR*, 2018.

[27] Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

[28] Zhenyu Liao and Romain Couillet. The dynamics of learning: A random matrix approach. *arXiv preprint arXiv:1805.11917*, 2018.

[29] Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018.

[30] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70), 2018.

[31] Tengyuan Liang and Alexander Rakhlin. Just interpolate: Kernel" ridgeless" regression can generalize. *arXiv preprint arXiv:1808.00387*, 2018.

[32] Lenaic Chizat and Francis Bach. A note on lazy training in supervised differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018.

[33] Grant M Rotskoff and Eric Vanden-Eijnden. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*, 2018.

[34] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layers neural networks. *arXiv preprint arXiv:1804.06561*, 2018.

[35] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks. *arXiv preprint arXiv:1805.01053*, 2018.

[36] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[37] Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *arXiv preprint arXiv:1903.07571*, 2019.

[38] Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.

[39] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

[40] Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. *arXiv preprint arXiv:1909.05989*, 2019.

[41] Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. *arXiv preprint arXiv:1909.11304*, 2019.

[42] Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. *arXiv preprint arXiv:1902.06015*, 2019.

[43] Phan-Minh Nguyen. Mean field limit of the learning dynamics of multilayer neural networks. *arXiv preprint arXiv:1902.02880*, 2019.

[44] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998.

[45] Pedro Domingos. A unified bias-variance decomposition. In *Proceedings of 17th International Conference on Machine Learning*, pages 231–238, 2000.

[46] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[47] Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems 22*, pages 342–350. Curran Associates, Inc., 2009.

[48] Jae Hoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *ICLR*, 2018.

[49] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

[50] David Saad and Sara A Solla. On-line learning in soft committee machines. *Physical Review E*, 52(4):4225, 1995.

[51] Andreas Engel and Christian Van den Broeck. *Statistical mechanics of learning*. Cambridge University Press, 2001.

[52] Siegfried Bös and Manfred Opper. Dynamics of training. In *Advances in Neural Information Processing Systems*, pages 141–147, 1997.

[53] Yann Le Cun, Ido Kanter, and Sara A Solla. Eigenvalues of covariance matrices: Application to neural-network learning. *Physical Review Letters*, 66(18):2396, 1991.

[54] Silvio Franz and Giorgio Parisi. The simplest model of jamming. *Journal of Physics A: Mathematical and Theoretical*, 49(14):145001, 2016.

[55] Silvio Franz, Sungmin Hwang, and Pierfrancesco Urbani. Jamming in multilayer supervised learning models. *arXiv preprint arXiv:1809.09945*, 2018.

[56] Silvio Franz, Giorgio Parisi, Pierfrancesco Urbani, and Francesco Zamponi. Universal spectrum of normal modes in low-temperature glasses. *Proceedings of the National Academy of Sciences*, 112(47):14539–14544, 2015.

[57] Silvio Franz, Giorgio Parisi, Maxime Sevelev, Pierfrancesco Urbani, and Francesco Zamponi. Universality of the sat-unsat (jamming) threshold in non-convex continuous constraint satisfaction problems. *SciPost Physics*, 2(3):019, 2017.

[58] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations*, 2014.

[59] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

A Materials and methods

Here follow some details on the initialization and training dynamics used for the fully-connected networks. The weights of the network are initialized according to the random orthogonal scheme [58] and all biases are initialized to zero. The network is not optimized using vanilla gradient descent, as learning was then too slow to acquire appropriate statistics. Instead we used ADAM [59] with full batch and learning rate set to $\min(10^{-1}h^{-1.5}, 10^{-4})$ in order to have a smooth dynamics for all values of h. The exponent -1.5 has been empirically chosen so that the number of steps to converge is independent of h [20]. The excellent match between theory and predictions support that our conclusions are robust for a range of choices of learning dynamics.

For convolutional networks the parameters are initialized with the standard Xavier initialization and training minimizes a linear-hinge loss[6] with stochastic gradient descent, with learning rate equal to $1/\mathsf{f}$ — f being the number of channels — and batch size 250. Momentum, weight decay, or data augmentation were not used.

B Robustness of the boundaries distance $\delta(x)$ estimate

Fig.9 shows that the linear estimate for the distance $\delta(x)$ between two decision boundaries, $\delta(x) = \delta f(x)/\|\nabla f(x)\|$, holds for ReLU nonlinear function and improves as $N \to \infty$.

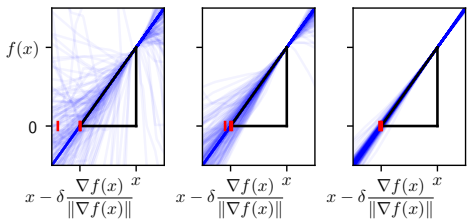


Figure 9: Value of the output function f, in the direction of its gradient starting from x. Here 200 curves are shown, corresponding to 200 data points x in the test set within the decision boundaries $f_N = 0$ and $\bar{f}_N = 0$ — i.e. $f_N(x)\bar{f}_N(x) < 0$. If the linear prediction is exact, then we expect $f(x - \delta\frac{\nabla f(x)}{\|\nabla f(x)\|}) = 0$ where $\delta = \delta f(x)/\|\nabla f(x)\|$. This prediction becomes accurate for large N. To make this statement quantitative, the 25%, 50%, 75% percentile of the intersection with zero are indicated with red ticks. Even for small N, the interval between the ticks is small, so that the prediction is typically accurate. From left to right $N = 938, 13623, 6414815$. Here $n_{\text{in}} = 10$, $L = 5$ and $P = 10k$.

Fig.10 illustrates the validity of the estimate of the typical distance between two boundary decisions presented in the main text $\delta \sim \|\delta f\|_\mu/\|\nabla f\|_\mu$, where μ corresponds to the uniform measure on all the test points.

[6] As in Eq. (1) without the square, namely $C = \frac{1}{P}\sum_{\mu=1}^{P}\max(0, \Delta_\mu)$.

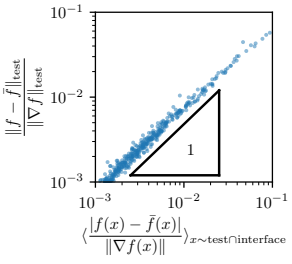


Figure 10: Test for the estimate of the distance δ between the boundary decision of f and $\bar{f}$. Each point is measured from a single ensemble average of various sizes. Here $n_{\text{in}} = 30$, $h = 60$, $L = 5$, $N = 16k$ and $P = 10k$.

C Central limit theorem of the NTK

In this section, we present a heuristic for the finite-size effects that are displayed by the NTK at initialization: informally, this is the Central Limit Theorem counterpart to the NTK asymptotic result, which can be viewed as a law of large numbers. A rigorous derivation, including the behavior during training, is beyond the scope of this paper.

The NTK can be re-written as:

$$\sum_\alpha \left[1+\tfrac{1}{h}\sum_{\beta\in v^-(\alpha)} a_\beta(x)a_\beta(x')\right]\left[\rho'(b_\alpha(x))\rho'(b_\alpha(x'))\tfrac{\partial f(x)}{\partial a_\alpha}\tfrac{\partial f(x')}{\partial a_\alpha}\right] \tag{15}$$

where $a_\alpha(x) = \rho(b_\alpha(x))$ is the activity of neuron α when data x is shown, while $b_\alpha(x)$ is its pre-activity and $v^-(\alpha)$ is the set of h neurons in the layer preceding α. The first bracket converges to a well-defined limit described by a so-called activation kernel, see [46, 47, 20]. The second bracket has fluctuations of size comparable to its mean. The normalization is chosen such that each layer contributes a finite amount to the kernel, so that the mean is of order $1/h$. For a given hidden layer, the contributions of two neurons can be shown to have a covariance that is positive and decays as $1/h^3$, and thus does not affect the scaling expected from the Central Limit Theorem for uncorrelated variables. For a rectangular network (i.e. where all hidden layers with the same size), this suggests that fluctuations associated with the contribution of one layer to the kernel is of order $1/\sqrt{h} \sim N^{-1/4}$.

D Fluctuations of output function for the mean square error loss

In this section, we discuss the fluctuations of the output function after training for the mean square error loss: $C(f) = \frac{1}{2P}\sum_i |y_i - f(x_i)|^2$. We first investigate the variance of $f_{N,t}$ in the limit $N \to \infty$, then we explain the deviations due to finite size effects, at last we discuss the hing loss case.

D.1 Infinite width

Let us first study the variance of $f_{N,t}$ in the limit $N \to \infty$. In this limit, the function $f_{\infty,t=0}$ at initialization is a centered

Gaussian process described by a covariance kernel Σ. During training, the dynamics of $f_{\infty,t}$ is described by a deterministic kernel (the large limit NTK) Θ_∞:

$$\partial_t f_{\infty,t}(x) = \frac{1}{P}\sum_i \Theta_\infty(x,x_i)\left(y_i - f_{\infty,t}(x_i)\right).$$

If the NTK is positive definite (which is proven when the inputs all lie on the unit circle and the non-linearity is not a polynomial function), the network reaches a global minimum at the end of training $t \to \infty$. In particular the values of the function on training set are deterministic: $f_{\infty,t=\infty}(x_i) = y_i$. The values of the function outside the training set can be studied using the vector of values of $f_{\infty,t}$ on the training set $\tilde{y}_t = (f_{\infty,t}(x_i))_{i=1,\dots P}$. Denoting by $\tilde{\Theta}_\infty = (\Theta_\infty(x_i,x_j))_{ij}$ the empirical Gram matrix:

$$y = \tilde{y}_{t=\infty} = \tilde{y}_{t=0} + \frac{1}{P}\int_0^\infty \tilde{\Theta}_\infty(y-\tilde{y}_t)dt,$$

so that

$$\frac{1}{P}\int_0^\infty (y-\tilde{y}_t)dt = \tilde{\Theta}_\infty^{-1}\left(y-\tilde{y}_{t=0}\right) = \tilde{\Theta}_\infty^{-1}y - \tilde{\Theta}_\infty^{-1}\tilde{y}_{t=0}.$$

These two terms represent the fact that the network needs to learn the labels y and forget the random initialization. We can therefore give a formula for the values outside the training set, using the vector $\tilde{\Theta}_{\infty,x} = (\Theta_\infty(x,x_i))_{i=1,\dots P}$:

$$\begin{aligned} f_{\infty,t}(x) &= f_{\infty,t=0}(x) + \tilde{\Theta}_{\infty,x}\frac{1}{P}\int_0^\infty (y-\tilde{y}_t)dt \\ &= f_{\infty,t=0}(x) - \tilde{\Theta}_{\infty,x}\tilde{\Theta}_\infty^{-1}\tilde{y}_{t=0} + \tilde{\Theta}_{\infty,x}\tilde{\Theta}_\infty^{-1}y. \end{aligned} \quad (16)$$

The first two terms are random, but they partly cancel each other, their sum is a centered Gaussian distribution with zero variance on the training set and a small variance for points close to the training set: the more training data points used, the lower the variance at initialization. The last term is equal to the kernel regression on y with respect to the NTK, it is not random.

This shows that even in the infinite-width limit, $f_{\infty,t=\infty}$ has some variance which is due to the variance of $f_{\infty,t=0}$ at initialization. Yet, in the setup where the number of data points is large enough, the variance due to initialization almost vanishes during training and the scaling of the variance due to finite-size effects in N will appear in the last term.

Finally, note that Eq.16 of this S.M. implies that $f_{\infty,t}(x)$ is smooth if both $\Theta_\infty(x,x')$ and $f_{\infty,t=0}(x)$ are smooth functions of x (this implication holds true for other choices of loss function). $\Theta_\infty(x,x')$ is smooth if the activation function is smooth [20], and so does $f_{\infty,t=0}(x)$ which is then a Gaussian function of smooth covariance $\Sigma(x,x')$. For Relu neurons, $\Theta_\infty(x,x')$ displays a cusp at $x=x'$ while $\Sigma(x,x')$ is smooth, so $f_{\infty,t}(x)$ is smooth except on the training set, as supported by Figure 1 of this S.M.

D.2 Finite width

For a finite width N, the training is also described by the NTK $\Theta_{N,t}$ which is random at initialization and varies during training because it depends on the parameters. The integral formula becomes

$$f_{N,t}(x) = f_{N,t=0}(x) + \int_0^\infty \tilde{\Theta}_{N,x,t}(y-\tilde{y}_t)dt$$

However the noise at initialization is of order $N^{-1/4}$, whereas the rate of change is only of order $\Omega(N^{-1/2})$. We can therefore make the approximation

$$f_{N,t}(x) = f_{N,t=0}(x) + \tilde{\Theta}_{N,x,t=0}\int_0^\infty (y-\tilde{y}_t)dt + \mathcal{O}(N^{-1/2}).$$

Assuming that there are enough parameters such that the Gram matrix $\tilde{\Theta}_{N,t=0}$ is invertible, we can again decompose the integral into two terms:

$$\int_0^\infty (y-\tilde{y}_t)dt = \tilde{\Theta}_N^{-1}y - \tilde{\Theta}_N^{-1}\tilde{y}_{t=0} + \mathcal{O}(N^{-1/2}),$$

giving that

$$f_{N,t}(x) = f_{N,t=0}(x) - \tilde{\Theta}_{N,x,t=0}\tilde{\Theta}_N^{-1}\tilde{y}_{t=0} + \tilde{\Theta}_{N,x,t=0}\tilde{\Theta}_N^{-1}y + \mathcal{O}(N^{-1/2}). \quad (17)$$

Here again the first two terms almost cancel each other, but the third term is random due to the randomness of the NTK which is of order $\mathcal{O}(N^{-1/4})$, as needed.

D.3 Hinge Loss

For the hinge loss setup, we do not have such a strong constraint on the value of the function $f_{N,t=\infty}$ on the training set $\tilde{y}_{t=\infty}$ as for regression, but we still know that they must satisfy the margin constraints

$$\tilde{y}_{i,t=\infty}y_i > 1.$$

The vector $\tilde{y}_{t=\infty}$ is therefore random for the hinge loss as a result of the random initialization of $f_{N,t=0}$ and the fluctuations of the NTK. Again it is natural to assume the first type of fluctuations to be subdominant and the second type to be of order $\mathcal{O}(N^{-1/4})$.

3 A Phase Diagram for Learning Regimes

The following paper is the preprint version of Geiger et al. (2020c).

Candidate contributions The candidate participated in the scientific discussions and performed the numerical experiments. The candidate proposed the model $\alpha(f(w)-f(w_0))$ and to use gradientflow to get rid of hyperparameters. The candidate observed that a kernel method with the NTK of the end of training gives the same performance as the network itself.

Disentangling feature and lazy training in deep neural networks

Mario Geiger, Stefano Spigler, Arthur Jacot and Matthieu Wyart

October 6, 2020

Abstract

Two distinct limits for deep learning have been derived as the network width $h \to \infty$, depending on how the weights of the last layer scale with h. In the Neural Tangent Kernel (NTK) limit, the dynamics becomes linear in the weights and is described by a *frozen* kernel Θ (the NTK). By contrast, in the Mean-Field limit, the dynamics can be expressed in terms of the distribution of the parameters associated with a neuron, that follows a partial differential equation. In this work we consider deep networks where the weights in the last layer scale as $\alpha h^{-1/2}$ at initialization. By varying α and h, we probe the crossover between the two limits. We observe two the previously identified regimes of "lazy training" and "feature training". In the lazy-training regime, the dynamics is almost linear and the NTK barely changes after initialization. The feature-training regime includes the mean-field formulation as a limiting case and is characterized by a kernel that evolves in time, and thus learns some features. We perform numerical experiments on MNIST, Fashion-MNIST, EMNIST and CIFAR10 and consider various architectures. We find that: (i) The two regimes are separated by an α^* that scales as $\frac{1}{\sqrt{h}}$. (ii) Network architecture and data structure play an important role in determining which regime is better: in our tests, fully-connected networks perform generally better in the lazy-training regime, unlike convolutional networks. (iii) In both regimes, the fluctuations δF induced on the learned function by initial conditions decay as $\delta F \sim 1/\sqrt{h}$, leading to a performance that increases with h. The same improvement can also be obtained at an intermediate width by ensemble-averaging several networks that are trained independently. (iv) In the feature-training regime we identify a time scale $t_1 \sim \sqrt{h}\alpha$, such that for $t \ll t_1$ the dynamics is linear. At $t \sim t_1$, the output has grown by a magnitude $\sqrt{h}$ and the changes of the tangent kernel $\|\Delta\Theta\|$ become significant. Ultimately, it follows $\|\Delta\Theta\| \sim (\sqrt{h}\alpha)^{-a}$ for *ReLU* and *Softplus* activation functions, with $a < 2$ and $a \to 2$ as depth grows. We provide scaling arguments supporting these findings.

1 Introduction and related works

Deep neural networks are successful at a variety of tasks, yet understanding why they work remains a challenge. A surprising observation is that their performance on supervised tasks keeps increasing with their width h in the over-parametrized regime where they already fit all the training data (Neyshabur et al., 2017; Bansal et al., 2018; Advani and Saxe, 2017; Spigler et al., 2018). This fact underlines the importance of describing deep learning in the limit $h \to \infty$.

The transmission of the signal through the network in the infinite-width limit is well understood at initialization. If the network weights are initialized as i.i.d. random variables with zero mean and a fixed variance of order $h^{-1/2}$, the output function $f(x)$ is a Gaussian random processes with some covariance that can be computed (Neal, 1996; Williams, 1997; Lee et al., 2018; de G. Matthews et al., 2018; Novak et al., 2019; Yang, 2019).

Until recently much less was know about the *evolution* of infinitely-wide networks. It turns out that two distinct limits emerge, depending on the initialization of the last layer of weights.

Mean Field limit A limiting behavior of neural networks, called "mean field" in the literature, has been studied in several works focusing mostly on one-hidden layer networks (Mei et al., 2018; Rotskoff and Vanden-Eijnden, 2018; Chizat and Bach, 2018; Sirignano and Spiliopoulos, 2018; Mei et al., 2019; Nguyen, 2019). In this setting the output function of the network with h hidden neurons corresponding to an input x is

$$f(w,x) = \frac{1}{h}\sum_{i=1}^{h} c_i \sigma(a_i \cdot x + b_i), \tag{1}$$

where $\sigma(\cdot)$ is the non-linear activation function and $w_i = (a_i, b_i, c_i)$ are the parameters associated with a hidden neuron. At initialization, the terms in the sum are independent random variables. We can invoke the law of large numbers: for large h the average tends to the expectation value

$$f(w,x) \to \int \mathrm{d}a\mathrm{d}b\mathrm{d}c\, \rho(a,b,c)\; c\sigma(a \cdot x + b), \tag{2}$$

and it has been shown in the literature that the training dynamics is controlled by a differential equation for the density of parameters ρ:

$$\partial_t \rho_t = 2\nabla \cdot (\rho_t \nabla \Psi(a,b,c;\rho_t)), \tag{3}$$

$$\Psi(a,b,c;\rho) = V(a,b,c) + \int \mathrm{d}a'\mathrm{d}b'\mathrm{d}c'\, \rho(a',b',c')\; U(a,b,c;a',b',c'). \tag{4}$$

Here ∇ is the gradient with respect to (a,b,c) and U, V are potentials defined in (Mei et al., 2018). This is equivalent to the hydrodynamic (continuous) description of interacting particles in some external potential. The performance of a network is then expected to plateau on approaching this limit, as $h \to \infty$.

NTK limit Another limit has been identified when the weights in the last layer scale as $h^{-1/2}$, which differs from the h^{-1} scaling used in the mean-field setting. This limit also applies to deep fully-connected networks and other architectures. The learning dynamics in this limit simplifies (Jacot et al., 2018; Du et al., 2019; Allen-Zhu et al., 2018; Lee et al., 2019; Arora et al., 2019; Park et al., 2019) and is entirely described by the *neural tangent kernel* (NTK) defined as:

$$\Theta(w, x_1, x_2) = \nabla_w f(w, x_1) \cdot \nabla_w f(w, x_2), \tag{5}$$

where x_1, x_2 are two inputs and ∇_w is the gradient with respect to the parameters w. The kernel Θ is defined at any time and typically evolves during the dynamics. As $h \to \infty$, $\Theta(w, x_1, x_2) \to \Theta_\infty(x_1, x_2)$ does not vary at initialization (and thus it does not depend on the specific choice of w) and does not evolve in time: the kernel is frozen. The dynamics is guaranteed to converge on a time independent of h to a global minimum of the loss, and as for usual kernel learning the function only evolves in the space spanned by the functions $\Theta_\infty(x_\mu, x)$, where $\{x_\mu, \mu = 1...n\}$ is the training set.

The existence of two distinct limits raises both fundamental and practical questions. First, which limit best characterizes the neural networks that are used in practice, and which one leads to a better performance? These questions are still debated. In (Chizat and Bach, 2019), based on teacher-student numerical experiments, it was argued that the NTK limit is unlikely to explain the success of neural networks. However, it was recently shown that the NTK limit is able to achieve good performance on real datasets (Arora et al., 2019) (significantly better than alternative kernel methods). Moreover, some predictions of the NTK limit agree with observations in networks of sizes that are used in practice (Lee et al., 2019). Second, it was argued that in the NTK limit the surprising improvement of performance with h stems from the $h^{-1/2}$ fluctuations

of the kernel at initialization, that ultimately leads to similar fluctuations in the learned function and degrades the performance of networks of small width (Geiger et al., 2019). These fluctuations can be removed by ensemble-averaging output functions obtained with different initial conditions and trained independently, leading to an excellent performance already near the underparametrized to overparametrized (or *jamming*) transition h^* beyond which all the training data are correctly fitted (Geiger et al., 2018). Does this line of thought hold true in the mean-field limit?

Finally, unlike in the NTK limit where preactivations vary very weakly during training, in the mean field limit feature training occurs. It is equivalent to saying that the tangent kernel (which can always be defined) evolves in time (Rotskoff and Vanden-Eijnden, 2018; Mei et al., 2019; Chizat and Bach, 2019). What are the characteristic time scales and magnitude of this evolution?

1.1 Our contributions

In this work we answer these questions empirically by learning a model of the form:

$$F(w,x) \equiv \alpha \left[f(w,x) - f(w_0,x) \right], \tag{6}$$

where $f(w,x)$ is a deep network and w_0 is the network parameters at initialization. This model is inspired by (Chizat and Bach, 2019). For $\alpha = \mathcal{O}(1)$ it falls into the framework of the NTK literature, whereas for $\alpha = \mathcal{O}(h^{-1/2})$ it corresponds to the mean-field framework. Our strategy is to consider a specific setup that is fast to train, simple and to some extent theoretically tractable. Thus we focus on fully-connected (FC) networks with *Softplus* activation functions and gradient-flow dynamics. In the main body of the work we consider the Fashion-MNIST dataset, later in Section 7 we extend our study to other datasets, architectures (including CNNs) and dynamics to verify the generality and robustness of our claims. For each setting we study systematically the role of both α (varied on 11 orders of magnitude) and the width h, learning ensembles of 10 to 20 networks so as to quantify precisely the magnitude of fluctuations induced by initialization and the benefit of ensemble averaging.

Our main findings are as follows: (i) In the (α, h) plane two distinct regimes can be identified, separated by $\alpha^* = \mathcal{O}(h^{-1/2})$, in which performance and dynamics are qualitatively different. (ii) Which regime achieves a lower generalization error depends on the data structure and the network architecture. We find that fully-connected architectures generally perform better in the lazy-training regime, while convolutional networks trained on CIFAR10 with ADAM achieves a smaller error in the feature-training regime. (iii) The fluctuations of the network output δF decay with the width as $\delta F = \mathcal{O}(h^{-1/2})$ in both regimes, leading to a performance that increases with the degree of overparametrization. Because of the nature of the fluctuations, a similar improvement can also be obtained at an intermediate width by ensemble-averaging several independently-trained networks. (iv) In the feature-training regime there exist a characteristic time scale $t_1 \sim \sqrt{h}\alpha$, such that for $t \ll t_1$ the dynamics is linear. At $t \sim t_1$, the output grows by a factor $\sqrt{h}$ and the tangent kernel $\|\Delta\Theta\|$ varies significantly and can not be considered approximately constant any longer. Ultimately, it follows $\|\Delta\Theta\| \sim (\sqrt{h}\alpha)^{-a}$ for *ReLU* and *Softplus* activation functions, with $a < 2$ and $a \to 2$ as depth grows. We provide scaling arguments supporting these findings.

The implications of our work are both practical (in terms of which parameters and architectures lead to improved performance) and conceptual (in quantifying the dynamics of feature training and providing informal explanations for these observations). More generally, it suggests that future empirical studies of deep learning would benefit from characterizing the regime in which they operate, since it will most likely impact their results.

The code used for this article is available online at `https://github.com/mariogeiger/feature_lazy/tree/article`.

1.2 Related work

Our work is most closely related to (Chizat and Bach, 2019) and to (Chizat et al., 2019) that appeared simultaneously to ours. The scale $\alpha \gg h^{-1/2}$ separating the lazy and feature learning regimes was justified for a one-hidden layer, in consistence with our findings that also apply to deep nets. The authors further suggest that the feature learning regime should outperform the lazy training one, as they observe in two different setups: i) a fully-connected one-hidden-layer network trained on data from a teacher network and ii) a VGG-11 deep network trained on CIFAR10 with a cross-entropy loss. Both setups achieve a smaller generalization error away from the lazy-training regime. Our empirical analysis is much more extensive both in terms of dataset and observables studied. It gives a different view, since for real data we find that for fully-connected networks lazy training tends to outperform feature learning.

2 Notations and set-up

The following setup is used for all empirical results in the article, except for those presented in Section 7, where we explore different settings.

We consider deep networks with L hidden layers performing a binary classification task. We denote by w the set of parameters (or weights), and by $f(w, x)$ the output of a network parametrized by w corresponding to an input pattern x. The set of training data is $\mathcal{T} = \{(x_\mu, y_\mu), \mu = 1, \dots, n\}$, where $y_\mu = \pm 1$ is the label associated with the pattern x_μ and n is the number of training data. In what follows $\dot{a}$ is the notation we use for the time derivative of a variable a during training.

Parameter α: Instead of training a network $f(w, x)$, we train $F(w, x) = \alpha(f(w, x) - f(w_0, x))$, *i.e* we use this quantity as our predictor and train the weights w accordingly. Here w_0 is the network's parameters at initialization. In the over-parametrized regime, this functional form ensures that for $\alpha \to \infty$, we enter in what we call the lazy-training regime, where changes of weights are small. Indeed in order to obtain a zero loss $\alpha(f(w, x) - f(w_0, x))$ must be $\mathcal{O}(1)$, thus $1 \sim \alpha(f(w, x) - f(w_0, x)) \sim \alpha \nabla_w f(w_0, x) \cdot dw$. For large α then $|f(w, x) - f(w_0, x)|$ is small: the dynamics can hence be considered linear (Chizat and Bach, 2019), with $f(w, x) - f(w_0, x) \approx \nabla_w f(w_0, x) \cdot (w - w_0)$. Since the gradient of $f(w_0, x)$ does not scale with α, this implies that $\|dw\| \equiv \|w - w_0\| \sim \alpha^{-1}$.

Learning is achieved via the minimization of the loss function

$$\mathcal{L}(w) = \frac{1}{\alpha^2 n} \sum_{(x,y)\in\mathcal{T}} \ell(\alpha(f(w, x) - f(w_0, x)), y), \tag{7}$$

where ℓ is the loss per pattern, $\mathcal{T}$ is the training set and $n = |\mathcal{T}|$. The prefactor α^{-2} ensures that the convergence time does not depend on α as $\alpha \to \infty$, since for that choice we have $\alpha \dot{f}(w_0) = \mathcal{O}(\alpha^0)$ (i.e. does not scale with α for large α).

Note that if we train directly $\alpha f(w, x)$ without removing its value at initialization, we expect no difference with the present setting for α of order one or much smaller. This statement is confirmed numerically in Appendix I. For $\alpha \gg 1$ however, we find empirically that the learning dynamics does not converge.

Dynamics Since our goal is to build a connection between empirical and theoretical approaches, we focus on a discrete version of continuous dynamics obeying simple differential equation. The simplest is the *vanilla gradient descent* which reads $\dot{w} = -\nabla_w \mathcal{L}$ and does not depend on any hyper-parameters. This dynamics is

run in a discretized form, with a time step that is adapted at each step to ensure that

$$\alpha \max_{x \in \mathcal{T}} |f(w_i, x) - f(w_{i+1}, x)| < 0.1, \tag{8}$$

$$\frac{\|\nabla_w \mathcal{L}_i - \nabla_w \mathcal{L}_{i+1}\|^2}{\|\nabla_w \mathcal{L}_i\| \, \|\nabla_w \mathcal{L}_{i+1}\|} < \epsilon_\nabla = 10^{-4}. \tag{9}$$

In Appendix G we checked that our results are independent of ϵ_∇ for $\epsilon_\nabla \leq 10^{-4}$.

Activation function The *Softplus* activation function is defined as $sp_\beta(x) = \frac{1}{\beta} \ln(1 + e^{\beta x})$. The larger the parameter β, the sharper is the *Softplus*, and as $\beta \to \infty$ it tends to a *ReLU* activation function. In our experiments we take $\beta = 5$. In order to have preactivations of unit variance at initialization we also multiply the *Softplus* by a prefactor $a \approx 1.404$. The activation function that we use at all hidden neurons is then

$$\sigma(x) = a \; sp_{\beta=5}(x) \tag{10}$$

Non-smooth activation functions like ReLU can introduce additional phenomena; for instance, we quantify its impact on the evolution of the neural tangent kernel in Appendix F.

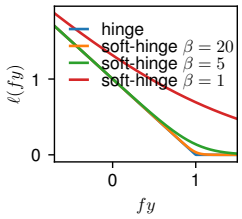


Figure 1: Comparison between the hinge loss and the soft-hinge losses $sp_\beta(1 - fy)$ with $\beta = 1, 5, 20$. As β increases, the soft-hinge loss tends to the hinge loss.

Loss function For the loss-per-pattern $\ell(f, y)$ we use the soft-hinge loss $\ell(f, y) = sp_\beta(1 - fy)$, where sp_β is again a *Softplus* function. This function is a smoothed version of the hinge loss, to which it tends as $\beta \to \infty$. We use the value $\beta = 20$ as a compromise between smoothness and being similar to the hinge loss, see Fig. 1. The stopping criterion to end the learning dynamics is met when all patterns are classified within a sufficient margin, that is when $\alpha \left[f(w, x_\mu) - f(w_0, x_\mu) \right] y_\mu > 1$ for all $\mu = 1, \ldots, n$. Keep in mind that this criterion makes sense only in the over-parametrized phase (as it will be the case in what follows), where networks manage to fit all the training set (Spigler et al., 2018). The hinge loss results in essentially identical performance to the commonly used cross-entropy in state-of-the-art architectures, but leads to a dynamics that stops in a finite time in the over-parametrized regime considered here, removing the need to introduce an arbitrary temporal cut-off (Spigler et al., 2018).

Architecture We use a constant-width fully-connected architecture based on (Jacot et al., 2018). Given an input pattern $x \in \mathbb{R}^d$, we denote by $\tilde{z}^\ell$ the vector of preactivations at each hidden layer and by z^ℓ the corresponding activations. The flow of signals through the network can be written iteratively as

$$z^\ell = \sigma(\tilde{z}^\ell), \tag{11}$$

$$\tilde{z}^1 = d^{-1/2} W^0 x, \tag{12}$$

$$\tilde{z}^{\ell+1} = h^{-1/2} W^\ell z^\ell, \tag{13}$$

$$f(w, x) = h^{-1/2} W^L z^L. \tag{14}$$

The matrices W^ℓ contain the parameters of the ℓ-th hidden layer, and the (vectorized) set of all these matrices has been previously denoted as w. The width and depth of the network are h and L; in our simulations we vary h but we mostly keep the depth $L = 3$ constant. The input patterns live a d-dimensional space.

Fig. 2 illustrates the architecture and explains our notation. In order to have preactivations of order $\mathcal{O}(1)$ at initialization, all the weights are initialized as standard Gaussian random variables, $W_{ij}^{\ell} \sim \mathcal{N}(0,1)$. Note that there is no bias, we discuss it in Appendix J.

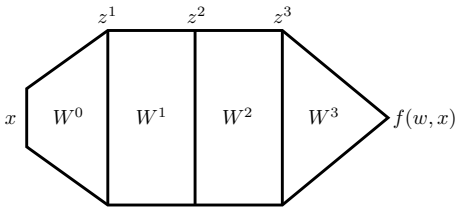


Figure 2: Fully-connected architecture with $L = 3$ hidden layers.

Dataset We train our network to classify 28×28 grayscale images of clothes from the Fashion-MNIST database (Xiao et al., 2017). For simplicity we split the original 10 classes in two sets and we perform binary classification. For $(x, y) \in \mathcal{T}$ we have $x \in \mathbb{R}^{28\times 28}$ ($d = 784$) and $y = \pm 1$. The input is normalized on the sphere, $\sum_i x_i^2 = d$ such that each component x_i has unit variance. We took 1000 images of each class (10000 in total) to make our train set and 5000 of each class (50000 in total) to make our test set. The train set is smaller than usual (10000 instead of 50000) in order to shorten the training time.

3 Disentangling feature training and lazy training according to performance

To argue the existence of two distinct regimes in deep neural networks, we evaluate their performance in the (α, h) plane. How we vary parameters to probe this plane is represented in Fig. 3 *(a)*. We consider plots obtained at fixed h and varying α (Fig. 3 *(c,d)*) as well as fixed $\sqrt{h}\alpha$ (Fig. 3 *(b)*).

Fig. 3 *(b)* shows the test error as a function of the width h, for different values of $\sqrt{h}\alpha$. For large h, we observe that as $\sqrt{h}\alpha$ is increased the performance also increases, up to a point where it converges to a limiting curve (for $\sqrt{h}\alpha \geq 10^3$ in the figures). This limiting curve coincides with the test error found if the NTK is frozen at initialization (see Appendix A for a description of our dynamics in that case), represented by a black solid line. Following (Chizat and Bach, 2019), we refer to this limiting behavior as lazy training.

There is a value of α for which we leave the lazy-training regime and enter a nonlinear regime that we call feature training.[1] To identify this regime, in Fig. 3 *(c)* we show the test error as a function of α for several widths h, and the rescaled test error (in such a way that it takes values between 0 and 1) as a function of $\sqrt{h}\alpha$ in Fig. 3 *(d)*. The curves collapse, supporting that in the (α, h) plane the boundary between the two regimes lies at a scale $\alpha^* = \mathcal{O}(h^{-1/2})$. It is precisely the scaling used in Mean Field.

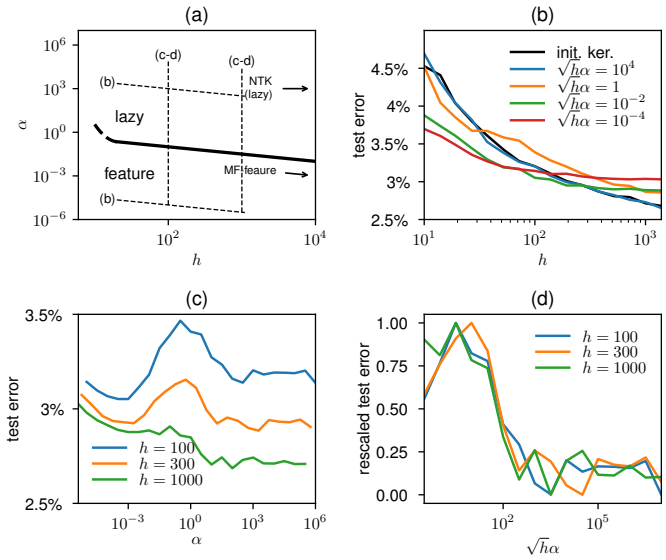


Figure 3: *(a)* Schematic representation of the parameters that we probe: either we fix $\alpha\sqrt{h}$ or we keep the width h constant and we vary α. The location of the cross-over between the lazy and feature-training regimes is also indicated. *(b)* Test error *v.s.* network's width h for different values of $\sqrt{h}\alpha$ as indicated in legend. The black solid line is the test error of the frozen NTK at initialization, a limit that is recovered as $\alpha\sqrt{h} \to \infty$. *(c)* Test error *v.s.* α, for different widths h. (averaged over 20 initializations) *(d)* Same data as in *(c)*: after an arbitrary affine transformation ($ax + b$) of the test error, the curves collapse when plotted against $\sqrt{h}\alpha$.

4 Fluctuations of the output function and the effect of ensemble averaging

As shown in Fig. 3, performance increases with h: adding more trainable parameters leads to better predictability and deep networks do not overfit. This surprising behavior was related to the fluctuations of the output function induced by initial conditions, observed to decrease with h (Neal et al., 2019; Geiger et al., 2019). To quantify the fluctuations in both regimes we train an ensemble of 20 identical functions $F(w_i, x) = \alpha\,[f(w_i, x) - f(w_{i,0}, x)]$ starting from different initial conditions, and then we measure the ensemble average $\bar{F}(x)$:

$$\bar{F}(x) \equiv \frac{\alpha}{20} \sum_{i=1}^{20} (f(w_i, x) - f(w_{i,0}, x)) \tag{15}$$

A single trained realization of the network fluctuates around the ensemble average $\bar{F}(x)$, and therefore

[1] In our parlance, the NTK and mean-field limits correspond respectively to the lazy-training and feature-training regimes in the $h \to \infty$ limit.

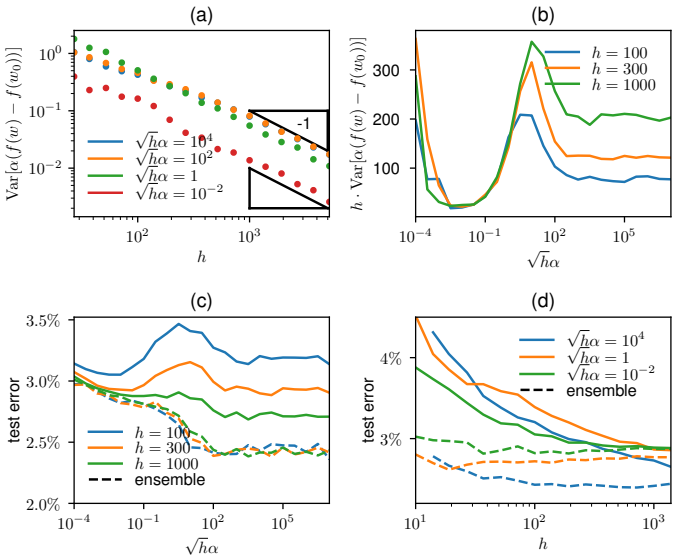


Figure 4: *(a)* Variance of the network's output *v.s.* its width h, for different values of $\sqrt{h}\alpha$. In both regimes the variance scales as $\mathrm{Var}[\alpha(f - f_0)] \sim h^{-1}$ (20 initializations per point). For this panel (a) alone, the size of the dataset was reduced to 10^3 because 10^4 requires higher h to observe the asymptotic behavior. See Appendix H for an analysis of the dependence in the size of the dataset. *(b)* Variance of the network's output times its width *v.s.* $\sqrt{h}\alpha$, for different network's width. In the lazy-training regime, it needs a larger h (or a smaller n) to observe the overlap of the curves. *(c)* Test error and ensemble-averaged test error *v.s.* $\sqrt{h}\alpha$, for several widths h (20 initializations per point). *(d)* Test error and ensemble-averaged test error *v.s.* h at fixed values of $\sqrt{h}\alpha$ (20 initializations per point). Once ensemble averaged, lazy training performs better than feature training

$\delta F(w, x) \equiv F(w, x) - \bar{F}(x)$ is a random function, whose fluctuations are quantified by the variance:

$$\mathrm{Var}\, F(w, x) = \left\langle \left[F(w_i, x_\mu) - \bar{F}(x_\mu)\right]^2 \right\rangle_{\substack{\mu \in \mathrm{test} \\ i \in \mathrm{ensemble}}}, \tag{16}$$

where the average is both over the 20 output functions and over all points x_μ in a test set. Other norms can be used to quantify this variance, all yielding the same picture. In Fig. 4 *(a-b)* we compute these fluctuations either in the feature-training or in the lazy-training regimes. In both cases we find the same decay with the network width:

$$\mathrm{Var}\ \alpha\left[f(w, x) - f(w_0, x)\right] \sim h^{-1}. \tag{17}$$

In the feature-training regime, this observation is consistent with the predictions from (Rotskoff and Vanden-Eijnden, 2018) (obtained for a one-hidden layer performing regression). Since δF is a random function with zero expectation value and finite variance $\mathrm{Var}\, F \sim h^{-1}$ we will write that $\delta F \sim h^{-1/2}$.

In Fig. 4 *(b)*, in the lazy-training regime, the curves do not overlap. This is a preasymptotic effect. In Appendix H we show that the asymptotic power law is reached for smaller dataset sizes.

It was argued in (Geiger et al., 2019) that in the lazy-training regime this scaling simply stems from the fluctuations of the NTK at initialization, that go as $\|\delta\Theta\| \sim 1/\sqrt{h}$ and lead to similar fluctuations in δF. These fluctuations were argued to lead to an asymptotic decrease of test error as $1/h$, consistent with observations. Interestingly, the same scaling for the fluctuations holds in the feature-training regime, presumably reflecting the approximations expected from the Central Limit Theorem (CLT) when Eq. (1) is replaced by an integral.

As a consequence of these fluctuations, ensemble averaging output functions leads to an enhanced performance in both regimes, as shown in Fig. 4 *(c-d)*. We remark that:

(i) In each regime, the test error of the ensemble average is essentially independent of h. It implies that the variation of performance with h is only a matter of diminishing fluctuations in the over-parametrized case considered here (as shown below, we always fit all training data in these runs). It also supports that the plateau value of the ensemble-average performance we observe corresponds to the performance of single network in the $h \to \infty$ limit.

(ii) Interestingly, it is reported in (Geiger et al., 2019) that for a fixed α, the smallest test error of the ensemble average is obtained at some finite $h_{\min}$ beyond h^* implying that past $h_{\min}$ performance is decreasing with growing h. It can now be simply explained: at fixed α one goes from feature to lazy training as h increases, since $\sqrt{h}\alpha$ also increases and must eventually become much larger than one, leading to a change in performance.

(iii) For small values of $\sqrt{h}\alpha$ (smaller than 10^{-4}) the variance blows up. We leave the study of this regime for future works. Since we observe that the test error also greatly increases for these values of $\sqrt{h}\alpha$, this regime is of less interest.

5 Training dynamics differs in the two regimes

The network is able to learn features in the feature-training regime, while it cannot in the lazy-training regime because the NTK is frozen. To quantify feature training we measure the total relative variation of the kernel at the end of training: $\|\Theta(w) - \Theta(w_0)\|/\|\Theta(w_0)\|$, where the norm of a kernel is defined as $\|\Theta(w)\|^2 = \sum_{\mu,\nu\in\text{test set}} \Theta(w, x_\mu, x_\nu)^2$. This quantity is plotted in Fig. 5 *(a, left)* versus α for several widths h, and versus $\sqrt{h}\alpha$ in Fig. 5 *(a, right)*. The fact that the curves collapse indicates that $\alpha\sqrt{h}$ is the parameter that controls feature training. In particular, the crossover $\alpha^* \sim h^{-1/2}$ precisely corresponds to the point where the change of the kernel is of the order of the norm of the kernel at initialization. Moreover, in the feature-training regime we find:

$$\|\Theta(w) - \Theta(w_0)\|/\|\Theta(w_0)\| \sim (\sqrt{h}\alpha)^{-a}, \tag{18}$$

with an exponent $a \approx 1.3$. By contrast in the lazy-training regime $\|\Theta(w) - \Theta(w_0)\|/\|\Theta(w_0)\| \sim 1/(\sqrt{h}\alpha)$, as expected for what concerns the dependency in h (Jacot et al., 2019; Geiger et al., 2019; Lee et al., 2019). In Appendix F, we show that for a non-smooth activation function like the ReLU there is a third intermediary regime where $\|\Theta(w) - \Theta(w_0)\|/\|\Theta(w_0)\| \sim \alpha^{-1/2}$.

In Appendix A, we show in Fig. 8 an example of the evolution of the kernel (represented by its Gram matrix). We also present an interesting discovery: once the network has been trained, performing kernel learning with the NTK obtained at the end essentially leads to the same generalization error.

We finally investigate the temporal evolution of learning, known to be characterized by several time scales (Baity-Jesi et al., 2018). The rescaled loss $\alpha^2\mathcal{L}$ is shown in Fig. 5 *(b)*. As expected (Jacot et al., 2018; Chizat and Bach, 2019), in the lazy-training regime $\alpha^2\mathcal{L}(t)$ depends neither on α nor h. This is not true in feature training however. We define $t_{\mathcal{L}/2}$ as the time for which the loss reduced by a half. Our key finding, visible in Fig. 5 *(b)*, is that the learning curves are very sharp in the feature-training regime and they overlap when we

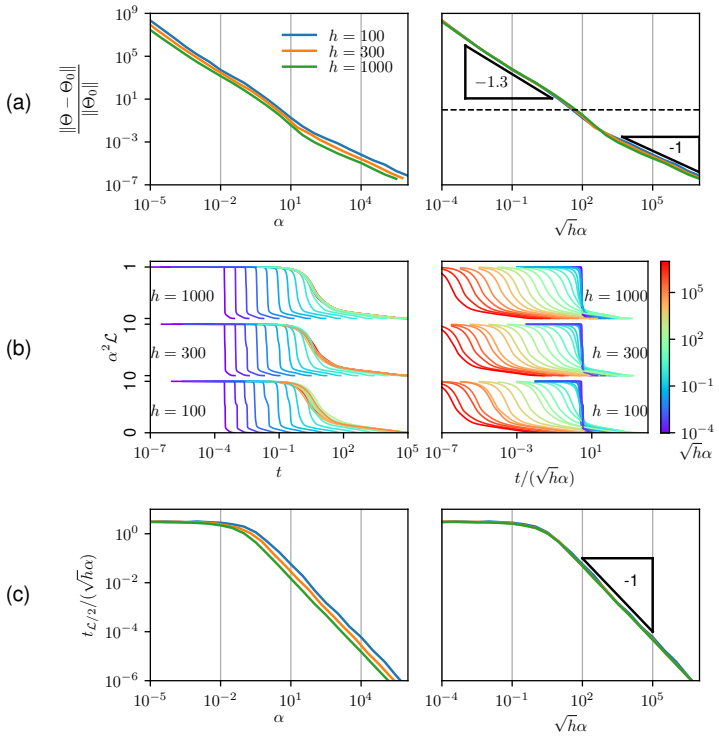


Figure 5: *(a)* Relative evolution of the kernel $\|\Theta(w) - \Theta(w_0)\|/\|\Theta(w_0)\|$ *v.s.* α (left) and $\sqrt{h}\alpha$ (right). *(b)* Rescaled loss $\alpha^2\mathcal{L}$ *v.s.* t (left) and $t/\sqrt{h}\alpha$ (right), for three different network's width and for different values of $\sqrt{h}\alpha$ indicated by the color bar. *(c)* The time for which the loss is reduced by half (divided by $\sqrt{h}\alpha$) *v.s.* α (left) and $\sqrt{h}\alpha$ (right). Notice that it reaches a plateau in the feature-training regime. For *(a)* and *(c)* each point is averaged over 10 initializations.

rescale the time axis properly. Fig. 5 *(b,c)* show that in the feature-training regime:

$$t_{\mathcal{L}/2} \sim \sqrt{h}\alpha. \tag{19}$$

In Section 6 we show that there is a time t_1 that marks the timescale below which the dynamics remains linear. In the feature-training regime we explain that $t_{\mathcal{L}/2}$ correspond to t_1 due to the nonlinearity that makes the dynamics evolve rapidly near t_1.

6 Arguments on kernel dynamics and regimes' boundary

The arguments proposed in this section do not have the status of mathematical proofs. They provide heuristic explanations for the scaling $t_1 \sim \alpha\sqrt{h}$ and $\alpha^* \sim 1/\sqrt{h}$, and support that the output function grows by a factor $\sqrt{h}$ before the dynamics become highly non-linear. A simple assumption on the nature of the ensuing non-linear dynamics leads to the prediction Eq. (18) with $a = 2/(1 + 1/L)$, which we view as a good approximation (not necessarily exact) of our observations.

Our starting point is that for the NTK initialization, we have at $t = 0$ that $\tilde{z}_\alpha = \mathcal{O}(1)$ and $\partial f/\partial \tilde{z}_\alpha = \mathcal{O}(1/\sqrt{h})$, where $\tilde{z}_\alpha$ is the preactivation of a hidden neuron. The second point can be derived iteratively starting from the last hidden layer. Denote by W^0 the weight matrix connected to the input, and W^ℓ the weight matrix connecting two hidden neurons in the $\ell-1$ and ℓ hidden layers respectively, and W^L the weight vector connected to the output. It is then straightforward to show using the chain rule and $\sigma'(\tilde{z}) = \mathcal{O}(1)$ that (see also (Arora et al., 2019)):

$$\frac{\partial f}{\partial W^0} = \mathcal{O}\left(\frac{1}{\sqrt{h}}\right); \quad \frac{\partial f}{\partial W^\ell} = \mathcal{O}\left(\frac{1}{h}\right); \quad \frac{\partial f}{\partial W^L} = \mathcal{O}\left(\frac{1}{\sqrt{h}}\right). \tag{20}$$

From which we deduce that:

$$\dot{W}^0 = \mathcal{O}\left(\frac{1}{\sqrt{h}\alpha}\right); \quad \dot{W}^\ell = \mathcal{O}\left(\frac{1}{h\alpha}\right); \quad \dot{W}^L = \mathcal{O}\left(\frac{1}{\sqrt{h}\alpha}\right) \tag{21}$$

by using the gradient-descent formula (i.e. $\dot{w} = -\nabla\mathcal{L}$).

Next, we consider how the neurons' preactivations evolve in time. From the composition of derivatives, one obtains $\dot{\tilde{z}}^{\ell+1} = h^{-1/2}(\dot{W}^\ell z^\ell + W^\ell \dot{z}^\ell)$. It is clear from Eq. (21) that $h^{-1/2}\dot{W}^\ell z^\ell = \mathcal{O}\left(\frac{1}{\sqrt{h}\alpha}\right)$. Concerning the product $W^\ell \dot{z}^\ell$, in the large-width limit it can be proven to be correctly estimated by considering that W^ℓ and $\dot{z}^\ell$ are independent (Dyer and Gur-Ari, 2019). (This result simply stems from the fact that the time-derivative of the preactivation of one neuron depends on all its h outgoing weights, and is therefore weakly correlated to any of them). From the central limit theorem, the vector $W^\ell \dot{z}^\ell$ is thus of order $\sqrt{h}\dot{z}^\ell$. Proceeding recursively from the input to the output we obtain:

$$\dot{\tilde{z}}^\ell = \mathcal{O}\left(\frac{1}{\sqrt{h}\alpha}\right). \tag{22}$$

We checked Eq. (22) numerically in Appendix D.

From Eq. (21) and Eq. (22) we expect that:

$$\forall t \ll t_1 \equiv \alpha\sqrt{h}, \;\; W^L(t) - W^L(0) = o(1); \quad \tilde{z}^\ell(t) - \tilde{z}^\ell(0) = o(1) \tag{23}$$

Thus for $t \ll t_1$, we are in the lazy-training regime where preactivations and weights did not have time to evolve, and we expect the kernel variations to be small (see (Mei et al., 2019) for a related discussion). Since the lazy-training regime finds a zero loss solution and stops in a time $\mathcal{O}(1)$ (see Section 2), if $t_1 = \alpha\sqrt{h} \gg 1$ the network remains in it throughout learning. Thus $\alpha^* \sim 1/\sqrt{h}$, as proposed for a single layer in (Chizat and Bach, 2019).

By contrast, if $\alpha\sqrt{h} \ll 1$ the dynamics has not stopped at times $t \sim t_1$, for which we have $W^L(t_1) - W^L(0) = \mathcal{O}(1)$ and $\tilde{z}(t_1) - \tilde{z}(0) = \mathcal{O}(1)$: both the preactivations and the weights of the last layer have changed significantly, leading to significant changes of $\nabla_w f$ and Θ. It is important to note that at $t \sim t_1$, the scale of the output function is expected to change. Indeed at initialization the output function, which is a sum made on the last layer of hidden neurons $f(w, x) = \frac{1}{\sqrt{h}} \sum_{i=1}^h W_i^L \sigma(\tilde{z}_i^L)$, is $\mathcal{O}(1)$ as expected from the CLT applied to h uncorrelated terms (Neal, 1996). However, for $t \sim t_1$ this independence does not hold anymore,

since the terms $W_i^L \sigma(\tilde{z}_i^L)$ have evolved by $\mathcal{O}(1)$ to change the function $f(w, x)$ in a specific direction. We thus expect these correlations to build up linearly in time for $t \in [0, t_1]$ and to ultimately increase the output by a factor $\sqrt{h}$ at $t \sim t_1$, as confirmed in Appendix C. Note that this effect does not appear at intermediate layers in the network, because the weights evolve much more slowly there as follows from Eq. (21).

Still, such an increase in the output is insufficient to find solutions deep in the feature-training regime, since $\alpha[f(w(t_1)) - f(w(0))] \sim \alpha\sqrt{h} \ll 1$. We propose that for activation functions that increase linearly at large arguments as those we use, the dynamics for $t \approx t_1$ approximately corresponds to an inflation of the weights along the direction $\dot{w}(t_1)$. Specifically, we define an amplification factor $\lambda(t) = \|w(t) - w(0)\|/\|w(t_1) - w(0)\|$, and assume for simplicity that this amplification is identical in each of the $L + 1$ layers of weights (we disregard in particular the fact that the last and first layer may behave differently, as discussed in (Arora et al., 2019)). By definition, $\lambda(t_1) = 1$. At the end t_2 of training, for activation functions that increase linearly at large arguments, we expect to have $\lambda(t_2) \sim [\alpha\sqrt{h}]^{-1/(L+1)}$ to ensure that $\alpha[f(w(t_2)) - f(w(0))] = \mathcal{O}(1)$. Gradient with respect to weights are increased by λ^L, leading to an overall inflation of the kernel:

$$\Theta(t_2) - \Theta(0) \sim \Theta(t_2) \sim \lambda^{2L} \sim [\alpha\sqrt{h}]^{-\frac{2}{1+1/L}} \tag{24}$$

leading to $a = 1.66$ consistent with Eq. (18). We have checked this prediction with success for shallow networks with $L = 2$, as shown in Appendix E.

7 Other experiments

We now check that our conclusions extend to other data sets, architectures and learning dynamics. In particular we consider under which circumstances feature training outperforms lazy training. Table 1 summarizes our results. The key observation is that which regime works best depends on the architecture and on the data. In particular, for the training set size 10^4 we focus on in this study (as it allows to study gradient descent in a reasonable time), we generally find that FC performs better under the lazy training regime. For the CNN architectures that we study, feature training tends to perform better.

MNIST and CIFAR10: We train the FC network defined in Section 2 on the MNIST and CIFAR10 datasets. CIFAR10 dataset contains $50000 + 10000$ 32×32 images in the trainset and testset, which are split into 10 classes. We reduced the trainset to 10000 images split into 2 classes (5 classes merges into 1) and the images are flatten into a vector of size 1024. In Fig. 6 we show the results. The picture is qualitatively similar to what we see for Fashion-MNIST.

Table 1: Performance for different setups.

Architecture	Dataset (binary, 10k)	Algorithm	Regime performing better
CNN (4 hidden layers)	CIFAR10	ADAM (batch size 32)	feature training
CNN (4 hidden layers)	Fashion-MNIST	ADAM (batch size 32)	Not clear (Fig. 7(b))
FC (3 hidden layers)	CIFAR10	ADAM (batch size 32)	lazy training
FC (3,9 hidden layers)	Fashion-MNIST	Gradient flow	lazy training
FC (3 hidden layers)	MNIST	Gradient flow	lazy training
FC (3 hidden layers)	EMNIST letters	Gradient flow	lazy training
FC (3 hidden layers)	CIFAR10	Gradient flow	lazy training
FC (5 hidden layers)	MNIST 10 PCA	Gradient flow	feature training

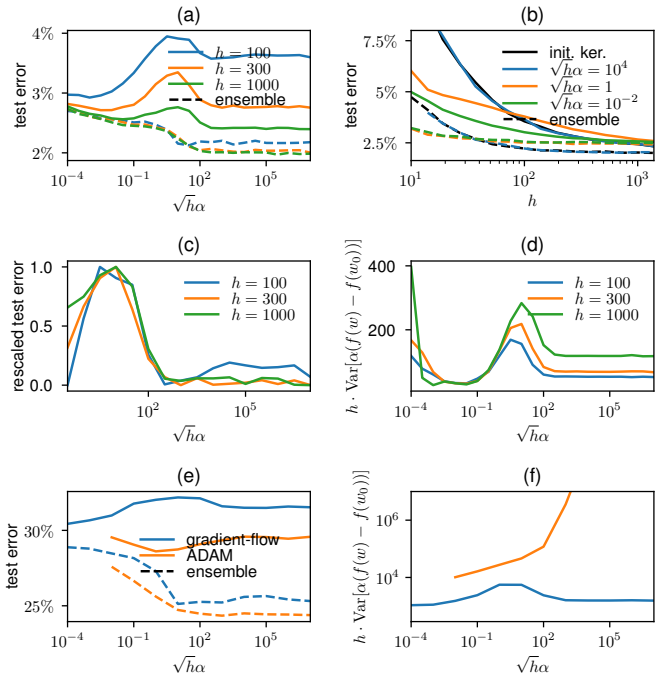


Figure 6: Binary classification on MNIST (*a-d*) and CIFAR10 (*e-f*) with the setup presented in Section 2. *(a)* MNIST test error (single shot and ensemble average over 20 instances) *v.s.* $\sqrt{h}\alpha$ for different widths h. *(b)* MNIST test error *v.s.* the width h for different values of $\sqrt{h}\alpha$. The black lines is the test error of the frozen NTK at initialization, limit that is recovered as $\alpha \to \infty$. Ensemble averages are computed over 10 instances. *(c)* Same data as in *(a)*: after rescaling the test error to be in $(0, 1)$ the curves collapse when plotted against $\sqrt{h}\alpha$. *(d)* The network's width times the variance of the output *v.s.* $\sqrt{h}\alpha$ for different widths h. This plot is computed for the MNIST dataset and is averaged over 20 initializations. *(e)* CIFAR10 test error (single shot and ensemble average over 20 instances) *v.s.* $\sqrt{h}\alpha$ for a network of width $h = 100$. *(f)* The network's width times the variance of the output *v.s.* $\sqrt{h}\alpha$. This plot is computed for the CIFAR10 dataset and is averaged over 20 initializations.

First ten principal components of MNIST PCA: As dataset we consider the projection of the handwritten digits in the MNIST dataset onto their first 10 principal components, obtained via principal-component analysis (PCA). This dataset is harder to fit than the original MNIST. Using the FC network defined in Section 2, we observe (see Fig. 7 *(a)*) that the test error for a single network as well as the ensemble-averaged test error are nearly identical, even somewhat smaller in the feature-training regime than in the lazy-training regime, contrarily to what was observed for the full MNIST dataset.

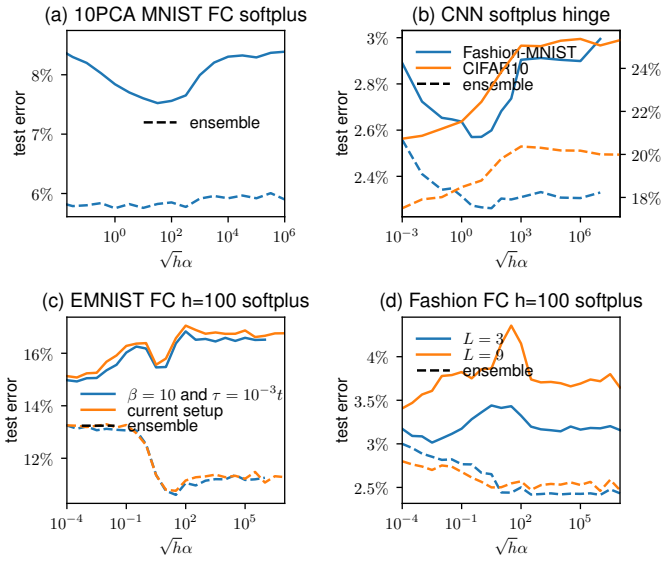


Figure 7: Test error and ensemble average test error *v.s.* $\sqrt{h}\alpha$ for different setup. *(a)* MNIST reduced to its first 10 PCA component trained on FC of width 100. *(b)* CIFAR10 trained on a CNN with the hinge loss and ADAM. *(c)* The letters from EMNIST. *(d)* Two different depth on Fashion-MNIST.

EMNIST: Using again the FC network defined in Section 2, we consider the pictures of handwritten letters in the EMNIST dataset. Fig. 7 *(c)* shows that the results are similar to what observed for Fashion-MNIST and MNIST. Here two setups are compared, the current setup (described in Section 2) and an alternative setup with $\beta = 10$ for the loss function and a value of τ proportional to the time.

CNN for CIFAR 10 and Fashion-MNIST: We train a convolutional neural network (CNN) with 4 hidden layers (with stride and padding). It has *Softplus* activation function and no biases. We initialize the parameters of the convolutional networks as standard Gaussians in such a way that the preactivations are of order unity. Our architecture has 4 hidden layers, and we use a stride of `/2` before the first layer and in the middle of the network. After the convolutions we average the spatial dimensions, and the last layer is a simple perceptron. The code is available on the repository. It is trained to classify images of the CIFAR10 dataset (32×32 images). Our trainset contains 10000 images split into 2 classes. Learning is achieved with the ADAM dynamics (Kingma and Ba, 2015). We observe, see Fig. 7 *(b)*, that individual and ensemble-averaged performance is better in the feature-training regime. For Fashion-MNIST, the optimal performance occurs in a narrow range of intermediate α.

Effect of Depth: In Fig. 7 *(d)* we compare networks with different depths, $L = 3$ with $L = 9$. The setup is defined in Section 2. We find that increasing depth does not change qualitatively the dependence of the

test error with $\sqrt{h}\alpha$. Quantitatively, depth has a very limited effect on the ensemble average performance, but does decrease the performance of individual networks — presumably indicating that depth increases fluctuations of the output function induced by random initialization.

8 Conclusion

We have shown that as the width h and the output scale α at initialization are varied, two regimes appear depending on the value of $\alpha\sqrt{h}$. In the feature-training regime features are learned (in the sense that the tangent kernel evolves during training), whereas in the lazy-training regime the dynamics is controlled by a frozen kernel. Our key findings are that: (i) In both regimes, fluctuations induced by initialization decrease with h, explaining why performance increases with the width. (ii) In feature training, the learning dynamics is linear for $t \ll t_1 \sim \sqrt{h}\alpha$, time at which the output of the model becomes of order $\sqrt{h}\alpha$. If $\sqrt{h}\alpha \ll 1$, in order to fit the data the dynamics enters a non-linear regime for $t \sim t_1$ that affects the magnitude of the kernel.

In treating these data sets, we have separated classes randomly into two groups. In the future, it would be interesting to test performance with a binary classification where the categories are intuitively meaningful. We have focused our comparison between these two learning regimes on a fixed data set size p, comparable in order of magnitude but smaller than the full data set. In a recent work(Paccolat et al., 2020) we performed that comparison as p varies for a CNN trained on MNIST. In that case, we found that for all p considered, feature learning outperforms lazy training in terms of generalization error ϵ. For the training curve $\epsilon \sim p^{-\beta}$ with $\beta_{\text{lazy}} = 1/3$ and $\beta_{\text{feature}} = 1/2$, indicting that the feature learning regime was improving faster with growing p than lazy training in relative terms.

On the empirical side, our work supports that studies of deep learning (e.g. on the role of regularization) should specify in which regime their networks operate, since it is very likely that it affects their results. On the theoretical side, there is little quantitative understanding on how much the performance should differ in two regimes. The results that we presented in Section 7 show that it depends on both the structure of the data and on the architecture of the network. Answering this point appears necessary to ultimately understand why deep learning works.

Acknowledgements

We thank Levent Sagun, Clément Hongler, Franck Gabriel, Giulio Biroli, Stéphane d'Ascoli for helpful discussions. We thank Riccardo Ravasio and Jonas Paccolat for proofreading. This work was partially supported by the grant from the Simons Foundation (#454953 Matthieu Wyart). M.W. thanks the Swiss National Science Foundation for support under Grant No. 200021-165509.

References

Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

Marco Baity-Jesi, Levent Sagun, Mario Geiger, Stefano Spigler, Gerard Ben Arous, Chiara Cammarota, Yann LeCun, Matthieu Wyart, and Giulio Biroli. Comparing dynamics: Deep neural networks versus glassy systems. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 314–323, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/baity-jesi18a.html.

Yamini Bansal, Madhu Advani, David D Cox, and Andrew M Saxe. Minnorm training: an algorithm for training overcomplete deep neural networks. *arXiv preprint arXiv:1806.00730*, 2018.

Lénaïc Chizat and Francis Bach. On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport. In *Advances in Neural Information Processing Systems 31*, pages 3040–3050. Curran Associates, Inc., 2018.

Lenaic Chizat and Francis Bach. A Note on Lazy Training in Supervised Differentiable Programming. working paper or preprint, February 2019. URL https://hal.inria.fr/hal-01945578.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On Lazy Training in Differentiable Programming. In *NeurIPS 2019 - 33rd Conference on Neural Information Processing Systems*, Vancouver, Canada, December 2019. URL https://hal.inria.fr/hal-01945578.

Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1-nGgWC-.

Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eK3i09YQ.

Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. *arXiv preprint arXiv:1909.11304*, 2019.

Mario Geiger, Stefano Spigler, Stéphane d'Ascoli, Levent Sagun, Marco Baity-Jesi, Giulio Biroli, and Matthieu Wyart. The jamming transition as a paradigm to understand the loss landscape of deep neural networks. *arXiv preprint arXiv:1809.09349*, 2018.

Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *arXiv preprint arXiv:1901.01608*, 2019.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 8580–8589, USA, 2018. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=3327757.3327948.

Arthur Jacot, Franck Gabriel, and Clément Hongler. The asymptotic spectrum of the hessian of dnn throughout training. *arXiv preprint arXiv:1910.02875*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Jae Hoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *ICLR*, 2018.

Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layers neural networks. *arXiv preprint arXiv:1804.06561*, 2018.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. *arXiv preprint arXiv:1902.06015*, 2019.

Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. 2019. URL `https://openreview.net/forum?id=HkgmzhC5F7`.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996. ISBN 0387947248.

Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of optimization and implicit regularization in deep learning. *arXiv preprint arXiv:1705.03071*, 2017.

Phan-Minh Nguyen. Mean field limit of the learning dynamics of multilayer neural networks. *arXiv preprint arXiv:1902.02880*, 2019.

Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=B1g30j0qF7`.

Jonas Paccolat, Leonardo Petrini, Mario Geiger, Kevin Tyloo, and Matthieu Wyart. Geometric compression of invariant manifolds in neural nets, 2020.

Daniel S Park, Jascha Sohl-Dickstein, Quoc V Le, and Samuel L Smith. The effect of network width on stochastic gradient descent and generalization: an empirical study. *arXiv preprint arXiv:1905.03776*, 2019.

Grant M Rotskoff and Eric Vanden-Eijnden. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*, 2018.

Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks. *arXiv preprint arXiv:1805.01053*, 2018.

Stefano Spigler, Mario Geiger, Stéphane d'Ascoli, Levent Sagun, Giulio Biroli, and Matthieu Wyart. A jamming transition from under-to over-parametrization affects loss landscape and generalization. *arXiv preprint arXiv:1810.09665*, 2018.

Christopher KI Williams. Computing with infinite networks. In *Advances in neural information processing systems*, pages 295–301, 1997.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

A Frozen NTK dynamics

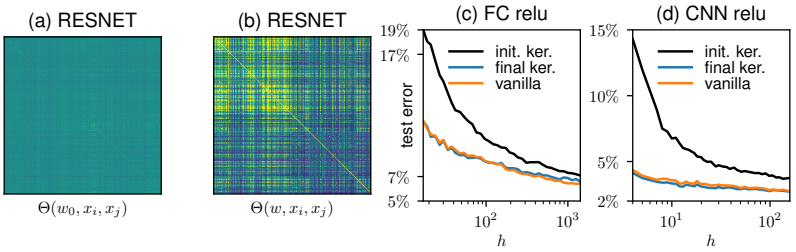


Figure 8: Gram matrix of the test set at initialization $\Theta(w_0, x_\mu, x_\nu)$ *(a)* and at the end of training $\Theta(w, x_\mu, x_\nu)$ *(b)*, for a wide-resnet 28x10 architecture (Zagoruyko and Komodakis, 2016) ($L = 25$ hidden layers) trained on a binary version of CIFAR10. The first half of the indices $\mu = 1 \dots {}^{n}/_{2}$ has label $y = 1$ and the other half has label $y = -1$. The kernel inflates during learning in a way that depends on the two classes. See Appendix A for a description of the architecture. *(c-d)* Test error *v.s.* the width h for the regular dynamics, the dynamics with the frozen kernel at initialization and the dynamics with the frozen kernel of the end of training. The training performance is captured by the kernel.

Let us consider the first order approximation $\tilde{f}_{w_1}(w, x)$ of a model $f(w, x)$ around $w = w_1$,

$$\tilde{f}_{w_1}(w, x) = \nabla_w f(w_1, x) \cdot w. \tag{25}$$

For instance, w_1 can be the value at initialization or at the end of another dynamics. We can then train this linearized model keeping the gradients fixed:

$$\dot{\tilde{f}}_{w_1}(w) = \nabla_w f(w_1) \cdot \dot{w}, \tag{26}$$

where $\dot{w}$ depends on the gradient descent procedure. Using gradient descent,

$$\dot{\tilde{f}}_{w_1}(w) = -\nabla_w f(w_1) \cdot \frac{1}{n} \sum_{(x,y)\in\mathcal{T}} \ell'(\tilde{f}_{w_1}(w, x), y) \nabla_w f(w_1, x) \tag{27}$$

By introducing the kernel we can rewrite the previous equation as

$$\dot{\tilde{f}}_{w_1}(w) = -\frac{1}{n} \sum_{(x,y)\in\mathcal{T}} \ell'(\tilde{f}_{w_1}(w, x), y) \Theta(w_1, x) \tag{28}$$

where Θ is the neural tangent kernel defined in Eq. (5). We call these equations the *frozen kernel dynamics*.

The results presented in Fig. 5 state that the network learns a kernel during the training dynamics, and that this learned kernel coincides with the frozen kernel in the lazy-training regime as $\sqrt{h}\alpha \to \infty$. Another way to see that the kernel changes during training is to plot the so-called Gram matrix of the frozen kernel, namely the matrix $(\Theta(w, x_\mu, x_\nu))_{\mu,\nu\in\text{test set}}$: in Fig. 8 *(a-b)* we show the Gram matrix of the neural tangent kernel evaluated before and after training, where it is clear that there is an emergent structure that depends on the dataset.

The architecture used in in Fig. 5 is a resnet based on (Zagoruyko and Komodakis, 2016). We use no batch normalization and initialization is as in our fully-connected networks. The code describing the architecture is available in the supplementary material.

It is interesting to test if the performance of deep networks after learning is entirely encapsulated in the kernel it has learned. We argue that indeed this is the case, and to make our point, we proceed as follows. At any time t during training, we can compute the instantaneous neural tangent kernel $\Theta(w(t))$ as in Eq. (5); then, we perform a frozen kernel dynamics using that instantaneous kernel, and we evaluate its performance on the test set. In Fig. 8 *(c-d)* we plot the test error of a network with $\alpha = 1$ (referred to as "vanilla") and compare it to the test error of the frozen kernel, both at initialization ($t = 0$) and at the end of training. Quite remarkably $\tilde{f}_{w(t_2)}$ achieves the full performance of f.

B Dynamics of the Weights

In Fig. 9 we plot the rescaled evolution of the parameters $\sqrt{h}\|w - w_0\|/\|w_0\|$ versus α *(left)* and versus $\sqrt{h}\alpha$ *(right)* showing that the curves collapse. We find:

$$\frac{\|w - w_0\|}{\|w_0\|} \sim \frac{1}{h\alpha} \tag{29}$$

in the lazy-training regime. It is expected from Eq. (21), and the fact that the dynamics lasts $\mathcal{O}(1)$ in this regime, jointly implying that the $\mathcal{O}(h^2)$ internal weights evolve by $\mathcal{O}(1/(h\alpha))$. Finally

$$\frac{\|w - w_0\|}{\|w_0\|} \sim \frac{(\sqrt{h}\alpha)^{-b}}{\sqrt{h}} \tag{30}$$

in the feature-training regime, where $b \approx 0.23$ is compatible with $1/(L+1) \approx 0.17$ as proposed in Section 6. Note that the denominator $\sqrt{h}$ is also expected from Section 6, where it corresponds to the term $\|w(t_1) - w(0)\|$ entering in the definition of λ. It comes from the fact that $\|w(t_1) - w(0)\| \sim \sqrt{h}$, as follows from the $\mathcal{O}(h^2)$ internal weights evolving by $\mathcal{O}(1/\sqrt{h})$ on the time scale t_1, as can be deduced from Eq. (21).

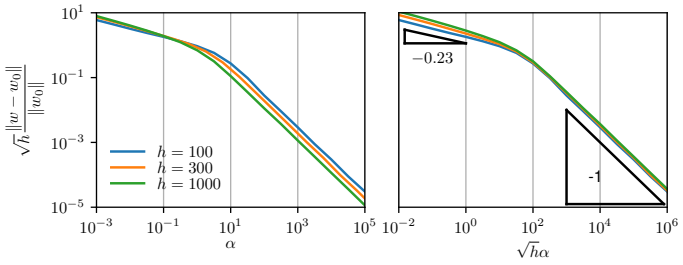


Figure 9: Relative evolution of the parameters $\sqrt{h}\|w-w_0\|/\|w_0\|$ *v.s.* α (left) and $\sqrt{h}\alpha$ (right). Each measure is averaged over 10 initializations.

C Dynamics of the Output function

To measure the amplitude of the output of a network f we define its norm as follow

$$\|f(w)\| = \sqrt{\langle f(w, x_\mu)^2 \rangle_{\mu \in \text{test}}} \tag{31}$$

In Section 6 we argued that the dynamics is linear for $t \ll t_1 \sim \alpha\sqrt{h}$. Fig. 10 (a,c,d) confirms that the dynamics, characterized by $\|f(w_t) - f(w_0)\|$, is indeed linear on a time scale of order t_1, independently of the value of α as shown in Fig. 10 (a) or h as shown in Fig. 10 (c,d).

Another important result of Section 6 is that at the end of the linear regime, the output has increased by a relative amount $\sim \sqrt{h}$. This result is confirmed in Fig. 10(b) showing that $\frac{1}{\sqrt{h}}\|f(w_t)\|$ is independent of h for $t \sim t_1$.

These two facts taken together imply:

$$\|f(w_t) - f(w_0)\| \sim \frac{t}{t_1}\sqrt{h}, \quad t \ll t_1 \text{ in feature training.} \tag{32}$$

This prediction is confirmed in Fig. 10 (d) showing $\alpha\|f(w_t) - f(w_0)\| \sim t/t_1(\sqrt{h}\alpha)$ which must behave as t/t_1 if $(\sqrt{h}\alpha)$ is hold fixed, as is the case in this figure.

D Preactivation evolution

Fig. 11 shows the amplitude of $\dot{\tilde{z}}$ at initialization. We measured it using finite-difference method by applying a single gradient descent step.

E Shallow network

In order to verify the depth dependence of our heuristic predictions about the powerlaw in α of the quantities $\|\Theta - \Theta_0\|$ and $\|w - w_0\|$, we reran the experiment with 2 hidden layers ($L = 2$) with the fully-connected network and *Softplus*. In Table 2 we summarize the exponent found numerically, they are compatible the our predictions.

