EPFL

Thèse n° 8308

Biologically plausible unsupervised learning in shallow and deep neural networks

Présentée le 12 novembre 2021

Faculté des sciences de la vie
Laboratoire de calcul neuromimétique (SV/IC)
Programme doctoral en neurosciences

pour l'obtention du grade de Docteur ès Sciences

par

Bernd Albert ILLING

Acceptée sur proposition du jury

Prof. M. C. Gastpar, président du jury
Prof. W. Gerstner, directeur de thèse
Prof. K. Körding, rapporteur
Prof. B. Grewe, rapporteur
Prof. A. Mathis, rapporteur

École polytechnique fédérale de Lausanne

2021

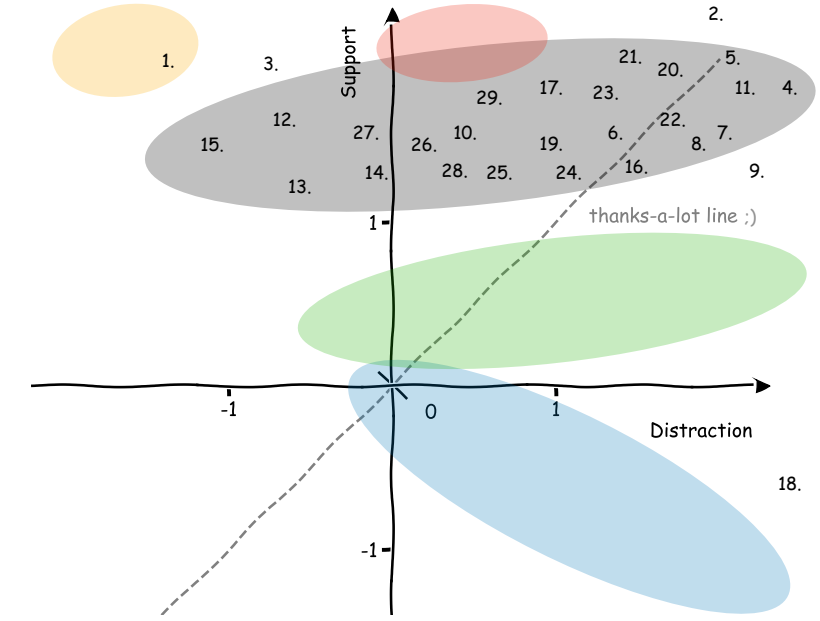
التكرار يعلم الحمار
‘Repetition teaches the donkey’

— Arabic saying and current interpretation of AI

To my grandma

Acknowledgements

A graph is worth a thousand words...



LCN

1. Wulfram
2. Johanni
3. Guillaume
4. Willem
5. Vasia
6. Olivia
7. Samuel
8. Flo
9. Marco
10. Tilo
11. Alex
12. Dane
13. Ho Ling
14. Hesam
15. Mohammad
16. Aditya
17. Chiara
18. Martin
19. Valentin
20. Noé
21. Alireza
22. Berfin
23. Christos
24. Flavio
25. Georgios
26. Sophia
27. Leoluca
28. Manu
29. Ariane
...

Konstanz people

der Fabi
Jonas
Patrick
Aaron
Benni
Jörg
Mario
Kathrin
Lugge
Liz
Julie
Caro
Simi
Lorenz
Timo
Lisa
Juliane
Jannis
Andy
Robin
Caro
Jan
Matze
Sabine
Aubergine
Nico
Chrissi
Lisa
Martha
Paul
Fabi
Geoffroy
Peter
...

Lausanne people

Karen
Bharath
Sandra
Bogdan
Mariam
Angie
Kiki
Sven
Fayez
Jakab
Sanne
Felix
Tanja
Julie
...

PhD Jury

M. Gastpar
W. Gerstner
K. Körding
B. Grewe
A. Mathis

Family

Oma
Mama
Papa
Susi
Jana
...

Lausanne, October 25, 2021 Bernd Illing

Abstract

The way our brain learns to disentangle complex signals into unambiguous concepts is fascinating but remains largely unknown. There is evidence, however, that hierarchical neural representations play a key role in the cortex. This thesis investigates biologically plausible models of unsupervised learning of hierarchical representations as found in the brain and modern computer vision models. We use computational modeling to address three main questions at the intersection of artificial intelligence (AI) and computational neuroscience.
The first question is: *What are useful neural representations and when are deep hierarchical representations needed?* We approach this point with a systematic study of biologically plausible unsupervised feature learning in a shallow 2-layer networks on digit (MNIST) and object (CIFAR10) classification. Surprisingly, random features support high performance, especially for large hidden layers. When combined with localized receptive fields, random feature networks approach the performance of supervised backpropagation on MNIST, but not on CIFAR10. We suggest that future models of biologically plausible learning should outperform such random feature benchmarks on MNIST, or that such models should be evaluated in different ways.
The second question is: *How can hierarchical representations be learned with mechanisms supported by neuroscientific evidence?* We cover this question by proposing a unifying Hebbian model, inspired by common models of V1 simple and complex cells based on unsupervised sparse coding and temporal invariance learning. In shallow 2-layer networks, our model reproduces learning of simple and complex cell receptive fields, as found in V1. In deeper networks, we stack multiple layers of Hebbian learning but find that it does not yield hierarchical representations of increasing usefulness. From this, we hypothesise that standard Hebbian rules are too constrained to build increasingly useful representations, as observed in higher areas of the visual cortex or deep artificial neural networks.
The third question is: *Can AI inspire learning models that build deep representations and are still biologically plausible?* We address this question by proposing a learning rule that takes inspiration from neuroscience and recent advances in self-supervised deep learning. The proposed rule is Hebbian, i.e. only depends on pre- and post-synaptic neuronal activity, but includes additional local factors, namely predictive dendritic input and widely broadcasted modulation factors. Algorithmically, this rule applies self-supervised contrastive predictive learning to a causal, biological setting using saccades. We find that networks trained with this generalised Hebbian rule build deep hierarchical representations of images, speech and video. We see our modeling as a potential starting point for both, new hypotheses, that can be tested experimentally, and novel AI models that could benefit from added biological realism.

Abstract

Keywords Representation learning, unsupervised and self-supervised learning, Hebbian learning, biologically plausible deep learning, hierarchical features, V1 simple and complex cells, sparse coding, temporal invariance learning, contrastive predictive learning

Zusammenfassung

Die Art und Weise wie unser Gehirn komplexe Signale in eindeutige Sinneseindrücke umwandelt ist faszinierend, jedoch weitesgehend unverstanden. Vieles deutet jedoch darauf hin, dass hierarchische neuronale Representationen im Cortex eine entscheidende Rolle spielen. Diese Thesis untersucht biologisch plausible Mechanismen zum unbeaufsichtigten Erlernen hierarchischer Representationen wie sie im Gehirn oder in modernen Computersichtmodellen vorkommen. Mithilfe computergestützter Modellierung behandeln wir drei Hauptfragen an der Schnittstelle zwischen künstlicher Intelligenz (AI) und theoretischer Neurowissenschaft.
Die erste Frage ist: *Was sind nützliche neuronal Representationen und wann sind tiefe, hierarchische Representationen notwendig?* Hierzu untersuchen wir biologisch plausible Algorithmen zum unbeaufsichtigten Erlernen von Mustern in 2-schichtigen neuronalen Netzen zur automatischen Erkennung von Ziffern (MNIST) und Objekten (CIFAR10). Zufallsmuster ermöglichen erstaunlich hohe Genauigkeiten in Netzen mit vielen Neuronen in der ersten Schicht. Wenn zudem lokal eingegrenzte rezeptive Felder verwendet werden, erreichen solche Netze auf MNIST die Genauigkeit von supervisierten Netzen, welche mit Fehler-Rückpropagierung trainiert wurden, jedoch nicht auf CIFAR10. Wir legen deshalb nahe, dass neue Modelle biologischen Lernens die Genauigkeit, die mit Zufallsmuster erreicht wird, eindeutig schlagen, oder in Zukunft bevorzugt mit anderen Mitteln evaluiert werden sollten.
Die zweite Frage ist: *Wie können hierarchische Representationen mit Mechanismen gelernt werden, für die es neurowissenschaftliche Belege gibt?* Zu dieser Frage kombinieren wir bekannte Modelle von simplen und komplexen V1 Zellen, basierend auf unbeaufsichtigter spärlicher Kodierung und zeitlichem Invarianzlernen, zu einem vereinenden Hebbschen Modell. In 2-schichtigen Netzen reproduziert unser Modell bekannte Eigenschaften und rezeptive Felder genannter V1 Zellen. In tieferen Netzen mit mehr als zwei Schichten, liefert unser Hebbsches Model jedoch keine hierarchischen Representationen, da die Nützlichkeit letzterer mit der Tiefe der Schicht abnimmt. Wir vermuten daher, dass übliches Hebbsches Lernen allein zu eingeschränkt ist um hierarchische Representationen zu lernen wie man sie aus höheren Arealen des Cortex oder aus künstlichen tiefen neuronalen Netzen kennt.
Die dritte Frage ist: *Können KI-Methoden uns zu neuen Modellen inspirieren welche sowohl funktional als auch biologisch plausibel sind?* Zur letzten Frage lassen wir uns vom kürzlichen Fortschritt im un- und selbstüberwachtes Lernen inspirieren. Wir stellen eine neue Hebbsche Lernregel vor, welche neben den üblichen pre- und post-synaptischen Faktoren noch zusützliche Faktoren mit einbezieht, und zwar prädiktive dendritische Signale sowie globale Modulatoren. Algorithmisch nutzt unsere Lernregel Saccaden, also Änderungen in

der Blickrichtung, um kontrastierende, prädiktive Kodierung in einem kausalen, biologischen Umfeld zu implementieren. Diese generalisierte Hebbsche Lernregel ist biologisch plausibel und liefert tiefe hierarchische Representationen von Bildern, Sprache und Videos, wie bekannt aus Cortex und tiefen neuronalen Netzen.
Wir sehen die mathematischen Modelle dieser Thesis sowohl als Startpunkt für wissenschaftliche Vorhersagen, die es experimentell zu überprüfen gilt, als auch als Ausgangspunkt für neue KI, welche von biologischer Plausibilität profitieren könnte.

Stichworte Repräsentationslernen, unbeaufsichtigtes und selbstüberwachtes Lernen, Hebbsches Lernen, biologisch plausibles tiefes Lernen, hierarchische Merkmale, einfache und komplexe V1-Zellen, spärliche Kodierung, zeitliches Invarianzlernen, kontrastives prädiktives Lernen

Résumé

La façon dont notre cerveau apprend à démêler des signaux complexes en concepts non ambigus est fascinante mais reste largement inconnue. Il existe cependant des preuves que les représentations neuronales hiérarchiques jouent un rôle clé dans le cortex. Cette thèse étudie les modèles biologiquement plausibles d'apprentissage non supervisé des représentations hiérarchiques tels qu'on les trouve dans le cerveau et dans les modèles modernes de vision par ordinateur. Nous utilisons la modélisation computationnelle pour aborder trois questions principales à l'intersection de l'intelligence artificielle (IA) et des neurosciences computationnelles.

La première question est : *Quelles sont les représentations neuronales utiles et quand des représentations hiérarchiques profondes sont-elles nécessaires ?* Nous abordons ce point avec une étude systématique de l'apprentissage non supervisé de caractéristiques biologiquement plausibles dans un réseau peu profond à deux couches sur la classification de chiffres (MNIST) et d'objets (CIFAR10). De manière surprenante, les caractéristiques aléatoires permettent d'obtenir des performances élevées, en particulier pour les grandes couches cachées. Lorsqu'ils sont combinés avec des champs réceptifs localisés, ces réseaux de caractéristiques aléatoires se rapprochent de la performance de la rétropropagation supervisée sur MNIST, mais pas sur CIFAR10. Nous suggérons que les futurs modèles d'apprentissage biologiquement plausible devraient surpasser les performances de tels repères de caractéristiques aléatoires sur MNIST, ou que de tels modèles devraient être évalués de manière différente.

La deuxième question est : *Comment les représentations hiérarchiques peuvent-elles être apprises avec des mécanismes soutenus par des preuves neuroscientifiques ?* Nous répondons à cette question en proposant un modèle hébbien unificateur, inspiré par des modèles communs de cellules simples et complexes de V1, basés sur un codage clairsemé et un apprentissage par invariance temporelle non supervisé. Dans les réseaux peu profonds à deux couches, notre modèle reproduit l'apprentissage des champs réceptifs des cellules simples et complexes, comme on le trouve dans V1. Dans les réseaux plus profonds, nous empilons plusieurs couches d'apprentissage hébbien mais nous constatons que cela ne produit pas de représentations hiérarchiques d'utilité croissante. Nous en déduisons que les règles hébbiennes standard sont trop limitées pour construire des représentations de plus en plus utiles, comme on l'observe dans les zones supérieures du cortex visuel ou dans les réseaux neuronaux artificiels profonds.

La troisième question est la suivante : *L'IA peut-elle inspirer des modèles d'apprentissage qui construisent des représentations profondes tout en étant biologiquement plausibles ?* Nous répondons à cette question en proposant une règle d'apprentissage qui s'inspire des neu-

rosciences et des avancées récentes en apprentissage profond auto-supervisé. La règle proposée est hébbienne, c'est-à-dire qu'elle ne dépend que de l'activité neuronale pré- et post-synaptique, mais inclut des facteurs locaux supplémentaires, à savoir une entrée dendritique prédictive et des facteurs de modulation largement diffusés. D'un point de vue algorithmique, cette règle applique l'apprentissage prédictif contrastif à un cadre causal et biologique en utilisant les saccades. Nous constatons que les réseaux formés avec cette règle hébbienne généralisée construisent des représentations hiérarchiques profondes des images, de la parole et de la vidéo.
Nous considérons notre modélisation comme un point de départ potentiel pour de nouvelles hypothèses, qui peuvent être testées expérimentalement, et pour de nouveaux modèles d'IA qui pourraient bénéficier d'un réalisme biologique accru.

Mots-clés Apprentissage des représentations, apprentissage non supervisé et auto-supervisé, apprentissage hébbien, apprentissage profond biologiquement plausible, caractéristiques hiérarchiques, cellules simples et complexes du V1, codage clairsemé, apprentissage par invariance temporelle, apprentissage prédictif contrastif

Contents

1 Introduction

1.1 Motivation

Understanding the brain and creating intelligent machines—these two challenges form a symbiosis and share an exciting history (Hassabis et al., 2017). In 1950, Alan Turing asked the provoking question whether machines can think, i.e. if machines can act like brains (Turing and Haugeland, 1950). Very soon, artificial intelligence (AI) made its way from science-fiction to an actual field of research and engineering in the late 1950s (Russell and Norvig, 2002), see Figure 1.1. AI offers unprecedented opportunities but also bears considerable threat, which is why a thorough understanding of artificial, as well as biological, intelligence is needed, see e.g. Russell (2019) and Figure 1.1 **a** (or **b**).

During the last decades, AI has seen multiple phases of excitement and depressions ('AI winters') (Russell, 2019), but recently enjoys a(nother) heyday, thanks to the revolutionising success with artificial neural networks (LeCun et al., 2015; Schmidhuber, 2015). Interestingly, artificial neural networks originated from early attempts to mimic the computations of biological neurons and neural networks, such as the *perceptron* (Rosenblatt, 1958) or the *Neocognitron* (Fukushima, 1988). Throughout time, AI surely took its own paths, far from biological inspiration or realism, driven by performance criteria. In recent years, however, the symbiotic loop seems to close: AI models inspire and help brain research to reach unprecedented understanding (Yamins et al., 2014; Marblestone et al., 2016; Hassabis et al., 2017; Richards et al., 2019). Some voices in the community even reverse Turing's original question, asking whether brains potentially implement successful AI methods, such as error-backpropagation (Hinton, 2007; Lillicrap et al., 2020).

The intersection of AI and neuroscience now defines a productive and growing research field (Hassabis et al., 2017). However, despite recent breakthroughs in both neuroscience methods (Deisseroth, 2011; Poldrack, 2012; Mathis et al., 2018; Steinmetz et al., 2018) and AI (LeCun et al., 2015; Schmidhuber, 2015), the current understanding of the brain and intelligence is still in its infancy and it is unclear how to overcome this with current methods (Jonas and Kording, 2017).

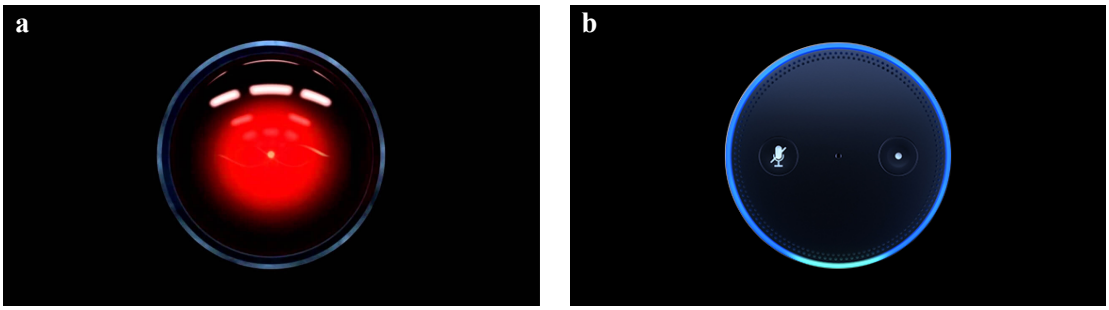


Figure 1.1: Artificial intelligence (AI), as conceived by **a** science fiction in 1968 (*HAL 9000* in Stanley Kubrick's *2001: A Space Odyssey*, image source: typeform.com), and **b** Amazon in 2013 onwards (*Amazon Alexa virtual assistant AI*, image source: edfenergy.com)

In the rest of this introduction, we introduce key concepts and literature that is necessary to understand the state of the art and the current challenges of computational brain modeling. The next section, section 1.2, presents the need and evidence for feature hierarchies in the brain and gives a compact overview of the connection between deep learning and neuroscience. In section 1.3, we introduce models of biologically plausible mechanisms to learn such hierarchies in real brains. We finally state the main contributions and the structure of this thesis (section 1.4). To keep the introduction concise, we deliberately keep explanations at a high level and refer to text books or review articles for further reading.

1.2 Hierarchical representations—in the brain and in deep neural networks

Our brains handle everyday tasks like vision, audition or motor-control with such ease, that we do not even realise how complex the tasks are and how many processing steps are required to solve them. There is a consensus in the field, that solving a task like vision requires hierarchical neural representations, and that such representations are in fact present in the visual cortex of the brain (Riesenhuber and Poggio, 1999; Yamins et al., 2014; Richards et al., 2019). Following, amongst others, (Fukushima, 1988; LeCun, 2012; Lillicrap et al., 2020; Payeur et al., 2021), we define such a *hierarchical representation* as a layered structure of features, that (i) builds higher-level features out of lower-level ones, and (ii) provides more useful features in higher layers, where *usefulness* depends on the task. In the context of vision, we measure usefulness as decodability of the stimulus object identity with a linear classifier.