Observable	$L = 5$	$L = 2$	Prediction
$\|\Theta - \Theta_0\|$	1.7 (1.66)	1.25 (1.33)	$\frac{2}{1+1/L}$
$\|w - w_0\|$	0.23 (0.166)	0.35 (0.333)	$\frac{1}{1+L}$

Table 2: Powerlaw dependence in α, measure and prediction (in parenthesis) of the exponent a where $O \sim \alpha^{-a}$ for $\alpha \ll 1$

F ReLU activation function

Fig. 12 shows the evolution of the kernel as a function of $\sqrt{h}\alpha$ for a network with *ReLU* activation function. Differently from the *Softplus* case (see Fig. 5 (a)), here we observe the existence of three regimes, each characterized by a different power law. The intermediate regime with slope $-1/2$ is not present for *Softplus*, and it is compatible with the following explanation. The *ReLU* function $x \mapsto \max(0, x)$ is non differentiable in $x = 0$. It implies that $w \mapsto f(w)$ is not differentiable. For a finite dataset, $w \mapsto \{f(w, x_\mu)\}_\mu$ is

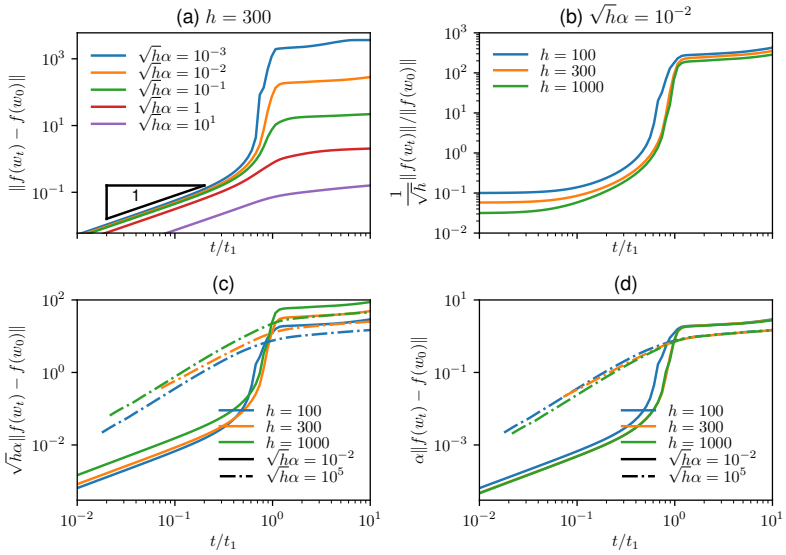


Figure 10: Different measures of the network norm *v.s.* t/t_1 for (a) a fixed width h and various α, (b,c,d) a fixed value of $\sqrt{h}\alpha$ and various h. Here t_1 is the time at which the loss reduced by half. The network used here has $L = 2$ hidden layers and uses a *Softplus* activation function. Each curve is averaged along the y axis for 10 realizations.

differentiable only on small patches. As we can see in Fig. 12 for large enough α, w evolution is so small that f remains in a differentiable patch and we get the predicted result of slope -1. But for intermediate values of α, the network change patches a number of times proportional to α^{-1} and assuming that each changes in the kernel induced by these changes of patch are not correlated, their sum scales with $\alpha^{-1/2}$.

G Gradient flow verification

As we can see in Fig. 13, the constraint we put on the relative difference of gradients for the dynamics ensure that the relative difference of the output is smaller than 10^{-3} when the constraint is divided by 100. Also we see that the time of convergence is roughly doubled when the constraint is divided by 100.

H Variance and the size of the dataset in the lazy-training regime

In Fig. 14 the variance is shown as a function of the width, for different sizes of dataset n. (i) We see the variance reaching the asymptotic behavior of h^{-1} when n is small enough. (ii) The data also suggest that the variance grows with the size of the trainset like $\sqrt{n}$.

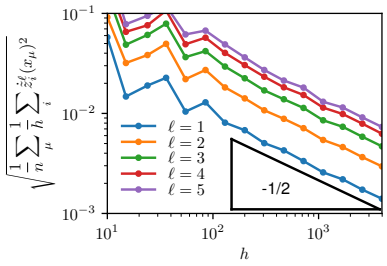


Figure 11: $\dot{\tilde{z}}$ at $t = 0$ *v.s.* h for different layers. Each measure is averaged over 5 networks. The network used here has $L = 5$ hidden layers and uses a *Softplus* activation function.

$$\mathrm{Var} f(w,x) = \Big\langle \big[f(w_i,x_\mu) - \bar{f}(x_\mu)\big]^2 \Big\rangle_{\substack{\mu\in\text{test} \\ i\in\text{ensemble}}} . \tag{33}$$

I $f - f_0$ versus f

Fig. 15 shows the difference between the model $F(w,x) = \alpha f(w,x)$ and $F(w,x) = \alpha(f(w,x) - f(w_0,x))$. Notice that removing the value of the output function at initialization drastically improves the performance of the network for large values of α. The generalization error is the same for small α.

J Effect of biases

In Fig. 16 we test the helpfulness of introducing biases in hidden layers, by comparing the test error and the variance of the output function in networks that have or have no biases. It turns out that biases are negligible in the present setting. The setup is described in Section 2. A possible reason to use biases would be that ReLU networks without biases are homogeneous functions. This mean that two data x_μ, x_ν that are *aligned*, in the sense that $x_\mu = |\lambda| x_\nu$, cannot have different labels. This problem can often be neglected for two reasons: first, in practice the datasets are normalized, so that two points are aligned only if they are identical; second, the pictures in some datasets typically have constant pixels. For instance, in MNIST the top-left pixel in every picture is black. Constant pixels behave effectively as a bias in any hidden neuron.

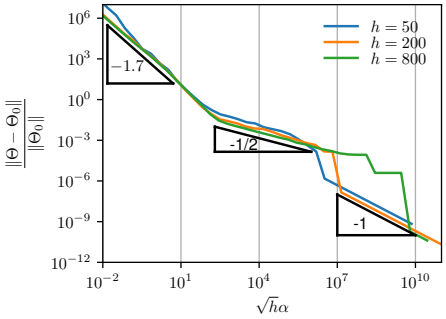


Figure 12: $\frac{\|\Theta-\Theta_0\|}{\|\Theta_0\|}$ *v.s.* $\sqrt{h}\alpha$ for different heights. Each measure is averaged over 3 networks. The network used here has $L = 5$ hidden layers and uses a *ReLU* activation function.

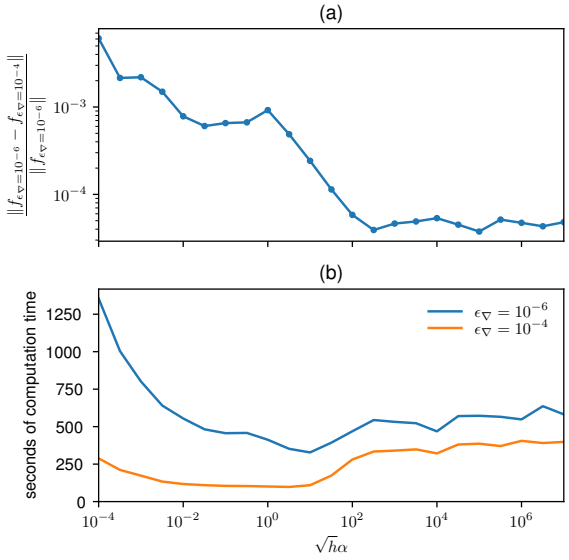


Figure 13: *(a)* Relative difference of the output for two different values of ϵ_∇ *v.s.* $\sqrt{h}\alpha$. *(b)* Relative difference of the output for two different momentum τ *v.s.* $\sqrt{h}\alpha$. *(c)* Computation time *v.s.* $\sqrt{h}\alpha$, for two different values of ϵ_∇. *(d)* Computation time *v.s.* $\sqrt{h}\alpha$, for two different momentum τ.

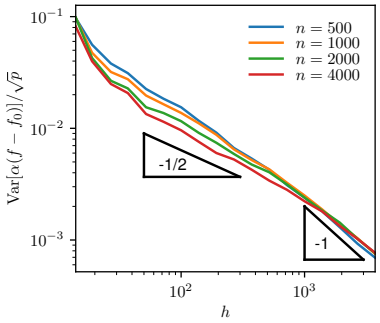


Figure 14: Variance of the output in the lazy-training regime ($\sqrt{h}\alpha = 10^6$) *v.s.* the network's width h for different size of trainset n. We believe that eventually all the curves asymptote with a slope -1. This asymptote is reached earlier for smaller n.

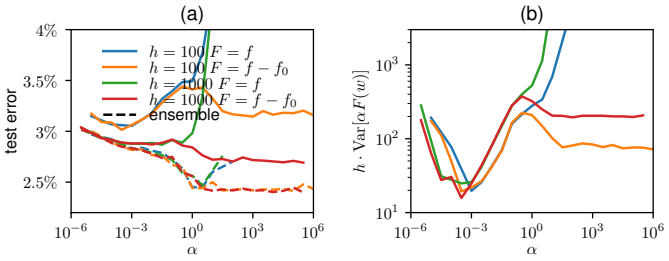


Figure 15: Comparison between the models $\alpha F(w, x) = \alpha(f(w) - f(w_0))$ and $\alpha F(w, x) = \alpha f(w, x)$. The setup is described in Section 2. *(a)* Test error (with ensemble average in dashed line) *v.s.* α for a network of two widths and for the two models. (averaged over 10 initializations) *(b)* The network's width time the variance of the output *v.s.* α. (averaged over 10 initializations). In the feature regime, both models behave similarly because $f(w_0)$ is negligible compare to $f(w)$.

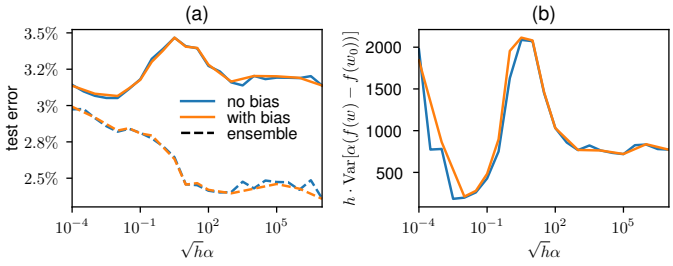


Figure 16: Same as Fig. 3 but with biases. *(a)* Test error (single shot and ensemble average) *v.s.* $\sqrt{h}\alpha$ for a network of width $h = 100$, averages are over 20 initializations. *(d)* We plot the width times the variance of the output function *v.s.* $\sqrt{h}\alpha$, averaged over 10 initializations.

The following paper is the preprint version of Geiger et al. (2021). It is a review article.

Candidate contributions The candidate contributed to this papers by doing the numerical experiments of Figures 4 and 5.

Perspective: A Phase Diagram for Deep Learning unifying Jamming, Feature Learning and Lazy Training

Mario Geiger[a], Leonardo Petrini[a], and Matthieu Wyart[a]

[a]Institute of Physics, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland

Abstract

Deep learning algorithms are responsible for a technological revolution in a variety of tasks including image recognition or Go playing. Yet, why they work is not understood. Ultimately, they manage to classify data lying in high dimension – a feat generically impossible due to the geometry of high dimensional space and the associated *curse of dimensionality*. Understanding what kind of structure, symmetry or invariance makes data such as images learnable is a fundamental challenge. Other puzzles include that (i) learning corresponds to minimizing a loss in high dimension, which is in general not convex and could well get stuck bad minima. (ii) Deep learning predicting power increases with the number of fitting parameters, even in a regime where data are perfectly fitted. In this manuscript, we review recent results elucidating (i,ii) and the perspective they offer on the (still unexplained) curse of dimensionality paradox. We base our theoretical discussion on the (h, α) plane where h is the network width and α the scale of the output of the network at initialization, and provide new systematic measures of performance in that plane for MNIST and CIFAR 10. We argue that different learning regimes can be organized into a phase diagram. A line of critical points sharply delimits an under-parametrised phase from an over-parametrized one. In over-parametrized nets, learning can operate in two regimes separated by a smooth cross-over. At large initialization, it corresponds to a kernel method, whereas for small initializations features can be learnt, together with invariants in the data. We review the properties of these different phases, of the transition separating them and some open questions. Our treatment emphasizes analogies with physical systems, scaling arguments and the development of numerical observables to quantitatively test these results empirically. Practical implications are also discussed, including the benefit of averaging nets with distinct initial weights, or the choice of parameters (h, α) optimizing performance.

1 Introduction

One of the prerequisites of human or artificial intelligence is to make sense of data that often lie in large dimension. A classical case is computer vision, where one seeks to classify the content of a picture [1] whose dimension is the number of pixels. In the supervised setting considered in this review, algorithms are trained from known, labeled data. For example, one is given a training set of one million pictures of cats and dogs, and knows which is which. The goal is to build an algorithm that can learn a rule from these examples, and

predict if a new picture presents a cat or a dog. After sixty years of rather moderate progress, machine learning is undergoing a revolution. Deep learning algorithms [2], which are inspired from the organisation of our visual cortex, are now remarkably successful at a wide range of tasks including speech [3] and text recognition [4], self-driving cars [5] and beating the best humans at Go [6] or video games [7]. Yet, there is very limited understanding as to why these algorithms work. As explained below, there are several reasons why deep learning in particular and supervised learning in general should not work. Most prominently, the *curse of dimensionality* associated with the geometry of space in large dimension prohibits learning in a generic setting. If high dimensional data can be learned, then there must have a lot of structure, invariances and symmetries. Understanding what is the nature of this structure and how it can be harvested by neural nets with suitable architectures is a challenge of our times.

In the part 1.1 of this introduction, we review the basic procedure of supervised learning with deep nets, together with the quantification of its success via a learning curve exponent. In part 1.2, we discuss several reasons making this success surprising. It includes the curse of dimensionality mentioned above, the putative presence of the bad minima in the loss landscape and the fact that deep learning typically works in an over-parametrized regime where the number of fitting parameters is much larger than the number of data. In part 1.3, we review two recent ideas seeking to address these paradoxes. First, in the limit where the network width (and thus the number of parameters) diverges, deep learning converges to two distinct, well-defined algorithms (*lazy training* and *feature learning*) depending on the scale of initialization. In each regime, a global minimum of the loss is found. Second, in the context of image classification it was hypothesized that deep learning efficiently deals with the curse of dimensionality because neural nets learn to become insensitive to smooth deformations of the image.

These two body of work raise various questions, detailed below. In a nutshell: (i) Is there a critical number of parameters where one enters in the over-parametrized regime and bad minima in the loss disappear? What is the nature of this transition, and the geometry of the loss-landscape in its vicinity? (ii) Why does performance tends to improve asymptotically with the number of parameters even past this transition? (iii) Once in the over-parametrized regime, where does the cross-over between lazy training and feature learning takes place? Which regime performs better, and how does it depend on the data structure and network architecture? (iv) How are simple invariants learnt in the feature learning regime, and how does it affect the learning curve exponent? The goal of this manuscript, laid out in the part 1.4 of this introduction, is both to review recent results addressing these questions in the context of classification in deep nets, as well as to provide new systematic empirical data unifying these results into a phase diagram.

1.1 Presentation of deep learning and quantification of its successes

Signal propagation in some architectures: Deep learning is a fitting procedure, in which the functional form used to interpolate the data depends on many parameters, and can be represented iteratively. State-of-the-art (SOTA) architectures characterising this functional form can be extremely complex, here we restrict ourselves to essential properties. We denote by $\tilde{f}_\theta(\mathbf{x})$ the output of a network corresponding to an input $\mathbf{x}$, parametrized by θ. We reserve the notation $f_\theta(\mathbf{x})$ for the prediction model, the relation between the two will be defined in Eq.8. For fully connected nets (FC) – which, from a theoretical point of view, are the

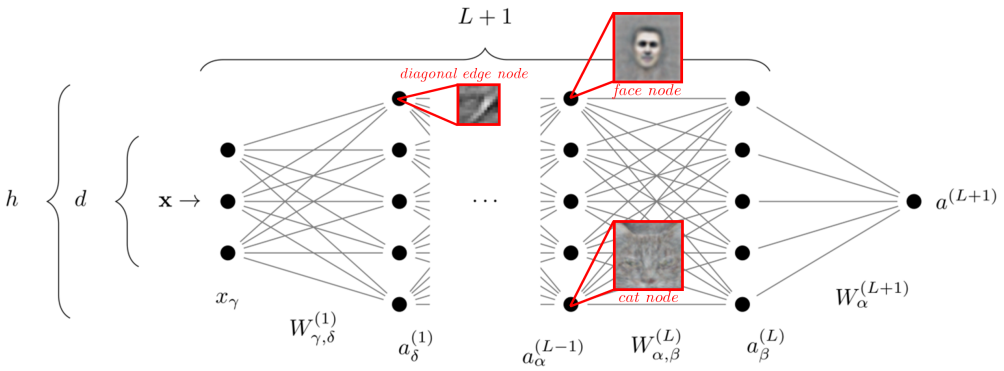


Figure 1: Architecture of a fully-connected network with L hidden layers of neurons of constant size h. Points indicate neurons, connections between them are characterized by a weight. Biases are not represented here. The W's are the weights, $\mathbf{x}$ is the input, and the a's are the preactivations as defined in Equations (1-3). Taken from [8]. In red: Neural representation of the data at different depths (illustrative). Neurons in the first layers respond to local, simple features, such as edges. Deeper in the network, neurons respond to more and more high-level features such as cats or human faces. Adapted from [9].

most studied architectures – the output function can be written recursively as:

$$\tilde{f}_\theta(\mathbf{x}) \equiv a^{(L+1)}, \tag{1}$$

$$a^{(i)}_\beta = \sum_\alpha \frac{1}{\sqrt{h}} W^{(i)}_{\alpha,\beta}\, \sigma\left(a^{(i-1)}_\alpha\right) + B^{(i)}_\beta, \tag{2}$$

$$a^{(1)}_\beta = \sum_\alpha \frac{1}{\sqrt{d}} W^{(1)}_{\alpha,\beta}\, x_\alpha + B^{(1)}_\beta, \tag{3}$$

where $a^{(i)}_\alpha$ is the preactivation of neuron α, located at depth i and L the total number of layers. We consider networks with a fixed number of hidden layers L, each of size h. In the following, given that the total number of parameters follows $N \sim Lh^2$, we'll use both h and N to refer to the network size. In our notation the set of parameters θ includes both the weights $W^{(i)}_{\alpha,\beta}$ and the biases $B^{(i)}_\alpha$. $\sigma(z)$ is the non-linear activation function, e.g. the ReLU $\sigma(z) = \max(z, 0)$. As is commonly done, we use the "LeCun initialization" of the weights, for which they are scaled by a factor $1/\sqrt{h}$. It ensures that the limit of infinite width is not trivial, see e.g. discussions in [10]. We represent the network in Fig. 1.

Convolutional neural nets (CNNs), inspired from the primate brain, perform much better than FCs for a variety of tasks including image classification. Each hidden layer is composed of a number of channels, each representing the data as a feature map – itself an image. CNNs perform the same operations at distinct locations in the image, and enforce that these operations are local. CNNs are thus more constrained than FCs, and they can be obtained from the latter by imposing that some weights are identical (to enforce translational invariance) and others zero (to enforce locality).

Learning Dynamics Weights are learnt by fitting the training set, which is done by minimising a loss or cost function $\mathcal{L}$. To simplify notations, we consider a binary classification problem, with a set of P distinct training data denoted $\{(\mathbf{x}_\mu, y_\mu)\}_{\mu=1}^P$. The vector $\mathbf{x}_\mu$ is the input, which lives in a d-dimensional space, and $y_\mu = \pm 1$ is its label [1]. $\mathcal{L}$ is a sum on all the training set of how each datum is well-fitted:

$$\mathcal{L}(\theta) = \frac{1}{P}\sum_{\mu=1}^{P} \ell(y_\mu f(\mathbf{x}_\mu)), \tag{4}$$

where f is a model and ℓ is a decreasing function of its argument. Popular choices include the linear ($\gamma = 1$) or quadratic ($\gamma = 2$) hinge loss $\ell(y_\mu f(\mathbf{x}_\mu)) = \max(0, \Delta_\mu)^\gamma$ with $\Delta_\mu \equiv 1 - y_\mu f(\mathbf{x}_\mu)$, or the cross-entropy $\ell(y_\mu f(\mathbf{x}_\mu)) = \log(1 + \exp(-y_\mu f(\mathbf{x}_\mu)))$. The hinge loss has the advantage that bringing the loss Eq.4 to zero is equivalent to satisfying a set of constraints $\Delta_\mu < 0$, corresponding to fitting the data by a margin unity. This fact is the basis for the analogy with other satisfiability problems encountered in physics, as discussed below. This choice however does not influence significantly performance for SOTA architectures on usual benchmarks [8].

Different procedures can then be used to minimise $\mathcal{L}$, starting from a random initialization of the weights (an important fact to keep in mind). In particular, gradient descent (GD) or stochastic gradient descent (SGD) are commonly used. For SGD, only a (changing) subset of the terms entering the sum of Eq.4 is considered at each minimisation time step. Various tricks can improve performance depending on the data considered, including early stopping (minimisation is stopped before the loss hits bottom) or weight decay (terms are then added to the loss to prevent weights to become too large).

Performance Once learning is done, generalization performance can be estimated from a test set (data not used to train, but whose distribution is identical to the training set) by computing the probability to misclassify one new datum, the so-called test error ϵ. How many data are needed to learn a task is characterized by the learning curve $\epsilon(P)$ [2]. Generally, it is observed that the test error is well described by a power law decay $P^{-\beta}$ in the range of training set size P available[3] with an exponent β that depends jointly on the data and the algorithm chosen. In [11], β is reported for SOTA architecture for several tasks: in *neural-machine translation* $\beta \approx 0.3$–0.36; in *language modeling* $\beta \approx 0.06$–0.09; in *speech recognition* $\beta \approx 0.3$ and in *image classification* (ImageNet) $\beta \approx 0.3$–0.5.

Deep nets learn a hierarchical representation of the data Once learning took place, it is possible to analyse to which features in the data neurons respond mostly to [9, 12]. Strikingly, findings are very similar to what is found in the visual cortex of primates: neurons learn to respond to more and more abstract aspects of the data as the signal progresses through the network, as illustrated in Fig.1. This observation is believed to be an important aspect of why deep learning works, yet understanding how abstract features are learnt dynamically, and how much it contributes to performance, remains a challenge.

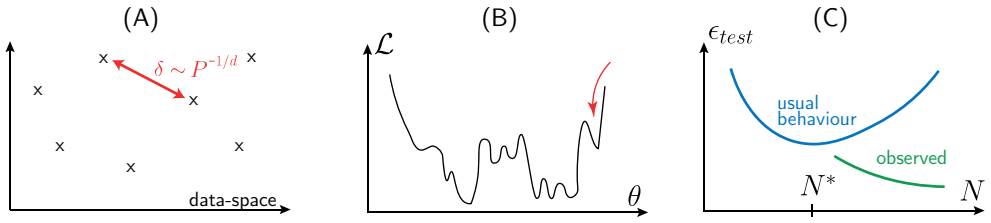


Figure 2: A: Curse of dimensionality. In high dimensions, every point lies far away from its neighbours, forbidding classification based on distances alone. B: Sketch of the loss landscape. Starting from a random initialization, the dynamics could get stuck in a bad minimum of the loss. C: Usual behaviour (blue line) of the test error as a function of the number of parameters, expected to be minimal when N is of order of the smallest value N^* where data can be fitted. Green: observations indicate that for deep learning, increasing the number of parameters generally improves behaviour, even in a regime where data are perfectly fitted.

1.2 Why deep learning should not work

Curse of dimensionality General arguments suggest that β should be extremely small — and learning thus essentially impossible — when the dimension d of the data is large, which is generally the case in practice. For example in a regression task, if the only assumption on the target function is that it is Lipschitz continuous, then the test error cannot be guaranteed to decay faster than with an exponent $\beta \sim 1/d$ [13]. This *curse of dimensionality* [14] stems from the geometrical fact that the distance δ among nearest-neighbour data points decays extremely slowly in large d as $\delta \sim P^{-1/d}$, as depicted in the Fig.2.A. Thus, interpolation or classification methods based on distances are expected to be very imprecise for generic data.

Bad minima in the loss landscape Another problem, which made deep learning less popular in the 90's, concerns learning. The loss function has no guarantees to be convex, and the parameters space is high-dimensional. What prevents then the learning dynamics to get stuck in poorly performing minima with high loss, as sketched in Fig.2.B? In other words, under which conditions can one guarantee that training data are well fitted? Is the loss landscape similar to the energy landscape of glassy systems in physics, where the number of minima is exponential in the number of degrees of freedom [15, 16]?

Over-parametrization Finally, neural nets are often trained in the *over-parametrized* regime, where the number of parameters N is significantly larger than the number of data points P. In that regime, their capacity is very large: they can fit the training set even if labels are randomized [17]. Statistical

[1] It is straightforward to extend the discussion to multi-class classification problem, as well as to regression.
[2] A different network is trained for every P.
[3] For data sets where a finite fraction of the training set is mislabeled, one expects the test error to eventually plateau to a finite value. However, such a saturation of the test error is not seen in practice for various benchmarks, suggesting that such a noise is small in magnitude.

learning theory then gives no guarantees that over-fitting does not occur, and that the model learnt has any predictive power. Indeed from statistics text books one expects a bell-shape learning curve relating the test error to the number of parameters of the model, as depicted in Fig.2.C. A very puzzling aspect of deep learning is that increasing N passed the point where all data are perfectly fitted does not destroy predictive power, but instead *improves* it [18–21] as shown in Fig.2.

1.3 Current insights on these paradoxes

No bad minima in over-parametrized networks Recently, it was realised that the landscape of deep learning is not glassy after all, if the number of parameters is sufficient. Theoretical works that focus on nets with a huge number of parameters (and one hidden layer), and empirical work with more realistic architectures [22–25], support that the loss function is characterized by a connected level set: any two points in parameter space with identical loss are connected by a path in which the loss is constant. Moreover, empirical studies of the curvature of the loss landscape (captured by the spectrum of the Hessian) [26–28] and of SGD dynamics [29, 30] reveal that the landscape displays a large number of flat directions, even at its bottom. It is not the case for under-parametrized networks [30].

These observations raise key questions. *Is there a phase transition in the landscape geometry as the number of parameters grows? If so, what is its universality class? How does it affect performance?*

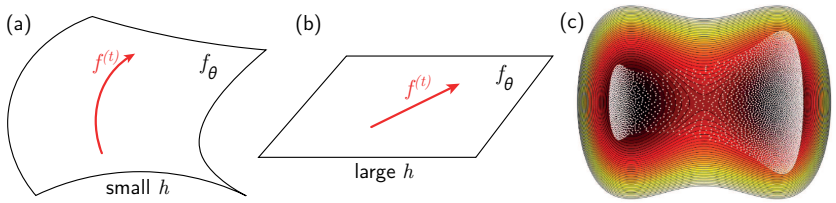


Figure 3: A: sketch of the dynamics in function space for a finite width net. As weights change the function evolves along the black manifold of expressible functions, which is curved. B: in the infinite width limit, this manifold becomes flat even though the function evolves by $\mathcal{O}(1)$: the relationship between function and weights became linear. C: interacting particles in a potential. When their number is large, describing their density instead of their individual motion becomes favorable (image graciously shared by Eric Vanden-Eijnden).

Implicit regularization and infinite width nets How can over-parametrized networks be predictive, if they can fit any data? It must imply that the GD or SGD dynamics lead to specific minima of the loss landscape where the function is more regular than for generic minima– a phenomenon coined *implicit regularization* [18, 19]. As sketched in Fig.2, performance is best as $N \to \infty$, triggering a huge interest in that limit. To which algorithms does GD correspond to in that case?

NTK: The propagation of the input signal through infinite-width FC nets at initialization is now well-understood. If the weights – defined in Eqs.3 – are initialized as i.i.d. random variables with zero mean and variance one (a set-up we consider throughout this work), the output function $\tilde{f}(x)$ is a Gaussian random process. Its covariance can be computed recursively [31–36].

Very recently it was realised that the *learning dynamics* also simplifies in this limit [10, 37–41]. The key insight of [10] is that the output becomes a linear function of the weights then, as sketched in Fig.3 A,B. Physically, very tiny changes of weights can interfere positively and change the output by $\mathcal{O}(1)$ which is sufficient to learn, but is not sufficient to change the gradient $\nabla_\theta \tilde{f}$.

More formally, the gradient flow dynamics at finite time can always be described by the *neural tangent kernel* (NTK) defined as:

$$\Theta(\theta, x, y) = \nabla_\theta \tilde{f}_\theta(x) \cdot \nabla_\theta \tilde{f}_\theta(y), \tag{5}$$

where x, y are two inputs and ∇_θ is the gradient with respect to the parameters θ. Specifically, $\partial \tilde{f}(x)/\partial t$ can be expressed as a linear combination of the $\Theta(\theta, x, x_i)$; i.e. $\tilde{f}$ evolves within a space of dimension P. In general, Θ depends on θ and thus on time and on the randomness of the initialization. Yet as $h \to \infty$, $\Theta(\theta, x, y)$ converges to a well-defined limit independent of initialization, and does not vary in time: deep learning in that limit is thus essentially a kernel method [10].

Feature learning (hydrodynamic) regime: The infinite width limit can be taken differently by adding a factor $1/\sqrt{h}$ in the definition of the output. Then the weights must change significantly for the output to become $\mathcal{O}(1)$ and fit the data. This recently discovered limit is called "mean field", "rich" but also "feature learning regime" in the literature, because the neurons learn how to respond to different aspects of the input data, as in Fig.1 – whereas, in the NTK limit, the neuron's response evolves infinitesimally. This regime has been studied in several works focusing mostly on one-hidden layer networks [42–48], with recent development for deeper nets, see e.g. [49]. In this setting the output function for a one hidden layer reads:

$$\tilde{f}_\theta(x) = \frac{1}{h} \sum_{i=1}^{h} W_i^{(2)} \sigma(\frac{1}{\sqrt{d}} W_i^{(1)} \cdot x + B_i), \tag{6}$$

The law of large number can then be invoked to replace Eq.6 by an integral:

$$\tilde{f}_\theta(x) \to \int \mathrm{d}W^{(2)} \mathrm{d}W^{(1)} \mathrm{d}B \, \rho(W^{(2)}, W^{(1)}, B) \; W^{(2)} \sigma(\frac{1}{\sqrt{d}} W^{(1)} \cdot x + B) \tag{7}$$

where ρ is the density of parameters. It is then straightforward to show that gradient flow leads to a dynamics on ρ of the usual type for conserved quantities: it is the divergence of a flux $\partial \rho / \partial t = -\nabla \cdot J$. Here the divergence is computed with respect to the set of weights associated with a single neuron, and the flux follows $J = \rho \Psi(W^{(2)}, W^{(1)}, B; \rho_t)$ where Ψ is some function that can be expressed in terms of the loss [42]. This formulation is equivalent to the hydrodynamics description of interacting particles in some external potential. Fig.3.C illustrates that when the number of particles (or neurons) is large, ρ is a more appropriate description than keeping track of all the particle positions (the weights in that case).

Lazy training: The fact that deep learning converges to well-defined algorithms as h diverges, explains why performance converges to a well-defined value in that limit (which will depend on the limit considered), as sketched in Fig.2. Yet, this distinction between a regime where the NTK does not change, and one where it evolves and features are learnt can be made at finite h. Chizat and Bach proposed a model of the form [50]:

$$f_\theta(x) \equiv \alpha \left[\tilde{f}_\theta(x) - \tilde{f}_{\theta(t=0)}(x) \right], \tag{8}$$

$\alpha = \mathcal{O}(1)$ corresponds to the NTK initialization and $\alpha = \mathcal{O}(h^{-1/2})$ to the mean-field limit. In the over-parametrised regime (our work below defines it sharply) where data are fitted, infinitesimal changes of weights are sufficient to learn when $\alpha \to \infty$ at finite h, and the NTK does not evolve in time (but has

fluctuations at initialization). This regime is sometimes referred to as lazy training, in our context we will interchangeably denote it the NTK regime.

Overall, these findings are recent breakthrough in our understanding of neural nets. Yet they ask many questions that are central to this review. *If two limits exist, which one best characterises neural networks that are used in practice? Which limit leads to a better performance? How does it depend on the architecture and data structure? Which effect causes the improvement of performance as h increases?*

Curse and invariance toward diffeomorphisms Mallat and Bruna [51, 52] proposed that invariance toward smooth deformations of the image, i.e. diffeomorphisms, may allow deep nets to beat the curse of dimensionality. Specifically, consider the case where the input vector x is an image. It can be thought as a function $x(s)$ describing intensity in position s, where $s \in [0,1]^2$ [4]. A diffeomorphism is a bijective deformation that changes the location s of the pixels to $s' = \tau(s)$. In our review below, it will be useful to introduce the pixel displacement field $\xi(s) = \tau(s) - s$, analogous to the displacement field central to elasticity. We denote by $T_\tau[x]$ the image deformed by τ, i.e. $T_\tau[x](s) = x(\tau^{-1}(s))$. One expects that smooth diffeomorphisms do not affect the label of an image: $Prob(y(x) \neq y(T_\tau[x])) \ll 1$ if $\|\nabla_s \xi\| \ll 1$.

Mallat and Bruna could handcraft CNNs, the "scattering transform", that are insensitive to smooth diffeomorphisms and perform well:

$$|f(x) - f(T_\tau[x])| \leq C_0 \|\nabla_s \xi\|. \tag{9}$$

They hypothesised that during training, CNNs learn to satisfy Eq.9. In this view, by becoming insensitive to many aspects of the data irrelevant for the task, CNNs effectively reduce the data dimension and make the problem tractable.

A limitation of this framework is that it says little about how invariants are learnt dynamically. Yet, it is the center of the problem. FC nets are more expressive than CNNs, and nothing a priori prevents them from learning these invariants – yet, they presumably don't, since their performance do not compare with CNNs. Along this route, an interesting question is how to develop observables to characterise empirically the dynamical emergence of invariants. Attempts have been made using mutual information approaches [53] which display problems in such a deterministic setting [54]; or measures based on the effective dimension of the neural representation of the data [55,56] which are informative but can sometimes lead to counter-intuitive results [5]. In Section 5 we discuss other observables based on kernel PCA of the NTK.

1.4 Organization of the manuscript

To think about the results above and the questions they raise in a unified manner, it is useful to consider Fig.4. It shows novel empirical data for the performance of neural nets in the plane $(h, \tilde{\alpha})$ where h is the width and $\tilde{\alpha}$ the rescaled initialization amplitude $\tilde{\alpha} = \sqrt{h}\alpha$. Specifically, the ensemble test error of fully-connected nets learning MNIST (a data set of images of digits [58]) and CIFAR10 (images of planes, dogs, etc...) respectively with gradient descent is shown. This ensemble test error is computed by preparing, for each point in the phase diagram, $M = 15$ nets with different initialization. The test error of the mean

[4] This space is in fact discrete due to the finite number of pixels, but this does not alter the discussion.

[5] The effective dimension is defined from how the distance δ among neighboring points varies with their number P, i.e. $\delta \sim P^{-1/d_{\text{eff}}}$. Its interpretation can be delicate, as it is observed to sometimes increase as the information propagates in the network, which cannot be the case for the true dimension of the manifold representing the data.

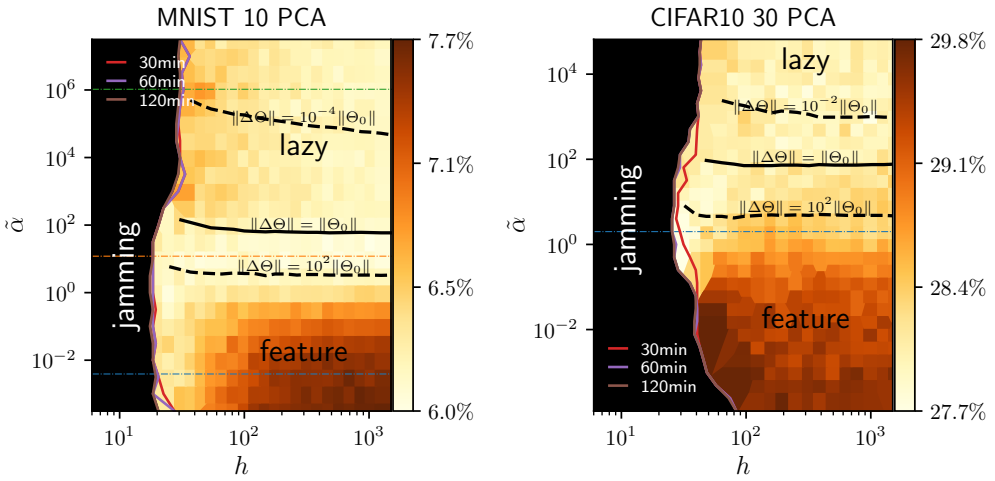


Figure 4: Phase diagram for the performance of deep learning in the width h *v.s.* rescaled initialization magnitude $\tilde{\alpha} = \sqrt{h}\alpha$ plane, where α is defined in Eq.8. The architecture corresponds to a fully-connected neural network with two hidden layers trained with gradient descent [57]. Binary classification was performed for Left MNIST binary (odd/even) and Right CIFAR10 binary (classes have been regrouped in two sets). The training sets have been reduced to 5000 samples. These data sets were first projected on 10 and 30 PCAs, respectively. This dimensionality reduction of the problem is useful to study the jamming transition[7]but not necessary to investigate the over-parametrized regime [57]. This choice does not affect qualitatively the geometry of the diagram however. The colormap shows the generalization of the ensemble average over 15 samples. The full black line indicates the cross-over from feature to lazy training. It is defined as the location for which the change of the NTK after training $\Delta\Theta = \Theta(t = \infty) - \Theta(0)$ is equal in norm to its magnitude at initialization, i.e. $\|\Delta\Theta\|/\|\Theta(0)\| = 1$. Choosing other values for this ratio (see dashed lines) does not affect qualitatively the geometry of the cross-over line. The vertical color lines indicate the jamming transition where the training loss hits zero (in the black region the training loss does not reach zero). Its precise position depends on the time the simulation is allowed to run [8], as indicated in legend (using a GPU V100 and the gradient flow algorithm of [57] (gradient descent with adaptive learning rate)). It is expected and corresponds to the usual critical slowing down occurring near phase transitions. Horizontal dashed lines correspond to the measures done in Fig.5. The code used to generate these data is available at `https://doi.org/10.5281/zenodo.4322828`.

predictor $\langle f \rangle$ over the 15 trained nets is then shown. The black region corresponds to the under-parametrized regime where the loss of individual nets does not converge to zero after learning. As discussed below, its boundary corresponds to a line of critical points corresponding to a "jamming transition" where the system stops displaying a rough landscape. In the colored over-parametrized regime, the black line indicates a cross-over separating the lazy training regime (where the total change of the NTK of individual nets during learning is small in relative terms) from a feature learning regime (where the NTK evolves significantly).

Fig.5 shows the curves for the test error and the ensemble test error for three values of $\tilde{\alpha}$, as h varies. At

the jamming transition, the test error displays a peak as observed in [59]. A similar peak occurs in regression problems [21], where it has a simpler origin and occurs when the number of parameters matches the number of data. This shape for the test error has since then been coined double-descent [60]. Interestingly, as shown in Fig.5 this phenomenon occurs independently of the value of $\tilde{\alpha}$ (and thus of the over-parametrized regime one enters into passed jamming) and vanishes when the ensemble average is taken, as observed in [61] and recently confirmed in [62].

Our goal is to review, in non-technical terms, concepts and arguments justifying such a phase diagram and discuss the learning of invariants in this context. In section 2, we will argue that the boundary of the black region in Fig.4 is a line of critical point, analogous to the jamming transition occurring in repulsive particles with finite-range interactions. For wider nets, the landscape is not glassy and displays many flat directions. We will provide a simple geometric argument justifying why this transition in deep nets fall into the universality classes of ellipsoid (rather than spherical) particles, which fixes the properties of the landscape (such as the spectrum of the Hessian) near the transition. We will mention an argument à la Landau justifying the cusp in the test error at that point. In Section 3, we will provide a quantitative explanation as to why the test error keeps improving as the width increases passed this transition. Increasing width turns out to eliminate the noise due to the random initialization of the weights, eventually leading to the well-defined algorithms introduced above. Ensemble averaging at finite width efficiently eliminates this source of noise as well as the double descent, as shown in Fig.5. In Section 4, we will explain why the cross-over between the lazy and feature learning regimes corresponds asymptotically to a flat line in Fig.4. We will review observations that lazy training outperforms feature learning for standard data sets of images for fully connected architectures, but not for CNNs architectures, corresponding to a larger learning curve exponent β. In Section 5, we review arguably the simplest model in which a neural net learns invariants in the data structure, by considering that labels do not depend on some directions in input space. As observed in real data, two distinct training exponents β in the lazy and feature learning can then be computed. We conclude by discussing open questions.

2 Loss landscape and Jamming transition

Minimizing a loss is very similar to minimizing an energy. Describing the energy landscape of physical systems is a much studied problem, especially in the context of glasses. Progress was made on this topic in the last fifteen years by considering finite range interactions, for which bringing the energy to zero is equivalent to satisfying a set of constraints. In that case, the landscape is controlled by a critical point [63–65], the so-called jamming transition. It occurs as the particle density ϕ increases. For $\phi > \phi_c$, a gradient descent from a random initial positions of the particles gets stuck in one – out of many – meta-stable states, corresponding to a glassy solid as depicted in Fig.6B,D. For lower densities, the gradient descent reaches zero energy, as sketched in Fig.6.E : particles can freely move without restoring forces, and the landscape has many flat directions. It corresponds to the situations depicted in Fig.6A,C.

There are two universality classes for jamming, leading to distinct properties for the curvature of the landscape (i.e. the spectrum of the Hessian) [66–71], for the structure of the packing obtained [72–75] and for the dynamical response to a perturbation [76, 77]. Spheres and ellipses fall in distinct classes as

[7] Otherwise, for FC architectures the value of h at which the jamming transition occurs is a few neurons, leading to new effects.

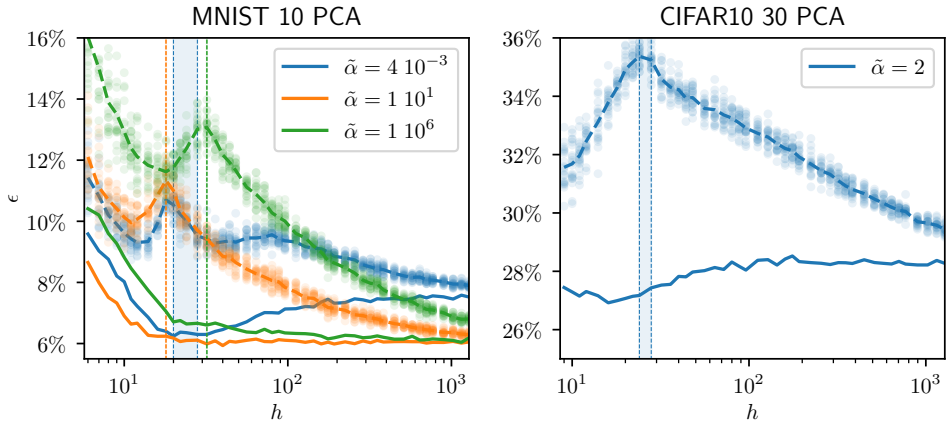


Figure 5: Comparison of the generalization error and the ensemble generalization error for the same data, architecture and learning dynamics used in Fig.4. Values of $\tilde{\alpha}$ used are indicated in legends, and corresponds to the colored horizontal dashed lines in Fig.4. The ensemble average is done over 20 samples. The vertical lines indicate the location, or interval of locations, where the jamming transition occurs. Note that ensemble averaging improves a lot performance at intermediate width, and eliminates the double descent both in the lazy or feature learning regime.

illustrated in Fig.6. More generally, the jamming transition occurs generically in satisfiability problems with continuous degrees of freedom (it can be defined with discrete degrees of freedom [78], but then differs qualitatively; in particular, the present discussion does not apply to the discrete case). It occurs in the perceptron [79–81] but also for deep nets [8,59].

We will recall below a geometric argument introduced in [8] determining the universality class of the jamming transition. For deep nets, jamming belongs to the universality class of ellipses [8,59]. The spectrum of the loss near the jamming transition displays zero modes, a gap and a continuous part, as measured in Fig7.C. Another implication of this analogy is the number N_Δ of data whose margin is smaller than unity after learning, which contribute to the loss. These data are conceptually similar to the support vectors central to SVM algorithms. For particles, N_Δ corresponds the number of pairs of particles still overlapping after energy minimization. As illustrated in Fig7.B, N_Δ/N jumps from zero to a value strictly smaller than one at jamming where the loss becomes positive, precisely as ellipses do.

Geometric argument fixing the universality class: Here we seek to give a simple intuition of the result of [8] (a more rigorous argument can be found there). We consider continuous satisfiability problems where one seeks to minimize an energy or loss function of the type:

$$\mathcal{U} = \sum_{\mu \in m} \frac{1}{2} \Delta_\mu^2. \tag{10}$$

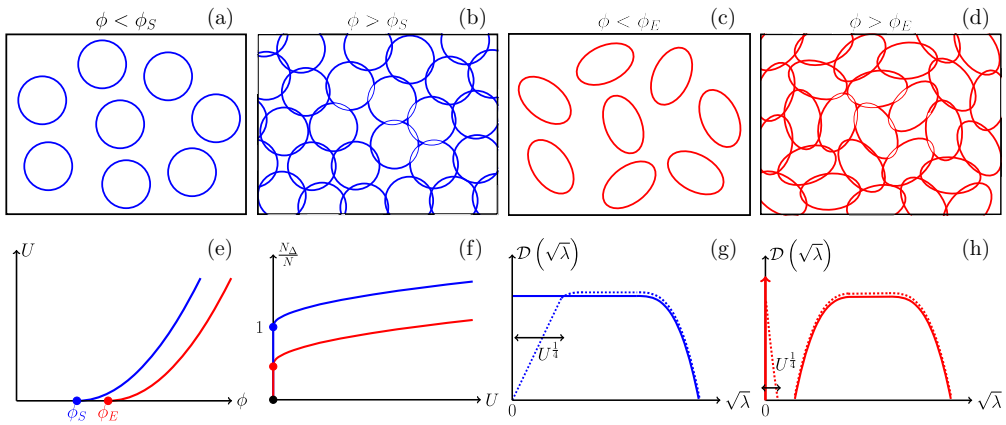


Figure 6: Sketch of the jamming transition for repulsive spheres and ellipses. (a,b,c,d) Both systems transition from a fluid to a solid as the density passes some threshold, noted ϕ_S for spheres and ϕ_E for ellipses. (e) For denser packings, the potential energy $\mathcal{U}$ becomes finite. (f) The ratio N_Δ/N between the number of particles in contact N_Δ (corresponding to unsatisfied constraints) and the number of degrees of freedom N jumps discontinuously to a finite value, which is unity for spheres but smaller for ellipses. (g,h) This difference has dramatic consequence on the energy landscape, in particular on the spectrum of the Hessian. In both cases, the spectrum becomes non-zero at jamming, but it displays a delta function with finite weight for ellipses (indicating strictly flat directions), followed by a gap with no eigenvalues, followed by a continuous spectrum (h, full line). For spheres, there is no delta function nor gap (g, full line). As one enters the jammed phase, in both cases a characteristic scale $\lambda \sim \sqrt{\mathcal{U}}$ appears in the spectrum (g and h, dotted lines). From [8].

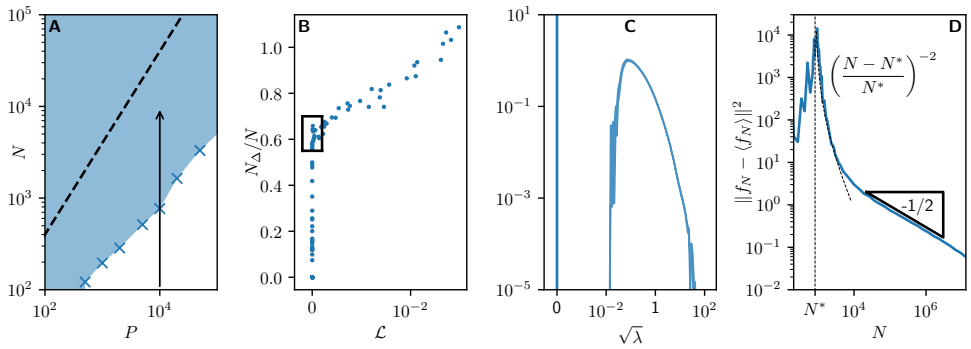


Figure 7: **A** Phase diagram indicating where the hinge loss hits zero (in blue) or not (in white) at the end of training, as the number of parameters N and training points P vary, for the MNIST data set discussed above (binary classification, 10 PCAs) trained with a FC nets with 3 hidden layers. **B** Number of unsatisfied patterns N_Δ/N as a function of the loss at the end of training, following the arrow indicated in panel A. Here N is fixed, and $P = 1, \dots 70k$. **C** The spectrum of the dominant term in the Hessian near jamming H_0 (averaged over the run at jamming, in the rectangle of **B**) presents a delta function in zero and a gaped continuous spectrum at high frequencies, a generic feature of hypostatic jamming. **D** Fluctuations of the output function along the arrow shown in **A**, averaged over 20 initial conditions. It decays asymptotically as a power-law $N^{-1/4}$ for the MNIST data set. At jamming, these fluctuations explode, consistent with a quadratic singularity. From [8,61].

where the sum is made on all constraints that are not satisfied, corresponding to $\Delta_\mu > 0$. The quantities Δ_μ can depend in general on the N degrees of freedom of the system. For systems of particles, $\mathcal{U}$ is an energy and Δ_μ is the overlap between a pair $\mu = (i, j)$ of particles i and j. For spheres it reads $\Delta_{ij} = 2R - r_{ij}$ where R is the particle radius and $r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$ the distance between particles i and j. In that case, N is the number of particles times the spatial dimension. For the perceptron or deep nets, $\mathcal{U}$ corresponds to a quadratic hinge loss generally denoted $\mathcal{L}$ defined in Eq.4 (it is straightforward to extend these arguments to other hinge losses with $\gamma > 1$ [8,63], the case $\gamma = 1$ of the linear hinge shares many similarities with those but also display interesting differences [82]). As defined in the introduction, in that case $\Delta_\mu = 1 - f(\mathbf{x}_\mu)y_\mu$.

As the jamming transition is a singular point, it is useful – but not strictly necessary [8] – to think of approaching from the glassy (large density ϕ or under-parametrized) phase. It corresponds to $\mathcal{U} \to 0$ as sketched in Fig. 6 E,F, implying that $\Delta_\mu \to 0$ $\forall \mu \in m$. As argued in [83], for each $\mu \in m$ the constraint $\Delta_\mu = 0$ defines a manifold of dimension $N - 1$. Satisfying N_Δ such equations thus generically leads to a manifold of solutions of dimension $N - N_\Delta$. Imposing that solutions exist thus implies that, at jamming,

[8] Particles can always sit right at jamming if an infinitesimal pressure is applied to the system. In the context of neural nets, it can also be achieved by introducing an additional term in the loss of the type $\lambda||\theta(t) - \theta(t = 0)||^2$ penalizing the square evolution of the weights (called "weight decay") of vanishing magnitude $\lambda \to 0^+$.

one has:

$$N_\Delta \leq N\,. \tag{11}$$

Note that this argument implicitly assumes that the N_Δ constraints are independent, see [8] for more discussions.

An opposite bound can be obtained by considerations of stability, by imposing that in a stable minimum the Hessian must be positive definite [66]. The Hessian is an $N \times N$ matrix which can be written as:

$$\mathcal{H}_U = \sum_{\mu \in m} \nabla \Delta_\mu \otimes \nabla \Delta_\mu + \sum_{\mu \in m} \Delta_\mu \mathcal{H}_{\Delta_\mu} \equiv \mathcal{H}_0 + \mathcal{H}_p\,, \tag{12}$$

where $\mathcal{H}_{\Delta_\mu}$ is the Hessian of Δ_μ, and $\mathcal{H}_0$ and $\mathcal{H}_p$ correspond to the first and second sum, respectively. $\mathcal{H}_0$ is positive semi-definite, since it is the sum of N_Δ positive semi-definite matrices of rank unity; thus $\mathrm{rk}(\mathcal{H}_0) \leq N_\Delta$, implying that the null-space of $\mathcal{H}_0$ is at least of dimension $N - N_\Delta$. $\mathcal{H}_p$ becomes very small approaching jamming, since the Δ_μ's vanish. Let us denote by N^- the number of negative eigenvalues of $\mathcal{H}_p$. Requiring that $\mathcal{H}_U$ has no negative eigenvalues thus implies:

$$N_\Delta \geq N^-\,. \tag{13}$$

For *spheres* [64] (as well as for the perceptron if a negative margin is used while constraining the norm of the weight vector), $\mathcal{H}_p$ is negative definite and $N^- = N$. This results stems from the fact that $\mathcal{H}_{\Delta_\mu}$ is negative semi-definite. Indeed $\mathcal{H}_{\Delta_\mu}$ characterizes the second order change of overlap between particles, and is negative because the distance between spheres moving transversely to their relative direction always increase following Pythagoras theorem. In that case we thus have $N_\Delta \geq N$. Together with Eq. (11) that leads to $N_\Delta = N$: as spheres jam the number of degrees of freedom and the number of constraints (stemming from contacts) are equal, as empirically observed [63]. This property is often called *isostaticity*.

However, in other problems such as ellipses and deep nets, $\mathcal{H}_{\Delta_\mu}$ and thus $\mathcal{H}_p$ have positive eigenvalues. Indeed the overlap between two ellipses can *increase* if one of them rotates. Likewise, for a fully-connected Relu net and random data at initialization, $\mathcal{H}_p$ has a symmetric spectrum and $N^- \approx N/2$. Generically one expects then jamming to occur with $N_\Delta < N$ as sketched for ellipses in Fig.6 and shown empirically for neural nets learning MNIST in Fig.7. Jamming is then referred to as "hypostatic". The associated consequences on the spectrum of the hessian shown on the same figures are derived in specific cases in [68, 69, 84]: it always presents a delta function in zero and a gap at jamming.

Effect of the number of training data P The location of the jamming transition $N^*(P)$ depends on P. This dependence is linear for random data but sub-linear for structured data [8], as exemplified in Fig.7.A. Denoting $N^- = C_0 N$, and using Eq.13 together with $N_\Delta \leq P$, we obtain $N^*(P) \leq P/C_0$, guaranteeing convergence to a zero loss for a number of parameters linear in P if $C_0 > 0$. We conjecture that C_0 to remain bounded by a strictly positive value for large N for generic data and architectures used in practice [9]. C_0 can be measured a posteriori, leading to the bound corresponds to the dotted line in Fig.7.A. C_0 a priori depends on the choice of architecture, dynamics and data set. Controlling its value a priori is yet out of reach [10], and would lead to a rather tight guarantee of convergence toward a global minimum of the loss.

[9] This fact does not hold for linear or polynomial activation function, see discussion in [8].

[10] Controlling H_0 and H_p during learning was done only in the limit $N \to \infty$ which does not apply near jamming [85].

Effect of the scale of initialization $\tilde{\alpha}$ It is apparent in Fig.4 that the location of the jamming transition depends on the scale of initialization $\tilde{\alpha}$. From this figure, we observe the following general trend: the jamming transition occurs with less parameters when the test error of the ensemble-averaged predictor is small. It follows the intuition that an easier rule to learn should correspond to less parameters to fit the data.

Effect of the Jamming transition on performance As the jamming transition is approached, the norm $\|f_N\|$ of the predictor diverges [8], which is responsible in the cusp in the double descent displayed by the test error in Fig.5. This divergence was first pointed out for regression [21]. Yet for classification, the divergence defers quantitatively and is compatible with an inverse power-law, as illustrated in Fig.7.D. It can be understood using an argument à la Landau, inspired by results on the perceptron [79, 81]. The intuitive idea is that, by increasing the norm of the predictor, one reduces effectively the unit margin required by the hinge loss to fit the data. For $N < N^*$, data cannot be fitted even with a vanishing margin.

Specifically, consider that the norm of the predictor $\|f_N\|$ is fixed during training and denote by $N^*(\epsilon)$ the jamming transition as a function of the margin ϵ. Assume (as occurs for the perceptron) that $N^*(\epsilon)$ is a smooth function of ϵ, such that $N^*(\epsilon) \approx N^*(0) + N'^*(0)\epsilon$. For N just above $N^*(0)$, margins of magnitude unity cannot be fitted for a fixed norm of the predictor, but they can if that norm is allowed to increase by a factor $N'^*(0)/(N - N^*(0))$. Indeed it effectively reduces the margin by that amount to some effective value $\tilde{\epsilon} \equiv \epsilon(N - N^*(0))/N'^*(0)$. By construction, $\tilde{\epsilon}$ satisfies $N^*(\tilde{\epsilon}) = N$. This argument justifies the observed inverse power-law.

Note that any perturbation to gradient flow (such as early stopping, using stochastic gradient descent or weight decay) will destroy this divergence [11]. Such regularizations are thus most efficient near jamming [59].

3 Double descent and the benefits of overparametrization

To avoid bad minima in the loss landscape, it is thus sufficient to crank up N passed N^*. But then, one would naively expect to lower performance and overfit, as illustrated in the right panel of Fig.2. It is not the case for classification and deep nets, as illustrated in Fig.5: performance is very poor and displays a cusp right at the jamming transition point N^*, and then continuously improves as $N \to \infty$!

A scaling theory can be built to explain quantitatively why performance keeps improving with N passed that point, and how fast the test error reaches its asymptotic behavior [61]. Essentially, at infinite width learning corresponds to well-defined algorithms as discussed in introduction, but these algorithms become noisier when the width is finite. Indeed as $N \sim h^2$ increases, the fluctuations on the learnt output function induced by the random initialization of the weights decrease [86]. It follows $\|f_N - \langle f_N \rangle\| \sim N^{-1/4}$ as shown in Fig.7.D, where $\langle f_N \rangle$ is obtained by ensemble-averaging outputs trained with different random initializations. This result is true both for the lazy training [61] and the feature learning [57] regime, as justified below. The associated increase in test error is quadratic in the fluctuations (it is obvious for a mean square error loss, but is also true for classification under reasonable assumptions [61]), leading to $\langle \epsilon(f_N) \rangle - \epsilon(\langle f_N \rangle) \sim N^{-1/2}$ as observed [61]. This effect is responsible for the double descent, as can be directly checked by considering the training curve $\epsilon(\langle f_N \rangle)$ of the ensemble average function $\langle f_N \rangle$ in Fig.5

[11] They are relevant perturbations in the sense of critical phenomena.

which does not display a second descent. More generally as apparent in Fig.4, at fixed $\tilde{\alpha} = \sqrt{h}\alpha$, the ensemble average test error weakly varies with h once in the over-parametrized regime.

In the lazy training regime, the fluctuations $\langle \|f_N - \langle f_N \rangle\| \rangle \sim N^{-1/4}$ simply mirror the fluctuations of the NTK induced by initialization, argued to follow $\|\Theta_N - \langle \Theta_N \rangle\| \sim N^{-1/4} \sim h^{-1/2}$ in [61], as analytically confirmed since then [87, 88]. For simple losses and gradient descent, the predictor f_N must acquire the same fluctuations [61], as observed.

Similar fluctuations induced by initial conditions $\langle \|f_N - \langle f_N \rangle\| \rangle \sim N^{-1/4}$ with identical consequences occur in the feature learning regimes in neural nets [57]. Such fluctuations are certainly expected at initialization for a one-hidden layer, since the density of neurons parameters ρ introduced in Eq.7 must have finite sampling fluctuations of order $\delta\rho \sim 1/\sqrt{h} \sim N^{-1/4}$ as expected from the central limit theorem. Because in the asymptotic regime $N \to \infty$ the value of ρ at initialization affects the learnt function, the magnitude of fluctuations $\delta\rho$ (and thus of the learnt function f_N) induced by the random initialization will still scale as $N^{-1/4}$ after learning, as rigorously proven for a one-hidden layer in [89].

Overall, this scaling theory supports that for generic data sets and deep architectures, the second descent results from the noise induced by finite N effects and initialization on the limiting algorithms reached as $N \to \infty$. One interesting practical implication, apparent in Fig.4 and Fig.5, is that optimal performance is found by ensemble averaging nets of limited width passed their jamming transition.

Since these arguments were proposed, the double descent curve has been analytically computed in the simple case of linear data in the limit of infinite dimension and random features machines [46, 90], where it was also shown [91, 92] to result from the fluctuations of the kernel that vanish with increasing number of neurons, confirming the present explanation.

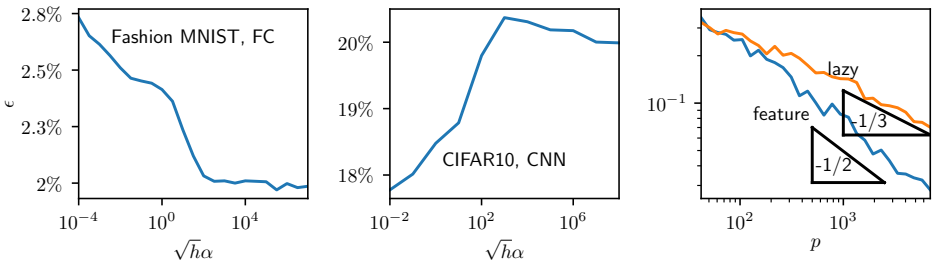


Figure 8: Left, Center: Ensemble-average test error *vs.* the scaling parameter $\tilde{\alpha} = \sqrt{h}\alpha$. It illustrates that the NTK regime (large $\sqrt{h}\alpha$) tends to outperform the feature learning regime (small $\sqrt{h}\alpha$) for FC architecture (Left) but not in CNNs (Center). In the latter case, the training curve exponent β is larger in the feature learning (blue curve, $\beta \approx 1/2$) regime than in the NTK one (orange curve, $\beta \approx 1/3$). From [57].

4 Disentangling the NTK and feature learning regimes

As discussed in the previous section, wider nets are more predictive. A comparable gain of performance can however be obtained already at intermediate width, by ensemble averaging outputs over multiple

initialization of the weights. As apparent in Fig.4, the rescaled magnitude of initialization $\tilde{\alpha} = \alpha\sqrt{h}$ which allows to cross-over between the lazy training and feature learning regimes does affect performance. Which regime best characterises deep nets used in practice? Which one performs better? Some predictions of the NTK regime appear to hold in realistic architectures [39], and training nets in NTK limit can achieve good performance on real datasets [40, 93, 94]. Yet, in several cases the feature learning regime beats the NTK [50, 95], in line with the common idea that building an abstract representation of the data such as sketched in Fig.1 is useful. Several theoretical works show that NTK under-performs for specific, simple models of data [14, 96–99].

In [57], this question was investigated systematically for deep nets in the (α, h) plane, where α is the scale of initialization introduced in [50] as defined in Eq.8. This study developed numerical methods of adaptive learning rates to follow gradient flow while changing α by ten decades. The main results are as follows:

(i) The cross-over between the two regimes occur when $\alpha\sqrt{h} = O(1)$ as apparent in Fig.4, extending the result of [50] limited to one hidden layer nets. For $\alpha\sqrt{h} \ll 1$, $\|\Delta\Theta\|/\|\Theta\| \gg 1$ while the opposite holds true when $\alpha\sqrt{h} \gg 1$. Here $\Delta\Theta = \Theta(t) - \Theta(t=0)$ characterises the evolution of the tangent kernel and t is the learning time. It is convenient to divide the loss by α^2 and consider $\tilde{\mathcal{L}} \equiv \mathcal{L}/\alpha^2$, which can be shown to ensure that the dynamics occurs on a time scale independent of α in the large α limit [50, 57]. This result holds true for the usual choices of loss (cross-entropy, hinge, etc...) at any finite time. It also holds true for $t \to \infty$ if the hinge loss is used, since in that case convergence to a zero loss occurs in finite time independently of h and α.

Here we provide a schematic argument justifying this result, see [57] for a more detailed analysis. The variation of the output f with respect to the pre-activation a (which are of order one at initialization) of a given neuron is of order $\partial f/\partial a \sim \alpha/\sqrt{h}$. This result is obvious for the last hidden layer (using that the last weights are of order $1/\sqrt{h}$), but can be justified recursively at all layers, as discussed in [57] and derived implicitly in the NTK study [10]. For gradient flow, the variation Δa of pre-activation due to the evolution of the bias (considering the previous weights leads to a similar scaling) must be of order $\Delta a \sim t(\partial\tilde{\mathcal{L}}/\partial f)(\partial f/\partial a) \sim t/\alpha\sqrt{h}$ using that $\partial\tilde{\mathcal{L}}/\partial f \sim 1/\alpha^2$. Thus Δa is of order $1/\alpha\sqrt{h}$ at the end of training (since a zero hinge loss is reached on a time scale $t = O(1)$). The NTK regime must thus break down when $\Delta a \sim 1$ when the relation between weights and output must become non-linear, corresponding to a cross-over for $\alpha^* \sim 1/\sqrt{h}$ for large h.

A similar line of thought can be used at intermediate width. When reducing h so as to approach the jamming transition from the over-parametrized phase, the norm of the output – and therefore weights and pre-activations – explode as reviewed in Section 2. One is never then in the lazy training regime, and the relationship between the variation of weights and the output must become non-linear. Thus, in the vicinity of the jamming transition, the networks always lie in the feature learning regime. Consequently, the cross-over lines separating lazy and feature learning must bend up and never cross the jamming line in Fig.4. We do observe curves qualitatively bending up as expected (yet it is hard to make precise measurements very close to jamming).

(ii) *For fully-connected nets and gradient descent, the NTK tends to outperform the feature learning regime*, as exemplified in the Left panel of Fig.8. This result was found for a variety of data sets (MNIST, Fashion-MNIST, CIFAR10, etc...), except for MNIST 10 PCA. It is apparent in Fig.4, where the best performance for MNIST 10 PCA appears to occur the cross-over region $\tilde{\alpha} \sim 1$.

(iii) For CNN architectures, feature learning outperforms the NTK regime as shown in the central panel of Fig.8. It corresponds to a larger training curve exponent β, as appears in the right panel of Fig.8. (ii,iii) were recently confirmed by the Google team [62].

These observations raise various questions. What are the advantages and drawbacks of feature learning, and why are the latter more apparent in FC rather than CNN architectures? For modern CNN architectures, is the improvement of the learning curve exponent β in the feature learning regime key to understand how the curse of dimensionality is beaten? Is it associated with learning invariants in the data? In our opinion, the answers to these questions are yet unknown. To start tackling these questions, in the next section we study a simple model of invariant data for which the improvement of the learning curve exponent β in the feature learning regime can be computed.

5 Learning simple invariants by compressing irrelevant input dimensions

How can the curse of dimensionality be beaten? A favorable aspect of various data sets such as images is that the data distribution $P(\mathbf{x})$ is very anisotropic, consistent with the notion that the data lie in a manifold of lower dimension. In that case, the distance between neighboring data reduces faster with growing P, improving kernel methods [100]. The positive effect of a moderate anisotropy can also be shown for kernel classification [101] and regression [99, 102]. Yet, even in simple data sets like MNIST of CIFAR10, the intrinsic dimensions remain significant ($d_{int} \approx 14$ and $d_{int} \approx 30$ respectively [100]). This effect helps but cannot resolve the curse of dimensionality for complex data set: indeed using Laplace or Gaussian kernels (which are very similar to the NTK of fully connected nets) lead to $\beta \approx 0.4$ for MNIST (which is decent) but $\beta \approx 0.1$ for CIFAR10, which implies very slow learning as the number of data increases.

As discussed in the introduction, another popular idea is that data sets as images are learnable because they display many invariant transformations that leave the labels unchanged. By becoming insensitive to those, the network essentially reduces the dimension of the data. This view is consistent with the notion that deep learning leads to an abstract neural representation of the data sketched in Fig.1. This effect may be responsible for our observation in the right panel of Fig.8 that the training curve exponent β is more favorable in the feature learning regime. However, understanding how invariants are built dynamically and how it affects performance and β remains a challenge.

Specific models of data can be built [43, 90, 99, 104] in which lazy training does not learn at all, whereas neural nets trained in the feature learning regime succeed. These models however do not capture the two finite and distinct learning curve exponents β observed in the two regimes.

The stripe model In [96, 101, 103], a simple model of invariant is considered. The labels do not depend on $d_\perp = d - d_\parallel$ directions of input space, so that $y(x) = y(x_\parallel)$ where $x = (x_\perp, x_\parallel)$. $x_\perp$ could correspond for example to uninformative pixels near the boundaries of pictures. For gradient descent, with *the logistic loss*, a one-hidden layer can be shown to correspond to a max-margin classifier in a certain non-Hilbertian space of functions [96]. Dimension-independent guarantees on performance can then be obtained if the data can be separated after projection in a low dimensional space. The analysis is rigorous and general, but requires to go to extremely long times not used in practice and does not predict values for β.

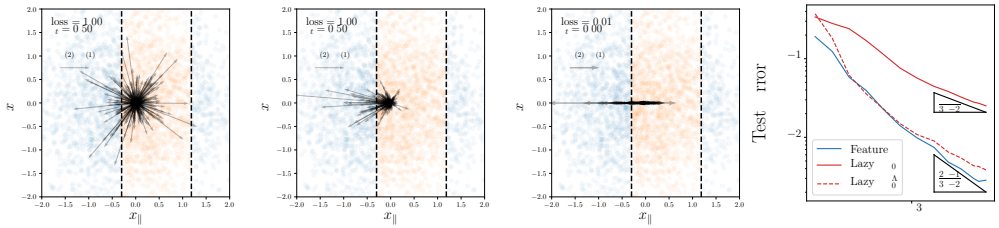


Figure 9: Left: Distribution of the weights vectors $\vec{W}^{(1)}$ pondered by the neuron second weight $W^{(2)}$ shown at different time points during the learning process (as characterised by the value of the loss in legend) for the stripe model. It reveals an alignment of the first layer of weight vectors in the informative direction. Training data points are also shown, and coloured depending on their labels. Right: Training curve $\epsilon(P)$ of a one hidden layer in the feature learning regime (blue), the NTK regime (red) and the NTK regime after compressing the data in the perpendicular direction by the factor Λ (dashed red). Note the similarity between this curve and the blue one, supporting that the main effect of learning feature for these data is the geometric compression of uninformative directions in input space. From [103].

In [101, 103], the hinge loss is considered. In that case, the dynamics stops after a reasonable time. If the density of data points does not vanish at the interface between labels, and if the latter is sufficiently smooth (e.g. planar or cylindrical), it is found that the test error decays as a power law in both regimes. For the lazy training regime, scaling arguments inspired by electrostatics lead to $\beta_{Lazy} = d/(3d-2)$. Feature learning performs better, as one find $\beta_{Feature} = (d + {}^{d_\perp}/_2)/(3d-2)$. The key effect leading to an improvement of the feature learning regime is illustrated in Fig.9 for $d_\parallel = 1$, called the stripe model: due to the absence of gradient in the orthogonal direction, the weights only grow along the $d_\parallel$ dimensions. Thus, for a infinitesimal initialization of the weights ($\alpha \to 0$), the neurons align along the informative coordinate in the data. Accordingly, in relative terms, they become less sensitive to $x_\perp$, which merely acts as a source of noise. This denoising is limited by the finite size of the training set. Specifically, it is found that $\Lambda \equiv W_\parallel/W_\perp \sim \sqrt{P}$ where $W_\parallel$ and $W_\perp$ are the characteristic scales of the first layer of weights in the informative and uninformative directions respectively. This effect is equivalent to a geometrical compression of magnitude Λ of the input in the uninformative direction. Indeed performing this compression by considering the transformed data $(x_\parallel, x_\perp/\Lambda)$, and learning these in the NTK regime gives very similar performance as learning the original data in the feature regime, see the right panel of Fig.9.

Kernel PCA of the NTK reveals a geometric compression of invariants The stripe models illustrates that neural nets in the feature learning regime can learn to become insensitive to transformations in the data that do not affect the labels. This insensitivity is limited by the sampling noise associated with a finite data set. Is this phenomenon also occurring for more modern architectures and more subtle invariants characterizing images? Measurements of the intrinsic dimension of the neural representation of data [55, 56] or mutual information estimates [53] suggest that it may be so, but as mentioned in introduction these observables have some limitations.

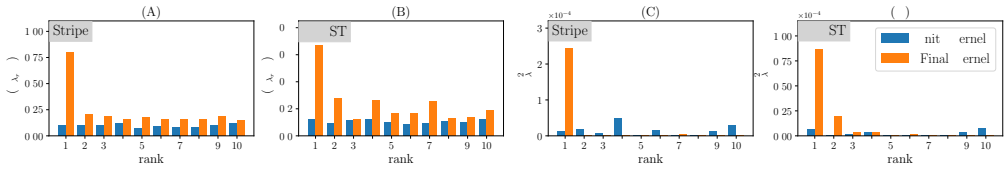


Figure 10: Relationship between the kernel PCA of the NTK (obtained both at initialization and after learning) and the labels. The mutual information (characterizing the amount of correlation) between the top eigenvectors of the Gram matrix and the labels (A, B) and their projection ω_λ (C, D) get large after learning for both the stripe model (learnt with a FC with one hidden-layer) and MNIST (learnt with a CNN). From [103].

In the feature learning regime, the NTK evolves in time. This evolution leads to a better kernel for the task considered. Indeed, using a kernel method based on the NTK obtained at the end of training leads to essentially identical performance than the neural net itself [103]. This observation is consistent with previous ones showing that the NTK "improves" in time [105, 106], in the following sense. For kernels we can always write $\Theta(x, y) = \psi(x) \cdot \psi(y)$, where $\psi(x)$ is a vector of features. In the case of the NTK, $\psi(x)$ can be chosen as the gradient of the output f with respect to the weights – see Eq.5. Kernels tend to perform better [107] if the vector of labels $\{y(x_i)\}_{i=1...P}$ has large coefficients along the first PCAs of the features vectors $\{\psi(x_i)\}_{i=1...P}$ (an operation called kernel PCA [108]). Such an alignment between the first PCAs of the features vectors of the NTK and the vector of labels is observed during learning [105, 106].

Although there is no general theory as to why such an alignment occurs, the stripe model provides a plausible explanation for this effect. In that case, this improvement must occur because one evolves from an isotropic kernel to an anisotropic one with diminished sensitivity to uninformative directions $x_\perp$. As a result, the top kernel PCA becomes more informative on the label (Fig.10.A) on which they project more (Fig.10.C). The same result is observed for a CNN trained on MNIST as shown in Fig.10.B,D. Overall, this view support that the improvement of the NTK reveals the geometric compression of uninformative directions in the data.

6 Conclusion

Don't be afraid of bad minima! Just crank-up the number of parameters until you pass a jamming transition. Beyond that point, bad minima are not encountered, and you can bring the loss to zero. Depending on the network width and on how you scale the value of your function at initialization, deep learning can then behave as a kernel method, or can alternatively learn features. Simple scaling arguments delimit the corresponding phase diagram, and appear to hold for benchmark data sets. Remarkably, these arguments depend very little on the specific data considered. Their practical implication is that ensemble averaging nets with different initialization past the jamming transition can be an effective procedure, as found by different groups [57, 61, 62].

Taking into account where you are in this phase diagram is arguably key to understand outstanding questions on deep learning. It is true for example for the role of stochasticity in the dynamics, which appears

to improve performance. It is natural that using stochastic gradient descent instead of gradient flow will help near jamming, because the noise will regularize the divergence of the predictor norm – an effect that can already be obtained with early stopping [59]. Likewise, for repulsive particles temperature regularizes singularity near the jamming transition [109]. Yet, it would be useful to study its effect on performance for the distinct regions of the phase diagram. It would be particularly interesting to know if one of the main effects of stochasticity in the over-parametrized limit is to push upward the cross-over between the lazy and feature learning regime in Fig.4, thus improving performance in CNNs where feature learning outperforms lazy training.

Ultimately, the central question left to understand is performance, and how the curse of dimensionality is beaten in deep nets. Even at an empirical level, the idea that invariance toward diffeomorphisms is the central phenomenon behind this success is not established. At the theoretical level, this view as not been combined with the recent improvement of our understanding of learning, which as mostly focused on fully connected nets. In comparison, the effort to model how CNNs learn features is more modest, despite that only those architectures tend to work well in practice. Such studies would arguably require to design simple canonical models of data presenting complex invariants, which are currently scarce.

Acknowledgments

We acknowledge G. Biroli, S. d'Ascoli, F. Cagnetta, A. Favero, F. Gabriel, C. Hongler, A. Jacot, J. Paccolat, L. Pillaud-Vivien, S. Spigler, L. Sagun, U. Tomasini, K. Tyloo and all members of the PCSL group for discussions. This work was partially supported by the grant from the Simons Foundation (#454953 Matthieu Wyart). M.W. thanks the Swiss National Science Foundation for support under Grant No. 200021-165509.

References

[1] David Marr and Tomaso Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 204(1156):301–328, 1979.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[3] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[4] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[5] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

[6] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[8] Mario Geiger, Stefano Spigler, Stéphane d'Ascoli, Levent Sagun, Marco Baity-Jesi, Giulio Biroli, and Matthieu Wyart. Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1):012115, July 2019. Publisher: American Physical Society.

[9] Quoc V Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE, 2013.

[10] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 8580–8589, USA, 2018. Curran Associates Inc.

[11] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[12] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.

[13] Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. *Journal of Machine Learning Research*, 5(Jun):669–695, 2004.

[14] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

[15] Ludovic Berthier and Giulio Biroli. Theoretical perspective on the glass transition and amorphous materials. *Reviews of Modern Physics*, 83(2):587, 2011.

[16] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[18] Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of optimization and implicit regularization in deep learning. *arXiv preprint arXiv:1705.03071*, 2017.

[19] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

[20] Yamini Bansal, Madhu Advani, David D Cox, and Andrew M Saxe. Minnorm training: an algorithm for training overcomplete deep neural networks. *arXiv preprint arXiv:1806.00730*, 2018.