Hubel and Wiesel (1962) gave a seminal example of hierarchical processing in the cat's primary visual cortex (V1). They investigated the *receptive fields* of neurons in V1, i.e. which stimuli such cells respond to, and found qualitatively different stimuli for different cells, see Figure 1.2. *Simple cells* responded to bright bars of a certain orientation *and* position, whereas *complex cells* were only selective to bar orientation, showing local positional invariance. We thus observe a hierarchy of features, where complex features (oriented bars regardless of location)

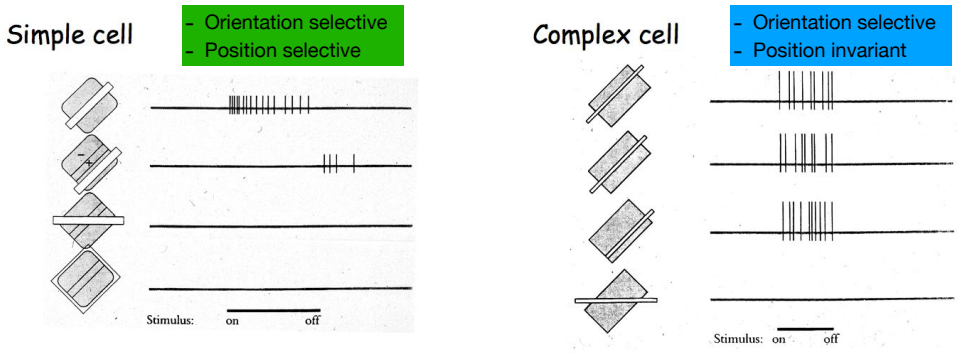


Figure 1.2: Simple and complex cells in primary visual cortex (V1). For each cell type, the presented stimuli and corresponding response are shown on the left and right respectively. We see that both cell types respond selectively to stimulus orientation, i.e. only if the stimulus (depicted as the white bar) aligns with the preferred orientation of the specific cell (depicted as the gray 'receptive field'). Simple cells are position selective since a correctly oriented, but misplaced stimulus fails to trigger a cell response. This is different for complex cells, whose response is position invariant over a certain range. Image taken from https://www.cns.nyu.edu/~david/courses/perception/lecturenotes/V1/lgn-V1.html.

are composed of simpler ones (oriented bars at certain locations). The key property of this hierarchical representation is the combination of *selectivity* and *invariance* at different levels. The generalisation of this concept to more than two processing steps, and hence to deep neural networks, leads to the most promising models of visual cortex, from V1 to the inferior temporal (IT) cortex (Hubel and Wiesel, 1962; Fukushima, 1980; Riesenhuber and Poggio, 1999; Li and DiCarlo, 2008), see Figure 1.3 **a** (top).

These hierarchical models of the cortex inspired early computer vision models like the 'Neocognitron' (Fukushima, 1980, 1988) or algorithms based on HMAX (Serre et al., 2007), see Figure 1.3 **a** (bottom). Importantly, this inspiration also led to the modern convolutional neural networks (CNN) (LeCun, 1989) that drive the recent deep learning revolution. Together with major improvements in hardware, such as GPU acceleration, CNNs surpassed previous algorithms by huge margins on difficult tasks in computer vision, games and language translation (Ciresan et al., 2011; Krizhevsky et al., 2012; LeCun et al., 2015; Schmidhuber, 2015; Vaswani et al., 2017; Silver et al., 2017; Mnih et al., 2015), see Figure 1.3 **b**. We emphasise that CNNs are not constrained to classical supervised learning, where every training sample has to be labelled (LeCun et al., 2015). They also revolutionised reinforcement learning, where only a scalar reward is available as training signal (Mnih et al., 2015), as well as unsupervised or self-supervised learning, where no external training signals are given (Caron et al., 2018; Zhuang et al., 2019; Van den Oord et al., 2018)

Interestingly, such goal-driven deep CNNs also turn out to be promising models of the brain (Yamins and DiCarlo, 2016; Zhuang et al., 2021). They explain neural data much better than traditional models of the cortex, such as the model of Riesenhuber and Poggio (1999), despite

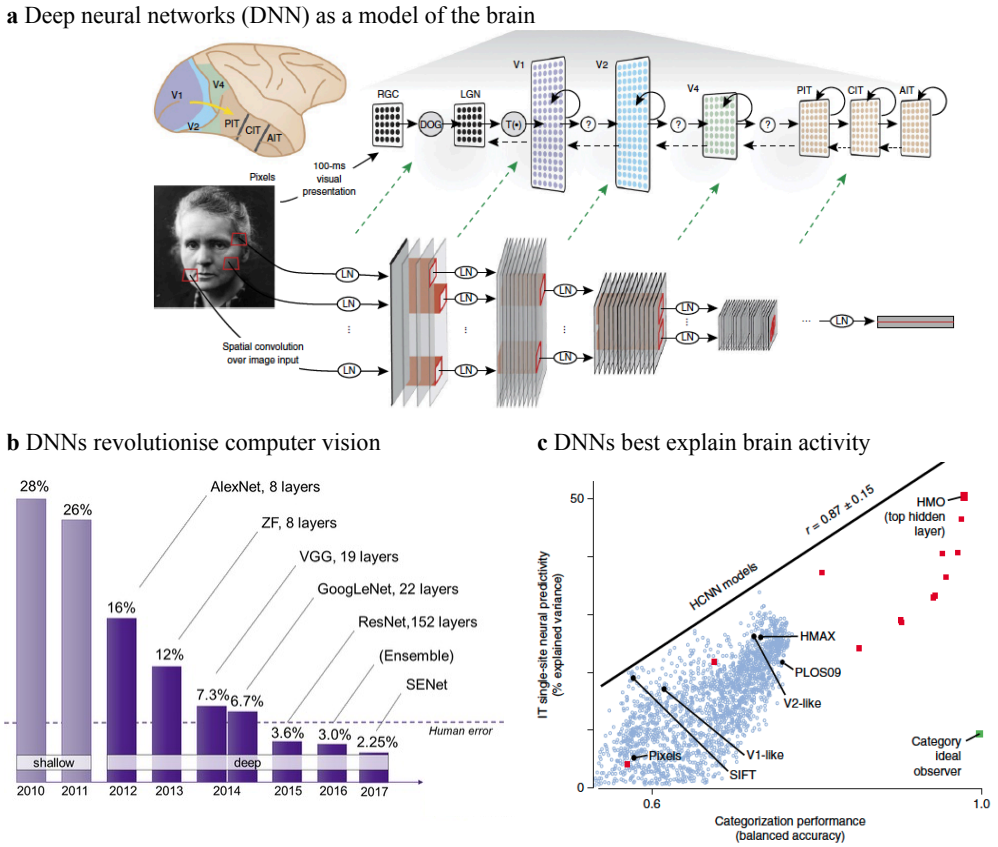


Figure 1.3: Deep neural networks (DNN) revolutionise machine learning and computational neuroscience. **a** Comparison of the hierarchical organisation of the visual cortex (top) and modern deep convolutional neural networks (CNN, bottom). In a CNN, each convolutional layer is made up of a linear-nonlinear (LN) combination of simple operations such as filtering, thresholding, pooling and normalization. **b** The use of DNNs/CNNs led to a major performance leap in machine learning tasks such as computer vision. E.g. on ImageNet classification (https://www.image-net.org/challenges/LSVRC/2012/index.php), DNNs clearly outperform traditional (shallow) computer vision models and even surpass human performance (image source: https://semiengineering.com/new-vision-technologies-for-real-world-applications/). **c** CNNs (HMO, red boxes) explain brain activity, as captured by fMRI, better than traditional models (SIFT, V1-like, HMAX), especially for higher visual cortices (e.g. IT cortex). Interestingly, the DNNs are not fit to cortical data but optimised on an auxiliary task such as object classification (Yamins et al., 2014). Image panels a & c taken from Yamins and DiCarlo (2016).

the fact that they are not fit to cortical data, but optimised on an auxiliary task such as object classification (Yamins et al., 2014), see Figure 1.3 **c**. These models, however, do not give a satisfying description of learning, i.e. of the emergence of such representations, since they rely on biologically unrealistic error-backpropagation (Crick, 1989).

Error-backpropagation (BP) (Hinton, 2002) is *the* workhorse of deep learning since it efficiently solves the *credit assignment problem*: given an objective (e.g. a regression cost function), and

given a strategy to optimize this objective (e.g. gradient descent), how do we have to update the parameters, deeply hidden in the nonlinear network, in order to improve on the objective? BP solves this by an efficient application of the chain rule, making the training of large CNNs feasible and successful. However, standard BP is not a realistic model for learning in biological neural networks, as diagnosed early on by Crick (1989). The specific question whether the brain can implement BP, at least approximately, has created a whole branch of research (Hinton, 2007; Marblestone et al., 2016; Lillicrap et al., 2016; Bartunov et al., 2018; Richards et al., 2019; Lillicrap et al., 2020; Kunin et al., 2020). However, there are potentially other ways to create deep representations that are more biologically plausible to begin with.

1.3 Local learning rules

Growing insight into the astounding power and complexity of cortical representations led to another question (Hebb, 1949): how does such complex processing arise? Is it fine-tuned by evolution, stored in the DNA and given to every animal by birth? Or does it evolve during lifetime, based on experience-dependent learning? As for many nature-vs-nurture discussions, the answer probably lies somewhere in the middle. However, there are good reasons to think that learning must be a big part of it: First, the DNA simply cannot store a plan of the connectivity of the *whole* cortex (Zador, 2019). Second, there is experimental evidence that cortical processing can be altered during development in the visual (Wiesel and Hubel, 1963; Hubel and Wiesel, 1963; Stryker and Harris, 1986; Fagiolini et al., 1994; Li and DiCarlo, 2008; Poort et al., 2015; Matteucci and Zoccolan, 2020), auditory (Recanzone et al., 1993; Buonomano and Merzenich, 1998), somatosensory (Merzenich et al., 1984) and motor cortex (Sanes and Donoghue, 2000). Hence, cortical representations are at least partly learned and plastic.

Hebbian learning The main mechanism of learning in the brain is synaptic plasticity, i.e. changes in the connection strengths between neurons (Hebb, 1949; Stent, 1973; Turrigiano et al., 1998; Citri and Malenka, 2008). In the framework of *Hebbian* learning, these changes depend only on the recent state of the pre- and post-synaptic neurons, as found in experiments (Sjöström et al., 2001; Caporale and Dan, 2008; Markram et al., 2011). Thus, a Hebbian learning rule for the update of a connection W_{ij}, connecting neuron j with neuron i, traditionally takes the form $\Delta W_{ij} = \text{post}_i \cdot \text{pre}_j$, with a pre- and a post-synaptic factor, e.g. representing recent neural activity. Other factors can influence synaptic plasticity as well, for instance in the form of a modulating third factor related to reward, attention or other high-level signals (Kuśmierz et al., 2017; Gerstner et al., 2018).

Hebbian learning rules can yield interesting patterns, for instance training a neuron to extract the first principal component of its inputs (Oja, 1982). However, in a network of multiple neurons that share the same input and learning rule, Hebbian learning faces a problem: how can one avoid obtaining multiple neurons that extract the same feature? A successful way to avoid this redundancy is to introduce competition between the neurons to 'let them

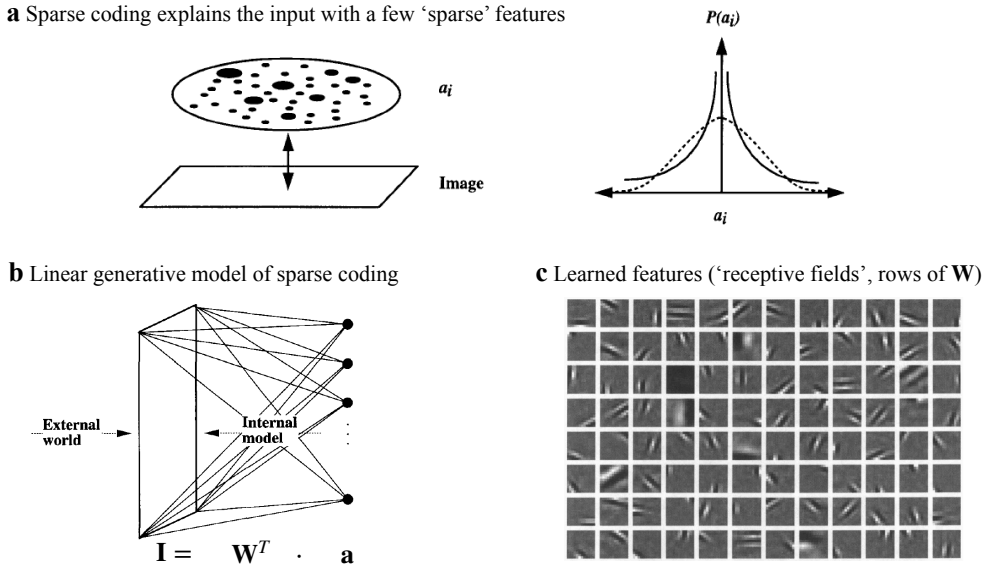


Figure 1.4: Sparse coding (SC) as an algorithm and as a model of perception. **a** (left) SC explains, or *reconstructs*, an input as a linear combination of features i with coefficients a_i, called the *code*. The coefficients implement a 'sparse' code in that only a few a_i are non-zero or large (radii of dots). (right) The distribution $P(a_i)$ of coefficients (solid line) has high kurtosis and is heavy-tailed, i.e. generates more zeros and larger values than a Gaussian distribution (dashed). **b** The reconstruction of the input can be seen as an internal linear generative model. The image I is reconstructed by the summed multiplication of the feature of each neuron i (rows of feature matrix W) with the code a_i. **c** In a neural network implementation of sparse coding, see Figure 3.1**a**, the rows of the feature matrix W can be interpreted as the synaptic input weights defining each neuron's *receptive field*. If each neuron connects to an image patch as the input, we can reshape and plot these synaptic weights as an image patch as well, as done in the figure. Each of the 8×12=96 tiles shows the input weights of one neuron in the SC layer exhibiting the typical Gabor-like filters. All figure panels adapted from Olshausen and Field (1997).

know' about each other. Such competition breaks the symmetry of the network and forces the neurons to extract different features. Competition is thought to be the result of mutual inhibition between neurons (see Figure 3.1) which, depending on the model, implements self-organising maps (Kohonen, 1982), (k-)winner-takes-all circuits (Majani et al., 1989), extraction of principal or independent components (Sanger, 1989; Hyvärinen and Oja, 1998; Isomura and Toyoizumi, 2018; Pehlevan et al., 2018) or sparse coding (Földiák, 1990; Olshausen and Field, 1997; Pehlevan and Chklovskii, 2015).

Sparse coding and simple cells Computational modeling of simple cells with competitive Hebbian learning has a long tradition (Von der Malsburg, 1973; Linsker, 1986a,b,c; Miller, 1994). Olshausen and Field (1996, 1997) introduced a normative and mathematically elegant simple cell model based on sparse coding (SC), which could be trained on realistic input data

and quickly became *the* computational model of simple cell learning. SC optimises a two-fold objective: reconstruct an input as a linear combination of features, using coefficients, called the code, that are as sparse as possible, see Figure 1.4. 'Sparse' usually refers to minimizing the L^0-norm of the code, which means keeping as few non-zero entries as possible. Since this constraint is non-convex, it is often relaxed in practice to an L^1-norm penalty on the code (Olshausen and Field, 1996). After training, every input is encoded with only a few non-zero elements and the distribution of the coefficients of the code has high kurtosis and is heavy-tailed, see Figure 1.4.

Interestingly, SC can be implemented with a neural network as in Figure 3.1 **a**, with neural activity representing the code, and the synaptic weights representing the features. Such a network exhibits competitive dynamics through mutual inhibition and the learning rule to update the synaptic weights is Hebbian (Olshausen and Field, 1997). Figure 1.4 **c** shows the synaptic weights of a network that was trained on natural image patches. Each neuron takes an image patch of size $n \times n$ as input and passes this information through its $n \times n$ input synapses. These input synapses are often referred to as (the linear part of) the neural *filter* or *feature*. To visualize such filters, we plot these synaptic weights as an $n \times n$ image patch. We see that the SC model learned Gabor-filter-like features exhibiting orientation, position and spatial frequency tuning, which are typical for simple cell receptive fields. We note that the feedforward pass, i.e. the *inference* of the sparse representation from the input, in such competitive networks is highly non-linear. However, Zylberberg et al. (2011) presented proof that, for certain sparse coding models, the set of synaptic weights ('projective fields') are statistically equivalent to the classical 'receptive fields', as defined by e.g. spike-triggered average (Hubel and Wiesel, 1962; Gerstner et al., 2014).

We note that several studies formulated other objectives that also lead to competitive dynamics, sparse codes and local learning rules, e.g. similarity matching (Pehlevan and Chklovskii, 2015; Pehlevan et al., 2018), independent component analysis (ICA) (Isomura and Toyoizumi, 2018) or in general objectives that depend non-linearly on post-synaptic neural activity (Brito and Gerstner, 2016). In some cases, a clear relation between the objective and the non-linearity of the modeled neurons can be found (Rozell et al., 2008; Brito and Gerstner, 2016).

Invariance learning and complex cells Having addressed learning of selective features by sparse coding, learning of invariances remains the second problem to be solved. The challenge here is to learn features that are invariant to low-level perturbations, such as jitter or light conditions, *yet* stay selective to higher level information, such as pose or even object identity (DiCarlo and Cox, 2007). To tackle this question, let us start by looking at the problem and environment that actual animals face: inferring object identity (e.g. a fellow ape) from noisy and dynamic input (e.g. retinal input), see Figure 1.5. This observation inspired temporal invariance learning (Földiák, 1991; Wallis and Rolls, 1997; Körding and König, 2001), which exploits temporal correlations and redundancies in natural input sequences to extract features that are stable over time. The first mathematical description that the invariance properties

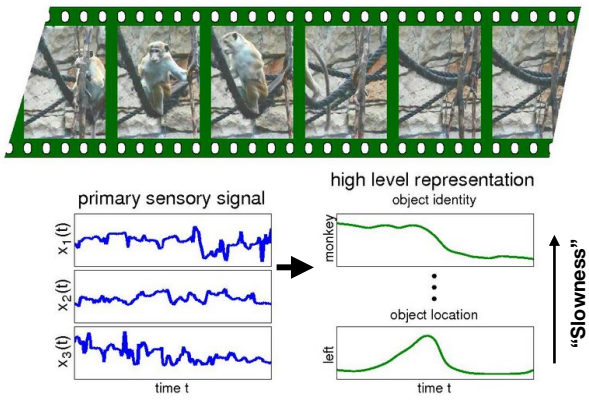


Figure 1.5: Illustration of slow feature analysis (SFA). High-dimensional temporal data, such as videos, contain fast features (e.g. pixel noise, object jitter or location) and slow features (e.g. monkey in the frame). SFA aims at extracting the slowest features that carry the most semantic information. Image taken from http://www.scholarpedia.org/article/File:SlowFeatureAnalysis-SlownessPrinciple.jpeg

of V1 complex cells could result from temporal learning was introduced by Földiák (1991) as the 'trace rule'. This learning rule heuristically extracts slow features by binding temporally adjacent inputs together with a slowly decaying memory trace. The update computation is local, i.e. consists of pre- and post-synaptic factors, and uses said memory trace as the post-synaptic factor, instead of instantaneous post-synaptic neural activity, as in classic Hebbian learning.