[21] Madhu S. Advani, Andrew M. Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, September 2020.

[22] C Daniel Freeman and Joan Bruna. Topology and geometry of deep rectified network optimization landscapes. *International Conference on Learning Representations*, 2017.

[23] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pages 1729–1739, 2017.

[24] Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

[25] Yaim Cooper. The loss landscape of overparameterized neural networks. *arXiv preprint arXiv:1804.10200*, 2018.

[26] Levent Sagun, Léon Bottou, and Yann LeCun. Singularity of the hessian in deep learning. *International Conference on Learning Representations*, 2017.

[27] Levent Sagun, Utku Evci, V. Uğur Güney, Yann Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *ICLR 2018 Workshop Contribution, arXiv:1706.04454*, 2017.

[28] Andrew J Ballard, Ritankar Das, Stefano Martiniani, Dhagash Mehta, Levent Sagun, Jacob D Stevenson, and David J Wales. Energy landscapes for machine learning. *Physical Chemistry Chemical Physics*, 2017.

[29] Zachary C Lipton. Stuck in a what? adventures in weight space. *International Conference on Learning Representations*, 2016.

[30] Marco Baity-Jesi, Levent Sagun, Mario Geiger, Stefano Spigler, Gerard Ben Arous, Chiara Cammarota, Yann LeCun, Matthieu Wyart, and Giulio Biroli. Comparing dynamics: Deep neural networks versus glassy systems. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 314–323, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[31] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[32] Christopher KI Williams. Computing with infinite networks. In *Advances in neural information processing systems*, pages 295–301, 1997.

[33] Jae Hoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *ICLR*, 2018.

[34] Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

[35] Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019.

[36] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[37] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[38] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A Convergence Theory for Deep Learning via Over-Parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, May 2019. ISSN: 2640-3498.

[39] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8572–8583. Curran Associates, Inc., 2019.

[40] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On Exact Computation with an Infinitely Wide Neural Net. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8141–8150. Curran Associates, Inc., 2019.

[41] Daniel Park, Jascha Sohl-Dickstein, Quoc Le, and Samuel Smith. The Effect of Network Width on Stochastic Gradient Descent and Generalization: an Empirical Study. In *International Conference on Machine Learning*, pages 5042–5051. PMLR, May 2019. ISSN: 2640-3498.

[42] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, August 2018. Publisher: National Academy of Sciences Section: PNAS Plus.

[43] Grant M Rotskoff and Eric Vanden-Eijnden. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*, 2018.

[44] Lénaïc Chizat and Francis Bach. On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport. In *Advances in Neural Information Processing Systems 31*, pages 3040–3050. Curran Associates, Inc., 2018.

[45] Justin Sirignano and Konstantinos Spiliopoulos. Mean Field Analysis of Neural Networks: A Law of Large Numbers. *SIAM Journal on Applied Mathematics*, 80(2):725–752, January 2020. Publisher: Society for Industrial and Applied Mathematics.

[46] Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on Learning Theory*, pages 2388–2464. PMLR, June 2019. ISSN: 2640-3498.

[47] Phan-Minh Nguyen. Mean field limit of the learning dynamics of multilayer neural networks. *arXiv preprint arXiv:1902.02880*, 2019.

[48] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852, 2020.

[49] Phan-Minh Nguyen and Huy Tuan Pham. A rigorous framework for the mean field limit of multilayer neural networks. *arXiv preprint arXiv:2001.11443*, 2020.

[50] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pages 2937–2947, 2019.

[51] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[52] Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, 2016.

[53] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[54] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[55] Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 6111–6122, 2019.

[56] Stefano Recanatesi, Matthew Farrell, Madhu Advani, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Dimensionality compression and expansion in deep neural networks. *arXiv preprint arXiv:1906.00443*, 2019.

[57] Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, November 2020. Publisher: IOP Publishing.

[58] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[59] S. Spigler, M. Geiger, S. d'Ascoli, L. Sagun, G. Biroli, and M. Wyart. A jamming transition from under- to over-parametrization affects generalization in deep learning. *Journal of Physics A: Mathematical and Theoretical*, 52(47):474001, October 2019. Publisher: IOP Publishing.

[60] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[61] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, February 2020. Publisher: IOP Publishing.

[62] Jaehoon Lee, Samuel S Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *arXiv preprint arXiv:2007.15801*, 2020.

[63] Corey S. O'Hern, Leonardo E. Silbert, Andrea J. Liu, and Sidney R. Nagel. Jamming at zero temperature and zero applied stress: The epitome of disorder. *Phys. Rev. E*, 68(1):011306–011324, Jul 2003.

[64] M. Wyart. On the rigidity of amorphous solids. *Annales de Phys*, 30(3):1–113, 2005.

[65] Andrea J Liu, Sidney R Nagel, W Saarloos, and Matthieu Wyart. *The jamming scenario - an introduction and outlook*. OUP Oxford, 06 2010.

[66] Matthieu Wyart, Leonardo E Silbert, Sidney R Nagel, and Thomas A Witten. Effects of compression on the vibrational modes of marginally jammed solids. *Physical Review E*, 72(5):051306, 2005.

[67] Eric DeGiuli, Adrien Laversanne-Finot, Gustavo Alberto Düring, Edan Lerner, and Matthieu Wyart. Effects of coordination and pressure on sound attenuation, boson peak and elasticity in amorphous solids. *Soft Matter*, 10(30):5628–5644, 2014.

[68] Silvio Franz, Giorgio Parisi, Pierfrancesco Urbani, and Francesco Zamponi. Universal spectrum of normal modes in low-temperature glasses. *Proceedings of the National Academy of Sciences*, 112(47):14539–14544, 2015.

[69] Carolina Brito, Harukuni Ikeda, Pierfrancesco Urbani, Matthieu Wyart, and Francesco Zamponi. Universality of jamming of nonspherical particles. *Proceedings of the National Academy of Sciences*, 115(46):11736–11741, November 2018. Publisher: National Academy of Sciences Section: Physical Sciences.

[70] Mitch Mailman, Carl F. Schreck, Corey S. O'Hern, and Bulbul Chakraborty. Jamming in systems composed of frictionless ellipse-shaped particles. *Phys. Rev. Lett.*, 102(25):255501, Jun 2009.

[71] Z. Zeravcic, N. Xu, A. J. Liu, S. R. Nagel, and W. van Saarloos. Excitations of ellipsoid packings near jamming. *Europhys. Lett.*, 87(2):26001, 2009.

[72] Matthieu Wyart. Marginal stability constrains force and pair distributions at random close packing. *Physical review letters*, 109(12):125502, 2012.

[73] Edan Lerner, Gustavo During, and Matthieu Wyart. Low-energy non-linear excitations in sphere packings. *Soft Matter*, 9:8252–8263, 2013.

[74] Patrick Charbonneau, Jorge Kurchan, Giorgio Parisi, Pierfrancesco Urbani, and Francesco Zamponi. Fractal free energy landscapes in structural glasses. *Nature Communications*, 5(3725), 2014.

[75] Patrick Charbonneau, Jorge Kurchan, Giorgio Parisi, Pierfrancesco Urbani, and Francesco Zamponi. Exact theory of dense amorphous hard spheres in high dimension. iii. the full replica symmetry breaking solution. *Journal of Statistical Mechanics: Theory and Experiment*, 2014(10):10009, 2014.

[76] Edan Lerner, Gustavo Düring, and Matthieu Wyart. A unified framework for non-brownian suspension flows and soft amorphous solids. *Proceedings of the National Academy of Sciences*, 109(13):4798–4803, 2012.

[77] E DeGiuli, G Düring, E Lerner, and M Wyart. Unified theory of inertial granular flows and non-brownian suspensions. *Physical Review E*, 91(6):062206, 2015.

[78] Florent Krzakala and Jorge Kurchan. Landscape analysis of constraint satisfaction problems. *Physical Review E*, 76(2):021122, 2007.

[79] Silvio Franz and Giorgio Parisi. The simplest model of jamming. *Journal of Physics A: Mathematical and Theoretical*, 49(14):145001, 2016.

[80] Silvio Franz, Giorgio Parisi, Maxime Sevelev, Pierfrancesco Urbani, and Francesco Zamponi. Universality of the sat-unsat (jamming) threshold in non-convex continuous constraint satisfaction problems. *SciPost Physics*, 2(3):019, 2017.

[81] Silvio Franz, Sungmin Hwang, and Pierfrancesco Urbani. Jamming in Multilayer Supervised Learning Models. *Physical Review Letters*, 123(16):160602, October 2019. Publisher: American Physical Society.

[82] Silvio Franz, Antonio Sclocchi, and Pierfrancesco Urbani. Critical jammed phase of the linear perceptron. *Physical Review Letters*, 123(11):115702, 2019.

[83] Alexei V. Tkachenko and Thomas A. Witten. Stress propagation through frictionless granular material. *Phys. Rev. E*, 60(1):687–696, Jul 1999.

[84] Matthieu Wyart. Scaling of phononic transport with connectivity in amorphous solids. *EPL (Europhysics Letters)*, 89(6):64001, 2010.

[85] Arthur Jacot, Franck Gabriel, and Clément Hongler. The asymptotic spectrum of the hessian of dnn throughout training. *arXiv preprint arXiv:1910.02875*, 2019.

[86] Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018.

[87] Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. *arXiv preprint arXiv:1909.05989*, 2019.

[88] Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. *arXiv preprint arXiv:1909.11304*, 2019.

[89] Zhengdao Chen, Grant Rotskoff, Joan Bruna, and Eric Vanden-Eijnden. A Dynamical Central Limit Theorem for Shallow Neural Networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[90] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension. *arXiv preprint arXiv:1904.12191*, 2019.

[91] Stéphane D'Ascoli, Maria Refinetti, Giulio Biroli, and Florent Krzakala. Double Trouble in Double Descent: Bias and Variance(s) in the Lazy Regime. In *International Conference on Machine Learning*, pages 2280–2290. PMLR, November 2020. ISSN: 2640-3498.

[92] Arthur Jacot, Berfin Simsek, Francesco Spadaro, Clement Hongler, and Franck Gabriel. Implicit Regularization of Random Feature Models. In *International Conference on Machine Learning*, pages 4631–4640. PMLR, November 2020. ISSN: 2640-3498.

[93] Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the Power of Infinitely Wide Deep Nets on Small-data Tasks. September 2019.

[94] Vaishaal Shankar, Alex Fang, Wenshuo Guo, Sara Fridovich-Keil, Jonathan Ragan-Kelley, Ludwig Schmidt, and Benjamin Recht. Neural Kernels Without Tangents. In *International Conference on Machine Learning*, pages 8614–8623. PMLR, November 2020. ISSN: 2640-3498.

[95] Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and Rich Regimes in Overparametrized Models. In *Conference on Learning Theory*, pages 3635–3673. PMLR, July 2020. ISSN: 2640-3498.

[96] Lénaïc Chizat and Francis Bach. Implicit Bias of Gradient Descent for Wide Two-layer Neural Networks Trained with the Logistic Loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR, July 2020. ISSN: 2640-3498.

[97] Gilad Yehudai and Ohad Shamir. On the power and limitations of random features for understanding neural networks. In *Advances in Neural Information Processing Systems*, pages 6598–6608, 2019.

[98] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Limitations of lazy training of two-layers neural network. In *Advances in Neural Information Processing Systems*, pages 9111–9121, 2019.

[99] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When Do Neural Networks Outperform Kernel Methods? *Advances in Neural Information Processing Systems*, 33, 2020.

[100] Stefano Spigler, Mario Geiger, and Matthieu Wyart. Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124001, December 2020. Publisher: IOP Publishing.

[101] Jonas Paccolat, Matthieu Wyart, and Stefano Spigler. How isotropic kernels perform on simple invariants. *Machine Learning: Science and Technology*, 2020.

[102] Sebastian Goldt, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. Modeling the Influence of Data Structure on Learning in Neural Networks: The Hidden Manifold Model. *Physical Review X*, 10(4):041044, December 2020. Publisher: American Physical Society.

[103] Jonas Paccolat, Leonardo Petrini, Mario Geiger, Kevin Tyloo, and Matthieu Wyart. Geometric compression of invariant manifolds in neural nets. *arXiv preprint arXiv:2007.11471*, 2020.

[104] Greg Ongie, Rebecca Willett, Daniel Soudry, and Nathan Srebro. A Function Space View of Bounded Norm Infinite Width ReLU Nets: The Multivariate Case. April 2020.

[105] Samet Oymak, Zalan Fabian, Mingchen Li, and Mahdi Soltanolkotabi. Generalization guarantees for neural networks via harnessing the low-rank structure of the jacobian. *arXiv preprint arXiv:1906.05392*, 2019.

[106] Dmitry Kopitkov and Vadim Indelman. Neural Spectrum Alignment: Empirical Study. *Artificial Neural Networks and Machine Learning – ICANN 2020*, 2020.

[107] Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum Dependent Learning Curves in Kernel Regression and Wide Neural Networks. In *International Conference on Machine Learning*, pages 1024–1034. PMLR, November 2020. ISSN: 2640-3498.

[108] Bernhard Scholkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Advances in Kernel Methods - Support Vector Learning*, pages 327–352. MIT Press, 1999.

[109] Eric Degiuli, E Lerner, and M Wyart. Theory of the jamming transition at finite temperature. *The Journal of chemical physics*, 142(16):164503, 2015.

Data Symmetries Part II

4 Diffeomorphisms of 2D Images: relative stability matters

The following paper is a preprint Petrini et al. (2021).

Candidate contributions The candidate contributed to this paper by proposing to deform images with diffeomorphisms and feed them to CNN to measure the response, realizing the first experiments (basically the content of appendix A and B) and participating during the discussions. The rest of the numerical experiments has been done by Leonardo Petrini.

Relative stability toward diffeomorphisms indicates performance in deep nets

Leonardo Petrini, Alessandro Favero, Mario Geiger, Matthieu Wyart

Institute of Physics
École Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland
{name.surname}@epfl.ch

Abstract

Understanding why deep nets can classify data in large dimensions remains a challenge. It has been proposed that they do so by becoming stable to diffeomorphisms, yet existing empirical measurements support that it is often not the case. We revisit this question by defining a maximum-entropy distribution on diffeomorphisms, that allows to study typical diffeomorphisms of a given norm. We confirm that stability toward diffeomorphisms does not strongly correlate to performance on benchmark data sets of images. By contrast, we find that the *stability toward diffeomorphisms relative to that of generic transformations* R_f correlates remarkably with the test error ϵ_t. It is of order unity at initialization but decreases by several decades during training for state-of-the-art architectures. For CIFAR10 and 15 known architectures we find $\epsilon_t \approx 0.2\sqrt{R_f}$, suggesting that obtaining a small R_f is important to achieve good performance. We study how R_f depends on the size of the training set and compare it to a simple model of invariant learning.

1 Introduction

Deep learning algorithms LeCun et al. (2015) are now remarkably successful at a wide range of tasks Amodei et al. (2016); Huval et al. (2015); Mnih et al. (2013); Shi et al. (2016); Silver et al. (2017). Yet, understanding how they can classify data in large dimensions remains a challenge. In particular, the curse of dimensionality associated with the geometry of space in large dimension prohibits learning in a generic setting Luxburg and Bousquet (2004). If high-dimensional data can be learnt, then they must be highly structured.

A popular idea is that during training, hidden layers of neurons learn a representation Le (2013) that is insensitive to aspects of the data unrelated to the task, effectively reducing the input dimension and making the problem tractable Ansuini et al. (2019); Recanatesi et al. (2019); Shwartz-Ziv and Tishby (2017). Several quantities have been introduced to study this effect empirically. It includes (i) the mutual information between the hidden and visible layers of neurons Saxe et al. (2019); Shwartz-Ziv and Tishby (2017), (ii) the intrinsic dimension of the neural representation of the data Ansuini et al. (2019); Recanatesi et al. (2019) and (iii) the projection of the label of the data on the main features of the network Kopitkov and Indelman (2020); Oymak et al. (2019); Paccolat et al. (2021a), the latter being defined from the top eigenvectors of the Gram matrix of the neural tangent kernel (NTK) Jacot et al. (2018). All these measures support that the neuronal representation of the data indeed becomes well-suited to the task. Yet, they are agnostic to the nature of what varies in the data that need not being represented by hidden neurons, and thus do not specify what it is.

Recently, there has been a considerable effort to understand the benefits of learning features for one-hidden-layer fully connected nets. Learning features can occur and improve performance when the

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

true function is highly anisotropic, in the sense that it depends only on a linear subspace of the input space Bach (2017); Chizat and Bach (2020); Ghorbani et al. (2019, 2020); Paccolat et al. (2021a); Refinetti et al. (2021); Yehudai and Shamir (2019). For image classification, such an anisotropy would occur for example if pixels on the edge of the image are unrelated to the task. Yet, fully-connected nets (unlike CNNs) acting on images tend to perform best in training regimes where features are not learnt Geiger et al. (2021, 2020); Lee et al. (2020), suggesting that such a linear invariance in the data is not central to the success of deep nets.

Instead, it has been proposed that images can be classified in high dimensions because classes are invariant to smooth deformations or diffeomorphisms of small magnitude Bruna and Mallat (2013); Mallat (2016). Specifically, Mallat and Bruna could handcraft convolution networks, the *scattering transforms*, that perform well and are stable to smooth transformations, in the sense that $\|f(x) - f(\tau x)\|$ is small if the norm of the diffeomorphism τ is small too. They hypothesized that during training deep nets learn to become stable and thus less sensitive to these deformations, thus improving performance. More recent works generalize this approach to more common CNNs and discuss stability at initialization Bietti and Mairal (2019a,b). Interestingly, enforcing such a stability can improve performance Kayhan and Gemert (2020).

Answering if deep nets become more stable to smooth deformations when trained and quantifying how it affects performance remains a challenge. Recent empirical results revealed that small shifts of images can change the output a lot Azulay and Weiss (2018); Dieleman et al. (2016); Zhang (2019), in apparent contradiction with that hypothesis. Yet in these works, image transformations (i) led to images whose statistics were very different from that of the training set or (ii) were cropping the image, thus are not diffeophormisms. In Ruderman et al. (2018), a class of diffeomorphisms (low-pass filter in spatial frequencies) was introduced to show that stability toward them can improve during training, especially in architectures where pooling layers are absent. Yet, these studies do not address how stability affects performance, and how it depends on the size of the training set. To quantify these properties and to find robust empirical behaviors across architectures, we will argue that the evolution of stability toward smooth deformations needs to be compared relatively to that of any deformation, which turns out to vary significantly during training.

Note that in the context of adversarial robustness, attacks that are geometric transformations of small norm that change the label have been studied Alaifari et al. (2018); Alcorn et al. (2019); Athalye et al. (2018); Engstrom et al. (2019); Fawzi and Frossard (2015); Kanbak et al. (2018); Xiao et al. (2018). These works differ for the literature above and from out study below in the sense that they consider worst-case perturbations instead of typical ones.

1.1 Our Contributions

- We introduce a *maximum entropy distribution* of diffeomorphisms, that allow us to generate typical diffeomorphisms of controlled norm. Their amplitude is governed by a "temperature" parameter T.
- We define the *relative stability to diffeomorphisms index* R_f that characterizes the square magnitude of the variation of the output function f with respect to the input when it is transformed along a diffeomorphism, relatively to that of a random transformation of the same amplitude. It is averaged on the test set as well as on the ensemble of diffeomorphisms considered.
- We find that at initialization, R_f is close to unity for various data sets and architectures, indicating that initially the output is as sensitive to smooth deformations as it is to random perturbations of the image.
- Our central result is that after training, R_f correlates very strongly with the test error ϵ_t: during training, R_f is reduced by several decades in current State Of The Art (SOTA) architectures on four benchmark datasets including MNIST Lecun et al. (1998), FashionMNIST Xiao et al. (2017), CIFAR-10 Krizhevsky (2009) and ImageNet Deng et al. (2009). For more primitive architectures (whose test error is higher) such as fully connected nets or simple CNNs, R_f remains of order unity. For CIFAR10 we study 15 known architectures and find empirically that $\epsilon_t \approx 0.2\sqrt{R_f}$.
- R_f decreases with the size of the training set P. We compare it to an inverse power $1/P$ expected in simple models of invariant learning Paccolat et al. (2021a).

The library implementing diffeomorphisms on images is available online at github.com/pcsl-epfl/diffeomorphism.
The code for training neural nets can be found at github.com/leonardopetrini/diffeo-sota and the corresponding pre-trained models at doi.org/10.5281/zenodo.5589870.

2 Maximum-entropy model of diffeomorphisms

2.1 Definition of maximum entropy model

We consider the case where the input vector x is an image. It can be thought as a function $x(s)$ describing intensity in position $s = (u, v) \in [0, 1]^2$, where u and v are the horizontal and vertical coordinates. To simplify notations we consider a single channel, in which case $x(s)$ is a scalar (but our analysis holds for colored images as well). We denote by τx the image deformed by τ, i.e. $[\tau x](s) = x(s - \tau(s))$. $\tau(s)$ is a vector field of components $(\tau_u(s), \tau_v(s))$. The deformation amplitude is measured by the norm

$$\|\nabla\tau\|^2 = \int_{[0,1]^2} ((\nabla\tau_u)^2 + (\nabla\tau_v)^2) du dv. \tag{1}$$

To test the stability of deep nets toward diffeomorphisms, we seek to build *typical* diffeomorphisms of controlled norm $\|\nabla\tau\|$. We thus consider the distribution over diffeomorphisms that maximizes the entropy with a norm constraint. It can be solved by introducing a Lagrange multiplier T and by decomposing these fields on their Fourier components, see e.g. Kardar (2007) or Appendix A. In this canonical ensemble, one finds that τ_u and τ_v are independent with identical statistics. For the picture frame not to be deformed, we impose fixed boundary conditions: $\tau = 0$ if $u = 0, 1$ or $v = 0, 1$. One then obtains:

$$\tau_u = \sum_{i,j\in\mathbb{N}^+} C_{ij} \sin(i\pi u) \sin(j\pi v) \tag{2}$$

where the C_{ij} are Gaussian variables of zero mean and variance $\langle C_{ij}^2 \rangle = T/(i^2 + j^2)$. If the picture is made of $n \times n$ pixels, the result is identical except that the sum runs on $0 < i, j \leq n$. For large n, the norm then reads $\|\nabla\tau\|^2 = (\pi^2/2)\, n^2 T$, and is dominated by high spatial frequency modes. It is useful to add another parameter c to cut-off the effect of high spatial frequencies, which can be simply done by constraining the sum in Eq.2 to $i^2 + j^2 \leq c^2$, one then has $\|\nabla\tau\|^2 = (\pi^3/8)\, c^2 T$.

Once τ is generated, pixels are displaced to random positions. A new pixelated image can then be obtained using standard interpolation methods. We use two interpolations, Gaussian and bi-linear[1], as described in Appendix C. As we shall see below, this choice does not affect our result as long as the diffeomorphism induced a displacement of order of the pixel size, or larger. Examples are shown in Fig.1 as a function of T and c.

2.2 Phase diagram of acceptable diffeomorphisms

Diffeomorphisms are bijective, which is not the case for our transformations if T is too large. When this condition breaks down, a single domain of the picture can break into several pieces, as apparent in Fig.1. It can be expressed as a condition on $\nabla\tau$ that must be satisfied in every point in space Lowe (2004), as recalled in Appendix B. This is satisfied locally with high probability if $\|\tau\|^2 \ll 1$, corresponding to $T \ll (8/\pi^3)/c^2$. In Appendix, we extract empirically a curve of similar form in the (T, c) plane at which a diffeomorphism is obtained with probability at least $1/2$. For much smaller T, diffeomorphisms are obtained almost surely.

Finally, for diffeomorphisms to have noticeable consequences, their associated displacement must be of the order of magnitude of the pixel size. Defining δ^2 as the average square norm of the pixel displacement at the center of the image in the unit of pixel size, it is straightforward to obtain from Eq.2 that asymptotically for large c (cf. Appendix B for the derivation),

$$\delta^2 = \frac{\pi}{4} n^2 T \ln(c). \tag{3}$$

The line $\delta = 1/2$ is indicated in Fig.1, using empirical measurements that add pre-asymptotic terms to Eq.3. Overall, the green region corresponds to transformations that (i) are diffeomorphisms with high probability and (ii) produce significant displacements at least of the order of the pixel size.

[1] Throughout the paper, if not specified otherwise, bi-linear interpolation is employed.

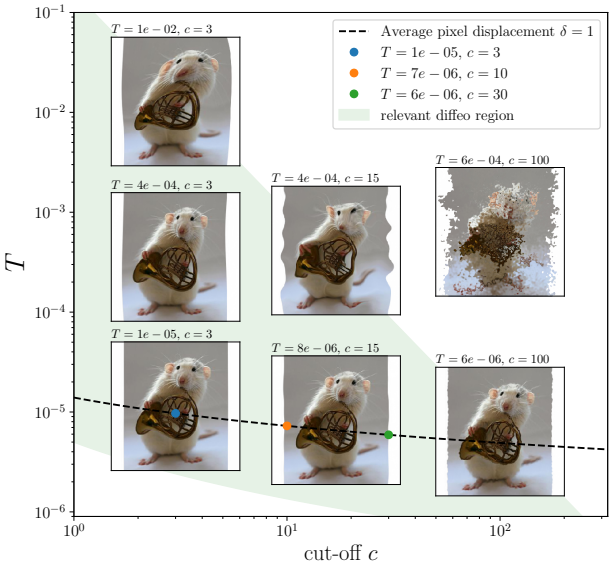


Figure 1: Samples of max-entropy diffeomorphisms for different temperatures T and high-frequency cut-offs c for an ImageNet datapoint of resolution 320×320. The green region corresponds to well behaving diffeomorphisms (see Section 2.2). The dashed line corresponds to $\delta = 1$. The colored points on the line are those we focus our study in Section 3.

3 Measuring the relative stability to diffeomorphisms

Relative stability to diffeomorphisms To quantify how a deep net f learns to become less sensitive to diffeomorphisms than to generic data transformations, we define the relative stability to diffeomorphisms R_f as:

$$R_f = \frac{\langle \|f(\tau x) - f(x)\|^2 \rangle_{x,\tau}}{\langle \|f(x+\eta) - f(x)\|^2 \rangle_{x,\eta}}. \tag{4}$$

where the notation $\langle \rangle_y$ can indicate alternatively the mean or the median with respect to the distribution of y. In the numerator, this operation is made over the test set and over the ensemble of diffeomorphisms of parameters (T, c) (on which R_f implicitly depends). In the denominator, the average is on the test set and on the vectors η sampled uniformly on the sphere of radius $\|\eta\| = \langle \|\tau x - x\| \rangle_{x,\tau}$. An illustration of what R_f captures is shown in Fig.2. In the main text, we consider median quantities, as they reflect better the typical values of distribution. In Appendix E.3 we show that our results for mean quantities, for which our conclusions also apply.

Dependence of R_f on the diffeomorphism magnitude Ideally, R_f could be defined for infinitesimal transformations, as it would then characterize the magnitude of the gradient of f along smooth deformations of the images, normalized by the magnitude of the gradient in random directions. However, infinitesimal diffeomorphisms move the image much less than the pixel size, and their definition thus depends significantly on the interpolation method used. It is illustrated in the left panels of Fig.3, showing the dependence of R_f in terms of the diffeomorphism magnitude (here characterised by the mean displacement magnitude at the center of the image δ) for several interpolation methods. We do see that R_f becomes independent of the interpolation when δ becomes of order unity. In what follows we thus focus on $R_f(\delta = 1)$, which we denote R_f.

SOTA architectures become relatively stable to diffeomorphisms during training, but are not at initialization The central panels of Fig.3 show R_f at initialization (shaded), and after training (full) for two SOTA architectures on four benchmark data sets. The first key result is that, at initialization, these architectures are as sensitive to diffeomorphisms as they are to random transformations. Relative stability to diffeomorphisms at initialization (guaranteed theoretically in some cases Bietti and Mairal (2019a,b)) thus does not appear to be indicative of successful architectures.

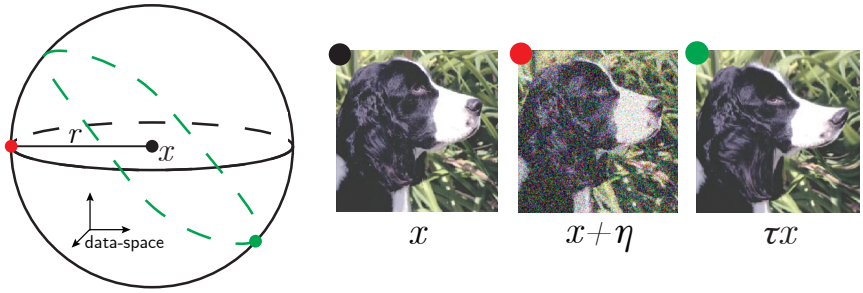


Figure 2: Illustrative drawing of the data-space $\mathbb{R}^{n\times n}$ around a data-point x (black point). We focus here on perturbations of fixed magnitude – i.e. on the sphere of radius r centered in x. The intersection between the images of x transformed via typical diffeomorphisms and the sphere is represented in dashed green. By contrast, the red point is an example of random transformation. For large n, it is equivalent to adding an i.i.d. Gaussian noise to all the pixel values of x. Figures on the right illustrate these transformations, the color of the dot labelling them corresponds to that of the left illustration. The relative stability to diffeomorphisms R_f characterizes how a net f varies in the green directions, normalized by random ones.

By contrast, for these SOTA architectures, relative stability toward diffeomorphisms builds up during training on all the data sets probed. It is a significant effect, with values of R_f after training generally found in the range $R_f \in [10^{-2}, 10^{-1}]$.

Standard data augmentation techniques (translations, crops, and horizontal flips) are employed for training. However, the results we find only mildly depend on using such techniques, see Fig.12 in Appendix.

Learning relative stability to diffeos requires large training sets How many data are needed to learn relative stability toward diffeomorphisms? To answer this question, newly initialized networks are trained on different training-sets of size P. R_f is then measured for CIFAR10, as indicated in the right panels of Fig.3. Neural nets need a certain number of training points ($P \sim 10^3$) in order to become relatively stable toward smooth deformations. Past that point, R_f monotonically decreases with P. In a range of P, this decrease is approximately compatible with the an inverse behavior $R_f \sim 1/P$ found in the simple model of Section 6. Additional results for MNIST and FashionMNIST can be found in Fig.13, Appendix E.3.

Simple architectures do not become relatively stable to diffeomorphisms To test the universality of these results, we focus on two simple architectures: (i) a 4-hidden-layer fully connected (FC) network (FullConn-L4) where each hidden layer has 64 neurons and (ii) LeNet LeCun et al. (1989) that consists of two convolutional layers followed by local max-pooling and three fully-connected layers.

Measurements of R_f for these networks are shown in Fig.4. For the FC net, $R_f \approx 1$ at initialization (as observed for SOTA nets) but *grows* after training on the full data set, showing that FC nets do not learn to become relatively stable to smooth deformations. It is consistent with the modest evolution of $R_f(P)$ with P, suggesting that huge training sets would be required to obtain $R_f < 1$. The situation is similar for the primitive CNN LeNet, which only becomes slightly insensitive ($R_f \approx 0.6$) in a single data set (CIFAR10), and otherwise remains larger than unity.

Layers' relative stability monotonically increases with depth Up to this point, we measured the relative stability of the output function for any given architecture. We now study how relative stability builds up as the input data propagate through the hidden layers. In Fig.14 of Appendix E.3, we report R_f as a function of depth for both simple and deep nets. What we observe is $R_{f_0} \approx 1$ independently

[2]With the only exception of the ImageNet results (central panel) in which only one trained network is considered.

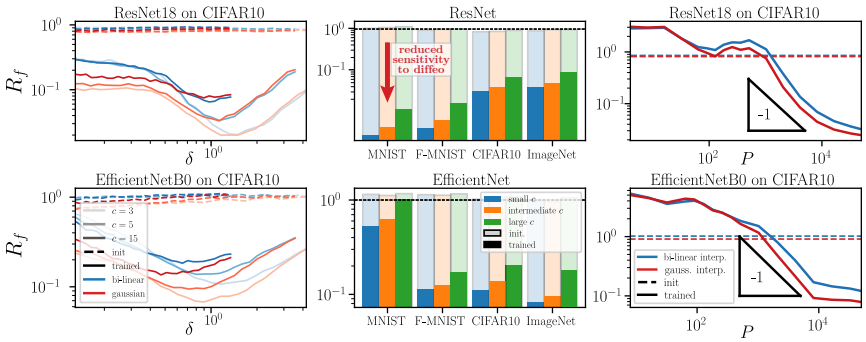


Figure 3: **Relative stability to diffeomorphisms R_f for SOTA architectures.** Left panels: R_f *vs.* diffeomorphism displacement magnitude δ at initialization (dashed lines) and after training (full lines) on the full data set of CIFAR10 ($P = 50k$) for several cut-off parameters c and two interpolations methods, as indicated in legend. ResNet is shown on the top and EfficientNet on the bottom. Central panels: $R_f(\delta = 1)$ for four different data-sets ($x-$axis) and two different architectures at initialization (shaded histograms) and after training (full histograms). The values of c (in different colors) are $(3, 5, 15)$ and $(3, 10, 30)$ for the first three data-sets and ImageNet, respectively. ResNet18 and EfficientNetB0 are employed for MNIST, F-MNIST and CIFAR10, ResNet101 and EfficientNetB2 for ImageNet. Right panels: $R_f(\delta = 1)$ *vs.* training set size P at $c = 3$ for ResNet18 (top) and EfficientNetB0 (bottom) trained on CIFAR10. The value of R_{f_0} at initialization is indicated with dashed lines. The triangles indicate the predicted slope $R_f \sim P^{-1}$ in a simple model of invariant learning, see Section 6. *Statistics*: Each point in the graphs[2] is obtained by training 16 differently initialized networks on 16 different subsets of the data-sets; each network is then probed with 500 test samples in order to measure stability to diffeomorphisms and Gaussian noise. The resulting R_f is obtained by log-averaging the results from single realizations.

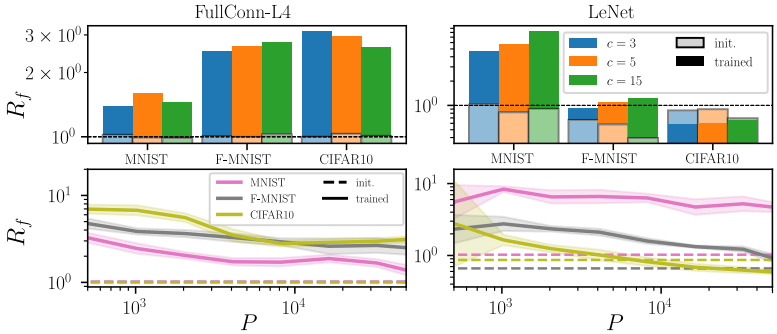


Figure 4: **Relative stability to diffeomorphisms R_f in primitive architectures.** Top panels: R_f at initialization (shaded) or for trained nets (full) for a fully connected net (left) or a primitive CNN (right) at $P = 50k$. Bottom panels: $R_f(P)$ for $c = 3$ and different data sets as indicated in legend. *Statistics:* see caption in the previous figure.

of depth at initialization, and monotonically decreases with depth after training. Overall, the gain in relative stability appears to be well-spread through the net, as is also found for stability alone Ruderman et al. (2018).

4 Relative stability to diffeomorphisms indicates performance

Thus, SOTA architectures appear to become relatively stable to diffeomorphisms after training, unlike primitive architectures. This observation suggests that high performance requires such a relative stability to build up. To test further this hypothesis, we select a set of architectures that have been relevant in the state of the art progress over the past decade; we systematically train them in order to compare R_f to their test error ϵ_t. Apart from fully connected nets, we consider the already cited LeNet (5 layers and $\approx 60k$ parameters); then AlexNet Krizhevsky et al. (2012) and VGG Simonyan and Zisserman (2015), deeper (8-19 layers) and highly over-parametrized (10-20M (million) params.) versions of the latter. We introduce *batch-normalization* in VGGs and *skip connections* with ResNets. Finally, we go to EfficientNets, that have all the advancements introduced in previous models and achieve SOTA performance with a relatively small number of parameters (<10M); this is accomplished by designing an efficient small network and properly scaling it up. Further details about these architectures can be found in Table 1, Appendix E.2.

The results are shown in Fig.5. The correlation between R_f and ϵ_t is remarkably high (corr. coeff.[3] : 0.97), suggesting that generating low relative sensitivity to diffeomorphisms R_f is important to obtain good performance. In Appendix E.3 we also report how changing the train set size P affects the position of a network in the (ϵ_t, R_f) plane, for the four architectures considered in the previous section (Fig.18). We also show that our results are robust to changes of δ, c (Fig.21) and data sets (Fig.20).

What architectures enable a low R_f value? The latter can be obtained with skip connections or not, and for quite different depths as indicated in Fig.5. Also, the same architecture (EfficientNetB0) trained by transfer learning from ImageNet – instead of directly on CIFAR10 – shows a large improvement both in performance and in diffeomorphisms invariance. Clearly, R_f is much better predicted by ϵ_t than by the specific features of the architecture indicated in Fig.5.

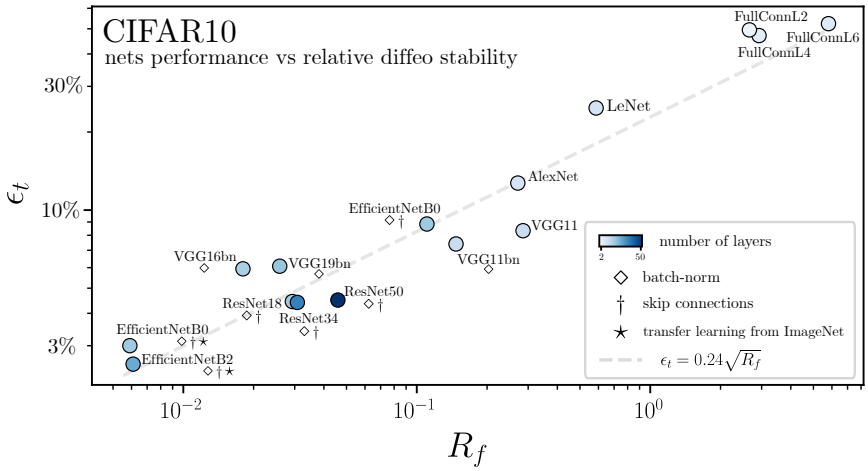


Figure 5: **Test error ϵ_t *vs.* relative stability to diffeomorphisms** R_f computed at $\delta = 1$ and $c = 3$ for common architectures when trained on the full 10-classes CIFAR10 dataset ($P = 50k$) with SGD and the cross-entropy loss; the EfficientNets achieving the best performance are trained by transfer learning from ImageNet ($\star$) – more details on the training procedures can be found in Appendix E.1. The color scale indicates depth, and the symbols the presence of batch-norm ($\diamond$) and skip connections ($\dagger$). Dashed grey line: power low fit $\epsilon_t \approx 0.2\sqrt{R_f}$. R_f strongly correlates to ϵ_t, much less so to depth or the presence of skip connections. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure R_f. The results are obtained by log-averaging over single realizations. Error bars – omitted here – are shown in Fig.19, Appendix E.3.

5 Stability toward diffeomorphisms *vs.* noise

The relative stability to diffeomorphisms R_f can be written as $R_f = {}^{D_f}/_{G_f}$ where G_f characterizes the stability with respect to additive noise and D_f the stability toward diffeomorphisms:

$$G_f = \frac{\langle \|f(x+\eta) - f(x)\|^2 \rangle_{x,\eta}}{\langle \|f(x) - f(z)\|^2 \rangle_{x,z}}, \qquad D_f = \frac{\langle \|f(\tau x) - f(x)\|^2 \rangle_{x,\tau}}{\langle \|f(x) - f(z)\|^2 \rangle_{x,z}}. \tag{5}$$

Here, we chose to normalize these stabilities with the variation of f over the test set (to which both x and z belong), and η is a random noise whose magnitude is prescribed as above. Stability toward additive noise has been studied previously in fully connected architectures Novak et al. (2018) and for CNNs as a function of spatial frequency in Tsuzuku and Sato (2019); Yin et al. (2019).

The decrease of R_f with growing training set size P could thus be due to an increase in the stability toward diffeomorphisms (i.e. D_f decreasing with P) or a decrease of stability toward noise (G_f increasing with P). To test these possibilities, we show in Fig.6 $G_f(P), D_f(P)$ and $R_f(P)$ for MNIST, Fashion MNIST and CIFAR10 for two SOTA architectures. The central results are that (i) stability toward noise is always reduced for larger training sets. This observation is natural: when more data needs to be fitted, the function becomes rougher. (ii) Stability toward diffeomorphisms does not behave universally: it can increase with P or decrease depending on the architecture and the training set. Additionally, G_f and D_f alone show a much smaller correlation with performance than R_f– see Figs.15,16,17 in Appendix E.3.

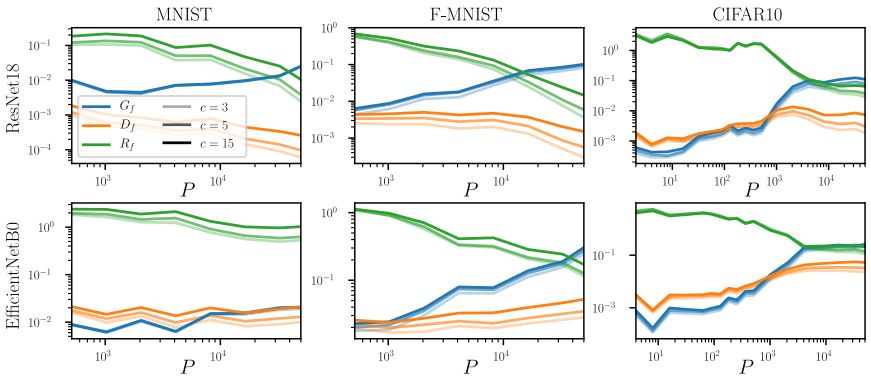


Figure 6: **Stability toward Gaussian noise (G_f) and diffeomorphisms (D_f) alone, and the relative stability R_f.** Columns correspond to different data-sets (MNIST, FashionMNIST and CIFAR10) and rows to architectures (ResNet18 and EfficientNetB0). Each panel reports G_f (blue), D_f (orange) and R_f (green) as a function of P and for different cut-off values c, as indicated in the legend. *Statistics:* cf. caption in Fig.3. Error bars – omitted here – are shown in Fig.22, Appendix E.3.

6 A minimal model for learning invariants

In this section, we discuss the simplest model of invariance in data where stability to transformation builds up, that can be compared with our observations of R_f above. Specifically, we consider the "stripe" model Paccolat et al. (2021b), corresponding to a binary classification task for Gaussian-distributed data points $x = (x_\parallel, x_\perp)$ where the label function depends only on one direction in data space, namely $y(x) = y(x_\parallel)$. Layers of $y = +1$ and $y = -1$ regions alternate along the direction $x_\parallel$, separated by parallel planes. Hence, the data present $d - 1$ invariant directions in input-space denoted by $x_\perp$ as illustrated in Fig.7-left.

When this model is learnt by a one-hidden-layer fully connected net, the first layer of weights can be shown to align with the informative direction Paccolat et al. (2021a). The projection of these weights

[3] Correlation coefficient: $\frac{\mathrm{Cov}(\log \epsilon_t, \log R_f)}{\sqrt{\mathrm{Var}(\log \epsilon_t)\mathrm{Var}(\log R_f)}}$.

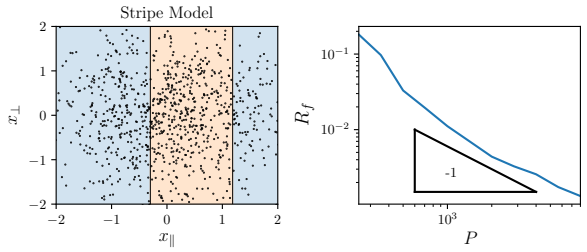


Figure 7: Left: example of the stripe model. Dots are datapoints, the vertical lines represent the decision boundary and the color the class label. Right: Relative stability R_f for the stripe model in $d = 30$. The slope of the curve is -1, as predicted.

on the orthogonal space vanishes with the training set size P as $1/\sqrt{P}$, an effect induced by the sampling noise associated to finite training sets.

In this model, R_f can be defined as:

$$R_f = \frac{\langle \|f(x_\parallel, x_\perp + \nu) - f(x_\parallel, x_\perp)\|^2 \rangle_{x,\nu}}{\langle \|f(x+\eta) - f(x)\|^2 \rangle_{x,\eta}}, \tag{6}$$

where we made explicit the dependence of f on the two linear subspaces. Here, the isotropic noise ν is added only in the invariant directions. Again, we impose $\|\eta\| = \|\nu\|$. $R_f(P)$ is shown in Fig. 7-right. We observe that $R_f(P) \sim P^{-1}$, as expected from the weight alignment mentioned above.

Interestingly, Fig.3 for CIFAR10 and SOTA architectures support that the $1/P$ behavior is compatible with the observations for some range of P. In Appendix E.3, Fig.13, we show analogous results for MNIST and Fashion-MNIST. We observe the $1/P$ power-law scaling for ResNets. It suggests that for these architectures, learning to become invariant to diffeomorphisms may be limited by a naive measure of sampling noise as well. By contrast for EfficientNets, in which the decrease in R_f is more limited, a $1/P$ behavior cannot be identified.

7 Discussion

A common belief is that stability to random noise (small G_f) and to diffeomorphisms (small D_f) are desirable properties of neural nets. Its underlying assumption is that the true data label mildly depends on such transformations when they are small. Our observations suggest an alternative view:

1. Figs.6,16: better predictors are more sensitive to small perturbations in input space.
2. As a consequence, the notion that predictors are especially insensitive to diffeomorphisms is not captured by stability alone, but rather by the relative stability $R_f = {}^{D_f}/_{G_f}$.
3. We propose the following interpretation of Fig.5: to perform well, the predictor must build large gradients in input space near the decision boundary – leading to a large G_f overall. Networks that are relatively insensitive to diffeomorphisms (small R_f) can discover with less data that strong gradients must be there and generalize them to larger regions of input space, improving performance and increasing G_f.

This last point can be illustrated in the simple model of Section 6, see Fig.7-left panel. Imagine two data points of different labels falling close to the – e.g. – left true decision boundary. These two points can be far from each other if their orthogonal coordinates differ. Yet, if $R_f = 0$ (now defined in Eq.6), then the output does not depend on the orthogonal coordinates, and it will need to build a strong gradient – in input space – along the parallel coordinate to fit these two data. This strong gradient will exist throughout that entire decision boundary, improving performance but also increasing G_f. Instead, if $R_f = 1$, fitting these two data will not lead to a strong gradient, since they can be far from each other in input space. Beyond this intuition, in this model decreasing R_f can quantitatively be shown to increase performance, see Paccolat et al. (2021b).

8 Conclusion

We have introduced a novel empirical framework to characterize how deep nets become invariant to diffeomorphisms. It is jointly based on a maximum-entropy distribution for diffeomorphisms, and on the realization that stability of these transformations relative to generic ones R_f strongly correlates to performance, instead of just the diffeomorphisms stability considered in the past.

The ensemble of smooth deformations we introduced may have interesting applications. It could serve as a complement to traditional data-augmentation techniques (whose effect on relative stability is discussed in Fig.12 of the Appendix). A similar idea is present in Hauberg et al. (2016); Shen et al. (2020) but our deformations have the advantage of being easier to sample and data agnostic. Moreover, the ensemble could be used to build adversarial attacks along smooth transformations, in the spirit of Alaifari et al. (2018); Engstrom et al. (2019); Kanbak et al. (2018). It would be interesting to test if networks robust to such attacks are more stable in relative terms, and how such robustness affects their performance.

Finally, the tight correlation between relative stability R_f and test error ϵ_t suggests that if a predictor displays a given R_f, its performance may be bounded from below. The relationships we observe $\epsilon_t(R_f)$ may then be indicative of this bound, which would be a fundamental property of a given data set. Can it be predicted in terms of simpler properties of the data? Introducing simplified models of data with controlled stability to diffeomorphisms beyond the toy model of Section 6 would be useful to investigate this key question.

Acknowledgements

We thank Alberto Bietti, Joan Bruna, Francesco Cagnetta, Pascal Frossard, Jonas Paccolat, Antonio Sclocchi and Umberto M. Tomasini for helpful discussions. This work was supported by a grant from the Simons Foundation (#454953 Matthieu Wyart).

References

Alaifari, R., Alberti, G. S., and Gauksson, T. (2018). ADef: an Iterative Algorithm to Construct Adversarial Deformations.

Alcorn, M. A., Li, Q., Gong, Z., Wang, C., Mai, L., Ku, W.-S., and Nguyen, A. (2019). Strike (With) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4840–4849, Long Beach, CA, USA. IEEE.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182.

Ansuini, A., Laio, A., Macke, J. H., and Zoccolan, D. (2019). Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 6111–6122.

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. (2018). Synthesizing Robust Adversarial Examples. In *International Conference on Machine Learning*, pages 284–293. PMLR. ISSN: 2640-3498.

Azulay, A. and Weiss, Y. (2018). Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*.

Bach, F. (2017). Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681.

Beale, P. (1996). *Statistical Mechanics*. Elsevier Science.

Bietti, A. and Mairal, J. (2019a). Group invariance, stability to deformations, and complexity of deep convolutional representations. *The Journal of Machine Learning Research*, 20(1):876–924.

Bietti, A. and Mairal, J. (2019b). On the inductive bias of neural tangent kernels. *arXiv preprint arXiv:1905.12173*.

Bruna, J. and Mallat, S. (2013). Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886.

Chizat, L. and Bach, F. (2020). Implicit Bias of Gradient Descent for Wide Two-layer Neural Networks Trained with the Logistic Loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR. ISSN: 2640-3498.

Deng, J., Dong, W., Socher, R., Li, L., Kai Li, and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. ISSN: 1063-6919.

Dieleman, S., De Fauw, J., and Kavukcuoglu, K. (2016). Exploiting cyclic symmetry in convolutional neural networks. *arXiv preprint arXiv:1602.02660*.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. (2019). Exploring the Landscape of Spatial Robustness. In *International Conference on Machine Learning*, pages 1802–1811. PMLR. ISSN: 2640-3498.

Fawzi, A. and Frossard, P. (2015). Manitest: Are classifiers really invariant? In *Procedings of the British Machine Vision Conference 2015*, pages 106.1–106.13, Swansea. British Machine Vision Association.

Geiger, M., Petrini, L., and Wyart, M. (2021). Landscape and training regimes in deep learning. *Physics Reports*.

Geiger, M., Spigler, S., Jacot, A., and Wyart, M. (2020). Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301. Publisher: IOP Publishing.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. (2019). Limitations of lazy training of two-layers neural network. In *Advances in Neural Information Processing Systems*, pages 9111–9121.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. (2020). When Do Neural Networks Outperform Kernel Methods? *Advances in Neural Information Processing Systems*, 33.

Hauberg, S., Freifeld, O., Larsen, A. B. L., Fisher III, J. W., and Hansen, L. K. (2016). Dreaming More Data: Class-dependent Distributions over Diffeomorphisms for Learned Data Augmentation. *arXiv:1510.02795 [cs]*. arXiv: 1510.02795.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. ISSN: 1063-6919.

Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., et al. (2015). An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 8580–8589, USA. Curran Associates Inc.

Kanbak, C., Moosavi-Dezfooli, S.-M., and Frossard, P. (2018). Geometric Robustness of Deep Networks: Analysis and Improvement. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4441–4449, Salt Lake City, UT. IEEE.

Kardar, M. (2007). *Statistical physics of fields*. Cambridge University Press.

Kayhan, O. S. and Gemert, J. C. v. (2020). On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285.

Kopitkov, D. and Indelman, V. (2020). Neural Spectrum Alignment: Empirical Study. *Artificial Neural Networks and Machine Learning – ICANN 2020*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Le, Q. V. (2013). Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. Conference Name: Proceedings of the IEEE.

Lee, J., Schoenholz, S. S., Pennington, J., Adlam, B., Xiao, L., Novak, R., and Sohl-Dickstein, J. (2020). Finite versus infinite neural networks: an empirical study. *arXiv preprint arXiv:2007.15801*.

Loshchilov, I. and Hutter, F. (2016). SGDR: Stochastic Gradient Descent with Warm Restarts.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Luxburg, U. v. and Bousquet, O. (2004). Distance-based classification with lipschitz functions. *Journal of Machine Learning Research*, 5(Jun):669–695.

Mallat, S. (2016). Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*.

Oymak, S., Fabian, Z., Li, M., and Soltanolkotabi, M. (2019). Generalization guarantees for neural networks via harnessing the low-rank structure of the jacobian. *arXiv preprint arXiv:1906.05392*.

Paccolat, J., Petrini, L., Geiger, M., Tyloo, K., and Wyart, M. (2021a). Geometric compression of invariant manifolds in neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(4):044001. Publisher: IOP Publishing.

Paccolat, J., Spigler, S., and Wyart, M. (2021b). How isotropic kernels perform on simple invariants. *Machine Learning: Science and Technology*, 2(2):025020. Publisher: IOP Publishing.

Recanatesi, S., Farrell, M., Advani, M., Moore, T., Lajoie, G., and Shea-Brown, E. (2019). Dimensionality compression and expansion in deep neural networks. *arXiv preprint arXiv:1906.00443*.

Refinetti, M., Goldt, S., Krzakala, F., and Zdeborová, L. (2021). Classifying high-dimensional gaussian mixtures: Where kernel methods fail and neural networks succeed. *arXiv preprint arXiv:2102.11742*.

Ruderman, A., Rabinowitz, N. C., Morcos, A. S., and Zoran, D. (2018). Pooling is neither necessary nor sufficient for appropriate deformation stability in CNNs. *arXiv:1804.04438 [cs, stat]*. arXiv: 1804.04438.

Saxe, A. M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B. D., and Cox, D. D. (2019). On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020.

Shen, Z., Xu, Z., Olut, S., and Niethammer, M. (2020). Anatomical Data Augmentation via Fluid-Based Image Registration. In Martel, A. L., Abolmaesumi, P., Stoyanov, D., Mateus, D., Zuluaga, M. A., Zhou, S. K., Racoceanu, D., and Joskowicz, L., editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, Lecture Notes in Computer Science, pages 318–328, Cham. Springer International Publishing.

Shi, B., Bai, X., and Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304.

Shwartz-Ziv, R. and Tishby, N. (2017). Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR*.

Tan, M. and Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR. ISSN: 2640-3498.

Tsuzuku, Y. and Sato, I. (2019). On the structural sensitivity of deep convolutional networks to the directions of fourier basis functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 51–60.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. (2018). Spatially Transformed Adversarial Examples.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*. arXiv: 1708.07747.

Yehudai, G. and Shamir, O. (2019). On the power and limitations of random features for understanding neural networks. In *Advances in Neural Information Processing Systems*, pages 6598–6608.

Yin, D., Lopes, R. G., Shlens, J., Cubuk, E. D., and Gilmer, J. (2019). A fourier perspective on model robustness in computer vision. *arXiv preprint arXiv:1906.08988*.

Zhang, R. (2019). Making convolutional networks shift-invariant again. *arXiv preprint arXiv:1904.11486*.

A Maximum entropy calculation

Under the constraint on the borders, τ_u and τ_v can be expressed in a real Fourier basis as in Eq.2. By injecting this form into $\|\nabla\tau\|^2$ we obtain:

$$\|\nabla\tau\|^2 = \frac{\pi^2}{4}\sum_{i,j\in\mathbb{N}^+}(C_{ij}^2 + D_{ij}^2)(i^2+j^2) \tag{7}$$

where D_{ij} are the Fourier coefficients of τ_v. We aim at computing the probability distributions that maximize their entropy while keeping the expectation value of $\|\nabla\tau\|^2$ fixed. Since we have a sum of quadratic random variables, the equipartition theorem Beale (1996) applies: the distributions are normal and every quadratic term contributes in average equally to $\|\nabla\tau\|^2$. Thus, the variance of the coefficients follows $\frac{T}{i^2+j^2}$ where the parameter T determines the magnitude of the diffeomorphism.

B Boundaries of studied diffeomorphisms

Average pixel displacement magnitude δ We derive here the large-c asymptotic behavior of δ (Eq.3). This is defined as the average square norm of the displacement field, in pixel units:

$$\begin{aligned}
\delta^2 &= n^2\int_{[0,1]^2}\|\tau(u,v)\|^2 du dv \\
&= 2Tn^2\sum_{i^2+j^2\leq c^2}\frac{1}{i^2+j^2}\int_{[0,1]^2}\sin^2(i\pi u)\sin^2(j\pi v)dudv \\
&= \frac{Tn^2}{2}\sum_{i^2+j^2\leq c^2}\frac{1}{i^2+j^2} \\
&\approx \frac{Tn^2}{2}\int_{1\leq x^2+y^2\leq c^2}\frac{1}{x^2+y^2}dxdy \\
&= \frac{\pi Tn^2}{4}\int_1^c\frac{1}{r}dr \\
&= \frac{\pi}{4}n^2T\log c,
\end{aligned}$$

where we approximated the sum with an integral, in the third step. The asymptotic relations for $\|\nabla\tau\|$ that are reported in the main text are computed in a similar fashion. In Fig.8, we check the agreement between asymptotic prediction and empirical measurements. If $\delta \ll 1$, our results strongly depend on the choice of interpolation method. To avoid it, we only consider conditions for which $\delta \geq 1/2$, leading to

$$T > \frac{1}{\pi n^2\log c}. \tag{8}$$

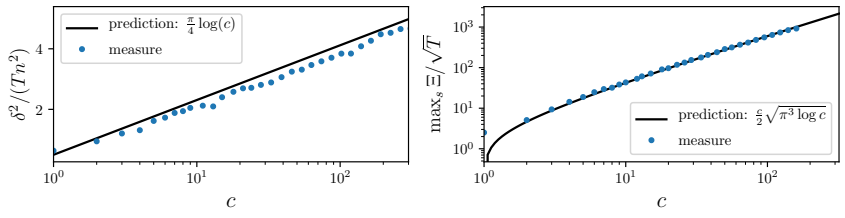
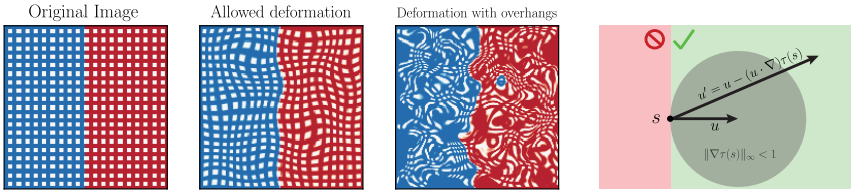


Figure 8: Left: The characteristic displacement $\delta(c,T)$ is observed to follow $\delta^2 \simeq \frac{\pi}{4}n^2T\log c$. Right: measurement of $\max_s \Xi$ supporting Eq.13.



(a) (b) (c) (d)

Figure 9: (a) Idealized image at $T = 0$. (b) Diffeomorphism of the image. (c) Deformation of the image at large T: colors get mixed-up together, shapes are not preserved anymore. (d) Allowed region for vector transformations under τ. For any point in the image s and any direction u, only displacement fields for which all the deformed direction u' is non-zero generate diffeomorphisms. The bound in Eq.12 ($u' \cdot u > 0$) correspond to the green region. The gray disc corresponds to the bound $\|\nabla\tau\|_\infty < 1$.

Condition for diffeomorphism in the (T, c) plane For a given value of c, there exists a temperature scale beyond which the transformation is not injective anymore, affecting the topology of the image and creating spurious boundaries, see Fig.9a-c for an illustration. Specifically, consider a curve passing by the point s in the deformed image. Its tangent direction is u at the point s. When going back to the original image ($s' = s - \tau(s)$) the curve gets deformed and its tangent becomes

$$u' = u - (u \cdot \nabla)\tau(s). \tag{9}$$

A smooth deformation is bijective iff all deformed curves remain curves which is equivalent to have non-zero tangents everywhere

$$\forall\, s, u \neq 0 \quad \|u'\| \neq 0. \tag{10}$$

Imposing $\|u'\| \neq 0$ does not give us any constraint on τ. Therefore, we constraint τ a bit more and allow only displacement fields such that $u \cdot u' > 0$, which is a sufficient condition for Eq.10 to be satisfied – cf. Fig. 9d. By extremizing over u, this condition translates into

$$\tfrac{1}{2}\left(\sqrt{(\partial_x\tau_x - \partial_y\tau_y)^2 + (\partial_x\tau_y + \partial_y\tau_x)^2} - \partial_x\tau_x - \partial_y\tau_y\right) < 1 \tag{11}$$

or, equivalently,

$$\Xi = \tfrac{1}{2}\left(\sqrt{||\nabla\tau||^2 - 2\det(\nabla\tau)} - \mathrm{Tr}(\nabla\tau)\right) < 1, \tag{12}$$

were we identified by Ξ the l.h.s. of the inequality. We find that the median of the maximum of Ξ over all the image ($\|\Xi(s)\|_\infty$) can be approximated by (see Fig.8b):

$$\max_s \Xi \simeq \frac{c}{2}\sqrt{\pi^3 T \log c}. \tag{13}$$

The resulting constraint on T reads

$$T < \frac{4}{\pi^3 c^2 \log c}. \tag{14}$$

C Interpolation methods

When a deformation is applied to an image x, each of its pixels gets mapped, from the original pixels grid, to new positions generally outside of the grid itself – cf. Fig. 9a-b. A procedure (interpolation method) needs to be defined to project the deformed image back into the original grid.

For simplicity of notation, we describe interpolation methods considering the square $[0, 1]^2$ as the region in between four pixels – see an illustration in Fig. 10a. We propose here two different ways to interpolate between pixels and then check that our measurements do not depend on the specific method considered.

Bi-linear Interpolation The bi-linear interpolation consists, as the name suggests, of two steps of linear interpolation, one on the horizontal, and one on the vertical direction – Fig. 10b. If we look at the square $[0, 1]^2$ and we apply a deformation τ such that $(0, 0) \mapsto (u, v)$, we have

$$x(u, v) = x(0, 0)(1 - u)(1 - v) + x(1, 0)u(1 - v) + x(0, 1)(1 - u)v + x(1, 1)uv. \tag{15}$$

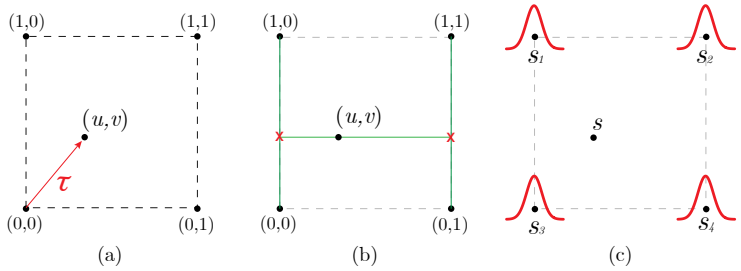


Figure 10: (a) We consider the region between four pixels as the square $[0,1]^2$ where, after the application of a deformation τ, the pixel $(0,0)$ is mapped into (u,v). (b) **Bi-linear interpolation**: the value of x in (u,v) is computed by two steps of linear interpolation. First, we compute x in the red crosses, by averaging values on the vertical axis. Then, a line interpolates horizontally the values in the red crosses to give the result. (c) **Gaussian interpolation**: we denote by s_i the pixel positions in the original grid. The interpolated value of s in any point of the image is given by a weighted sum of $n \times n$ Gaussian centered in each s_i – in red.

Gaussian Interpolation In this case, a Gaussian function[4] is placed on top of each point in the grid – cf. Fig.10. The pixel intensity x can be evaluated at any point outside the grid by computing

$$x(s) = \frac{\sum_i x(s_i) G(s - s_i)}{\sum_i G(s - s_i)}. \tag{16}$$

In order to fix the standard deviation σ of G, we introduce the *participation ratio* n. Given $\Psi_i = G(s, s_i)|_{s=(0.5,0.5)}$, we define

$$n = \frac{\left(\sum_i \Psi_i^2\right)^2}{\sum_i \Psi_i^4}. \tag{17}$$

The participation ratio is a measure of how many pixels contribute to the value of a new pixel, which results from interpolation. We fix σ in such a way that the participation ratio for the Gaussian interpolation matches the one for the bi-linear ($n = 4$), when the new pixel is equidistant from the four pixels around. This gives $\sigma = 0.4715$.

Notice that this interpolation method is such that it applies a Gaussian smoothing of the image even if τ is the identity. Consequently, when computing observables for f with the Gaussian interpolation, we always compare $f(\tau x)$ to $f(\tilde{x})$, where $\tilde{x}$ is the smoothed version of x, in such a way that $f(\tau^{[T=0]} x) = f(\tilde{x})$.

Empirical results dependence on interpolation Finally, we checked to which extent our results are affected by the specific choice of interpolation method. In particular, blue and red colors in Figs3, 13 correspond to bi-linear and Gaussian interpolation, respectively. The interpolation method only affects the results in the small displacement limit ($\delta \to 0$).

Note: throughout the paper, if not specified otherwise, bi-linear interpolation is employed.

[4] $G(s) = (2\pi\sigma^2)^{-1/2} e^{-s^2/2\sigma^2}$.

D Stability to additive noise *vs.* noise magnitude

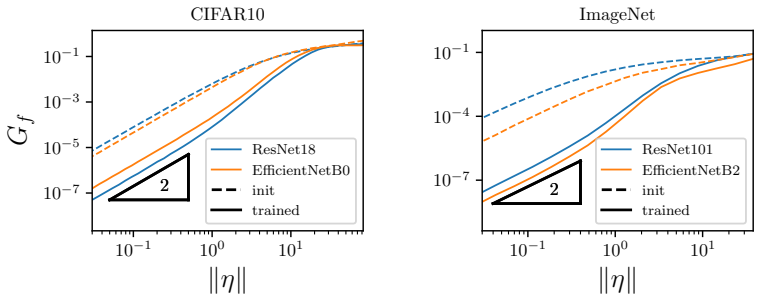


Figure 11: **Stability to isotropic noise** G_f as a function of the noise magnitude $\|\eta\|$ for CIFAR10 (left) and ImageNet (right). The color corresponds to two different classes of SOTA architecture: ResNet and EfficientNet. The slope 2 at small $\|\eta\|$ identifies the linear regime. For larger noise magnitudes, non-linearities appear.

We introduced in Section 5 the stability toward additive noise:

$$G_f = \frac{\langle \|f(x+\eta) - f(x)\|^2 \rangle_{x,\eta}}{\langle \|f(x) - f(z)\|^2 \rangle_{x,z}}. \tag{18}$$

We study here the dependence of G_f on the noise magnitude $\|\eta\|$. In the $\eta \to 0$ limit, we expect the network function to behave as its first-order Taylor expansion, leading to $G_f \propto \|\eta\|^2$. Hence, for small noise, G_f gives an estimate of the average magnitude of the gradient of f in a random direction η.

Empirical results Measurements of G_f on SOTA nets trained on benchmark data-sets are shown in Figure 11. We observe that the effect of non-linearities start to be significant around $\|\eta\| = 1$. For large values of the noise – i.e. far away from data-points – the average gradient of f does not change with training.

E Numerical experiments

In this Appendix, we provide details on the training procedure, on the different architectures employed and some additional experimental results.

E.1 Image classification training set-up:

- Trainings are performed in `PyTorch`, the code can be found here github.com/leonardopetrini/diffeo-sota.
- Loss function: cross-entropy.
- Batch size: 128.
- Dynamics:
  - Fully connected nets: ADAM with `learning rate` $= 0.1$ and no scheduling.
  - Transfer learning: SGD with `learning rate` $= 10^{-2}$ for the last layer and 10^{-3} for the rest of the network, `momentum` $= 0.9$ and `weight decay` $= 10^{-3}$. Both learning rates decay exponentially during training with a factor $\gamma = 0.975$.
  - All the other networks are trained with SGD with `learning rate` $= 0.1$, `momentum` $= 0.9$ and `weight decay` $= 5 \times 10^{-4}$. The learning rate follows a cosine annealing scheduling Loshchilov and Hutter (2016).

- Early-stopping is performed – i.e. results shown are computed with the network obtaining the best validation accuracy out of 250 training epochs.
- For the experiments involving a training on a subset of the training date of size $P < P_{\max}$, the total number of epochs is accordingly re-scaled in order to keep constant the total number of optimizer steps.
- Standard data augmentation is employed: different random translations and horizontal flips of the input images are generated at each epoch. As a safety check, we verify that the invariance learnt by the nets is not purely due to such augmentation (Fig.12).
- Experiments are run on 16 GPUs NVIDIA V100. Individual trainings run in ~ 1 hour of wall time. We estimate a total of a few thousands hours of computing time for running the preliminary and actual experiments present in this work.

The stripe model is trained with an approximation of gradient flow introduced in Geiger et al. (2020), see Paccolat et al. (2021a) for details.

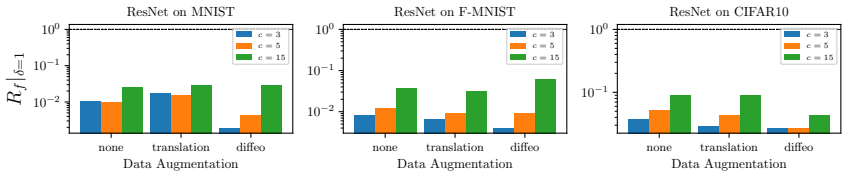


Figure 12: **Effect of data augmentation on R_f.** Relative stability to diffeomorphisms R_f after training with different data augmentations: "none" (1st group of bars in each plot) for no data augmentation, "translation" (2nd bars) corresponds to training on randomly translated (by 4 pixels) and cropped inputs, and "diffeo" (3rd bars) to training on randomly deformed images with max-entropy diffeomorphisms ($T = 10^{-2}$, $c = 1$). Results are averaged over 5 trainings of ResNet18 on MNIST (left), FashionMNIST (center), CIFAR10 (right). Colors indicate different cut-off values when probing the trained networks. Different augmentations have a small quantitative, and no qualitative effect on the results. As expected, augmenting the input images with smooth deformations makes the net more invariant to such transformations.

A note on computing stabilities at init. in presence of batch-norm We recall that batch-norm (BN) can work in either of two modes: *training* and *evaluation*. During training, BN computes the mean and variance on the current batch and uses them to normalize the output of a given layer. At the same time, it keeps memory of the running statistics on such batches, and this is used for the normalization steps at inference time (evaluation mode). When probing a network at initialization for computing stabilities, we put the network in evaluation mode, except for batch-norm (BN), which operates in train mode. This is because BN running mean and variance are initialized to 0 and 1, in such a way that its evaluation mode at initialization would correspond to not having BN at all, compromising the input signal propagation in deep architectures.

E.2 Networks architectures

All networks implementations can be found at github.com/leonardopetrini/diffeo-sota/tree/main/models. In Table 1, we report salient features of the network architectures considered.

Table 1: **Network architectures, main characteristics.** We list here (columns) the classes of net architectures used throughout the paper specifying some salient features (depth, number of parameters, etc...) for each of them.

features	**FullConn**	**LeNet** LeCun et al. (1989)	**AlexNet** Krizhevsky et al. (2012)
depth	2, 4, 6	5	8
num. parameters	200k	62k	23 M
FC layers	2, 4, 6	3	3
activation	ReLU	ReLU	ReLU
pooling	/	max	max
dropout	/	/	yes
batch norm	/	/	/
skip connections	/	/	/

features	**VGG** Simonyan and Zisserman (2015)	**ResNet** He et al. (2016)	**EfficientNetB0-2** Tan and Le (2019)
depth	11, 16, 19	18, 34, 50	18, 25
num. parameters	9-20 M	11-24 M	5, 9 M
FC layers	1	1	1
activation	ReLU	ReLU	swish
pooling	max	avg. (last layer only)	avg. (last layer only)
dropout	/	/	yes + dropconnect
batch norm	if 'bn' in name	yes	yes
skip connections	/	yes	yes (inv. residuals)

E.3 Additional figures

We present here:

- Fig.13: R_f as a function of P for MNIST and FashionMNIST with the corresponding predicted slope, omitted in the main text.
- Fig.14: Relative diffeomorphisms stability R_f as a function of depth for simple and deep nets.
- Figs15,16: diffeomorphisms and inverse of the Gaussian stability D_f and $1/G_f$ *vs.* test error for CIFAR10 and the set of architectures considered in Section 4.
- Fig.17: D_f, $1/G_f$ and R_f when using the mean in place of the median for computing averages $\langle\cdot\rangle$.
- Fig.18: curves in the (ϵ_t, R_f) plane when varying the training set size P for FullyConnL4, LeNet, ResNet18 and EfficientNetB0.
- Figs19, 22: error estimates for the main quantities of interest – often omitted in the main text for the sake of figures' clarity.

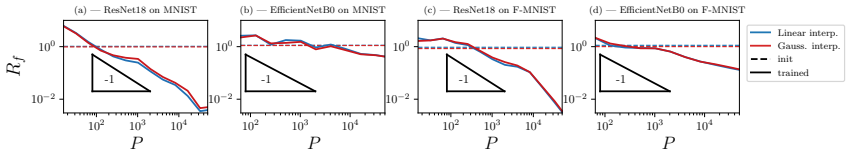


Figure 13: **Relative stability to diffeomorphisms $R_f(P)$ at $\delta = 1$.** Analogous to Figure 3-right but here we have MNIST (a-b) and FashionMNIST (c-d) in place of CIFAR10. Stability monotonically decreases with P. The triangles give a reference for the predicted slope in the stripe model – i.e. $R_f \sim P^{-1}$ – see Section 6. The slopes in case of ResNets are compatible with the prediction. For EfficientNets, the second panel of Fig.3 suggests that stability to diffeomorphisms is less important. Here, we also see that it builds up more slowly when increasing the training set size. Finally, blue and red colors indicate different interpolation methods used for generating image deformations, as discussed in Appendix C. Results are not affected by this choice.

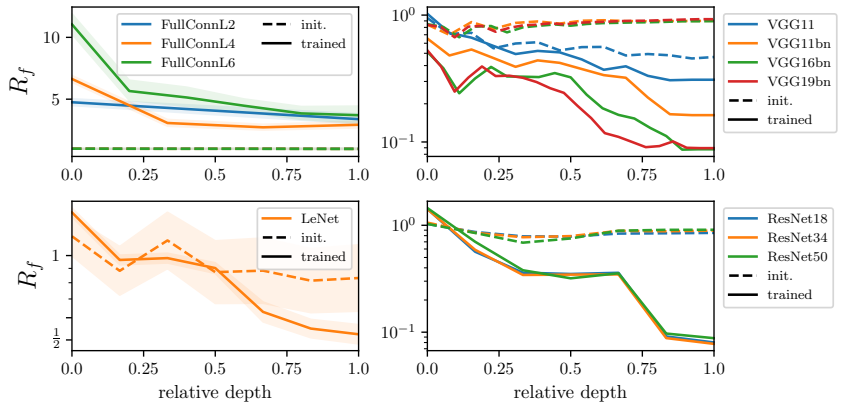


Figure 14: **Relative stability to diffeomorphisms as a function of depth.** R_f as a function of the layers relative depth (i.e. $\frac{\text{current layer depth}}{\text{total depth}}$) where "0" identifies the output of the 1st layer and "1" the last. The relative stability is measured for the output of layers (or blocks of layers) inside the nets for simple architectures (1st column) and deep ones (2nd column) at initialization (dashed) and after training (full lines). All nets are trained on the full CIFAR10 dataset. $R_{f_0} \approx 1$ independently of depth at initialization while it decreases monotonically as a function of depth after training. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure R_f. The results are obtained by log-averaging over single realizations.

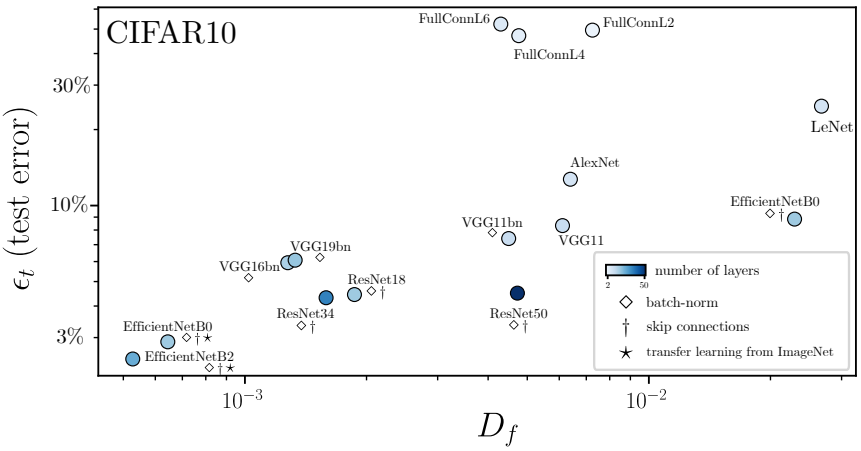


Figure 15: **Test error ϵ_t *vs.* stability to diffeomorphisms** D_f for common architectures when trained on the full 10-classes CIFAR10 dataset ($P = 50k$) with SGD and the cross-entropy loss; the EfficientNets achieving the best performance are trained by transfer learning from ImageNet ($\star$) – more details on the training procedures can be found in Appendix E.1. The color scale indicates depth, and the symbols the presence of batch-norm ($\diamond$) and skip connections ($\dagger$). D_f correlation with ϵ_t (corr. coeff.: 0.62) is much smaller than the one measured for R_f – see Fig.3. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure D_f. The results are obtained by log-averaging over single realizations.