It was only in the 2000s, that slowness learning became more principled through slow feature analysis (SFA) by Wiskott and Sejnowski (2002). SFA was originally introduced as an offline algorithm, only vaguely related to biologically realistic neural implementations and far from local learning rules. Berkes and Wiskott (2005) found that SFA leads to complex cell properties, however, the algorithm was applied offline and no biological model for learning was proposed. A clear connection between SFA and learning rules was introduced by Sprekeler et al. (2007) and Sprekeler and Wiskott (2011) showing that SFA can be interpreted as a Hebbian learning rule and that this rule resembles Földiák's trace rule, giving the latter a normative justification.

Invariance learning with trace-like rules has inspired multiple models of V1. Most models consist of two layers that combine a first layer of simple cell learning with a second layer of invariant complex cell learning (Körding and König, 2001; Einhäuser et al., 2002; Sullivan and de Sa, 2004; Masquelier et al., 2007; Chen et al., 2018). These models use different rules for simple cell learning (see previous paragraph) or sometimes even hard-code the weights of the first layer to Gabor-filters. Invariance learning in the complex cell layer follows the idea of SFA, either explicitly or through trace-like rules. As a more principled approach, there are attempts to design generative models that yield learning rules for slowness learning or in general complex cell properties (Hurri and Hyvärinen, 2003; Hosoya, 2012).

We note in passing, that models independent of slowness learning also achieve complex cell behaviour. One way is partly hard-coding the invariance of complex cells through an energy model which quadratically combines simple cell filters with different phases (Adelson and Bergen, 1985; Concetta Morrone and Burr, 1988; Heeger, 1992), e.g. in 2-layer networks implementing sparse coding (Hyvärinen and Hoyer, 2001), or optimizing a stability criterion

(Körding et al., 2004; Lies et al., 2014). Another way is learning spatial maps of simple and complex cell layers, with smooth changes of orientation preferences, and using spatial adjacency to learn complex cell connectivity (Antolik and Bednar, 2011; Chandrapala and Shi, 2015). However, the update rules used in these models usually deviate from the Hebbian learning framework and are partly non-local.

Models of local learning in hierarchies beyond V1 Despite the progress in modeling V1 simple and complex cells, the step towards understanding the representations and learning in higher areas like V4 or IT has so far remained a challenge. Early attempts to create deep feature hierarchies date back to early computer vision models like the 'Neocognitron' (Fukushima, 1980) or to models of the visual cortex beyond the primary visual cortex, like 'HMAX' (Riesenhuber and Poggio, 1999; Serre et al., 2007). These models took inspiration from the hierarchical organisation and invariant representations of the visual cortex and in turn inspired a large number of vision models, such as VisNet, which applies Földiák's trace-rule to an HMAX architecture (Wallis and Rolls, 1997; Rolls and Milward, 2000; Robinson and Rolls, 2015), deep template matching (Serre et al., 2007) and multi-layer convolutional spike-timing dependent plasticity (Masquelier and Thorpe, 2007; Tavanaei and Maida, 2016). Unsupervised slowness learning is even considered as a model for high-level areas like IT cortex (Li and DiCarlo, 2008). However, classical competitive Hebbian learning in deep networks was found to be limited, both for brain modeling (Yamins et al., 2014), and machine learning, as models that implement deep sparse coding (He et al., 2014; Boutin et al., 2019) or other versions of multi-layer Hebbian learning (Bahroun et al., 2017; Stuhr and Brauer, 2019; Amato et al., 2019; Obeid et al., 2019; Talloen et al., 2021; Lagani et al., 2021; Miconi, 2021) show only limited usage of the network depth, or even degrading performance for higher layers.

1.4 Thesis structure and contribution

The present thesis is placed at the intersection of AI and computational neuroscience and covers three main questions: (i) Which neural representations are useful for tasks like vision, and when are deep hierarchical representations needed?, (ii) How can such representations evolve in the brain with learning rules that are compatible with neuroscientific evidence?, and (iii) Can AI inspire us to novel learning models that build deep representations and are still biologically plausible?

In chapter 2, we address question (i) with a *bottom-up* approach. Bottom-up modeling starts from detailed neuroscientific knowledge of single components (e.g. a neuron model and a learning rule) and builds up a model (e.g. a neural network) from these components—hoping that the result fulfils expectations (e.g. the network learns to identify ostriches). More specifically, we investigate how far we can go with different biologically plausible learning mechanisms in shallow 2-layer networks.

Question (ii) is addressed in chapter 3, again with bottom-up modeling. The idea is to build

a hierarchy of features with competitive temporal Hebbian learning. We start again with a 2-layer network and demonstrate emerging properties as found in simple and complex cells in V1. To go beyond V1 modeling, we then investigate training of deep convolutional networks with such learning rules.

Finally, we address question (iii) in chapter 4. This time, we pursue a *top-down* strategy, which takes inspiration from a model that functions, but cannot serve as a neuroscientific model per se (e.g. a biologically implausible AI algorithm, that correctly identifies ostriches), and then breaks it down into small, interpretable building blocks (e.g. neuron models). To this end, we gain inspiration from recent unsupervised deep learning models using contrastive prediction (Van den Oord et al., 2018; Löwe et al., 2019) to develop a novel neo-Hebbian learning rule that successfully learns deep representations.

In chapter 5, we summarize our main results and draw final conclusions.

2 Biologically plausible deep learning—but how far can we go with shallow networks?

Paper information

Authors **Bernd Illing**, Wulfram Gerstner, Johanni Brea

Abstract Training deep neural networks with the error backpropagation algorithm is considered implausible from a biological perspective. Numerous recent publications suggest elaborate models for biologically plausible variants of deep learning, typically defining success as reaching around 98% test accuracy on the MNIST data set. Here, we investigate how far we can go on digit (MNIST) and object (CIFAR10) classification with biologically plausible, local learning rules in a network with one hidden layer and a single readout layer. The hidden layer weights are either fixed (random or random Gabor filters) or trained with unsupervised methods (Principal/Independent Component Analysis or Sparse Coding) that can be implemented by local learning rules. The readout layer is trained with a supervised, local learning rule. We first implement these models with rate neurons. This comparison reveals, first, that unsupervised learning does not lead to better performance than fixed random projections or Gabor filters for large hidden layers. Second, networks with localized receptive fields perform significantly better than networks with all-to-all connectivity and can reach backpropagation performance on MNIST. We then implement two of the networks - fixed, localized, random & random Gabor filters in the hidden layer - with spiking leaky integrate-and-fire neurons and spike timing dependent plasticity to train the readout layer. These spiking models achieve > 98.2% test accuracy on MNIST, which is close to the performance of rate networks with one hidden layer trained with backpropagation. The performance of our shallow network models is comparable to most current biologically plausible models of deep learning. Furthermore, our results with a shallow spiking network provide an important reference and suggest the use of datasets other than MNIST for testing the performance of future models of biologically plausible deep learning.

Chapter 2. Biologically plausible deep learning—but how far can we go with shallow networks?

Keywords Deep learning, local learning rules, random projections, unsupervised feature learning, spiking networks, MNIST, CIFAR10

Author contributions All authors designed the project and wrote the paper. Simulations were done by BI.

Publication This chapter was published as a paper in Neural Networks (Illing et al., 2019).

Funding This research was supported by the Swiss National Science Foundation (no. 200020_165538 and 200020_184615) and by the European Union Horizon 2020 Framework Program under grant agreement no. 785907 (HumanBrain Project, SGA2).

2.1 Introduction

While learning a new task, synapses deep in the brain undergo task-relevant changes (Hayashi-Takagi et al., 2015). These synapses are often many neurons downstream of sensors and many neurons upstream of actuators. Since the rules that govern such changes deep in the brain are poorly understood, it is appealing to draw inspiration from deep artificial neural networks (DNNs) (LeCun et al., 2015). DNNs and the cerebral cortex share that information is processed in multiple layers of many neurons (Yamins and DiCarlo, 2016; Kriegeskorte, 2015) and that learning depends on changes of synaptic strengths (Hebb, 1949). However, learning rules in the brain are most likely different from the backpropagation algorithm (Crick, 1989; Marblestone et al., 2016; Whittington and Bogacz, 2019). Furthermore, biological neurons communicate by sending discrete spikes as opposed to real-valued numbers used in DNNs. Differences like these suggest that there exist other, possibly nearly equally powerful, algorithms that are capable to solve the same tasks by using different, more biologically plausible mechanisms. Thus, an important question in computational neuroscience is how to explain the fascinating learning capabilities of the brain with biologically plausible network architectures and learning rules. Moreover from a pure machine learning perspective there is increasing interest in neuron-like architectures with local learning rules, mainly motivated by the current advances in neuromorphic hardware (Nawrocki et al., 2016).

Image recognition is a popular task to test the performance of neural networks. Because of its relative simplicity and popularity, the MNIST dataset (28×28-pixel grey level images of handwritten digits, LeCun (1998)) is often used for benchmarking. Typical performances of existing models are around 97-99% classification accuracy on the MNIST test set (see section 2.2 and Table 2.2). Since the performances of many classical DNNs trained with backpropagation (but without data augmentation or convolutional layers, see table in LeCun (1998)) also fall in this region, accuracies around these values are assumed to be an empirical signature of backpropagation-like deep learning (Lillicrap et al., 2016; Sacramento et al., 2018; Tavanaei et al., 2018; Whittington and Bogacz, 2019). It is noteworthy, however, that several of the most promising approaches that perform well on MNIST have been found to fail on harder tasks (Bartunov et al., 2018) or at least need major modifications to scale to deeper networks (Moskovitz et al., 2018).

There are two obvious alternatives to supervised training of all layers with backpropagation. The first one is to fix weights in the first layer(s) at random values , as proposed by general approximation theory (Barron, 1993) and the extreme learning field (Huang et al., 2006). The second alternative is unsupervised training in the first layer(s). In both cases, only the weights of a readout layer are learned with supervised training. Unsupervised methods are appealing since they can be implemented with local learning rules, see e.g. “Oja’s rule” (Oja, 1982; Sanger, 1989) for principal component analysis, nonlinear extensions for independent component analysis (Hyvärinen and Oja, 1998) or algorithms in Olshausen and Field (1997); Rozell et al. (2008); Liu and Jia (2012); Brito and Gerstner (2016) for sparse coding. A single readout layer can be implemented with a local rule as well. A candidate is the delta-rule (also called “perceptron rule”), which may be implemented by pyramidal spiking neurons with

dendritic prediction of somatic spiking (Urbanczik and Senn, 2014). Since straightforward stacking of multiple fully connected layers of unsupervised learning does not reveal more complex features (Olshausen and Field, 1997) we focus here on networks with a single hidden layer (see also Krotov et al. (2019)).
The main objective of this study is to see how far we can go with networks with a single hidden layer and biologically plausible, local learning rules, preferably using spiking neurons. To do so we first compare the classification performance of different rate networks: networks trained with backpropagation, networks with fixed random projections or random Gabor filters in the hidden layer and networks where the hidden layer is trained with unsupervised methods (subsection 2.3.1). Since sparse connectivity is sometimes superior to dense connectivity (Litwin-Kumar et al., 2017; Bartunov et al., 2018) and successful convolutional networks leverage local receptive fields, we investigate sparse connectivity between input and hidden layer, where each hidden neuron receives input only from a few neighboring pixels of the input image (subsection 2.3.2). Finally we implement the simplest, yet promising and biologically plausible models - localized random projections and random Gabor filters - with spiking leaky integrate-and-fire neurons and spike timing dependent plasticity (subsection 2.3.3). We discuss the performance and implications of this simplistic model with respect to current models of biologically plausible deep learning.

2.2 Related work

In recent years, many biologically plausible approaches to deep learning have been proposed, see e.g. Marblestone et al. (2016); Whittington and Bogacz (2019); Tavanaei et al. (2018) for reviews. Existing approaches usually use either involved architectures or elaborate mechanisms to approximate the backpropagation algorithm. Examples include the use of convolutional layers (Tavanaei and Maida, 2016; Tavanaei et al., 2018; Lee et al., 2018; Kheradpisheh et al., 2018) (and tables therein), dendritic computations (Hussain et al., 2014; Guerguiev et al., 2017; Sacramento et al., 2018) or backpropagation approximations such as feedback alignment (Lillicrap et al., 2016; Baldi et al., 2016; Nok; Samadi et al., 2017; Kohan et al., 2018; Bartunov et al., 2018) equilibrium propagation (Scellier and Bengio, 2017), membrane potential based backpropagation (Lee et al., 2016), restricted Boltzmann machines and deep belief networks (O'Connor et al., 2013; Neftci et al., 2014), (localized) difference target propagation (Lee et al., 2015; Bartunov et al., 2018), using reinforcement-signals (Rombouts et al., 2015; Pozzi et al., 2018) or approaches using predictive coding (Whittington and Bogacz, 2017). Many models implement spiking neurons to stress bio-plausibility (Liu et al., 2016; Neftci et al., 2017; Kulkarni and Rajendran, 2018; Wu et al., 2018; Liu and Yue, 2018; Tavanaei et al., 2018) (and tables therein) or coding efficiency (O'Connor et al., 2017). The conversion of DNNs to spiking neural networks (SNN) after training with backpropagation (Diehl et al., 2015) is a common technique to evade the difficulties of training with spikes. Furthermore, there are models including recurrent activity (Spoerer et al., 2017; Bellec et al., 2018), starting directly from realistic circuits (Delahunt and Kutz, 2018), or combining unsupervised and supervised train-

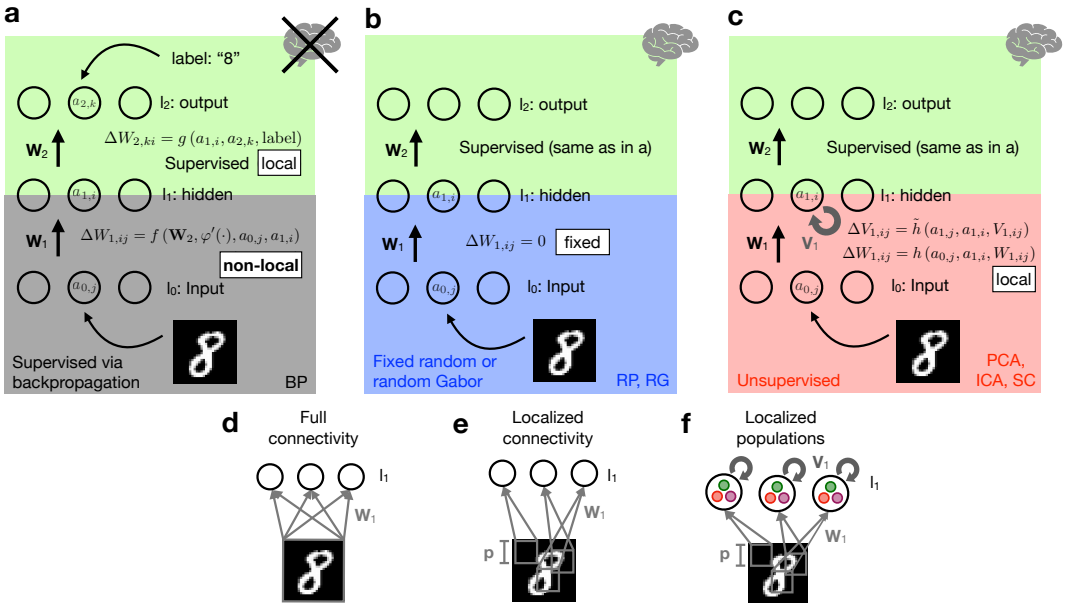


Figure 2.1: The proposed network model has one hidden layer (l_1) and one readout layer (l_2) of nonlinear units (nonlinearity $\varphi(\cdot)$). Respective neural activations (e.g. $a_{0,j}$) and update rules (e.g. $\Delta W_{1,ij}$) are added. ($f(\cdot)$) $g(\cdot), h(\cdot)$ & $\tilde{h}(\cdot)$ are (non-)local plasticity functions, i.e. using only variables (not) available at the synapse for the respective update. **a** Training with backpropagation (BP) through one hidden layer is biologically implausible since it is nonlocal (e.g. using $\mathbf{W}_2$ & $\varphi'(\cdot)$ from higher layers to update $\mathbf{W}_1$, see subsection 2.5.4). **b** & **c** Biologically plausible architecture with fixed Random Projections (RP) or fixed random Gabor filters (RG) (blue box in **b**) or unsupervised feature learning in the first layer (red box in **c**), and a supervised classifier in the readout layer l_2 (green boxes). All weight updates are local. **W** stands for feed-forward, **V** for recurrent, inhibitory weights. (Crossed out) brain icons in a,b & c stand for (non-)bio-plausibility of the whole network. **d** & **e** Illustration of fully connected and localized receptive fields of $\mathbf{W}_1$. **f** For localized Principal/Independent Component Analysis (l-PCA/l-ICA) and Sparse Coding (l-SC) the hidden layer is composed of independent populations. Neurons within each population share the same localized receptive field and compete with each other while the populations are conditionally independent. For more model details, see subsection 2.5.1 - subsection 2.5.4.

ing (Krotov et al., 2019) as in this paper. We refer to Table 2.2 for an extensive list of current biologically plausible models tested on MNIST (see Table 2.1 for abbreviations).

2.3 Results

We study networks that consist of an input (l_0), one hidden (l_1) and an output-layer (l_2) of (nonlinear) units, connected by weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ (Figure 2.1). Training the hidden layer weights $\mathbf{W}_1$ with standard supervised training involves (non-local) error backpropagation using summation over output units, the derivative of the units' nonlinearity ($\varphi'(\cdot)$) and the transposed weight matrix $\mathbf{W}_2^T$ (Figure 2.1a). In the biologically plausible network consid-

Table 2.1: Alphabetical list of abbreviations in this paper.

Abbreviation	Description
AE	Autoencoder
ANN	Artificial Neural Network
BP	(Error-) Backpropagation
CNN / Conv.	Convolutional Neural Network
DBN	Deep Belief Network
DNN	Deep Neural Network
FA	Feedback Alignment
ICA	Independent Component Analysis
$l-\ldots$	localized connectivity between input and hidden layer
LIF	Leaky Integrate-and-Fire
PCA	Principal Component Analysis
RBM	Restricted Boltzmann Machine
RG	Random Gabor filters
RL	Reinforcement Learning
RP	Random Projections
SC	Sparse Coding
SGD	Stochastic Gradient Descent
SNN	Spiking Neural Network
SP	Simple Perceptron
STDP	Spike Timing Dependent Plasticity
SVM	Support Vector Machine

Table 2.2: MNIST benchmarks for biologically plausible models of deep learning compared with models in this paper (**bold**). SNN: Spiking Neural Network, for other abbreviations see section 2.3. Models are ranked by MNIST test accuracy (rightmost column). Parts of this table are taken from (Tavanaei et al., 2018; Kheradpisheh et al., 2018; Diehl and Cook, 2015). Models using convolutional layers (CNN) are marked in orange. See Table 2.1 for abbreviations. For conventional ANN/DNN/CNN MNIST benchmarks see table in LeCun (1998).