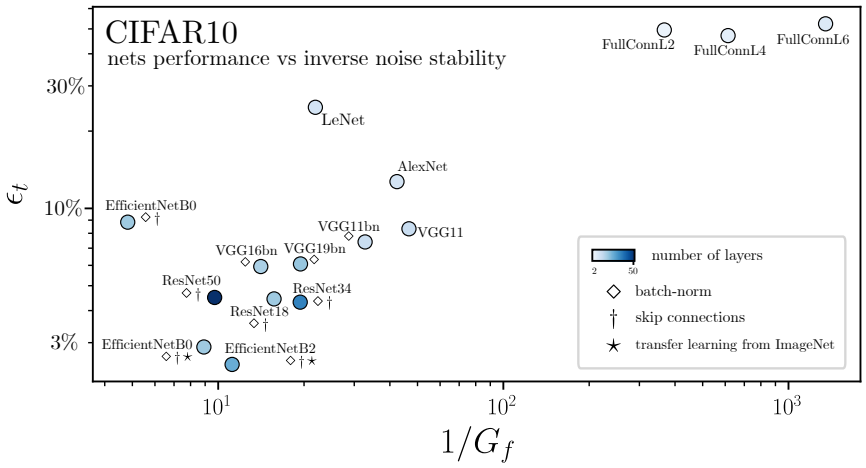


Figure 16: **Test error ϵ_t *vs.* inverse of stability to noise** $1/G_f$ for common architectures when trained on the full 10-classes CIFAR10 dataset ($P = 50k$) with SGD and the cross-entropy loss; the EfficientNets achieving the best performance are trained by transfer learning from ImageNet ($\star$) – more details on the training procedures can be found in Appendix E.1. The color scale indicates depth, and the symbols the presence of batch-norm ($\diamond$) and skip connections (†). G_f correlation with ϵ_t (corr. coeff.: 0.85) is less important than the one measured for R_f – see Fig.3. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure G_f. The results are obtained by log-averaging over single realizations.

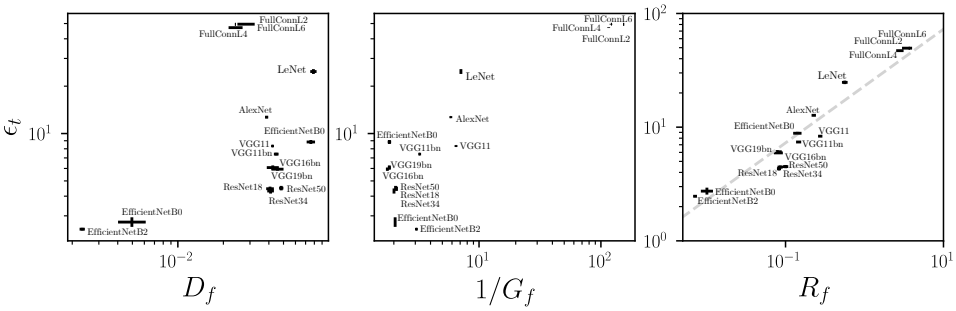


Figure 17: **Test error ϵ_t *vs.* D_f, $1/G_f$ and R_f where $\langle\cdot\rangle$ is the mean.** Analogous to Figs15-19, we use here the mean instead of the median to compute averages over samples and transformations.

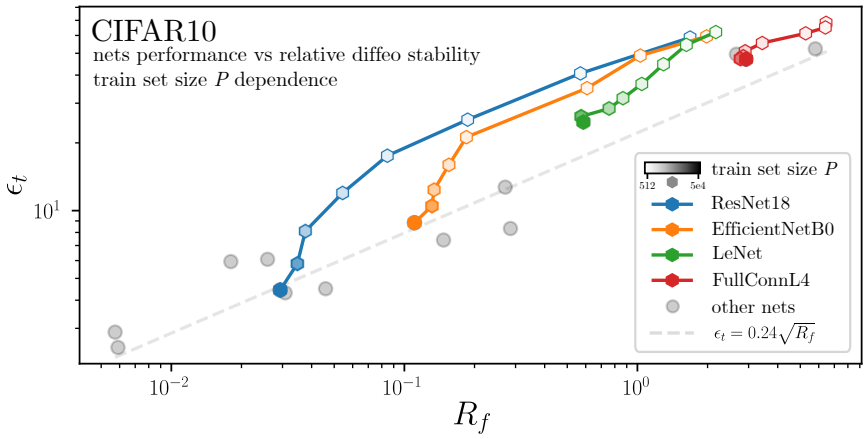


Figure 18: **Test error ϵ_t *vs.* relative stability to diffeomorphisms R_f for different training set sizes P.** Same data as Fig.5, we report here curves corresponding to training on different set sizes for 4 architectures. The other architectures considered together with the power-law fit are left in background. For a small training set, CNNs behave similarly. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure R_f. The results are obtained by log-averaging over single realizations.

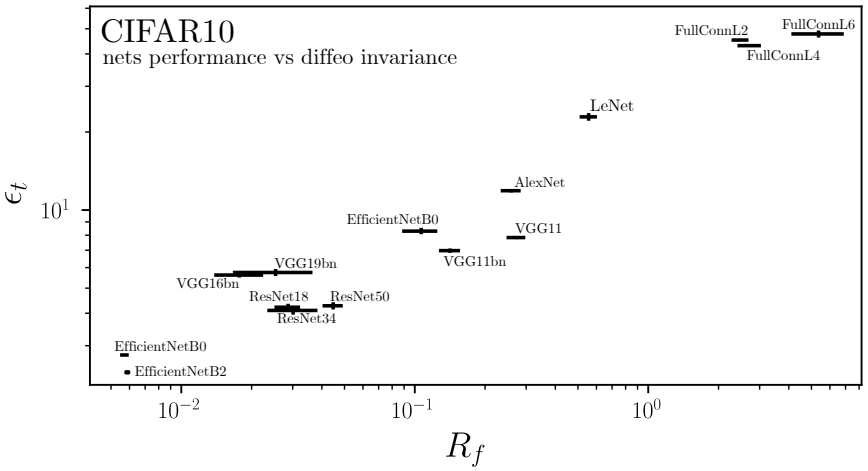


Figure 19: **Test error ϵ_t *vs.* relative stability to diffeomorphisms R_f with error estimates.** Same data as Fig.5, we report error bars here. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure R_f. The results are obtained by log-averaging over single realizations.

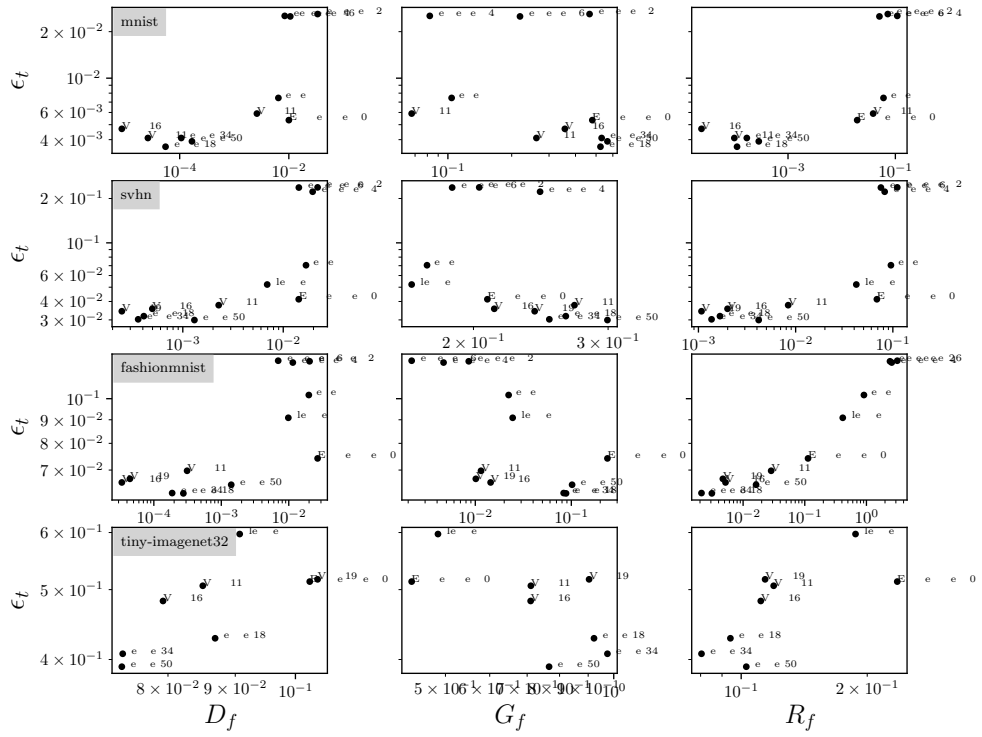


Figure 20: **Test error ϵ_t *vs.* D_f, G_f and R_f (on the columns) for different data sets (on the rows).** The corresponding correlation coefficients are shown in Table 2. Lines 1-2: MNIST and SVHN both contain images of digits and show a similar $\epsilon_t(R_f)$. Line 3: FashionMNIST results are comparable to the CIFAR10 ones shown in the main text. Line 4: Tiny ImageNet32 is a re-scaled (32x32 pixels) version of ImageNet with 200 classes and 100'000 training points. The task is harder than the other data sets and is such that we could not train simple networks (FC, LeNet) on it – i.e. the loss stays $\mathcal{O}(1)$ throughout training – so these are not reported here.

Table 2: **Test error vs. stability: correlation coefficients for different data sets.**

data-set	D_f	G_f	R_f
MNIST	0.71	-0.43	0.75
SVHN	0.87	-0.28	0.81
FashionMNIST	0.72	-0.68	0.94
Tiny ImageNet	0.69	-0.66	0.74

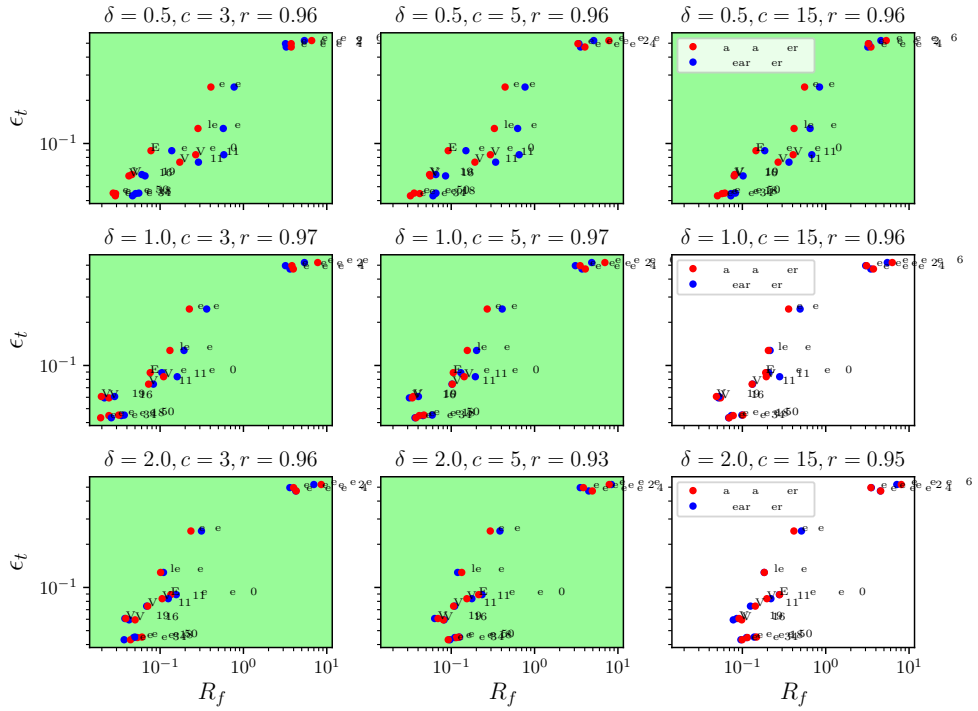


Figure 21: **Test error ϵ_t *vs.* D_f, G_f and R_f for CIFAR10 and varying δ and cut-off c.** Titles report the values of the varying parameters together with corr. coeffs. Parameters corresponding to allowed diffeo are indicated by the green background. Red and blue colors correspond to different interpolation methods. Overall, results are robust when varying these parameters.

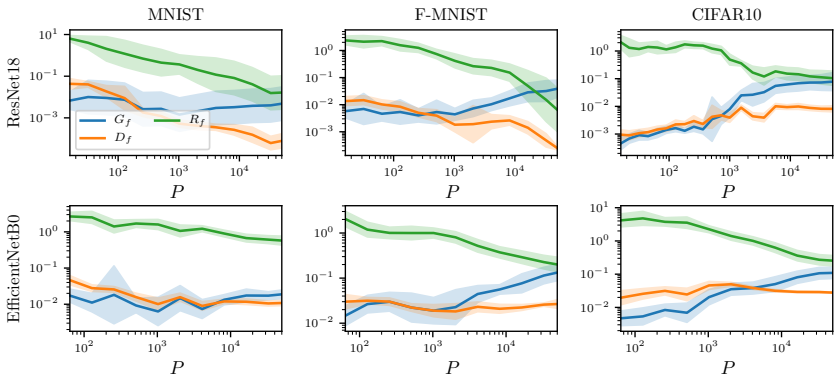


Figure 22: **Stability toward Gaussian noise (G_f) and diffeomorphisms (D_f) alone, and the relative stability R_f with the relative errors.** Analogous to Fig.6 in which error estimates are omitted to favour clarity. Here we fix the cut-off to $c = 3$ and show error estimates instead. Columns correspond to different data-sets (MNIST, FashionMNIST and CIFAR10) and rows to architectures (ResNet18 and EfficientNetB0). Each panel reports G_f (blue), D_f (orange) and R_f (green) as a function of P and for different cut-off values c, as indicated in the legend. *Statistics*: Each point in the graphs is obtained by training 16 differently initialized networks on 16 different subsets of the data-sets; each network is then probed with 500 test samples in order to measure stability to diffeomorphisms and Gaussian noise. The resulting R_f is obtained by log-averaging the results from single realizations. As we are plotting quantities in log scale, we report the relative error (shaded).

5 Equivariant Architectures

5.1 Spherical CNN

The following paper is the preprint version of Cohen et al. (2018).

Candidate contributions After developing an equivariant 2D CNN for the dihedral group (90 degree rotation and mirrors) during his master thesis, the candidate started a collaboration with Taco Cohen. Taco Cohen wanted to develop Spherical CNN as a proof of concept for neural networks equivariant with respect to a continuous group (a Lie group). The candidate contributed to this paper by implementing most of the code and executing the numerical experiment of recognition of the 3d shapes.

Published as a conference paper at ICLR 2018

SPHERICAL CNNS

Taco S. Cohen*
University of Amsterdam

Mario Geiger*
EPFL

Jonas Köhler*
University of Amsterdam

Max Welling
University of Amsterdam & CIFAR

ABSTRACT

Convolutional Neural Networks (CNNs) have become the method of choice for learning problems involving 2D planar images. However, a number of problems of recent interest have created a demand for models that can analyze spherical images. Examples include omnidirectional vision for drones, robots, and autonomous cars, molecular regression problems, and global weather and climate modelling. A naive application of convolutional networks to a planar projection of the spherical signal is destined to fail, because the space-varying distortions introduced by such a projection will make translational weight sharing ineffective.

In this paper we introduce the building blocks for constructing spherical CNNs. We propose a definition for the spherical cross-correlation that is both expressive and rotation-equivariant. The spherical correlation satisfies a generalized Fourier theorem, which allows us to compute it efficiently using a generalized (non-commutative) Fast Fourier Transform (FFT) algorithm. We demonstrate the computational efficiency, numerical accuracy, and effectiveness of spherical CNNs applied to 3D model recognition and atomization energy regression.

1 INTRODUCTION

Convolutional networks are able to detect local patterns regardless of their position in the image. Like patterns in a planar image, patterns on the sphere can move around, but in this case the "move" is a 3D rotation instead of a translation. In analogy to the planar CNN, we would like to build a network that can detect patterns regardless of how they are rotated over the sphere.

As shown in Figure 1, there is no good way to use translational convolution or cross-correlation[1] to analyze spherical signals. The most obvious approach, then, is to change the definition of cross-correlation by replacing filter translations by rotations. Doing so, we run into a subtle but important difference between the plane and the sphere: whereas the space of moves for the plane (2D translations) is itself isomorphic to the plane, the space of moves for the sphere (3D rotations) is a different, *three-dimensional* manifold called $\mathrm{SO}(3)$[2]. It follows that the result of a spherical correlation (the output feature map) is to be considered a signal on $\mathrm{SO}(3)$, not a signal on the sphere, S^2. For this reason, we deploy $\mathrm{SO}(3)$ group correlation in the higher layers of a spherical CNN (Cohen and Welling, 2016).

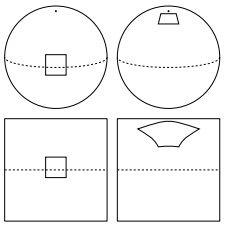


Figure 1: Any planar projection of a spherical signal will result in distortions. Rotation of a spherical signal cannot be emulated by translation of its planar projection.

*Equal contribution.

[1]Despite the name, CNNs typically use cross-correlation instead of convolution in the forward pass. In this paper we will generally use the term cross-correlation, or correlation for short.

[2]To be more precise: although the symmetry group of the plane contains more than just translations, the translations form a subgroup that acts on the plane. In the case of the sphere there is no coherent way to define a composition for points on the sphere, and so the sphere cannot act on itself (it is not a group). For this reason, we must consider the whole of $\mathrm{SO}(3)$.

The implementation of a spherical CNN (S^2-CNN) involves two major challenges. Whereas a square grid of pixels has discrete translation symmetries, no perfectly symmetrical grids for the sphere exist. This means that there is no simple way to define the rotation of a spherical filter by one pixel. Instead, in order to rotate a filter we would need to perform some kind of interpolation. The other challenge is computational efficiency; $\mathrm{SO}(3)$ is a three-dimensional manifold, so a naive implementation of $\mathrm{SO}(3)$ correlation is $O(n^6)$.

We address both of these problems using techniques from non-commutative harmonic analysis (Chirikjian and Kyatkin, 2001; Folland, 1995). This field presents us with a far-reaching generalization of the Fourier transform, which is applicable to signals on the sphere as well as the rotation group. It is known that the $\mathrm{SO}(3)$ correlation satisfies a Fourier theorem with respect to the $\mathrm{SO}(3)$ Fourier transform, and the same is true for our definition of S^2 correlation. Hence, the S^2 and $\mathrm{SO}(3)$ correlation can be implemented efficiently using generalized FFT algorithms.

Because we are the first to use cross-correlation on a continuous group inside a multi-layer neural network, we rigorously evaluate the degree to which the mathematical properties predicted by the continuous theory hold in practice for our discretized implementation.

Furthermore, we demonstrate the utility of spherical CNNs for rotation invariant classification and regression problems by experiments on three datasets. First, we show that spherical CNNs are much better at rotation invariant classification of Spherical MNIST images than planar CNNs. Second, we use the CNN for classifying 3D shapes. In a third experiment we use the model for molecular energy regression, an important problem in computational chemistry.

CONTRIBUTIONS

The main contributions of this work are the following:

1. The theory of spherical CNNs.
2. The first automatically differentiable implementation of the generalized Fourier transform for S^2 and $\mathrm{SO}(3)$. Our PyTorch code is easy to use, fast, and memory efficient.
3. The first empirical support for the utility of spherical CNNs for rotation-invariant learning problems.

2 RELATED WORK

It is well understood that the power of CNNs stems in large part from their ability to exploit (translational) symmetries though a combination of weight sharing and translation equivariance. It thus becomes natural to consider generalizations that exploit larger groups of symmetries, and indeed this has been the subject of several recent papers by Gens and Domingos (2014); Olah (2014); Dieleman et al. (2015; 2016); Cohen and Welling (2016); Ravanbakhsh et al. (2017); Zaheer et al. (2017b); Guttenberg et al. (2016); Cohen and Welling (2017). With the exception of $\mathrm{SO}(2)$-steerable networks (Worrall et al., 2017; Weiler et al., 2017), these networks are all limited to discrete groups, such as discrete rotations acting on planar images or permutations acting on point clouds. Other very recent work is concerned with the analysis of spherical images, but does not define an equivariant architecture (Su and Grauman, 2017; Boomsma and Frellsen, 2017). Our work is the first to achieve equivariance to a continuous, non-commutative group ($\mathrm{SO}(3)$), and the first to use the generalized Fourier transform for fast group correlation. A preliminary version of this work appeared as Cohen et al. (2017).

To efficiently perform cross-correlations on the sphere and rotation group, we use generalized FFT algorithms. Generalized Fourier analysis, sometimes called abstract- or noncommutative harmonic analysis, has a long history in mathematics and many books have been written on the subject (Sugiura, 1990; Taylor, 1986; Folland, 1995). For a good engineering-oriented treatment which covers generalized FFT algorithms, see (Chirikjian and Kyatkin, 2001). Other important works include (Driscoll and Healy, 1994; Healy et al., 2003; Potts et al., 1998; Kunis and Potts, 2003; Drake et al., 2008; Maslen, 1998; Rockmore, 2004; Kostelec and Rockmore, 2007; 2008; Potts et al., 2009; Makadia et al., 2007; Gutman et al., 2008).

3 CORRELATION ON THE SPHERE AND ROTATION GROUP

We will explain the S^2 and $\mathrm{SO}(3)$ correlation by analogy to the classical planar $\mathbb{Z}^2$ correlation. The planar correlation can be understood as follows:

> The value of the output feature map at translation $x \in \mathbb{Z}^2$ is computed as an inner product between the input feature map and a filter, shifted by x.

Similarly, the spherical correlation can be understood as follows:

> The value of the output feature map evaluated at rotation $R \in \mathrm{SO}(3)$ is computed as an inner product between the input feature map and a filter, rotated by R.

Because the output feature map is indexed by a rotation, it is modelled as a function on $\mathrm{SO}(3)$. We will discuss this issue in more detail shortly.

The above definition refers to various concepts that we have not yet defined mathematically. In what follows, we will go through the required concepts one by one and provide a precise definition. Our goal for this section is only to present a mathematical model of spherical CNNs. Generalized Fourier theory and implementation details will be treated later.

The Unit Sphere S^2 can be defined as the set of points $x \in \mathbb{R}^3$ with norm 1. It is a two-dimensional manifold, which can be parameterized by spherical coordinates $\alpha \in [0, 2\pi]$ and $\beta \in [0, \pi]$.

Spherical Signals We model spherical images and filters as continuous functions $f : S^2 \to \mathbb{R}^K$, where K is the number of channels.

Rotations The set of rotations in three dimensions is called $\mathrm{SO}(3)$, the "special orthogonal group". Rotations can be represented by 3×3 matrices that preserve distance (i.e. $||Rx|| = ||x||$) and orientation ($\det(R) = +1$). If we represent points on the sphere as 3D unit vectors x, we can perform a rotation using the matrix-vector product Rx. The rotation group $\mathrm{SO}(3)$ is a three-dimensional manifold, and can be parameterized by ZYZ-Euler angles $\alpha \in [0, 2\pi]$, $\beta \in [0, \pi]$, and $\gamma \in [0, 2\pi]$.

Rotation of Spherical Signals In order to define the spherical correlation, we need to know not only how to rotate points $x \in S^2$ but also how to rotate filters (i.e. functions) on the sphere. To this end, we introduce the rotation operator L_R that takes a function f and produces a rotated function $L_R f$ by composing f with the rotation R^{-1}:

$$[L_R f](x) = f(R^{-1}x). \tag{1}$$

Due to the inverse on R, we have $L_{RR'} = L_R L_{R'}$.

Inner products The inner product on the vector space of spherical signals is defined as:

$$\langle \psi, f \rangle = \int_{S^2} \sum_{k=1}^{K} \psi_k(x) f_k(x) dx, \tag{2}$$

The integration measure dx denotes the standard rotation invariant integration measure on the sphere, which can be expressed as $d\alpha \sin(\beta) d\beta / 4\pi$ in spherical coordinates (see Appendix A). The invariance of the measure ensures that $\int_{S^2} f(Rx)dx = \int_{S^2} f(x)dx$, for any rotation $R \in \mathrm{SO}(3)$. That is, the volume under a spherical heightmap does not change when rotated. Using this fact, we can show that $L_{R^{-1}}$ is adjoint to L_R, which implies that L_R is unitary:

$$\begin{aligned} \langle L_R \psi, f \rangle &= \int_{S^2} \sum_{k=1}^{K} \psi_k(R^{-1}x) f_k(x) dx \\ &= \int_{S^2} \sum_{k=1}^{K} \psi_k(x) f_k(Rx) dx \\ &= \langle \psi, L_{R^{-1}} f \rangle. \end{aligned} \tag{3}$$

Spherical Correlation With these ingredients in place, we are now ready to state mathematically what was stated in words before. For spherical signals f and ψ, we define the correlation as:

$$[\psi \star f](R) = \langle L_R \psi, f \rangle = \int_{S^2} \sum_{k=1}^{K} \psi_k(R^{-1}x) f_k(x) dx. \tag{4}$$

As mentioned before, the output of the spherical correlation is a function on $\mathrm{SO}(3)$. This is perhaps somewhat counterintuitive, and indeed the conventional definition of spherical convolution gives as output a function on the sphere. However, as shown in Appendix B, the conventional definition effectively restricts the filter to be circularly symmetric about the Z axis, which would greatly limit the expressive capacity of the network.

Rotation of $\mathrm{SO}(3)$ Signals We defined the rotation operator L_R for spherical signals (eq. 1), and used it to define spherical cross-correlation (eq. 4). To define the $\mathrm{SO}(3)$ correlation, we need to generalize the rotation operator so that it can act on signals defined on $\mathrm{SO}(3)$. As we will show, naively reusing eq. 1 is the way to go. That is, for $f : \mathrm{SO}(3) \to \mathbb{R}^K$, and $R, Q \in \mathrm{SO}(3)$:

$$[L_R f](Q) = f(R^{-1}Q). \tag{5}$$

Note that while the argument $R^{-1}x$ in Eq. 1 denotes the rotation of $x \in S^2$ by $R^{-1} \in \mathrm{SO}(3)$, the analogous term $R^{-1}Q$ in Eq. 5 denotes to the composition of rotations (i.e. matrix multiplication).

Rotation Group Correlation Using the same analogy as before, we can define the correlation of two signals on the rotation group, $f, \psi : \mathrm{SO}(3) \to \mathbb{R}^K$, as follows:

$$[\psi \star f](R) = \langle L_R \psi, f \rangle = \int_{\mathrm{SO}(3)} \sum_{k=1}^{K} \psi_k(R^{-1}Q) f_k(Q) dQ. \tag{6}$$

The integration measure dQ is the invariant measure on $\mathrm{SO}(3)$, which may be expressed in ZYZ-Euler angles as $d\alpha \sin(\beta) d\beta d\gamma / (8\pi^2)$ (see Appendix A).

Equivariance As we have seen, correlation is defined in terms of the rotation operator L_R. This operator acts naturally on the input space of the network, but what justification do we have for using it in the second layer and beyond?

The justification is provided by an important property, shared by all kinds of convolution and correlation, called equivariance. A layer Φ is equivariant if $\Phi \circ L_R = T_R \circ \Phi$, for some operator T_R. Using the definition of correlation and the unitarity of L_R, showing equivariance is a one liner:

$$[\psi \star [L_Q f]](R) = \langle L_R \psi, L_Q f \rangle = \langle L_{Q^{-1}R} \psi, f \rangle = [\psi \star f](Q^{-1}R) = [L_Q [\psi \star f]](R). \tag{7}$$

The derivation is valid for spherical correlation as well as rotation group correlation.

4 Fast Spherical Correlation with G-FFT

It is well known that correlations and convolutions can be computed efficiently using the Fast Fourier Transform (FFT). This is a result of the Fourier theorem, which states that $\widehat{f * \psi} = \hat{f} \cdot \hat{\psi}$. Since the FFT can be computed in $O(n \log n)$ time and the product $\cdot$ has linear complexity, implementing the correlation using FFTs is asymptotically faster than the naive $O(n^2)$ spatial implementation.

For functions on the sphere and rotation group, there is an analogous transform, which we will refer to as the generalized Fourier transform (GFT) and a corresponding fast algorithm (GFFT). This transform finds it roots in the representation theory of groups, but due to space constraints we will not go into details here and instead refer the interested reader to Sugiura (1990) and Folland (1995).

Conceptually, the GFT is nothing more than the linear projection of a function onto a set of orthogonal basis functions called "matrix element of irreducible unitary representations". For the circle (S^1) or line ($\mathbb{R}$), these are the familiar complex exponentials $\exp(in\theta)$. For $\mathrm{SO}(3)$, we have the Wigner D-functions $D^l_{mn}(R)$ indexed by $l \geq 0$ and $-l \leq m, n \leq l$. For S^2, these are the spherical harmonics[3] $Y^l_m(x)$ indexed by $l \geq 0$ and $-l \leq m \leq l$.

Denoting the manifold (S^2 or $\mathrm{SO}(3)$) by X and the corresponding basis functions by U^l (which is either vector-valued (Y^l) or matrix-valued (D^l)), we can write the GFT of a function $f : X \to \mathbb{R}$ as

$$\hat{f}^l = \int_X f(x) \overline{U^l(x)} dx. \tag{8}$$

[3]Technically, S^2 is not a group and therefore does not have irreducible representations, but it is a quotient of groups $\mathrm{SO}(3)/\mathrm{SO}(2)$ and we have the relation $Y^l_m = D^l_{m0}|_{S^2}$

This integral can be computed efficiently using a GFFT algorithm (see Section 4.1).

The inverse $\mathrm{SO}(3)$ Fourier transform is defined as:

$$f(R) = \sum_{l=0}^{b} (2l+1) \sum_{m=-l}^{l} \sum_{n=-l}^{l} \hat{f}^l_{mn} D^l_{mn}(R), \tag{9}$$

and similarly for S^2. The maximum frequency b is known as the bandwidth, and is related to the resolution of the spatial grid (Kostelec and Rockmore, 2007).

Using the well-known (in fact, defining) property of the Wigner D-functions that $D^l(R)D^l(R') = D^l(RR')$ and $D^l(R^{-1}) = D^l(R)^\dagger$, it can be shown (see Appendix D) that the $\mathrm{SO}(3)$ correlation satisfies a Fourier theorem[4]: $\widehat{\psi \star f} = \hat{f} \cdot \hat{\psi}^\dagger$, where $\cdot$ denotes matrix multiplication of the two block matrices $\hat{f}$ and $\hat{\psi}^\dagger$.

Similarly, using $Y(Rx) = D(R)Y(x)$ and $Y^l_m = D^l_{m0}|_{S^2}$, one can derive an analogous S^2 convolution theorem: $\widehat{\psi \star f}^l = \hat{f}^l \cdot \hat{\psi}^{l\dagger}$, where $\hat{f}^l$ and $\hat{\psi}^l$ are now vectors. This says that the $\mathrm{SO}(3)$-FT of the S^2 correlation of two spherical signals can be computed by taking the outer product of the S^2-FTs of the signals. This is shown in figure 2.

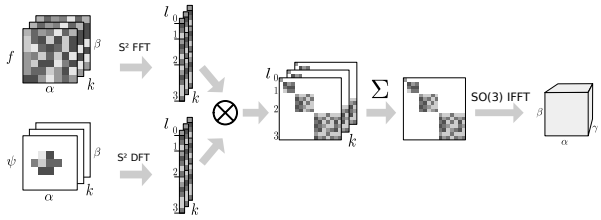


Figure 2: Spherical correlation in the spectrum. The signal f and the locally-supported filter ψ are Fourier transformed, block-wise tensored, summed over input channels, and finally inverse transformed. Note that because the filter is locally supported, it is faster to use a matrix multiplication (DFT) than an FFT algorithm for it. We parameterize the sphere using spherical coordinates α, β, and $\mathrm{SO}(3)$ with ZYZ-Euler angles α, β, γ.

4.1 Implementation of G-FFT and Spectral G-Conv

Here we sketch the implementation of GFFTs. For details, see (Kostelec and Rockmore, 2007).

The input of the $\mathrm{SO}(3)$ FFT is a spatial signal f on $\mathrm{SO}(3)$, sampled on a discrete grid and stored as a 3D array. The axes correspond to the ZYZ-Euler angles α, β, γ. The first step of the $\mathrm{SO}(3)$-FFT is to perform a standard 2D translational FFT over the α and γ axes. The FFT'ed axes correspond to the m, n axes of the result. The second and last step is a linear contraction of the β axis of the FFT'ed array with a precomputed array of samples from the Wigner-d (small-d) functions $d^l_{mn}(\beta)$. Because the shape of d^l depends on l (it is $(2l+1) \times (2l+1)$), this linear contraction is implemented as a custom GPU kernel. The output is a set of Fourier coefficients $\hat{f}^l_{mn}$ for $l \geq n, m \geq -l$ and $l = 0, \ldots, L_{\max}$.

The algorithm for the S^2-FFTs is very similar, only in this case we FFT over the α axis only, and do a linear contraction with precomputed Legendre functions over the β axis.

Our code is available at `https://github.com/jonas-koehler/s2cnn`.

5 Experiments

In a first sequence of experiments, we evaluate the numerical stability and accuracy of our algorithm. In a second sequence of experiments, we showcase that the new cross-correlation layers we have

[4]This result is valid for real functions. For complex functions, conjugate ψ on the left hand side.

introduced are indeed useful building blocks for several real problems involving spherical signals. Our examples for this are recognition of 3D shapes and predicting the atomization energy of molecules.

5.1 EQUIVARIANCE ERROR

In this paper we have presented the first instance of a group equivariant CNN for a continuous, non-commutative group. In the discrete case, one can prove that the network is exactly equivariant, but although we can prove $[L_R f] * \psi = L_R[f * \psi]$ for continuous functions f and ψ on the sphere or rotation group, this is not exactly true for the discretized version that we actually compute. Hence, it is reasonable to ask if there are any significant discretization artifacts and whether they affect the equivariance properties of the network. If equivariance can not be maintained for many layers, one may expect the weight sharing scheme to become much less effective.

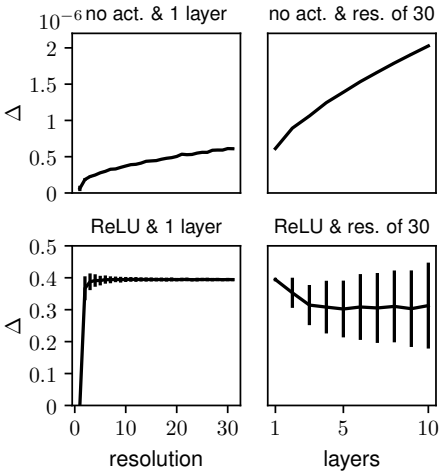


Figure 3: Δ as a function of the resolution and the number of layers.

We first tested the equivariance of the $\mathrm{SO}(3)$ correlation at various resolutions b. We do this by first sampling $n = 500$ random rotations R_i as well as n feature maps f_i with $K = 10$ channels. Then we compute $\Delta = \frac{1}{n}\sum_{i=1}^{n} \mathrm{std}(L_{R_i}\Phi(f_i) - \Phi(L_{R_i} f_i))/\,\mathrm{std}(\Phi(f_i))$, where Φ is a composition of $\mathrm{SO}(3)$ correlation layers with randomly initialized filters. In case of perfect equivariance, we expect this quantity to be zero. The results (figure 3 (top)), show that although the approximation error Δ grows with the resolution and the number of layers, it stays manageable for the range of resolutions of interest.

We repeat the experiment with ReLU activation function after each correlation operation. As shown in figure 3 (bottom), the error is higher but stays flat. This indicates that the error is not due to the network layers, but due to the feature map rotation, which is exact only for bandlimited functions.

5.2 ROTATED MNIST ON THE SPHERE

In this experiment we evaluate the generalization performance with respect to rotations of the input. For testing we propose a version MNIST dataset projected on the sphere (see fig. 4). We created two instances of this dataset: one in which each digit is projected on the northern hemisphere and one in which each projected digit is additionally randomly rotated.

Architecture and Hyperparameters As a baseline model, we use a simple CNN with layers conv-ReLU-conv-ReLU-FC-softmax, with filters of size 5×5, $k = 32, 64, 10$ channels, and stride 3 in both layers ($\approx$ 68K parameters). We compare to a spherical CNN with layers S^2conv-ReLU-$\mathrm{SO}(3)$conv-ReLU-FC-softmax, bandwidth $b = 30, 10, 6$ and $k = 20, 40, 10$ channels ($\approx$ 58K parameters).

Figure 4: Two MNIST digits projected onto the sphere using stereographic projection. Mapping back to the plane results in non-linear distortions.

Results We trained each model on the non-rotated (NR) and the rotated (R) training set and evaluated it on the non-rotated and rotated test set. See table 1. While the planar CNN achieves high accuracy in the NR / NR regime, its performance in the R / R regime is much worse, while the spherical CNN is unaffected. When trained on the

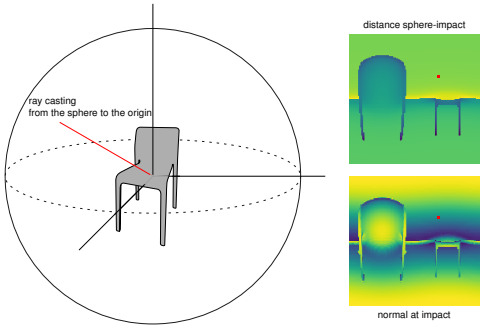


Figure 5: The ray is cast from the surface of the sphere towards the origin. The first intersection with the model gives the values of the signal. The two images of the right represent two spherical signals in (α, β) coordinates. They contain respectively the distance from the sphere and the cosine of the ray with the normal of the model. The red dot corresponds to the pixel set by the red line.

non-rotated dataset and evaluated on the rotated dataset (NR / R), the planar CNN does no better than random chance. The spherical CNN shows a slight decrease in performance compared to R/R, but still performs very well.

	NR / NR	R / R	NR / R
planar	0.98	0.23	0.11
spherical	0.96	0.95	0.94

Table 1: Test accuracy for the networks evaluated on the spherical MNIST dataset. Here R = rotated, NR = non-rotated and X / Y denotes, that the network was trained on X and evaluated on Y.

5.3 Recognition of 3D Shapes

Next, we applied S^2CNN to 3D shape classification. The SHREC17 task (Savva et al., 2017) contains 51300 3D models taken from the ShapeNet dataset (Chang et al., 2015) which have to be classified into 55 common categories (tables, airplanes, persons, etc.). There is a consistently aligned regular dataset and a version in which all models are randomly perturbed by rotations. We concentrate on the latter to test the quality of our rotation equivariant representations learned by S^2CNN.

Representation We project the 3D meshes onto an enclosing sphere using a straightforward ray casting scheme (see Fig. 5). For each point on the sphere we send a ray towards the origin and collect 3 types of information from the intersection: ray length and $\cos$ / $\sin$ of the surface angle. We further augment this information with ray casting information for the convex hull of the model, which in total gives us 6 channels for the signal. This signal is discretized using a Driscoll-Healy grid (Driscoll and Healy, 1994) with bandwidth $b = 128$. Ignoring non-convexity of surfaces we assume this projection captures enough information of the shape to be useful for the recognition task.

Architecture and Hyperparameters Our network consists of an initial S^2conv-BN-ReLU block followed by two $\mathrm{SO}(3)$conv-BN-ReLU blocks. The resulting filters are pooled using a max pooling layer followed by a last batch normalization and then fed into a linear layer for the final classification. It is important to note that the the max pooling happens over the group SO(3): if f_k is the k-th filter in the final layer (a function on SO(3)) the result of the pooling is $\max_{x \in SO(3)} f_k(x)$. We used 50, 70, and 350 features for the S^2 and the two SO(3) layers, respectively. Further, in each layer we reduce the resolution b, from 128, 32, 22 to 7 in the final layer. Each filter kernel ψ on SO(3) has non-local support, where $\psi(\alpha, \beta, \gamma) \neq 0$ iff $\beta = \frac{\pi}{2}$ and $\gamma = 0$ and the number of points of the discretization is proportional to the bandwidth in each layer. The final network contains ≈ 1.4M parameters, takes 8GB of memory at batch size 16, and takes 50 hours to train.

Method	P@N	R@N	F1@N	mAP	NDCG
Tatsuma_ReVGG	0.705	0.769	0.719	0.696	0.783
Furuya_DLAN	0.814	0.683	0.706	0.656	0.754
SHREC16-Bai_GIFT	0.678	0.667	0.661	0.607	0.735
Deng_CM-VGG5-6DB	0.412	0.706	0.472	0.524	0.624
Ours	0.701 (3rd)	0.711 (2nd)	0.699 (3rd)	0.676 (2nd)	0.756 (2nd)

Table 2: Results and best competing methods for the SHREC17 competition.

Results We evaluated our trained model using the official metrics and compared to the top three competitors in each category (see table 2 for results). Except for precision and F1@N, in which our model ranks third, it is the runner up on each other metric. The main competitors, Tatsuma_ReVGG and Furuya_DLAN use input representations and network architectures that are highly specialized to the SHREC17 task. Given the rather task agnostic architecture of our model and the lossy input representation we use, we interpret our models performance as strong empirical support for the effectiveness of Spherical CNNs.

5.4 Prediction of Atomization Energies from Molecular Geometry

Finally, we apply S^2CNN on molecular energy regression. In the QM7 task (Blum and Reymond, 2009; Rupp et al., 2012) the atomization energy of molecules has to be predicted from geometry and charges. Molecules contain up to $N = 23$ atoms of $T = 5$ types (H, C, N, O, S). They are given as a list of positions p_i and charges z_i for each atom i.

Representation by Coulomb matrices Rupp et al. (2012) propose a rotation and translation invariant representation of molecules by defining the *Coulomb matrix* $C \in \mathbb{R}^{N\times N}$ (CM). For each pair of atoms $i \neq j$ they set $C_{ij} = (z_i z_j)/(|p_i - p_j|)$ and $C_{ii} = 0.5 z_i^{2.4}$. Diagonal elements encode the atomic energy by nuclear charge, while other elements encode Coulomb repulsion between atoms. This representation is not permutation invariant. To this end Rupp et al. (2012) propose a distance measure between Coulomb matrices used within Gaussian kernels whereas Montavon et al. (2012) propose sorting C or random sampling index permutations.

Representation as a spherical signal We utilize spherical symmetries in the geometry by defining a sphere S_i around around p_i for each atom i. The radius is kept uniform across atoms and molecules and chosen minimal such that no intersections among spheres in the training set happen. Generalizing the Coulomb matrix approach we define for each possible z and for each point x on S_i potential functions $U_z(x) = \sum_{j\neq i, z_j = z} \frac{z_i \cdot z}{|x-p_i|}$ producing a T channel spherical signal for each atom in the molecule (see figure 6). This representation is invariant with respect to translations and equivariant with respect to rotations. However, it is still not permutation invariant. The signal is discretized using a Driscoll-Healy (Driscoll and Healy, 1994) grid with bandwidth $b = 10$ representing the molecule as a sparse $N \times T \times 2b \times 2b$ tensor.

Architecture and Hyperparameters We use a deep ResNet style S^2CNN. Each ResNet block is made of S^2/SO(3)conv-BN-ReLU-SO(3)conv-BN after which the input is added to the result. We share weights among atoms making filters permutation invariant, by pushing the atom dimension into the batch dimension. In each layer we downsample the bandwidth, while increasing the number of features F. After integrating the signal over SO(3) each molecule becomes a $N \times F$ tensor. For permutation invariance over atoms we follow Zaheer et al. (2017a) and embed each resulting feature vector of an atom into a latent space using a MLP ϕ. Then we sum these latent representations over the atom dimension and get our final regression value for the molecule by mapping with another MLP ψ. Both ϕ and ψ are jointly optimized. Training a simple MLP only on the 5 frequencies of atom types in a molecule already gives a RMSE of ~ 19. Thus, we train the S^2CNN on the residual only, which improved convergence speed and stability over direct training. The final architecture is sketched in table 3. It has about 1.4M parameters, consumes 7GB of memory at batch size 20, and takes 3 hours to train.

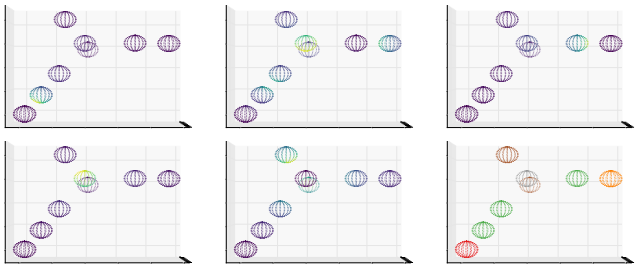


Figure 6: The five potential channels U_z with $z \in \{1, 6, 7, 8, 16\}$ for a molecule containing atoms H (red), C (green), N (orange), O (brown), S (gray).

Method	Author	RMSE
MLP / random CM	(a)	5.96
LGIKA(RF)	(b)	10.82
RBF kernels / random CM	(a)	11.40
RBF kernels / sorted CM	(a)	12.59
MLP / sorted CM	(a)	16.06
Ours		**8.47**

S^2CNN	Layer	Bandwidth	Features
	Input		5
	ResBlock	10	20
	ResBlock	8	40
	ResBlock	6	60
	ResBlock	4	80
	ResBlock	2	160
DeepSet	Layer	Input/Hidden	
	ϕ (MLP)	160/150	
	ψ (MLP)	100/50	

Table 3: Left: Experiment results for the QM7 task: (a) Montavon et al. (2012) (b) Raj et al. (2016). Right: ResNet architecture for the molecule task.

Results We evaluate by RMSE and compare our results to Montavon et al. (2012) and Raj et al. (2016) (see table 3). Our learned representation outperforms all kernel-based approaches and a MLP trained on sorted Coulomb matrices. Superior performance could only be achieved for an MLP trained on randomly permuted Coulomb matrices. However, sufficient sampling of random permutations grows exponentially with N, so this method is unlikely to scale to large molecules.

6 Discussion & Conclusion

In this paper we have presented the theory of Spherical CNNs and evaluated them on two important learning problems. We have defined S^2 and $\mathrm{SO}(3)$ cross-correlations, analyzed their properties, and implemented a Generalized FFT-based correlation algorithm. Our numerical results confirm the stability and accuracy of this algorithm, even for deep networks. Furthermore, we have shown that Spherical CNNs can effectively generalize across rotations, and achieve near state-of-the-art results on competitive 3D Model Recognition and Molecular Energy Regression challenges, without excessive feature engineering and task-tuning.

For intrinsically volumetric tasks like 3D model recognition, we believe that further improvements can be attained by generalizing further beyond $\mathrm{SO}(3)$ to the roto-translation group $\mathrm{SE}(3)$. The development of Spherical CNNs is an important first step in this direction. Another interesting generalization is the development of a Steerable CNN for the sphere (Cohen and Welling, 2017), which would make it possible to analyze vector fields such as global wind directions, as well as other sections of vector bundles over the sphere.

Perhaps the most exciting future application of the Spherical CNN is in omnidirectional vision. Although very little omnidirectional image data is currently available in public repositories, the increasing prevalence of omnidirectional sensors in drones, robots, and autonomous cars makes this a very compelling application of our work.

REFERENCES

L. C. Blum and J.-L. Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.*, 131:8732, 2009.

W. Boomsma and J. Frellsen. Spherical convolutions and their application in molecular modelling. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3436–3446. Curran Associates, Inc., 2017.

A.X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

G.S. Chirikjian and A.B. Kyatkin. *Engineering Applications of Noncommutative Harmonic Analysis*. CRC Press, 1 edition, may 2001. ISBN 9781420041767.

T.S. Cohen and M. Welling. Group equivariant convolutional networks. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, volume 48, pages 2990–2999, 2016.

T.S. Cohen and M. Welling. Steerable CNNs. In *ICLR*, 2017.

T.S. Cohen, M. Geiger, J. Koehler, and M. Welling. Convolutional networks for spherical signals. In *ICML Workshop on Principled Approaches to Deep Learning*, 2017.

S. Dieleman, K. W. Willett, and J. Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2), 2015.

S. Dieleman, J. De Fauw, and K. Kavukcuoglu. Exploiting Cyclic Symmetry in Convolutional Neural Networks. In *International Conference on Machine Learning (ICML)*, 2016.

J.B. Drake, P.H. Worley, and E.F. D'Azevedo. Algorithm 888: Spherical harmonic transform algorithms. *ACM Trans. Math. Softw.*, 35(3):23:1–23:23, 2008. doi: 10.1145/1391989.1404581.

J.R. Driscoll and D.M. Healy. Computing Fourier transforms and convolutions on the 2-sphere. *Advances in applied mathematics*, 1994.

G.B. Folland. *A Course in Abstract Harmonic Analysis*. CRC Press, 1995.

R. Gens and P. Domingos. Deep Symmetry Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

B. Gutman, Y. Wang, T. Chan, P.M. Thompson, and others. Shape registration with spherical cross correlation. *2nd MICCAI workshop*, 2008.

N. Guttenberg, N. Virgo, O. Witkowski, H. Aoki, and R. Kanai. Permutation-equivariant neural networks applied to dynamics prediction. 2016.

D. Healy, D. Rockmore, P. Kostelec, and S. Moore. FFTs for the 2-Sphere – Improvements and Variations. *The journal of Fourier analysis and applications*, 9(4):340–385, 2003.

P.J. Kostelec and D.N. Rockmore. SOFT: SO(3) Fourier Transforms. 2007. URL `http://www.cs.dartmouth.edu/~geelong/soft/soft20_fx.pdf`.

P.J. Kostelec and D.N. Rockmore. FFTs on the rotation group. *Journal of Fourier Analysis and Applications*, 14(2):145–179, 2008.

S. Kunis and D. Potts. Fast spherical Fourier algorithms. *Journal of Computational and Applied Mathematics*, 161:75–98, 2003.

A. Makadia, C. Geyer, and K. Daniilidis. Correspondence-free structure from motion. *Int. J. Comput. Vis.*, 75(3):311–327, December 2007.

D.K. Maslen. Efficient Computation of Fourier Transforms on Compact Groups. *Journal of Fourier Analysis and Applications*, 4(1), 1998.

G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, O.A. von Lilienfeld, and K. Müller. Learning invariant representations of molecules for atomization energy prediction. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 449–457. 2012.

L. Nachbin. *The Haar Integral*. 1965.

C. Olah. Groups and Group Convolutions, 2014. URL `https://colah.github.io/posts/2014-12-Groups-Convolution/`.

D. Potts, G. Steidl, and M. Tasche. Fast and stable algorithms for discrete spherical Fourier transforms. *Linear Algebra and its Applications*, 275:433–450, 1998.

D. Potts, J. Prestin, and A. Vollrath. A fast algorithm for nonequispaced Fourier transforms on the rotation group. *Numerical Algorithms*, pages 1–28, 2009.

A. Raj, A. Kumar, Y. Mroueh, P.T. Fletcher, et al. Local group invariant representations via orbit embeddings. *arXiv preprint arXiv:1612.01988*, 2016.

S. Ravanbakhsh, J. Schneider, and B. Poczos. Deep learning with sets and point clouds. In *International Conference on Learning Representations (ICLR) – workshop track*, 2017.

D.N. Rockmore. Recent Progress and Applications in Group FFTS. *NATO Science Series II: Mathematics, Physics and Chemistry*, 136:227–254, 2004.

M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, 108:058301, 2012.

M. Savva, F. Yu, H. Su, A. Kanezaki, T. Furuya, R. Ohbuchi, Z. Zhou, R. Yu, S. Bai, X. Bai, M. Aono, A. Tatsuma, S. Thermos, A. Axenopoulos, G. Th. Papadopoulos, P. Daras, X. Deng, Z. Lian, B. Li, H. Johan, Y. Lu, and S. Mk. Large-Scale 3D Shape Retrieval from ShapeNet Core55. In Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2017. ISBN 978-3-03868-030-7. doi: 10.2312/3dor.20171050.

Y.C. Su and K. Grauman. Learning spherical convolution for fast features from 360 imagery. *Adv. Neural Inf. Process. Syst.*, 2017.

M. Sugiura. *Unitary Representations and Harmonic Analysis*. John Wiley & Sons, New York, London, Sydney, Toronto, 2nd edition, 1990.

M.E. Taylor. *Noncommutative Harmonic Analysis*. American Mathematical Society, 1986. ISBN 0821815237.

M. Weiler, F.A. Hamprecht, and M. Storath. Learning steerable filters for rotation equivariant CNNs. 2017.

D.E. Worrall, S.J. Garbin, D. Turmukhambetov, and G.J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *CVPR*, 2017.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017a.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R.R. Salakhutdinov, and A.J. Smola. Deep sets. In *Advances in Neural Information Processing Systems 30*, pages 3393–3403, 2017b.

APPENDIX A: PARAMETERIZATION OF AND INTEGRATION ON S^2 AND $\mathrm{SO}(3)$

We use the ZYZ Euler parameterization for $\mathrm{SO}(3)$. An element $R \in \mathrm{SO}(3)$ is written as

$$R = R(\alpha, \beta, \gamma) = Z(\alpha)Y(\beta)Z(\gamma), \tag{10}$$

where $\alpha \in [0, 2\pi]$, $\beta \in [0, \pi]$ and $\gamma \in [0, 2\pi]$, and Z resp. Y are rotations around the Z and Y axes.

Using this parameterization, the normalized Haar measure is

$$dR = \frac{d\alpha}{2\pi}\frac{d\beta \sin(\beta)}{2}\frac{d\gamma}{2\pi} \tag{11}$$

We have $\int_{\mathrm{SO}(3)} dR = 1$. The Haar measure (Nachbin, 1965; Chirikjian and Kyatkin, 2001) is sometimes called the invariant measure because it has the property that $\int_{\mathrm{SO}(3)} f(R'R)dR = \int_{\mathrm{SO}(3)} f(R)dR$ (this is analogous to the more familiar property $\int_{\mathbb{R}} f(x+y)dx = \int_{\mathbb{R}} f(x)dx$ for functions on the line). This invariance property allows us to do many useful substitutions.

We have a related parameterization for the sphere. An element $x \in S^2$ is written

$$x(\alpha, \beta) = Z(\alpha)Y(\beta)n \tag{12}$$

where n is the north pole.

This parameterization makes explicit the fact that the sphere is a quotient $S^2 = \mathrm{SO}(3)/\mathrm{SO}(2)$, where $H = \mathrm{SO}(2)$ is the subgroup of rotations around the Z axis. Elements of this subgroup H leave the north pole invariant, and have the form $Z(\gamma)$. The point $x(\alpha, \beta) \in S^2$ is associated with the coset representative $\bar{x} = R(\alpha, \beta, 0) \in \mathrm{SO}(3)$. This element represents the coset $\bar{x}H = \{R(\alpha, \beta, \gamma) | \gamma \in [0, 2\pi]\}$.

The normalized Haar measure for the sphere is

$$dx = \frac{d\alpha}{2\pi}\frac{d\beta \sin\beta}{2} \tag{13}$$

The normalized Haar measure for $\mathrm{SO}(2)$ is

$$dh = \frac{d\gamma}{2\pi} \tag{14}$$

So we have $dR = dx\, dh$, again reflecting the quotient structure.

We can think of a function on S^2 as a γ-invariant function on $\mathrm{SO}(3)$. Given a function $f : S^2 \to \mathbb{C}$ we associate the function $\bar{f}(\alpha, \beta, \gamma) = f(\alpha, \beta)$. When using normalized Haar measures, we have:

$$\begin{aligned}
\int_{\mathrm{SO}(3)} \bar{f}(R)dR &= \frac{1}{8\pi^2}\int_0^{2\pi} d\alpha \int_0^{\pi} \sin\beta d\beta \int_0^{2\pi} d\gamma \bar{f}(\alpha, \beta, \gamma) \\
&= \frac{1}{8\pi^2}\int_0^{2\pi} d\alpha \int_0^{\pi} \sin\beta d\beta f(\alpha, \beta) \int_0^{2\pi} d\gamma \\
&= \frac{1}{4\pi}\int_0^{2\pi} d\alpha \int_0^{\pi} \sin\beta d\beta f(\alpha, \beta) \\
&= \int_{S^2} f(x)dx
\end{aligned} \tag{15}$$

This will allow us to define the Fourier transform on S^2 from the Fourier transform on $\mathrm{SO}(3)$, by viewing a function on S^2 as a γ-invariant function on $\mathrm{SO}(3)$ and taking its $\mathrm{SO}(3)$-Fourier transform.

APPENDIX B: CORRELATION & EQUIVARIANCE

We have defined the S^2 correlation as

$$[\psi \star f](R) = \langle L_R \psi, f \rangle = \int_{S^2} \sum_{k=1}^{K} \psi_k(R^{-1}x) f_k(x) dx. \tag{16}$$

Without loss of generality, we will analyze here the single-channel case $K = 1$.

This operation is equivariant:

$$\begin{aligned}
[\psi \star [L_Q f]](R) &= \int_{S^2} \psi(R^{-1}x) f(Q^{-1}x) dx \\
&= \int_{S^2} \psi(R^{-1}Qx) f(x) dx \\
&= \int_{S^2} \psi((Q^{-1}R)^{-1}x) f(x) dx \\
&= [\psi \star f](Q^{-1}R) \\
&= [L_Q[\psi \star f]](R)
\end{aligned} \tag{17}$$

A similar derivation can be made for the $\mathrm{SO}(3)$ correlation.

The spherical convolution defined by Driscoll and Healy (1994) is:

$$[f * \psi](x) = \int_{\mathrm{SO}(3)} f(Rn)\psi(R^{-1}x) dR \tag{18}$$

where n is the north pole. Note that in this definition, the output of the spherical convolution is a function on the sphere, not a function on $\mathrm{SO}(3)$ as in our definition of cross-correlation. Note further that unlike our definition, this definition involves an integral over $\mathrm{SO}(3)$.

If we write out the integral in terms of Euler angles, noting that the north-pole n is invariant to Z-axis rotations by γ, i.e. $R(\alpha, \beta, \gamma)n = Z(\alpha)Y(\beta)Z(\gamma)n = Z(\alpha)Y(\beta)n$, we see that this definition implicitly integrates over γ in only one of the factors (namely ψ), making it invariant wrt γ rotation. In other words, the filter is first "averaged" (making it circularly symmetric) before it is combined with f (This was observed before by Makadia et al. (2007)). We consider this to be much too limited for the purpose of pattern matching in spherical CNNs.

APPENDIX C: GENERALIZED FOURIER TRANSFORM

With each compact topological group (like $\mathrm{SO}(3)$) is associated a discrete set of orthogonal functions that arise as matrix elements of irreducible unitary representations of these groups. For the circle (the group $\mathrm{SO}(2)$) these are the complex exponentials (in the complex case) or sinusoids (for real functions). For $\mathrm{SO}(3)$, these functions are known as the Wigner D-functions.

As discussed in the paper, the Wigner D-functions are parameterized by a degree parameter $l \geq 0$ and order parameters $m, n \in [-l, \ldots, l]$. In other words, we have a set of matrix-valued functions $D^l : \mathrm{SO}(3) \to \mathbb{C}^{(2l+1)\times(2l+1)}$.

The Wigner D-functions are orthogonal:

$$\langle D^l_{mn}, D^{l'}_{m'n'} \rangle = \int_0^{2\pi} \frac{d\alpha}{2\pi} \int_0^{\pi} \frac{d\beta \sin\beta}{2} \int_0^{2\pi} \frac{d\gamma}{2\pi} D^l_{mn}(\alpha, \beta, \gamma) \overline{D^{l'}_{m'n'}}(\alpha, \beta, \gamma) = \frac{\delta_{ll'}\delta_{mm'}\delta_{nn'}}{2l+1} \tag{19}$$

Furthermore, they are complete, meaning that any well behaved function $f : \mathrm{SO}(3) \to \mathbb{C}$ can be written as a linear combination of Wigner D-functions. This is the idea of the Generalized Fourier Transform $\mathcal{F}$ on $\mathrm{SO}(3)$:

$$f(R) = [\mathcal{F}^{-1}\hat{f}](R) = \sum_{l=0}^{\infty} (2l+1) \sum_{m=-l}^{l} \sum_{n=-l}^{l} \hat{f}^l_{mn} D^l_{mn}(R) \tag{20}$$

where $\hat{f}^l_{mn}$ are called the Fourier coefficients of f. Using the orthogonality property of the Wigner D-functions, one can see that the Fourier coefficients can be retrieved by computing the inner product with the Wigner D-functions:

$$\begin{aligned}
[\mathcal{F}f]^l_{mn} &= \int_{\mathrm{SO}(3)} f(R)\overline{D^l_{mn}(R)}dR \\
&= \int_{\mathrm{SO}(3)} \left[\sum_{l'=0}^{\infty}(2l'+1)\sum_{m'=-l'}^{l'}\sum_{n'=-l'}^{l'}\hat{f}^{l'}_{m'n'}D^{l'}_{m'n'}(R)\right]\overline{D^l_{mn}(R)}dR \\
&= \sum_{l'=0}^{\infty}(2l'+1)\sum_{m'=-l'}^{l'}\sum_{n'=-l'}^{l'}\hat{f}^{l'}_{m'n'}\int_{\mathrm{SO}(3)} D^{l'}_{m'n'}(R)\overline{D^l_{mn}}dR \\
&= \hat{f}^l_{mn}
\end{aligned} \tag{21}$$

APPENDIX D: FOURIER THEOREMS

Fourier convolution theorems for $\mathrm{SO}(3)$ and $\S^2$ can be found in Kostelec and Rockmore (2008); Makadia et al. (2007); Gutman et al. (2008). We derive them here for completeness.

To derive the convolution theorems, we will use the defining property of the Wigner D-matrices: that they are (irreducible, unitary) *representations* of $\mathrm{SO}(3)$. This means that they satisfy:

$$D^l(R)D^l(R') = D^l(RR'), \tag{22}$$

for any $R, R' \in \mathrm{SO}(3)$. Notice that the complex exponentials satisfy an analogous criterion for the circle group $S^1 \cong \mathrm{SO}(2)$. That is, $e^{inx}e^{iny} = e^{in(x+y)}$, where $x + y$ is the group operation for $\mathrm{SO}(2)$.

Unitarity means that $D^l(R)D^{l\dagger}(R) = I$. Irreducibility means, essentially, that the set of matrices $\{D^l(R) \,|\, R \in \mathrm{SO}(3)\}$ cannot be simultaneously block-diagonalized.

To derive the Fourier theorem for $\mathrm{SO}(3)$, we use the invariance of the integration measure dR: $\int_{\mathrm{SO}(3)} f(R'R)dR = \int_{\mathrm{SO}(3)} f(R)dR$.

With these facts understood, we can proceed to derive:

$$\begin{aligned}
\widehat{\psi \star f}^l &= \int_{\mathrm{SO}(3)} (\psi \star f)(R)\overline{D^l(R)}dR \\
&= \int_{\mathrm{SO}(3)}\int_{\mathrm{SO}(3)} \psi(R^{-1}R')f(R')dR'\overline{D^l(R)}dR \\
&= \int_{\mathrm{SO}(3)}\int_{\mathrm{SO}(3)} \psi(R^{-1})f(R')\overline{D^l(R'R)}dR'dR \\
&= \int_{\mathrm{SO}(3)} f(R')\overline{D^l(R')}dR' \int_{\mathrm{SO}(3)} \psi(R^{-1})\overline{D^l(R)}dR \\
&= \int_{\mathrm{SO}(3)} f(R')\overline{D^l(R')}dR' \int_{\mathrm{SO}(3)} \psi(R)\overline{D^l(R)}^{\dagger}dR \\
&= \hat{f}^l\,\hat{\psi}^{l\dagger}
\end{aligned} \tag{23}$$

So the $\mathrm{SO}(3)$-Fourier transform of the $\mathrm{SO}(3)$ convolution of ψ and f is equal to the matrix product of the $\mathrm{SO}(3)$-Fourier transforms $\hat{f}$ and $\hat{\psi}$.

For the sphere, we can derive an analogous transform that is sometimes called the spherical harmonics transform. The spherical harmonics $Y^l_m : S^2 \to \mathbb{C}$ are a complete orthogonal family of functions. The spherical harmonics are related to the Wigner D functions by the relation $D^l_{mn}(\alpha, \beta, \gamma) = Y^l_m(\alpha, \beta)e^{in\gamma}$, so that $Y^l_m(\alpha, \beta) = D^l_{m0}(\alpha, \beta, 0)$.

The S^2 convolution of f_1 and f_2 is equivalent to the $\mathrm{SO}(3)$ convolution of the associated right-invariant functions $\bar{f}_1, \bar{f}_2$ (see Appendix A):

$$\begin{aligned}
[f_1 \star f_2](R) &= \int_{S^2} f_1(R^{-1}x) f_2(x) dx \\
&= \int_{\mathrm{SO}(2)} \int_{S^2} f_1(R^{-1}x) f_2(x) dx dh \\
&= \int_{\mathrm{SO}(3)} \bar{f}_1(R^{-1}R') \bar{f}_2(R') dR' \\
&= [\bar{f}_1 \star \bar{f}_2](R)
\end{aligned} \tag{24}$$

The Fourier transform of a right invariant function on $\mathrm{SO}(3)$ equals

$$\begin{aligned}
[\mathcal{F}\bar{f}]^l_{mn} &= \int_0^{2\pi} \frac{d\alpha}{2\pi} \int_0^{\pi} \frac{d\beta \sin\beta}{2} \int_0^{2\pi} \frac{d\gamma}{2\pi} \bar{f}(\alpha,\beta,\gamma) \overline{D^l_{mn}(\alpha,\beta,\gamma)} \\
&= \int_0^{2\pi} \frac{d\alpha}{2\pi} \int_0^{\pi} \frac{d\beta \sin\beta}{2} f(\alpha,\beta) \int_0^{2\pi} \frac{d\gamma}{2\pi} \overline{D^l_{mn}(\alpha,\beta,\gamma)} \\
&= \delta_{n0} \int_0^{2\pi} \frac{d\alpha}{2\pi} \int_0^{\pi} \frac{d\beta \sin\beta}{2} f(\alpha,\beta) \overline{D^l_{m0}(\alpha,\beta,0)} \\
&= \delta_{n0} \int_{S^2} f(x) \overline{Y^l_m(x)} dx
\end{aligned} \tag{25}$$

So we can think of the S^2 Fourier transform of a function on S^2 as the $n = 0$ column of the $\mathrm{SO}(3)$ Fourier transform of the associated right-invariant function. This is a beautiful result that we have not been able to find a reference for, though it seems likely that it has been observed before.

5.2 3D Steerable CNN

The following paper is the preprint version of Weiler et al. (2018).

Candidate contributions Motivated by Taco Cohen's vision, the candidate initiated the project, derived the kernel space constraint, wrote the first network implementation and ran the Shrec17 experiment.

3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data

Maurice Weiler*
University of Amsterdam
m.weiler@uva.nl

Mario Geiger*
EPFL
mario.geiger@epfl.ch

Max Welling
University of Amsterdam, CIFAR,
Qualcomm AI Research
m.welling@uva.nl

Wouter Boomsma
University of Copenhagen
wb@di.ku.dk

Taco Cohen
Qualcomm AI Research
taco.cohen@gmail.com

Abstract

We present a convolutional network that is equivariant to rigid body motions. The model uses scalar-, vector-, and tensor fields over 3D Euclidean space to represent data, and equivariant convolutions to map between such representations. These $\mathrm{SE}(3)$-equivariant convolutions utilize kernels which are parameterized as a linear combination of a complete steerable kernel basis, which is derived analytically in this paper. We prove that equivariant convolutions are the most general equivariant linear maps between fields over $\mathbb{R}^3$. Our experimental results confirm the effectiveness of 3D Steerable CNNs for the problem of amino acid propensity prediction and protein structure classification, both of which have inherent $\mathrm{SE}(3)$ symmetry.

1 Introduction

Increasingly, machine learning techniques are being applied in the natural sciences. Many problems in this domain, such as the analysis of protein structure, exhibit exact or approximate symmetries. It has long been understood that the equations that define a model or natural law should respect the symmetries of the system under study, and that knowledge of symmetries provides a powerful constraint on the space of admissible models. Indeed, in theoretical physics, this idea is enshrined as a fundamental principle, known as Einstein's principle of general covariance. Machine learning, which is, like physics, concerned with the induction of predictive models, is no different: our models must respect known symmetries in order to produce physically meaningful results.

A lot of recent work, reviewed in Sec. 2, has focused on the problem of developing equivariant networks, which respect some known symmetry. In this paper, we develop the theory of $\mathrm{SE}(3)$-equivariant networks. This is far from trivial, because $\mathrm{SE}(3)$ is both non-commutative and non-compact. Nevertheless, at run-time, all that is required to make a 3D convolution equivariant using our method, is to parameterize the convolution kernel as a linear combination of pre-computed steerable basis kernels. Hence, the 3D Steerable CNN incorporates equivariance to symmetry transformations without deviating far from current engineering best practices.

The architectures presented here fall within the framework of Steerable G-CNNs [8, 10, 41, 46], which represent their input as fields over a homogeneous space ($\mathbb{R}^3$ in this case), and use steerable

* Equal Contribution. MG initiated the project, derived the kernel space constraint, wrote the first network implementation and ran the Shrec17 experiment. MW solved the kernel constraint analytically, designed the anti-aliased kernel sampling in discrete space and coded / ran many of the CATH experiments.
Source code is available at https://github.com/mariogeiger/se3cnn.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

filters [15, 38] to map between such representations. In this paper, the convolution kernel is modeled as a tensor field satisfying an equivariance constraint, from which steerable filters arise automatically.

We evaluate the 3D Steerable CNN on two challenging problems: prediction of amino acid preferences from atomic environments, and classification of protein structure. We show that a 3D Steerable CNN improves upon state of the art performance on the former task. For the latter task, we introduce a new and challenging dataset, and show that the 3D Steerable CNN consistently outperforms a strong CNN baseline over a wide range of trainingset sizes.

2 Related Work

There is a rapidly growing body of work on neural networks that are equivariant to some group of symmetries [3, 9, 10, 12, 19, 20, 29, 31–33, 37, 43, 47]. At a high level, these models can be categorized along two axes: the group of symmetries they are equivariant to, and the type of geometrical features they use [8]. The class of regular G-CNNs represents the input signal in terms of *scalar fields* on a group G (e.g. $\mathrm{SE}(3)$) or homogeneous space G/H (e.g. $\mathbb{R}^3 = \mathrm{SE}(3)/\mathrm{SO}(3)$) and maps between feature spaces of consecutive layers via group convolutions [9, 30]. Regular G-CNNs can be seen as a special case of steerable (or induced) G-CNNs which represent features in terms of *more general fields* over a homogeneous space [8, 10, 28, 31, 41]. The models described in this paper are of the steerable kind, since they use general fields over $\mathbb{R}^3$. These fields typically consist of multiple independently transforming geometrical quantities (vectors, tensors, etc.), and can thus be seen as a formalization of the idea of convolutional capsules [18, 35].

Regular 3D G-CNNs operating on voxelized data via group convolutions were proposed in [44, 45]. These architectures were shown to achieve superior data efficiency over conventional 3D CNNs in tasks like medical imaging and 3D model recognition. In contrast to 3D Steerable CNNs, both networks are equivariant to certain discrete rotations only.

The most closely related works achieving full $\mathrm{SE}(3)$ equivariance are the Tensor Field Network (TFN) [41] and the N-Body networks (NBNs) [27]. The main difference between 3D Steerable CNNs and both TFN and NBN is that the latter work on irregular point clouds, whereas our model operates on regular 3D grids. Point clouds are more general, but regular grids can be processed more efficiently on current hardware. The second difference is that whereas the TFN and NBN use Clebsch-Gordan coefficients to parameterize the network, we simply parameterize the convolution kernel as a linear combination of steerable basis filters. Clebsch-Gordan coefficient tensors have 6 indices, and depend on various phase and normalization conventions, making them tricky to work with. Our implementation requires only a very minimal change from the conventional 3D CNN. Specifically, we compute conventional 3D convolutions with filters that are a linear combination of pre-computed basis filters. Further, in contrast to TFN, we derive this filter basis directly from an equivariance constraint and can therefore prove its completeness.

The two dimensional analog of our work is the $\mathrm{SE}(2)$ equivariant harmonic network [46]. The harmonic network and 3D steerable CNN use features that transform under irreducible representations of $\mathrm{SO}(2)$ resp. $\mathrm{SO}(3)$, and use filters related to the circular resp. spherical harmonics.

$\mathrm{SE}(3)$ equivariant models were already investigated in classical computer vision and signal processing. In [34, 39], a spherical tensor algebra was utilized to expand signals in terms of spherical tensor fields. In contrast to 3D Steerable CNNs, this expansion is fixed and not learned. Similar approaches were used for detection and crossing preserving enhancement of fibrous structures in volumetric biomedical images [13, 22, 23].

3 Convolutional feature spaces as fields

A convolutional network produces a stack of K_n feature maps f_k in each layer n. In 3D, we can model the feature maps as (well-behaved) functions $f_k : \mathbb{R}^3 \to \mathbb{R}$. Written another way, we have a map $f : \mathbb{R}^3 \to \mathbb{R}^{K_n}$ that assigns to each position x a feature vector $f(x)$ that lives in what we call the *fiber* $\mathbb{R}^{K_n}$ at x. In practice f will have compact support, meaning that $f(x) = 0$ outside of some compact domain $\Omega \in \mathbb{R}^3$. We thus define the feature space $\mathcal{F}_n$ as the vector space of continuous maps from $\mathbb{R}^3$ to $\mathbb{R}^{K_n}$ with compact support.

In this paper, we impose additional structure on the fibers. Specifically, we assume the fiber consists of a number of geometrical quantities, such as scalars, vectors, and tensors, stacked into a single

K_n-dimensional vector. The assignment of such a geometrical quantity to each point in space is called a *field*. Thus, the feature spaces consist of a number of fields, each of which consists of a number of channels (dimensions).

Before deriving $\mathrm{SE}(3)$-equivariant networks in Sec. 4 we discuss the transformation properties of fields and the kinds of fields we use in 3D Steerable CNNs.

3.1 Fields, Transformations and Disentangling

What makes a geometrical quantity (e.g. a vector) anything more than an arbitrary grouping of feature channels? The answer is that under rigid body motions, information flows within the channels of a single geometrical quantity, but not between different quantities. This idea is known as Weyl's principle, and has been proposed as a way of formalizing the notion of disentangling [6, 24].

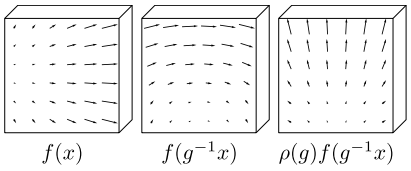


Figure 1: To transform a vector field (L) by a $90°$ rotation g, first move each arrow to its new position (C), keeping its orientation the same, then rotate the vector itself (R). This is described by the induced representation $\pi = \mathrm{Ind}_{\mathrm{SO}(3)}^{\mathrm{SE}(2)} \rho$, where $\rho(g)$ is a 3×3 rotation matrix that mixes the three coordinate channels.

As an example, consider the three-dimensional vector field over $\mathbb{R}^3$, shown in Figure 1. At each point $x \in \mathbb{R}^3$ there is a vector $f(x)$ of dimension $K = 3$. If the field is translated by t, each vector $x - t$ would simply move to a new (translated) position x. When the field is rotated, however, two things happen: the vector at $r^{-1}x$ is moved to a new (rotated) position x, *and* each vector is itself rotated by a 3×3 rotation matrix $\rho(r)$. Thus, the rotation operator $\pi(r)$ for vector fields is defined as $[\pi(r)f](x) := \rho(r)f(r^{-1}x)$. Notice that in order to rotate this field, we need all three channels: we cannot rotate each channel independently, because ρ introduces a functional dependency between them. For contrast, consider the common situation where in the input space we have an RGB image with $K = 3$ channels. Then $f(x) \in \mathbb{R}^3$, and the rotation can be described using the same formula $\rho(r)f(r^{-1}x)$ if we choose $\rho(r) = I_3$ to be the 3×3 identity matrix for all r. Since $\rho(r)$ is diagonal for all r, the channels do not get mixed, and so in geometrical terms, we would describe this feature space as consisting of three scalar fields, *not* a 3D vector field. The RGB channels each have an independent physical meaning, while the x and y coordinate channels of a vector do not.

The RGB and 3D-vector cases constitute two examples of fields, each one determined by a different choice of ρ. As one might guess, there is a one-to-one correspondence between the type of field and the type of transformation law (group representation) ρ. Hence, we can speak of a ρ-field.