Model	Neural coding	Learning type	Comments	Test accuracy (%)
Conv. SNN (Wu et al., 2018)	Spikes	Supervised	5 conv. layers, Spatio-Temporal BP	99.3
Conv. SNN (Diehl et al., 2015)	Rate	Supervised	Conversion: rate → spike	99.1
Conv. Spiking AE(Panda and Roy, 2016)	Spikes	Un/Supervised	Stacked conv. AE with BP + sym. weights	99.1
***l*-RG** (this paper)	Rate	Un/Supervised	Only output layer learned	**98.9**
***l*-BP** (this paper)	Rate	Supervised	BP-benchmark of this paper	**98.8**
***l*-ICA** (this paper)	Rate	Un/Supervised	ICs as features for SGD	**98.8**
***l*-FA** (Bartunov et al., 2018) (& this paper)	Rate	Supervised	FA with localized rec. fields	**98.7**
SNN (Lee et al., 2016)	Spikes	Supervised	BP approx., weight symmetry	98.7
spiking LIF *l*-RG (this paper)	Spikes	Supervised	STDP (only output layer learned)	**98.6**
(Stoch.) Diff. Target Prop. (Lee et al., 2015)	Rate	Supervised	Layer-wise AE, Target Prop.	98.5
Nonlin. Hebb + SGD (Krotov et al., 2019)	Rate	Un/Supervised	nonlin. Hebb + SGD (similar to this paper)	98.5
***l*-RP** (this paper)	Rate	Supervised	Only output layer learned	**98.4**
***l*-SC** (this paper)	Rate	Un/Supervised	SC for 1. layer, SGD for 2. layer	**98.4**
Conv. SNN (Kheradpisheh et al., 2018)	Spikes	Unsupervised	3 Conv. layers, STDP, ext. SVM	98.4
SNN (O'Connor et al., 2017)	Pseudo-spike	Supervised	Sparse, discrete activities, STDP	98.3
Direct FA (Nok)	Rate	Supervised	Many hidden layers	98.3
Spiking FA (Lillicrap et al., 2016)	Spikes	Supervised	3 hidden layers	98.2
spiking LIF *l*-RP (this paper)	Spikes	Supervised	STDP (only output layer learned)	**98.2**
***l*-PCA** (this paper)	Rate	Un/Supervised	PCs as features for SGD	**98.2**
Q-AGREL (RL-like) (Pozzi et al., 2018)	Rate	RL-like	RL-like BP-approx.	98.2
Forward propagation (FP) (Kohan et al., 2018)	Rate	Supervised	FP: BP approximation	98.1
Spiking FA (Neftci et al., 2017)	Spikes	Supervised	Direct FA	98
Predictive coding (Whittington and Bogacz, 2017)	Rate	Supervised	BP approx. by pred. coding	98
Spiking CNN (Tavanaei and Maida, 2016)	Rate/Spikes	Unsupervised	Semi-online, STDP, ext. SVM	98
Equilibrium Prop. (Scellier and Bengio, 2017)	Rate	Supervised	1 - 3 hidden layers	97 - 98
Dendr. BP (Sacramento et al., 2018)	Spikes	Supervised	Dendritic comp. for BP approx.	97.5
Spiking FA (Samadi et al., 2017)	Spikes	Supervised	3 hidden layers	97
Sparse/Skip FA (Baldi et al., 2016)	Rate	Supervised	Sparse- & Skip-FA	96 - 97
Spiking CNN (Thiele et al., 2018)	Spikes	Unsupervised	Recurrent Inhib., STDP	96.6
Spiking FA (Guerguiev et al., 2017)	Spikes	Supervised	Dendritic comp. for BP approx.	96.3
2 layer network (Diehl and Cook, 2015)	Spikes	Unsupervised	Recurrent Inhib., purely unsuperv.	95
Spiking RBM/DBN (O'Connor et al., 2013)	Rate	Supervised	Conversion rate → spike	94.1
2 layer network (Querlioz et al., 2013)	Spikes	Unsupervised	Memristive device	93.5
Spiking HMAX/CNN (Liu and Yue, 2018)	Spikes	Supervised	STDP, HMAX preprocess.	93
Spiking RBM/DBN (Neftci et al., 2014)	Rate	Supervised	Neural sampling	92.6
Spiking RBM/DBN (Neftci et al., 2014)	Spikes	Supervised	Neural sampling	91.9
SP (this paper)	Rate	Supervised	Direct classification on MNIST data	**91.9**
Spiking CNN (Zhao et al., 2015)	Spike	Supervised	Tempotron rule, sensor MNIST	91.3
Dendritic neurons (Hussain et al., 2014)	Rate	Supervised	Nonlin. dendrites, neuromorphic appl.	90.3

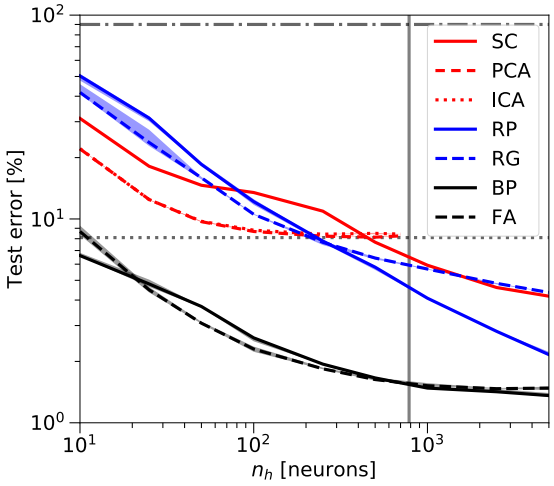


Figure 2.2: MNIST classification with rate networks according to Figure 2.1a-c with full connectivity (Figure 2.1d). The test error decreases for increasing hidden layer size n_h for all methods, i.e. Principal/Independent Component Analysis (PCA/ICA, curves are highly overlapping), Sparse Coding (SC), fixed Random Projections (RP) and fixed random Gabor filters (RG) as well as for the fully supervised reference algorithms Backpropagation (BP) and Feedback Alignment (FA). The dash-dotted line at 90 % is chance level, the dotted line around 8 % is the performance of a Simple Perceptron (SP) without hidden layer. The vertical line marks the input dimension $d = 784$, i.e. the transition from under- to overcomplete hidden representations. Note the log-log scale.

ered in this paper (Figure 2.1b & c), the input-to-hidden weights $\mathbf{W}_1$ are either fixed random, random Gabor filters or learned with an unsupervised method (Principal/ Independent Component Analysis or Sparse Coding). The unsupervised learning algorithms assume recurrent inhibitory weights $\mathbf{V}_1$ between hidden units to implement competition, i.e. to make different hidden units learn different features. For more model details we refer to subsection 2.5.1 - subsection 2.5.4. Code for all (rate & spiking) models discussed below is publicly available at https://github.com/EPFL-LCN/pub-illing2019-nnetworks.

2.3.1 Benchmarking biologically plausible rate models and backpropagation

To see how far we can go with a single hidden layer, we systematically investigate rate models using different methods to initialize or learn the hidden layer weights $\mathbf{W}_1$ (see Figure 2.1 and methods subsection 2.5.1-subsection 2.5.3 for details). We use two different ways to set the weights $\mathbf{W}_1$ of the hidden layer: either using fixed Random Projections (**RP**) or Random Gabor filters (**RG**), see Figure 2.1b & blue curves in Figure 2.2, or using one of the unsupervised methods Principal Component Analysis (**PCA**), Independent Component Analysis (**ICA**) or Sparse Coding (**SC**), see Figure 2.1c & red curves in Figure 2.2. All these methods can be implemented with local, biologically plausible learning rules (Oja, 1982; Hyvärinen and Oja, 1998; Olshausen and Field, 1997). We refer to the methods subsection 2.5.2 for further details. As a reference, we train networks with the same architecture with standard backpropagation (**BP**, see Figure 2.1a).

As a step from BP towards increased biologically plausibility, we include Feedback Alignment (**FA**, Lillicrap et al. (2016)) with fixed random feedback weights for error backpropagation (see methods subsection 2.5.4 for further explanation). A Simple Perceptron (**SP**) without a hidden layer serves as a further reference, since it corresponds to direct classification of the input. We expect any biologically plausible learning algorithm to achieve results somewhere between SP ("lower") and BP ("upper performance bound")
The hidden-to-output weights $\mathbf{W}_2$ are trained with standard stochastic gradient descent (SGD), using a one-hot representation of the class label as target. Since no error backpropagation is needed for a single layer, the learning rule is local ("delta" or "perceptron"-rule). Therefore the two-layer network as a whole is biologically plausible in terms of online learning and synaptic updates using only local variables. For computational efficiency, we first train the hidden layer and then the output layer, however, both layers could be trained simultaneously.
We compare the test errors on the MNIST digit recognition data set for varying numbers of hidden neurons n_h (Figure 2.2). The PCA (red dashed) and ICA (red dotted) curves in Figure 2.2 end at the vertical line $n_h = d = 784$ because the number of principal/independent components (PCs/ICs), i.e. the number of hidden units n_h, is limited by the input dimension d. Since the PCs span the subspace of highest variance, classification performance quickly improves when adding more PCs for small n_h and then saturates for larger n_h. ICA does not seem to discover significantly more useful features than PCA, leading to similar classification performance.
SC (red solid line) extracts sparse representations that can be overcomplete ($n_h > d$), leading to a remarkable classification performance of around 96 % test accuracy. This suggests that the sparse representation and the features extracted by SC are indeed useful for classification, especially in the overcomplete case.
As expected, the performance of RP (blue solid) for small numbers of hidden units ($n_h < d$) is worse than for feature extractors like PCA, ICA or SC. Also for large hidden layers, performance improves only slowly with n_h, which is in line with theory (Barron, 1993) and findings in the extreme learning field (Huang et al., 2006). However, for large hidden layers sizes, RP outperforms SC.
As a reference, we also studied fixing the hidden layer weights to Gabor filters of random orientation, phase and size, located at the image center (RG, blue dashed, see subsection 2.5.3). For hidden layers with more than 1000 neurons, SC is only marginally better than the network with fixed random Gabor filters.

For all tested methods and hidden layer sizes, performance is significantly worse than the one reached with BP (black solid in Figure 2.2). In line with (Lillicrap et al., 2016), we find that FA (black dashed) performs as well as BP on MNIST. Universal function approximation theory predicts lower bounds for the squared error that follow a power law with hidden layer size n_h for both BP ($\mathcal{O}(1/n_h)$) and RP ($\mathcal{O}(1/n_h^{2/d})$, where d is the input dimension (Barron et al., 1994; Barron, 1993)). In the log-log-plot in Figure 2.2 this would correspond to a factor $d/2 = 784/2 = 392$ between the slopes of the curves of BP and RP, or at least a factor $d_{\text{eff}}/2 \approx 10$ using an effective dimensionality of MNIST (see methods 2.5.1). We find a much faster decay

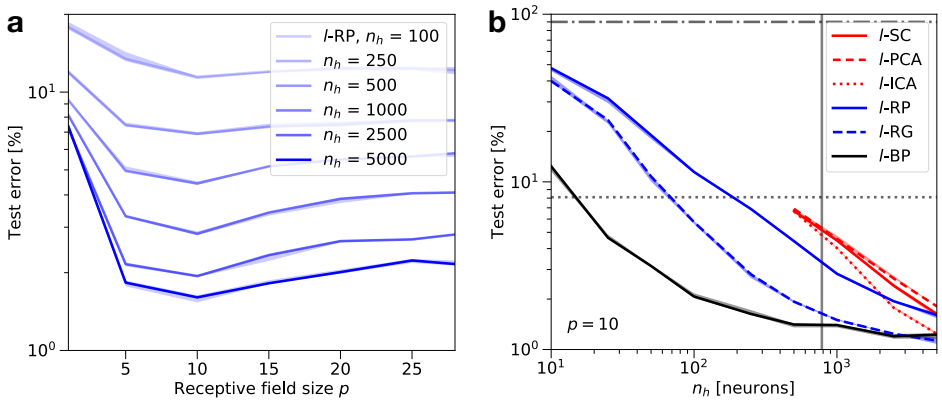


Figure 2.3: Effect of localized connectivity on MNIST. **a** Test error for localized Random Projections (l-RP), dependent on receptive field size p for different hidden layer sizes n_h. The optimum for receptive field size $p = 10$ is more pronounced for large hidden layer sizes. Full connectivity is equivalent to $p = 28$. Note the log-lin scale. **b** Localized receptive fields decrease test errors for all tested networks (compare Figure 2.2): Principal/Independent Component Analysis (l-PCA/l-ICA), Sparse Coding (l-SC), Random Projections (l-RP), Random Gabor filters (l-RG) and Backpropagation (l-BP). The effect is most significant for l-ICA and l-RG, which approach l-BP performance for large n_h and $p = 10$, while all other methods reach test errors between $1-2\%$. All other reference lines as in Figure 2.2. l-PCA/l-ICA & l-SC use 500 independent populations in the hidden layer (see Figure 2.1f) which constrains the hidden layer size to $n_h \geq 500$. Note the log-log scale.

of classification error in RP and a smaller difference between RP and BP slopes than suggested by the theoretical lower bounds.
Taken together, these results show that the high dimensionality of the hidden layers is more important for reaching high performance than the global features extracted by PCA, ICA or SC. Tests on the object recognition task CIFAR10 lead to the same conclusion, indicating that this observation is not entirely task specific (see subsection 2.3.2 for further analysis on CIFAR10).

2.3.2 Localized receptive fields boost performance

There are good reasons to reduce the connectivity from all-to-all to localized receptive fields (Figure 2.1e & f): local connectivity patterns are observed in real neural circuits (Hubel and Wiesel, 1962), useful theoretically (Litwin-Kumar et al., 2017) and empirically (Bartunov et al., 2018), and successfully used in convolutional neural networks (CNNs). Even though this modification seems well justified from both biological and algorithmic sides, it reduces the generality of the algorithm to input data such as images where neighborhood relations between pixels (i.e. input dimensions) are important.
To obtain localized receptive fields (called "l-" methods in the following) patches spanning $p \times p$ pixels in the input space are assigned to the hidden neurons. The centers of the patches are chosen at random positions in the input space, see Figure 2.1e & f. For localized Random Projections (l-RP) and localized random Gabor filters (l-RG) the weights within the patches

are randomly drawn from the respective distribution and then fixed. For the localized unsupervised learning methods (l-PCA, l-ICA & l-SC) the hidden layer is split into 500 independent populations. Neurons within each population compete with each other while different populations are independent, see Figure 2.1f. This split implies a minimum number of $n_h = 500$ hidden neurons for these methods. For l-PCA and l-ICA a thresholding nonlinearity was added to the hidden layer to leverage the local structure (otherwise PCA/ICA act globally due to their linear nature, see methods subsection 2.5.2).

We test l-RP for different patch sizes p and find an optimum around $p \approx 10$ (see Figure 2.3a). Note that $p = 1$ corresponds to resampling the data with random weights, and $p = 28$ recovers fully connected RP performance. The other methods show similar optimal values around $p = 10$ (not shown). The main finding here is the significant improvement in performance using localized receptive fields. All tested methods improve by a large margin when switching from full image to localized patches and some methods (l-RG and l-ICA) even reach BP performance for $n_h = 5000$ hidden neurons (see Figure 2.3b). To achieve a fair comparison BP is also implemented with localized receptive fields (l-BP) which leads to a minor improvement compared to global BP. This makes local random projections or local unsupervised learning strong competitors to BP as biologically plausible algorithms in the regime of large, overcomplete hidden layers $n_h > d$ - at least for MNIST classification.

To test whether localized receptive fields only work for the relatively simple MNIST data set (centered digits, uninformative margin pixels, no clutter, uniform features and perspective etc.) or generalizes to more difficult tasks, we apply it to the CIFAR10 data set (Krizhevsky, 2013). We first reproduce a typical benchmark performance of a fully connected network with one hidden layer trained with standard BP ($\approx 56\%$ test accuracy, $n_h = 5000$, see also Lin and Memisevic (2016)). Again, classification performance increases for increasing hidden layer size n_h and localized receptive fields perform better than full connectivity for all methods. Furthermore, as on MNIST, we can see similar performances for local feature learning methods (l-PCA, l-ICA & l-SC) and local random features (l-RP, l-RG) in the case of large, overcomplete hidden layers (see Table 2.3). Also on CIFAR10, localized random filters and local feature learning reach the performance of biologically plausible models of deep learning (Bartunov et al., 2018; Krotov et al., 2019) and come close to the performance of the reference algorithm l-BP. However, the difference remains statistically significant here. Given that the state-of-the-art performance on CIFAR10 with deep convolutional neural networks is close to 98% (e.g. Real et al. (2018)), the limitations of our shallow local network and the well-known differences in difficulty between MNIST and CIFAR10 become apparent.

In summary, the main message of this section is that unsupervised methods, as well as random features, perform significantly better when applied locally. Equipped with local receptive fields our shallow network can outperform many current models of biologically plausible deep learning (see Table 2.2). On MNIST some models (l-RG & l-ICA) even reach backpropagation performance, while on CIFAR10 large differences to state-of-the-art deep convolutional networks remain.

Table 2.3: Test accuracies (%) on MNIST and CIFAR10 for rate networks and spiking LIF models. The Simple Perceptron (SP) is equivalent to direct classification on the data without hidden layer. All other methods use $n_h = 5000$ hidden neurons and receptive field size $p = 10$. Note that CIFAR10 has $d = 32\times32\times3 = 3072$ input channels (the third factor is due to the color channels), MNIST only $d = 28\times28 = 784$. The rate (spiking) models are trained for 167 (117) epochs. Best performing in bold.

		SP	l-PCA	l-ICA	l-SC	l-RP	l-RG	l-BP
Rate	CIFAR10	41.1 ± 0.1	50.8 ± 0.3	53.9 ± 0.3	50.2 ± 0.2	52.0 ± 0.4	55.6 ± 0.2	**58.3 ± 0.2**
	MNIST	91.9 ± 0.1	98.2 ± 0.02	**98.8 ± 0.03**	98.4 ± 0.07	98.4 ± 0.1	**98.9 ± 0.05**	**98.8 ± 0.1**
Spiking	MNIST			-		98.2 ± 0.05	98.6 ± 0.1	-

2.3.3 Spiking localized random projections

Real neural circuits communicate with short electrical pulses, called spikes, instead of real numbers such as rates. We thus extend our shallow network model to networks of leaky integrate-and-fire (**LIF**) neurons. The network architecture is the same as in Figure 2.1b. To keep it simple we implement the two models with fixed random weights with LIF neurons: fixed localized Random Projections (l-**RP**) and fixed localized random Gabor filters (l-**RG**) with patches of size $p \times p$ - as in subsection 2.3.2. The output layer weights $\mathbf{W}_2$ are trained with a supervised spike timing dependent plasticity (**STDP**) rule.