So far, we have concentrated on the behaviour of a field under rotations and translations separately. A 3D rigid body motion $g \in \mathrm{SE}(3)$ can always be decomposed into a rotation $r \in \mathrm{SO}(3)$ and a translation $t \in \mathbb{R}^3$, written as $g = tr$. So the transformation law for a ρ-field is given by the formula

$$[\pi(tr)f](x) := \rho(r)f(r^{-1}(x - t)). \tag{1}$$

The map π is known as the representation of $\mathrm{SE}(3)$ induced by the representation ρ of $\mathrm{SO}(3)$, which is denoted by $\pi = \mathrm{Ind}_{\mathrm{SO}(3)}^{\mathrm{SE}(3)} \rho$. For more information on induced representations, see [5, 8, 17].

3.2 Irreducible $\mathrm{SO}(3)$ features

We have seen that there is a correspondence between the type of field and the type of inducing representation ρ, which describes the rotation behaviour of a single fiber. To get a better understanding of the space of possible fields, we will now define precisely what it means to be a representation of $\mathrm{SO}(3)$, and explain how any such representation can be constructed from elementary building blocks called irreducible representations.

A group representation ρ assigns to each element in the group an invertible $n \times n$ matrix. Here n is the dimension of the representation, which can be any positive integer (or even infinite). For ρ to be called a representation of G, it has to satisfy $\rho(gg') = \rho(g)\rho(g')$, where gg' denotes the composition of two transformations $g, g' \in G$, and $\rho(g)\rho(g')$ denotes matrix multiplication.

To make this more concrete, and to introduce the concept of an irreducible representation, we consider the classical example of a rank-2 tensor (i.e. matrix). A 3×3 matrix A transforms under rotations as $A \mapsto R(r)AR(r)^T$, where $R(r)$ is the 3×3 rotation matrix representation of the abstract group element $r \in \mathrm{SO}(3)$. This can be written in matrix-vector form using the Kronecker / tensor product: $\mathrm{vec}(A) \mapsto [R(r) \otimes R(r)]\,\mathrm{vec}(A) \equiv \rho(r)\,\mathrm{vec}(A)$. This is a 9-dimensional representation of $\mathrm{SO}(3)$.

One can easily verify that the symmetric and anti-symmetric parts of A remain symmetric respectively anti-symmetric under rotations. This splits $\mathbb{R}^{3\times3}$ into 6- and 3-dimensional linear subspaces that transform independently. According to Weyl's principle, these may be considered as distinct quantities, even if it is not immediately visible by looking at the coordinates A_{ij}. The 6-dimensional space can be further broken down, because scalar matrices $A_{ij} = \alpha\delta_{ij}$ (which are invariant under rotation) and traceless symmetric matrices also transform independently. Thus a rank-2 tensor decomposes into representations of dimension 1 (trace), 3 (anti-symmetric part), and 5 (traceless symmetric part). In representation-theoretic terms, we have reduced the 9-dimensional representation ρ into irreducible representations of dimension $1, 3$ and 5. We can write this as

$$\rho(r) = Q^{-1}\left[\bigoplus_{l=0}^{2} D^l(r)\right]Q, \tag{2}$$

where we use $\bigoplus$ to denote the construction of a block-diagonal matrix with blocks $D^l(r)$, and Q is a change of basis matrix that extracts the trace, symmetric-traceless and anti-symmetric parts of A.

More generally, it can be shown that any representation of $\mathrm{SO}(3)$ can be decomposed into irreducible representations of dimension $2l+1$, for $l = 0, 1, 2, \ldots, \infty$. The irreducible representation acting on this $2l+1$ dimensional space is known as the Wigner-D matrix of order l, denoted $D^l(r)$. Note that the Wigner-D matrix of order 4 is a representation of dimension 9, it has the same dimension as the representation ρ acting on A but these are two different representations.

Since any $\mathrm{SO}(3)$ representation can be decomposed into irreducibles, we only use irreducible features in our networks. This means that the feature vector $f(x)$ in layer n is a stack of F_n features $f^i(x) \in \mathbb{R}^{2l_i+1}$, so that $K_n = \sum_{i=1}^{F_n} 2l_{in} + 1$.

4 $\mathrm{SE}(3)$-Equivariant Networks

Our general approach to building $\mathrm{SE}(3)$-equivariant networks will be as follows: First, we will specify for each layer n a linear transformation law $\pi_n(g) : \mathcal{F}_n \to \mathcal{F}_n$, which describes how the feature space $\mathcal{F}_n$ transforms under transformations of the input by $g \in \mathrm{SE}(3)$. Then, we will study the vector space $\mathrm{Hom}_{\mathrm{SE}(3)}(\mathcal{F}_n, \mathcal{F}_{n+1})$ of equivariant linear maps (intertwiners) Φ between adjacent feature spaces:

$$\mathrm{Hom}_{\mathrm{SE}(3)}(\mathcal{F}_n, \mathcal{F}_{n+1}) = \{\Phi \in \mathrm{Hom}(\mathcal{F}_n, \mathcal{F}_{n+1}) \mid \Phi\pi_n(g) = \pi_{n+1}(g)\Phi, \;\; \forall g \in \mathrm{SE}(3)\} \tag{3}$$

Here $\mathrm{Hom}(\mathcal{F}_n, \mathcal{F}_{n+1})$ is the space of linear (not necessarily equivariant) maps from $\mathcal{F}_n$ to $\mathcal{F}_{n+1}$.

By finding a basis for the space of intertwiners and parameterizing Φ_n as a linear combination of basis maps, we can make sure that layer $n+1$ transforms according to π_{n+1} if layer n transforms according to π_n, thus guaranteeing equivariance of the whole network by induction.

As explained in the previous section, fields transform according to induced representations [5, 8, 10, 17]. In this section we show that equivariant maps between induced representations of $\mathrm{SE}(3)$ can always be expressed as convolutions with equivariant / steerable filter banks. The space of equivariant filter banks turns out to be a linear subspace of the space of filter banks of a conventional 3D CNN. The filter banks of our network are expanded in terms of a basis of this subspace with parameters corresponding to expansion coefficients.

Sec. 4.1 derives the linear constraint on the kernel space for arbitrary induced representations. From Sec. 4.2 on we specialize to representations induced from irreducible representations of $\mathrm{SO}(3)$ and derive a basis of the equivariant kernel space for this choice analytically. Subsequent sections discuss choices of equivariant nonlinearities and the actual discretized implementation.

4.1 The Subspace of Equivariant Kernels

A continuous linear map between $\mathcal{F}_n$ and $\mathcal{F}_{n+1}$ can be written using a continuous kernel κ with signature $\kappa : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^{K_{n+1} \times K_n}$, as follows:

$$[\kappa \cdot f](x) = \int_{\mathbb{R}^3} \kappa(x, y) f(y) dy \tag{4}$$

Lemma 1. *The map $f \mapsto \kappa \cdot f$ is equivariant if and only if for all $g \in \mathrm{SE}(3)$,*

$$\kappa(gx, gy) = \rho_2(r) \kappa(x, y) \rho_1(r)^{-1}, \tag{5}$$

Proof. For this map to be equivariant, it must satisfy $\kappa \cdot [\pi_1(g) f] = \pi_2(g)[\kappa \cdot f]$. Expanding the left hand side of this constraint, using $g = tr$, and the substitution $y \mapsto gy$, we find:

$$\kappa \cdot [\pi_1(g) f](x) = \int_{\mathbb{R}^3} \kappa(x, gy) \rho_1(r) f(y) dy \tag{6}$$

For the right hand side,

$$\pi_2(g)[\kappa \cdot f](x) = \rho_2(r) \int_{\mathbb{R}^3} \kappa(g^{-1}x, y) f(y) dy. \tag{7}$$

Equating these, and using that the equality has to hold for arbitrary $f \in \mathcal{F}_n$, we conclude:

$$\rho_2(r) \kappa(g^{-1}x, y) = \kappa(x, gy) \rho_1(r). \tag{8}$$

Substitution of $x \mapsto gx$ and right-multiplication by $\rho_1(r)^{-1}$ yields the result. □

Theorem 2. *A linear map from $\mathcal{F}_n$ to $\mathcal{F}_{n+1}$ is equivariant if and only if it is a cross-correlation with a rotation-steerable kernel.*

Proof. Lemma 1 implies that we can write κ in terms of a one-argument kernel, since for $g = -x$:

$$\kappa(x, y) = \kappa(0, y - x) \equiv \kappa(y - x). \tag{9}$$

Substituting this into Equation 4, we find

$$[\kappa \cdot f](x) = \int_{\mathbb{R}^3} \kappa(x, y) f(y) dy = \int_{\mathbb{R}^3} \kappa(y - x) f(y) dy = [\kappa \star f](x). \tag{10}$$

Cross-correlation is always translation-equivariant, but Eq. 5 still constrains κ rotationally:

$$\kappa(rx) = \rho_2(r) \kappa(x) \rho_1(r)^{-1}. \tag{11}$$

A kernel satisfying this constraint is called *rotation-steerable*. □

We note that $\kappa \star f$ (Eq. 10) is exactly the operation used in a conventional convolutional network, just written in an unconventional form, using a matrix-valued kernel ("propagator") $\kappa : \mathbb{R}^3 \to \mathbb{R}^{K_{n+1} \times K_n}$.

Since Eq. 11 is a linear constraint on the correlation kernel κ, the space of equivariant kernels (i.e. those satisfying Eq. 11) forms a vector space. We will now proceed to compute a basis for this space, so that we can parameterize the kernel as a linear combination of basis kernels.

4.2 Solving for the Equivariant Kernel Basis

As mentioned before, we assume that the K_n-dimensional feature vectors $f(x) = \oplus_i f^i(x)$ consist of irreducible features $f^i(x)$ of dimension $2\,l_{in} + 1$. In other words, the representation $\rho_n(r)$ that acts on fibers in layer n is block-diagonal, with irreducible representation $D^{l_{in}}(r)$ as the i-th block. This implies that the kernel $\kappa : \mathbb{R}^3 \to \mathbb{R}^{K_{n+1} \times K_n}$ splits into blocks[1] $\kappa^{jl} : \mathbb{R}^3 \to \mathbb{R}^{(2j+1) \times (2l+1)}$ mapping between irreducible features. The blocks themselves are by Eq. 11 constrained to transform as

$$\kappa^{jl}(rx) = D^j(r) \kappa^{jl}(x) D^l(r)^{-1}. \tag{12}$$

[1] For more details on the block structure see Sec. 2.7 of [10]

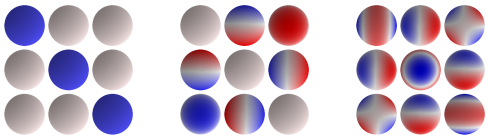


Figure 2: Angular part of the basis for the space of steerable kernels κ^{jl} (for $j = l = 1$, i.e. 3D *vector* fields as input and output). From left to right we plot three 3×3 matrices, for $j - l \leq J \leq j + l$ i.e. $J = 0, 1, 2$. Each 3×3 matrix corresponds to one learnable parameter per radial basis function φ^m. A seasoned eye will see the identity, the curl ($\nabla\wedge$) and the gradient of the divergence ($\nabla\nabla\cdot$).

To bring this constraint into a more manageable form, we vectorize these kernel blocks to $\mathrm{vec}(\kappa^{jl}(x))$, so that we can rewrite the constraint as a matrix-vector equation[2]

$$\mathrm{vec}(\kappa^{jl}(rx)) = [D^j \otimes D^l](r)\,\mathrm{vec}(\kappa^{jl}(x)), \tag{13}$$

where we used the orthogonality of D^l. The tensor product of representations is itself a representation, and hence can be decomposed into irreducible representations. For irreducible $\mathrm{SO}(3)$ representations D^j and D^l of order j and l it is well known [17] that $D^j \otimes D^l$ can be decomposed in terms of $2\min(j, l) + 1$ irreducible representations of order[3] $|j - l| \leq J \leq j + l$. That is, we can find a change of basis matrix[4] Q of shape $(2l + 1)(2j + 1) \times (2l + 1)(2j + 1)$ such that the representation becomes block diagonal:

$$[D^j \otimes D^l](r) = Q^T \left[\bigoplus\nolimits_{J=|j-l|}^{j+l} D^J(r)\right] Q \tag{14}$$

Thus, we can change the basis to $\eta^{jl}(x) := Q\,\mathrm{vec}(\kappa^{jl}(x))$ such that constraint 12 becomes

$$\eta^{jl}(rx) = \left[\bigoplus\nolimits_{J=|j-l|}^{j+l} D^J(r)\right] \eta^{jl}(x). \tag{15}$$

The block diagonal form of the representation in this basis reveals that η^{jl} decomposes into $2\min(j, l) + 1$ invariant subspaces of dimension $2J + 1$ with separated constraints:

$$\eta^{jl}(x) = \bigoplus\nolimits_{J=|j-l|}^{j+l} \eta^{jl,J}(x)\,, \qquad \eta^{jl,J}(rx) = D^J(r)\eta^{jl,J}(x) \tag{16}$$

This is a famous equation for which the *unique* and *complete* solution is well-known to be given by the spherical harmonics $Y^J(x) = (Y^J_{-J}(x), \ldots, Y^J_J(x)) \in \mathbb{R}^{2J+1}$. More specifically, since x lives in $\mathbb{R}^3$ instead of the sphere, the constraint only restricts the angular part of η^{jl} but leaves its radial part free. Therefore, the solutions are given by spherical harmonics modulated by an arbitrary continuous radial function $\varphi : \mathbb{R}^+ \to \mathbb{R}$ as $\eta^{jl,J}(x) = \varphi(\|x\|)Y^J(x/\|x\|)$.

To obtain a complete basis, we can choose a set of radial basis functions $\varphi^m : \mathbb{R}_+ \to \mathbb{R}$, and define kernel basis functions $\eta^{jl,Jm}(x) = \varphi^m(\|x\|)\,Y^J(x/\|x\|)$. Following [43], we choose a Gaussian radial shell $\varphi^m(\|x\|) = \exp\left(-\frac{1}{2}(\|x\| - m)^2/\sigma^2\right)$ in our implementation. The angular dependency at a fixed radius of the basis for $j = l = 1$ is shown in Figure 2.

By mapping each $\eta^{jl,Jm}$ back to the original basis via Q^T and unvectorizing, we obtain a basis $\kappa^{jl,Jm}$ for the space of equivariant kernels between features of order j and l. This basis is indexed by the radial index m and frequency index J. In the forward pass, we linearly combine the basis kernels as $\kappa^{jl} = \sum_{Jm} w^{jl,Jm}\kappa^{jl,Jm}$ using learnable weights w, and stack them into a complete kernel κ, which is passed to a standard 3D convolution routine.

4.3 Equivariant Nonlinearities

In order for the whole network to be equivariant, every layer, including the nonlinearities, must be equivariant. In a regular G-CNN, any elementwise nonlinearity will be equivariant because the regular representation acts by permuting the activations. In a steerable G-CNN however, special equivariant nonlinearities are required.

[2] vectorize correspond to flatten it in `numpy` and the tensor product correspond to `np.kron`

[3] There is a fascinating analogy with the quantum states of a two particle system for which the angular momentum states decompose in a similar fashion.

[4] Q can be expressed in terms of Clebsch-Gordan coefficients, but here we only need to know it exists.

Trivial irreducible features, corresponding to scalar fields, do not transform under rotations. So for these features we use conventional nonlinearities like ReLUs or sigmoids. For higher order features we considered tensor product nonlinearities [27] and norm nonlinearities [46], but settled on a novel gated nonlinearity. For each non-scalar irreducible feature $\kappa_n^i \star f_{n-1}(x) = f_n^i(x) \in \mathbb{R}^{2l_{in}+1}$ in layer n, we produce a scalar gate $\sigma(\gamma_n^i \star f_{n-1}(x))$, where σ denotes the sigmoid function and γ_n^i is another learnable rotation-steerable kernel. Then, we multiply the feature (a non-scalar field) by the gate (a scalar field): $f_n^i(x)\,\sigma(\gamma_n^i \star f_{n-1}(x))$. Since $\gamma_n^i \star f_{n-1}$ is a scalar field, $\sigma(\gamma_n^i \star f_{n-1})$ is a scalar field, and multiplying any feature by a scalar is equivariant. See Section 1.3 and Figure 5 in the Supplementary Material for details.

4.4 Discretized Implementation

In a computer implementation of $\mathrm{SE}(3)$ equivariant networks, we need to sample both the fields / feature maps and the kernel on a discrete sampling grid in $\mathbb{Z}^3$. Since this could introduce aliasing artifacts, care is required to make sure that high-frequency filters, corresponding to large values of J, are not sampled on a grid of low spatial resolution. This is particularly important for small radii since near the origin only a small number of pixels is covered per solid angle. In order to prevent aliasing we hence introduce a radially dependent angular frequency cutoff. Aliasing effect originating from the radial part of the kernel basis are counteracted by choosing a smooth Gaussian radial profile as described above. Below we describe how our implementation works in detail.

4.4.1 Kernel space precomputation

Before training, we compute basis kernels $\kappa^{jl,Jm}(x_i)$ sampled on a $s \times s \times s$ cubic grid of points $x_i \in \mathbb{Z}^3$, as follows. For each pair of output and input orders j and l we first sample spherical harmonics $Y^J, |j-l| \leq J \leq j+l$ in a radially independent manner in an array of shape $(2J+1) \times s \times s \times s$. Then, we transform the spherical harmonics back to the original basis by multiplying by $Q^J \in \mathbb{R}^{(2j+1)(2l+1)\times(2J+1)}$, consisting of $2J+1$ adjacent columns of Q, and unvectorize the resulting array to $\mathrm{unvec}(Q^J Y^J(x_i))$ which has shape $(2j+1) \times (2l+1) \times s \times s \times s$.

The matrix Q itself could be expressed in terms of Clebsch-Gordan coefficients [17], but we find it easier to compute it by numerically solving Eq. 14.

The radial dependence is introduced by multiplying the cubes with each windowing function φ^m. We use integer means $m = 0, \ldots, \lfloor s/2 \rfloor$ and a fixed width of $\sigma = 0.6$ for the radial Gaussian windows.

Sampling high-order spherical harmonics will introduce aliasing effects, particularly near the origin. Hence, we introduce a radius-dependent bandlimit J^m_{max}, and create basis functions only for $|j-l| \leq J \leq J^m_{\text{max}}$. Each basis kernel is scaled to unit norm for effective signal propagation [43]. In total we get $B = \sum_{m=0}^{\lfloor s/2 \rfloor} \sum_{|j-l|}^{J^m_{\text{max}}} 1 \leq (\lfloor s/2 \rfloor + 1)(2\min(j,l)+1)$ basis kernels mapping between fields of order j and l, and thus a basis array of shape $B \times (2j+1) \times (2l+1) \times s \times s \times s$.

4.4.2 Spatial dimension reduction

We found that the performance of the Steerable CNN models depends critically on the way of downsampling the fields. In particular, the standard procedure of downsampling via strided convolutions performed poorly compared to smoothing features maps before subsampling. We followed [1] and experiment with applying a low pass filtering before performing the downsampling step which can be implemented either via an additional strided convolution with a Gaussian kernel or via an average pooling. We observed significant improvements of the rotational equivariance by doing so. See Table 2 in the Supplementary Material for a comparison between performances with and without low pass filtering.

4.4.3 Forward pass

At training time, we linearly combine the basis kernels using learned weights, and stack them together into a full filter bank of shape $K_{n+1} \times K_n \times s \times s \times s$, which is used in a standard convolution routine. Once the network is trained, we can convert the network to a standard 3D CNN by linearly combining the basis kernels with the learned weights, and storing only the resulting filter bank.

5 Experiments

We performed several experiments to gauge the performance and data efficiency of our model.

5.1 Tetris

In order to confirm the equivariance of our model, we performed a variant of the Tetris experiments reported by [41]. We constructed a 4-layer 3D Steerable CNN and trained it to classify 8 kinds of Tetris blocks, stored as voxel grids, in a fixed orientation. Then we test on Tetris blocks rotated by random rotations in $\mathrm{SO}(3)$. As expected, the 3D Steerable CNN generalizes over rotations and achieves $99 \pm 2\%$ accuracy on the test set. In contrast, a conventional CNN is not able to generalize over larger unseen rotations and gets a result of only $27 \pm 7\%$. For both networks we repeated the experiment over 17 runs.

5.2 3D model classification

Moving beyond the simple Tetris blocks, we next considered classification of more complex 3D objects. The SHREC17 task [36], which contains 51300 models of 3D shapes belonging to 55 classes (chair, table, light, oven, keyboard, etc), has a 'perturbed' category where images are arbitrarily rotated, making it a well-suited test case for our model. We converted the input into voxel grids of size 64x64x64, and used an architecture similar to the Tetris case, but with an increased number of layers (see Table 3 in the Supplementary Material). Although we have not done extensive fine-tuning on this dataset, we find our model to perform comparably to the current state of the art, see Figure 3 and Table 4 in the Supplementary Material.

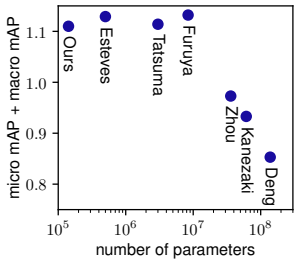


Figure 3: Shrec17 results[2, 7, 14, 16, 25, 36, 40]. Comparison of different architectures by number of parameters and score. See Table 4 in the Supplementary Material for all the details.

5.3 Visualization of the equivariance property

We made a movie to show the action of rotating the input on the internal fields. We found that the action are remarkably stable. A visualization is provided in `https://youtu.be/ENLJACPHSEA`.

5.4 Amino acid environments

Next, we considered the task of predicting amino acid preferences from the atomic environments, a problem which has been studied by several groups in the last year [4, 42]. Since physical forces are primarily a function of distance, one of the previous studies argued for the use of a concentric grid, investigated strategies for conducting convolutions on such grids, and reported substantial gains when using such convolutions over a standard 3D convolution in a regular grid (0.56 vs 0.50 accuracy) [4].

Since the classification of molecular environments involves the recognition of particular interactions between atoms (e.g. hydrogen bonds), one would expect rotational equivariant convolutions to be more suitable for the extraction of relevant features. We tested this hypothesis by constructing the exact same network as used in the original study, merely replacing the conventional convolutional layers with equivalent 3D steerable convolutional layers. Since the latter use substantially fewer parameters per channel, we chose to use the same number of *fields* as the number of channels in the original model, which still only corresponds to roughly half the number of parameters (32.6M vs 61.1M (regular grid), and 75.3M (concentric representation)). Without any alterations to the model and using the same training procedure (apart from adjustment of learning rate and regularization factor), we obtained a test accuracy of 0.58, substantially outperforming the conventional CNN on this task, and also providing an improvement over the state-of-the-art on this problem.

5.5 CATH: Protein structure classification

The molecular environments considered in the task above are oriented based on the protein backbone. Similar to standard images, this implies that the images have a natural orientation. For the final experiment, we wished to investigate the performance of our Steerable 3D convolutions on a problem domain with full rotational invariance, i.e. where the images have no inherent orientation. For this purpose, we consider the task of classifying the overall shape of protein structures.

We constructed a new data set, based on the CATH protein structure classification database [11], version 4.2 (see `http://cathdb.info/browse/tree`). The database is a classification hierarchy containing millions of experimentally determined protein domains at different levels of structural detail. For this experiment, we considered the CATH classification-level of "architecture", which splits proteins based on how protein secondary structure elements are organized in three dimensional space. Predicting the architecture from the raw protein structure thus poses a particularly challenging task for the model, which is required to not only detect the secondary structure elements at any orientation in the 3D volume, but also detect how these secondary structures orient themselves relative to one another. We limited ourselves to architectures with at least 500 proteins, which left us with 10 categories. For each of these, we balanced the data set so that all categories are represented by the same number of structures (711), also ensuring that no two proteins within the set have more than 40% sequence identity. See Supplementary Material for details. The new dataset is available at `https://github.com/wouterboomsma/cath_datasets`.

We first established a state-of-the-art baseline consisting of a conventional 3D CNN, by conducting a range of experiments with various architectures. We converged on a ResNet34-inspired architecture with half as many channels as the original, and global pooling at the end. The final model consists of $15,878,764$ parameters. For details on the experiments done to obtain the baseline, see Supplementary Material.

Following the same ResNet template, we then constructed a 3D Steerable network by replacing each layer by an equivariant version, keeping the number of 3D channels fixed. The channels are allocated such that there is an equal number of fields of order $l = 0, 1, 2, 3$ in each layer except the last, where we only used scalar fields ($l = 0$). This network contains only $143,560$ parameters, more than a factor hundred less than the baseline.

We used the first seven of the ten splits for training, the eighth for validation and the last two for testing. The data set was augmented by randomly rotating the input proteins whenever they were presented to the model during training. Note that due to their rotational equivariance, 3D Steerable CNNs benefit only marginally from rotational data augmentation compared to the baseline CNN. We train the models for 100 epochs using the Adam optimizer [26], with an exponential learning rate decay of 0.94 per epoch starting after an initial burn-in phase of 40 epochs.

Despite having 100 times fewer parameters, a comparison between the accuracy on the test set shows a clear benefit to the 3D Steerable CNN on this dataset (Figure 4, leftmost value). We proceeded with an investigation of the dependency of this performance on the size of the dataset by considering reductions of the size of each training split in the dataset by increasing powers of two, maintaining the same network architecture but re-optimizing the regularization parameters of the networks. We found that the proposed model outperforms the baseline even when trained on a fraction of the training set size. The results further demonstrate the accuracy improvements across these reductions to be robust (Figure 4).

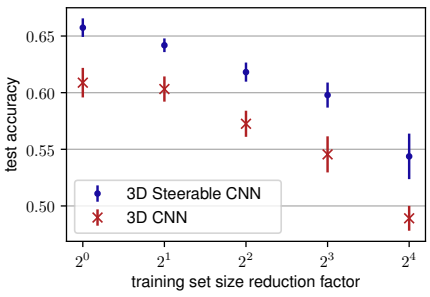


Figure 4: Accuracy on the CATH test set as a function of increasing reduction in training set size.

6 Conclusion

In this paper we have presented 3D Steerable CNNs, a class of $\mathrm{SE}(3)$-equivariant networks which represents data in terms of various kinds of fields over $\mathbb{R}^3$. We have presented a comprehensive theory of 3D Steerable CNNs, and have proven that convolutions with $\mathrm{SO}(3)$-steerable filters provide the most general way of mapping between fields in an equivariant manner, thus establishing $\mathrm{SE}(3)$-equivariant networks as a universal class of architectures. 3D Steerable CNNs require only a minor adaptation to the code of a 3D CNN, and can be converted to a conventional 3D CNN after training. Our results show that 3D Steerable CNNs are indeed equivariant, and that they show excellent accuracy and data efficiency in amino acid propensity prediction and protein structure classification.

References

[1] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*, abs/1805.12177, 2018.

[2] Song Bai, Xiang Bai, Zhichao Zhou, Zhaoxiang Zhang, and Longin Jan Latecki. Gift: A real-time and scalable 3d shape search engine. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[3] Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen AJ Eppenhof, and Josien PW Pluim. Roto-translation covariant convolutional networks for medical image analysis. *arXiv preprint arXiv:1804.03393*, 2018.

[4] Wouter Boomsma and Jes Frellsen. Spherical convolutions and their application in molecular modelling. In *Advances in Neural Information Processing Systems 30*, pages 3436–3446. 2017.

[5] Tullio Ceccherini-Silberstein, A Machì, Fabio Scarabotti, and Filippo Tolli. Induced representations and mackey theory. *Journal of Mathematical Sciences*, 156(1):11–28, 2009.

[6] Taco Cohen and Max Welling. Learning the irreducible representations of commutative lie groups. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, volume 31, pages 1755–1763, 2014.

[7] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *International Conference on Learning Representations (ICLR)*, 2018.

[8] Taco S Cohen, Mario Geiger, and Maurice Weiler. Intertwiners between induced representations (with applications to the theory of equivariant neural networks). *arXiv preprint arXiv:1803.10743*, 2018.

[9] Taco S Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, volume 48, pages 2990–2999, 2016.

[10] Taco S Cohen and Max Welling. Steerable CNNs. In *International Conference on Learning Representations (ICLR)*, 2017.

[11] Natalie L Dawson, Tony E Lewis, Sayoni Das, Jonathan G Lees, David Lee, Paul Ashford, Christine A Orengo, and Ian Sillitoe. CATH: an expanded resource to predict protein function through structure and sequence. *Nucleic acids research*, 45(D1):D289–D295, 2016.

[12] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2016.

[13] Remco Duits and Erik Franken. Left-invariant diffusions on the space of positions and orientations and their application to crossing-preserving smoothing of hardi images. *International Journal of Computer Vision*, 92(3):231–264, 2011.

[14] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. 3D object classification and retrieval with Spherical CNNs. *arXiv preprint arXiv:1711.06721*, abs/1711.06721, 2017.

[15] William T. Freeman and Edward H Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):891–906, 1991.

[16] Takahiko Furuya and Ryutarou Ohbuchi. Deep aggregation of local 3d geometric features for 3d model retrieval. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 121.1–121.12, September 2016.

[17] David Gurarie. *Symmetries and Laplacians: Introduction to Harmonic Analysis, Group Representations and Applications*. 1992.

[18] Geoffrey Hinton, Nicholas Frosst, and Sabour Sara. Matrix capsules with EM routing. In *International Conference on Learning Representations (ICLR)*, 2018.

[19] Emiel Hoogeboom, Jorn W T Peters, Taco S Cohen, and Max Welling. HexaConv. In *International Conference on Learning Representations (ICLR)*, 2018.

[20] Truong Son Hy, Shubhendu Trivedi, Horace Pan, Brandon M. Anderson, and Risi Kondor. Predicting molecular properties with covariant compositional networks. *The Journal of Chemical Physics*, 148(24):241745, 2018.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[22] Michiel HJ Janssen, Tom CJ Dela Haije, Frank C Martin, Erik J Bekkers, and Remco Duits. The hessian of axially symmetric functions on se (3) and application in 3d image analysis. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 643–655. Springer, 2017.

[23] Michiel HJ Janssen, Augustus JEM Janssen, Erik J Bekkers, Javier Oliván Bescós, and Remco Duits. Design and processing of invertible orientation scores of 3d images. *Journal of Mathematical Imaging and Vision*, pages 1–32, 2018.

[24] Kenichi Kanatani. *Group-Theoretical Methods in Image Understanding*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1990.

[25] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints, 2018.

[26] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[27] Risi Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv preprint arXiv:1803.01588*, 2018.

[28] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch–gordan nets: a fully fourier space spherical convolutional neural network. In *Neural Information Processing Systems (NIPS)*, 2018.

[29] Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. In *International Conference on Learning Representations (ICLR)*, 2018.

[30] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *arXiv preprint arXiv:1802.03690*, 2018.

[31] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *International Conference on Computer Vision (ICCV)*, 2017.

[32] Chris Olah. Groups and group convolutions. `https://colah.github.io/posts/2014-12-Groups-Convolution/`, 2014.

[33] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through parameter-sharing. *arXiv preprint arXiv:1702.08389*, 2017.

[34] Marco Reisert and Hans Burkhardt. Efficient tensor voting with 3d tensorial harmonics. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–7. IEEE, 2008.

[35] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30*, pages 3856–3866. 2017.

[36] Manolis Savva, Fisher Yu, Hao Su, Asako Kanezaki, Takahiko Furuya, Ryutarou Ohbuchi, Zhichao Zhou, Rui Yu, Song Bai, Xiang Bai, Masaki Aono, Atsushi Tatsuma, S. Thermos, A. Axenopoulos, G. Th. Papadopoulos, P. Daras, Xiao Deng, Zhouhui Lian, Bo Li, Henry Johan, Yijuan Lu, and Sanjeev Mk. Large-Scale 3D Shape Retrieval from ShapeNet Core55. In Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2017.

[37] Laurent Sifre and Stephane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[38] Eero P Simoncelli and William T Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *Image Processing, 1995. Proceedings., International Conference on*, volume 3, pages 444–447. IEEE, 1995.

[39] Henrik Skibbe. *Spherical Tensor Algebra for Biomedical Image Analysis*. PhD thesis, 2013.

[40] Atsushi Tatsuma and Masaki Aono. Multi-fourier spectra descriptor and augmentation with spectral clustering for 3d shape retrieval. *The Visual Computer*, 25(8):785–804, Aug 2009.

[41] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor Field Networks: Rotation-and Translation-Equivariant Neural Networks for 3D Point Clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[42] Wen Torng and Russ B Altman. 3D deep convolutional neural networks for amino acid environment similarity analysis. *BMC Bioinformatics*, 18(1):302, June 2017.

[43] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant CNNs. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[44] Marysia Winkels and Taco S Cohen. 3D G-CNNs for Pulmonary Nodule Detection. *arXiv preprint arXiv:1804.04656*, 2018.

[45] Daniel Worrall and Gabriel Brostow. CubeNet: Equivariance to 3D Rotation and Translation. *arXiv preprint arXiv:1804.04458*, 2018.

[46] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[47] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.

Supplementary material
3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data

1 Design choices

1.1 Feature types and multiplicities

The choice of the types and multiplicities of the features is a hyperparameter of our network comparable to the choice of channels in a conventional CNN. As in the latter we follow the logic of doubling the number of multiplicities when downsampling the feature maps. The types and multiplicities of the network's input and output are prescribed by the problem to be solved. If one uses only scalar fields, then the kernels can only be isotropic, higher order representation allows more complex kernels. A more detailed investigation of the choice of these hyperparameters is left open for future work.

1.2 Normalization

We implemented an equivariant version of batch normalization [21]. For scalar fields, our implementation matches with the usual batch normalization. For the nonscalar fields we normalize them with the average of their norms:

$$f_i(x) \mapsto f_i(x) \left(\frac{1}{|\mathcal{B}|} \sum_{j \in \mathcal{B}} \frac{1}{V} \int dx \|f_j(x)\|^2 + \epsilon \right)^{-1/2} \tag{17}$$

where $\mathcal{B}$ is the batch and i, j are the batch indices.

In order to reduce the memory consumption, we merged the batch normalization operation with the convolution

$$\kappa \star \underbrace{(Af + B)}_{BN} = (A\kappa) \star f + \kappa \star B.$$

1.3 Nonlinearities

The nonlinearities of an equivariant network need to be adapted to be equivariant themselves. Note that the domain and codomain of the nonlinearities might transform under different representations. We give an overview over the nonlinearities with which we experimented in the following paragraphs.

Elementwise nonlinearities Scalar features do not transform under rotations. As a consequence, they can be acted on by elementwise nonlinearities as in conventional CNNs. We chose ReLU nonlinearities for all scalar features except those which are used as gates (see below).

$$\begin{array}{ccc} \mathcal{F}_n^{\text{scalar}} & \xrightarrow{\quad \mathrm{Ind}_{\mathrm{SO}(3)}^{\mathrm{SE}(3)}[\mathrm{id}](\mathrm{g}) \quad} & \mathcal{F}_n^{\text{scalar}} \\ \Big\downarrow \mathrm{ReLU} & & \Big\downarrow \mathrm{ReLU} \\ \mathcal{F}_{n+1}^{\text{scalar}} & \xrightarrow[\quad \mathrm{Ind}_{\mathrm{SO}(3)}^{\mathrm{SE}(3)}[\mathrm{id}](\mathrm{g}) \quad]{} & \mathcal{F}_{n+1}^{\text{scalar}} \end{array}$$

Norm nonlinearity The representations we are considering are all orthogonal and hence preserve the norm of feature vectors:

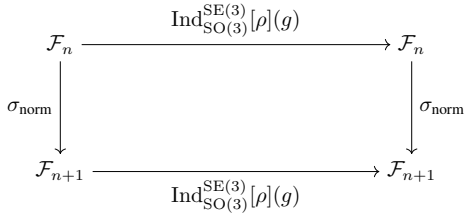
$$\|\rho(r)f(x)\| = f^T(x)\rho^T(r)\rho(r)f(x) = f^T(x)f(x) = \|f(x)\| \quad \forall r \in \mathrm{SO}(3),\ f \in \mathcal{F}$$

It follows that any nonlinearity applied to the norm of the feature commutes with the group transformation. Denoting a positive bias by $\beta \in \mathbb{R}_+$, we experimented with norm nonlinearites of the

form

$$f(x) \;\mapsto\; \sigma_{\text{norm}}(f)(x) \;:=\; \text{ReLU}\left(\|f(x)\| - \beta\right) \frac{f(x)}{\|f(x)\|}.$$

Intuitively, the bias acts as a threshold on the norm of the feature vectors, setting small vectors to zero and preserving the orientation of large feature vectors. In practice, this kind of nonlinearity tended to converge slower than the gated nonlinearities, therefore we did not use them in our final experiments. This issue might be related to the problem of finding a suitable initialization of the learned biases for which we could not derive a proper scale. Norm nonlinearities were considered before in [46].



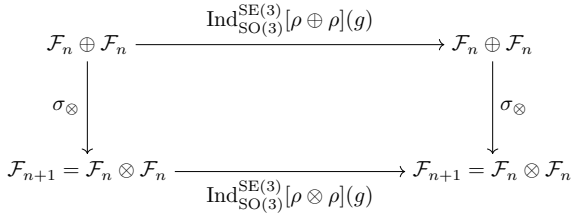
Tensor product nonlinearity The tensor product of two fields f^1 and f^2 is in index notation defined by

$$[f^1 \otimes f^2]_{\mu\nu}(x) = f^1_\mu(x) f^2_\nu(x).$$

This operation is nonlinear and equivariant and hence can be used in neural networks. We denote this nonlinearity by

$$\sigma_\otimes :\; \mathcal{F}_n \oplus \mathcal{F}_n \;\to\; \mathcal{F}_{n+1} := \mathcal{F}_n \otimes \mathcal{F}_n.$$

Note that the output of this operation transforms under the tensor product representation $\rho \otimes \rho$ of the input representations ρ. In our framework we could perform a change of basis Q defined by $Q[\rho \otimes \rho]Q^{-1} = \bigoplus_j D^j$ to obtain features transforming under irreducible representations.



Gated nonlinearity The gated nonlinearity acts on any feature vector by scaling it with a data dependent gate. We compute the gating scalars for each output feature via a sigmoid nonlinearity $\sigma : \mathcal{F}_n^{\text{scalar}} \to \mathcal{F}_n^{\text{scalar}}$ acting on an associated scalar feature. Figure 5 shows how the gated nonlinaritiy is coupled with the convolution operation. One can view the gated nonlinearity as a special case of the norm nonlinearity since it operates by changing the length of the feature vector. Simultaneously it can also be seen as a tensor product nonlinearity where one of the two fields as a scalar field. We found that the gated nonlinearities work in practice better than the the other options described above.

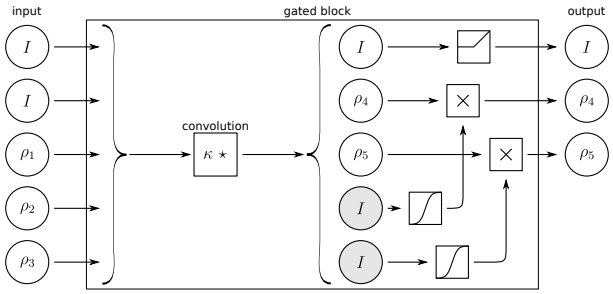


Figure 5: A gated nonlinearity requires one extra scalar field (represented by gray circles with an I) per nonscalar output fields (represented by circles with a ρ). Specifically, the number of scalar output channels for the preceding convolution operator is increased by the number of features acted on by gated nonlinearities, and the extra scalar fields are computed in the same way as any other scalar field. We use sigmoid for the gate fields. In this picture, there is one scalar field in the output. It is activated with a ReLU.

2 Reduced parameter cost of 3D Steerable CNNs

In the main paper, we demonstrated that the 3D Steerable CNN outperforms a conventional CNN despite having many fewer parameters. To ensure that the reduced number of parameters would not be an advantage also for the conventional CNN (due to overfitting with the high-capacity network), we trained a series of conventional CNNs with reduced number of filters in each layer (Figure 6). Note that the relative performance gain of our model increases dramatically if we restrict the conventional CNN to use the same number of parameters as the Steerable CNN.

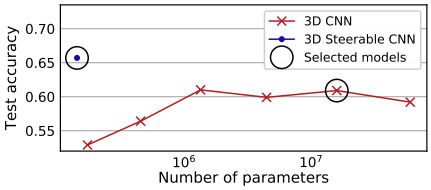


Figure 6: Performance of our 3D Steerable CNN compared to a conventional 3D CNN with varying numbers of filters.

3 The Tetris experiment

The architecture used for the Tetris experiment has 4 hidden layers, the kernel size is 5 and the padding is 4. We didn't use batch normalization. Table 1 shows the multiplicities of the fields representations and the sizes of the fields. We compare with a regular CNN that has the same feature map sizes. The CNN is like the SE3 network simply without the constraint of being equivariant for rotation. It has therefore much more parameters since its kernels are unconstrained. The SE3 network has 41k parameters and the CNN has 6M parameters.

	$l=0$	$l=1$	$l=2$	$l=3$	size	CNN features
input	1				36^3	1
layer 1	4	4	4	1	40^3	43
layer 2	16	16	16		22^3	144
layer 3	32	16	16		13^3	160
layer 4	128				17^3	128
output	8				1	8

Table 1: Architecture of the network for the Tetris experiment. Between layer 1-2 and 2-3 there is a stride of 2. Between layer 4 and the output there is a global average pooling.

low pass filter	disabled	enabled
CNN	$24\% \pm 4\%$	$27\% \pm 7\%$
SE3	$36\% \pm 6\%$	$99\% \pm 2\%$

Table 2: Test accuracy to classify rotated pieces of Tetris. Average and standard deviation over 17 runs.

4 3D Model classification

To find the model we ran 10 different models by changing depth, multiplicities, dropout, low pass filter or stride and two initialization method.

For this experiment we used a kernel size of 5 and a padding of 4. We used batch normalization. In this architecture we did't used the low pass filters. Table 3 shows the multiplicities of the fields representations and the sizes of the fields. This network has 142k parameters.

We converted the 3d models into voxels of size $64 \times 64 \times 64$ with the following code `https://github.com/mariogeiger/obj2voxel`.

Table 4 compares our results with results of the original competition and two other articles [7, 14].

	$l=0$	$l=1$	$l=2$	size
input	1			64^3
layer 1	8	4	2	34^3
layer 2	8	4	2	38^3
layer 3	16	8	4	21^3
layer 4	16	8	4	25^3
layer 5	32	16	8	15^3
layer 6	32	16	8	19^3
layer 7	32	16	8	12^3
layer 8	512			16^3
output	55			1

Table 3: Architecture of the network for the 3D Model experiment. Where the size decrease we used a stride of 2. Between the last hidden layer and the output there is a global average pooling.

	micro			macro			total		
	P@R	R@N	mAP	P@R	R@N	mAP	score	input size	params
Furuya [16]	**0.814**	0.683	0.656	**0.607**	0.539	**0.476**	**1.13**	126×10^3	8.4M
Esteves [14]	0.717	0.737	0.685	0.450	0.550	0.444	**1.13**	$\mathbf{2 \times 64^2}$	0.5M
Tatsuma [40]	0.705	**0.769**	**0.696**	0.424	**0.563**	0.418	1.11	38×224^2	3M
Ours	0.704	0.706	0.661	0.490	0.549	0.449	1.11	1×64^3	**142k**
Cohen [7]	0.701	0.711	0.676	-	-	-	-	6×128^2	1.4M
Zhou [2]	0.660	0.650	0.567	0.443	0.508	0.406	0.97	50×224^2	36M
Kanezaki [25]	0.655	0.652	0.606	0.372	0.393	0.327	0.93	-	61M
Deng [36]	0.418	0.717	0.540	0.122	0.667	0.339	0.85	-	138M

Table 4: Results of the SHREC17 experiment.

5 The CATH experiment

5.1 The data set

The protein structures used in the CATH study were simplified to include only C_α atoms (one atom per amino acid in the backbone), and placed at the center of a 50^3vx grid, where each voxel spans 0.2 nm. The values of the voxels were set to the densities arising from placing a Gaussian at each atom position, with a standard deviation of half the voxel width. Since we limit ourselves to grids of size 5 nm, we exclude proteins which expand beyond a 5 nm sphere centered around their center of mass. This constraint is only violated by a small fraction of the original dataset, and thus constitutes no severe restriction.

For training purposes, we constructed a 10-fold split of the data. To rule out any overlap between the splits (in addition to the 40% homology reduction), we further introduce a constraint that any two

members from different splits are guaranteed to originate from different categories at the "superfamily" level in the CATH hierarchy (the lowest level in the hierarchy), and all splits are guaranteed to have members from all 10 architectures. Further details about the data set are provided on the website (`https://github.com/wouterboomsma/cath_datasets`).

5.2 Establishing a state-of-the-art baseline

The baseline 3D CNN architecture for the CATH task was determined through a range of experiments, ultimately converging on a ResNet34-like architecture, with half the number of channels compared to the original implementation (but with an extra spatial dimension), and using a global pooling at the end to obtain translational invariance. After establishing the architecture, we conducted additional experiments to establish good values for the learning and drop-out rates (both in the linear and in the convolutional layers). We settled on a 0.01 dropout rate in the convolutional layers, and L1 and L2 regularization values of 10^{-7}. The final model consists of $15,878,764$ parameters.

5.3 Architecture details

Following the same ResNet template, we then constructed a 3D Steerable network, by replacing each layer with its equivariant equivalent. In contrast to the model architecture for the amino acid environment, we here opted for a minimal architecture, where we use exactly the same number of *3D channels* as in the baseline model, which leads to a model with the following block structure: $(2,2,2,2), (((2,2,2,2)\times 2)\times 3), (((4,4,4,4)\times 2)\times 4), (((8,8,8,8)\times 2)\times 6), (((16,16,16,16)\times 2)\times 2 + ((256,0,0,0))$. Here the 4-tuples represent fields of order $l = 0,1,2,3$, respectively. The final block deviates slightly from the rest, since we wish to reduce to a scalar representation prior to the pooling. Optimal regularization settings were found to be a capsule-wide convolutional dropout rate of 0.1, and L1 and L2 regularization values of $10^{-8.5}$. In this minimal setup, the model contains only $143,560$ parameters, more than a factor hundred less than the baseline.

5.3 Euclidean Neural Networks

We implemented Euclidean Neural Networks (`e3nn`), a library that unifies Cohen et al. (2018); Weiler et al. (2018); Thomas et al. (2018). In which we also added parity, an efficient implementation of spherical harmonics and an algorithm to automatically decompose geometric tensors into irreps.

The code library has been written in `PyTorch`, a python library for neural networks.[1] The library works for both point cloud and voxel data. The SE(3) equivariance has been extended to E(3) by adding the mirror symmetry giving the name of the library: Euclidean Neural Networks (`e3nn`). The library implements efficient tensor products of irreps of O(3). It's a modular framework that allows to implement a wide class of architectures including SE(3)-Transformers Fuchs et al. (2020). The library has proven its efficiency for molecular dynamics and is particularly good at predicting non scalar quantities like forces Batzner et al. (2021).

In Smidt et al. (2021), we used `e3nn` as a tool to discover broken symmetries.

5.4 A General Theory of Equivariant CNNs on Homogeneous Spaces

The following paper is the preprint version of Cohen et al. (2019).

Candidate contributions The candidate contributed with the other authors to the elaboration of the theory and to the mathematical proofs.

[1] The library is hosted on Github https://github.com/e3nn/e3nn

A General Theory of Equivariant CNNs on Homogeneous Spaces

Taco S. Cohen
Qualcomm AI Research*
Qualcomm Technologies Netherlands B.V.
tacos@qti.qualcomm.com

Mario Geiger
PCSL Research Group
EPFL
mario.geiger@epfl.ch

Maurice Weiler
QUVA Lab
U. of Amsterdam
m.weiler@uva.nl

Abstract

We present a general theory of Group equivariant Convolutional Neural Networks (G-CNNs) on homogeneous spaces such as Euclidean space and the sphere. Feature maps in these networks represent fields on a homogeneous base space, and layers are equivariant maps between spaces of fields. The theory enables a systematic classification of all existing G-CNNs in terms of their symmetry group, base space, and field type. We also consider a fundamental question: what is the most general kind of equivariant linear map between feature spaces (fields) of given types? Following Mackey, we show that such maps correspond one-to-one with convolutions using equivariant kernels, and characterize the space of such kernels.

1 Introduction

Through the use of convolution layers, Convolutional Neural Networks (CNNs) have a built-in understanding of *locality* and *translational symmetry* that is inherent in many learning problems. Because convolutions are *translation equivariant* (a shift of the input leads to a shift of the output), convolution layers preserve the translation symmetry. This is important, because it means that further layers of the network can also exploit the symmetry.

Motivated by the success of CNNs, many researchers have worked on generalizations, leading to a growing body of work on *Group equivariant CNNs* (G-CNNs) for signals on Euclidean space and the sphere [1–7] as well as graphs [8, 9]. With the proliferation of equivariant network layers, it has become difficult to see the relations between the various approaches. Furthermore, when faced with a new modality (diffusion tensor MRI, say), it may not be immediately obvious how to create an equivariant network for it, or whether a given kind of equivariant layer is the most general one.

In this paper we present a general theory of homogeneous G-CNNs. Feature spaces are modelled as spaces of fields on a homogeneous space. They are characterized by a group of symmetries G, a subgroup $H \leq G$ that together with G determines a homogeneous space $B \simeq G/H$, and a representation ρ of H that determines the type of field (vector, tensor, etc.). Related work is classified by (G, H, ρ). The main theorems say that equivariant linear maps between fields over B can be written as convolutions with an equivariant kernel, and that the space of equivariant kernels can be realized in three equivalent ways. We will assume some familiarity with groups, cosets, quotients, representations and related notions (see Appendix A).

This paper does not contain truly new mathematics (in the sense that a professional mathematician with expertise in the relevant subjects would not be surprised by our results), but instead provides a new formalism for the study of equivariant convolutional networks. This formalism turns out to be a remarkably good fit for describing real-world G-CNNs. Moreover, by describing G-CNNs in a language used throughout modern physics and mathematics (fields, fiber bundles, etc.), it becomes possible to apply knowledge gained over many decades in those domains to machine learning.

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

1.1 Overview of the Theory

This paper has two main parts. First, in Sec. 2, we introduce a mathematical model for convolutional feature spaces. The basic idea is that feature maps represent fields over a homogeneous space. As it turns out, defining the notion of a field is quite a bit of work. So in order to motivate the introduction of each of the required concepts, we will in this section provide an overview of the relevant concepts and their relations, using the example of a Spherical CNN with vector field feature maps.

The second part of this paper (Section 3) is about maps between the feature spaces. We require these to be equivariant, and focus in particular on the linear layers. The main theorems (3.1–3.4) show that linear equivariant maps between the feature spaces are in one-to-one correspondence with equivariant convolution kernels (i.e. *convolution is all you need*), and that the space of equivariant kernels can be realized as a space of matrix-valued functions on a group, coset space, or double coset space, subject to linear constraints.

In order to specify a convolutional feature space, we need to specify two things: a homogeneous space B over which the field is defined, and the type of field (e.g. vector field, tensor field, etc.). A homogeneous space for a group G is a space B where for any two $x, y \in B$ there is a transformation $g \in G$ that relates them via $gx = y$. Here we consider the example of a vector field on the sphere $B = S^2$ with symmetry group $G = \mathrm{SO}(3)$, the group of 3D rotations. The sphere is a homogeneous space for $\mathrm{SO}(3)$ because we can map any point on the sphere to any other via a rotation.

Formally, a field is defined as a *section* of a *vector bundle associated to a principal bundle*. In order to understand what this means, we must first know what a *fiber bundle* is (Sec. 2.1), and understand how the group G can be viewed as a *principal bundle* (Sec. 2.2). Briefly, a fiber bundle formalizes the idea of parameterizing a set of identical spaces called *fibers* by another space called the *base space*.

The first way in which fiber bundles play a role in the theory is that the action of G on B allows us to think of G as a "bundle of groups" or *principal bundle*. Roughly speaking, this works as follows: if we fix an origin $o \in B$, we can consider the *stabilizer subgroup* $H \leq G$ of transformations that leave o unchanged: $H = \{g \in G \mid go = o\}$. For example, on the sphere the stabilizer is $\mathrm{SO}(2)$, the group of rotations around the axis through o (e.g. the north pole). As we will see in Section 2.2, this allows us to view G as a bundle with *base space* $B \simeq G/H$ and a *fiber* H. This is shown for the sphere in Fig. 1 (cartoon). In this case, we can think of $\mathrm{SO}(3)$ as a bundle of circles ($H = \mathrm{SO}(2)$) over the sphere, which itself is the quotient $S^2 \simeq \mathrm{SO}(3)/\mathrm{SO}(2)$.

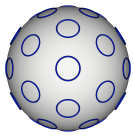


Figure 1: $\mathrm{SO}(3)$ as a principal $\mathrm{SO}(2)$ bundle over S^2.

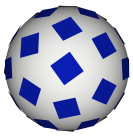


Figure 2: Tangent bundle of S^2.

To define the *associated bundle* (Sec. 2.3) we take the principal bundle G and replace the fiber H by a vector space V on which H acts linearly via a *group representation* ρ. This yields a vector bundle with the same base space B and a new fiber V. For example, the tangent bundle of S^2 (Fig. 2) is obtained by replacing the circular $\mathrm{SO}(2)$ fibers in Fig. 1 by 2D planes. Under the action of $H = \mathrm{SO}(2)$, a tangent vector at the north pole is rotated (even though the north pole itself is fixed by $\mathrm{SO}(2)$), so we let $\rho(h)$ be a 2×2 rotation matrix. In a general convolutional feature space with n channels, V would be an n-dimensional vector space. Finally, fields are defined as *sections* of this bundle, i.e. an assignment to each point x of an element in the fiber over x (see Fig. 3).

Having defined the feature space, we need to specify how it transforms (e.g. say how a vector field on S^2 is rotated). The natural way to transform a ρ-field is via the *induced representation* $\pi = \mathrm{Ind}_H^G \rho$ of G (Section 2.4), which combines the action of G on the base space B and the action of ρ on the fiber V to produce an action on sections of the associated bundle (See Figure 3). Finally, having defined the feature spaces and their transformation laws, we can study equivariant linear maps between them (Section 3). In Sec. 4–6 we cover implementation aspects, related work, and concrete examples, respectively.

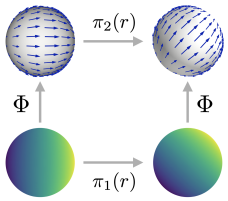


Figure 3: Φ maps scalar fields to vector fields, and is equivariant to the induced representation $\pi_i = \mathrm{Ind}_{\mathrm{SO}(2)}^{\mathrm{SO}(3)} \rho_i$.

2 Convolutional Feature Spaces

2.1 Fiber Bundles

Intuitively, a fiber bundle is a parameterization of a set of isomorphic spaces (the fibers) by another space (the base). For example, we can think of a feature space in a classical CNN as a set of vector spaces $V_x \simeq \mathbb{R}^n$ (n being the number of channels), one per position x in the plane [2]. This is an example of a *trivial bundle*, because it is simply the Cartesian product of the plane and $\mathbb{R}^n$. General fiber bundles are only *locally trivial*, meaning that they locally look like a product while having a different global topological structure.

The simplest example of a non-trivial bundle is the Mobius strip, which locally looks like a product of the circle (the base) with a line segment (the fiber), but is globally distinct from a cylinder (see Fig. 4). A more practically relevant example is given by the tangent bundle of the sphere (Fig. 2), which has as base space S^2 and fibers that look like $\mathbb{R}^2$, but is topologically distinct from $S^2 \times \mathbb{R}^2$ as a bundle.

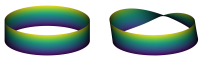


Figure 4: Cylinder and Möbius strip

Formally, a bundle consists of topological spaces E (total space), B (base space), F (canonical fiber), and a projection map $p : E \to B$, satisfying a local triviality condition. Basically, this condition says that locally, the bundle looks like a product $U \times F$ of a piece $U \subseteq B$ of the base space, and F the canonical fiber. Formally, the condition is that for every $a \in E$, there is an open neighbourhood $U \subseteq B$ of $p(a)$ and a homeomorphism $\varphi : p^{-1}(U) \to U \times F$ so that the map $p^{-1}(U) \xrightarrow{\varphi} U \times F \xrightarrow{\mathrm{proj}_1} U$ agrees with $p : p^{-1}(U) \to U$ (where $\mathrm{proj}_1(u, f) = u$). The homeomorphism φ is said to locally trivialize the bundle above the trivializing neighbourhood U.

Considering that for any $x \in U$ the preimage ${\mathrm{proj}_1}^{-1}(x)$ is F, and φ is a homeomorphism, we see that the preimage $F_x = p^{-1}(x)$ for $x \in B$ is also homeomorphic to F. Thus, we call F_x the fiber over x, and see that all fibers are homeomorphic. Knowing this, we can denote a bundle by its projection map $p : E \to B$, leaving the canonical fiber F implicit.

Various more refined notions of fiber bundle exist, each corresponding to a different kind of fiber. In this paper we will work with principal bundles (bundles of groups) and vector bundles (bundles of vector spaces).

A *section* s of a fiber bundle is an assignment to each $x \in B$ of an element $s(x) \in F_x$. Formally, it is a map $s : B \to E$ that satisfies $p \circ s = \mathrm{id}_B$. If the bundle is trivial, a section is equivalent to a function $f : B \to F$, but for a non-trivial bundle we cannot continuously align all the fibers simultaneously, and so we must keep each $s(x)$ in its own fiber F_x. Nevertheless, on a trivializing neighbourhood $U \subseteq B$, we can describe the section as a function $s_U : U \to F$, by setting $\varphi(s(x)) = (x, s_U(x))$.

2.2 G as a Principal H-Bundle

Recall (Sec. 1.1) that with every feature space of a G-CNN is associated a homogeneous space B (e.g. the sphere, projective space, hyperbolic space, Grassmann & Stiefel manifolds, etc.), and recall further that such a space has a stabilizer subgroup $H = \{g \in G \mid go = o\}$ (this group being independent of origin o up to isomorphism). As discussed in Appendix A, the cosets gH of H (e.g. the circles in Fig. 1) partition G, and the set of cosets, denoted G/H (e.g. the sphere in Fig. 1), can be identified with B (up to a choice of origin).

It is this partitioning of G into cosets that induces a special kind of bundle structure on G. The projection map that defines the bundle structure sends an element $g \in G$ to the coset gH it belongs to. Thus, it is a map $p : G \to G/H$, and we have a bundle with total space G, base space G/H and canonical fiber H. Intuitively, this allows us to think of G as a base space G/H with a copy of H attached at each point $x \in G/H$. The copies of H are glued together in a potentially twisted manner.

This bundle is called a *principal H-bundle*, because we have a transitive and fixed-point free group action $G \times H \to G$ that preserves the fibers. This action is given by right multiplication, $g \mapsto gh$, which preserves fibers because $p(gh) = ghH = gH = p(g)$. That is, by right-multiplying an element $g \in G$ by $h \in H$, we get an element gh that is in general different from g but is still within the same coset (i.e. fiber). That the action is transitive and free on cosets follows immediately from the group axioms.

One can think of a principal bundle as a bundle of generalized frames or *gauges* relative to which geometrical quantities can be expressed numerically. Under this interpretation the fiber at x is a space of generalized frames, and the action by H is a change of frame. For instance, each point on the circles in Fig. 1 can be identified with a right-handed orthogonal frame, and the action of $\mathrm{SO}(2)$ corresponds to a rotation of this frame. The group H may also include internal symmetries, such as color space rotations, which do not relate in any way to the spatial dimensions of B.

In order to numerically represent a field on some neighbourhood $U \subseteq G/H$, we need to choose a frame for each $x \in U$ in a continuous manner. This is formalized as a section of the principal bundle. Recall that a section of $p : G \to G/H$ is a map $s : G/H \to G$ that satisfies $p \circ s = \mathrm{id}_{G/H}$. Since p projects g to its coset gH, the section chooses a representative $s(gH) \in gH$ for each coset gH. Non-trivial principal bundles do not have continuous global sections, but we can always use a local section on $U \subseteq G/H$, and represent a field on overlapping local patches covering G/H.

Aside from the right action of H, which turns G into a principal H-bundle, we also have a left action of G on itself, as well as an action of G on the base space G/H. In general, the action of G on G/H does not agree with the action on G, in that $gs(x) \neq s(gx)$, because the action on G includes a twist of the fiber. This twist is described by the function $\mathrm{h} : G/H \times G \to H$ defined by $gs(x) = s(gx)\mathrm{h}(x, g)$ (whenever both $s(x)$ and $s(gx)$ are defined). This function will be used in various calculations below. We note for the interested reader that h satisfies the cocycle condition $\mathrm{h}(x, g_1 g_2) = \mathrm{h}(g_2 x, g_1)\mathrm{h}(x, g_2)$.

2.3 The Associated Vector Bundle

Feature spaces are defined as spaces of sections of the associated vector bundle, which we will now define. In physics, a section of an associated bundle is simply called a field.

To define the associated vector bundle, we start with the principal H-bundle $G \xrightarrow{p} G/H$, and essentially replace the fibers (cosets) by vector spaces V. The space $V \simeq \mathbb{R}^n$ carries a group representation ρ of H that describes the transformation behaviour of the feature vectors in V under a change of frame. These features could for instance transform as a scalar, a vector, a tensor, or some other geometrical quantity [2, 6, 8]. Figure 3 shows an example of a vector field ($\rho(h)$ being a 2×2 rotation matrix in this case) and a scalar field ($\rho(h) = 1$).

The first step in constructing the associated vector bundle is to take the product $G \times V$. In the context of representation learning, we can think of an element (g, v) of $G \times V$ as a feature vector $v \in V$ and an associated pose variable $g \in G$ that describes how the feature detector was steered to obtain v. For instance, in a Spherical CNN [10] one would rotate a filter bank by $g \in \mathrm{SO}(3)$ and match it with the input to obtain v. If we apply a transformation $h \in H$ to g and simultaneously apply its inverse to v, we get an equivalent element $(gh, \rho(h^{-1})v)$. In a Spherical CNN, this would correspond to a change in orientation of the filters by $h \in \mathrm{SO}(2)$.

So in order to create the associated bundle, we take the quotient of the product $G \times V$ by this action: $A = G \times_\rho V = (G \times V)/H$. In other words, the elements of A are orbits, defined as $[g, v] = \{(gh, \rho(h^{-1})v) \mid h \in H\}$. The projection $p_A : A \to G/H$ is defined as $p_A([g, v]) = gH$. One may check that this is well defined, i.e. independent of the orbit representative g of $[g, v] = [gh, \rho(h^{-1})v]$. Thus, the associated bundle has base G/H and fiber V, meaning that locally it looks like $G/H \times V$. We note that the associated bundle construction works for any principal H-bundle, nog just $p : G \to G/H$, which suggests a direction for further generalization [11].

A field ("stack of feature maps") is a section of the associated bundle, meaning that it is a map $s : G/H \to A$ such that $\pi_\rho \circ s = \mathrm{id}_{G/H}$. We will refer to the space of sections of the associated vector bundle as $\mathcal{I}$. Concretely, we have two ways to encode a section: as functions $f : G \to V$ subject to a constraint, and as local functions from $U \subseteq G/H$ to V. We will now define both.

2.3.1 Sections as Mackey Functions

The construction of the associated bundle as a product $G \times V$ subject to an equivalence relation suggests a way to describe sections concretely: a section can be represented by a function $f : G \to V$ subject to the equivariance condition

$$f(gh) = \rho(h^{-1})f(g). \tag{1}$$

Such functions are called Mackey functions. They provide a redundant encoding of a section of A, by encoding the value of the section relative to any choice of frame / section of the principal bundle simultaneously, with the equivariance constraint ensuring consistency.

A linear combination of Mackey functions is a Mackey function, so they form a vector space, which we will refer to as $\mathcal{I}_G$. Mackey functions are easy to work with because they allow a concrete and global description of a field, but their redundancy makes them unsuitable for computer implementation.

2.3.2 Local Sections as Functions on G/H

The associated bundle has base G/H and fiber V, so locally, we can describe a section as an unconstrained function $f : U \to V$ where $U \subseteq G/H$ is a trivializing neighbourhood (see Sec. 2.1). We refer to the space of such sections as $\mathcal{I}_C$. Given a local section $f \in \mathcal{I}_C$, we can encode it as a Mackey function through the following lifting isomorphism $\Lambda : \mathcal{I}_C \to \mathcal{I}_G$:

$$\begin{aligned}[\Lambda f](g) &= \rho(\mathrm{h}(g)^{-1}) f(gH), \\ [\Lambda^{-1} f'](x) &= f'(s(x)),\end{aligned} \tag{2}$$

where $\mathrm{h}(g) = \mathrm{h}(H, g) = s(gH)^{-1} g \in H$ and $s(x) \in G$ is a coset representative for $x \in G/H$. This map is analogous to the lifting defined by [12] for scalar fields (i.e. $\rho(h) = I$), and can be defined more generally for any principal / associated bundle [13].

2.4 The Induced Representation

The induced representation $\pi = \mathrm{Ind}_H^G \rho$ describes the action of G on fields. In $\mathcal{I}_G$, it is defined as:

$$[\pi_G(g) f](k) = f(g^{-1} k). \tag{3}$$

In $\mathcal{I}_C$, we can define the induced representation π_C on a local neighbourhood U as

$$[\pi_C(g) f](x) = \rho(\mathrm{h}(g^{-1}, x)^{-1}) f(g^{-1} x). \tag{4}$$

Here we have assumed that h is defined at (g^{-1}, x). If it is not, one would need to change to a different section of $G \to G/H$. One may verify, using the composition law for h (Sec. 2.2), that Eq. 4 does indeed define a representation of G. Moreover, one may verify that $\pi_G(g) \circ \Lambda = \Lambda \circ \pi_C(g)$, i.e. they define isomorphic representations.

We can interpret Eq. 4 as follows. To transform a field, we move the fiber at $g^{-1}x$ to x, and we apply a transformation to the fiber itself using ρ. This is visualized in Fig. 5 for a planar vector field. Some other examples include an RGB image ($\rho(h) = I_3$), a field of wind directions on earth ($\rho(h)$ a 2×2 rotation matrix), a diffusion tensor MRI image ($\rho(h)$ a representation of SO(3) acting on 2-tensors), a regular G-CNN on $\mathbb{Z}^3$ [14, 15] (ρ a regular representation of H).

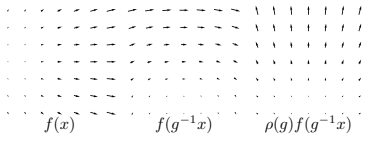


Figure 5: The rotation of a planar vector field in two steps: moving each vector to its new position without changing its orientation, and then rotating the vectors.

3 Equivariant Maps and Convolutions

Each feature space in a G-CNN is defined as the space of sections of some associated vector bundle, defined by a choice of base G/H and representation ρ of H that describes how the fibers transform. A layer in a G-CNN is a map between these feature spaces that is equivariant to the induced representations acting on them. In this section we will show that equivariant *linear* maps can always be written as a convolution-like operation using an equivariant kernel. We will first derive this result for the induced representation realized in the space $\mathcal{I}_G$ of Mackey functions, and then convert the result to local sections of the associated vector bundle in Section 3.2. We will assume that G is locally compact and unimodular.

Consider adjacent feature spaces $i = 1, 2$ with a representation (ρ_i, V_i) of $H_i \leq G$. Let $\pi_i = \mathrm{Ind}_{H_i}^G \rho_i$ be the representation acting on $\mathcal{I}_G^i$. A bounded linear operator $\mathcal{I}_G^1 \to \mathcal{I}_G^2$ can be written as

$$[\kappa \cdot f](g) = \int_G \kappa(g,\, g') f(g') dg', \tag{5}$$

using a two-argument linear operator-valued kernel $\kappa : G \times G \to \mathrm{Hom}(V_1, V_2)$, where $\mathrm{Hom}(V_1, V_2)$ denotes the space of linear maps $V_1 \to V_2$. Choosing bases, we get a matrix-valued kernel.

We are interested in the space of equivariant linear maps between induced representations, defined as $\mathcal{H} = \mathrm{Hom}_G(\mathcal{I}^1, \mathcal{I}^2) = \{\Phi \in \mathrm{Hom}(\mathcal{I}^1, \mathcal{I}^2) \,|\, \Phi\pi_1(g) = \pi_2(g)\Phi,\ \forall g \in G\}$. In order for Eq. 5 to define an equivariant map $\Phi \in \mathcal{H}$, the kernel κ must satisfy a constraint. By (partially) resolving this constraint, we will show that Eq. 5 can always be written as a cross-correlation[1]

Theorem 3.1. *(convolution is all you need) An equivariant map $\Phi \in \mathcal{H}$ can always be written as a convolution-like integral.*

Proof. Since we are only interested in equivariant maps, we get a constraint on κ. For all $u, g \in G$:

$$\begin{aligned} & [\kappa \cdot [\pi_1(u)f]](g) &&= [\pi_2(u)[\kappa \cdot f]](g) \\ \Leftrightarrow \quad & \int_G \kappa(g, g')f(u^{-1}g')dg' &&= \int_G \kappa(u^{-1}g, g')f(g')dg' \\ \Leftrightarrow \quad & \int_G \kappa(g, ug')f(g')dg' &&= \int_G \kappa(u^{-1}g, g')f(g')dg' \\ \Leftrightarrow \quad & \kappa(g, ug') &&= \kappa(u^{-1}g, g') \\ \Leftrightarrow \quad & \kappa(ug, ug') &&= \kappa(g, g') \end{aligned} \tag{6}$$

Hence, without loss of generality, we can define the two-argument kernel $\kappa(\cdot, \cdot)$ in terms of a one-argument kernel: $\kappa(g^{-1}g') \equiv \kappa(e, g^{-1}g') = \kappa(ge, gg^{-1}g') = \kappa(g, g')$.

The application of κ to f thus reduces to a cross-correlation:

$$[\kappa \cdot f](g) = \int_G \kappa(g, g')f(g')dg' = \int_G \kappa(g^{-1}g')f(g')dg' = [\kappa \star f](g). \tag{7}$$

□

3.1 The Space of Equivariant Kernels

The constraint Eq. 6 implies a constraint on the one-argument kernel κ. The space of admissible kernels is in one-to-one correspondence with the space of equivariant maps. Here we give three different characterizations of this space of kernels. Detailed proofs can be found in Appendix B.

Theorem 3.2. *$\mathcal{H}$ is isomorphic to the space of bi-equivariant kernels on G, defined as:*

$$\begin{aligned} \mathcal{K}_G = \{\kappa : G \to \mathrm{Hom}(V_1, V_2) \,|\, & \kappa(h_2 g h_1) = \rho_2(h_2)\kappa(g)\rho_1(h_1), \\ & \forall g \in G, h_1 \in H_1, h_2 \in H_2\}. \end{aligned} \tag{8}$$

Proof. It is easily verified (see supp. mat.) that right equivariance follows from the fact that $f \in \mathcal{I}^1_G$ is a Mackey function, and left equivariance follows from the requirement that $\kappa \star f \in \mathcal{I}^2_G$ should be a Mackey function. The isomorphism is given by $\Gamma_G : \mathcal{K}_G \to \mathcal{H}$ defined as $[\Gamma_G\kappa]f = \kappa \star f$. □

The analogous result for the two argument kernel is that $\kappa(gh_2, g'h_1)$ should be equal to $\rho_2(h_2^{-1})\kappa(g, g')\rho_1(h_1)$ for $g, g' \in G, h_1 \in H_1, h_2 \in H_2$. This has the following interesting interpretation: κ is a section of a certain associated bundle. We define a right-action of $H_1 \times H_2$ on $G \times G$ by setting $(g, g') \cdot (h_1, h_2) = (gh_1, g'h_2)$ and a representation ρ_{12} of $H_1 \times H_2$ on $\mathrm{Hom}(V_1, V_2)$ by setting $\rho_{12}(h_1, h_2)\Psi = \rho_2(h_2)\Psi\rho_1(h_1^{-1})$ for $\Psi \in \mathrm{Hom}(V_1, V_2)$. Then the constraint on $\kappa(\cdot, \cdot)$ can be written as $\kappa((g, g') \cdot (h_1, h_2)) = \rho_{12}((h_1, h_2)^{-1})\kappa((g, g'))$. We recognize this as the condition of being a Mackey function (Eq. 1) for the bundle $(G \times G) \times_{\rho_{12}} \mathrm{Hom}(V_1, V_2)$.

There is another another way to characterize the space of equivariant kernels:

Theorem 3.3. *$\mathcal{H}$ is isomorphic to the space of left-equivariant kernels on G/H_1, defined as:*

$$\begin{aligned} \mathcal{K}_C = \{\overleftarrow{\kappa} : G/H_1 \to \mathrm{Hom}(V_1, V_2) \,|\, & \overleftarrow{\kappa}(h_2 x) = \rho_2(h_2)\overleftarrow{\kappa}(x)\rho_1(\mathrm{h}_1(x, h_2)^{-1}), \\ & \forall h_2 \in H_2, x \in G/H_1\} \end{aligned} \tag{9}$$

[1] As in most of the CNN literature, we will not be precise about distinguishing convolution and correlation.

Proof. using the decomposition $g = s(gH_1)\mathrm{h}_1(g)$ (see Appendix A), we can define

$$\kappa(g) = \kappa(s(gH_1)\mathrm{h}_1(g)) = \kappa(s(gH_1))\,\rho_1(\mathrm{h}_1(g)) \equiv \overleftarrow{\kappa}(gH_1)\rho_1(\mathrm{h}_1(g)), \tag{10}$$

This defines the lifting isomorphism for kernels, $\Lambda_{\mathcal{K}} : \mathcal{K}_C \to \mathcal{K}_G$. It is easy to verify that when defined in this way, κ satisfies right H_1-equivariance.

We still have the left H_2-equivariance constraint from Eq. 8, which translates to $\overleftarrow{\kappa}$ as follows (details in supp. mat.). For $g \in G$, $h_2 \in H_2$ and $x \in G/H_1$,

$$\kappa(h_2 g) = \rho_2(h_2)\kappa(g) \Leftrightarrow \overleftarrow{\kappa}(h_2 x) = \rho_2(h_2)\overleftarrow{\kappa}(x)\rho_1(\mathrm{h}_1(x, h_2)^{-1}). \tag{11}$$

□

Theorem 3.4. *$\mathcal{H}$ is isomorphic to the space of $H_2^{\gamma(x)H_1}$-equivariant kernels on $H_2\backslash G/H_1$:*

$$\begin{aligned}\mathcal{K}_D = \{\bar{\kappa} : H_2\backslash G/H_1 \to \mathrm{Hom}(V_1, V_2) \,|\, \bar{\kappa}(x) = \rho_2(h)\bar{\kappa}(x)\rho_1^x(h)^{-1}, \\ \forall x \in H_2\backslash G/H_1, h \in H_2^{\gamma(x)H_1}\},\end{aligned} \tag{12}$$

Where $\gamma : H_2\backslash G/H_1 \to G$ is a choice of double coset representatives, and ρ_1^x is a representation of the stabilizer $H_2^{\gamma(x)H_1} = \{h \in H_2 \,|\, h\gamma(x)H_1 = \gamma(x)H_1\} \leq H_1$, defined as

$$\rho_1^x(h) = \rho_1(\mathrm{h}_1(\gamma(x)H_1, h)) = \rho_1(\gamma(x)^{-1}h\gamma(x)), \tag{13}$$

Proof. In supplementary material. For examples, see Section 6. □

3.2 Local Sections on G/H

We have seen that an equivariant map between spaces of Mackey functions can always be realized as a cross-correlation on G, and we have studied the properties of the kernel, which can be encoded as a kernel on G or G/H_1 or $H_2\backslash G/H_1$, subject to the appropriate constraints. When implementing a G-CNN, it would be wasteful to use a Mackey function on G, so we need to understand what it means for fields realized by local functions $f : U \to V$ for $U \subseteq G/H_1$. This is done by sandwiching the cross-correlation $\kappa\star : \mathcal{I}_G^1 \to \mathcal{I}_G^2$ with the lifting isomorphisms $\Lambda_i : \mathcal{I}_C^i \to \mathcal{I}_G^i$.

$$\begin{aligned}[\Lambda_2^{-1}[\kappa \star [\Lambda_1 f]]](x) &= \int_G \kappa(s_2(x)^{-1}s_1(y))f(y)dy \\ &= \int_{G/H_1} \overleftarrow{\kappa}(s_2(x)^{-1}y)\rho_1(\mathrm{h}_1(s_2(x)^{-1}s_1(y)))f(y)dy\end{aligned} \tag{14}$$

Which we refer to as the ρ_1-twisted cross-correlation on G/H_1. We note that for semidirect product groups, the ρ_1 factor disappears and we are left with a standard cross-correlation on G/H_1 with an equivariant kernel $\overleftarrow{\kappa} \in \mathcal{K}_C$. We note the similarity of this expression to gauge equivariant convolution as defined in [11].

3.3 Equivariant Nonlinearities

The network as a whole is equivariant if all of its layers are equivariant. So our theory would not be complete without a discussion of equivariant nonlinearities and other kinds of layers. In a regular G-CNN [1], ρ is the regular representation of H, which means that it can be realized by permutation matrices. Since permutations and pointwise nonlinearities commute, any such nonlinearity can be used. For other kinds of representations ρ, special equivariant nonlinearities must be used. Some choices include norm nonlinearities [3] for unitary representations, tensor product nonlinearities [8], or gated nonlinearities where a scalar field is normalized by a sigmoid and then multiplied by another field [6]. Other constructions, such as batchnorm and ResNets, can also be made equivariant [1, 2]. A comprehensive overview and comparison over equivariant nonlinearities can be found in [7].

4 Implementation

Several different approaches to implementing group equivariant CNNs have been proposed in the literature. The implementation details thereby depend on the specific choice of symmetry group G, the homogeneous space G/H, its discretization and the representation ρ. In any case, since the equivariance constraints on convolution kernels are linear, the space of H-equivariant kernels is a linear subspace of the unrestricted kernel space. This implies that it is sufficient to solve for a basis of H-equivariant kernels, in terms of which any equivariant kernel can be expanded using learned weights.

A case of high practical importance are equivariant CNNs on Euclidean spaces $\mathbb{R}^d$. Implementations mostly operate on discrete pixel grids. In this case, the steerable kernel basis is typically pre-sampled on a small grid, linearly combined during the forward pass, and then used in a standard convolution routine. The sampling procedure requires particular attention since it might introduce aliasing artifacts [4, 6]. A more in depth discussion of an implementation of equivariant CNNs, operating on Euclidean pixel grids, is provided in [7]. Alternatively to processing signals on a pixel grid, signals on Euclidean spaces might be sampled on an irregular point cloud. In this case the steerable kernel space is typically implemented as an analytical function, which is subsequently sampled on the cloud [5].

Implementations of spherical CNNs depend on the choice of signal representation as well. In [10], the authors choose a spectral approach to represent the signal and kernels in Fourier space. The equivariant convolution is performed by exploiting the Fourier theorem. Other approaches define the convolution spatially. In these cases, some grid on the sphere is chosen on which the signal is sampled. As in the Euclidean case, the convolution is performed by matching the signal with a H-equivariant kernel, which is being expanded in terms of a pre-computed basis.

5 Related Work

In Appendix D, we provide a systematic classification of equivariant CNNs on homogeneous spaces, according to the theory presented in this paper. Besides these references, several papers deserve special mention. Most closely related is the work of [12], whose theory is analogous to ours, but only covers scalar fields (corresponding to using a trivial representation $\rho(h) = I$ in our theory). A proper treatment of general fields as we do here is more difficult, as it requires the use of fiber bundles and induced representations. The first use of induced representations and fields in CNNs is [2], and the first CNN on a non-trivial homogeneous space (the Sphere) is [16].

A framework for (non-convolutional) networks equivariant to finite groups was presented by [17], and equivariant set and graph networks are analyzed by [18–21]. Our use of fields (with ρ block-diagonal) can be viewed as a formalization of convolutional capsules [22, 23]. Other related work includes [24–31]. A preliminary version of this paper appeared as [32].

For mathematical background, we recommend [13, 33–37]. The study of induced representations and equivariant maps between them was pioneered by Mackey [38–41], who rigorously proved results essentially similar to the ones in this paper, though presented in a more abstract form that may not be easy to recognize as having relevance to the theory of equivariant CNNs.

6 Concrete Examples

6.1 The rotation group $\mathrm{SO}(3)$ and spherical CNNs

The group of 3D rotations $\mathrm{SO}(3)$ is a three-dimensional manifold that can be parameterized by ZYZ Euler angles $\alpha \in [0, 2\pi)$, $\beta \in [0, \pi]$ and $\gamma \in [0, 2\pi)$, i.e. $g = Z(\alpha)Y(\beta)Z(\gamma)$, (where Z and Y denote rotations around the Z and Y axes). For this example we choose $H = H_1 = H_2 = \mathrm{SO}(2) = \{Z(\alpha) \,|\, \alpha \in [0, 2\pi)\}$ as the group of rotations around the Z-axis, i.e. the stabilizer subgroup of the north pole of the sphere. A left H-coset is then a subset of $\mathrm{SO}(3)$ of the form

$$gH = \{Z(\alpha)Y(\beta)Z(\gamma)Z(\alpha') \,|\, \alpha' \in [0, 2\pi)\} = \{Z(\alpha)Y(\beta)Z(\alpha') \,|\, \alpha' \in [0, 2\pi)\}.$$

Thus, the coset space G/H is the sphere S^2, parameterized by spherical coordinates α and β. As expected, the stabilizer H_x of a point $x \in S^2$ is the set of rotations around the axis through x, which is isomorphic to $H = \mathrm{SO}(2)$.

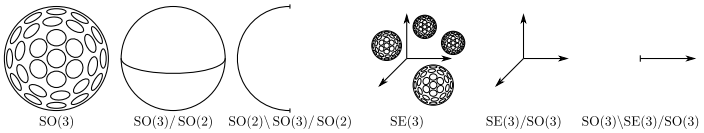


Figure 6: Quotients of $\mathrm{SO}(3)$ and $\mathrm{SE}(3)$.

What about the double coset space (Appendix A.1)? The orbit of a point $x(\alpha, \beta) \in S^2$ under H is a circle around the Z axis at lattitude β, so the double coset space $H\backslash G/H$, which indexes these orbits, is the segment $[0, \pi)$ (see Fig. 6).

The section $s : G/H \to G$ may be defined (almost everywhere) as $s(\alpha, \beta) = Z(\alpha)Y(\beta) \in \mathrm{SO}(3)$, and $\gamma(\beta) = Y(\beta) \in \mathrm{SO}(3)$. Then the stabilizer $H_2^{\gamma(\beta)H_1}$ for $\beta \in H\backslash G/H$ is the set of Z-axis rotations that leave the point $\gamma(\beta)H_1 = (0, \beta) \in S^2$ invariant. For the north and south pole ($\beta = 0$ or $\beta = \pi$), this stabilizer is all of $H = \mathrm{SO}(2)$, but for other points it is the trivial subgroup $\{e\}$.

Thus, according to Theorem 3.4, the equivariant kernels are matrix-valued functions on the segment $[0, \pi)$, that are mostly unconstrained (except at the poles). As functions on G/H_1 (Theorem 3.3), they are matrix-valued functions satisfying $\overleftarrow{\kappa}(rx) = \rho_2(r)\overleftarrow{\kappa}(x)\rho_1(\mathrm{h}_1(x, r)^{-1})$ for $r \in \mathrm{SO}(2)$ and $x \in S^2$. This says that as a function on the sphere $\overleftarrow{\kappa}$ is determined on $\mathrm{SO}(2)$-orbits $\{rx \mid r \in \mathrm{SO}(2)\}$ (lattitudinal circles around the Z axis) by its value on one point of the orbit. Indeed, if $\rho(h) = 1$ is the trivial representation, we see that $\overleftarrow{\kappa}$ is constant on these orbits, in agreement with [42] who use isotropic filters. For ρ_2 a regular representation of $\mathrm{SO}(2)$, we recover the non-isotropic method of [10]. For segmentation tasks, one can use a trivial representation for ρ_2 in the output layer to obtain a scalar feature map on S^2, analogous to [43]. Other choices, such as ρ the standard 2D representation of $\mathrm{SO}(2)$, would make it possible to build spherical CNNs that can process vector fields, but this has not been done yet.

6.2 The roto-translation group $\mathrm{SE}(3)$ and 3D Steerable CNNs

The group of rigid body motions $\mathrm{SE}(3)$ is a 6D manifold $\mathbb{R}^3 \rtimes \mathrm{SO}(3)$. We choose $H = H_1 = H_2 = \mathrm{SO}(3)$ (rotations around the origin). A left H-coset is a set of the form $gH = trH = \{trr' \,|\, r' \in \mathrm{SO}(3)\} = \{tr \,|\, r \in \mathrm{SO}(3)\}$ where t is the translation component of g. Thus, the coset space G/H is $\mathbb{R}^3$. The stabilizer H_x of a point $x \in \mathbb{R}^3$ is the set of rotations around x, which is isomorphic to $\mathrm{SO}(3)$. The orbit of a point $x \in \mathbb{R}^3$ is a spherical shell of radius $\|x\|$, so the double coset space $H\backslash G/H$, which indexes these orbits, is the set of radii $[0, \infty)$.

Since $\mathrm{SE}(3)$ is a trivial principal $\mathrm{SO}(3)$ bundle, we can choose a global section $s : G/H \to G$ by taking $s(x)$ to be the translation by x. As double coset representatives we can choose $\gamma(\|x\|)$ to be the translation by $(0, 0, \|x\|)$. Then the stabilizer $H_2^{\gamma(\|x\|)H_1}$ for $\|x\| \in H\backslash G/H$ is the set of rotations around Z, i.e. $\mathrm{SO}(2)$, except for $\|x\| = 0$, where it is $\mathrm{SO}(3)$.

For any representations ρ_1, ρ_2, the equivariant maps between sections of the associated vector bundle are given by convolutions with matrix-valued kernels on $\mathbb{R}^3$ that satisfy $\overleftarrow{\kappa}(rx) = \rho_2(r)\overleftarrow{\kappa}(x)\rho_1(r^{-1})$ for $r \in \mathrm{SO}(3)$ and $x \in \mathbb{R}^3$. This follows from Theorem 3.3 with the simplification $\mathrm{h}_1(x, r) = r$ for all $r \in H$, because $\mathrm{SE}(3)$ is a semidirect product (Appendix A.2). Alternatively, we can define $\overleftarrow{\kappa}$ in terms of $\bar{\kappa}$, which is a kernel on $H\backslash G/H = [0, \infty)$ satisfying $\bar{\kappa}(x) = \rho_2(r)\bar{\kappa}(x)\rho_1(r)$ for $r \in \mathrm{SO}(2)$ and $x \in [0, \infty)$. This is in agreement with the results obtained by [6].

7 Conclusion

In this paper we have developed a general theory of equivariant convolutional networks on homogeneous spaces using the formalism of fiber bundles and fields. Field theories are the de facto standard formalism for modern physical theories, and this paper shows that the same formalism can elegantly describe the de facto standard learning machine: the convolutional network and its generalizations. By connecting this very successful class of networks to modern theories in mathematics and physics, our theory provides many opportunities for the development of new theoretical insights about deep learning, and the development of new equivariant network architectures.

References

[1] Taco S Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, volume 48, pages 2990–2999, 2016.

[2] Taco S Cohen and Max Welling. Steerable CNNs. In *ICLR*, 2017.

[3] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant CNNs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and Translation-Equivariant neural networks for 3D point clouds. *arXiv:1802.08219 [cs.LG]*, 2018.

[6] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[7] Maurice Weiler and Gabriele Cesa. General E(2)-Equivariant Steerable CNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[8] Risi Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv:1803.01588 [cs.LG]*, 2018.

[9] Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. *arXiv:1801.02144 [cs.LG]*, January 2018.

[10] Taco S Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical CNNs. In *International Conference on Learning Representations (ICLR)*, 2018.

[11] Taco S. Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge Equivariant Convolutional Networks and the Icosahedral CNN. In *International Conference on Machine Learning (ICML)*, 2019.

[12] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning (ICML)*, 2018.

[13] Mark Hamilton. *Mathematical Gauge Theory: With Applications to the Standard Model of Particle Physics*. Universitext. Springer International Publishing, 2017. ISBN 978-3-319-68438-3. doi: 10.1007/978-3-319-68439-0.

[14] Marysia Winkels and Taco S Cohen. 3D G-CNNs for pulmonary nodule detection. In *International Conference on Medical Imaging with Deep Learning (MIDL)*, 2018.

[15] Daniel Worrall and Gabriel Brostow. CubeNet: Equivariance to 3D rotation and translation. In *European Conference on Computer Vision (ECCV)*, 2018.

[16] Taco S Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Convolutional Networks for Spherical Signals. In *ICML Workshop on Principled Approaches to Deep Learning*, 2017.

[17] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through Parameter-Sharing. In *International Conference on Machine Learning (ICML)*, 2017.

[18] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and Equivariant Graph Networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[19] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the Universality of Invariant Networks. In *International Conference on Machine Learning (ICML)*, 2019.

[20] Nimrod Segol and Yaron Lipman. On Universal Equivariant Set Networks. *arXiv:1910.02421 [cs, stat]*, October 2019.

[21] Nicolas Keriven and Gabriel Peyré. Universal Invariant and Equivariant Graph Neural Networks. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[22] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3856–3866. Curran Associates, Inc., 2017.

[23] Geoffrey Hinton, Nicholas Frosst, and Sara Sabour. Matrix capsules with EM routing. In *International Conference on Learning Representations (ICLR)*, 2018.

[24] Chris Olah. Groups and group convolutions. `https://colah.github.io/posts/2014-12-Groups-Convolution/`, 2014.

[25] R Gens and P Domingos. Deep symmetry networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[26] Laurent Sifre and Stephane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[27] E Oyallon and S Mallat. Deep Roto-Translation scattering for object classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2865–2873, 2015.

[28] Stéphane Mallat. Understanding deep convolutional networks. *Philos. Trans. A Math. Phys. Eng. Sci.*, 374(2065):20150203, April 2016.

[29] Jan J Koenderink. The brain a geometry engine. *Psychol. Res.*, 52(2-3):122–127, 1990.

[30] Jan Koenderink and Andrea van Doorn. The structure of visual spaces. *J. Math. Imaging Vis.*, 31(2):171, April 2008.

[31] Jean Petitot. The neurogeometry of pinwheels as a sub-riemannian contact structure. *J. Physiol. Paris*, 97(2-3):265–309, 2003.

[32] Taco S Cohen, Mario Geiger, and Maurice Weiler. Intertwiners between induced representations (with applications to the theory of equivariant neural networks). *arXiv:1803.10743 [cs.LG]*, March 2018.

[33] R W Sharpe. *Differential Geometry: Cartan's Generalization of Klein's Erlangen Program*. 1997.

[34] Adam Marsh. Gauge theories and fiber bundles: Definitions, pictures, and results. July 2016.

[35] G B Folland. *A Course in Abstract Harmonic Analysis*. CRC Press, 1995.

[36] T Ceccherini-Silberstein, A Machí, F Scarabotti, and F Tolli. Induced representations and mackey theory. *J. Math. Sci.*, 156(1):11–28, January 2009.

[37] David Gurarie. *Symmetries and Laplacians: Introduction to Harmonic Analysis, Group Representations and Applications*. Elsevier B.V., 1992.

[38] George W Mackey. On induced representations of groups. *Amer. J. Math.*, 73(3):576–592, July 1951.

[39] George W Mackey. Induced representations of locally compact groups I. *Ann. Math.*, 55(1): 101–139, 1952.

[40] George W Mackey. Induced representations of locally compact groups II. the frobenius reciprocity theorem. *Ann. Math.*, 58(2):193–221, 1953.

[41] George W Mackey. *Induced Representations of Groups and Quantum Mechanics*. W.A. Benjamin Inc., New York-Amsterdam, 1968.

[42] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. 3D object classification and retrieval with spherical CNNs. In *European Conference on Computer Vision (ECCV)*, 2018.

[43] Jim Winkens, Jasper Linmans, Bastiaan S Veeling, Taco S. Cohen, and Max Welling. Improved Semantic Segmentation for Histopathology using Rotation Equivariant Convolutional Networks. In *International Conference on Medical Imaging with Deep Learning (MIDL workshop)*, 2018.

[44] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine 34*, 2017.

[45] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, R E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a Back-Propagation network. In D S Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann, 1990.

[46] S Dieleman, J De Fauw, and K Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2016.

[47] Emiel Hoogeboom, Jorn W T Peters, Taco S Cohen, and Max Welling. HexaConv. In *International Conference on Learning Representations (ICLR)*, 2018.

[48] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. In *CVPR*, 2017.

[49] Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen A J Eppenhof, and Josien P W Pluim. Roto-Translation covariant convolutional networks for medical image analysis. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2018.

[50] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *International Conference on Computer Vision (ICCV)*, 2017.

[51] Rohan Ghosh and Anupam K Gupta. Scale steerable filters for locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1906.03861*, 2019.

[52] Daniel E Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. In *International Conference on Machine Learning (ICML)*, 2019.

[53] Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks, 2019.

[54] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch–Gordan Nets: A Fully Fourier Space Spherical Convolutional Neural Network. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[55] Brandon Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *arXiv preprint arXiv:1906.04015*, 2019.

[56] Nathanaël Perraudin, Michaël Defferrard, Tomasz Kacprzak, and Raphael Sgier. DeepSphere: Efficient spherical Convolutional Neural Network with HEALPix sampling for cosmological applications. *Astronomy and Computing 27*, 2018.

[57] Chiyu Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Niessner. Spherical CNNs on unstructured grids. In *International Conference on Learning Representations (ICLR)*, 2019.

A General facts about Groups and Quotients

Let G be a group and H a subgroup of G. A left coset of H in G is a set $gH = \{gh \mid h \in H\}$ for $g \in G$. The cosets form a partition of G. The set of all cosets is called the quotient space or coset space, and is denoted G/H. There is a canonical projection $p : G \to G/H$ that assigns to each element g the coset it is in. This can be written as $p(g) = gH$. Fig. 7 provides an illustration for the group of symmetries of a triangle, and the subgroup H of reflections.

The quotient space carries a left action of G, which we denote with ux for $u \in G$ and $x \in G/H$. This works fine because this action is associative with the group operation:

$$u(gH) = (ug)H. \tag{15}$$

for $u, g \in G$. One may verify that this action is well defined, i.e. does not depend on the particular coset representative g. Furthermore, the action is transitive, meaning that we can reach any coset from any other coset by transforming it with an appropriate $u \in G$. A space like G/H on which G acts transitively is called a homogeneous space for G. Indeed, any homogeneous space is isomorphic to some quotient space G/H.

A section of p is a map $s : G/H \to G$ such that $p \circ s = \mathrm{id}_{G/H}$. We can think of s as choosing a coset representative for each coset, i.e. $s(x) \in x$. In general, although p is unique, s is not; there can be many ways to choose coset representatives. However, the constructions we consider will always be independent of the particular choice of section.

Although it is not strictly necessary, we will assume that s maps the coset $H = eH$ of the identity to the identity $e \in G$:

$$s(H) = e \tag{16}$$

We can always do this, for given a section s' with $s'(H) = h \neq e$, we can define the section $s(x) = h^{-1}s'(hx)$ so that $s(H) = h^{-1}s'(hH) = h^{-1}s'(H) = h^{-1}h = e$. This is indeed a section, for $p(s(x)) = p(h^{-1}s'(hx)) = h^{-1}p(s'(hx)) = h^{-1}hx = x$ (where we used Eq. 15 which can be rewritten as $up(g) = p(ug)$).

One useful rule of calculation is

$$(gs(x))H = g(s(x)H) = gx = s(gx)H, \tag{17}$$

for $g \in G$ and $x \in G/H$. The projection onto H is necessary, for in general $gs(x) \neq s(gx)$. These two terms are however related, through a function $\mathrm{h} : G/H \times G \to H$, defined as follows:

$$gs(x) = s(gx)\mathrm{h}(x, g) \tag{18}$$

That is,

$$\boxed{\mathrm{h}(x, g) = s(gx)^{-1}gs(x)}. \tag{19}$$

We can think of $\mathrm{h}(x, g)$ as the element of H that we can apply to $s(gx)$ (on the right) to get $gs(x)$. The h function will play an important role in the definition of the induced representation, and is illustrated in Fig. 7.

From the fiber bundle perspective, we can interpret Eq. 19 as follows. The group G can be viewed as a principal bundle with base space G/H and fibers gH. If we apply g to the coset representative $s(x)$, we move to a different coset, namely the one represented by $s(gx)$ (representing a different point in the base space). Additionally, the fiber is twisted by the right action of $\mathrm{h}(x, g)$. That is, $\mathrm{h}(x, g)$ moves $s(gx)$ to another element in its coset, namely to $gs(x)$.

The following composition rule for h is very useful in derivations:

$$\begin{aligned} \mathrm{h}(x, g_1g_2) &= s(g_1g_2x)^{-1}g_1g_2s(x) \\ &= [s(g_1g_2x)^{-1}g_1s(g_2x)][s(g_2x)^{-1}g_2s(x)] \\ &= \mathrm{h}(g_2x, g_1)\mathrm{h}(x, g_2) \end{aligned} \tag{20}$$

For elements $h \in H$, we find:

$$\mathrm{h}(H, h) = s(H)^{-1}hs(H) = h. \tag{21}$$

Also, for any coset x,

$$\mathrm{h}(H, s(x)) = s(s(x)H)^{-1}s(x)s(H) = s(H) = e. \tag{22}$$

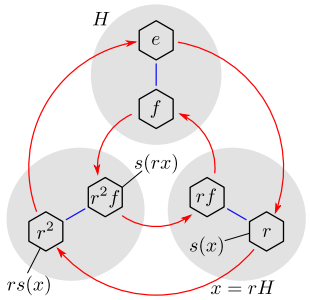


Figure 7: A Cayley diagram of the group $D3$ of symmetries of a triangle. The group is generated by rotations r and flips f. The elements of the group are indicated by hexagons. The red arrows correspond to right multiplication by r, while the blue lines correspond to right multiplication by f. Cosets of the group of flips ($H = \{e, f\}$) are shaded in gray. As always, the cosets partition the group. As coset representatives, we choose $s(H) = e$, $s(rH) = r$, and $s(r^2H) = r^2f$. The difference between $s(rx)$ and $rs(x)$ is indicated. For this choice of section, we must set $\mathrm{h}(x, r) = \mathrm{h}(rH, r) = f$, so that $s(rx)\mathrm{h}(x, r) = (r^2f)(f) = r^2 = rs(x)$.

Using Eq. 20 and 22, this yields,

$$\mathrm{h}(H, s(x)h) = \mathrm{h}(hH, s(x))\mathrm{h}(H, h) = h, \tag{23}$$

for any $h \in H$ and $x \in G/H$.

For $x = H$, Eq. 19 specializes to:

$$g = gs(H) = s(gH)\mathrm{h}(H, g) \equiv s(gH)\mathrm{h}(g), \tag{24}$$

where we defined

$$\boxed{\mathrm{h}(g) = \mathrm{h}(H, g) = s(gH)^{-1}g} \tag{25}$$

This shows that we can always factorize g *uniquely* into a part $s(gH)$ that represents the coset of g, and a part $\mathrm{h}(g) \in H$ that tells us where g is within the coset:

$$g = s(gH)\mathrm{h}(g) \tag{26}$$

A useful property of $\mathrm{h}(g)$ is that for any $h \in H$,

$$\mathrm{h}(gh) = s(ghH)^{-1}gh = s(gH)^{-1}gh = \mathrm{h}(g)h. \tag{27}$$

It is also easy to see that

$$\mathrm{h}(s(x)) = e. \tag{28}$$

When dealing with different subgroups H_1 and H_2 of G (associated with the input and output space of an intertwiner), we will write h_i for an element of H_i, $s_i : G/H_i \to G$, for the corresponding section, and $\mathrm{h}_i : G/H_i \times G \to H_i$ for the h-function (for $i = 1, 2$).

A.1 Double cosets

A (H_2, H_1)-double coset is a set of the form H_2gH_1 for H_2, H_1 subgroups of G. The space of (H_2, H_1)-double cosets is called $H_2\backslash G/H_1 \equiv \{H_2gH_1 \,|\, g \in G\}$. As with left cosets, we assume a section $\gamma : H_2\backslash G/H_1 \to G$ is given, satisfying $\gamma(H_2gH_1) \in H_2gH_1$.

The double coset space $H_2\backslash G/H_1$ can be understood as the space of H_2-orbits in G/H_1, that is, $H_2\backslash G/H_1 = \{H_2x | x \in G/H_1\}$. Note that although G acts transitively on G/H_1 (meaning that there is only one G-orbit in G/H_1), the subgroup H_2 does not. Hence, the space G/H_1 splits into a number of disjoint orbits H_2x (for $x = gH_1 \in G/H_1$), and these are precisely the double cosets H_2gH_1.

Of course, H_2 *does* act transitively within a single orbit H_2x, sending $x \mapsto h_2x$ (both of which are in H_2x, for $x \in G/H_1$). In general this action is not necessarily fixed point free which means that there

may exist some $h_2 \in H_2$ which map the left cosets to themselves. These are exactly the elements in the stabilizer of $x = gH_1$, given by

$$\begin{aligned} H_2^x &= \{h \in H_2 \,|\, hx = x\} \\ &= \{h \in H_2 \,|\, hs_1(x)H_1 = s_1(x)H_1\} \\ &= \{h \in H_2 \,|\, hs_1(x) \in s_1(x)H_1\} \\ &= \{h \in H_2 \,|\, h \in s_1(x)H_1 s_1(x)^{-1}\} \\ &= s_1(x)H_1 s_1(x)^{-1} \cap H_2. \end{aligned} \tag{29}$$

Clearly, H_2^x is a subgroup of H_2. Furthermore, H_2^x is conjugate to (and hence isomorphic to) the subgroup $s_1(x)^{-1}H_2^x s_1(x) = H_1 \cap s_1(x)^{-1}H_2 s_1(x)$, which is a subgroup of H_1.

For double cosets $x \in H_2 \backslash G / H_1$, we will overload the notation to $H_2^x \equiv H_2^{\gamma(x)H_1}$. Like the coset stabilizer, this double coset stabilizer can be expressed as

$$H_2^x = \gamma(x)H_1\gamma(x)^{-1} \cap H_2 \tag{30}$$

A.2 Semidirect products

For a semidirect product group G, such as $\mathrm{SE}(2) = \mathbb{R}^2 \rtimes \mathrm{SO}(2)$, some things simplify. Let $G = N \rtimes H$ where $H \leq G$ is a subgroup, $N \leq G$ is a normal subgroup and $N \cap H = \{e\}$. For every $g \in G$ there is a unique way of decomposing it into nh where $n \in N$ and $h \in H$. Thus, the left H coset of $g \in G$ depends only on the N part of g:

$$gH = nhH = nH \tag{31}$$

It follows that for a semidirect product group, we can define the section so that it always outputs an element of $N \subseteq G$, instead of a general element of G. Specifically, we can set $s(gH) = s(nhH) = s(nH) = n$. It follows that $s(nx) = ns(x) \quad \forall n \in N, x \in G/H$. This allow us to simplify expressions involving h:

$$\begin{aligned} \mathrm{h}(x, g) &= s(gx)^{-1} g s(x) \\ &= s(gs(x)H)^{-1} g s(x) \\ &= s(\underbrace{gs(x)g^{-1}}_{\in N} gH)^{-1} g s(x) \\ &= \left(gs(x)g^{-1}\, s(gH)\right)^{-1} g s(x) \\ &= s(gH)^{-1} g \\ &= \mathrm{h}(g) \end{aligned} \tag{32}$$

A.3 Haar measure

When we integrate over a group G, we will use the Haar measure, which is the essentially unique measure dg that is invariant in the following sense:

$$\int_G f(g)dg = \int_G f(ug)dg \quad \forall u \in G. \tag{33}$$

Such measures always exist for locally compact groups, thus covering most cases of interest [35]. For discrete groups, the Haar measure is the counting measure, and integration can be understood as a discrete sum.

We can integrate over G/H by using an integral over G,

$$\int_{G/H} f(x)dx = \int_G f(gH)dg. \tag{34}$$

B Proofs

B.1 Bi-equivariance of one-argument kernels on G

B.1.1 Left equivariance of κ

We want the result $\kappa \star f$ (or $\kappa \cdot f$) to live in $\mathcal{I}_G^2$, which means that this function has to satisfy the Mackey condition,

$$\begin{aligned} & [\kappa \star f](gh_2) &&= \rho_2(h_2^{-1})[\kappa \star f](g) \\ \Leftrightarrow \quad & \int_G \kappa((gh_2)^{-1}g')f(g')dg' &&= \rho_2(h_2^{-1}) \int_G \kappa(g^{-1}g')f(g')dg' \\ \Leftrightarrow \quad & \kappa(h_2^{-1}g^{-1}g') &&= \rho_2(h_2^{-1})\kappa(g^{-1}g') \\ \Leftrightarrow \quad & \kappa(h_2 g) &&= \rho_2(h_2)\kappa(g) \end{aligned} \tag{35}$$

for all $h_2 \in H_2$ and $g \in G$.

B.1.2 Right equivariance of κ

The fact that $f \in \mathcal{I}_G^1$ satisfies the Mackey condition ($f(gh) = \rho_1(h)f(g)$ for $h \in H_1$) implies a symmetry in the correlation $\kappa \star f$. That is, if we apply a right-H_1-shift to the kernel, i.e. $[R_h\kappa](g) = \kappa(gh)$, we find that

$$\begin{aligned} [[R_h\kappa] \star f](g) &= \int_G \kappa(g^{-1}uh)f(u)du \\ &= \int_G \kappa(g^{-1}u)f(uh^{-1})du \\ &= \int_G \kappa(g^{-1}u)\rho_1(h)f(u)du. \end{aligned} \tag{36}$$

It follows that we can take (for $h \in H_1$),

$$\kappa(gh) = \kappa(g)\rho_1(h). \tag{37}$$

B.2 Kernels on $H_2\backslash G/H_1$

We have seen the space $\mathcal{K}_C$ of H_2-equivariant kernels on G/H_1 appear in our analysis of both $\mathcal{I}_G$ and $\mathcal{I}_C$. Kernels in this space have to satisfy the constraint (for $h \in H_2$):

$$\overleftarrow{\kappa}(hy) = \rho_2(h)\overleftarrow{\kappa}(y)\rho_1(\mathrm{h}_1(y,h)^{-1}) \tag{38}$$

Here we will show that this space is equivalent to the space

$$\boxed{\begin{aligned} \mathcal{K}_D = \{\bar{\kappa} : H_2\backslash G/H_1 \to \mathrm{Hom}(V_1, V_2) \,|\, & \bar{\kappa}(x) = \rho_2(h)\bar{\kappa}(x)\rho_1^x(h)^{-1}, \\ & \forall x \in H_2\backslash G/H_1, h \in H_2^{\gamma(x)H_1}\}, \end{aligned}} \tag{39}$$

where we defined the representation ρ_1^x of the stabilizer $H_2^{\gamma(x)H_1}$,

$$\begin{aligned} \rho_1^x(h) &= \rho_1(\mathrm{h}_1(\gamma(x)H_1, h)) \\ &= \rho_1(\gamma(x)^{-1}h\gamma(x)), \end{aligned} \tag{40}$$

with the section $\gamma : H_2\backslash G/H_1 \to G$ being defined as in section A.1. To show the equivalence of $\mathcal{K}_C$ and $\mathcal{K}_D$, we define an ismorphism $\Omega_{\mathcal{K}} : \mathcal{K}_D \to \mathcal{K}_C$. We begin by defining $\Omega_{\mathcal{K}}^{-1}$:

$$\bar{\kappa}(x) = [\Omega_{\mathcal{K}}^{-1}\overleftarrow{\kappa}](x) = \overleftarrow{\kappa}(\gamma(x)H_1). \tag{41}$$

We verify that for $\overleftarrow{\kappa} \in \mathcal{K}_C$ we have $\bar{\kappa} \in \mathcal{K}_D$. Let $h \in H_2^{\gamma(x)H_1}$, then

$$\begin{aligned} \bar{\kappa}(x) &= \overleftarrow{\kappa}(\gamma(x)H_1) \\ &= \overleftarrow{\kappa}(h\gamma(x)H_1) \\ &= \rho_2(h)\overleftarrow{\kappa}(\gamma(x)H_1)\rho_1(\mathrm{h}_1(\gamma(x)H_1, h))^{-1} \\ &= \rho_2(h)\bar{\kappa}(x)\rho_1^x(h)^{-1} \end{aligned} \tag{42}$$

To define $\Omega_{\mathcal{K}}$, we use the decomposition $y = h\gamma(H_2 y)H_1$ for $y \in G/H_1$ and $h \in H_2$. Note that h may not be unique, because H_2 does not in general act freely on G/H_1.

$$\overleftarrow{\kappa}(y) = [\Omega_{\mathcal{K}}\bar{\kappa}](y) = [\Omega_{\mathcal{K}}\bar{\kappa}](h\gamma(H_2 y)H_1) = \rho_2(h)\bar{\kappa}(H_2 y)\rho_1(\mathrm{h}_1(\gamma(H_2 y)H_1, h))^{-1}. \tag{43}$$

We verify that for $\bar{\kappa} \in \mathcal{K}_D$ we have $\overleftarrow{\kappa} \in \mathcal{K}_C$.

$$\begin{aligned}
\overleftarrow{\kappa}(h'y) &= \overleftarrow{\kappa}(h'h\gamma(H_2 y)H_1) \\
&= \rho_2(h'h)\bar{\kappa}(H_2 y)\rho_1(\mathrm{h}_1(\gamma(H_2 y)H_1, h'h))^{-1} \\
&= \rho_2(h'h)\bar{\kappa}(H_2 y)\rho_1(\mathrm{h}_1(h\gamma(H_2 y)H_1, h')\mathrm{h}_1(\gamma(H_2 y)H_1, h))^{-1} \\
&= \rho_2(h')\rho_2(h)\bar{\kappa}(H_2 y)\rho_1(\mathrm{h}_1(\gamma(H_2 y)H_1, h))^{-1}\rho_1(\mathrm{h}_1(h\gamma(H_2 y)H_1, h'))^{-1} \\
&= \rho_2(h')\rho_2(h)\bar{\kappa}(H_2 y)\rho_1(\mathrm{h}_1(\gamma(H_2 y)H_1, h))^{-1}\rho_1(\mathrm{h}_1(y, h'))^{-1} \\
&= \rho_2(h')\overleftarrow{\kappa}(y)\rho_1(\mathrm{h}_1(y, h'))^{-1}
\end{aligned} \tag{44}$$

We verify that $\Omega_{\mathcal{K}}$ and $\Omega_{\mathcal{K}}^{-1}$ are indeed inverses:

$$\begin{aligned}
[\Omega_{\mathcal{K}}[\Omega_{\mathcal{K}}^{-1}\overleftarrow{\kappa}]](y) &= [\Omega_{\mathcal{K}}[\Omega_{\mathcal{K}}^{-1}\overleftarrow{\kappa}]](h\gamma(H_2 y)H_1) \\
&= \rho_2(h)[\Omega_{\mathcal{K}}^{-1}\overleftarrow{\kappa}](H_2 y)\rho_1(\mathrm{h}_1(\gamma(H_2 y)H_1, h))^{-1} \\
&= \rho_2(h)\overleftarrow{\kappa}(\gamma(H_2 y)H_1)\rho_1(\mathrm{h}_1(\gamma(H_2 y)H_1, h))^{-1} \\
&= \overleftarrow{\kappa}(h\gamma(H_2 y)H_1) \\
&= \overleftarrow{\kappa}(y).
\end{aligned} \tag{45}$$

In the other direction,

$$\begin{aligned}
[\Omega_{\mathcal{K}}^{-1}[\Omega_{\mathcal{K}}\bar{\kappa}]](x) &= [\Omega_{\mathcal{K}}\bar{\kappa}](\gamma(x)H_1) \\
&= [\Omega_{\mathcal{K}}\bar{\kappa}](\gamma(H_2\gamma(x)H_1)H_1) \\
&= \rho_2(e)\bar{\kappa}(H_2\gamma(x)H_1)\rho_1(\mathrm{h}_1(\gamma(H_2\gamma(x)H_1)H_1, e))^{-1} \\
&= \bar{\kappa}(x)
\end{aligned} \tag{46}$$

C Limitations of the Theory

The theory presented here is quite general but still has several limitations. Firstly, we only cover fields over homogeneous spaces. Although fields can be defined over more general manifolds, and indeed there has been some effort aimed at defining convolutional networks on general (or Riemannian) manifolds [44], we restrict our attention to homogeneous spaces because they come naturally equipped with a group action to which the network can be made equivariant. A more general theory would not be able to make use of this additional structure.

For reasons of mathematical elegance and simplicity, the theory idealizes feature maps as fields over a possibly continuous base space, but a computer implementation will usually involve discretizing this space. A similar approach is used in signal processing, where discretization is justified by various sampling theorems and band-limit assumptions. It seems likely that a similar theory can be developed for deep networks, but this has not been done yet.

D Classification of Equivariant CNNs

G	H	G/H	ρ	Reference
$\mathbb{Z}^2$	$\{1\}$	$\mathbb{Z}^2$	regular	Lecun 1990 [45]
$p4, p4m$	C_4, D_4	$\mathbb{Z}^2$	regular	Cohen 2016 [1],
"	"	"	"	Dieleman 2016 [46]
$p4, p4m$	C_4, D_4	$\mathbb{Z}^2$	irrep & regular	Cohen 2017 [2]
$p6, p6m$	C_6, D_6	$\mathbb{Z}^2$	regular	Hoogeboom 2018 [47]
$\mathbb{Z}^3 \rtimes H$	D_4, D_{4h}, O, O_h	$\mathbb{Z}^3$	regular	Winkels 2018 [14]
$\mathbb{Z}^3 \rtimes H$	V, T_4, O	$\mathbb{Z}^3$	regular	Worrall 2018 [15]
$\mathbb{R}^2 \rtimes C_N$	C_N	$\mathbb{R}^2$	regular	Weiler 2017 [4]
"	"	"	"	Zhou 2017 [48]
"	"	"	"	Bekkers 2018 [49]
"	"	"	irrep & regular	Marcos 2017 [50]
$\mathrm{SE}(2)$	$\mathrm{SO}(2)$	$\mathbb{R}^2$	irrep	Worrall 2017 [3]
$\mathbb{R}^2 \rtimes H \leq \mathrm{E}(2)$	$\mathrm{O}(2), \mathrm{SO}(2), C_N, D_N$	$\mathbb{R}^2$	any representation	Weiler 2019 [7]
$\mathbb{R}^2 \rtimes (\mathbb{R}^+, *)$	$(\mathbb{R}^+, *)$	$\mathbb{R}^2$	regular & trivial	Ghosh 2019 [51]
"	"	"	regular	Worrall 2019 [52]
"	"	"	"	Sosnovik 2019 [53]
$\mathrm{SE}(3)$	$\mathrm{SO}(3)$	$\mathbb{R}^3$	irrep	Kondor 2018 [8]
"	"	"	irrep & regular	Thomas 2018 [5]
"	"	"	irrep	Weiler 2018 [6]
"	"	"	irrep	Kondor 2018 [54]
"	"	"	irrep	Anderson 2019 [55]
$\mathrm{SO}(3)$	$\mathrm{SO}(2)$	S^2	regular	Cohen 2018 [10]
"	"	"	trivial	Esteves 2018 [42]
"	"	"	"	Perraudin 2018 [56]
"	"	"	irrep	Jiang 2019 [57]
G	H	G/H	trivial	Kondor 2018 [12]

Table 1: A taxonomy of G-CNNs. Methods are classified by the group G they are equivariant to, the subgroup H that acts on the fibers, the base space G/H to which the fibers are attached (implied by G and H), and the type of field ρ (regular, irreducible or trivial).

Miscellaneous Part III

6 Training Curve: Asymptotic Behavior of Kernel Regression

The following paper is the preprint version of Spigler et al. (2020).

Candidate contributions The candidate contributed by participating to all the discussions.

Asymptotic learning curves of kernel methods: empirical data *v.s.* Teacher-Student paradigm

Stefano Spigler[a], Mario Geiger[a], and Matthieu Wyart[a]

[a]Institute of Physics, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland

August 19, 2020

Abstract

How many training data are needed to learn a supervised task? It is often observed that the generalization error decreases as $n^{-\beta}$ where n is the number of training examples and β an exponent that depends on both data and algorithm. In this work we measure β when applying kernel methods to real datasets. For MNIST we find $\beta \approx 0.4$ and for CIFAR10 $\beta \approx 0.1$, for both regression and classification tasks, and for Gaussian or Laplace kernels. To rationalize the existence of non-trivial exponents that can be independent of the specific kernel used, we study the Teacher-Student framework for kernels. In this scheme, a Teacher generates data according to a Gaussian random field, and a Student learns them via kernel regression. With a simplifying assumption — namely that the data are sampled from a regular lattice — we derive analytically β for translation invariant kernels, using previous results from the kriging literature. Provided that the Student is not too sensitive to high frequencies, β depends only on the smoothness and dimension of the training data. We confirm numerically that these predictions hold when the training points are sampled at random on a hypersphere. Overall, the test error is found to be controlled by the magnitude of the projection of the true function on the kernel eigenvectors whose rank is larger than n. Using this idea we predict the exponent β from real data by performing kernel PCA, leading to $\beta \approx 0.36$ for MNIST and $\beta \approx 0.07$ for CIFAR10, in good agreement with observations. We argue that these rather large exponents are possible due to the small effective dimension of the data.

1 Introduction

In supervised learning machines learn from a finite collection of n training data, and their generalization error is then evaluated on unseen data drawn from the same distribution. How many data are needed to learn a task is characterized by the learning curve relating generalization error to n. In various cases, the generalization error decays as a power law $n^{-\beta}$, with an exponent β that depends on both the data and the algorithm. In [1] β is reported for state-of-the-art (SOTA) deep neural networks for various tasks: in for *neural-machine translation* $\beta \approx 0.3$–0.36 (for fixed model size) or $\beta \approx 0.13$ (for best-fit models at any n); *language modeling* shows $\beta \approx 0.06$–0.09; in *speech recognition* $\beta \approx 0.3$; SOTA models for *image classification* (on ImageNet) have exponents $\beta \approx 0.3$–0.5. Currently there is no available theory of deep learning to rationalize these observations. Recently it was shown that for a proper initialization of the weights, deep learning in the infinite-width limit [2] converges to kernel learning. Moreover, it is nowadays part of the lore that there exist kernels whose performance is nearly comparable to deep networks [3, 4], at least for some tasks. It is thus of great interest to understand the learning curves of kernels. For regression, if the target function being learned is simply assumed to be Lipschitz, then the best guarantee is $\beta = 1/d$ [5, 6] where d is the data dimension. Thus for large d, β is very small: learning is completely inefficient, a phenomenon referred to as the *curse of dimensionality*. As a result, various works on kernel regression make the much stronger assumption that the training points are sampled from a target function that belongs to the *reproducing kernel Hilbert space* (RKHS) of the kernel (for Gaussian r [7]). With this assumption β does not depend on d (for instance in [8] $\beta = 1/2$ is guaranteed). Yet, RKHS is a very strong assumption which requires the smoothness of the target function to increase with d [6] (for Gaussian random fields see Appendix H), which may not be realistic in large dimensions.

In Section 3 we compute β empirically for kernel methods applied on MNIST and CIFAR10 datasets. We find $\beta_{\text{MNIST}} \approx 0.4$ and $\beta_{\text{CIFAR10}} \approx 0.1$ respectively. Quite remarkably, we observe essentially the same exponents for regression and classification tasks, using either a Gaussian or a Laplace kernel. Thus the exponents are not as small as $1/d$ ($d = 784$ for MNIST, $d = 3072$ for CIFAR10), but neither are they $1/2$ as one would expect under the RKHS assumption. These facts call for frameworks in which assumptions on the smoothness of the data can be intermediary between Lipschitz and RKHS. Here we study such a framework for regression, in which the target function is assumed to be a Gaussian random field of zero mean with translation-invariant isotropic covariance $K_T(\underline{x})$. The data can equivalently be thought as being synthesized by a "Teacher" kernel $K_T(\underline{x})$. Learning is performed with a "Student" kernel $K_S(\underline{x})$ that minimizes the mean-square error. In general $K_T(\underline{x}) \neq K_S(\underline{x})$. In this set-up learning is very similar to a technique referred to as *kriging*, or Gaussian process regression, originally developed in the geostatistics community [9, 10].

To quantify learning, in Section 4 we first perform numerical experiments for data points distributed uniformly at random on a hypersphere of varying dimension d, focusing on a Laplace kernel for the Student, and considering a Laplace or Gaussian kernel for the Teacher. We observe that in both cases $\beta(d)$ is a decreasing function. In Section 5, to derive $\beta(d)$ we consider the simplified situation where the Gaussian random field is sampled at training points lying on a regular lattice. Building on the kriging literature [10], we show that β is controlled by the high-frequency scaling of both the Teacher and Student kernels: assuming that the Fourier transforms of the kernels decay as $\tilde{K}_T(\underline{w}) = c_T \|\underline{w}\|^{-\alpha_T} + o\left(\|\underline{w}\|^{-\alpha_T}\right)$ and $\tilde{K}_S(\underline{w}) = c_S \|\underline{w}\|^{-\alpha_S} + o\left(\|\underline{w}\|^{-\alpha_S}\right)$, we obtain

$$\beta = \frac{1}{d} \min(\alpha_T - d, 2\alpha_S). \tag{1}$$

Importantly (i) Eq. (1) leads to a prediction for $\beta(d)$ that accurately matches our numerical study for random training data points, leading to the conjecture that Eq. (1) holds in that case as well. We offer the following interpretation: ultimately, kernel methods are performing a local interpolation whose quality depends on the distance $\delta(n)$ between adjacent data points. $\delta(n)$ is asymptotically similar for random data or data sitting on a lattice. (ii) If the kernel K_S is not too sensitive to high-frequencies, then learning is optimal as far as scaling is concerned and $\beta = (\alpha_T - d)/d$. We will argue that the smoothness index $s \equiv [(\alpha_T - d)/2]$ characterizes the number of derivatives of the target function that are continuous. We thus recover the curse of dimensionality: s needs to be of order d to have non-vanishing β in large dimensions.

We show that in some regime, the test error for Gaussian data is controlled by an exponent a describing how the coefficients of the true function in the eigenbasis of the kernel decay with rank. We estimate a by the kernel principal component analysis (kernel PCA) based on diagonalizing the Gram matrix. This measure yields a prediction for the learning curve exponent β that matches the numerical fit, with $\beta_{\text{MNIST}} \approx 0.36$ and $\beta_{\text{CIFAR10}} \approx 0.07$. We show in Appendix I using the recent formalism of [11], which does not assume Gaussianity but makes more technical assumptions, that the result of our theorem Eq.1 is recovered, supporting further its validity for real data.

Finally, we discuss the following apparent paradox: β is significant for MNIST and CIFAR10, for which d is a priori very large, leading to a smoothness value s in the hundreds in both cases, which appears unrealistic. In Section 7 The paradox is resolved by considering that real datasets actually live on lower-dimensional manifolds. As far as kernel learning is concerned, our findings support that the correct definition of dimension should be based on how the nearest-neighbors distance $\delta(n)$ scales with n: $\delta(n) \sim n^{-1/d_{\text{eff}}}$. Direct measurements of $\delta(n)$ support that MNIST and CIFAR10 live on manifolds of lower dimensions $d^{\text{eff}}_{\text{MNIST}} \approx 15$ and $d^{\text{eff}}_{\text{CIFAR10}} \approx 35$.

2 Related works

Part of the literature has investigated the problem of kernel regression from a different point of view, namely the *optimal worst-case performance* (see for instance [12, 13, 14]). The target function is not assumed to be generated by a Gaussian random field, but its regularity is controlled using a *source condition* that constrains the decay of its coefficients in the eigenbasis of the kernel. For uniform data distributions and isotropic kernels this is similar to controlling how the Fourier transform of the target function decays at high frequency. What we study in the present work is, on the contrary, the *typical performance*. Indeed, it turns out that both the worst-case and the typical learning curve decay as power laws, and the latter decays faster.

The Teacher-Student framework for kernel regression was previously introduced in [15, 16], where a formula for the learning curve was derived based on a few uncontrolled approximations. It is easy to show that their results match the predictions of our theorem, although in [16] the case where the performance is limited by the Student ($\alpha_T - d > 2\alpha_S$) is ignored. More recently, [11] generalized this approach using similar approximations and extended it to kernel regression applied to any target function (or ensemble thereof). Kernel PCA on MNIST was used to support that these approximations hold well on real data. An asymptotic scaling relation between β and a was obtained, again the presence of other regimes was not noted. By contrast we perform an exact calculations for the asymptotic behavior of Gaussian data on a lattice. In Appendix I we show that the two approaches are consistent and lead to the same asymptotic predictions for β.

Our set-up of Teacher-Student learning with kernels is also referred to as *kriging*, or Gaussian process regression, and it was originally developed in the geostatistics community [9]. In Section 5 we present our theorem, that allows one to know the rate at which the test error decreases as we increase the number of training points, assumed to lie on a high-dimensional regular lattice. Similar results have been previously derived in the kriging literature [10] when sampling occurs on the regular lattice with the exception of the origin, where the inference is made. Here we propose an alternative derivation that some readers might find simpler. We also study a slightly different problem: instead of computing the test error when the inference is carried on at the origin, we compute the average error for a test point that lie at an arbitrary point, sampled uniformly at random and not necessarily on the lattice. Then, in what follows we show, via extensive numerical simulations, that such predictions are accurate even when the training points do not lie on a regular lattice, but are taken at random on a hypersphere. An exact proof of our result in such a general setting is difficult and cannot be found even in the kriging literature. To our knowledge the results that get closer to the point are those discussed in [17], where the author studies one-dimensional processes where the training data are not necessarily evenly spaced.

In this work the effective dimension of the data plays an import role, as it controls how the distance between nearest neighbors scales with the dataset size. Of course, there exists a vast literature [18, 19, 20, 21, 22, 23, 24] devoted to the study of effective dimensions, where other definitions are analyzed. The effective dimensions that we find are compatible with those obtained with more refined methods.

3 Learning curve for kernel methods applied to real data

In what follows we apply kernel methods to the MNIST and CIFAR10 datasets, each consisting of a set of images $(\underline{x}_\mu)_{\mu=1}^n$. We simplify the problem by considering only two classes whose label $Z(\underline{x}_\mu) = \pm 1$ correspond to odd and even numbers for MNIST, and to two groups of 5 classes in CIFAR10. The goal is to infer the value of the label $\hat{Z}_S(\underline{x})$ of an image $\underline{x}$ that does not belong to the dataset. The S subscript reminds us that inference is performed using a positive definite kernel K_S. We perform inference in both a *regression* and a *classification* setting. The following algorithms and associated results can be found in [25].

Regression. Learning corresponds to minimizing a mean-square error:

$$\min \sum_{\mu=1}^{n} \left[\hat{Z}_S(\underline{x}_\mu) - Z(\underline{x}_\mu) \right]^2 . \tag{2}$$

For algorithms seeking solutions of the form $\hat{Z}_S(\underline{x}) = \sum_\mu a_\mu K_S(\underline{x}_\mu, \underline{x}) \equiv \underline{a} \cdot \underline{k}_S(\underline{x})$ by minimizing the man-square loss over the vector $\underline{a}$, one obtains:

$$\hat{Z}_S(\underline{x}) = \underline{k}_S(\underline{x}) \cdot \mathbb{K}_S^{-1} \underline{Z}, \tag{3}$$

where the vector $\underline{Z}$ contains all the labels in the training set, $\underline{Z} \equiv (Z(\underline{x}_\mu))_{\mu=1}^n$, and $\mathbb{K}_{S,\mu\nu} \equiv K_S(\underline{x}_\mu, \underline{x}_\nu)$ is the Gram matrix. The Gram matrix is always invertible if the kernel K_S is positive definite. The generalization error is then evaluated as the expected mean-square error on unseen data, estimated by averaging over a test set composed of n_{test} unseen data points:

$$\text{MSE} = \frac{1}{n_{\text{test}}} \sum_{\mu=1}^{n_{\text{test}}} \left[\hat{Z}_S(\underline{x}_\mu) - Z(\underline{x}_\mu) \right]^2 . \tag{4}$$

Classification. We perform kernel classification via the algorithm *soft-margin SVM*. The details can be found in Appendix A. After learning from the training data with a student kernel K_S, performance is evaluated via the generalization error. It is estimated as the fraction of correctly predicted labels for data points belonging to a test set with n_{test} elements.

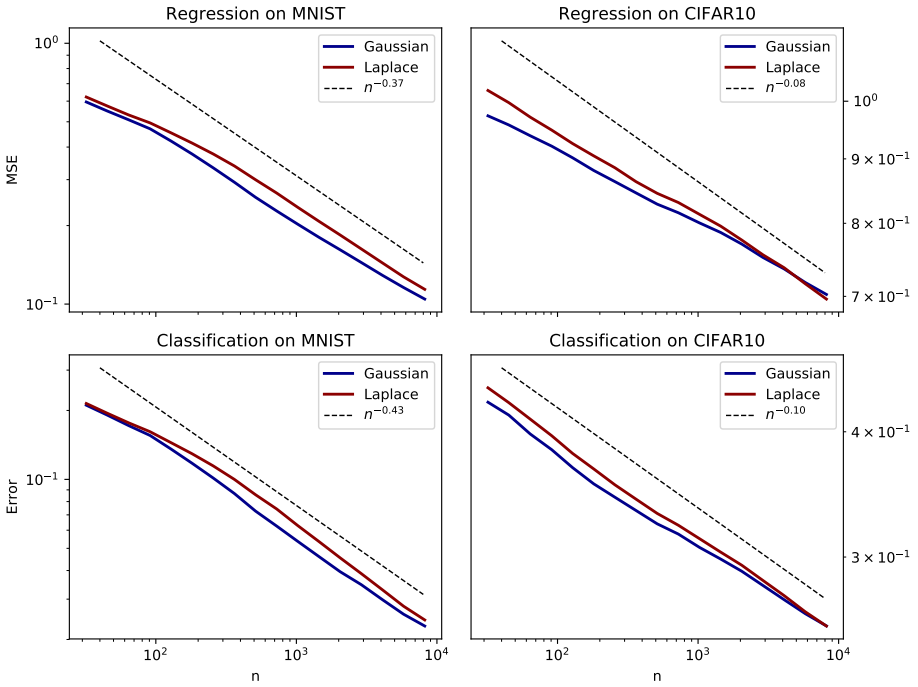


Figure 1: Learning curves for regression on MNIST and CIFAR10 *(*top row*)*; and for classification on MNIST and CIFAR10 *(*bottom row*)*. Curves are averaged over 400 runs. A power law is plotted to estimate the asymptotic behavior at large n: the exponent is fitted on the last decade on the average of the two curves, since it does not seem to depend significantly on the specific kernel or on the task. In each setting we use both a Gaussian kernel $K(\underline{x}) \propto \exp(-\|\underline{x}\|^2/(2\sigma^2))$ and a Laplace one $K(\underline{x}) \propto \exp(-\|\underline{x}\|/\sigma)$, with $\sigma = 1000$.

In Fig. 1 we present the learning curves for (binary) MNIST and CIFAR10, for regression and classification. Learning is performed both with a Gaussian kernel $K(\underline{x}) \propto \exp(-\|\underline{x}\|^2/(2\sigma^2))$ and a Laplace one $K(\underline{x}) \propto \exp(-\|\underline{x}\|/\sigma)$. Remarkably, the power laws in the two tasks are essentially identical (although the estimated exponent appears to be slightly larger, in absolute value, for classification). Moreover, the two kernels display a very similar behavior, compatible with the same exponent: about -0.4 for MNIST and -0.1 for CIFAR10. The presented data are for $\sigma = 1000$; in Appendix B we show that the same behaviour is observed for different values.

4 Generalization scaling in kernel Teacher-Student problems

We study β in a simplified setting where the data is assumed to follow a Gaussian distribution with known covariance. It falls into the class of teacher-Student problems, which are characterized by a machine (the Teacher) that generates the data, and another machine (the Student) that tries to learn from them. The Teacher-Student paradigm has been broadly used to study supervised learning [26, 27, 15, 16, 28, 29, 30, 31, 32, 33]. He we restrict our attention to kernel methods: we assume that a target function is distributed according to a Gaussian random field $Z \sim \mathcal{N}(0, K_T)$ — the Teacher — characterized by a translation-invariant isotropic covariance function $K_T(\underline{x}, \underline{x}') = K_T(\|\underline{x} - \underline{x}'\|)$, and that

the training dataset consists the finite set of n observations $\underline{Z} = (Z(\underline{x}_\mu))_{\mu=1}^n$. This is equivalent to saying that the vector of training points follows a centered Gaussian distribution with a covariance matrix that depends on K_T and on the location of the points $(\underline{x}_\mu)_{\mu=1}^n$:

$$\underline{Z} \sim \mathcal{N}(\underline{0}, \mathbb{K}_T), \quad \text{where} \quad \mathbb{K}_T = (K_T(\underline{x}_\mu, \underline{x}_\nu))_{\mu,\nu=1}^n. \tag{5}$$

Once the Teacher has generated the dataset, the rest follows as in the kernel regression described in the previous section. We use another translation-invariant isotropic kernel $K_S(\underline{x}, \underline{x}')$ — the Student — to infer the value of the field at another point, $\hat{Z}_S(\underline{x})$, with a regression task, i.e. minimizing the mean-square error in Eq. (2). The solution is therefore given again by Eq. (3).

Fig. 2 *(a-b)* shows the mean-square error obtained numerically. In the examples the Student is always taken to be a Laplace kernel, and the Teacher is either a Laplace kernel or a Gaussian kernel. The points $(\underline{x}_\mu)_{\mu=1}^n$ are taken uniformly at random on the unit d-dimensional hypersphere for several dimensions d and for several dataset sizes n. We take $\sigma_S = \sigma_T = d$ as we observed that with this choice smaller datasets were enough to approach a limiting curve — in Appendix C we show the plots for the case $\sigma_S = \sigma_T = 10$, which appears to converge to the same limit curve with increasing n, but at a smaller pace. The figure shows that when n is large enough, the mean-square error behaves as a power law (dashed lines) with an exponent that depends on the spatial dimension of the data, as well as on the kernels. The fitted exponents are plotted in Fig. 2 *(c-d)* as a function of the spatial dimension d for different dataset sizes n. In the next section we will discuss the theoretical prediction, that in the figure is plotted a thick black line. The figure shows that as the dataset gets bigger, the asymptotic exponent tends to our prediction. In Appendix D we present the learning curves of Gaussian Students with both a Laplace and a Gaussian kernel. When both kernels are Gaussian the test error decays exponentially fast, a result that matches our theoretical prediction. In Appendix E we also provide further numerical results for the case where the Teacher kernel is a Matérn kernel (as defined therein).

5 Analytic asymptotics for the kernel Teacher-Student problem on a lattice

In this section we compute analytically the exponent that describe the asymptotic decay of the generalization error when the number n of training data increases. In order to derive the result we assume that both the Teacher Gaussian random field lives on a bounded hypercube, $\underline{x} \in \mathcal{V} \equiv [0, L]^d$, where L is a constant and d is the spatial dimension. The fields and the kernels can then be thought of as L-periodic along each dimension. Furthermore, to make the problem tractable we assume that the points $(\underline{x}_\mu)_{\mu=1}^n$ live on a *regular lattice*, covering all the hypercube $\mathcal{V}$. Therefore, the linear spacing between neighboring points is $\delta = Ln^{-1/d}$. This is a different setting than the one used in the numerical simulations presented in the previous section for which the data distribution if Gaussian, showing that our results below are robust to such differences.

Generalization error is then evaluated via the typical mean-square error

$$\mathbb{E}\,\mathrm{MSE} = \mathbb{E}\left[Z(\underline{x}) - \hat{Z}_S(\underline{x})\right]^2, \tag{6}$$

where the expectation is taken over both the Teacher process and the point $\underline{x}$ at which we estimate the field, assumed to be uniformly distributed in the hypercube $\mathcal{V}$. In Appendix F we prove the following:

Theorem 1. *Let $\tilde{K}_T(\underline{w}) = c_T\|\underline{w}\|^{-\alpha_T} + o(\|\underline{w}\|^{-\alpha_T})$ and $\tilde{K}_S(\underline{w}) = c_S\|\underline{w}\|^{-\alpha_S} + o(\|\underline{w}\|^{-\alpha_S})$ as $\|\underline{w}\| \to \infty$, where $\tilde{K}_T(\underline{w})$ and $\tilde{K}_S(\underline{w})$ are the Fourier transforms of the kernels $K_T(\underline{x})$, $K_S(\underline{x})$ respectively, assumed to be positive definite. We assume that $\tilde{K}_T(\underline{w})$ and $\tilde{K}_S(\underline{w})$ have a finite limit as $\|\underline{w}\| \to 0$ and that $K(\underline{0}) < \infty$. Then,*

$$\mathbb{E}\,\mathrm{MSE} = n^{-\beta} + o\left(n^{-\beta}\right) \quad \text{with} \quad \beta = \frac{1}{d}\min(\alpha_T - d, 2\alpha_S). \tag{7}$$

Moreover, in the case of a Gaussian kernel the result holds valid if we take the corresponding exponent to be $\alpha = \infty$.

Apart from the specific value of the exponent in Eq. (7), Theorem 1 implies that if the Student kernel decays fast enough in the frequency domain, then β depends only on the data through the behaviour

of the Teacher kernel at high frequencies. One then recovers $\beta = (\alpha_T - d)/d$, also found for the *Bayes-optimal* setting where the Student is identical to the Teacher.

Consider the predictions of Theorem 1 in the cases presented in Fig. 2 *(a-b)* of Gaussian and Laplace kernels. If both kernels are Laplace kernels then $\alpha_T = \alpha_S = d+1$ and $\mathbb{E}\,\mathrm{MSE} \sim n^{-1/d}$, which scales very slowly with the dataset size in large dimensions. If the Teacher is a Gaussian kernel ($\alpha_T = \infty$) and the Student is a Laplace kernel then $\beta = 2(1+1/d)$, leading to $\beta \to 2$ as $d \to \infty$. In Fig. 2 *(c-d)* we compare these predictions with the exponents extracted from Fig. 2 *(a-b)*. We plot $\log \mathbb{E}\,\mathrm{MSE}/\log n \equiv -\beta$, against the dimension d of the data, varying the dataset size n. The exponents extracted numerically tend to our analytical predictions when n is large enough.

Notice that, although the theory and the experiments do not assume the same distribution for the sampling points $(\underline{x}_\mu)_{\mu=1}^n$, this does not seem to yield any difference in the asymptotic behavior of the generalization error, leading to the conjecture that our predictions are exact even when the training set is random, and does not correspond to a lattice. The conjecture can be proven in one dimension following results of the kriging literature [17], but generalization to higher d is a much harder problem. Intuitively, for kernel learning performs an expansion, whose quality is governed by the target function smoothness and the typical distance $\delta_{\min}$ between a point and its nearest neighbors in the training set. Both for random points or on a lattice, one has $\delta_{\min} \sim n^{-1/d}$ when n is large enough, thus both situations lead to the same β. This is shown in Fig. 4 (*left*).

Theorem 1 underlines that kernel methods are subjected to the curse of dimensionality. Indeed for

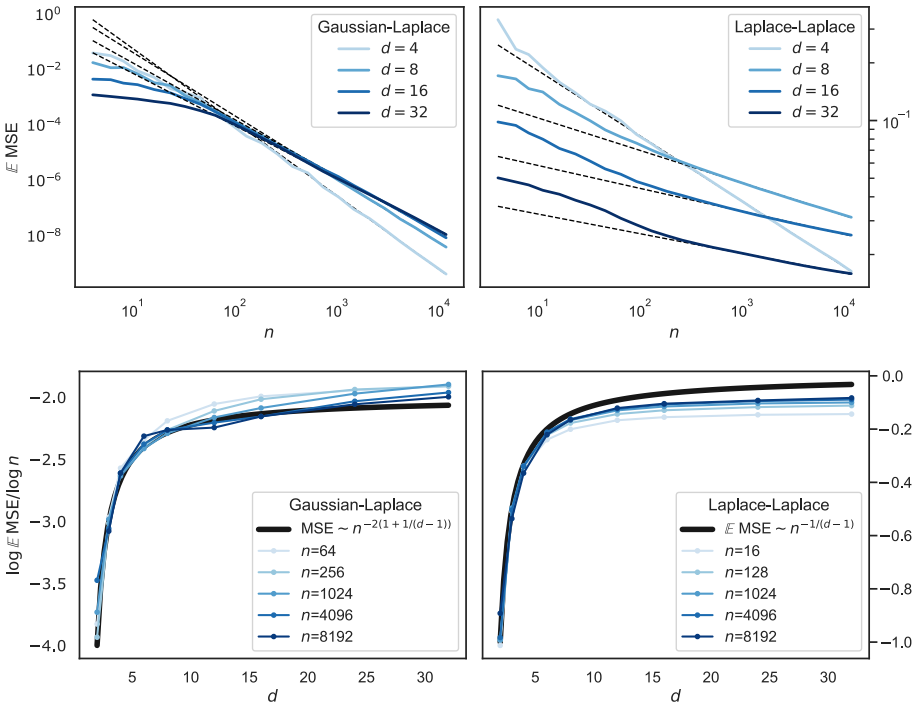


Figure 2: Results for the Teacher-Student kernel regression problem, where the Student is always a Laplace kernel. Data points are sampled uniformly at random on a d-dimensional hypersphere. (*Top row*) Mean-square error versus the size of the training dataset, for Gaussian and Laplace Teachers and for multiple spatial dimensions. Dotted lines are the fitted power laws — we fit starting from $n = 700$. (*Bottom row*) Fitted exponent $-\beta = \log \mathbb{E}\,\mathrm{MSE}/\log n$ against the spatial dimension, for several dataset sizes. We fit from $n = 0$ to a varying n (written in the legends). The thick black lines are the theoretical predictions.

appropriate students, one obtains $\beta = (\alpha_T - d)/d$. Let us define the smoothness index $s \equiv [(\alpha_T - d)/2] = \beta d/2$, which must be $\mathcal{O}(d)$ to avoid $\beta \to 0$ for large d. The two Lemmas below, derived in Appendix, indicate that the target function is s time differentiable (in a mean-square sense). Thus learning with kernels in very large dimension can only occur if the target function is $\mathcal{O}(d)$ times differentiable, a condition that appears very restrictive in large d.

Lemma 1. *Let $K(\underline{x}, \underline{x}')$ be a translation-invariant isotropic kernel such that $\tilde{K}(\underline{w}) = c\|\underline{w}\|^{-\alpha} + o(\|\underline{w}\|^{-\alpha})$ as $\|w\| \to \infty$ and $\|\underline{w}\|^d \tilde{K}(\underline{w}) \to 0$ as $\|w\| \to 0$. If $\alpha > d + n$ for some $n \in \mathbb{Z}^+$, then $K(\underline{x}) \in C^n$, that is, it is at least n-times differentiable. (Proof in Appendix G).*

Lemma 2. *Let $Z \sim \mathcal{N}(0, K)$ be a d-dimensional Gaussian random field, with $K \in C^{2n}$ being a $2n$-times differentiable kernel. Then Z is n-times* mean-square *differentiable in the sense that*

- *derivatives of $Z(\underline{x})$ are a Gaussian random fields;*
- $\mathbb{E}\partial_{x_1}^{n_1} \cdots \partial_{x_d}^{n_d} Z(\underline{x}) = 0$;
- $\mathbb{E}\partial_{x_1}^{n_1} \cdots \partial_{x_d}^{n_d} Z(\underline{x}) \cdot \partial_{x_1}^{n'_1} \cdots \partial_{x_d}^{n'_d} Z(\underline{x}') = \partial_{x_1}^{n_1+n'_1} \cdots \partial_{x_d}^{n_d+n'_d} K(\underline{x} - \underline{x}') < \infty$ *if the derivatives of K exist.*

In particular, $\mathbb{E}\partial_{x_i}^m Z(\underline{x}) \cdot \partial_{x_i}^m Z(\underline{x}') = \partial_{x_i}^{2m} K(\underline{x} - \underline{x}') < \infty \; \forall m \leq n$. (Proof in Appendix G).

Interpretation of Theorem 1: When the student does not limit performance, i.e. when $2\alpha_S > \alpha_T - d$ and $\beta = \frac{\alpha_T - d}{d}$, we can interpret the result as follows. An isotropic Student kernel corresponds to a Gaussian prior on the Fourier coefficients of the target function being learned. The student puts large (low) power at low (high) frequencies, and it can then reconstruct a number of the order of n largest Fourier coefficients, which corresponds to frequencies $\underline{w}$ of norm $\|\underline{w}\| \leq 1/\delta \sim n^{1/d}$. Fourier coefficients $\tilde{Z}(\underline{w})$ at higher frequencies cannot be learned, and the mean square error is then simply of order of the sum of the squares of these coefficients:

$$\mathbb{E}\,\mathrm{MSE} \sim \sum_{\|\underline{w}\| \geq n^{1/d}} |\tilde{Z}(\underline{w})|^2 \sim \sum_{\|\underline{w}\| \geq n^{1/d}} \|\underline{w}\|^{-\alpha_T} \sim n^{\frac{d-\alpha_T}{d}} \sim n^{-\beta}. \tag{8}$$

6 Learning curve exponent of real data

Eq. (8) is not readily applicable to real data which are neither Gaussian nor uniformly distributed. However it supports the following broader result: kernel methods can predict well of order n first coefficients of the true function in the eigenbasis of the kernel, but not the following ones. For any student kernel K_S, let $\lambda_1 \geq \lambda_\rho \geq \dots$ be its eigenvalues (positive and real, because of symmetry and positive definiteness) and $\phi_\rho(\underline{x})$ the associated eigenfunctions:

$$\int \mathrm{d}^d \underline{y}\, p(\underline{y}) K_S(\underline{x} - \underline{y}) \phi_\rho(\underline{y}) = \lambda_\rho \phi_\rho(\underline{x}), \tag{9}$$

where $p(\underline{x})$ is the density of the data points. Then the kernel can be decomposed in its eigenmodes:

$$K_S(\underline{x} - \underline{y}) = \sum_{\rho \geq 1} \lambda_\rho \phi_\rho(\underline{x}) \phi_\rho(\underline{y}). \tag{10}$$

The eigenfunctions of K make a complete basis, and we can write any function $Z(\underline{x})$ as

$$Z(\underline{x}) = \sum_\rho q_\rho \phi_\rho(\underline{x}). \tag{11}$$

The generalization of our result then simply reads:

$$\mathbb{E}\,\mathrm{MSE} \sim \sum_{\rho \geq n} q_\rho^2. \tag{12}$$

Assuming a power-law behavior $q_\rho^2 \sim \rho^{-a}$ then leads to $\beta = a - 1$.

To extract the exponent a and test this prediction for real data, we first approximate the eigenvalue equation Eq. (9) for the Student kernel with the diagonalization of its finite-dimensional Gram matrix $\mathbb{K}_S$ computed on a large dataset of size $\tilde{n}$:

$$\mathbb{K}_S \underline{\phi}_\rho \sim \lambda_\rho \underline{\phi}_\rho, \tag{13}$$

where now we have $\tilde{n}$ eigenvalues $\lambda_1 \geq \cdots \geq \lambda_{\tilde{n}}$ and the eigenvectors $\underline{\phi}_\rho$ are $\tilde{n}$-dimensional eigenvectors. Computing this diagonalization for a given training set is referred to (uncentered) *kernel PCA*. This procedure is a discretized version of Eq. (10) and yields only an approximation to the largest $\tilde{n}$ eigenvalues of the kernel, that are exactly recovered as $\tilde{n} \to \infty$ (in Fig. 9 in Appendix J we show that the eigenvalues of the Gram matrix converge when $\tilde{n}$ increases, and that their density displays the power-law behavior that one can extract from the kernel operator with a uniform distribution $p(\underline{x})$). The coefficient q_ρ is then estimated by the scalar products $\left(\underline{Z} \cdot \underline{\phi}_\rho\right)$, where $\underline{Z} = (Z(\underline{x}_1), \cdots, Z(\underline{x}_{\tilde{n}}))$ is the vector of the target function's values on the train set.

Finally, we approximate Eq. (12) as:

$$\sum_{\rho=n}^{\tilde{n}} \left(\underline{Z} \cdot \underline{\phi}_\rho\right)^2. \tag{14}$$

This quantity is plotted in Fig. 3, where we show that it correlates remarkably well with the true learning curve. Fitting these cumulative curves whose exponent is $1 - a$ for asymptotically large $\tilde{n}$ (here we plotted several curves for growing $\tilde{n}$) we extract an exponent $a_{\text{MNIST}} = 1.36$ leading to $\hat{\beta}_{\text{MNIST}} \approx 0.36$ and $a_{\text{CIFAR10}} = 1.07$ leading to $\hat{\beta}_{\text{CIFAR10}} \approx 0.07$ that are very close to the exponents that we measured in Section 3.

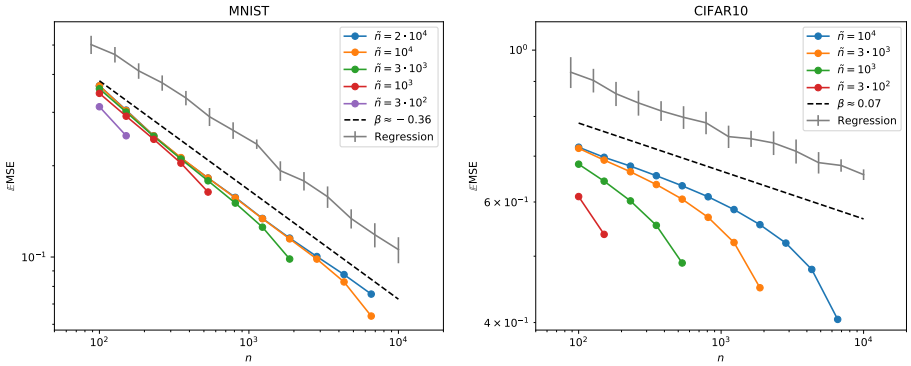


Figure 3: Several measures of the learning curves for MNIST (*left*) and CIFAR10 (*right*). In every plot, the gray solid line is the numerical evaluation of the generalization error (shifted for clarity). Colored lines are computed using Eq. (14) for several values of $\tilde{n}$, and the dashed black line is the a fit of the power-law decay with which we extract the predicted exponents $\hat{\beta}_{\text{MNIST}} \approx 0.36$ and $\hat{\beta}_{\text{CIFAR10}} \approx 0.07$.

Support for the genericity of Eq. (12) can be obtained from the recent paper [11], where the authors derived a formula for the generalization error based on the decomposition of the target function on the eigenbasis of the kernel. The formula is derived with uncontrolled approximations, but applies to a generic target function (or ensembles thereof) and a generic data point distribution $p(\underline{x})$, and matches well their numerical experiments. In Appendix I we show that the asymptotic limit (large n) of their formula yields Eq. (12). Furthermore, Eq.7 is recovered if a power-law decay of the coefficient of the true function in the eigenbasis of the kernel is assumed- thus generalizing our result to non-Gaussian data.

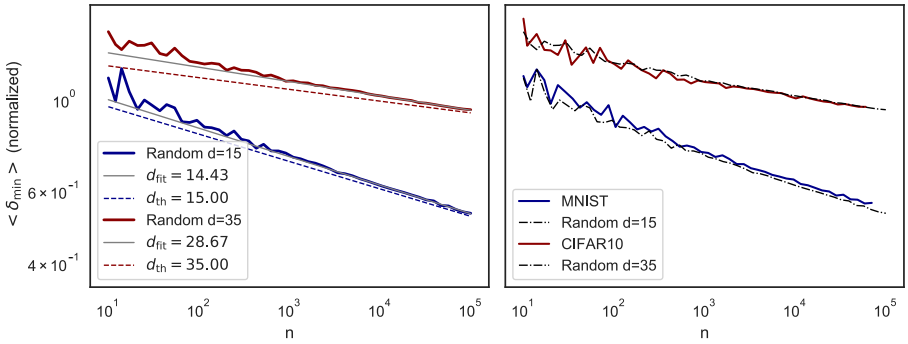


Figure 4: Average distance from one point to its nearest neighbor as a function of the dataset size n. (*Left*) For random points on d-dimensional hypersphere, $\langle \delta_{\min} \rangle \sim n^{-1/d}$. Colored solid curves are found numerically, dashed lines are the theoretical asymptotic prediction and the gray lines are numerical fit (we fitted only starting from $n \approx 6000$ to reduce finite size effects, and the fit have been rescaled to match the data at $n = 10$). The larger d, the stronger the preasymptotic effects (a larger n is needed to observe the predicted scaling). (*Right*) Comparison between random data on 15- and 35-dimensional hyperspheres and the MNIST, CIFAR10 datasets. According to this definition of effective dimension, MNIST live on a 15-dimensional manifold and CIFAR10 on a 35-dimensional one. Data have been rescaled along the y-axis for ease of comparison.

7 Effective dimension of real data

Both our predictions and empirical observations support rather large values of β. From a Gaussian random process point of view, it is surprising: the exponent β avoids the curse of dimensionality only if the smoothness of the Teacher is of the order of the data dimension. If these observations were to hold true also for real data, it would seem to imply that MNIST and CIFAR10 must be hundreds or thousands of times differentiable! However, there is a simple catch: real data actually live on a manifold of much lower dimensionality. Above we have argued that the quantity that governs the asymptotic learning curve is the typical distance $\delta_{\min}$ between neighboring points in the training set. A simple way to measure the effective dimension d_{eff} of real data consists then in plotting the (asymptotic) dependence of $\delta_{\min}$ on the number of points n in a random subset of the dataset, and fitting

$$\delta_{\min} \sim n^{-1/d_{\text{eff}}}. \tag{15}$$

In Fig. 4 (*right*) we show that for MNIST and CIFAR10 there is indeed a power-law relation linking $\delta_{\min}$ to n, and that the effective dimension extracted this way is much smaller than the embedding dimension of the datasets:

$$d_{\text{eff}}^{\text{MNIST}} \approx 15 \ll 784 = d^{\text{MNIST}}, \tag{16}$$

$$d_{\text{eff}}^{\text{CIFAR10}} \approx 35 \ll 3072 = d^{\text{CIFAR10}}. \tag{17}$$

This measure is consistent with previous extrapolations of the intrinsic dimension of MNIST [19, 20, 22, 23].