The spiking dynamics follow the usual LIF equations (see methods subsection 2.5.5) and the readout weights $\mathbf{W}_2$ evolve according to a supervised delta rule via spike timing dependent plasticity (STDP) using post-synaptic spike-traces $\mathrm{tr}_i(t)$ and a post-synaptic target trace $\mathrm{tgt}_i(t)$

$$\begin{aligned} \tau_{\mathrm{tr}} \frac{d\mathrm{tr}_i(t)}{dt} &= -\mathrm{tr}_i(t) + \sum_f \delta\left(t - t_i^f\right) \\ \Delta w_{2,ij} &= \alpha \cdot \left(\mathrm{tgt}_i^{\mathrm{post}}(t) - \mathrm{tr}_i^{\mathrm{post}}(t)\right) \delta\left(t - t_j^f\right), \end{aligned} \tag{2.1}$$

where α is the learning rate. Thus, for a specific readout weight $w_{2,ij}$, the post-synaptic trace is updated at every post-synaptic spike time t_i^f and the weight is updated at every pre-synaptic spike time t_j^f. The target trace is constant while a pattern is presented and uses a standard one-hot coding for the supervisor signal in the output layer (l_2).
To illustrate the LIF and STDP dynamics, a toy example consisting of one pre- connected to one post-synaptic neuron is integrated for 650 ms. The pre- and post-synaptic membrane potentials show periodic spiking (Figure 2.4a) which induces post-synaptic spike traces and corresponding weight changes (Figure 2.4b), according to Equation 2.1. For the MNIST task, Figure 2.4c shows a raster plot for an exemplary training and testing protocol. During activity transients after a switch from one pattern to the next, learning is disabled until regular spiking is recovered. We experienced that without disabling learning during these transient phases the networks never reached a low test error. This is not surprising, since in this phase the network activities carry information both about the previously presented pattern and the current one, but the learning rule is designed for network activities in response to a single input pattern. It is also known that LIF neurons differ from biological neurons in response to step currents

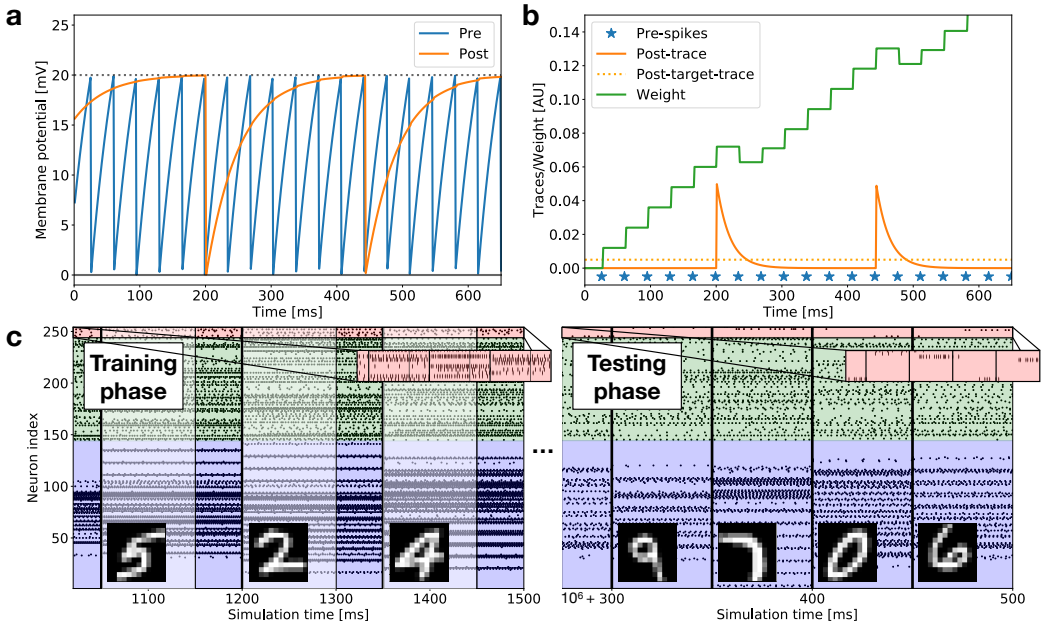


Figure 2.4: Spiking LIF and STDP dynamics. **a** Dynamics of the pre- and postsynaptic membrane potentials, spike-traces and the weight value (**b**) of a toy example with two neurons and one interconnecting synapse. The weight decreases when the post-trace is above the post-target-trace (see Equation 2.1 and subsection 2.5.5). Both neurons receive static suprathreshold external input: $I^{\text{ext}}_{\text{pre}} \gg I^{\text{ext}}_{\text{post}} \approx \vartheta$ (spiking threshold). Note that presynaptic spikes only slightly alter the postsynaptic potential since the weight is initially zero. **c** Rasterplot of a network trained on MNIST, where every spike is marked with a dot. The background color indicates the corresponding layers: input (blue, $n_0 = 144$ neurons), hidden (green, $n_1 = n_h = 100$) and output (red, $n_2 = 10$). Bold vertical lines indicate pattern switches, thin lines indicate ends of transient phases (indicated by semi-transparency), during which learning is disabled. Left: Behaviour at the beginning of the training phase. Right: Testing period (learning off) after $6 \cdot 10^4$ presented patterns (1 epoch). As can be seen in the zoomed view of the 10 output layer neurons (red), the output layer has started to learn useful, 1-hot encoded class predictions. A downsampled (12×12) version of MNIST is used for improved visibility.

(see Naud et al. (2008) and references therein). During the testing period, learning is shut off permanently (see methods section 2.5.5 for more details). The LIF and STDP dynamics can be mapped to a rate model (see e.g. Diehl et al. (2015) and subsection 2.5.5 for details). However all following results are obtained with the fully spiking LIF/STDP model.

When directly trained with the STDP rule of Equation 2.1, the spiking LIF models closely approach the performance of their rate counterparts. Table 2.3 compares the performances of the rate and spiking LIF l-RP & l-RG models with the reference algorithm l-BP (for same hidden layer size n_h and patch size p, see subsection 2.3.2). The remaining gap ($< 0.3\%$) between rate model and spiking LIF model presumably stems from noise introduced by the spiking approximation of rates and the activity transients mentioned above. Both, the rate and spiking LIF model of l-RP/l-RG achieve accuracies close to the backpropagation reference

algorithm l-BP and fall in the range of performance of prominent, biologically plausible models, i.e. 98-99% test accuracy (see section 2.2 and Table 2.2). Based on these numbers we conclude that the spiking LIF model of localized random projections using STDP is capable of learning the MNIST task to a level that is competitive with known benchmarks for spiking networks.

2.4 Discussion

In contrast to biologically plausible deep learning algorithms that are derived from approximations of the backpropagation algorithm (Whittington and Bogacz, 2019; Lillicrap et al., 2016; Sacramento et al., 2018; Pozzi et al., 2018), we focus here on shallow networks with only one hidden layer. The weights from the input to the hidden layer are either learned by unsupervised algorithms with local learning rules; or they are fixed. If fixed, they are drawn randomly or represent random Gabor filters. The readout layer is trained with a supervised, local learning rule.

When applied globally, randomly initialized fixed weights/ Gabor filters (RP/RG) of large hidden layers lead to better classification performance than training them with unsupervised methods like Principal/Independent Component Analysis (PCA/ICA) or Sparse Coding (SC). Such observations also occur in different contexts, e.g. Dasgupta et al. (2018) showed that (sparse) random projections, combined with dimensionality expansion outperform known algorithms for locality-sensitive hashing. It may be interesting to search for alternative unsupervised, local learning rules with an inductive bias that is better adapted to image processing tasks than the one of SC.

Replacing all-to-all connectivity with localized input filters is such an inductive bias that already proved useful in supervised models (Bartunov et al., 2018) but turns out to be particularly powerful in conjunction with unsupervised learning (l-PCA, l-ICA & l-SC). Interestingly, non of the local unsupervised methods could significantly outperform localized random Gabor filters (l-RG). Furthermore, we find that the performance scaling with the number of hidden units n_h is orders of magnitudes better than the lower bound suggested by universal function approximation theory (Barron, 1993).

To move closer to realistic neural circuits we implement our shallow, biologically plausible network with spiking neurons and spike timing dependent plasticity to train the readout layer. Spiking localized random projections (l-RP) and localized Gabor filters (l-RG) reach >98% test accuracy on MNIST which lies within the range of current benchmarks for biologically plausible models for deep learning (see section 2.2 and Table 2.2). Our network model is particularly simple, i.e. it has only one trainable layer and does not depend on sophisticated architectural or algorithmic features typically necessary to approximate backpropagation (Whittington and Bogacz, 2019). Instead it only relies on the properties of high-dimensional localized random projections.

Since we want to keep our models as simple as possible, we use online stochastic gradient descent (SGD, no mini-batches) with a constant learning rate. There are many known ways

to further tweak the final performance, e.g. with adaptive learning rate schedules or data augmentation, but our goal here is to demonstrate that even a simple model with constant learning rate achieves results that are comparable with more elaborate approaches that use e.g. convolutional layers with weight sharing (Panda and Roy, 2016), backpropagation approximations (Lee et al., 2016), multiple hidden layers (Lillicrap et al., 2016), dendritic neurons (Sacramento et al., 2018), recurrence (Diehl and Cook, 2015) or conversion from rate to spikes (Diehl et al., 2015).

Above 98% accuracy we also have to take into account a saturating effect of the network training: better models will only lead to subtle improvements in accuracy. It is not obvious whether improvements are really a proof of having achieved deep learning or just the result of tweaking the models towards the peculiarities of the MNIST dataset. Localized random filters or local unsupervised feature learning perform remarkably well compared to fully-connected backpropagation in shallow networks, even on more challenging data sets such as CIFAR10. This makes our model an important benchmark for future, biologically plausible models but also clearly highlights the limitations of our shallow two-layer model. A long time ago state-of-the-art deep learning has moved from MNIST to harder datasets, such as CIFAR10 or ImageNet (Deng et al., 2009). Yet MNIST seems to be the current reference task for most biologically plausible deep learning models (see section 2.2 and Table 2.2). We suggest that novel, progressive approaches to biologically plausible deep learning should significantly outperform the results presented here. Furthermore, they should be tested on tasks other than MNIST, where real deep learning capabilities become necessary.

2.5 Supplemental information

2.5.1 General rate model details

We use a 3-layer (input l_0, hidden $l_1 = l_h$ and output l_2) feed-forward rate-based architecture with layer sizes (n_0 for input), n_1 (hidden) and n_2 (output, with $n_2 = 10 =$ number of classes). The layers are connected via weight matrices $\mathbf{W}_1 \in \mathbb{R}^{n_1 \times n_0}$ and $\mathbf{W}_2 \in \mathbb{R}^{n_2 \times n_1}$ and each neuron receives bias from the bias vectors $\mathbf{b}_1 \in \mathbb{R}^{n_1}$ and $\mathbf{b}_2 \in \mathbb{R}^{n_2}$ respectively (see Figure 2.1). The neurons themselves are nonlinear units with an element-wise, possibly layer-specific, nonlinearity $\mathbf{a}_i = \varphi_l(\mathbf{u}_i)$. The feed-forward pass of this model thus reads

$$\begin{aligned} \mathbf{u}_{l+1} &= \mathbf{W}_{l+1}\mathbf{a}_l + \mathbf{b}_{l+1} \\ \mathbf{a}_{l+1} &= \varphi_{l+1}(\mathbf{u}_{l+1}). \end{aligned} \tag{2.2}$$

The Simple Perceptron (SP) only consists of one layer (l_2, $\mathbf{W}_2 \in \mathbb{R}^{n_2 \times n_0}$, $\mathbf{b}_2 \in \mathbb{R}^{n_2}$). The sparse coding (SC) model assumes recurrent inhibition within the hidden layer l_1. This inhibition is not modeled by an explicit inhibitory population, as required by Dale's principle (Dale, 1935), but direct, plastic, inhibitory synapses $\mathbf{V}_1 \in \mathbb{R}^{n_1 \times n_1}$ are assumed between neurons in

l_1. Classification error variances in Figure 2.2 & Figure 2.3 are displayed as shaded, semi-transparent areas with the same colors as the corresponding curves. Their lower and upper bounds correspond to the 25% and 75% percentiles of at least 10 independent runs.
An effective dimensionality d_{eff} of the MNIST data set can be obtained, e.g. via eigen-spectrum analysis, keeping 90% of the variance. We obtain values around $d_{\text{eff}} \approx 20$. The measure proposed by Litwin-Kumar et al. (2017) gives the same value $d_{\text{eff}} \approx 20$. We checked that training a perceptron (1 hidden layer, $n_h = 1000$, 10^7 iterations, ReLU, standard BP) on the first 25 PCs of MNIST instead of the full data set leads to a comparable MNIST performance (1.7% vs 1.5% test error respectively). Together, these findings suggest that the MNIST dataset lies mostly in a low-dimensional linear subspace with $d_{\text{eff}} \approx 25 \ll d$. The MNIST (& CIFAR10) data was rescaled to values in [0,1] and mean centered, which means that the pixel-wise average over the data was subtracted from the pixel values of every image. Simulations were implemented and performed in the `Julia`-language. The code for the implementation of our rate network models is publicly available at https://github.com/EPFL-LCN/pub-illing2019-nnetworks.

2.5.2 Unsupervised methods (PCA, ICA & SC)

In this paper we do not implement PCA/ICA learning explicitly as a neural learning algorithm but by a standard PCA/ICA algorithm (MultivariateStats.jl) since biologically plausible online algorithms for both methods are well known (Sanger, 1989; Hyvärinen and Oja, 1998). For d-dimensional data such algorithms output the values of the $n \leq d$ first principal/ independent components as well as the corresponding subspace projection matrix $\mathbf{P} \in \mathbb{R}^{n \times d}$. This matrix can directly be used as feedforward matrix $\mathbf{W}_1$ in our network since the lines of $\mathbf{P}$ correspond to the projections of the data onto the single/independent principal components. In other words each neuron in the hidden layer l_1 extracts another principal/independent component of the data. ICA was performed with the usual pre-whitening of the data.
Since PCA/ICA is a linear model, biases $\mathbf{b}_1$ were set to $\mathbf{0}$ and $\varphi_1(\mathbf{u}) = \mathbf{u}$. With this, we can write the (trained) feed-forward pass of the first layer of our PCA/ICA model as follows:

$$\mathbf{a}_1 = \mathbf{u}_1 = \mathbf{W}_1 \cdot \mathbf{a}_0 \quad \text{with } \mathbf{W}_1 = \mathbf{P} \tag{2.3}$$

Since the maximum number of principal/independent components that can be extracted is the dimensionality of the data, $n_{\max} = d$, the number of neurons in the hidden layer n_1 is limited by d. This makes PCA/ICA unusable for overcomplete hidden representations as investigated for SC and RP. In the localized version of PCA/ICA we assume the hidden layer to consist of independent populations, each extracting PCs/ICs of its respective localized receptive field (see Figure 2.1). The hidden layer was divided into 500 of those populations, resulting in a minimum number of $n_h = 500$ hidden neurons (1 PC/IC per population) for these methods (and up to 10 PCs/ICs per population for $n_h = 5000$). The classifier was then trained on the combined activations of all populations of the hidden layer. Because PCA/ICA are linear methods the localized PCA/ICA version would not extract significantly different

features unless we introduce a nonlinearity in the hidden units. This was done by simply thresholding the hidden activations (ReLU with threshold 0). No further optimization in terms of nonlinearity- and threshold-tuning was performed.

Sparse coding (SC) aims at finding a feature dictionary $\mathbf{W} \in \mathbb{R}^{h \times d}$ (for d-dimensional data) that leads to an optimal representation $\mathbf{a}_1 \in \mathbb{R}^h$ which is sparse, i.e. has as few non-zero elements as possible. The corresponding optimization problem reads:

$$\begin{aligned} \mathbf{W}^{opt}, \mathbf{a}_1^{opt} &= \operatorname{argmin} \mathscr{L}(\mathbf{W}, \mathbf{a}_1) \\ \mathscr{L}(\mathbf{W}, \mathbf{a}_1) &= \frac{1}{2}\|\mathbf{a}_0 - \mathbf{W}^\top \mathbf{a}_1\|_2^2 + \lambda \|\mathbf{a}_1\|_1. \end{aligned} \tag{2.4}$$

Since this is a nonlinear optimization problem with latent variables (hidden layer) it cannot be solved directly. Usually an iterative two step procedure is applied (akin to the expectation-maximization algorithm) until convergence: First optimize with respect to the activities $\mathbf{a}$ with fixed weights $\mathbf{W}$. Second, assuming fixed activities, perform a gradient step w.r.t to weights. We implement a biologically plausible SC model using a 2-layer network with recurrent inhibition and local plasticity rules similar to the one in Brito and Gerstner (2016). For a rigorous motivation (and derivation) that such a network architecture can indeed implement sparse coding we refer to Olshausen and Field (1997); Zylberberg et al. (2011); Pehlevan and Chklovskii (2015); Brito and Gerstner (2016). We apply the above mentioned two step optimization procedure to solve the SC problem given our network model. The following two steps are repeated in alternation until convergence of the weights:

1. **Optimizing the hidden activations:**
   We assume given and fixed weights $\mathbf{W}_1$ and $\mathbf{V}_1$ and ask for optimal hidden activations $\mathbf{a}_1$. Because of the recurrent inhibition $\mathbf{V}_1$ the resulting equation for the hidden activities $\mathbf{a}_1$ is nonlinear and implicit. To solve this equation iteratively, we simulate the dynamics of a neural model with time-dependent internal and external variables $\mathbf{u}_1(t)$ and $\mathbf{a}_1(t)$ respectively. The dynamics of the system is then given by Zylberberg et al. (2011); Brito and Gerstner (2016):

   $$\begin{aligned} \tau_u \frac{d\mathbf{u}_1(t)}{dt} &= -\mathbf{u}_1(t) + (\mathbf{W}_1 \mathbf{a}_0(t) - \mathbf{V}_1 \mathbf{a}_1(t)) \\ \mathbf{a}_1(t) &= \varphi(\mathbf{u}_1(t)) \end{aligned} \tag{2.5}$$

   In practice the dynamics is simulated for $N_{\text{iter}} = 50$ iterations, which leads to satisfying convergence (change in hidden activations $< 5\%$).

2. **Optimizing the weights:**
   Now the activities $\mathbf{a}_1$ are kept fixed and we update the weights following the gradient of the loss function. The weight update rules are Hebbian-type local learning rules (Brito

and Gerstner, 2016):

$$\begin{aligned} \Delta W_{1,ji} &= \alpha_w \cdot a_{0,i} \cdot a_{1,j} \\ \Delta V_{1,jk} &= \alpha_v \cdot a_{1,k} \cdot \left(a_{1,j} - \langle a_{1,j} \rangle\right) \end{aligned} \tag{2.6}$$

$\langle \cdot \rangle$ is a moving average (low-pass filter) over several past hidden representations (after convergence of the recurrent dynamics) with some time constant τ_{mav}, e.g. τ_{mav} = 100 patterns. At the beginning of the simulation (or after a new pattern presentation) τ_{mav} is increased starting from 0 to τ_{mav} during the first τ_{mav}. The values of the rows of $\mathbf{W}_1$ are normalized after each update, however this can also be achieved by adding a weight decay term. Additionally the values of $\mathbf{V}_1$ are clamped to positive values after each update to ensure that the recurrent input is inhibitory. Also the diagonal of $\mathbf{V}_1$ is kept at zero to avoid self-inhibition.

During SC learning, at every iteration, the variables $\mathbf{u}_1(t)$ and $\mathbf{a}_1(t)$ are reset (to avoid transients) before an input is presented. Then for every of the N iterations, equation 2.5 is iterated for N_{iter} steps and the weights are updated according to equation 2.6.
Similar to localized PCA/ICA, the localized version of SC uses independent populations in the hidden layer (see Figure 2.1). The SC algorithm above was applied to each population and its respective receptive field independently. The classifier was then trained on the combined activations of all populations of the hidden layer.

2.5.3 Fixed Random Filters (RP & RG)

For RP, the weight matrix $\mathbf{W}_1$ between input and hidden layer is initialized randomly $\mathbf{W}_1 \sim \mathcal{N}(0, \sigma^2)$ with variance-preserving scaling: $\sigma^2 \propto 1/n_0$. The biases $\mathbf{b}_1$ are initialized by sampling from a uniform distribution $\mathcal{U}([0, 0.1])$ between 0 and 0.1. In practice we used the specific initialization

$$\begin{aligned} \mathbf{W}_1 &\sim \mathcal{N}(0, \sigma^2) \;\; \sigma^2 = \frac{1}{100\, n_0} \\ \mathbf{b}_1 &\sim \mathcal{U}([0, 0.1]) \end{aligned} \tag{2.7}$$

for RP (keeping weights fixed), SC, SP and also BP & RF (both layers with $\mathbf{W}_2, \mathbf{b}_2$ and n_1 respectively).
For localized RP (l-RP), neurons in the hidden layer receive input only from a fraction of the input units called a receptive field. Receptive fields are chosen to form a compact patch over neighbouring pixels in the image space. For each hidden neuron a receptive field of size $p \times p$ ($p \in \mathbb{N}$) input neurons is created at a random position in the input space. The weight values

for each receptive field (rf) and the biases are initialized as:

$$\mathbf{W}_{1,\mathrm{rf}} \sim \mathcal{N}(0,\sigma^2_{\mathrm{rf}}) \quad \sigma^2_{\mathrm{rf}} = \frac{c}{100\ p} \tag{2.8}$$

$$\mathbf{b}_1 \sim \mathcal{U}([0,0.1]) \tag{2.9}$$

were the parameter $c = 3$ was found empirically through a grid-search optimization of classification performance. For exact parameter values, see Table 2.4.
The (localized) random Gabor filters in RG have the same receptive field structure as in l-RP (see subsection 2.5.3) but instead of choosing the weights within the receptive field as random values, they are choosen according to Gabor filters $\mathbf{W}_1 \propto g(x,y)$. Here, x and y denote the pixel coordinates within the localized receptive field relative to the patch center. The Gabor filters have the following functional form:

$$\begin{aligned} g\left(x,y;\lambda,\Theta,\psi,\sigma,\gamma\right) &= \\ \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right) &\cdot \cos\left(2\pi\frac{x'}{\lambda}+\psi\right) \\ \begin{pmatrix}x'\\y'\end{pmatrix} &= \begin{pmatrix}\cos\Theta & \sin\Theta\\ -\sin\Theta & \cos\Theta\end{pmatrix}\cdot\begin{pmatrix}x\\y\end{pmatrix} \end{aligned} \tag{2.10}$$

To obtain diverse, random receptive fields we draw the parameters $\lambda,\Theta,\psi,\sigma,\gamma$ of the Gabor functions from uniform distributions over some intervals. The bounds of the sampling interval are optimized using Bayesian optimization (BayesianOptimization.jl) with respect to classification accuracy on the training set.

2.5.4 Classifier & Supervised reference algorithms (BP, FA & SP)

The connections $\mathbf{W}_2$ from hidden to output layer are updated by a simple delta-rule which is equivalent to BP in a single-layer network and hence is biologically plausible. For having a reference for our biologically plausible models (Figure 2.1b & c), we compare it to networks with the same architecture (number of layers, neurons, connectivity) but trained in a fully supervised way with standard backpropagation (Figure 2.1a). The forward pass of the model reads:

$$\begin{aligned} \mathbf{u}_{l+1} &= \mathbf{W}_{l+1}\mathbf{a}_l + \mathbf{b}_{l+1} \\ \mathbf{a}_{l+1} &= \varphi_{l+1}(\mathbf{u}_{l+1}) \end{aligned} \tag{2.11}$$

Given the one-hot encoded target activations **tgt**, the error $\tilde{\mathbf{e}}_L$ is

$$\tilde{\mathbf{e}}_L = \mathbf{tgt} - \mathbf{a}_L \tag{2.12}$$

when minimizing mean squared error (MSE)

$$\mathscr{L}_{\mathrm{MSE}} = \frac{1}{2}\|\mathbf{tgt}-\mathbf{a}_L\|_2^2 \tag{2.13}$$

or

$$\begin{aligned}\mathbf{p} &= \mathrm{softmax}(\mathbf{a}_L)\\ \tilde{\mathbf{e}}_L &= \mathbf{tgt}-\mathbf{p}\end{aligned} \tag{2.14}$$

for the softmax/cross-entropy loss (CE),

$$\mathscr{L}_{\mathrm{CE}} = -\sum_{i=1}^{n_L}\mathrm{tgt}_i\cdot\log\left(p_i\right).$$

Classification results (on the test set) for MSE- and CE-loss were found to be not significantly different. Rectified linear units (ReLU) were used as nonlinearity $\varphi(\mathbf{u}_l)$ for all layers (MSE-loss) or for the first layer only (CE-loss).
In BP the weight and bias update is obtained by stochastic gradient descent, i.e. $\Delta W_{l,ij}\propto\frac{\partial\mathscr{L}}{\partial W_{l,ij}}$. The full BP algorithm for deep networks reads (Rumelhart et al., 1986):

$$\begin{aligned}\mathbf{e}_L &= \varphi'_L(\mathbf{u}_L)\odot\tilde{\mathbf{e}}_L\\ \mathbf{e}_{l-1} &= \varphi'_{l-1}(\mathbf{u}_l)\odot\mathbf{W}_l^\top\mathbf{e}_l\\ \Delta\mathbf{W}_l &= \alpha\cdot\mathbf{e}_l\otimes\mathbf{a}_{l-1}\\ \Delta\mathbf{b}_l &= \alpha\cdot\mathbf{e}_l\end{aligned} \tag{2.15}$$

where $\odot$ stands for element-wise multiplication, $\otimes$ is the outer (dyadic) product, $\varphi'_l(\cdot)$ is the derivative of the nonlinearity and α is the learning rate. FA (Lillicrap et al., 2016) uses a fixed random matrix $\mathbf{R}_l$ instead of the transpose of the weight matrix $\mathbf{W}_l^\top$ for the error backpropagation step in equation 2.15.

To allow for a fair comparison with l-RP, BP and FA were implemented with full connectivity and with localized receptive fields with the same initialization as in l-RP. During training with BP (or FA), the usual weight update equation 2.15 was applied to the weights within the receptive fields. The exact parameter values can be found in Table 2.4.

2.5.5 Spiking implementation of RP & RG

The spiking simulations were performed with a custom-made event-based leaky integrate-and-fire (LIF) integrator written in the `Julia`-language. Code is available at https://github.com/EPFL-LCN/pub-illing2019-nnetworks. For large network sizes, the exact, event-based integration can be inefficient due to a large frequency of events. We thus also added an Euler-forward integration mode to the framework. For sufficiently small time

discretization (e.g. $\Delta t \leq 5 \cdot 10^{-2}$ ms for the parameters given in Table 2.6) the error of Euler-forward integration does not have negative consequences on the learning outcome. The dynamics of the LIF network is given by:

$$\begin{aligned} \tau_m \frac{du_i(t)}{dt} &= -u_i(t) + RI_i(t) \\ \text{with } I_i(t) &= I_i^{ff}(t) + I_i^{ext}(t) \\ &= \sum_{j,f} w_{ij} \epsilon\left(t - t_j^f\right) + I_i^{ext}(t) \end{aligned} \tag{2.16}$$

and the spiking condition: $u_i(t) \geq \vartheta_i$: $u_i \rightarrow u_{\text{reset}}$, where $u_i(t)$ is the membrane potential, τ_m the membrane time-constant, R the membrane resistance, w_{ij} are the synaptic weights, $\epsilon(t) = \delta(t)/\tau_m$ is the post-synaptic potential evoked by a pre-synaptic spike arrival, ϑ_i is the spiking threshold and u_{reset} the reset potential after a spike.
The input is split into a feed-forward ($I^{ff}(t)$) and an external ($I^{ext}(t)$) contribution. Each neuron in the input layer l_0 ($n_0 = d$) receives only external input I^{ext} proportional to one pixel value in the data. To avoid synchrony between the spikes of different neurons, the starting potentials and parameters (e.g. thresholds) for the different neurons are drawn from a (small) range around the respective mean values.
We implement STDP using post-synaptic spike-traces $\text{tr}_i(t)$ and a post-synaptic target-trace $\text{tgt}_i(t)$.

$$\begin{aligned} \tau_{\text{tr}} \frac{d\text{tr}_i(t)}{dt} &= -\text{tr}_i(t) + \sum_f \delta\left(t - t_i^f\right) \\ \Delta w_{ij} &= g\left(\text{tr}_i^{\text{post}}(t), \text{tgt}_i^{\text{post}}(t)\right) \delta\left(t - t_j^f\right) \end{aligned} \tag{2.17}$$

with the plasticity function

$$g\left(\text{tr}_i^{\text{post}}(t), \text{tgt}_i(t)\right) = \alpha \cdot \left(\text{tgt}_i^{\text{post}}(t) - \text{tr}_i^{\text{post}}(t)\right). \tag{2.18}$$

To train the network, we present patterns to the input layer and a target-trace to the output layer. The MNIST input is scaled by the input amplitude amp_{inp}, the targets $\mathbf{tgt}(t)$ of the output layer are the one-hot-coded classes, scaled by the target amplitude amp_{tgt}. Additionally, every neuron receives a static bias input $I_{\text{bias}}^{\text{ext}} \approx \vartheta$ to avoid silent units in the hidden layer. Every pattern is presented as fixed input for a time T_{pat} and the LIF dynamics as well as the learning evolves according to equation 2.16 and equation 2.17 respectively. Learning is disabled after pattern switches for a duration of $T_{\text{trans}} = 4\tau_m$ since the noise introduced by these transient phases was found to deteriorate learning progress. With the parameters we used for the simulations (see Table 2.6), firing rates of single neurons in the whole network stayed below 1 kHz which was considered as a biologically plausible regime. For the toy example in Figure 2.4a& b we used static input and target with the parameters $\text{amp}_{\text{inp}} = 40$, $\text{amp}_{\text{tgt}} = 5$ (i.e. target trace = 0.005), $\vartheta_{\text{mean}} = 20$, $\sigma_\vartheta = 0$, $\tau_m = 50$, $\alpha = 1.2 \cdot 10^{-5}$. For the raster plot in Figure 2.4c we used $\text{amp}_{\text{inp}} = 300$, $\text{amp}_{\text{tgt}} = 300$, $\vartheta_{\text{mean}} = 20$, $\sigma_\vartheta = 0$, $\tau_m = 50$, $\alpha = 1.2 \cdot 10^{-5}$,

$T_{\text{pat}} = 50$ ms, $T_{\text{trans}} = 100$ ms. The LIF dynamics can be mapped to a rate model described by the following equations:

$$\begin{aligned} \mathbf{u}_l &= \mathbf{W}_l \mathbf{u}_{l-1} + R\mathbf{I}^{ext} \\ \mathbf{a}_l &= \varphi_{\text{LIF}}(\mathbf{u}_l) \\ \Delta w_{ij} &= \tilde{g}\left(a_j^{\text{pre}}, a_i^{\text{post}}, \text{tgt}_i^{\text{post}}\right) \end{aligned} \tag{2.19}$$

with the (element-wise) LIF-activation function $\varphi_{\text{LIF}}(\cdot)$ and the modified plasticity function $\tilde{g}(\cdot)$:

$$\begin{aligned} \varphi_{\text{LIF}}(u_k) &= \left[\Delta_{\text{abs}} - \tau_m \ln\left(1 - \frac{\vartheta_k}{u_k}\right)\right]^{-1} \\ \tilde{g}\left(a_j^{\text{pre}}, a_i^{\text{post}}, \text{tgt}_i^{\text{post}}\right) &= \tilde{\alpha} \cdot a_j^{\text{pre}} \cdot \left(\text{tgt}_i^{\text{post}} - a_i^{\text{post}}\right) \end{aligned}$$

The latter can be obtained by integrating the STDP rule of Equation 2.17 and taking the expectation over spike times. Most of the parameters of the spiking- and the LIF rate models can be mapped to each other directly (see Table 2.6). The learningrate α must be adapted since the LIF weight change depends on the presentation time of a pattern T_{pat}. In the limit of long pattern presentation times ($T_{\text{pat}} \gg \tau_m, \tau_{\text{tr}}$), the theoretical transition from the learning rate of the LIF rate model ($\tilde{\alpha}$) to the one of the spiking LIF model (α) is $\alpha = \frac{1000\text{ ms}}{T_{\text{pat}}\text{ [ms]}} \cdot 1000 \cdot \tilde{\alpha}$, where the second factor comes from a unit change from Hz to kHz. It is also possible to train weight matrices computationally efficient in the LIF rate model and plug them into the spiking LIF model afterwards. The reasons for the remaining difference in performance presumably lie in transients and single-spike effects that cannot be captured by the rate model. Furthermore the new target was presented immediately after a pattern switch even though the activity obviously needs at least a couple time constants (τ_{tr} or τ_m) to propagate through the network. Removing this asynchrony between input and target should further shrink the discrepancy between rate and spiking models.

2.5.6 Parameter tables

For all simulations, we scaled the learning rate proportional to $1/n_h$ for $n_h > 5000$ to ensure convergence.

Table 2.4: (Hyper-)Parameters for (l-) BP, FA, RP, RG (apart from weight initialization, see subsection 2.5.3) & SP as well as the supervised classifier on top of (l-) PCA, ICA and SC representations. Best performing parameters in bold.

Parameter	Description	Value
$n_h = n_1$	Number of hidden units	[10,25,50,100,250,500,1000,2500,**5000**]
p	Rec. field sizes (edge length) in units	[1,5,**10**,15,20,25,28]
α_l	Learning rate	1e-3
N	Number of iterations	1e7 ($\approx$ 167 epochs)
$\mathbf{W}_l^{\text{init}}$	Feed-forward weight initialization	$W_{l,ij} \sim \mathcal{N}(0,1)/(10\sqrt{n_{l-1}})$
$\mathbf{b}_1^{\text{init}}$	Bias initialization	$b_{l,i} \sim \mathcal{U}([0,1])/10$
$\varphi_l(\cdot)$	nonlinearity	ReLU
N_{pop}	Number of populations in hidden layer (l-PCA, l-ICA & l-SC)	[50,100,**500**]

Table 2.5: (Hyper-)Parameters for SC. Best performing parameters in bold.

Parameter	Description	Value
$n_h = n_1$	Number of hidden units	[10,25,50,100,250,500,1000,2500,**5000**]
p	Rec. field sizes (edge length) in units	[1,5,**10**,15,20,25,28]
α_w	Learning rate for $\mathbf{W}_1$	1e-3
α_v	Learning rate for $\mathbf{V}_1$	1e-2
λ	Sparsity parameter	[1e-4,1e-3,**1e-2**,1e-1,1e-0]
S	Resulting sparsity (fraction of 0-elements in l_1)	90 - 99% (dependent on n_h)
τ_{mav}	Time constant of the moving average	1e-2 [1/patterns]
τ_u	Time constant of inner variable $\mathbf{u}_1(t)$	1e-1 [1/iterations]
N_{iter}	Number of iterations solving Equation 2.5	50
N	Number of iterations for SC	1e5
$\mathbf{W}_l^{\text{init}}$	Feed-forward weight initialization	$W_{l,ij} \sim \mathcal{N}(0,1)/(10\sqrt{n_{l-1}})$
$\mathbf{V}_1^{\text{init}}$	Reccurent weight initialization	$\mathbf{0}$
$\mathbf{b}_1^{\text{init}}$	Bias initialization	$\mathbf{0}$ (and kept fixed)
$\varphi_1(\cdot)$	nonlinearity of hidden SC units	ReLU $\max(0, \cdot - \lambda)$

Table 2.6: (Hyper-)Parameters for the spiking LIF l-RP & l-RG models (apart from weight initialization, subsection 2.5.3). Input and target amplitudes are implausibly high due to the arbitrary convention $R = 1\ \Omega$. Best performing parameters in bold.

Parameter	Description	Value
$n_h = n_1$	Number of hidden units	[10,25,50,100,250,500,1000,2500,**5000**]
p	Rec. field sizes (edge length) in units	[1,**10**,28]
τ_m	Membrane time constant	25 ms
R	Membrane resistance	1 Ω
Δ_{abs}	Absolute refractory period	0 ms
ϑ_i	Spiking thresholds	$\vartheta_i \sim \mathcal{N}(\vartheta_{\text{mean}}, \sigma_\vartheta)$
ϑ_{mean}	Mean spiking threshold	20 mV
σ_ϑ	Variance of spiking thresholds	1 mV
amp_{inp}	Input amplitude	500 mA
amp_{tgt}	Target amplitude	500 mA
$I_{\text{bias}}^{\text{ext}}$	External bias input to all neurons	ϑ_{mean}/R
τ_{tr}	Spike trace time constant	20 ms
u_{reset}	Reset potential	0 mV
α	Learning rate	2e-4 (n_h = 5000, 5e-4 for Euler forward)
$\tilde{\alpha}$	Learning rate for LIF rate model	1e-8 (for n_h = 5000)
N	Number of iterations for spiking/rate model	6e6/1e7 ($\approx$ 117/167 epochs)
$\mathbf{W}_l^{\text{init}}$	Feed-forward weight initialization	$W_{l,ij} \sim \mathcal{N}(0,1) \cdot 20/\sqrt{n_{l-1}}$
$\bar{\mathbf{W}}_l^{\text{init}}$	Feed-forward weight initialization (LIF rate)	$W_{l,ij} \sim \mathcal{N}(0,1) \cdot 20/\sqrt{n_{l-1}}$
T_{pat}	Duration of pattern presentation	50 ms (train, 200 ms during testing)
T_{trans}	Duration of the transient without learning	100 ms
Δt	Time step for Euler integrator	$\leq$ 5e-2 ms

3 V1 and beyond? The assets and limitations of competitive temporal Hebbian learning

This unpublished chapter presents ongoing work on competitive Hebbian learning. The results document the current status of the project and are meant as a basis and inspiration for further investigation. In section 3.1, we provide a brief introduction and an overview of related work. Our own model and results are presented in section 3.2 and 3.3, respectively.

Abstract

While reading these words, you are presumably neither aware of the staggering amount of information that floods your retina at every moment, nor of the process making sense of this visual scene. One reason of this ease lies in the powerful representation of visual information in the visual cortex. Since this ability is not given to us by birth, a key question is how such representations can be learned from (i) realistic input and (ii) using learning mechanisms that are biologically realistic, such as Hebbian learning. Here, we address both points and investigate learning on natural images and videos using competitive temporal Hebbian learning. We first build on established learning rules to create a two-layer model that learns simple and complex cell receptive fields, as found in primary visual cortex (V1), from videos of natural image patches. Our model uses simple Hebbian learning rules that learn a sparse code in the first layer, and slow features in the second layer. Second, we explore possibilities to model higher areas by stacking multiple layers of neurons with Hebbian learning to test models in deep convolutional architectures. We evaluate the quality of learned higher representations with a linear classification task on a labeled dataset and find that higher layers do not improve performance. Higher layers learned with competitive Hebbian learning thus seem not to learn hierarchical representations of increasing usefulness—at least not in the tested setup (datasets, architecture, hyper-parameters etc.). Multiple avenues to improve the model are proposed and discussed.