8 Conclusion

In this work we have shown for CIFAR10 and MNIST respectively that kernel regression and classification display a power-law decay in the learning curves, quite remarkably with essentially the same exponent β regardless of task and kernel — a fact yet to be explained. These exponents are much larger than $\beta = 1/d$ expected for Lipschitz target functions and smaller than $\beta = 1/2$ expected for RKHS target functions.

This observation led us to study a Teacher-Student framework for regression in which data are modeled as Gaussian random fields of varying smoothness, in which intermediary values of β are obtained. We

find two regimes depending on the respective smoothness of the Teacher and Student kernels. If the student is smooth enough — i.e. it puts a sufficiently low prior on high frequency components — then β is entirely controlled by the Teacher. We obtain that the smoothness index must scale with the dimension for β to be finite as $d \to \infty$, recovering the curse of dimensionality.

In our calculations, the dimension enters as the parameter relating the number of points to the nearest-neighbor distance $\delta \sim n^{-1/d}$. Thus in practice the parameter d considered should be the effective dimension d_{eff} of the data, which is much smaller than the number of pixels for MNIST and CIFAR data. It explains why β is not very small in these cases.

Finally, for Gaussian fields our result is equivalent to the statement that β is governed by the power of the true function past the first $\sim n$ eigenvectors of the kernel. We test this more general idea both for CIFAR and MNIST and find that it correctly predicts β. Understanding what controls this power in a general setting (which include the effective dimension of the data and presumably a generalized quantity characterizing smoothness) thus appears necessary to understand how many data are required to learn a task.

Acknowledgments

We acknowledge G. Biroli, C. Hongler and F. Gabriel for the discussions that stimulated this work, and we thank S. d'Ascoli, A. Jacot, C. Pehlevan, L. Sagun and M.L. Stein for discussions. This work was partially supported by the grant from the Simons Foundation (#454953 Matthieu Wyart). M.W. thanks the Swiss National Science Foundation for support under Grant No. 200021-165509.

References

[1] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Frederick Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *CoRR*, abs/1712.00409, 2017.

[2] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[3] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[4] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

[5] Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. *Journal of Machine Learning Research*, 5(Jun):669–695, 2004.

[6] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

[7] Alex J Smola, Bernhard Schölkopf, and Klaus-Robert Müller. The connection between regularization operators and support vector kernels. *Neural networks*, 11(4):637–649, 1998.

[8] Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, pages 3215–3225, 2017.

[9] Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

[10] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 1999.

[11] Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. *arXiv preprint arXiv:2002.02561*, 2020.

[12] Simon Fischer and Ingo Steinwart. Sobolev norm learning rates for regularized least-squares algorithm. *arXiv preprint arXiv:1702.07254*, 2017.

[13] Andrea Caponnetto and Ernesto De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.

[14] Loucas Pillaud-Vivien, Alessandro Rudi, and Francis Bach. Statistical optimality of stochastic gradient descent on hard learning problems through multiple passes. In *Advances in Neural Information Processing Systems*, pages 8114–8124, 2018.

[15] Peter Sollich. Learning curves for gaussian processes. In *Advances in neural information processing systems*, pages 344–350, 1999.

[16] Peter Sollich. Gaussian process regression with mismatched models. In *Advances in Neural Information Processing Systems*, pages 519–526, 2002.

[17] Michael L Stein. Predicting random fields with increasing dense observations. *The Annals of Applied Probability*, 9(1):242–273, 1999.

[18] Peter Grassberger and Itamar Procaccia. Measuring the strangeness of strange attractors. *Physica D: Nonlinear Phenomena*, 9(1-2):189–208, 1983.

[19] Jose A Costa and Alfred O Hero. Learning intrinsic dimension and intrinsic entropy of high-dimensional datasets. In *2004 12th European Signal Processing Conference*, pages 369–372. IEEE, 2004.

[20] Matthias Hein and Jean-Yves Audibert. Intrinsic dimensionality estimation of submanifolds in r d. In *Proceedings of the 22nd international conference on Machine learning*, pages 289–296. ACM, 2005.

[21] Elizaveta Levina and Peter J Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in neural information processing systems*, pages 777–784, 2005.

[22] Alessandro Rozza, Gabriele Lombardi, Claudio Ceruti, Elena Casiraghi, and Paola Campadelli. Novel high intrinsic dimensionality estimators. *Machine learning*, 89(1-2):37–65, 2012.

[23] Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):12140, 2017.

[24] Michele Allegra, Elena Facco, Alessandro Laio, and Antonietta Mira. Clustering by the local intrinsic dimension: the hidden structure of real-world data. *arXiv preprint arXiv:1902.10459*, 2019.

[25] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[26] David Saad and Sara A Solla. On-line learning in soft committee machines. *Physical Review E*, 52(4):4225, 1995.

[27] Rémi Monasson and Riccardo Zecchina. Weight space structure and internal representations: a direct approach to learning and generalization in multilayer neural networks. *Physical review letters*, 75(12):2432, 1995.

[28] Manfred Opper and David Saad. *Advanced mean field methods: Theory and practice*. MIT press, 2001.

[29] Andreas Engel and Christian Van den Broeck. *Statistical mechanics of learning*. Cambridge University Press, 2001.

[30] Jean Barbier, Florent Krzakala, Nicolas Macris, Léo Miolane, and Lenka Zdeborova. Optimal errors and phase transitions in high-dimensional generalized linear models. *Proceedings of the National Academy of Sciences*, 116:201802705, 03 2019.

[31] Marylou Gabrié, Andre Manoel, Clément Luneau, Nicolas Macris, Florent Krzakala, Lenka Zdeborová, et al. Entropy and mutual information in models of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1821–1831, 2018.

[32] Benjamin Aubin, Antoine Maillard, Florent Krzakala, Nicolas Macris, Lenka Zdeborová, et al. The committee machine: Computational to statistical gaps in learning a two-layers neural network. In *Advances in Neural Information Processing Systems*, pages 3223–3234, 2018.

[33] Silvio Franz, Sungmin Hwang, and Pierfrancesco Urbani. Jamming in multilayer supervised learning models. *arXiv preprint arXiv:1809.09945*, 2018.

[34] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA, 2006.

A Soft-margin Support Vector Machines

The kernel classification task is performed via the algorithm known as *soft-margin Support Vector Machine*.

We want to find a function $\hat{Z}_S(\underline{x})$ such that its sign correctly predicts the label of the data. In this context we model such a function as a linear prediction after projecting the data on a *feature space* via $\underline{x} \to \underline{\phi}(\underline{x})$:

$$\hat{Z}_S(\underline{x}) = \underline{w} \cdot \underline{\phi}_S(\underline{x}_\mu) + b, \tag{18}$$

where $\underline{w}, b$ are parameters to be learned from the training data. The kernel is related to the feature space via $K_S(\underline{x}, \underline{x}') = \underline{\phi}_S(\underline{x}) \cdot \underline{\phi}_S(\underline{x}')$. We require that $Z(\underline{x}_\mu)\hat{Z}_S(\underline{x}_\mu) > 1 - \xi_\mu$ for all training points. Ideally we want to have some large margins $1 - \xi_\mu = 1$, but we allow some of them to be smaller by introducing the *slack variables* ξ_μ and penalizing large values. To achieve this the following constrained minimization is performed:

$$\min_{\underline{w},b,\underline{\xi}} \frac{1}{2}\|\underline{w}\|^2 + C\sum_\mu \xi_\mu \quad \text{subjected to} \quad \forall \mu \; Z(\underline{x}_\mu)\left[\underline{w} \cdot \underline{\phi}_S(\underline{x}_\mu) + b\right] \geq 1 - \xi_\mu, \; \xi_\mu \geq 0. \tag{19}$$

This problem can be expressed in a dual formulation as

$$\min_{\underline{a}} \frac{1}{2}\underline{a} \cdot \mathbb{Q}_S \underline{a} - \sum_{\mu=1}^{n} a_\mu \quad \text{subjected to} \quad \underline{Z} \cdot \underline{a} = 0, \; 0 \leq a_\mu \leq C, \tag{20}$$

where $\mathbb{Q}_{S,\mu,\nu} = Z(\underline{x}_\mu)Z(\underline{x}_\nu)K_S(\underline{x}_\mu, \underline{x}_\nu)$ and $\underline{Z}$ is the vector of the labels of the training points. Here C ($= 10^4$ in our simulations) controls the trade-off between minimizing the training error and maximizing the *margins* $1 - \xi_\mu$. For the details we refer to [25]. If $\underline{a}^\star$ is the solution to the minimization problem, than

$$\underline{w}^\star = \sum_\mu a^\star_\mu \underline{\phi}_S(\underline{x}_\mu), \tag{21}$$

$$b^\star = Z(\underline{x}_\mu) - \sum_\nu a_\nu y_\nu K_S(\underline{x}_\mu, \underline{x}_\nu) \quad \text{for any } \mu \text{ such that } a_\mu < C. \tag{22}$$

The predicted label for unseen data points is then

$$\text{sign}(\hat{Z}_S(\underline{x})) = \text{sign}(\sum_\mu Z(\underline{x}_\mu) a_\mu K_S(\underline{x}_\mu, \underline{x}) + b^\star) \tag{23}$$

The generalization error is now defined as the probability that an unseen image has a predicted label different from the true one, and such a probability is again estimated as an average over a test set with n_{test} elements:

$$\text{Error} = \frac{1}{n_{\text{test}}} \sum_{\mu=1}^{n_{\text{test}}} \theta\left[-\text{sign}\left(\hat{Z}_S(\underline{x}_\mu)\right) Z(\underline{x}_\mu)\right]. \tag{24}$$

B Different kernel variances

In Fig. 5 we show the learning curves for kernel regression on the MNIST (parity) dataset — the same setting as in Fig. 1 *(a)*. Several Laplace kernels of varying variance σ are used. The variance ranges several orders of magnitude and the learning curves all decay with the same exponent, although for $\sigma = 10$ the algorithm achieves suboptimal performance and the test errors are increased by some factor.

C Different choice of kernel variances

In Fig. 6 we show the learning curves for the Teacher-Student kernel regression problem, with a Student kernel that is always Laplace and a Teacher that can be either Gaussian or Laplace. We show

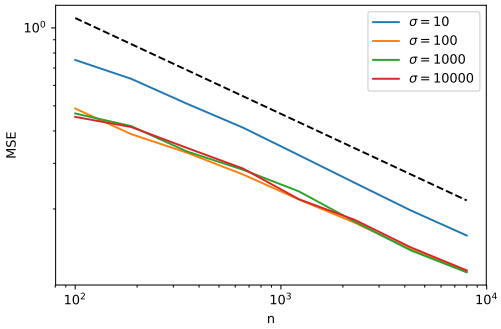


Figure 5: Learning curves for kernel regression on the MNIST dataset. Regression is performed with several Laplace kernels of varying variance σ ranging from $\sigma = 10$ to $\sigma = 10000$.

how the test error decays with the size of the training dataset and how the asymptotic exponent depends on the spatial dimension. Every experiment is run with two different choices of the kernel variances: in one case $\sigma_T = \sigma_S = d$ and in the other $\sigma_T = \sigma_S = 10$. We observed that scaling the variances with the spatial dimension leads faster to the results that we predicted in this paper, but overall the choice has little effect on the exponents (both tend towards the prediction as the dataset size is increased).

D Gaussian Students

In this appendix we present the learning curves of Gaussian Students: the Fourier transform of these kernels decays faster than any power law and one can effectively consider $\alpha_S = \infty$. If the Teacher is Laplace ($\alpha_T = d+1$) then the predicted exponent is finite and takes the values $\beta = \frac{1}{d}\min(\alpha_T - d, 2\alpha_S) = \frac{1}{d}\min(1, \infty) = \frac{1}{d}$. Such a case is displayed in Fig. 7 *(left)* in dimension $d = 6$. However, if we consider the Teacher to be Gaussian as well, then the predicted exponent would be $\beta = \frac{1}{d}\min(\infty, \infty) = \infty$. This case corresponds to Fig. 7 *(center)*: the test errors decays faster than a power law. In Fig. 7 *(right)* we compare the case where both kernels are Gaussian to the case where both kernels are Laplace: while the latter decays as a power law, the former decays much faster.

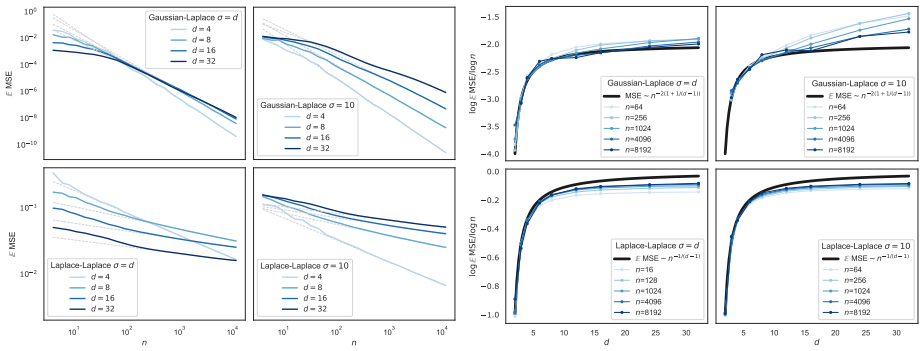


Figure 6: In these plots we show the results for the Teacher-Student kernel regression. The Student is always a Laplace kernel, the Teacher is either Gaussian or Laplace. The four plots on the left depict the mean-square error against the size of the dataset for different spatial dimensions of the data, those on the right show the fitted asymptotic exponent against the spatial dimension for different dataset sizes. For every case we show both the the results for $\sigma_T = \sigma_S = d$ and $\sigma_T = \sigma_S = 10$.

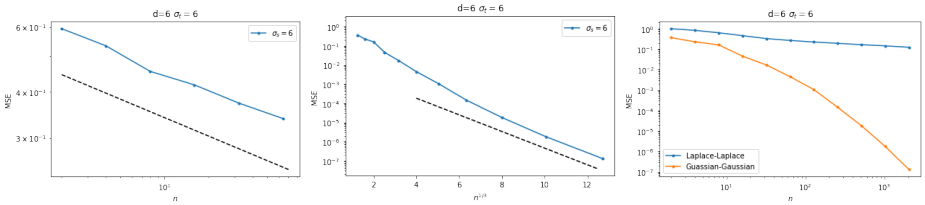


Figure 7: **Left:** The test error of a Laplace Teacher ($\alpha_T = d + 1$) with a Gaussian Student ($\alpha_S = \infty$) decays as a power law with the predicted exponent $\beta = \frac{1}{d}\min(1, \infty) = \frac{1}{6}$ in $d = 6$ dimensions. **Center:** When both the Teacher and the Student are Gaussian the test error decays faster than any power law as the number n of data is increased. This plot confirm this by showing that the logarithm of the test error decays linearly as a function of $n^{\frac{1}{3}}$. **Right:** Comparison between the learning curves for the cases where both kernels are either Laplace (top blue line) or Gaussian (bottom orange line). While the former decays algebraically with the predicted exponent, the latter decays exponentially, in agreement with the prediction $\beta = \infty$ found within our framework. In all these plots we have taken the variances of both the Teacher and Student kernels to be equal to the dimension $d = 6$.

E Matérn Teachers

To further test the applicability of our theory, we show here some numerical simulations for a Teacher kernel that is a Matérn covariance function and a Laplace kernel as student. We ran the simulations in 1d: the data points are sampled uniformly on a 1-dimensional circle embedded in $\mathbb{R}^2$. Matérn kernels are parametrized by a parameter $\nu > 0$:

$$K_T(\underline{x}) = \frac{2^{1-\nu}}{\Gamma(\nu)} z^\nu \mathcal{K}_\nu(z), \tag{25}$$

where $z = \sqrt{2\nu}\frac{\|\underline{x}\|}{\sigma}$ (σ being the kernel variance), Γ is the gamma function and $\mathcal{K}_\nu$ is the Bessel function of the second kind with parameter ν. Interestingly we recover the Laplace kernel for $\nu = 1/2$ and the Gaussian kernel for $\nu = \infty$. As one can find in e.g. [34], the exponent α_T that governs the decay at high frequency of this kernels is $\alpha_T = d + 2\nu$. Varying ν we can change the smoothness of the target function.

For $d = 1$ our prediction for the learning curve exponent β is

$$\beta = \frac{1}{d}\min(\alpha_T - d, 2\alpha_S) = \min(2\nu, 4). \tag{26}$$

In Fig. 8 we verify that our prediction matches the numerical results.

F Proof of theorem

We prove here Theorem 1:

Theorem 1 *Let $\tilde{K}_T(\underline{w}) = c_T\|\underline{w}\|^{-\alpha_T} + o\left(\|\underline{w}\|^{-\alpha_T}\right)$ and $\tilde{K}_S(\underline{w}) = c_S\|\underline{w}\|^{-\alpha_S} + o\left(\|\underline{w}\|^{-\alpha_S}\right)$ as $\|\underline{w}\| \to \infty$, where $\tilde{K}_T(\underline{w})$ and $\tilde{K}_S(\underline{w})$ are the Fourier transforms of the kernels $K_T(\underline{x})$, $K_S(\underline{x})$ respectively, assumed to be positive definite. We assume that $\tilde{K}_T(\underline{w})$ and $\tilde{K}_S(\underline{w})$ have a finite limit as $\|\underline{w}\| \to 0$ and that $K(\underline{0}) < \infty$. Then,*

$$\mathbb{E}\,\mathrm{MSE} = n^{-\beta} + o\left(n^{-\beta}\right) \quad \text{with} \quad \beta = \frac{1}{d}\min(\alpha_T - d, 2\alpha_S). \tag{27}$$

Moreover, in the case of a Gaussian kernel the result holds valid if we take the corresponding exponent to be $\alpha = \infty$.

Proof. Our strategy is to compute how the mean-square test error scales with distance δ between two nearest neighbors on the d-dimensional regular lattice. At the end, we will use the fact that $\delta \propto n^{-1/d}$, where n is the number of sampled points on the lattice.

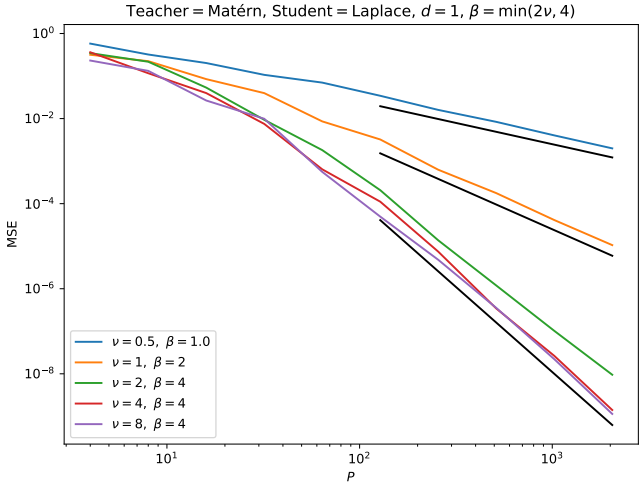


Figure 8: Mean-squared error for Matérn Teacher kernels and Laplace students. The variance of the kernels is equal to 2 for all the curves.

We denote by $\tilde{F}(\underline{w})$ the Fourier transform of a function $F : \mathcal{V} \to \mathbb{R}$:

$$\tilde{F}(\underline{w}) = L^{-d/2} \int_{\mathcal{V}} \mathrm{d}\underline{x}\, e^{-i\underline{w}\cdot\underline{x}} F(\underline{x}), \quad \text{where } \underline{w} \in \mathbb{L} \equiv \frac{2\pi}{L}\mathbb{Z}^d, \tag{28}$$

$$F(\underline{x}) = L^{-d/2} \sum_{\underline{w}\in\mathbb{L}} e^{i\underline{w}\cdot\underline{x}} \tilde{F}(\underline{w}). \tag{29}$$

If $Z \sim \mathcal{N}(0, K)$ is a Gaussian field with translation-invariant covariance K then by definition

$$\mathbb{E}Z(\underline{x})Z(\underline{x}') = K(\underline{x} - \underline{x}'). \tag{30}$$

Properties of the Fourier transform of a Gaussian field:

$$\tilde{K}(\underline{w}) = \tilde{K}(-\underline{w}) \in \mathbb{R}, \tag{31}$$

$$\mathbb{E}\tilde{Z}(\underline{w}) = 0, \tag{32}$$

$$\mathbb{E}\tilde{Z}(\underline{w})\overline{\tilde{Z}(\underline{w}')} = L^{d/2}\delta_{\underline{w}\underline{w}'}\tilde{K}(\underline{w}). \tag{33}$$

Eq. (31) comes from the fact that $K(\underline{x})$ is an even, real-valued function. The real and imaginary parts of $\tilde{Z}(\underline{w})$ are Gaussian random variables. They are all independent except that $\tilde{Z}(-\underline{w}) = \overline{\tilde{Z}(\underline{w})}$. Eq. (33) follows from the fact that $Z(\underline{x})$ and $K(\underline{x})$ are L-periodic functions, and therefore $e^{i\underline{w}\cdot\underline{x}}\tilde{K}(\underline{w})$ is the Fourier transform of $K(\cdot + \underline{x})$ if $\underline{w} \in \frac{2\pi}{L}\mathbb{Z}^d$. #

The solution Eq. (3) for kernel regression has two interpretations. In Section 4 we introduced it as the quantity that minimizes a quadratic error, but it can also be seen as the *maximum-a-posteriori* (MAP) estimation of another formulation of the problem [34]. The field $Z(\underline{x})$ is assumed to be drawn from a Gaussian distribution with covariance function $K_S(\underline{x})$: K_S therefore plays a role in the *prior* distribution of the data $\underline{Z} = (Z(\underline{x}_\mu)_{\mu=1}^n)$. Inference about the value of the field $\hat{Z}_S(\underline{x})$ at another location is then performed by maximizing its posterior distribution,

$$\hat{Z}_S(\underline{x}) \equiv \arg\max \mathcal{P}\left(Z(\underline{x})|\underline{Z}\right). \tag{34}$$

Such a posterior distribution is Gaussian, and its mean — and therefore also the value that maximizes the probability — is exactly Eq. (3):

$$\hat{Z}_S(\underline{x}) = \underline{k}_S(\underline{x}) \cdot \mathbb{K}_S^{-1}\underline{Z}, \tag{35}$$

where where $\underline{Z} = \left(Z(\underline{x}_\mu)\right)_{\mu=1}^n$ are the training data, $\underline{k}_S(\underline{x}) = \left(K_S(\underline{x}_\mu, \underline{x})\right)_{\mu=1}^n$ and $\mathbb{K}_S = \left(K_S(\underline{x}_\mu, \underline{x}_\nu)\right)_{\mu,\nu=1}^n$ is the Gram matrix, that is invertible since the kernel K_S is assumed to be positive definite. By Fourier transforming this relation we find

$$\tilde{Z}_S(\underline{w}) = \tilde{Z}^\star(\underline{w}) \frac{\tilde{K}_S(\underline{w})}{\tilde{K}_S^\star(\underline{w})}, \tag{36}$$

where we have defined $F^\star(\underline{w}) \equiv \sum_{\underline{n}\in\mathbb{Z}^d} F\left(\underline{w} + \frac{2\pi\underline{n}}{\delta}\right)$ for a generic function F.

Another way to reach Eq. (36) is to consider that we are observing the quantities

$$\tilde{Z}^\star(\underline{w}) \equiv \delta^d L^{-d/2} \sum_{\underline{x}\in\text{lattice}} e^{-i\underline{w}\cdot\underline{x}} Z(\underline{x}) \equiv \sum_{\underline{n}\in\mathbb{Z}^d} \tilde{Z}\left(\underline{w} + \frac{2\pi\underline{n}}{\delta}\right). \tag{37}$$

Given that we know the prior distribution of the Fourier components on the right-hand side in Eq. (37), we can infer their posterior distribution once their sums are constrained by the value of $\tilde{Z}^\star(\underline{w})$, and it is straightforward to see that we recover Eq. (36).

The mean-square error can then be written using the Parseval-Plancherel identity,

$$\mathbb{E}\,\text{MSE} = L^{-d}\mathbb{E}\int_{\mathcal{V}} \mathrm{d}\underline{x}\,[Z(\underline{x}) - \hat{Z}_S(\underline{x})]^2 = L^{-d}\mathbb{E}\sum_{\underline{w}\in\mathbb{L}} \left|\tilde{Z}(\underline{w}) - \tilde{Z}^\star(\underline{w})\frac{\tilde{K}_S(\underline{w})}{\tilde{K}_S^\star(\underline{w})}\right|^2. \tag{38}$$

By taking the expectation value with respect to the Teacher and using Eq. (31)-Eq. (33) we can write the mean-square error as

$$\begin{aligned}
\mathbb{E}\,\text{MSE} &= L^{-d}\mathbb{E}\sum_{\underline{w}\in\mathbb{L}}\left[\tilde{Z}(\underline{w})\overline{\tilde{Z}(\underline{w})} - 2\tilde{Z}(\underline{w})\overline{\tilde{Z}^\star(\underline{w})}\frac{\tilde{K}_S(\underline{w})}{\tilde{K}_S^\star(\underline{w})} + \tilde{Z}^\star(\underline{w})\overline{\tilde{Z}^\star(\underline{w})}\frac{\tilde{K}_S^2(\underline{w})}{\tilde{K}_S^\star{}^2(\underline{w})}\right] = \\
&= L^{-d}\mathbb{E}\sum_{\underline{w}\in\mathbb{L}}\tilde{Z}(\underline{w})\overline{\tilde{Z}(\underline{w})} - 2\frac{\tilde{K}_S(\underline{w})}{\tilde{K}_S^\star(\underline{w})}\sum_{\underline{n}\in\mathbb{Z}^d}\tilde{Z}(\underline{w})\overline{\tilde{Z}\left(\underline{w}+\frac{2\pi\underline{n}}{\delta}\right)} + \\
&\quad + \frac{\tilde{K}_S^2(\underline{w})}{\tilde{K}_S^\star{}^2(\underline{w})}\sum_{\underline{n},\underline{n}'\in\mathbb{Z}^d}\tilde{Z}\left(\underline{w}+\frac{2\pi\underline{n}}{\delta}\right)\overline{\tilde{Z}\left(\underline{w}+\frac{2\pi\underline{n}'}{\delta}\right)} = \\
&= L^{-d/2}\sum_{\underline{w}\in\mathbb{L}}\tilde{K}_T(\underline{w}) - 2\frac{\tilde{K}_S(\underline{w})}{\tilde{K}_S^\star(\underline{w})}\tilde{K}_T(\underline{w}) + \frac{\tilde{K}_S^2(\underline{w})}{\tilde{K}_S^\star{}^2(\underline{w})}\sum_{\underline{n}\in\mathbb{Z}^d}\tilde{K}_T\left(\underline{w}+\frac{2\pi\underline{n}}{\delta}\right) = \\
&= L^{-d/2}\sum_{\underline{w}\in\mathbb{L}\cap\mathcal{B}}\tilde{K}_T^\star(\underline{w}) - 2\frac{[\tilde{K}_T\tilde{K}_S]^\star(\underline{w})}{\tilde{K}_S^\star(\underline{w})} + \frac{\tilde{K}_T^\star(\underline{w})[\tilde{K}_S^2]^\star(\underline{w})}{\tilde{K}_S^\star(\underline{w})^2},
\end{aligned} \tag{39}$$

where $\mathcal{B} = \left[-\frac{\pi}{\delta}, \frac{\pi}{\delta}\right]^d$ is the Brillouin zone.

At high frequencies, $\tilde{K}_T(\underline{w}) = c_T\|\underline{w}\|^{-\alpha_T} + o\left(\|\underline{w}\|^{-\alpha_T}\right)$ and $\tilde{K}_S(\underline{w}) = c_S\|\underline{w}\|^{-\alpha_S} + o\left(\|\underline{w}\|^{-\alpha_S}\right)$. Therefore:

$$\begin{aligned}
\tilde{K}_T^\star(\underline{w}) &= \tilde{K}_T(\underline{w}) + \delta^{\alpha_T} c_T \sum_{\underline{n}\in\mathbb{Z}^d\setminus\{\underline{0}\}} \|\underline{w}\delta + 2\pi\underline{n}\|^{-\alpha_T} + o\left(\|\underline{w}\|^{-\alpha_T}\right) \equiv \\
&\equiv \tilde{K}_T(\underline{w}) + \delta^{\alpha_T} c_T\,\psi_T(\underline{w}\delta) + o\left(\|\underline{w}\|^{-\alpha_T}\right).
\end{aligned} \tag{40}$$

This equation defines the function ψ_T, and a similar equation holds for the Student as well. The hypothesis $K_T(\underline{0}) \propto \int \mathrm{d}\underline{w}\,\tilde{K}_T(\underline{w}) < \infty$ implies $\alpha_T > d$ and therefore $\sum_{\underline{n}\in\mathbb{Z}^d}\|\underline{n}\|^{-\alpha_T} < \infty$ (and likewise for the Student). Then, $\psi_{\alpha_T}(\underline{0}), \psi_{\alpha_S}(\underline{0})$ are finite; furthermore, the $\underline{w}$'s in the sum Eq. (39) are at most of order $\mathcal{O}\left(\delta^{-1}\right)$, therefore the terms $\psi_\alpha(\underline{w}\delta)$ are $\mathcal{O}(\delta^0)$ and do not influence how Eq. (39) scales with δ. Applying Eq. (40), expanding for $\delta \ll 1$ and keeping only the leading orders, we find

$$\begin{aligned}
&\mathbb{E}\,\text{MSE} = \\
&= L^{-d/2}\left[\sum_{\underline{w}\in\mathbb{L}\cap\mathcal{B}} 2c_T\psi_{\alpha_T}(\underline{w}\delta)\delta^{\alpha_T} + c_S^2\psi_{2\alpha_S}(\underline{w}\delta)\frac{\tilde{K}_T(\underline{w})}{\tilde{K}_S^2(\underline{w})}\delta^{2\alpha_S} + o\left(\|\underline{w}\|^{-\alpha_T}\right) + o\left(\|\underline{w}\|^{-2\alpha_S}\right)\right] = \\
&= L^{-d/2}\left[\sum_{\underline{w}\in\mathbb{L}\cap\mathcal{B}} 2c_T\psi_{\alpha_T}(\underline{w}\delta)\delta^{\alpha_T} + c_S^2\psi_{2\alpha_S}(\underline{w}\delta)\frac{\tilde{K}_T(\underline{w})}{\tilde{K}_S^2(\underline{w})}\delta^{2\alpha_S}\right] + o\left(\|\underline{w}\|^{-\alpha_T-d}\right) + o\left(\|\underline{w}\|^{-2\alpha_S-d}\right).
\end{aligned} \tag{41}$$

We have neglected terms proportional to, for instance, $\delta^{\alpha_T+\alpha_S}$, since they are subleading with respect to δ^{α_T}, but we must keep both δ^{α_T} and δ^{α_S} since we do not know a priori which one is dominant. The additional term δ^{-d} in the subleading terms comes from the fact that $|\mathbb{L}\cap\mathcal{B}| = \mathcal{O}\left(\delta^{-d}\right)$.

The first term in Eq. (41) is the simplest to deal with: since $\|\underline{w}\delta\|$ is smaller than some constant for all $\underline{w}\in\mathbb{L}\cap\mathcal{B}$ and the function $\psi_{\alpha_T}(\underline{w}\delta)$ has a finite limit, we have

$$\delta^{\alpha_T}\sum_{\underline{w}\in\mathbb{L}\cap\mathcal{B}} 2c_T\psi_{\alpha_T}(\underline{w}\delta) = \mathcal{O}\left(\delta^{\alpha_T}|\mathbb{L}\cap\mathcal{B}|\right) = \mathcal{O}\left(\delta^{\alpha_T-d}\right). \tag{42}$$

We then split the second term in Eq. (41) in two contributions:

Small $\|\underline{w}\|$ We consider "small" all the terms $\underline{w}\in\mathbb{L}\cap\mathcal{B}$ such that $\|\underline{w}\|<\Gamma$, where $\Gamma\gg 1$ is $\mathcal{O}(\delta^0)$ but large. As $\delta\to 0$, $\psi_{2\alpha_S}(\underline{w}\delta)\to\psi_{2\alpha_S}(\underline{0})$ which is finite because $K(\underline{0})<\infty$. Therefore

$$\delta^{2\alpha_S}\sum_{\substack{\underline{w}\in\mathbb{L}\cap\mathcal{B}\\ \|\underline{w}\|<\Gamma}} c_S^2\psi_{2\alpha_S}(\underline{w}\delta)\frac{\tilde{K}_T(\underline{w})}{\tilde{K}_S^2(\underline{w})} \to \delta^{2\alpha_S}c_S^2\psi_{2\alpha_S}(\underline{0})\sum_{\substack{\underline{w}\in\mathbb{L}\cap\mathcal{B}\\ \|\underline{w}\|<\Gamma}}\frac{\tilde{K}_T(\underline{w})}{\tilde{K}_S^2(\underline{w})}. \tag{43}$$

The summand is real and strictly positive because the positive definiteness of the kernels implies that their Fourier transforms are strictly positive. Moreover, as $\delta\to 0$, $\mathbb{L}\cap\mathcal{B}\cap\{\|\underline{w}\|<\Gamma\}\to\mathbb{L}\cap\{\|\underline{w}\|<\Gamma\}$, which contains a finite number of elements, independent of δ. Therefore

$$\delta^{2\alpha_S}\sum_{\substack{\underline{w}\in\mathbb{L}\cap\mathcal{B}\\ \|\underline{w}\|<\Gamma}} c_S^2\psi_{2\alpha_S}(\underline{w}\delta)\frac{\tilde{K}_T(\underline{w})}{\tilde{K}_S^2(\underline{w})} = \mathcal{O}\left(\delta^{2\alpha_S}\right). \tag{44}$$

Large $\|\underline{w}\|$ "Large" $\underline{w}$ are those with $\|\underline{w}\|>\Gamma$: we recall that $\Gamma\gg 1$ is $\mathcal{O}(\delta^0)$ but large. This allows us to approximate $\tilde{K}_T$, $\tilde{K}_S$ in the sum with their asymptotic behavior:

$$\begin{aligned}\delta^{2\alpha_S}\sum_{\substack{\underline{w}\in\mathbb{L}\cap\mathcal{B}\\ \|\underline{w}\|>\Gamma}} c_S^2\psi_{2\alpha_S}(\underline{w}\delta)\frac{\tilde{K}_T(\underline{w})}{\tilde{K}_S^2(\underline{w})} &\propto \delta^{2\alpha_S}\sum_{\substack{\underline{w}\in\mathbb{L}\cap\mathcal{B}\\ \|\underline{w}\|>\Gamma}}\|\underline{w}\|^{-\alpha_T+2\alpha_S}+o\left(\|\underline{w}\|^{-\alpha_T+2\alpha_S}\right)\approx\\ &\approx \delta^{2\alpha_S}\int_\Gamma^{1/\delta}\mathrm{d}w\, w^{d-1-\alpha_T+2\alpha_S}+o\left(\|\underline{w}\|^{-\alpha_T+2\alpha_S}\right)=\mathcal{O}\left(\delta^{\min(\alpha_T-d,2\alpha_S)}\right).\end{aligned} \tag{45}$$

Finally, putting Eq. (42), Eq. (44) and Eq. (45) together,

$$\mathbb{E}\,\mathrm{MSE} = \mathcal{O}\left(\delta^{\min(\alpha_T-d,2\alpha_S)}\right). \tag{46}$$

The proof is concluded by considering that $\delta=\mathcal{O}\left(n^{-1/d}\right)$.

In the case of a Gaussian kernel $K(\underline{x})\propto\exp(-\|\underline{x}\|^2/(2\sigma^2))$ — and therefore $\tilde{K}(\underline{w})\propto\exp(-\sigma^2\|\underline{w}\|^2/2)$ — one has to redo the calculations starting from Eq. (39), but the final result can be easily recovered by taking the limit $\alpha\to+\infty$ (Gaussian kernels decay faster than any power law).

□

G Proofs of lemmas

Lemma 1 *Let $K(\underline{x},\underline{x}')$ be a translation-invariant isotropic kernel such that $\tilde{K}(\underline{w}) = c\|\underline{w}\|^{-\alpha} + o\left(\|\underline{w}\|^{-\alpha}\right)$ as $\|w\|\to\infty$ and $\|\underline{w}\|^d\tilde{K}(\underline{w})\to 0$ as $\|w\|\to 0$. If $\alpha>d+n$ for some $n\in\mathbb{Z}^+$, then $K(\underline{x})\in C^n$, that is, it is at least n-times differentiable.*

Proof. The kernel is rotational invariant in real space ($K(\underline{x})=K(\|\underline{x}\|)$) and therefore also in the frequency domain. Then, calling $\hat{\epsilon}_1=(1,0,\dots)$ the unitary vector along the first dimension x_1,

$$K(x)\propto\int\mathrm{d}\underline{w}\, e^{i\underline{w}\cdot\hat{\epsilon}_1 x}\tilde{K}(\|\underline{w}\|). \tag{47}$$

It follows that

$$|\partial^m K(x)| \propto \left| \int \mathrm{d}\underline{w}\, (\underline{w}\cdot\hat{\epsilon}_1)^m e^{i\underline{w}\cdot\hat{\epsilon}_1 x} \tilde{K}(\|\underline{w}\|) \right| < \int \mathrm{d}\underline{w}\, |\underline{w}\cdot\hat{\epsilon}_1|^m |\tilde{K}(\|\underline{w}\|)| \propto$$
$$\propto \int_0^\infty \mathrm{d}w\, w^{d-1+m} |\tilde{K}(w)| \int_0^\pi \mathrm{d}\phi_1 |\cos(\phi_1)|^m \propto \int_0^\infty \mathrm{d}w\, w^{d-1+m} |\tilde{K}(w)|. \quad (48)$$

We want to claim that this quantity is finite if $m \leq n$. Convergence at infinity requires $m < \alpha - d$, that is always smaller than or equal to n because of the hypothesis of the lemma. Convergence in zero requires that $w^{d+m}|\tilde{K}(w)| \to 0$, and we want this to hold for all $0 \leq m < \alpha - d$, the most constraining one being the condition with $m = 0$. □

Lemma 2 *Let $Z \sim \mathcal{N}(0, K)$ be a d-dimensional Gaussian random field, with $K \in C^{2n}$ being a $2n$-times differentiable kernel. Then Z is n-times differentiable in the sense that*

- *derivatives of $Z(\underline{x})$ are a Gaussian random fields;*
- $\mathbb{E}\partial_{x_1}^{n_1} \cdots \partial_{x_d}^{n_d} Z(\underline{x}) = 0$*;*
- $\mathbb{E}\partial_{x_1}^{n_1} \cdots \partial_{x_d}^{n_d} Z(\underline{x}) \cdot \partial_{x_1}^{n_1'} \cdots \partial_{x_d}^{n_d'} Z(\underline{x}') = \partial_{x_1}^{n_1+n_1'} \cdots \partial_{x_d}^{n_d+n_d'} K(\underline{x}-\underline{x}') < \infty$ *if the derivatives of K exist.*

In particular, $\mathbb{E}\partial_{x_i}^m Z(\underline{x}) \cdot \partial_{x_i}^m Z(\underline{x}') = \partial_{x_i}^{2m} K(\underline{x}-\underline{x}') < \infty \ \forall m \leq n$.

Proof. Derivatives of $Z(\underline{x})$ are defined as limits of sums and differences of the field Z evaluated at different points, therefore they are Gaussian random fields too, and furthermore it is straightforward to see that their expected value is always 0 if the field itself is zero centered.

The correlation can be computed via induction. Assume that $\mathbb{E}\partial_{x_1}^{n_1} \cdots \partial_{x_d}^{n_d} Z(\underline{x}) \cdot \partial_{x_1}^{n_1'} \cdots \partial_{x_d}^{n_d'} Z(\underline{x}') = \partial_{x_1}^{n_1+n_1'} \cdots \partial_{x_d}^{n_d+n_d'} K(\underline{x}-\underline{x}')$ holds true. Then, if we increment n_1:

$$\mathbb{E}\partial_{x_1}^{n_1+1} \cdots \partial_{x_d}^{n_d} Z(\underline{x}) \cdot \partial_{x_1}^{n_1'} \cdots \partial_{x_d}^{n_d'} Z(\underline{x}') =$$
$$= \lim_{h\to 0} h^{-1} \mathbb{E}\left[\partial_{x_1}^{n_1} \cdots \partial_{x_d}^{n_d} Z(\underline{x}+h\hat{\epsilon}_1) - \partial_{x_1}^{n_1} \cdots \partial_{x_d}^{n_d} Z(\underline{x})\right] \cdot \partial_{x_1}^{n_1'} \cdots \partial_{x_d}^{n_d'} Z(\underline{x}') =$$
$$= \lim_{h\to 0} h^{-1} \left[\partial_{x_1}^{n_1+n_1'} \cdots \partial_{x_d}^{n_d+n_d'} K(\underline{x}-\underline{x}'+h\hat{\epsilon}_1) - \partial_{x_1}^{n_1+n_1'} \cdots \partial_{x_d}^{n_d+n_d'} K(\underline{x}-\underline{x}')\right] =$$
$$= \partial_{x_1}^{n_1+1+n_1'} \cdots \partial_{x_d}^{n_d+n_d'} K(\underline{x}-\underline{x}'). \quad (49)$$

Of course by symmetry the same can be said about the increase of any other exponent. To conclude the induction proof we simply recall that by definition $\mathbb{E}Z(\underline{x})Z(\underline{x}') = K(\underline{x}-\underline{x}')$. □

H RKHS hypothesis ans smoothness

It is important to note the high degree of smoothness underlying the RKHS hypothesis. Consider for instance realizations $Z(\underline{x})$ of a Teacher Gaussian process with covariance K_T and assume that they lie in the RKHS of the Student kernel K_S (notice that they never belong to the RKHS of the same kernel K_T), namely

$$\mathbb{E}\|Z\|_{K_S}^2 = \mathbb{E}\int \mathrm{d}\underline{x}\mathrm{d}y Z(\underline{x}) K_S^{-1}(\underline{x}-\underline{y}) Z(\underline{y}) = \int \mathrm{d}\underline{w}\, \tilde{K}_T(\underline{w}) \tilde{K}_S^{-1}(\underline{w}) < \infty. \quad (50)$$

If the Teacher and Student kernels decay in the frequency domain with exponents α_T and α_S respectively, convergence requires $\alpha_T > \alpha_S + d$, and $K_S(\underline{0}) \propto \int \mathrm{d}\underline{w}\, \tilde{K}_S(\underline{w}) < \infty$ (true for many commonly used kernels) implies $\alpha_S > d$. Then using Lemma 1 and Lemma 2 we can conclude that the realizations $Z(\underline{x})$ must be at least $\lfloor d/2 \rfloor$-times mean-square differentiable to be RKHS.

I Asymptotic limit of the PDE approximation

In this appendix we show how to recover the prediction of Theorem 1 using the approach presented in [11], that can be applied to a generic target function (not necessarily Gaussian nor evaluated only on a regular lattice). They derive their main formula, that we write below, by computing the amount of generalization error due to each eigenmode of the (Student) kernel. In order to carry out the calculations they introduce a partial differential equation that they solve with two different approximations.

In the following we denote by $\lambda_1 \geq \cdots \geq \lambda_\rho \geq \cdots$ the eigenvalues of the kernel, and by $\phi_\rho(\underline{x})$ the corresponding eigenfunctions. In [11] they show that the generalization error can be written as:

$$\mathbb{E}\text{MSE} = \sum_\rho E_\rho(n), \tag{51}$$

$$E_\rho(n) = \sum_\rho \frac{\mathbb{E}w_\rho^2}{\lambda_\rho}\left(\frac{1}{\lambda_\rho} + \frac{n}{t(n)}\right)^{-2}\left(1 - \frac{n\gamma(n)}{t(n)^2}\right)^{-1}, \tag{52}$$

$$\gamma(n) = \sum_\rho \frac{\lambda_\rho^2}{\left(1 + \lambda_\rho \frac{n}{t(n)}\right)^2}, \tag{53}$$

$$t(n) = \sum_\rho \frac{\lambda_\rho}{1 + \lambda_\rho \frac{n}{t(n)}}. \tag{54}$$

The term $\mathbb{E}w_\rho^2$ is the variance of the coefficients of the target function in the kernel eigenbasis, defined as:

$$w_\rho = \lambda_\rho^{-1/2} \langle Z, \phi_\rho \rangle, \tag{55}$$

(the factor in front of the scalar product is to keep our notation consistent with that of [11]), to help the reader compare the two works. Notice that the variance can be computed with respect to an ensemble of target functions, but this ensemble may contain one deterministic function only.

In order to compute sums over the eigenmodes we will always replace them with integrals over eigenvalues. To do so, we must also introduce a density of eigenvalues $\mathcal{D}(\lambda)$: $\sum_\rho f(\lambda_\rho) \to \int \mathrm{d}\lambda \mathcal{D}(\lambda) f(\lambda)$. The asymptotic behavior of this density for small eigenvalues can be derived as follows (for a given kernel whose Fourier transform decays with an exponent α):

$$\mathcal{D}(\lambda) = \sum_\rho \delta(\lambda - \lambda_\rho) \sim \int \mathrm{d}^d \underline{w}\, \delta(\lambda - \|\underline{w}\|^{-\alpha}) \sim \int_0^\infty \mathrm{d}w\, w^{d-1} \delta(\lambda - w^{-\alpha}) = \lambda^{-\theta}, \tag{56}$$

where we have defined the exponent $\theta \equiv 1 + \frac{d}{\alpha}$. Notice that $1 < \theta < 2$, and that of course this exponent depends on the kernel. We can use this density also to derive a scaling behavior of small eigenvalues: indeed, the ρ-th ($\gg 1$) eigenvalue can be estimated by

$$\rho \sim \int_{\lambda_\rho}^{\lambda_1} \mathrm{d}\lambda\, \mathcal{D}(\lambda) \sim \int_{\lambda_\rho}^{\lambda_1} \mathrm{d}\lambda \lambda^{-\theta} \sim \lambda_\rho^{-(\theta-1)}. \tag{57}$$

The last equation follows from the fact that $\lambda_\rho \ll \lambda_1$ and that $\theta > 1$.

We now have to estimate the asymptotic behavior of the implicitly defined function $t(n)$. It is easy to see that this function must go to 0 as $n \to \infty$, therefore we can assume it small. Splitting the integral according to whether the denominator in the definition of $t(n)$ is dominated by the first or second term,

$$t(n) \sim \int \mathrm{d}\lambda\, \lambda^{-\theta} \frac{\lambda}{1 + \lambda \frac{n}{t(n)}} \sim \int_0^{\frac{t(n)}{n}} \mathrm{d}\lambda\, \lambda^{-\theta} + \int_{\frac{t(n)}{n}}^{\lambda_1} \mathrm{d}\lambda\, \lambda^{-\theta} \frac{t(n)}{n} \sim \left(\frac{t(n)}{n}\right)^{2-\theta}. \tag{58}$$

Therefore, $t(n) \sim n^{-\frac{2-\theta}{\theta-1}}$, and with a similar approximation we can also deduce that $\gamma(n) \sim n^{-\frac{3-\theta}{\theta-1}}$. Injecting all we know in the formula for the generalization error and splitting the integral we find

$$\mathbb{E}\text{MSE} \sim \sum_\rho \frac{\mathbb{E}w_\rho^2}{\lambda_\rho}\left(\frac{1}{\lambda_\rho} + n^{\frac{1}{\theta-1}}\right)^{-2} (1 - n^0)^{-1} \sim \sum_\rho \frac{\mathbb{E}w_\rho^2}{\lambda_\rho}\left(\frac{1}{\lambda_\rho} + \frac{1}{\lambda_n}\right)^{-2} \sim \lambda_n^2 \sum_{\rho \leq n} \frac{\mathbb{E}w_\rho^2}{\lambda_\rho} + \sum_{\rho > n} \mathbb{E}w_\rho^2 \lambda_\rho. \tag{59}$$

In the second equality we have used the fact that $\lambda_n \sim n^{-\frac{1}{\theta-1}}$ to introduce the n-th eigenvalue into the formula. Then, we approximated the sum by splitting it in two sums, one over the first n eigenvalues

($\rho \leq n$, therefore $\lambda_\rho \geq \lambda_n$) and one over the remaining ones ($\rho > n$). Notice that the second sum is indeed the sum in Eq. (12).

Next we assume that $\mathbb{E}w_\rho^2$ behaves asymptotically as a power law with respect to small eigenvalues, $\mathbb{E}w_\rho^2 \sim \lambda_\rho^q$, with an exponent q that can be either positive or negative. We can now compute each of the integrals in the previous equation:

$$\lambda_n^2 \sum_{\rho \leq n} \frac{\mathbb{E}w_\rho^2}{\lambda_\rho} \sim \lambda_n^2 \int_{\lambda_n}^{\lambda_1} \mathrm{d}\lambda\, \lambda^{-\theta} \lambda^{q-1}, \tag{60}$$

$$\sum_{\rho > n} \mathbb{E}w_\rho^2 \lambda_\rho \sim \int_0^{\lambda_n} \mathrm{d}\lambda\, \lambda^{-\theta} \lambda^{q+1} \sim \lambda_n^{q-\theta+2} \sim n^{-\frac{q-\theta+2}{\theta-1}}. \tag{61}$$

For the second integral to converge we have assumed that the exponent q is larger than $\theta - 1$. The first integral behaves differently according to whether $q > \theta$ or not: if $q > \theta$, the integral scales as $\lambda_n^2 \sim n^{-\frac{2}{\theta-1}}$; if $q < \theta$, then it scales as $\lambda_n^{2-\theta+q} \sim n^{-\frac{q-\theta+2}{\theta-1}}$. Therefore,

$$\mathbb{E}\,\mathrm{MSE} \sim n^{-\frac{\min(q-\theta,0)+2}{\theta-1}}. \tag{62}$$

A consequence of Eq. (60) and Eq. (61) is that if $q < \theta$ (which always occur if the student is smooth enough, so that α characterizing the decay of the Fourier coefficient is small and θ is large), then the scaling of the generalization error is given by Eq. (61) alone, and we recover Eq. (12) from Eq. (59), justifying why this equation applies to real, non-Gaussian data.

Notice that if the target function is generated by a Teacher Gaussian process, the exponent q takes the value $\frac{\theta-\theta_T}{\theta_T-1}$, where $\theta_T = 1 + \frac{d}{\alpha_T}$ and α_T is the exponent characterising the decay of the Fourier transform of the Teacher kernel. With some manipulations we then recover our Theorem 1

$$\mathbb{E}\mathrm{MSE} \sim n^{-\frac{1}{d}\min(\alpha_T - d, 2\alpha_S)}. \tag{63}$$

J Convergence of the spectrum of the Gram matrix

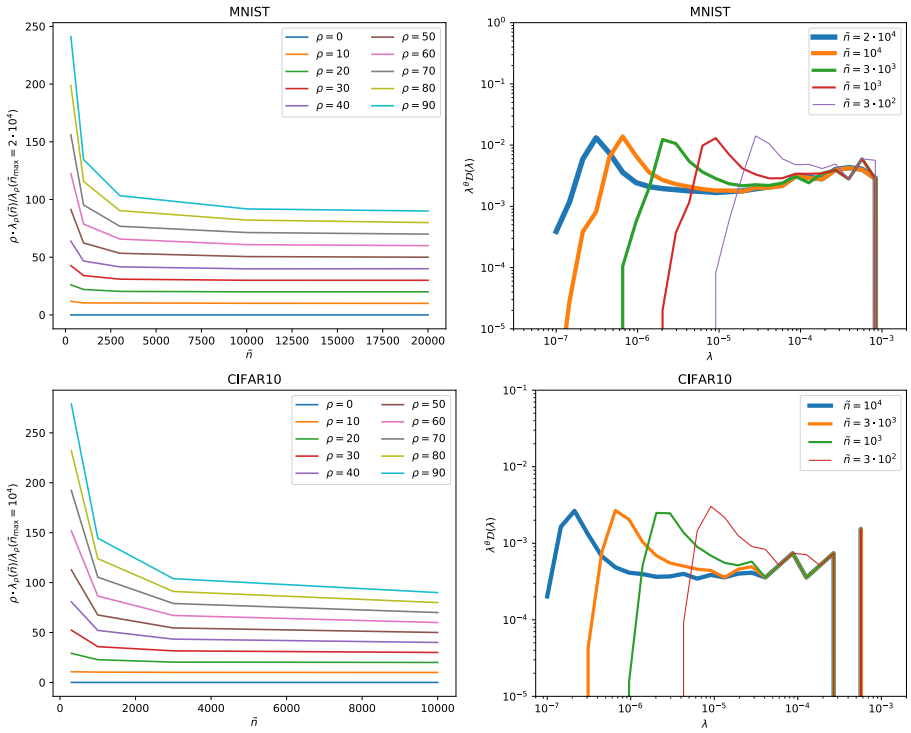


Figure 9: **Left:** we plot the first eigenvalues λ_ρ of Gram matrices of size $\tilde{n}$, rescaled by the corresponding eigenvalue of the largest Gram matrix (top row is MNIST, bottom row is CIFAR10). Increasing $\tilde{n}$ the eigenvalues are expected to converge, and indeed these ratios asymptote to one. We are plotting one eigenvalue every 10 for the first 100 eigenvalues. In order to make the plot clearer we have multiplied each curve by a factor ρ, equal to the eigenvalue index. **Right:** Density of eigenvalues of the Gram matrix, for several sizes $\tilde{n}$, for MNIST (top) and CIFAR10 (bottom). The density is divided by the predicted asymptotic behavior $\left(\lambda_\rho^S\right)^{-\theta}$, with $\theta = 1 + \frac{\alpha}{d_{\text{eff}}}$. For a Laplace kernel $\alpha_S = d_{\text{eff}} + 1$ and for the effective dimension we used the values extracted in Section 7, resulting in $\theta \approx 1.937$ for MNIST and $\theta \approx 1.972$ for CIFAR10. This plot shows that the density of eigenvalues converges when $\tilde{n}$ increases, and that the predicted power law is consistent with observations.

7 Loss landscape in a simple model of reinforcement learning

Candidate contributions The candidate contributed by participating to all the discussions, performing the numerical experiments, some of the calculations and by writing the mathematical content of the article.

Under review as a conference paper at ICLR 2022

HOW MEMORY ARCHITECTURE AFFECTS LEARNING IN A SIMPLE POMDP: THE TWO-HYPOTHESIS TESTING PROBLEM

Mario Geiger
Institute of Physics
École Polytechnique Fédéral de Lausanne
1015 Lausanne, Switzerland
mario.geiger@epfl.ch

Christophe Eloy
Institut de Recherche sur les Phenomènes Hors Équilibres
Centrale Marseille
CNRS
Aix-Marseille Université
13013 Marseille, France
christophe.eloy@irphe.univ-mrs.fr

Matthieu Wyart
Institute of Physics
École Polytechnique Fédéral de Lausanne
1015 Lausanne, Switzerland
matthieu.wyart@epfl.ch

ABSTRACT

Reinforcement learning is generally difficult for partially observable Markov decision processes (POMDPs), which occurs when the agent's observation is partial or noisy. To seek good performance in POMDPs, one strategy is to endow the agent with a finite memory, whose update is governed by the policy. However, policy optimization is non-convex in that case and can lead to poor training performance for random initialization. The performance can be empirically improved by constraining the memory architecture, then sacrificing optimality to facilitate training. Here we study this trade-off in a two-hypothesis testing problem, akin to the two-arm bandit problem. We compare two extreme cases: (i) the random access memory where any transitions between M memory states are allowed and (ii) a fixed memory where the agent can access its last m actions and rewards. For (i), the probability q to play the worst arm is known to be exponentially small in M for the optimal policy. Our main result is to show that similar performance can be reached for (ii) as well, despite the simplicity of the memory architecture: using a conjecture on Gray-ordered binary necklaces, we find policies for which q is exponentially small in 2^m, i.e. $q \sim \alpha^{2^m}$ with $\alpha < 1$. In addition, we observe empirically that training from random initialization leads to very poor results for (i), and significantly better results for (ii) thanks to the constraints on the memory architecture.

1 INTRODUCTION

Reinforcement learning is aimed at finding the sequence of actions that should take an agent to maximise a long-term reward (Sutton & Barto (2018)). This sequential decision-making is usually modeled as a Markov decision process (MDP): at each time step, the agent chooses an action based on a policy (a function that relates the agent's state to its action), with the aim of maximizing its value (the expected discounted sum of rewards). Deterministic optimal policies can be found through dynamic programming (Bellman (1966)) when MDPs are discrete (both states and actions belong to discrete sets) and the agent fully knows its environment (Watkins & Dayan (1992)).

A practical difficulty arises when the agent only have a partial observation of its environment or when this observation is imperfect or stochastic. The mathematical framework is then known as a partially observable Markov decision process (POMDP) (Smallwood & Sondik (1973)). In this framework, the agent's state is replaced by the agent's belief, which is the probability distribution

over all possible states. At each time step, the agent's belief can be updated through Bayesian inference to account for observations. In the belief space, the problem becomes fully observable again and the POMDP can thus be solved as a "belief MDP". However, the dimension of the belief space is much larger than the state space and solving the belief MDP can be challenging in practical problems. Some approaches seek to resolve this difficulty by approximating of the belief and the value functions (Hauskrecht (2000); Roy et al. (2005); Silver & Veness (2010); Somani et al. (2013)), or use deep model-free reinforcement learning where the neural network is complemented with a memory (Oh et al. (2016); Khan et al. (2017)) or a recurrency (Hausknecht & Stone (2015); Li et al. (2015)) to better approximate history-based policies.

Here we focus on the idea of Littman (1993), who proposed to give the agent a limited number of bits of memory, an idea that has been developed independently in the robotics community where it is known as a finite-state controller (Meuleau et al. (1999; 2013)). These works show that adding a memory usually increases the performance in POMDPs. But to this day, attempts to find optimal memory allocation have been essentially empirical (Peshkin et al. (2001); Zhang et al. (2016); Toro Icarte et al. (2020)). One central difficulty is that the value is a non-convex function of policy for POMDPs (Jaakkola et al. (1995)): learning will thus generally get stuck in poor local maxima for random policy initialization. This problem is even more acute when memory is large or when all transitions between memory states are allowed. To improve learning, restricting the policy space to specific memory architectures where most transitions are forbidden is key (Peshkin et al. (2001); Zhang et al. (2016); Toro Icarte et al. (2020)). However, there is no general principles to optimize the memory architectures or the policy initialization. In fact, this question is not understood satisfyingly even in the simplest tasks- arguably a necessary step to later achieve a broad understanding.

Here, we work out how the memory architecture affects optimal solutions in perhaps the simplest POMDP, and find that these solutions are intriguingly complex. Specifically, we consider the two-hypothesis testing problem. At each time step, the agent chooses to pull one of two arms that yield random rewards with different means. We compare two memory structures: (i) a random access memory (RAM) in which all possible transitions between M distinct memory states are allowed; (ii) a Memento memory in which the agent can access its last m actions and rewards.

When the agent is provided with a RAM memory, we study the performance of a "column of confidence" policy (CCP): the agent keeps repeating the same action and updates its confidence in it by moving up and down the memory sites until it reaches the bottom of the column and the alternative action is tried. The performance of this policy is assessed through the calculation of the expected frequency q to play the worst arm (thus the smaller q, the better). For the CCP, q can be shown to be exponentially small in M. This result is closely related to the work of Hellman & Cover (1970) on hypothesis testing and its extension to finite horizon (Wilson (2014)). In practice, we find that learning a policy with a RAM memory and random initialization leads to poor results, far from the performance of the column of confidence policy. Restricting memory transitions to chain-like transitions leads to much better results, although still sub-optimal.

Our main findings concerns the Memento memory architecture. Surprisingly, despite the lack of flexibility of the memory structure, excellent policies exist. Specifically, using a conjecture on Gray-ordered binary necklaces (Degni & Drisko (2007)), we find a policy for which q is exponentially small in 2^m —which is considerably better than $q \sim \ln(m)/m$, optimal for an agent that only plays m times. For Memento memory, we also observe empirically that learning is faster and perform better than in the RAM case.

The code to reproduce the experiments is available at `https://anonymous.4open.science/r/two-hypothesis-BAB3`, and uses a function defined here `https://anonymous.4open.science/r/gradientflow/gradientflow`. The experiments where executed on CPUs for about 10 thousand CPU hours.

2 POMDPs AND THE TWO-HYPOTHESIS TESTING PROBLEM

2.1 GENERAL FORMULATION

Definition 2.1 (POMDP)**.** A discrete-time POMDP model is defined as the 8-tuple $(S, A, T, R, \Omega, O, p_0, \gamma)$: S is a set of states, A is a set of actions, T is a conditional transition

probability function $T(s'|s,a)$ where $s', s \in S$ and $a \in A$, $R : S \to \mathbb{R}$ is the reward function[1], Ω is a set of observations, $O(o|s)$ is a conditional observation probability with $o \in \Omega$ and $s \in S$, $p_0(s) : S \to \mathbb{R}$ is the probability to start in a given state s, and $\gamma \in [0, 1)$ is the discount factor.

A state $s \in S$ specifies everything about the world at a given time (the agent, its memory and all the rest). The agent starts its journey in a state $s \in S$ with probability $p_0(s)$. Based on an observation $o \in \Omega$ obtained with probability $O(o|s)$ the agent takes an action $a \in A$. This action causes a transition to the state s' with probability $T(s'|s,a)$ and the agent gains the reward $R(s)$. And so on.

Definition 2.2 (Policy)**.** A policy $\pi(a|o)$ is a conditional probability of executing an action $a \in A$ given an observation $o \in \Omega$.

Definition 2.3 (Policy State Transition)**.** Given a policy π, the state transition T_π is given by

$$T_\pi(s'|s) = \sum_{o,a \in \Omega \times A} T(s'|s,a)\pi(a|o)O(o|s). \tag{1}$$

Definition 2.4 (Expected sum of discounted rewards)**.** The expected sum of future discounted rewards of a policy π is

$$G_\pi = \mathbb{E}_{\substack{s_0 \sim p_0 \\ s_1 \sim T_\pi(\cdot|s_0) \\ s_2 \sim T_\pi(\cdot|s_1) \\ \dots}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \right]. \tag{2}$$

Note that a POMDP with an expected sum of future discounted rewards with discount factor γ can be reduced to an undiscounted POMDP (Altman (1999)), as we now recall (see proof in Appendix A):

Lemma 2.1. *The discounted POMDP defined in 2.1 with a discount γ is equivalent to an undiscounted POMDP with a probability $r = 1 - \gamma$ to be reset from any state toward an initial state. In the undiscounted POMDP, the agent reaches a steady state $p(s)$ which can be used to calculate the expected sum of discounted rewards $G_\pi = \frac{1}{r}\mathbb{E}_{s\sim p}[R(s)]$.*

2.2 Optimization algorithm

To optimize a policy algorithmically, we apply gradient descent on the expected sum of discounted rewards. First, we parametrize a policy with parameters $w \in \mathbb{R}^{|A|\times|\Omega|}$, normalized to get a probability using the softmax function $\pi_w(a|o) = \frac{\exp(w_{ao})}{\sum_b \exp(w_{bo})}$. Then, we compute the transition matrix $\tilde{T}_\pi$, from which we obtain the steady state p using the power method (See Appendix B). Finally, we calculate G_π by the Lemma 2.1.

Using an algorithm that keeps track of the operations (we use `pytorch` Paszke et al. (2017)), we can compute the gradient of G_π with respect to the parameters w and perform gradient descent with adaptive time steps (i.e. a gradient flow dynamics):

$$\frac{d}{dt}w = \frac{d}{dw}G_\pi(w). \tag{3}$$

2.3 Two-hypothesis testing problem

The problem we consider is the two-hypothesis testing problem. We label two arms by the letters A and B. The two arms gives a reward of $+1$ or -1 with a Bernoulli distribution. The probabilities to obtain a positive reward are noted k_A and k_B respectively. The environment is entirely defined by the couple (k_A, k_B). With equal probability, the environment is in one of the following two configurations (hypothesis):

$$\begin{cases} k_A = \frac{1+\mu}{2} & k_B = \frac{1-\mu}{2} \quad \text{(hypothesis } H_A\text{)} \\ k_A = \frac{1-\mu}{2} & k_B = \frac{1+\mu}{2} \quad \text{(hypothesis } H_B\text{)} \end{cases} \tag{4}$$

where the hypothesis H_A (resp. H_B) corresponds to A (resp. B) being the best arm. In expectation over the environments, an agent that plays randomly or always the same arm will have a reward 0.

[1] In the literature, R also depends on the action: $R : S \times A \to \mathbb{R}$. Our notation is not a loss of generality. The set of state can be made bigger $S \to S \times A$ in order to contain the last action.

Note that this problem is similar to the Bandit problem, except that in the latter (k_A, k_B) can take any value in the square $[0, 1]^2$. When $r \to 0$, our results below can be generalized to the bandit problem, as done in Cover & Hellman (1970) by recasting the latter as finding the correct hypothesis ('Arm A is better' or 'Arm B is better').

In our setup, the agent only knows its last arm played, reward obtained (if there were some) and the state of its memory (different memories are described below). Based on that, it chooses an arm (play A or B) and how to update its memory state.

In the POMDP formalism (c.f. 2.1), the state s contains the environment (k_A, k_B), the memory state, the last arm played and reward obtained. We only consider agents that have a complete access to their memory, therefore O is deterministic and simply projects s by removing the environment information.

$$\begin{aligned} s &= \text{environment } H_A \text{ or } H_B), \text{ memory state, last arm played (A or B) and last reward } (1 \text{ or } -1) \\ o &= \text{memory state, last arm played and last reward} \\ a &= \text{arm to play, memory update} \end{aligned} \tag{5}$$

We define the function $q(s)$, which is 1 if the agent just played the "wrong" arm in state s, and 0 if he played the correct one. The probability to play the wrong arm is then $q_\pi = \mathbb{E}_{s\sim p}[q(s)]$ where p is the steady state of the problem with reset r. The expected sum of discounted gains G_π can be related to q_π as: $G_\pi = \frac{\mu}{r}(1 - 2q_\pi)$. In the following, we will use q_π as a measure of performance, trying to find a policy π that minimizes q_π (and thus maximizes G_π).

2.4 Types of memory considered

Definition 2.5 (Random Access Memory (RAM))**.** RAM is the most flexible memory setting. The agent has M memory states and has full control over it. It has $|A| = M \times 2$ possible actions: the choice of the next memory state and which arm to play. There is a high degeneracy in the space of strategies since any permutation of the memory states leads to the same performance. Note that, since our agent can use the information of the last arm and reward, the total number of memory states is in fact $M_{\text{eff}} = 4M$, which corresponds to $2 + \log_2 M$ bits.

Definition 2.6 (Memento Memory[2])**.** The agent only has access to the information of its past m actions and rewards. For instance, for $m = 4$, an observation could be `AABB++-+`. We use the notation of the most recent action/reward on the right (here, the last action was B and the reward was $+1$). If the agent plays A and obtains a positive reward, the next memory state would be `ABBA+-++`. In this memory architecture, the agent writes in its memory only through the plays he does ($|A| = 2$). Here, the number of bits is $2m$ and the total number of memory states is in fact $M_{\text{eff}} = 4^m$.

3 Gain and exploration with a Random Access Memory

Hellman & Cover (1970); Cover & Hellman (1970) described an optimal policy for the RAM architecture in the limit of a small reset r. In their optimal policy, the memory states $i = 1...M$ are organized linearly with transitions only occurring between i and $i - 1$ (if the last observation supports H_A) or $i + 1$ (if the last observation supports H_B). In the limit $r \to 0$, transition probabilities can be shown to be independent on i for $1 < i < M$. The two extreme memory states $i = 1$ and $i = M$ are special as they present a vanishing exit rate $\epsilon \to 0$. Thus, only these two states are visited with finite probability in that limit. For such a policy, one obtains an optimal probability to play the worst arm $q_{\text{HC}}(M) = \alpha^{M-1}/(\alpha^{M-1} + 1)$, with $\alpha = (1 - \mu)/(1 + \mu)$. [3]

[2] *Memento* is a Christopher Nolan's film where the hero has a short-term memory loss every few minutes. Using photos and tattoos, the hero keeps track of information he will eventually forget, thus encoding information into his own actions.

[3] To see why a linear policy is optimal, introduce $\lambda_i = P(H_A|i)/P(H_B|i)$, with $P(H_A|i)$ the conditional probability that H_A is true if the memory state i is visited. Choose the labels i such that $\lambda_1 \geq \lambda_2 ... \geq \lambda_M$. Then it can be shown that $\lambda_i/\lambda_{i+1} \leq \alpha^{-1}$ (Hellman & Cover (1970)). The linear policy saturates this bound for all i, leading to the maximal ratio that can be obtained between any two states $\lambda_1/\lambda_M = \alpha^{1-M}$. This ratio controls the gain in the limit $\epsilon \to 0$ where only these two states are visited with finite probabilities.

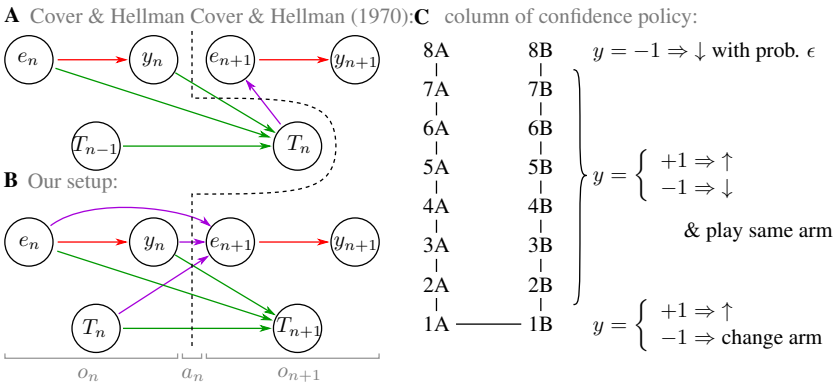


Figure 1: **A** Update scheme from Cover & Hellman (1970) vs **B** our update scheme. Using the same notation as Cover & Hellman (1970), y_n is the reward obtained by playing the arm e_n and T_n is the memory state (not to confuse with the transition probability T of Sec 2.1). To make the link with Sec 2.1 in our setup the state s_n correspond to the tuple $(e_n, y_n, T_n, k_A, k_B)$, the observation o_n to the triplet (e_n, y_n, T_n) and the action a_n is represented by the purple and green arrows. The red arrow can be seen as part of the conditional probability $T(s_{n+1}|s_n, a_n)$. **C** The column of confidence policy, that can be used in the RAM case, exemplified for $M = 8$. The memory states are organized into two columns. The distribution p_0 initializes the agent's memory into states $1A$ or $1B$ with equal chance. The agent keeps playing the same arm, moving up and down a column depending on the reward, unless it is in the memory state 1 (it then switches arm after a negative reward). Once the agent reaches a state at the top of a column, it can only step down with small probability ϵ if a negative reward is obtained. This two-column arrangment effectively double the number of memory states, by creating $2M = 16$ distinct states. In this policy, the value of the memory can be viewed as a measure of the confidence in the arm being played.

Our set-up is slightly different, as we allow for the choice of arm to depend on both the memory state and the information of the last arm played and reward obtained (Figure 1**A-B**). In that case, the policy can be improved, as demonstrated by considering the *column of confidence* policy.

Definition 3.1 (column of confidence policy (CCP))**.** It is a RAM policy with M memory states. It is depicted in Figure 1**C**. Essentially, the agent uses its last arm played to effectively increase the size of its memory by a factor 2.

The probability q to play the worst arm by following CCP is derived in Appendix C for general r, by writing the transition probability matrix T_ϵ with a generic ϵ, for any of the two hypotheses H_A or H_B. From the stationary distribution $T_\epsilon \vec{p} = \vec{p}$, one obtains $q(\epsilon)$, which reaches its minimum value for:

$$\epsilon = \sqrt{2}\frac{\sqrt{\alpha - \alpha^{3M-1} + \alpha^M(2\mu - 3) + \alpha^{2M}(2\mu + 3)}}{\sqrt{1-\alpha^M}(1 - \alpha^M - \mu - \alpha^M\mu)}\sqrt{r} + \mathcal{O}(r) \tag{6}$$

The result $\epsilon \sim \sqrt{r}$ indicates a non-trivial balance between exploration and exploitation, in which the time spent in each extreme state of the memory grows only as the square root of the horizon time $1/r$. A similar result was obtained for hypothesis testing (Wilson (2014)).

For $r \to 0$ (a result generalized for any k_B and k_A in Appendix C), we obtain:

$$q_{\text{CCP}}(M) = \frac{\alpha^{2M-1}}{\alpha^{2M-1} + 1}. \tag{7}$$

Note that it is the optimal gain of a RAM memory of size $2M$. Since our agent memory size is $M_{\text{eff}} = 4M$ (the factor 4 coming from the 2 possible past actions and two possible rewards), the results of Hellman & Cover (1970); Cover & Hellman (1970) show that CCP is nearly optimal, in the sense that no policies with one less bit of memory can do better. In Figure 2, we confirm empirically

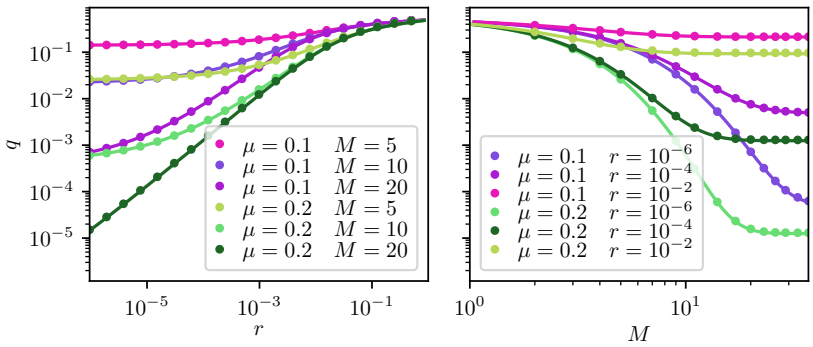


Figure 2: **Left** Probability to play the worst arm q *vs.* the reset probability r for different memory sizes M and two values of μ (the difference between the mean outcome of the two arms). **Right** q *vs.* memory size M for different r and μ indicated in legend. The solid lines show the analytical result corresponding to column of confidence policy (CCP), whereas symbols show the results of the learning algorithm (see Sec 2.2) for a RAM memory with a policy initialized close to the predicted optimal column of confidence policy ($\pi_0 \approx \pi_{\mathrm{CCP}} + 10^{-4}$). Learning does not find strategies that perform better than the optimal column of confidence policy, even for large values of r, indicating that it is a local optimum. In this figure, error bars would be smaller than the symbols.

that it is at least a local policy optimum, as performing policy optimization near this solution leads to no further improvements.

4 Gain for Memento memory

Are there efficient policies when the agent memorizes the last m arms played and rewards obtained (Memento memory cf. 2.6)? In the classical two-arm bandit problem, after a time m, the optimal strategy selects the worst arm with probability $q = O(\ln m/m)$ (Auer et al. (2002)). It turns out that our agent can use his own actions to encode events over a time much longer than m, leading to q exponentially small in 2^m in a stationary state with long horizon $r \to 0$.

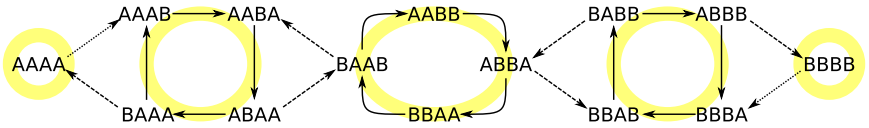


Figure 3: Necklace policy with memory of the $m = 4$ last arms played and rewards obtained (Memento memory cf. 2.6). Memory states are organized into 3 (5 if we consider the two end states) cycles of arms played, called necklaces in combinatorics. The memory state also contains the rewards obtained, but these are not explicitly shown here for the sake of readability. Most of time, the agent stays in the same necklace by playing the oldest action he remembers. Each necklace has 2 inputs and 2 outputs states (some states can be both input and output). The agent has a finite probability to leave its current necklace only when the output state is maximally informative: all 4 rewards are $+$ for A and $-$ for B to move to the left, or the opposite to move to the right. Thicker arrows represent high probability transition (deterministic in some situations); dashed arrows represent input/output transitions between necklaces; and thin arrows represent the small probabilities to leave the two end states.