3.1 Introduction

Despite enormous progress, the way the brain uses and learns hierarchical representations, to disentangle complex input signals into unambiguous concepts, remains elusive (DiCarlo and Cox, 2007; Yamins et al., 2014). As introduced in section 1.2, a hierarchical representation is a structure of features, that (i) builds higher-level features out of lower-level ones, and (ii) provides more useful features in higher layers, where usefulness depends on the task (Fukushima, 1988; LeCun, 2012). In this chapter, we will measure usefulness of a representation by the linear classification performance we can achieve when classifying from this representation.

The seminal experiments of Hubel and Wiesel (1962) gave first evidence for such hierarchical processing in the cat's primary visual cortex (V1): orientation-selective V1 simple cells provide basic features for complex cells with more invariant response properties, see Figure 1.2. Since then, numerous experiments and models have grown the understanding of the hierarchical processing in the visual cortex, from V1 to the inferior temporal (IT) cortex (Hubel and Wiesel, 1962; Fukushima, 1980; Riesenhuber and Poggio, 1999; Li and DiCarlo, 2008; Yamins and DiCarlo, 2016).

Growing knowledge of the hierarchical processing in the visual cortex raised the question how such representations can arise. Even though significant amounts of brain structure could be predetermined at birth (Zador, 2019), experimental evidence suggests that cortical representations are at least partly learned and plastic (Wiesel and Hubel, 1963; Hubel and Wiesel, 1963; Stryker and Harris, 1986; Fagiolini et al., 1994; Li and DiCarlo, 2008; Poort et al., 2015).

In our current understanding, synaptic plasticity is the main mechanism of learning in the brain (Hebb, 1949; Stent, 1973; Citri and Malenka, 2008). Many physiological findings suggest that the synaptic modifications underlying learning are Hebbian (Hebb, 1949), i.e. that they depend primarily on the recent state of the pre- and post-synaptic neurons (Sjöström et al., 2001; Caporale and Dan, 2008; Markram et al., 2011). The class of such Hebbian rules can learn interesting features, especially in competitive circuits where multiple neurons inhibit each other (Oja, 1982; Kohonen, 1982; Majani et al., 1989; Hyvärinen and Oja, 1998; Földiák, 1990; Olshausen and Field, 1997; Pehlevan et al., 2018).

Competitive Hebbian learning has also proven useful to model the learning of the specific receptive fields found in V1 simple cells (Von der Malsburg, 1973; Linsker, 1986a,b,c; Miller, 1994), specifically within the framework of sparse coding (SC) (Földiák, 1990; Olshausen and Field, 1996, 1997; Pehlevan and Chklovskii, 2015; Brito and Gerstner, 2016). Similarly, Hebbian learning can model the development of the properties of V1 complex cells through invariance learning (Földiák, 1991; Wallis and Rolls, 1997; Körding and König, 2001), which exploits temporal correlations and redundancies in natural input sequences to extract meaningful features. The first temporal Hebbian rules for invariance learning used a memory trace of neural activity as the post-synaptic factor, instead of instantaneous post-synaptic neural activity, as in classic Hebbian learning (Földiák, 1991). Slow feature analysis (SFA) Wiskott and

Sejnowski (2002), and the connection between SFA and complex cell properties (Berkes and Wiskott, 2005) and Hebbian learning (Sprekeler et al., 2007; Sprekeler and Wiskott, 2011), gave such heuristic learning rules a normative justification.

Beyond models of V1 simple and complex cells, the modeling of neural representations and learning in higher areas in the visual cortex with Hebbian learning remains a challenge. Hierarchical models of the cortex explain higher representations to some extent (Fukushima, 1980; Riesenhuber and Poggio, 1999; Serre et al., 2007; Yamins et al., 2014), but do not provide a satisfying and biologically plausible model for learning of these representations. Furthermore, classical competitive Hebbian learning in deep hierarchical networks was found to be of limited value for brain modeling (Yamins et al., 2014) and machine learning (Bahroun et al., 2017; Stuhr and Brauer, 2019; Amato et al., 2019; Obeid et al., 2019; Talloen et al., 2021; Lagani et al., 2021; Miconi, 2021).

Previous models have investigated simple cells, complex cells or feature hierarchies, with varying extent of biological plausibility and locality in their respective learning rules. Our model, introduced in the next section, builds on this large body of models and aims at combining existing ideas into a simple set of learning rules, in order to advance the understanding of Hebbian learning in deep networks. Single building blocks of our model are closely related to existing models, such as the sparse coding model (Földiák, 1990), invariance learning using the trace-rule (Földiák, 1991) or the idea of building a hierarchy through invariance learning (Wallis and Rolls, 1997). Our main contributions are the formulation of simple, local and on-line learning rules in a unifying framework that enables stacking of simple and complex cell layers, as well as a thorough investigation of the learned representations.

3.2 Competitive temporal Hebbian learning: a unifying learning rule for V1 cells

We now introduce our model, competitive temporal Hebbian learning, which combines models of competitive learning (Oja, 1982; Földiák, 1990) with temporal invariance learning (Földiák, 1991; Rolls and Milward, 2000). Our basic network model consists of two layers of neurons, with feedforward connections between layers and recurrent inhibitory connections within layers, see Figure 3.1. There are no feedback (top-down) connections in this model. The first layer receives external input and passes its activity to the next layer through the feedforward connections. The transmitted signal serves as input to the second layer etc. Different layers in the network contain different types of model neurons, representing either simple or complex cells. Neurons in both layers use the same dynamics and the difference between the two cell types lies solely in the learning rule, i.e. in the way synaptic weights are updated.

As depicted in Figure 3.1 **a**, a layer of the neural network model is defined as a dynamical system of the membrane potentials $\boldsymbol{u}(t)$ and neural activities $\boldsymbol{a}(t)$, as common for competitive

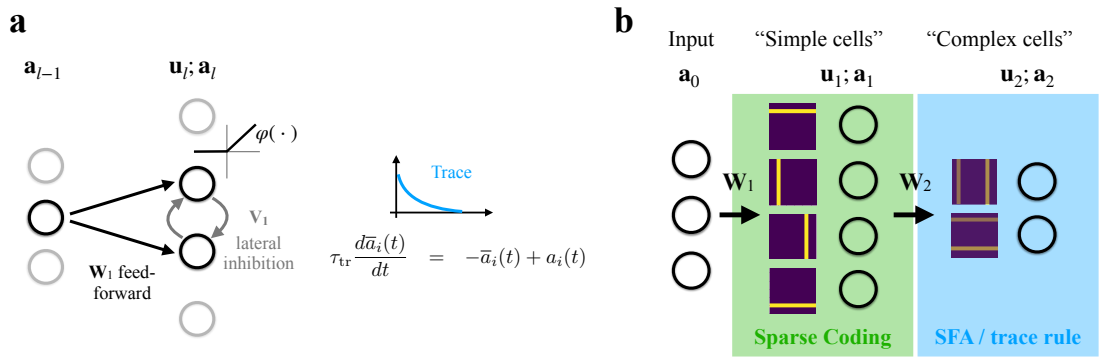


Figure 3.1: Network model for competitive temporal Hebbian learning. **a** Every layer in the model receives input from the layer below (or the input layer) through feedforward weights $\mathbf{W}$ and implements competition through lateral inhibitory weights $\mathbf{V}$. For simplicity we omit the inhibitory interneurons. Neural activities are determined by Equation 3.1. To keep a memory of recent activity, a trace is calculated as an exponential average with time scale τ_{tr} over past activity. **b** A two-layer model of V1 simple and complex cells contains two layers with the same dynamics, but different learning rules, as summarized in Equation 3.3. The first layer learns a sparse code to model V1 simple cells, the second layer extracts slow features to learn invariant features, such as V1 complex cells.

networks (Hopfield, 1984; Olshausen and Field, 1996; Miller and Fumarola, 2012). We model the activity of neuron i using its spike rate a_i, i.e. one real number per neuron, that is an instantaneous function of the neuron's membrane potential: $a_i(t) = \varphi(u_i(t) - b_i)$ (Gerstner et al., 2014). The neuron-wise function φ is the neuronal activation function, i.e. the f-I-curve (Gerstner et al., 2014), which we approximate here by a piece-wise linear rectifying linear unit (ReLU). The set of firing rates $(a_1, a_2, \ldots)^\top$ forms the activity vector $\mathbf{a}$. The vector $\boldsymbol{b}$ contains the set of biases b_i, or thresholds, depending on the interpretation. The vector $\mathbf{u} = (u_1, u_2, \ldots)^\top$ is the set of leaky membrane potentials that integrate feedforward input $\mathbf{I}$ through feedforward synapses $\mathbf{W}$ and recurrent input $\boldsymbol{a}(t)$ through recurrent synapses $\mathbf{V}$ with a membrane time constant τ_m. To implement competitive dynamics, the recurrent synapses $\mathbf{V}$ are assumed to be inhibitory (Olshausen and Field, 1996; Földiák, 1990), which we model by a global negative sign (and $\mathbf{V} \geq 0$). For simplicity, we omit the inhibitory interneurons required by Dale's law (Dale, 1935) and assume that the neurons in the main network directly inhibit each other—a common trick for such networks (Olshausen and Field, 1996; Földiák, 1990). With this, the dynamics of the model are

$$\begin{aligned} \tau_m \frac{d\mathbf{u}(t)}{dt} &= -\mathbf{u}(t) + \mathbf{W} \cdot \mathbf{I} - \mathbf{V} \cdot \mathbf{a}(t) \\ \mathbf{a}(t) &= \varphi(\mathbf{u}(t) - \mathbf{b}) \,, \end{aligned} \tag{3.1}$$

which we numerically integrate to obtain neural activities. Through the competitive dynamics,

the most active neurons will inhibit the other neurons, which makes the latter likely to drop below the firing threshold and 'lose' the competition. If we assume that **V** is symmetric, this dynamical system is guaranteed to converge to a fixed point (Hopfield, 1982; Olshausen and Field, 1996; Gerstner et al., 2014), however, in practice, convergence is often achieved even for asymmetric **V** (Olshausen and Field, 1997; Brito and Gerstner, 2016).

We now introduce the learning rules that describe the plasticity of the synaptic strengths of the feedforward connections **W** and recurrent connections **V**, as well as the values of the biases **b**. All updates are Hebbian and only use quantities that are locally available, such as pre- and post-synaptic activity, the value of the synaptic strength itself, and constants. To give learning access to temporal correlations, we define a temporal trace of recent activity as an exponential average with time scale $\tau_{tr} \gg \tau_m$ (Földiák, 1991),

$$\tau_{tr} \frac{d\overline{a}_i(t)}{dt} = -\overline{a}_i(t) + a_i(t) \,, \tag{3.2}$$

and add such traces to the set of eligible quantities that can be used to calculate updates (Gerstner et al., 2018). With this we can summarize our learning rules for a layer l as:

$$\begin{aligned} \Delta W_{l,ij} &= \eta_W \, \text{post}_i \left[a_{l-1,j} - \text{post}_i \, W_{l,ij} \right] \\ \Delta V_{l,ih} &= \eta_V \left[\left(a_{l,i} - \langle a_{l,i} \rangle \right) \left(a_{l,h} - \langle a_{l,h} \rangle \right) - \langle a_{l,i} \rangle \langle a_{l,h} \rangle \, V_{l,ih} \right] \\ \Delta b_{l,i} &= \eta_b \left(\theta \left(a_{l,i} \right) - p \right) \,, \end{aligned} \tag{3.3}$$

where $\eta_V > \eta_b > \eta_W$ are learning rates and $\langle \cdot \rangle$ denotes a long-term average with a large time scale $\tau \gg \tau_{tr} \gg \tau_m$. $\theta(\cdot)$ denotes the Heaviside step function and the parameter p determines the final sparsity of the representation.

The learning rules in Equation 3.3 describe the plasticity of both cells types, simple and complex cells, with only the post-synaptic factor being different: for simple cells it is simply the neural activity of layer l,

$$\text{post}_i = a_{l,i}, \tag{3.4}$$

and for the complex cells it is the memory trace of that activity,

$$\text{post}_i = \overline{a}_{l,i}. \tag{3.5}$$

Using the trace as the post-factor implements a version of the 'trace rule' introduced by Földiák (1991) which was shown to extract slow features similar to SFA (Sprekeler et al., 2007). Our learning rules in Equation 3.3 were inspired by the competitive learning rules of Földiák (1990) and the trace rule in Földiák (1991), however there are important differences: (i) Compared to Földiák (1990, 1991), we include a homeostatic factor, i.e. *weight decay*, as in Oja (1982); Zylberberg et al. (2011) to stabilise the L^2 norm of the feedforward weights **W**, see first line of Equation 3.3. (ii) Compared to other competitive learning rules (Földiák, 1990; Olshausen and Field, 1997; Zylberberg et al., 2011), our learning rule for the recurrent inhibitory synapses **V** implements decorrelation using normalised neural activities (second line of Equation 3.3).

That is, after convergence, the inhibitory weights **V** implement the cross-correlation matrix of neural activities, as we derive by setting the expectation of the second line of Equation 3.3 to zero:

$$\begin{aligned} \langle \Delta V_{l,ih} \rangle &\stackrel{!}{=} 0 \\ V_{l,ih} &= \left\langle \frac{\left(a_{l,i} - \langle a_{l,i} \rangle\right)\left(a_{l,h} - \langle a_{l,h} \rangle\right)}{\langle a_{l,i} \rangle \langle a_{l,h} \rangle} \right\rangle \end{aligned} \tag{3.6}$$

Following (Földiák, 1991), we prevent the inhibitory weights to become excitatory throughout learning by setting all emerging negative elements of **V** to zero (note the global negative sign in Equation 3.3). Likewise, we exclude self-inhibition of neurons by keeping the diagonal of the recurrent weights at zero: $V_{l,ii} = 0 \; \forall i$. (iii) Compared to Földiák (1990); Zylberberg et al. (2011), who regulate the L^1 norm of the learned representations, the Heaviside step function $\theta(\cdot)$ in our bias learning rule (last line of Equation 3.3) directly regulates the L^0 norm, i.e. the number of non-zero elements, of the representations. Equation 3.1 and Equation 3.3 summarize the network dynamics and learning rules for both, simple and complex cell models, and thus provide a novel unifying framework for simple and complex cell modeling.

The actual implementation of dynamics (Equation 3.1) and plasticity (Equation 3.3) uses the deep learning framework `PyTorch` (https://pytorch.org) to allow batch processing, pre-coded optimizers like ADAM (Kingma and Ba, 2015) and easy GPU acceleration. Code is available at: https://github.com/berndilling/SparsePooling/tree/master/SparsePooling_pytorch.

3.3 Empirical results

We first study competitive temporal Hebbian learning in our basic 2-layer model: one layer of simple cells followed by one layer of complex cells, see Figure 3.1 **b**. The two layers are fully connected, which means that every simple cell's receptive field spans the whole input (e.g. an image patch), and every complex cell can connect to every simple cell.

In a second part, we extend this model to convolutional architectures with more than two layers. The first 2-layer model, comprising one simple cell layer followed by a complex cell layer, will serve as the building block of this deeper model.

3.3.1 Emerging simple and complex cell properties in a 2-layer model

Moving bar toy example As a first proof of concept, we check whether our 2-layer model learns features akin to these of simple and complex cells in the case where the input data is simple and fully controlled and the learned features are easily interpretable.

To this end, we apply our model to randomly generated synthetic videos of binary horizontal or vertical bars that move orthogonally to their orientation. The video frames are 10×10 pixels

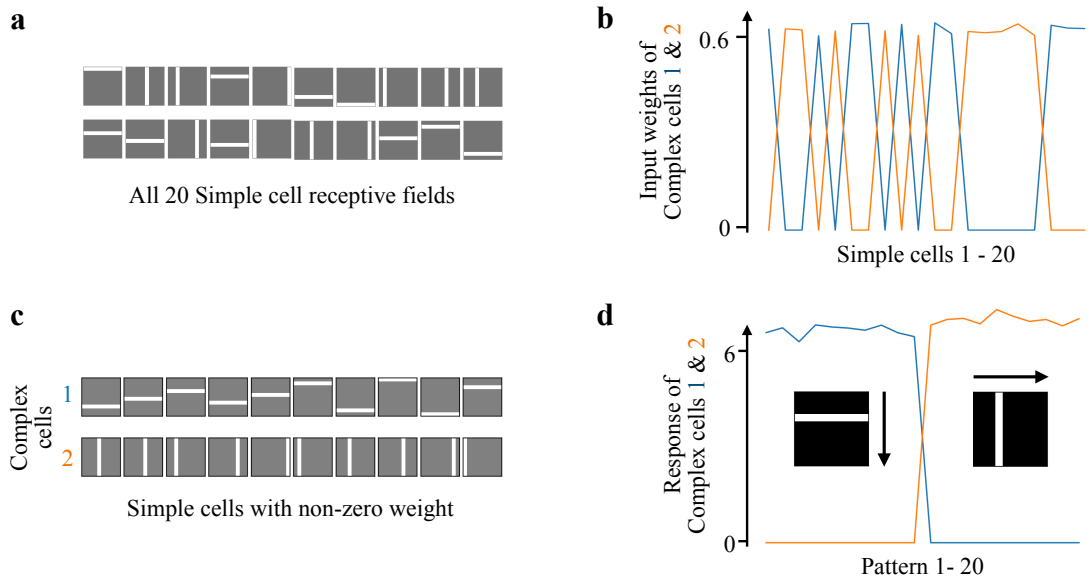


Figure 3.2: Emergence of simple and complex cells on a moving bar toy data set. **a** Depiction of all 20 simple cell receptive fields, displayed as images. The learned features perfectly recover all possible input frames of this simple data set. **b** After learning, the weights $\mathbf{W}_2$ between simple and complex cell layer connect two distinct sets of 10 simple cells to two different complex cells. **c** For each complex cell (lines) we collect all simple cells with non-zero connection and represent the latter by their receptive fields. We see that each complex cell only connects to simple cells sharing the same orientation preference. This suggest that complex cells are orientation selective but position invariant. **d** Looking at the complex cell responses to images of horizontal and vertical bars at different locations, we identify complex cell behaviour: orientation selectivity and positional invariance.

and the bars span the whole image along the direction of their orientation and can move across the whole image along the direction of their movement. For every video, we randomly draw the bar orientation (horizontal or vertical), the starting position of the bar on the image, as well as the moving direction at the start. Bars move at a constant speed of one pixel per time step and the videos comprise 20 frames each, assuming periodic boundaries.

The network model has a receptive field size of 10×10 pixels to match the image size of the video frames. A first layer of 20 simple cells is followed by a second layer of 2 complex cells, with a memory trace time constant of $\tau_{tr} = 8$ frames. The sparsity parameter was set to $p = 1/20 = 0.05$ for the simple cells and $p = 1/\text{number of complex cells} = 1/2 = 0.5$ for the complex cells, respectively. We initialise the feedforward weights $\mathbf{W}_1, \mathbf{W}_2$ and the biases $\mathbf{b}_1, \mathbf{b}_2$ randomly and the recurrent weights $\mathbf{V}_1, \mathbf{V}_2$ at zero. We then train both layers of the model simultaneously for 5 epochs (à 10000 videos à 20 frames) by which time learning has fully converged. We use ADAM with standard parameters, a learning rate of 0.005, and a batch size of 640 frames.