Definition 4.1 (Necklace policy)**.** The necklace policy is based on 4 key ingredients (Figure 3).

(i) Most of the time, the agent plays the oldest action in its memory (i.e. the arm played m actions before). Doing so, it memorizes actions cycle inside binary necklaces of length m (in combinatorics, a necklace is an equivalent class of character strings under cyclic rotation, here the strings are words of length m made of the letters A and B, hence binary). When m is prime, there are exactly $N(m) = 2 + (2^m - 2)/m$ distinct necklaces. For any m, the number of necklaces can be derived from Pólya (1937)'s enumeration theorem and is equal to $N(m) = \frac{1}{m}\sum_{d|m}\varphi(d)2^{m/d}$, where φ is the Euler's totient function.

(ii) We provide a Gray order on the necklaces. It means that necklaces are numbered and two successive necklaces can only differ by one letter. We order the necklaces from $i = 1$ for the necklace where all actions are A to $i = n(m)$ for the necklace where all actions are B. The necklace $i = 1$ (resp. $i = n(m)$) also corresponds to the maximum confidence in hypothesis H_A (resp. H_B). In general, the longest possible chain of necklace, $n(m)$, is unknown and less than the total number $N(m)$ of necklaces. But, when m is prime, it has been conjectured (and checked for $m \leq 37$) that there exists a Gray order of all distinct necklaces (Degni & Drisko (2007)): in other words, $n(m) = N(m) = 2 + (2^m - 2)/m$, for m prime.

(iii) The probability to exit a necklace is zero, except for two exit configurations for which this probability is $\epsilon_1 > 0$ if two conditions are met. First, the memorized actions must allow the agent to switch from the necklace i to the necklace $i - 1$ or $i + 1$ by taking a new action. Second, the sequence of rewards must be maximally informative: to switch to the necklace $i - 1$ (i.e. gaining confidence in H_A), all rewards have to be $+1$ for the arm A and -1 for the arm B and the opposite to switch to the necklace $i + 1$.

(iv) In the two extreme states, the probability of exit is ϵ_0 when all rewards are negative. Below, we consider the limit $\lim_{\epsilon_1\to 0}\lim_{\epsilon_0\to 0} q(\epsilon_0, \epsilon_1)$ of this strategy. This order of limits ensures that only the extreme states are visited with a finite probability and that the agent cycles many times in each necklace before exit.

In order to compute the optimal gain of the necklace policy we introduce the following two lemmas.

Lemma 4.1. *Assume a discrete random walk on a chain of sites indexed by i, with probabilities r_i to step from i to $i + 1$ and l_i to step from i to $i - 1$. Starting in site i, the probability to reach site $j + 1$ ($i \leq j$) before site $i - 1$ is*

$$P_{i\to j} = \left(1 + \frac{l_i}{r_i} + \frac{l_i}{r_i}\frac{l_{i+1}}{r_{i+1}} + \cdots + \frac{l_i}{r_i}\ldots\frac{l_j}{r_j}\right)^{-1}. \tag{8}$$

Proof. The proof is developed in Appendix D.1. □

Lemma 4.2. *Assume a discrete random walk on a chain of $n+2$ sites indexed by $i = 0, 1, \ldots n+1$ (with probabilities r_i to step to the right and l_i to the left). If $r_0 = \epsilon R$ and $l_{n+1} = \epsilon L$, in the limit $\epsilon \to 0$, the probability to be on site $n + 1$ is*

$$p(n+1) = \left(1 + \frac{L}{R}\prod_{i=1}^{n}\frac{l_i}{r_i}\right)^{-1}. \tag{9}$$

Proof. The proof is developed in Appendix D.2. □

Using these two lemma the following theorem can be proved.

Theorem 4.3. *Consider the two-hypothesis problem described in* (4) *in the limit of small reset $r \to 0$. The necklace policy described in Definition 4.1 with parameters (ϵ_0, ϵ_1) has a probability q to play the worst arm satisfying $q \geq q^*$, with:*

$$q^* = \left(1 + \alpha^{m(1-n(m))}\right)^{-1} \quad \textit{with } \alpha = \frac{1-\mu}{1+\mu}. \tag{10}$$

The optimal necklace policy converges to q^* when $r \ll \epsilon_0 \ll \epsilon_1 \ll 1$.

Proof. The detailed proof is contained in Appendix D.3. □

Note At leading order $q^* = \alpha^{2^m} + o(\alpha^{2^m})$. This is because the number of distinct necklaces $N(m)$ only differs from $2 + (2^m - 2)/m$ at second order, and because we expect to find Gray orders within those distinct necklaces whose length only differ from $N(m)$ at second order.

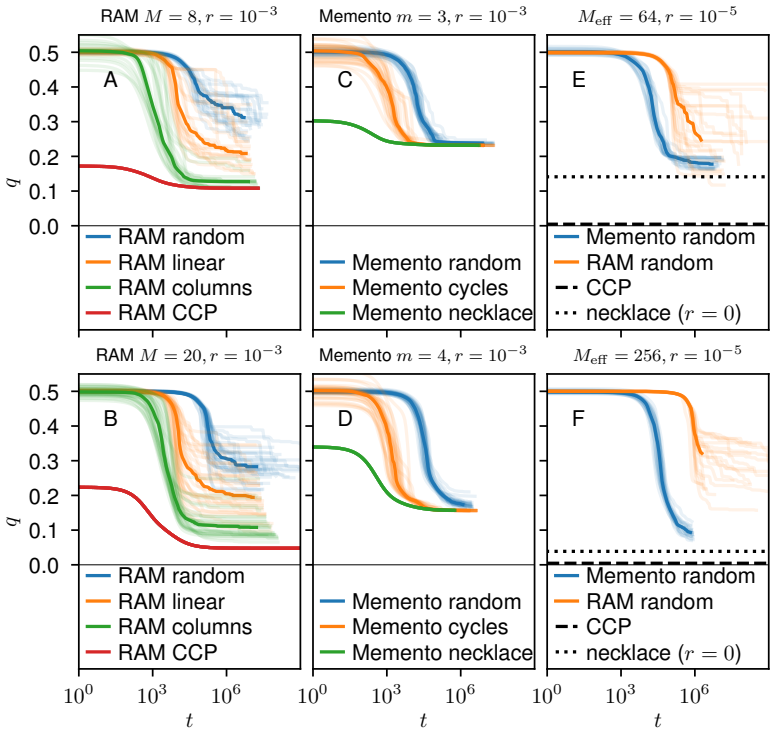


Figure 4: Dynamics of the optimization algorithm for different initialization methods. Probability to play the worse arm q *vs.* algorithm time t (time defined in (3)) for different seeds. 20 initialization seeds are shown in light color and the median is shown in solid color. The wall time is caped to 1 hour per optimization. **A-B** RAM with $M = 8$ and $M = 20$, we compare the *random* initialization, the *linear* initialization where the jumps in memory states are initialized to be contiguous, the *columns* initialization corresponding to linear initialization with the extra constraint that the last action is repeated except for the memory state 1 and the *CCP* initialization, very close to the column of confidence policy (i.e. $\pi_0 \approx \pi_{\text{CCP}} + 10^{-4}$, a difference that explains why the red curves can decrease). **C-D** q *vs.* t for the Memento memory $m = 3$ and $m = 4$. We compare the *random* initialization with the *cycles* initialization that repeats the oldest action, except if all the remembered plays correspond to a maximally informative event (during training a path between the cycles has to be learned) and the *necklace* where ϵ_0 and ϵ_1 has to be learned. **E-F** q *vs.* t for randomly initialized policies. The values of m (3 and 4) and M (16 and 64) are chosen such that the total memory needed to perform these strategies M_{eff} is identical in each panel. The dashed line corresponds to calculation of q for the CCP and for the necklace policy (for which we only have a prediction for $r = 0$). In this figure $\mu = 0.1$.

5 POLICY OPTIMIZATION AND LOCAL MINIMA

To study empirically how learning depends on the memory architecture, we measure how the probability $q(t)$ to play the worse arm after a training time t depends on the initialization of policy. For the RAM memory, we find that random initialization (blue curves) leads to very poor results (panels A and B of Figure 4). Results however improve when a linear structure for memory states is imposed (orange curves) and when the arms played are segregated on the two sides of that linear structure to form two columns (green). However, even in that case, training does not converge towards the optimal column of confidence parameters, unless parameters are initialized near the optimal values (red curves).

By contrast, training with the Memento memory (consisting in the last m actions and rewards, cf. 2.6) appears less sensitive to initialization. As shown in the panel C and D of Figure 4, initializing the policy randomly (blue) performs does not perform as well as initializing the policy with necklaces (orange), however the difference is not significant.

Although the RAM architecture is in principle more flexible (and in fact include Memento memories), we find that, for random initialization, the Memento architecture leads to actually *better* policies after training. The comparison is shown in panels E and F of Figure 4, where the two memory architectutes are compared keeping the effective memory size M_{eff} constant. This finding emphasizes the need to constrain memory architecture, so as to obtain smoother optimization landscapes.

6 CONCLUSION

	Policies	
Memory scheme	Memento (cf. 2.6)	RAM (cf. 2.5)
Policy	necklace (cf. 4.1)	CCP (cf. 3.1)
Effective memory	$M_{\text{eff}} = 4^m$	$M_{\text{eff}} = 4M$
Performance $(q^{-1} - 1)$	$\alpha^{m(1-n(m))} \sim \alpha^{-2^m}$	$\alpha^{-(2M-1)}$
... as function of M_{eff}	$\sim \alpha^{-\sqrt{M_{\text{eff}}}}$	$\sim \alpha^{-M_{\text{eff}}/2}$
... more generally for $(k_A > k_B)$	$\left(\frac{1-k_B}{1-k_A}\right)^m \left(\frac{1/k_B-1}{1/k_A-1}\right)^{m(n(m)-2)/2}$	$\frac{1-k_B}{1-k_A}\left(\frac{1/k_B-1}{1/k_A-1}\right)^{M-1}$

Table 1: Comparison of the performances of the necklace and CCP strategies in the limit $r \to 0$. As shown in the last line, the gain of these policies can be generalized to any distribution of the Bernoulli probabilities k_A and k_B (but needs not be optimal then).

Our results are summarized in Table 1 that compares the necklace policy (cf. 4.1) and the column of confidence policy (cf. 3.1). For each of these policies, we provide the optimal performance, reached in the limit $r \to 0$. We conjecture that these policies are the optimal ones for the Memento and RAM memory schemes respectively. Concerning the Memento memory, this conjecture is supported by the simulations shown in Figure 3: the best numerical policies found for $m = 3$ and $m = 4$ are in fact the necklace policy.

An interesting additional questions for the future is the generalization of these ideas to a broader set of tasks. The CCP appears well-suited for multiple hypotheses testing (Chandrasekaran & Lakshmanan (1978); Yakowitz et al. (1974)), where it would correspond to a "star" policy with a branch for each hypothesis. Classifying optimal policies for more complex hierarchical tasks, such as those involved in navigation (Theocharous et al. (2004); Toussaint et al. (2008)), would have practical applications. Looking ahead, it would be interesting to understand if these ideas have applications to other approaches dealing with POMDPs, including recurrent networks (Li et al. (2015)) whose theoretical understanding remains very limited.

Finally, it is intriguing that for all memory structures studied, a linear organization of memory states appears to be optimal. Despite the fact that our set-up is intrinsically digital, optimal policies approach an analog memory architecture with a single degree of freedom: it corresponds to the position along the chain, and measures the relative belief of one hypothesis over the other. In neuroscience,

dominants models of decision making often present a single analogue variable being updated by observations (Gold & Shadlen (2007); Rescorla & Wagner (1972)). It would be interesting to test experimentally, in situations where the environment can change with a small probability r between two distinct classes, if animals stick to two extreme believes, and leave them for exploration with some rate $\sim \sqrt{r}$.

REFERENCES

Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Balakrishnan Chandrasekaran and Kadathur B. Lakshmanan. Finite memory multiple hypothesis testing: Close-to-optimal schemes for bernoulli problems. *IEEE Transactions on Information Theory*, 24(6):755–759, 1978.

Thomas M. Cover and Martin E. Hellman. The two-armed-bandit problem with time-invariant finite memory. *IEEE Transactions on Information Theory*, 16(2):185–195, 1970. doi: 10.1109/TIT.1970.1054427.

Christopher Degni and Arthur A. Drisko. Gray-ordered binary necklaces. *the electronic journal of combinatorics*, pp. R7–R7, 2007.

Joshua I. Gold and Michael N. Shadlen. The neural basis of decision making. *Annual review of neuroscience*, 30, 2007.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.

Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of artificial intelligence research*, 13:33–94, 2000.

Martin E. Hellman and Thomas M. Cover. Learning with Finite Memory. *The Annals of Mathematical Statistics*, 41(3):765–782, 1970. ISSN 0003-4851.

Tommi Jaakkola, Satinder P. Singh, and Michael I. Jordan. Reinforcement learning algorithm for partially observable markov decision problems. *Advances in neural information processing systems*, pp. 345–352, 1995.

Arbaaz Khan, Clark Zhang, Nikolay Atanasov, Konstantinos Karydis, Vijay Kumar, and Daniel D. Lee. Memory augmented control networks. *arXiv preprint arXiv:1709.05706*, 2017.

Xiujun Li, Lihong Li, Jianfeng Gao, Xiaodong He, Jianshu Chen, Li Deng, and Ji He. Recurrent reinforcement learning: a hybrid approach. *arXiv preprint arXiv:1509.03044*, 2015.

Michael L. Littman. An optimization-based categorization of reinforcement learning environments. *From animals to animats*, 2:262–270, 1993.

Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra. Solving pomdps by searching the space of finite policies. In Kathryn B. Laskey and Henri Prade (eds.), *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pp. 417–426. Morgan Kaufmann, 1999. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=194&proceeding_id=15.

Nicolas Meuleau, Leonid Peshkin, Kee-Eung Kim, and Leslie Pack Kaelbling. Learning finite-state controllers for partially observable environments. *arXiv preprint arXiv:1301.6721*, 2013.

Junhyuk Oh, Valliappa Chockalingam, Honglak Lee, et al. Control of memory, active perception, and action in minecraft. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2016.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Leonid Peshkin, Nicolas Meuleau, and Leslie Kaelbling. Learning policies with external memory. *arXiv preprint cs/0103003*, 2001.

George Pólya. Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen. *Acta mathematica*, 68(1):145–254, 1937.

Robert A. Rescorla and Allan R. Wagner. *A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement*, pp. 64–99. Appleton-Century-Crofts, 1972.

Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *Journal of artificial intelligence research*, 23:1–40, 2005.

David Silver and Joel Veness. Monte-carlo planning in large pomdps. Neural Information Processing Systems, 2010.

Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *NIPS*, volume 13, pp. 1772–1780, 2013.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Georgios Theocharous, Kevin Murphy, and Leslie Pack Kaelbling. Representing hierarchical pomdps as dbns for multi-scale robot localization. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 1, pp. 1045–1051. IEEE, 2004.

Rodrigo Toro Icarte, Richard Valenzano, Toryn Q. Klassen, Phillip Christoffersen, Amir-massoud Farahmand, and Sheila A. McIlraith. The act of remembering: A study in partially observable reinforcement learning. *arXiv:2010.01753 [cs]*, 2020.

Marc Toussaint, Laurent Charlin, and Pascal Poupart. Hierarchical pomdp controller optimization by likelihood maximization. In *UAI*, volume 24, pp. 562–570, 2008.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Andrea Wilson. Bounded Memory and Biases in Information Processing. *Econometrica*, 82(6): 2257–2294, 2014. ISSN 1468-0262. doi: 10.3982/ECTA12188.

Sidney Yakowitz et al. Multiple hypothesis testing by finite memory algorithms. *The Annals of Statistics*, 2(2):323–336, 1974.

Marvin Zhang, Zoe McCarthy, Chelsea Finn, Sergey Levine, and Pieter Abbeel. Learning deep neural network policies with continuous memory states. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 520–527, 2016. doi: 10.1109/ICRA.2016.7487174.

Appendices

A Proof of Lemma 2.1

Proof. The state transition matrix of the undiscounted POMDP is

$$\tilde{T}_\pi(s'|s) = rp_0(s') + (1-r)T_\pi(s'|s). \tag{11}$$

The steady state $p(s)$ has to be stable through $\tilde{T}_\pi$, thus satisfying $p(s') = \sum_s \tilde{T}_\pi(s'|s)p(s)$. In the tensor form, it can be written $\vec{p} = r\vec{p_0} + (1-r)T_\pi\vec{p}$. Applying recursively this formula n times we obtain:

$$\vec{p} = r\sum_{t=0}^{n-1}(1-r)^t T_\pi^t \vec{p_0} + (1-r)^n T_\pi^n \vec{p}. \tag{12}$$

Translating it into an expectation value expression we obtain:

$$\mathbb{E}_{s\sim p}[f(s)] = r\,\mathbb{E}_{s_t\sim T_\pi^t p_0}\left[\sum_{t=0}^{n-1}(1-r)^t f(s_t)\right] + (1-r)^n \mathbb{E}_{s\sim T_\pi^n p}[f(s)] \tag{13}$$

for any function f and where $T_\pi p$ is understood as a matrix-vector product (note that $T_\pi p \neq p$). By replacing f by R and by taking the limit $n \to \infty$, for $r = 1-\gamma$, we can identify (13) with (2), thus obtaining the Lemma 2.1. □

B Implementation details

To compute the steady state we use the power method algorithm: Alg.1.

When we have multiple independent environments (by environment we mean subset of S that the agent cannot escape with its actions), S is the disjoint union of these environments: $S = S_1 + S_2 + \dots$. If p_0 factorize as follow $p_0(s) = P(S_i)P(s|S_i)$ for $s \in S_i$, we can compute the steady state by computing those of each independent environments.

The initial state of the memory of the agent can be optimized by allowing gradient flow to modify specific part of p_0. It could mathematically be reformulated as special actions done on special initial states to initialize the memory.

Data: A transition matrix M
Result: The steady state
while *the columns of M differ (with a given tolerance)* **do**
  $MM \to M$;
end
return a column of M;

Algorithm 1: Power method

C Exact computation for column of confidence policy

The *mathematica* notebook is provided along with the code, see the link in Section 1.

Here we compute the optimal value of ϵ (that maximize the gain) given r and μ.

First observe that the states (`3B+`, `1A-`, ...) can be arranged along a line:

$$\begin{array}{cccccccc} \mathrm{MA}+ & \dots & 2\mathrm{A}+ & 1\mathrm{A}+ & 1\mathrm{B}+ & 2\mathrm{B}+ & \dots & \mathrm{MB}+ \\ \mathrm{MA}- & \dots & 2\mathrm{A}- & 1\mathrm{A}- & 1\mathrm{B}- & 2\mathrm{B}- & \dots & \mathrm{MB}- \end{array} \tag{14}$$

where the probability transition only occurs between two consecutive states.

If we merge the $\pm$ we have $2M$ states and we can compute their transition matrix without reset:

$$Q = \begin{bmatrix} 1-\epsilon k_B & k_A & & & & \\ \epsilon k_B & 0 & k_A & & & \\ & k_B & 0 & \ddots & & \\ & & k_B & \ddots & k_A & \\ & & & \ddots & 0 & \epsilon k_A \\ & & & & k_B & 1-\epsilon k_A \end{bmatrix} \tag{15}$$

where $k_A = (1+\mu)/2$ and $k_B = (1-\mu)/2$

Including the reset, the transition probability becomes

$$M_{ij} = (1-r)Q_{ij} + rJ_i \tag{16}$$

where J is $1/2$ in the two central states `1A` and `1B`.

The steady state p_i is the solution of

$$\sum_j M_{ij}p_j = p_i \quad \Longleftrightarrow \quad \sum_j ((1-r)Q_{ij} - \delta_{ij})p_j = -J_i \tag{17}$$

We can split this problem in 5 regions: the A border, the A bulk, the center, the B bulk, the B border. In the bulks the (17) is

$$(1-r)k_B p_{i-1} - p_i + (1-r)k_A p_{i+1} = 0 \tag{18}$$

The solutions are of the form $p_i = c_1 w_+^i + c_2 w_-^i$ with

$$w_\pm = \frac{1 \pm \sqrt{1-(1-r)^2(1-\mu^2)}}{(1-r)(1+\mu)} \tag{19}$$

To fix the coefficients c_1, c_2 in the two bulks we have 6 equations:

- two on the left border (two first lines of (17))
- two in the center, at the injection (two middle lines of (17))
- two on the right border (two last lines of (17))

Solving these equations, we can compute the probability to play the wrong arm (B, assuming $\mu > 0$).

$$q = \frac{\begin{array}{c} w_+(1+w_+)(1-w_-)w_-^{2M}(1+w_+(\epsilon-1))(w_-+\epsilon-1) \\ +(w_+ \leftrightarrow w_-) \\ +(w_-w_+)^M(w_-w_+-1)((w_--1)(w_+-1)(w_-+w_+)(\epsilon-1)+2w_-w_+\epsilon^2) \end{array}}{2(w_--w_+)(w_+^{2M}w_-(1+w_-(\epsilon-1))(w_++\epsilon-1)-(w_+ \leftrightarrow w_-))} \tag{20}$$

in this expression of q we expressed r, μ and k_A, k_B in function of $w_\pm$

Optimizing q with respect to ϵ leads to

$$\epsilon = (w_-w_+)^{-M} \frac{-\sqrt{\begin{array}{c} \frac{(w_--1)w_-^M(w_+-1)w_+^M(w_-w_+-1)(w_-^Mw_+-w_-w_+^M)^2}{(w_--1)(w_+-1)(w_-w_+)^{1+2M}(w_-w_+-1)^2(w_+^M-w_-^M)} \\ \left(\begin{array}{c} w_-^{3M}w_+^2(1+w_-)(1+w_+) \\ -(w_+ \leftrightarrow w_-) \\ +w_+^{2M}w_-^{1+M}(2w_++w_-+w_+w_-(7+2w_+)+w_-^2(w_+-1)) \\ -(w_+ \leftrightarrow w_-) \end{array}\right) \end{array}}}{(w_-w_+-1)\left(\begin{array}{c} w_-^2w_+^{2M}(1+w_-(w_+-1)+w_+) \\ +(w_+ \leftrightarrow w_-) \\ -2w_-^{1+M}w_+^{1+M}(1+w_-w_+) \end{array}\right)} \tag{21}$$

We can make the Taylor expansion of ϵ with respect to r

$$\epsilon = \sqrt{2}\frac{\sqrt{\alpha - \alpha^{3M-1} + \alpha^M(2\mu - 3) + \alpha^{2M}(2\mu + 3)}}{\sqrt{1-\alpha^M}(1 - \alpha^M - \mu - \alpha^M\mu)}\sqrt{r} + \mathcal{O}(r) \tag{22}$$

with $\alpha = \frac{1-\mu}{1+\mu}$

For M large, it converge quickly toward

$$\epsilon = \sqrt{\frac{2r}{1-\mu^2}} + \mathcal{O}(r) \tag{23}$$

In the limit $r \to 0$ we get

$$q = \frac{\alpha^{2M-1}}{\alpha^{2M-1}+1} + \mathcal{O}(\sqrt{r}) \tag{24}$$

D RANDOM WALK ALONG A CHAIN

D.1 PROBABILITY TO TRAVERSE THE CHAIN

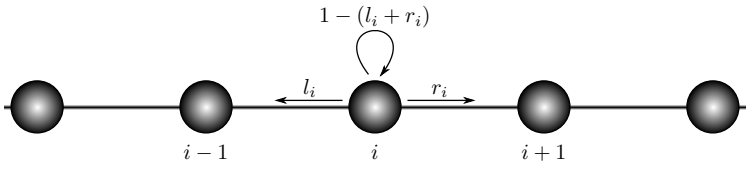


Figure 5: Markov chain

Consider a random walk on a chain of sites with probabilities r_i to move from site i to $i+1$ and l_i to move from i to $i-1$ (Figure 5). First, we want to compute the probability $P_{i\to j}$ (with $i \leq j$) that, starting at site i, the walker visits the site $j+1$ before the site $i-1$. Note that $P_{i\to j}$ only depends on $\{l_k\}_{k=i}^{j}$ and $\{r_k\}_{k=i}^{j}$. Note also that according to this definition $P_{i\to i} = \frac{r_i}{l_i+r_i}$. We also define $P_{i\leftarrow j}$ as the probability to reach $i-1$ before $j+1$ by starting in j, then we have $P_{i\leftarrow i} = \frac{l_i}{l_i+r_i}$.

Starting in i, after one time step the walker is either (i) in $i-1$ (with probability l_i) and the probability to reach $j+1$ before reaching $i-1$ becomes null, (ii) still in i (with probability $1-(l_i+r_i)$) and the probability to each $j+1$ before $i-1$ is still given by $P_{i\to j}$, (iii) in $i+1$ (with probability r_i). Once in $i+1$ there are two possibilities: either the walker never comes back to i and it reaches $j+1$ (with probability $P_{i+1\to j}$), or it does (with probability $1-P_{i+1\to j}$) and it again has the same probability $P_{i\to j}$ to reach $j+1$ before $i-1$. Thus we have:

$$P_{i\to j} = (1-(l_i+r_i))P_{i\to j} + r_i\left(P_{i+1\to j} + (1-P_{i+1\to j})P_{i\to j}\right). \tag{25}$$

The quantities that matter are $p_i \equiv r_i/(l_i+r_i)$. If we isolate $P_{i\to j}$ to the l.h.s. we get

$$P_{i\to j} = \frac{p_i P_{i+1\to j}}{1-p_i(1-P_{i+1\to j})}. \tag{26}$$

Making the changes of variable $X_{i\to j} \equiv \frac{1-P_{i\to j}}{P_{i\to j}}$ and $x_i \equiv \frac{1-p_i}{p_i}$ (note that $x_i = l_i/r_i$) we obtain

$$X_{i\to j} = x_i(1 + X_{i+1\to j}). \tag{27}$$

By repeating the formula we see that we get

$$X_{i\to j} = \sum_{k=i}^{j}\prod_{l=i}^{k} x_l = x_i + x_i x_{i+1} + x_i x_{i+1} x_{i+2} + \cdots + x_i \cdots x_j \tag{28}$$

from which we can get $P_{i\to j}$ by the inverse transformation

$$P_{i\to j} = \left(1 + \frac{l_i}{r_i} + \frac{l_i}{r_i}\frac{l_{i+1}}{r_{i+1}} + \cdots + \frac{l_i}{r_i}\cdots\frac{l_j}{r_j}\right)^{-1}. \tag{29}$$

D.2 CHAIN WITH FINAL STATES

Let us consider the same chain as above with a finite length n such that sites are labelled from $i = 1$ to n. Now let us add a final state at each end of the chain (sites $i = 0$ and $n + 1$). We consider the probabilities to leave the final states asymptotically small, of order ϵ. More precisely, the probability to move from $i = 0$ to 1 is ϵR (we call it R as it is a probability to go to the Right, although it concerns the most left site) and the probability to move from $n+1$ to n is ϵL. These probability and the probabilities $p(0)$ and $p(n + 1)$ to be on the site 0 and $n + 1$ are related by a balance of the flow from 0 to $n + 1$:

$$p(0)\epsilon R P_{1\to n} = p(n+1)\epsilon L P_{1\leftarrow n}. \tag{30}$$

In the limit $\epsilon \to 0$, the probabilities to be in the extreme states tends to 1 and we thus have $p(0) = 1 - p(n + 1)$, yielding

$$p(n+1) = \left(1 + \frac{L}{R}\frac{P_{1\leftarrow n}}{P_{1\to n}}\right)^{-1}. \tag{31}$$

This result can be simplified using the equality

$$\frac{P_{1\leftarrow n}}{P_{1\to n}} = \frac{1 + \frac{l_1}{r_1} + \cdots + \frac{l_1}{r_1}\cdots\frac{l_n}{r_n}}{1 + \frac{r_n}{l_n} + \cdots + \frac{r_n}{l_n}\cdots\frac{r_1}{l_1}} = \prod_{i=1}^{n}\frac{l_i}{r_i}, \tag{32}$$

to finally obtain the formula

$$p(n+1) = \left(1 + \frac{L}{R}\prod_{i=1}^{n}\frac{l_i}{r_i}\right)^{-1}. \tag{33}$$

D.3 CHAIN OF NECKLACES

To compute the gain of the necklace policy, the idea is to show that necklaces can be arranged on a chain so that we can use (33) (see Figure 3).

For m prime, the number of non trivial necklaces is $(2^m - 2)/m$, the trivial necklaces being the two final states (i.e., the words `AAA..A` and `BBB..B`). Using the conjecture of Degni & Drisko (2007), there is a Gray order on these necklaces when m is prime. In other words, there is a chain from one final state to the other that passes exactly once by each necklace, the difference between two successive necklaces being exactly one bit (i.e. a single `A` is changed into `B` or vice versa).

In any case (m prime or not), we call $n(m)$ the length of the longest chain with Gray order. We call $y(m)$ the length of the smallest necklace in that longest chain (arguably the smallest prime factor of m).

A necklace is characterized by the numbers a and b of letters `A` and `B` in the m-long word, with $a + b = m$. After at least one complete loop in the necklace (i.e. at least $y(m)$ actions), the probabilities to leave that necklace (when we are at the exit states) are

$$l_i = \epsilon_1 k_A^a (1 - k_B)^b, \tag{34}$$

$$r_i = \epsilon_1 (1 - k_A)^a k_B^b. \tag{35}$$

With the odds to do at least one loop increasing as ϵ_1 goes to zero or $y(m)$ goes to ∞.

In the two final states, and again after one loop, the probabilities to leave are $\epsilon_0 L = \epsilon_0(1 - k_B)^m$ and $\epsilon_0 R = \epsilon_0(1 - k_A)^m$. We can now use the formula (33) (when $\epsilon_0 \ll \epsilon_1 \ll 1$), in which the

product simplifies as

$$\prod_{i=1}^{n} \frac{l_i}{r_i} = \prod_{i=1}^{n(m)-2} \frac{k_A^{a_i}(1-k_B)^{b_i}}{(1-k_A)^{a_i}k_B^{b_i}} \tag{36}$$

$$= \frac{\prod_{i=1}^{n(m)-2}(1/k_B-1)^{b_i}}{\prod_{i=1}^{n(m)-2}(1/k_A-1)^{a_i}} \tag{37}$$

$$= \frac{(1/k_B-1)^{\sum_i b_i}}{(1/k_A-1)^{\sum_i a_i}} \tag{38}$$

$$= \left(\frac{1/k_B-1}{1/k_A-1}\right)^{m(n(m)-2)/2} \qquad \text{since} \sum_{i=1}^{n(m)-2}(a_i+b_i) = m(n(m)-2) \tag{39}$$

where a_i and b_i are the occurrences of A and B in the necklace i.

Inserting (39) into (33) and using the value of k_A and k_B given in (4) for hypothesis H_A leads to

$$p(n+1) = \left(1+\alpha^{m(1-n(m))}\right)^{-1}, \qquad \text{with } \alpha = \frac{1-\mu}{1+\mu}. \tag{40}$$

and $p(0)$ can be obtained by changing μ into $-\mu$ or α into $1/\alpha$, by symmetry of the necklace policy. The probabilities under hypothesis H_B are obtained by exchanging $p(0)$ and $p(n+1)$, again by symmetry.

Under hypothesis H_A (resp. H_B), the value $p(n+1)$ (resp. $p(0)$) corresponds to the probability q^* to play the worst arm if the probabilities of the non-final necklaces are zero (which is asymptotically true if $\epsilon_0 \ll \epsilon_1$). To reach q^*, we also need ϵ_1 to be asymptotically small in order to guarantee at least one loop in each necklace and ϵ_0 has to be asymptotically larger than the reset r. In summary, we need $r \ll \epsilon_0 \ll \epsilon_1 \ll 1$ to reach asymptotically q^*, otherwise the probability q will be larger than q^*.

D.4 COLUMN OF CONFIDENCE POLICY WITH NO RESET

When there is no reset $r = 0$ we can compute the performance of the column of confidence policy for two arms of probabilities k_A and k_B. We obtain via (33) (assuming $k_A > k_B$)

$$q^{-1} - 1 = \frac{1-k_B}{1-k_A}\left(\frac{1/k_B-1}{1/k_A-1}\right)^{M-1} \tag{41}$$

8 Discussions and Future works

8.1 Exact models of Double-Descent

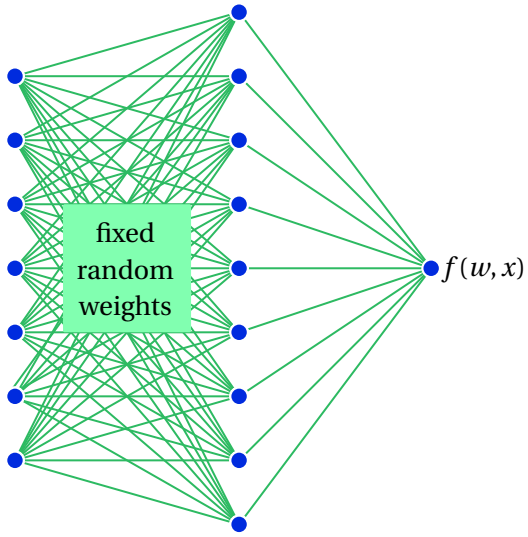


Figure 8.1 – The random features model corresponds to a FC with one hidden layer who has its first layer of weights frozen during learning.

While our results on double-descent applies for deep FCs, our arguments do not have the rigor of mathematical proofs. Recent literature provides rigorous proofs predicting double-descent in simple models and data.

Mei and Montanari (2021) proved double-descent in a simple model. They used a random features model i.e. a 1 hidden layer FC with the first layer frozen, see Figure 8.1. Their model is trained on random point x on the sphere of dimension d with labels that are function of $x \cdot w$ for some vector w. Using random matrix theory, they computed the test error in the limit $h, P, d \to \infty$ with h/d and P/d.

Using the same model, d'Ascoli et al. (2020) proved that ensemble averaging flattens the test

error after the interpolation threshold. They thus confirmed that the second descent is indeed due to random initialisation, consistent with our predictions.

The same results has then been generalized for any data of finite size P Jacot et al. (2020).

8.2 Mean-Field limit and initialization

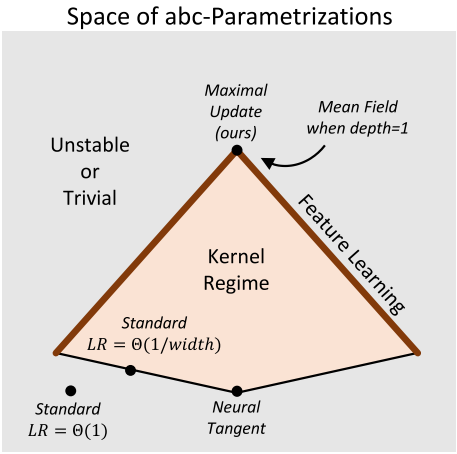


Figure 8.2 – Phase space of initializations proposed by Yang and Hu (2021). Source Yang and Hu (2021)

The Neural Tangent Kernel limit, who has been first computed for FCs Jacot et al. (2018), has been generalized to all architectures Yang (2019b, 2020a,b). The same is true for the Mean-Field limit, originally defined for one hidden layer FC, then extended to deeper FC Nguyen and Pham (2020), has been generalized to all architectures Yang and Hu (2021).

It allowed, for the FC, to determine rigorously an initialization method in which all layers learn in the $h \to \infty$ limit Yang and Hu (2021). This initialization has been coined *Maximal Update*. To find this initialization, they define a space of all possible initialization where the initialization of each set of parameters is parametrized by three constants (a, b, c). a characterize the scale of the prefactor in front of the weight. b characterize the initialization amplitude of the weight. And c characterize the learning rate for that weight. Then they rigorously show which initializations among the nontrivial (weights change in a finite time) and stable (SGD does not blowup) allow a Mean-Field limit or an NTK limit. Among the initializations leading to a Mean-Field limit the identify the maximally updated ones for which all layers evolve during training. The Figure 8.2 shows a sketch of their classification.

The initialization we defined as MFP (see Table 1 in Section 0.1) is equivalent to the Maximal Update initialization. This formalism is consistent with our results: it allows to predict for new architecture, the proper initialization such that the model $F(w) = \alpha(f(w) - f(w_0))$ has its feature-lazy cross over at $\alpha \sim 1$, as we argued in a special case.

8.3 Stochastic Gradient Descent

The most common algorithm for training neural networks is Stochastic Gradient Descent (SGD), defined in Section 0.2. SGD needs two parameters: the learning rate and the batch size.

In Chapter 3, we used gradientflow in order to have a well defined optimization algorithm without hyper parameter (that would otherwise add an extra dimension in the phase diagram; or would have needed to be optimized for each point of phase space).

However it is known that SGD improves performance over gradientflow He et al. (2019). The difference between the two algorithms lie in the random batch that introduce randomness in the gradient, but also in the finite step of the learning rate. Note that in the limit of zero learning rate, the noise from the batch is eventually averaged over the steps and therefore suppressed, leading to gradientflow.

It is still not clear why SGD improves performance. SGD has been modeled as a continuous Fokker-Plank dynamics Chaudhari and Soatto (2018b) in which the diffusion matrix is assumed to no depend on the trajectory. Other works show that SGD acts as an implicit regularizer for the trace of the diffusion matrix (variance of the gradients) Roberts (2021b); Smith et al. (2021a).

An interesting question for the future is to explore the impact of SGD in the perspective of the $(h, \tilde{\alpha})$ phase space. Near jamming, we expect that SGD noise is a relevant perturbation, that will smooth the divergence of f and improve performance. What about the over-parametrized regime? We have preliminary evidence that SGD can shift the location of the feature-lazy cross-over. However, we, observed other effects. SGD could improve performance in feature regime even if the lazy regime works better. Understanding these observations is a key question for the future.

9 Conclusion

In this thesis we studied why the dynamics of neural networks is not hampered by local minima of the loss landscape and why neural networks do not overfit although they have more parameters than points in the trainset. We also addressed the question of how neural networks do not suffer from the curse of dimensionality.

Bad minima The loss landscape is a priori not a convex function of the parameters, and it depends on many parameters. This situation is similar to the energy landscape of glasses; which is also non-convex and depends on many degrees of freedom. The energy landscape of glasses has many local minima, in which the system gets trapped. We call a landscape with lot of local minima, *glassy*. In the past, the field of ML has worried that the landscape of neural networks was glassy; and that during training the neural network is stuck in local minima, and therefore cannot reach a global minimum. However, observations indicated that it is not so. There are two regimes controlled by the number of parameters and the size of the trainset. With a small number of parameters, the landscape is indeed glassy. However with large number of parameters, the landscape is not glassy anymore and the network reaches the global minima of the loss. With enough parameters, it can even learn a dataset of images whose labels has been shuffled Zhang et al. (2017). This raises the question of the geometry of the loss landscape.

In the glasses, replacing long range interactions (Van der Waals) by finite range interactions (granular particles) allowed to map the problem of minimizing the energy to a SAT/UNSAT problem with continuous degrees of freedom coined the *jamming transition*. The jamming transition is a transition between two regimes driven by the density, the number of particles per total volume. If the density is low, the system can reach an energy of zero and bring all forces to zero. If the density is high, the system is glassy and gets stuck in local minima. At the transition, particles are in contact but with no overlap (no forces, assuming a quadratic potential). If the density is slightly increased from the transition, the contacts start to make overlaps. The main result is that jamming separates into two universality classes. These two classes can be characterized by the number of overlaps N_Δ and the total number of degrees of

freedoms N (the number of particles times the dimension). The two classes are:

- The isostatic class: at the transition, N_Δ jumps from 0 to N. In this class, the Hessian is gap-less: its spectrum is populated up to 0.
- The hypostatic class: at the transition N_Δ jumps from 0 to a finite fraction of N ($C_0 N$ with $C_0 < 1$). Here the Hessian's spectrum has a delta in zero and a gap followed by a bulk.

The jamming of spheres is isostatic while the jamming of ellipses is hypostatic.

The jump in N_Δ can be understood intuitively for the case of spheres. While increasing progressively the density, the jamming transition is reached when all directions in phase space create overlaps. In these aspects, spheres are isostatic because as long as $N_\Delta < N$ there will exists a direction in the phase space that is orthogonal to all normal vectors at the contacts

$$\exists \{\vec{\delta}_i\}, \forall (i,j) \in m, \quad \vec{n}_{ij} \cdot (\vec{\delta}_i - \vec{\delta}_j) = 0 \tag{9.1}$$

where $\{\vec{\delta}_i\}$ is the direction orthogonal to all the normals $\vec{n}_{ij}$ of the set m of contacts. When moving along this direction, $\vec{x}_i \to \vec{x}_i + \epsilon\vec{\delta}_i$, due to the shape of the spheres, all the contacts will open as ϵ^2. It imposes that there exists no solutions to (9.1) in a jammed packing, otherwise they would be unstable. Therefore N_Δ has to be equal or greater than N at the transition. Another argument shows that N_Δ has to be smaller or equal to N at the transition (otherwise particles would overlap). Hence $N_\Delta = N$ for spheres. But in the case of ellipses, due to the additional rotational degrees of freedom, directions orthogonal to the contacts do not necessarily open contacts, and are stabilized by non-linearities.

It can be shown that these geometrical properties can be related to properties of the Hessian of the energy. Indeed, the sign of the eigenvalues of the Hessian describe the stability the system. When the energy can be written as a sum over unsatisfied constraints $E = \sum_{(i,j)\in m} V(\Delta_{ij})$ where Δ_{ij} are the overlaps and m is the set of contacts, the Hessian can be decomposed into

$$\mathscr{H} = \underbrace{\sum_{(i,j)\in m} V''(\Delta_{ij})\, \nabla\Delta_{ij} \otimes \nabla\Delta_{ij}}_{\mathscr{H}_0} + \underbrace{\sum_{(i,j)\in m} V'(\Delta_{ij}) \mathscr{H}_{\Delta_{ij}}}_{\mathscr{H}_p} \tag{9.2}$$

where $\mathscr{H}_{\Delta_{ij}}$ are the Hessians of Δ_{ij}, they describe how the overlaps Δ_{ij} change at second order. $\mathscr{H}_0$ always has N_Δ positive eigenvalues but the spectrum of $\mathscr{H}_p$ depends on the problem. In the isostatic class, $\mathscr{H}_p$ has only negative eigenvalues compared to the hypostatic class where it has eigenvalues with both signs.

The same approach can be used in neural networks by picking a custom loss function: the quadratic hinge loss $\ell(y,f) = \max(0, 1-yf)^2$ who actually performs as well as the usual cross entropy. For the neural network, P/N plays the role of the density. The data points contributing to the loss ($yf < 1$) play the role of the overlaps. N_Δ is their number. While, with particles,

pairs of particles are the constraint, with the neural network each data point is a constraint. At the jamming transition, particle contacts are equivalent to data points right at the margin ($yf = 1$), similar to *support vectors* for support vector machines.

We showed in a special case and confirmed broadly empirically that moving orthogonally to constraints can lead to overlaps: $\mathscr{H}_p$ has both positive and negative eigen-values. We expect that the jamming of neural networks is generically hypostatic.

As a consequence, we obtain that:

1. For any algorithm (GD, SGD, ADAM, gradientflow, ...), there exists a critical number of parameters N^* at which there is a sharp jamming transition. The transition delineate a glassy under-parametrized phase from an easy over-parametrized one.

2. As shown in Figure 9.1(left), at the transition, the number of support vectors is, like for the ellipses, equal to a finite fraction of the parameters: $C_0 N < N_\Delta < N$ where C_0 is the fraction of negative eigenvalues in $\mathscr{H}_p$. It's a property we don't control a priori but that we measure a posteriori.

3. Near that transition, the spectrum of the Hessian has a delta in zero, then a gap followed by a bulk.

4. We argue that $N^* < P/C_0$. While measuring the Hessian we observed $C_0 \approx 0.5$ in `relu` FCs and $C_0 \approx 0.43$ in `tanh` FCs for random normal dataset and MNIST. It suggest that a dataset can be fitted with N of the order of number of data point in the dataset.

We verified all the consequences on different datasets, dataset sizes and network depths.

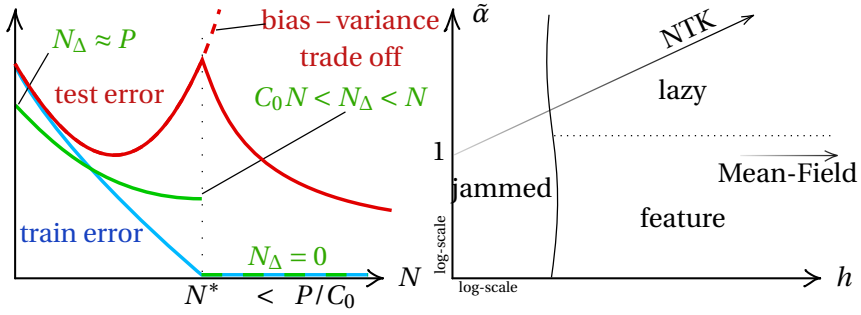


Figure 9.1 – (Left) train error, test error and N_Δ as function of the number of parameters. At jamming, when $N = N^*$, the train error reaches zero, N_Δ falls from a finite value to zero, and the test error displays a peak instead of growing as predicted by the bias–variance trade-off. (Right) Phase diagram of the loss landscape. $\tilde{\alpha}$ ($= \sqrt{h}\alpha$) controls the learning regime. For $\tilde{\alpha} < \sqrt{h}\alpha^*$ it enters in the feature regime. When h is smaller than h^*, who depends on $\tilde{\alpha}$, the network is trapped in local minima. When $\tilde{\alpha}$ scales as $\sqrt{h}$, the network converges to the NTK limit but when $\tilde{\alpha} \sim 1$, it converges to the Mean-Field limit.

It is interesting to look how N^* depends on P. For most dataset we observed $N^* \sim P^a$ with $a = 1$ for random data with random labels and $a < 1$ for structured data. For instance we measured $a \approx 0.75$ for MNIST. It would be interesting to study this dependency further. For a given dataset size P we have a corresponding $N^*(P)$. Consider a network at jamming. Discarding the finite N fluctuations, assume it can fit the trainset with N^* parameters but cannot with $N^* - 1$. If one transfers a data point from the testset to the trainset, two cases can happen. Either, with probability $1 - \epsilon$, it will be fitted by the model and therefore the jamming threshold N^* would be unaffected. Or, in case the data point is wrongly predicted by the model, we expect N^* to grow. Assuming a new wrongly fitted data is as hard to learn as a random data without structure, we conjecture that $\frac{\mathrm{d}}{\mathrm{d}P} N^*(P) \propto \epsilon$.

From a practitioner point of view, the consequences of all these results are that: (i) There is a sharp transition from under to over parametrized whose location varies for different training dynamics and (ii) increasing the number of parameters allows to escape bad minima, which sounds like a very simple solution.

Yet according to the bias-variance trade-off, there is a trade-off in the model complexity. With too low complexity, the error is dominated by the bias: The model is not able to fit the trainset which induce a bias. While at too high complexity the error is dominated by the variance: The model has too much degrees of freedom and the random fluctuations from initialization affect the prediction. One should thus worry that having more parameters than number of data points in the trainset ($P/C_0 < N$) would lead to over-fitting.

Over-fitting The bias–variance trade-off predicts that the test error as a function of the model complexity has a U shape, see the red dashed line in Figure 9.1(left). However for neural networks, we discovered in Spigler et al. (2019) that the test error of a classification task has that U shape for $N < N^*$, but for $N^* < N$, it surprisingly decreases as the number of parameters increases, see the red curve in Figure 9.1(left). This phenomena has later been coined *double-descent* by Belkin et al. (2019).

The double-descent was known in regression problems Advani et al. (2020) where the peak happens at $N = P$ also where the train error reaches zero. It has been later studied in many works including Nakkiran et al. (2019); Lee et al. (2020); Belkin et al. (2020).

We explain the two behaviors of double-descent: a peak at jamming and the decrease toward the $N \to \infty$ limit.

Close to jamming, in order to satisfy the constraints, the norm of the model has to diverge. We argued and observed that this divergence scales like

$$\|f\| \sim 1/(N - N^*) \tag{9.3}$$

which differs from the prediction of $\|f\| \sim 1/(N-P)^2$ in regression Advani et al. (2020); Liao and Couillet (2018). This divergence increases fluctuation which in turn jeopardize the perfor-

mance. It is consistent with the peak of the test error at jamming.

Above jamming, the decrease of the test error is due to the randomness of initialization. In the $N \to \infty$ limit, depending on the parametrization, the neural network converges either in:

- The Neural Tangent Kernel limit Jacot et al. (2018); Du et al. (2019); Allen-Zhu et al. (2019); Lee et al. (2019); Arora et al. (2019); Park et al. (2019)
- The Mean-field limit Mei et al. (2018); Rotskoff and Vanden-Eijnden (2018); Chizat and Bach (2018); Sirignano and Spiliopoulos (2020b); Mei et al. (2019); Nguyen (2019); Sirignano and Spiliopoulos (2020a); Nguyen and Pham (2020); Yang (2019a)

In both limits there is no dependency to the random initialization of the weights. These limits shows that an overparametrized network converges to well behaved models and explains why $\epsilon(N) \xrightarrow{N\to\infty} \epsilon_\infty$

In order to explain that the test error *decreases* toward the infinite limit, we explain that using a finite N makes the output of the network depend on the initial conditions of the weights. It brings fluctuations, which decrease the performance. These fluctuations can be removed by averaging the outputs of several networks initialized with different random seeds. It is called *ensemble averaging*. We observed that most of the variation of the test error in N is suppressed when we compute the ensemble average. It confirm that fluctuations dominate the behavior of the test error.

Then we explained how these fluctuations induce an increase of the test error and how they scale with N. Under mild assumptions, we argued that

$$\langle \epsilon(f) \rangle - \epsilon(\langle f \rangle) \sim \mathrm{Var}(f) \tag{9.4}$$

like for regression. Then for the NTK limit, the fluctuations due to initialization scale as $\mathrm{Var}(\Theta(0)) \sim 1/h$, so we concluded that

$$\mathrm{Var}(f) \sim 1/h \tag{9.5}$$

For the Mean-Field limit, we can argue at initialization that using the law of large numbers applied on the sum of neurons gives the fluctuation around the expectation value, leading to the same scaling as for the NTK. It has been proven rigorously also after training by Chen et al. (2020).

Finally, we obtain a prediction for the scaling of the test error

$$\epsilon(N) = \underbrace{\epsilon_\infty}_{\text{large } N \text{ limit}} + \underbrace{B_0 N^{-1/2}}_{\text{fluctuations of } f} + o(N^{-1/2}) \tag{9.6}$$

consistent with our observation on MNIST. We also verified numerically all the intermediate

steps of the reasoning: the scaling of the divergence at jamming, the scaling of the fluctuations at large N, etc.

Overall there is no overfitting: the second descent corresponds simply to a convergence to well-defined limiting behaviors.

Lazy and feature To unify the two limits: NTK and Mean-Field, we introduced the same model as Chizat et al. (2019)

$$F(w,x) = \alpha(f(w,x) - f(w_0,x)) \tag{9.7}$$

Where $\alpha \in \mathbb{R}$ and f is initialized in the NTP parametrization (see Table 1 in Section 0.1). If $\alpha \sim 1/\sqrt{h}$ the network converge to the Mean-Field limit and if $\alpha \sim 1$ it converge to the NTK limit. α and h parameterize a phase space shown in Figure 9.1(right). The parameter α controls a cross-over between two regimes: feature and lazy. These regimes are the finite h equivalent of Mean-Field and NTK. To get an idea of the difference between feature and lazy it's useful to visualize the network as a manifold. The network can be seen as a manifold parametrized by the parameters embedded in the space of functions from the input space to the output space

$$f : W \longrightarrow (X \to Y) \tag{9.8}$$

This manifold has a curvature but for small changes of the parameters around the initialization w_0, f behaves linearly. The linearization of f at some w is called the tangent space. While feature and Mean-Field dynamics leave the tangent space, lazy and NTK dynamics stay in it, see Table 9.1. Each network has a characteristic time t_1, characterizing the time to exit the

	$N^* \ll N < \infty$	$N \to \infty$
remain in the tangent space	lazy	NTK
leave the tangent space	feature	Mean-Field

Table 9.1 – Classification of the regimes

tangent space. It can be seen as an approximate measure of the curvature of the manifold. [1] If the training stops before t_1, the network remains in the lazy regime otherwise it enters the feature regime. We showed how varying the margin of the loss (equivalent to $1/\alpha$) allows us to control the training time and therefore the learning regime. We found that: (i) The two regimes are separated by an α^* that scales as $1/\sqrt{h}$, extending a result for 1-hidden layer Chizat et al. (2019). (ii) Which regime leads to the best performance depends on the dataset and architecture: in our tests, FCs performed generally better in the lazy regime while CNNs happen to work better in feature regime, as recently confirmed by Lee et al. (2020).

We argue that taking into account where the model is in the phase diagram sketched in

[1] Contrary to the manifold curvature measures from geometry, t_1 depends on the parametrization and the optimization algorithm.

Figure 9.1(right) is important for future empirical studies and as well as for practitioners. This phase diagram gives a good picture for the dependency in the number of parameters. We miss a similar picture for the dependency in the number of data points and an explanation for the curse of dimensionality.

Curse of dimensionality General arguments suggest that $P \sim (1/\epsilon)^d$ — and learning thus essentially impossible for generic data — when the dimension d of the data is large, which is generally the case in practice. However neural networks have good performance. It indicates that the datasets contain a lot of structure and redundancy. One example are the translations in the images. Translations form a symmetry group. Groups are rich mathematical structures. However translations only reduce the dimensionality by only a few: $P = e^d \rightarrow P = e^d/d = e^{d-\ln(d)}$. Another family of transformation that are very close to translation are the diffeomorphisms. They do not form a group, however they reduce much more the dimensionality. We observed that FCs trained on images are not stable to diffeomorphisms. However we observed that CNNs, that compare to FCs are equivariant to translation and locally connected, do learn to become stable to diffeomorphisms. We observed, by training different architecture on CIFAR10 that the test error is strongly correlated to

$$R_f = \frac{D_f}{G_f} \tag{9.9}$$

where D_f is the response to diffeomorphism and G_f is the response to Gaussian noise. It supports that learning diffeomorphisms is key to the success of deep neural networks. This difference between FC and CNN may explain why FC perform better in lazy than feature: FC lack the architectural bias CNN has that allows for learning beneficial features. It raises many questions: (i) How is relative stability established while training? (ii) Is relative stability necessary for good performance?

Equivariant architectures CNNs perform better than FCs. Assuming this gain in performance can be exported to other data than images, we developed new architectures that are equivariant to other groups of symmetry. We mainly treated 3D rotations, see Table 9.2. These

	Group	Representations
CNNs	$T(2)$	scalar lattice
Harmonic Networks Worrall et al. (2017)	$E(2)$	irreps lattice
Spherical CNN Cohen et al. (2018)	$SO(3)$	scalar on $s^2/SO(3)$
Tensor field networks Thomas et al. (2018)	$SE(3)$	irreps on graph
3D steerable CNN Weiler et al. (2018)	$SE(3)$	irreps lattice
Euclidean Neural Networks Geiger et al. (2020b)	$E(3)$	irreps on lattice/graph

Table 9.2 – Selection of equivariant architectures. T: Translations, $(S)E$: (Special) Euclidean groups, SO: Special orthogonal groups. Our work in bold.

architectures should need less data than their non-equivariant counterparts, at least of the order of the order of their group of symmetry Bietti et al. (2021). Their performance is confirmed in different applications including Batzner et al. (2021); Chen et al. (2021); Miller et al. (2020); Smidt et al. (2021). Similar architectures has been used recently for protein folding Jumper et al. (2021); Baek et al. (2021).

With these new architectures comes many questions. Once the symmetry group chosen, the choice of representation might affect performance. When shall we prefer regular from irreducible representations? Or other representations? If different representations are used together, how to chose the multiplicity in each intermediate layer? Moreover, the weights can be distinguished by the representations they link together. How to tune their relative learning speed?

These architectures are equivariant which distinguish them from invariant ones Smidt (2020). Could it, like for the case of CNNs, helps them to learn to becomes stable to different kinds of diffeomorphisms (relatively to generic deformations)?

The field of geometry in neural networks is wide and a lot remains to discover Bronstein et al. (2021)!

Bibliography

Advani, M. S., Saxe, A. M., and Sompolinsky, H. (2020). High-dimensional dynamics of generalization error in neural networks. *Neural Networks*.

Allen-Zhu, Z., Li, Y., and Song, Z. (2019). A Convergence Theory for Deep Learning via Over-Parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR. ISSN: 2640-3498.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. (2019). On Exact Computation with an Infinitely Wide Neural Net. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8141–8150. Curran Associates, Inc.

Azulay, A. and Weiss, Y. (2018). Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*.

Bach, F. (2017). Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681.

Baek, M., DiMaio, F., Anishchenko, I., Dauparas, J., Ovchinnikov, S., Lee, G. R., Wang, J., Cong, Q., Kinch, L. N., Schaeffer, R. D., Millán, C., Park, H., Adams, C., Glassman, C. R., DeGiovanni, A., Pereira, J. H., Rodrigues, A. V., van Dijk, A. A., Ebrecht, A. C., Opperman, D. J., Sagmeister, T., Buhlheller, C., Pavkov-Keller, T., Rathinaswamy, M. K., Dalwadi, U., Yip, C. K., Burke, J. E., Garcia, K. C., Grishin, N. V., Adams, P. D., Read, R. J., and Baker, D. (2021). Accurate prediction of protein structures and interactions using a three-track neural network. *Science*.

Baity-Jesi, M., Sagun, L., Geiger, M., Spigler, S., Arous, G. B., Cammarota, C., LeCun, Y., Wyart, M., and Biroli, G. (2018). Comparing dynamics: Deep neural networks versus glassy systems. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 314–323, Stockholmsmässan, Stockholm Sweden. PMLR.

Bansal, Y., Advani, M., Cox, D. D., and Saxe, A. M. (2018). Minnorm training: an algorithm for training overcomplete deep neural networks. *arXiv preprint arXiv:1806.00730*.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. (2021). Se(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.

Belkin, M., Hsu, D., and Xu, J. (2020). Two models of double descent for weak features. *SIAM Journal on Mathematics of Data Science*, 2(4):1167–1180.

Berthier, L. and Biroli, G. (2011). Theoretical perspective on the glass transition and amorphous materials. *Reviews of Modern Physics*, 83(2):587.

Bietti, A., Venturi, L., and Bruna, J. (2021). On the sample complexity of learning with geometric stability.

Brito, C., Ikeda, H., Urbani, P., Wyart, M., and Zamponi, F. (2018). Universality of jamming of nonspherical particles. *Proceedings of the National Academy of Sciences*, 115(46):11736–11741. Publisher: National Academy of Sciences Section: Physical Sciences.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges.

Bruna, J. and Mallat, S. (2013). Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886.

Charbonneau, P., Kurchan, J., Parisi, G., Urbani, P., and Zamponi, F. (2014a). Exact theory of dense amorphous hard spheres in high dimension. iii. the full replica symmetry breaking solution. *Journal of Statistical Mechanics: Theory and Experiment*, 2014(10):10009.

Charbonneau, P., Kurchan, J., Parisi, G., Urbani, P., and Zamponi, F. (2014b). Fractal free energy landscapes in structural glasses. *Nature Communications*, 5(3725).

Chaudhari, P. and Soatto, S. (2018a). Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop, ITA 2018, San Diego, CA, USA, February 11-16, 2018*, pages 1–10. IEEE.

Chaudhari, P. and Soatto, S. (2018b). Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *International Conference on Learning Representations*.

Chen, Z., Andrejevic, N., Smidt, T., Ding, Z., Xu, Q., Chi, Y., Nguyen, Q. T., Alatas, A., Kong, J., and Li, M. (2021). Direct prediction of phonon density of states with euclidean neural networks. *Advanced Science*, 8(12):2004214.

Chen, Z., Rotskoff, G., Bruna, J., and Vanden-Eijnden, E. (2020). A Dynamical Central Limit Theorem for Shallow Neural Networks. *Advances in Neural Information Processing Systems*, 33.

Chizat, L. and Bach, F. (2018). On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport. In *Advances in Neural Information Processing Systems 31*, pages 3040–3050. Curran Associates, Inc.

Chizat, L., Oyallon, E., and Bach, F. (2019). On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pages 2937–2947.

Choromanska, A., Henaff, M., Mathieu, M., Ben Arous, G., and LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA. PMLR.

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. (2018). Spherical CNNs. In *International Conference on Learning Representations*.

Cohen, T. S., Geiger, M., and Weiler, M. (2019). A general theory of equivariant cnns on homogeneous spaces. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Cooper, Y. (2018). The loss landscape of overparameterized neural networks. *arXiv preprint arXiv:1804.10200*.

Cucker, F. and Smale, S. (2002). Best choices for regularization parameters in learning theory: On the bias-variance problem. *Found. Comput. Math.*, 2(4):413–428.

d'Ascoli, S., Refinetti, M., Biroli, G., and Krzakala, F. (2020). Double trouble in double descent: Bias and variance(s) in the lazy regime. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2280–2290. PMLR.

de G. Matthews, A. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*.

DeGiuli, E., Düring, G., Lerner, E., and Wyart, M. (2015). Unified theory of inertial granular flows and non-brownian suspensions. *Physical Review E*, 91(6):062206.

DeGiuli, E., Laversanne-Finot, A., Düring, G. A., Lerner, E., and Wyart, M. (2014). Effects of coordination and pressure on sound attenuation, boson peak and elasticity in amorphous solids. *Soft Matter*, 10(30):5628–5644.

Dieleman, S., De Fauw, J., and Kavukcuoglu, K. (2016). Exploiting cyclic symmetry in convolutional neural networks. *arXiv preprint arXiv:1602.02660*.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2019). Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*.

Franz, S., Parisi, G., Urbani, P., and Zamponi, F. (2015). Universal spectrum of normal modes in low-temperature glasses. *Proceedings of the National Academy of Sciences*, 112(47):14539–14544.

Freeman, C. D. and Bruna, J. (2017). Topology and geometry of deep rectified network optimization landscapes. *International Conference on Learning Representations*.

Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. (2020). Se(3)-transformers: 3d roto-translation equivariant attention networks.

Geiger, M., Jacot, A., Spigler, S., Gabriel, F., Sagun, L., d'Ascoli, S., Biroli, G., Hongler, C., and Wyart, M. (2020a). Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401. Publisher: IOP Publishing.

Geiger, M., Petrini, L., and Wyart, M. (2021). Landscape and training regimes in deep learning. *Physics Reports*, 924:1–18. Landscape and training regimes in deep learning.

Geiger, M., Smidt, T., M., A., Miller, B. K., Boomsma, W., Dice, B., Lapchevskyi, K., Weiler, M., Tyszkiewicz, M., Batzner, S., Uhrin, M., Frellsen, J., Jung, N., Sanborn, S., Rackers, J., and Bailey, M. (2020b). Euclidean neural networks: e3nn.

Geiger, M., Spigler, S., d'Ascoli, S., Sagun, L., Baity-Jesi, M., Biroli, G., and Wyart, M. (2019). Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1):012115. Publisher: American Physical Society.

Geiger, M., Spigler, S., Jacot, A., and Wyart, M. (2020c). Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301. Publisher: IOP Publishing.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, D. M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org.

Györfi, L., Kohler, M., Krzyzak, A., and Walk, H. (2002). *A Distribution-Free Theory of Nonparametric Regression*. Springer series in statistics. Springer.

He, F., Liu, T., and Tao, D. (2019). Control batch size and learning rate to generalize well: Theoretical and empirical evidence. *Advances in Neural Information Processing Systems*, 32:1143–1152.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M., Ali, M., Yang, Y., and Zhou, Y. (2017). Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*.

Hu, W., Xiao, L., and Pennington, J. (2020). Provable benefit of orthogonal initialization in optimizing deep linear networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., et al. (2015). An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*.

J Liu, A., R Nagel, S., Saarloos, W., and Wyart, M. (2010). *The jamming scenario - an introduction and outlook*. OUP Oxford.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 8580–8589, USA. Curran Associates Inc.

Jacot, A., Simsek, B., Spadaro, F., Hongler, C., and Gabriel, F. (2020). Implicit regularization of random feature models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4631–4640. PMLR.

Ji, J., Kong, X., Zhang, Y., Xu, T., and Zhang, J. (2021). A new adaptive variable step size natural gradient BSS algorithm. *J. Intell. Fuzzy Syst.*, 41(1):57–68.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 8107–8116. Computer Vision Foundation / IEEE.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Krzakala, F. and Kurchan, J. (2007). Landscape analysis of constraint satisfaction problems. *Physical Review E*, 76(2):021122.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

Lee, J., Schoenholz, S. S., Pennington, J., Adlam, B., Xiao, L., Novak, R., and Sohl-Dickstein, J. (2020). Finite versus infinite neural networks: an empirical study. *arXiv preprint arXiv:2007.15801*.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. (2019). Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8572–8583. Curran Associates, Inc.

Lee, J. H., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2018). Deep neural networks as gaussian processes. *ICLR*.

Lerner, E., Düring, G., and Wyart, M. (2012). A unified framework for non-brownian suspension flows and soft amorphous solids. *Proceedings of the National Academy of Sciences*, 109(13):4798–4803.

Lerner, E., During, G., and Wyart, M. (2013). Low-energy non-linear excitations in sphere packings. *Soft Matter*, 9:8252–8263.

Liao, Z. and Couillet, R. (2018). The dynamics of learning: A random matrix approach. *arXiv preprint arXiv:1805.11917*.

Lipton, Z. C. (2016). Stuck in a what? adventures in weight space. *International Conference on Learning Representations*.

Luxburg, U. v. and Bousquet, O. (2004). Distance-based classification with lipschitz functions. *Journal of Machine Learning Research*, 5(Jun):669–695.

Mailman, M., Schreck, C. F., O'Hern, C. S., and Chakraborty, B. (2009). Jamming in systems composed of frictionless ellipse-shaped particles. *Phys. Rev. Lett.*, 102(25):255501.

Mallat, S. (2016). Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203.

Marr, D. and Poggio, T. (1979). A computational theory of human stereo vision. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 204(1156):301–328.

Martens, J. (2010). Deep learning via hessian-free optimization. In Fürnkranz, J. and Joachims, T., editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 735–742. Omnipress.

Mei, S., Misiakiewicz, T., and Montanari, A. (2019). Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on Learning Theory*, pages 2388–2464. PMLR. ISSN: 2640-3498.

Mei, S. and Montanari, A. (2021). The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*.

Mei, S., Montanari, A., and Nguyen, P.-M. (2018). A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671. Publisher: National Academy of Sciences Section: PNAS Plus.

Miller, B. K., Geiger, M., Smidt, T. E., and Noé, F. (2020). Relevance of rotationally equivariant convolutions for predicting molecular properties.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. (2019). Deep double descent: Where bigger models and more data hurt.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. (2018). Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*.

Neyshabur, B., Tomioka, R., Salakhutdinov, R., and Srebro, N. (2017). Geometry of optimization and implicit regularization in deep learning. *arXiv preprint arXiv:1705.03071*.

Nguyen, P.-M. (2019). Mean field limit of the learning dynamics of multilayer neural networks. *arXiv preprint arXiv:1902.02880*.

Nguyen, P.-M. and Pham, H. T. (2020). A rigorous framework for the mean field limit of multilayer neural networks. *arXiv preprint arXiv:2001.11443*.

Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Abolafia, D. A., Pennington, J., and Sohl-dickstein, J. (2019). Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*.

O'Hern, C. S., Silbert, L. E., Liu, A. J., and Nagel, S. R. (2003). Jamming at zero temperature and zero applied stress: The epitome of disorder. *Phys. Rev. E*, 68(1):011306–011324.

Park, D., Sohl-Dickstein, J., Le, Q., and Smith, S. (2019). The Effect of Network Width on Stochastic Gradient Descent and Generalization: an Empirical Study. In *International Conference on Machine Learning*, pages 5042–5051. PMLR. ISSN: 2640-3498.

Petrini, L., Favero, A., Geiger, M., and Wyart, M. (2021). Relative stability toward diffeomorphisms indicates performance in deep nets.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

Roberts, D. A. (2021a). SGD implicitly regularizes generalization error. *CoRR*, abs/2104.04874.

Roberts, D. A. (2021b). Sgd implicitly regularizes generalization error.

Rotskoff, G. M. and Vanden-Eijnden, E. (2018). Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*.

Sagun, L., Bottou, L., and LeCun, Y. (2017a). Singularity of the hessian in deep learning. *International Conference on Learning Representations*.

Sagun, L., Evci, U., Güney, V. U., Dauphin, Y., and Bottou, L. (2017b). Empirical analysis of the hessian of over-parametrized neural networks. *ICLR 2018 Workshop Contribution, arXiv:1706.04454*.

Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Shi, B., Bai, X., and Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Sirignano, J. and Spiliopoulos, K. (2020a). Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852.

Sirignano, J. and Spiliopoulos, K. (2020b). Mean Field Analysis of Neural Networks: A Law of Large Numbers. *SIAM Journal on Applied Mathematics*, 80(2):725–752. Publisher: Society for Industrial and Applied Mathematics.

Smidt, T. (2020). Euclidean symmetry and equivariance in machine learning. *ChemRxiv*.

Smidt, T. E., Geiger, M., and Miller, B. K. (2021). Finding symmetry breaking order parameters with euclidean neural networks. *Phys. Rev. Research*, 3:L012002.

Smith, S. L., Dherin, B., Barrett, D., and De, S. (2021a). On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*.

Smith, S. L., Dherin, B., Barrett, D. G. T., and De, S. (2021b). On the origin of implicit regularization in stochastic gradient descent. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Smith, S. L. and Le, Q. V. (2018). A bayesian perspective on generalization and stochastic gradient descent. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Soudry, D. and Carmon, Y. (2016). No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*.

Spigler, S., Geiger, M., d'Ascoli, S., Sagun, L., Biroli, G., and Wyart, M. (2019). A jamming transition from under- to over-parametrization affects generalization in deep learning. *Journal of Physics A: Mathematical and Theoretical*, 52(47):474001. Publisher: IOP Publishing.

Spigler, S., Geiger, M., and Wyart, M. (2020). Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124001. Publisher: IOP Publishing.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. (2018). Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds.

Topirceanu, A., Barina, G., and Udrescu, M. (2014). Musenet: Collaboration in the music artists industry. In *2014 European Network Intelligence Conference, ENIC 2014, Wroclaw, Poland, September 29-30, 2014*, pages 89–94. IEEE Computer Society.

Vapnik, V. (1999). An overview of statistical learning theory. *IEEE Trans. Neural Networks*, 10(5):988–999.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. (2018). 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Williams, C. K. (1997). Computing with infinite networks. In *Advances in neural information processing systems*, pages 295–301.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. (2017). Harmonic networks: Deep translation and rotation equivariance. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7168–7177.

Wyart, M. (2005). On the rigidity of amorphous solids. *Annales de Phys*, 30(3):1–113.

Wyart, M. (2012). Marginal stability constrains force and pair distributions at random close packing. *Physical review letters*, 109(12):125502.

Wyart, M., Silbert, L. E., Nagel, S. R., and Witten, T. A. (2005). Effects of compression on the vibrational modes of marginally jammed solids. *Physical Review E*, 72(5):051306.

Yang, G. (2019a). Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*.

Yang, G. (2019b). Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes.

Yang, G. (2020a). Tensor programs ii: Neural tangent kernel for any architecture.

Yang, G. (2020b). Tensor programs iii: Neural matrix laws.

Yang, G. and Hu, E. J. (2020). Feature learning in infinite-width neural networks.

Yang, G. and Hu, E. J. (2021). Tensor programs IV: feature learning in infinite-width neural networks. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 11727–11737. PMLR.

Yuan, L., Hou, Q., Jiang, Z., Feng, J., and Yan, S. (2021). Volo: Vision outlooker for visual recognition.

Zeravcic, Z., Xu, N., Liu, A. J., Nagel, S. R., and van Saarloos, W. (2009). Excitations of ellipsoid packings near jamming. *Europhys. Lett.*, 87(2):26001.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2021). Scaling vision transformers. *CoRR*, abs/2106.04560.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Zhang, R. (2019). Making convolutional networks shift-invariant again. *arXiv preprint arXiv:1904.11486*.

Mario GEIGER

Rue de Bassenges 36, 1024 Ecublens VD, Suisse
Mobile: +41(0)792907749
E-mail: mge@ik.me
Languages: French (mother tongue), English (fluent)

Education

October 2017 - present	PhD student with Matthieu Wyart École Polytechnique Fédérale de Lausanne *Loss landscape and symmetries in Neural Networks*
June 2017 - August 2017	Internship with Taco Cohen University von Amsterdam *Spherical CNN*
September 2015 - March 2017	Master Degree in Physics École Polytechnique Fédérale de Lausanne *5.57/6*
September 2012 - June 2015	Bachelor Degree in Physics École Polytechnique Fédérale de Lausanne *5.64/6*
September 2006 - June 2010	Federal Diploma of Vocational Education and Training École Polytechnique Fédérale de Lausanne *Physics laboratory assistant*

Teaching Assistant

- EPFL physics master, Practical work: pytorch and neural networks, 2017-2021
- EPFL physics master, Statistical Physics 2: phase transition, renormalization group, Spring 2020

Publications

- A. Kostro, M. Geiger, N. Jolissaint, M. Lazo, J. Scartezzini, Y. Leterrier, A. Schüler, "Embedded microstructures for daylighting and seasonal thermal control" Nonimaging Optics: Efficient Design for Illumination and Solar Concentration IX (2012)
- A. Paone, M. Geiger, R. Sanjines, A. Schüler, "Thermal solar collector with VO2 absorber coating and V1-xWxO2 thermochromic glazing–Temperature matching and triggering" Solar energy (2014)
- A. Kostro, M. Geiger, J. Scartezzini, A. Schüler, "CFSpro: ray tracing for design and optimization of complex fenestration systems using mixed dimensionality approach" Applied optics (2016)

- C. Schaefer, M. Geiger, T. Kuntzer, J. Kneib, "Deep convolutional neural networks as strong gravitational lens detectors" Astronomy & Astrophysics (2018)
- M. Baity-Jesi, L. Sagun, M. Geiger, S. Spigler, G. Arous, C. Cammarota, Y. LeCun, M. Wyart, G. Biroli, "Comparing dynamics: Deep neural networks versus glassy systems" International Conference on Machine Learning (2018)
- M. Weiler, M. Geiger, M. Welling, W. Boomsma, T. Cohen, "3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data" Conference on Neural Information Processing Systems (2018)
- T. Cohen, M. Geiger, J. Köhler, M. Welling, "Spherical CNNs" International Conference on Machine Learning (2018)
- M. Geiger, S. Spigler, S. d'Ascoli, L. Sagun, M. Baity-Jesi, G. Biroli, M. Wyart, "Jamming transition as a paradigm to understand the loss landscape of deep neural networks" Physical Review E (2019)
- R. Metcalf, M. Meneghetti, C. Avestruz, F. Bellagamba, C. Bom, E. Bertin, R. Cabanac, F. Courbin, A. Davies, E. Decencière, "The strong gravitational lens finding challenge" Astronomy & Astrophysics (2019)
- S. Spigler, M. Geiger, S. d'Ascoli, L. Sagun, G. Biroli, M. Wyart, "A jamming transition from under-to over-parametrization affects generalization in deep learning" Journal of Physics A: Mathematical and Theoretical (2019)
- T. Cohen, M. Geiger, M. Weiler, "A General Theory of Equivariant CNNs on Homogeneous Spaces" Conference on Neural Information Processing Systems (2019)
- B. Miller, M. Geiger, T. Smidt, F. Noé, "Relevance of rotationally equivariant convolutions for predicting molecular properties" arXiv preprint arXiv:2008.08461 (2020)
- M. Geiger, A. Jacot, S. Spigler, F. Gabriel, L. Sagun, S. d'Ascoli, G. Biroli, C. Hongler, M. Wyart, "Scaling description of generalization with number of parameters in deep learning" Journal of Statistical Mechanics: Theory and Experiment (2020)
- M. Geiger, S. Spigler, A. Jacot, M. Wyart, "Disentangling feature and lazy training in deep neural networks" Journal of Statistical Mechanics: Theory and Experiment (2020)
- S. Spigler, M. Geiger, M. Wyart, "Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm" Journal of Statistical Mechanics: Theory and Experiment (2020)
- J. Paccolat, L. Petrini, M. Geiger, K. Tyloo, M. Wyart, "Geometric compression of invariant manifolds in neural networks" Journal of Statistical Mechanics: Theory and Experiment (2021)
- M. Geiger, L. Petrini, M. Wyart, "Landscape and training regimes in deep learning" Physics Reports (2021)
- S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. Mailoa, M. Kornbluth, N. Molinari, T. Smidt, B. Kozinsky, "SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials" arXiv preprint arXiv:2101.03164 (2021)
- T. Smidt, M. Geiger, B. Miller, "Finding symmetry breaking order parameters with Euclidean neural networks" Physical Review Research (2021)

Software

- Lead developer of the open-source library for Euclidean Neural Networks https://github.com/e3nn/e3nn