After training, we investigate the trained model by looking at the learned weights and neural

responses, as summarized in Figure 3.2. First, we look at the feedforward weights $\mathbf{W}_1$ of the first layer of simple cells. In Figure 3.2 **a**, we show the receptive field of each of the 20 neurons by plotting its fan-in weights (respective row in $\mathbf{W}_1$) as an image. The sparse features learned by the simple cells perfectly recover all possible input frames of the binary bar data set. Figure 3.2 **b** & **c** show the connections $\mathbf{W}_2$ between simple and complex cell layer. After learning, the two complex cells receive input from two distinct sets of 10 simple cells each. For each complex cell (lines in **c**) we collect all simple cells with non-zero connection and represent these cells by their receptive fields, as in **a**. We see that complex cell 1 or 2 only connect to simple cells with horizontal or vertical orientation preference, respectively. Hence, complex cells receive input from simple cells sharing the same orientation preference at different positions, suggesting orientation selectivity and position invariance. To verify this intuition, we look at complex cell responses to images of horizontal and vertical bars at different locations in Figure 3.2 **d**. Indeed, complex cell 1 exclusively responds to horizontal bars, independent of their position. Conversely, complex cell 2 only responds to vertical bars but features the same positional invariance.

Taken together, we conclude that our 2-layer model recovers simple and complex cell behaviour when trained on a simple toy dataset of videos of moving binary bars. The interpretability of both, model and dataset, allowed us to probe and understand the emerging representations, e.g. simple cells encoding separate bars and complex cells encoding the orientation. In the next paragraph we move to more realistic data and a bigger network model.

Videos of natural image patches We now apply our 2-layer model of simple and complex cells to the more biologically realistic scenario of videos of natural image patches. As opposed to natural videos (Einhäuser et al., 2002), we focus on simplified dynamics as depicted in Figure 3.3 **a**: we extract movies of natural image patches by sliding a 10×10 pixel window over the whitened natural images provided by Olshausen and Field (1996, 1997). To obtain rich data, we randomly draw the natural image, the starting position on the image, as well as the sliding direction for every patch video. The speed of the sliding window is fixed to one pixel per time step. After video creation, the set of all frames is re-whitened using ZCA whitening, see e.g. Krizhevsky (2009) appendix A.

As in the moving bar toy example above, the network layers are fully connected, allowing each simple cell to connect to all inputs and each complex cell to connect to all simple cells. Again, the network model has a receptive field size of 10×10 pixels to match the image size of the video frames. The first layer contains 400 simple cells, the second layer 10 complex cells, with a memory trace time constant of $\tau_{tr} = 8$ frames. The sparsity parameter was set to $p = 0.05$ for the simple cells and $p = 1/\text{number of complex cells} = 0.1$ for the complex cells, respectively. We initialise all feedforward weights and biases randomly, recurrent weights to zero, and train both layers of the model simultaneously for 100 epochs (à 10000 videos à 20 frames) by which time learning has fully converged. We use ADAM with standard parameters, a learning rate of 0.005, and a batch size of 640 frames.

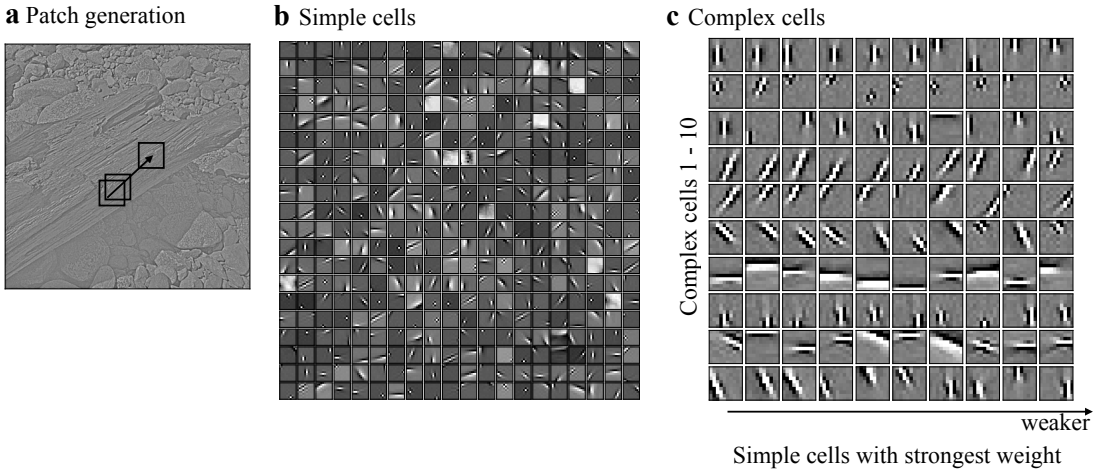


Figure 3.3: Emergence of simple and complex cells on videos of natural image patches. **a** We use the whitened natural images provided by Olshausen and Field (1996, 1997) and extract movies of natural image patches by sliding a patch window over the image. For every patch video, the natural image, the starting position on the image and the sliding direction are randomly drawn. After video creation, the set of all frames is re-whitened using ZCA whitening. **b** Depiction of all 400 simple cell receptive fields, displayed as image patches. The simple cell extract a rich repertoire of Gabor-like filters featuring a wide range of orientations, positions and spatial frequencies. **c** For each complex cell (lines) we collect the simple cells (represented by their receptive fields) with the strongest connection to this complex cell. We see that each complex cell mainly connects to simple cells sharing the same orientation preference.

Figure 3.3 **b** shows all 400 simple cell receptive fields after learning, displayed as image patches. We observe a rich repertoire of Gabor-like filters featuring a range of orientations, positions and spatial frequencies, just as expected for the sparse coding model of simple cells (Olshausen and Field, 1997). To study the receptive fields of the complex cell layer, we collect the simple cells (represented by their receptive fields) with the strongest connection to each complex cell, see Figure 3.3 **c**. Similar to the moving bar example, each complex cell receives input mainly from simple cells sharing the same orientation preference.

We test the trained model for simple and complex cell properties by looking at cell responses to synthetic stimuli. As depicted in Figure 3.4 **a**, we generate 6 sequences of images of bars of different orientation sweeping across different positions. Figure 3.4 **c** shows the responses of simple cells 1-100 (out of 400) and all 10 complex cells to the six input sequences. The simple cells respond orientation and position selective, leading to a sparse and quickly varying code. Complex cells responses are still highly orientation selective but position invariant over a large range. The resulting sparse but slowly varying code demonstrates that our complex cells extract slow features. Interestingly, the code is still highly orientation selective and distributed, as can be seen comparing sequences iii and v: as we infer from Figure 3.4 **b**, there is no complex cell receiving input from simple cells with an orientation preference at 45° that match the bars in sequence iii. The resulting response is a distributed code with multiple complex cells

'sharing' the response (purple boxes). On the other hand, the bars at 65° in sequence v are matched to the simple cells' orientation preference that connect to complex cell 7 (orange).

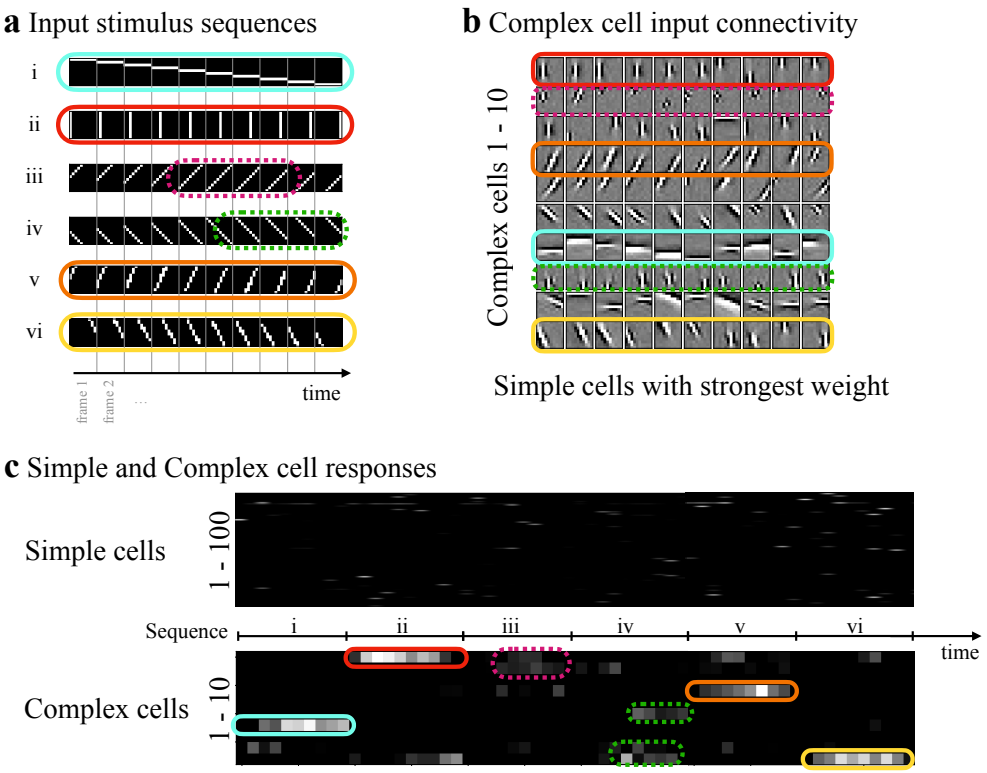


Figure 3.4: Probing simple and complex cells trained on videos of natural image patches. **a** We test our model on sequences of 10 binary white bars on black background for 6 different orientations. Every sequence evolves from the leftmost 10×10 pixel image to the rightmost one (images separated by thin black vertical lines). The bar orientations are (i) 0° (horizontal, highlighted by light blue box), (ii) 90° (vertical, red box), (iii) 45°, (iv) 135°, (v) 65° (orange box) and (vi) 300° (yellow box). **b** Same as in Figure 3.3 c. Colored boxes indicate the maximally activating input sequence for selected complex cells. **c** Simple cell (1-100 out of 400) and complex cell (1-10 out of 10) responses for input sequences i-vi. Simple cell responses are orientation and position selective, leading to a sparse and quickly varying code. Complex cell responses are still highly orientation selective (see e.g. sequence iii (purple) vs v (orange)) but position invariant over a large range, leading to a sparse but slowly varying code.

As a result, complex cell 7 clearly responds, confirming the sharp orientation selectivity. The same observation can be made for complex cell 1 comparing sequences iv (green) and vi (yellow).

In summary, we observe emerging simple and complex cell properties in our 2-layer model as illustrated in Figure 3.1. The model follows the dynamics in Equation 3.1, applies the learning rules in Equation 3.3 and was trained on biologically realistic temporal data such as videos of natural image patches. We note here, that the same qualitative results were obtained when the competitive dynamics are replaced by a much simpler k-winners-take-all dynamics that

massively speeds up simulations, see subsection 3.5.2.

The above analyses are qualitative and more quantitative investigation, as e.g. in Berkes and Wiskott (2005) or Rehn and Sommer (2007), would be needed to compare our model to more established models of V1 simple and complex cells. However, the main focus of this work lies on generalising simple learning rules, that lead to V1-like cells, to hierarchical networks and *not* on improving a specific aspect of V1 cell modeling. With this in mind, we conclude this section by noting (i) the qualitative success of our model and (ii) its simplicity, which facilitates generalisation to a hierarchical model in the next section.

3.3.2 Towards hierarchical Hebbian learning in deep convolutional networks

General idea We have demonstrated that competitive temporal Hebbian learning in shallow 2-layer networks leads to complex and biologically realistic representations comparable to those in primary visual cortex V1. It is natural to ask whether we can use the same learning rules for deeper networks with more layers to learn hierarchical representations, which is what we address in this section.

When going from the shallow 2-layer network to more layers, there are four prominent design choices to make: (i) After one simple- and one complex cell layer, which layer type(s) should follow?, (ii) What architecture should our network have, i.e. fully connected or convolutional?, (iii) What data should we train the model on, and (iv) How do we define and evaluate success?

Concerning point (i), we follow existing work on modeling the hierarchy of the visual cortex (Riesenhuber and Poggio, 1999; Serre et al., 2007) and early convolutional neural network architectures (LeCun, 1989) and suggest an alternating order of simple and complex cell layers. The intuition behind these architectures is to extract interesting features (simple cells), then achieve some local invariance by spatial 'pooling' over features (complex cells), and repeat this concept iteratively. The key idea is to achieve this pooling by the temporal invariance learning of our complex cell model, implementing slow feature analysis (SFA). In practice, we consider the 2-layer model (Figure 3.1 **b**) as one 'module' and stack several of these modules to obtain a deep network with alternating simple cells, that implement sparse coding, and complex cells, that implement SFA. We note here, that simply stacking simple cell layers (i.e. sparse coding) is not expected to lead to interesting hierarchical features due to the underlying linear generative model (Olshausen and Field, 1997), i.e. the generative model of the deep network would still be linear.

Following this train of thought, we immediately see how to address point (ii): in a fully-connected architecture, the receptive field size of a first simple cell layer is the same as for a second layer of complex cell, and spans the whole input image (or whole patch). Thus, the pooling of the complex layer cells will happen over the whole image and subsequent layers of simple or complex cells will not be able to extract useful, hierarchical features. To solve this, we choose a convolutional architecture, as depicted in Figure 3.5, where higher layers have

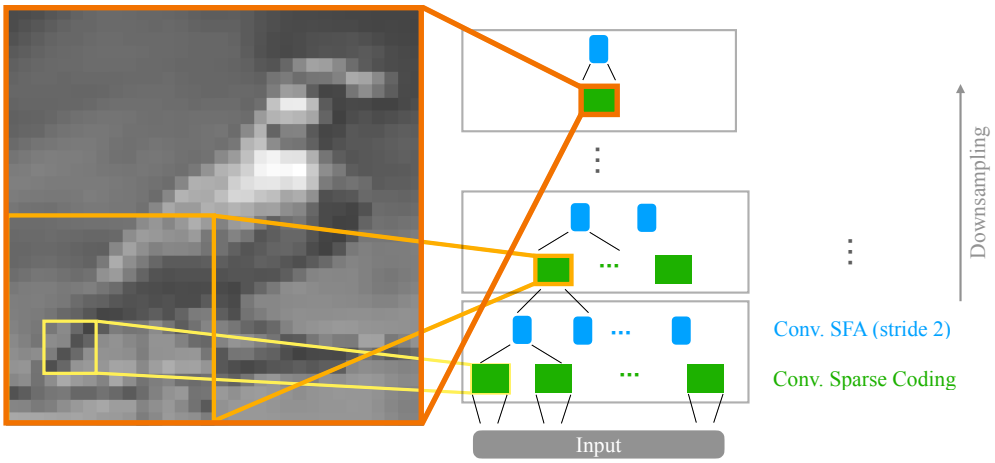


Figure 3.5: Sketch of convolutional architecture for hierarchical competitive temporal Hebbian learning. A convolutional version of the 2-layer model of Figure 3.1 forms a 'module', with simple cell layers (green) implementing convolutional sparse coding, and complex cell layers (blue) implementing convolutional slow feature analysis (SFA), following the learning rules in Equation 3.3. Several modules are then stacked to obtain a deep network. Neurons in deeper layers have larger receptive fields (yellow, orange and red boxes) due to downsampling through strided convolutional layers.

increasingly larger receptive fields.

We do not explicitly address the biological implausibilities arising from weight sharing in such architectures. However, our learning rules (Equation 3.3) are fully local and unsupervised and we expect similar learning happening at all locations, given spatially homogeneous image statistics. In a network without weight sharing, i.e. with independent weights at different locations, we thus expect that these independent weights naturally converge to similar values at all locations.

Because our complex cell model is based on temporal learning through SFA, we use patch videos extracted from the STL-10 dataset (https://cs.stanford.edu/~acoates/stl10/). We choose STL-10 because we evaluate our model on this data set, see below. The procedure to extract patch videos from static images is similar to the one in subsection 3.3.1 (more details below).

It is hard to define and evaluate the quality of deep representations beyond one or two layers (point (iv)). Looking at receptive fields becomes vague and ambiguous beyond the complex cell level, and representations become hard to interpret. We choose to evaluate the learned deep representations of our model by linear classification on the labeled part of the STL-10 dataset (https://cs.stanford.edu/~acoates/stl10/, see Figure 3.6 **a**). We thus take the linear separability of the 10 classes in STL-10 as a proxy for the quality of the representation and define success as the test accuracy of this linear classification.

Simulation details As model evaluation is done using the STL-10 dataset, we use patch videos extracted from the STL-10 dataset as training data. The procedure to obtain videos of moving image patches is the same as in subsection 3.3.1, except that we (i) choose a patch size of 25×25, pixels to account for the larger receptive field of the whole convolutional network, and (ii) skip the step of ZCA whitening after video extraction.

We use a convolutional architecture, as sketched in Figure 3.5. The network comprises a stack of three 'modules', each of which is a convolutional version of the 2-layer model of simple and complex cells shown in Figure 3.1. We refer to this network as *SC-SFA*. As controls, we implemented three algorithms using the same network architecture: *SC-MaxPool* which replaces the complex cell layers with standard MaxPool-layers, *Supervised BP* which has the same architecture but is trained with standard supervised backpropagation (BP) and *Fixed random*, which is a network with fixed random weights (i.e. weights frozen at random values and not updated). The exact architecture, along with all relevant parameters, is summarized in Table 3.1.

We initialise all feedforward weights and biases randomly, all recurrent weights at zero, and train all layers of the model simultaneously for 100 epochs (à 10000 videos à 20 frames) by which time learning has converged. We use ADAM with standard parameters, a learning rate of 0.005, and a batch size of 640 frames.

For model evaluation, we train a linear classifier on the representations of the labeled STL-10 training set. To evaluate, for instance, the representation of layer number 5 of our model, we feed the standard static STL-10 images through our model and save the image representations of layer 5. These representations are then (i) averaged over the spatial dimensions for each feature (global MeanPool) and (ii) normalised to ensure a constant L^2-norm across representations of different layers. We then train a linear classifier on the representations of the train set and evaluate test classification accuracy using the representations of the test set. The linear classifier is a linear, fully-connected layer, trained with the softmax-crossentropy loss on the image labels. We train this classifier for 100 epochs, using ADAM with standard parameters, a learning rate of 0.005, and a batch size of 128.

Table 3.1: Convolutional architecture for hierarchical Hebbian learning. Convolutional sparse coding layers (conv-SC) have stride (1, 1), convolutional SFA (conv-SFA) or MaxPool layers use stride (2, 2).

Module	Layer	Layer type	Number of neurons	p	τ_{tr} [frames]
1	1	3×3 conv-SC	100	0.05	-
	2	2×2 conv-SFA or MaxPool	100	0.18	8
2	3	3×3 conv-SC	200	0.05	-
	4	2×2 conv-SFA or MaxPool	200	0.18	8
3	5	3×3 conv-SC	400	0.05	-
	6	2×2 conv-SFA or MaxPool	400	0.18	8

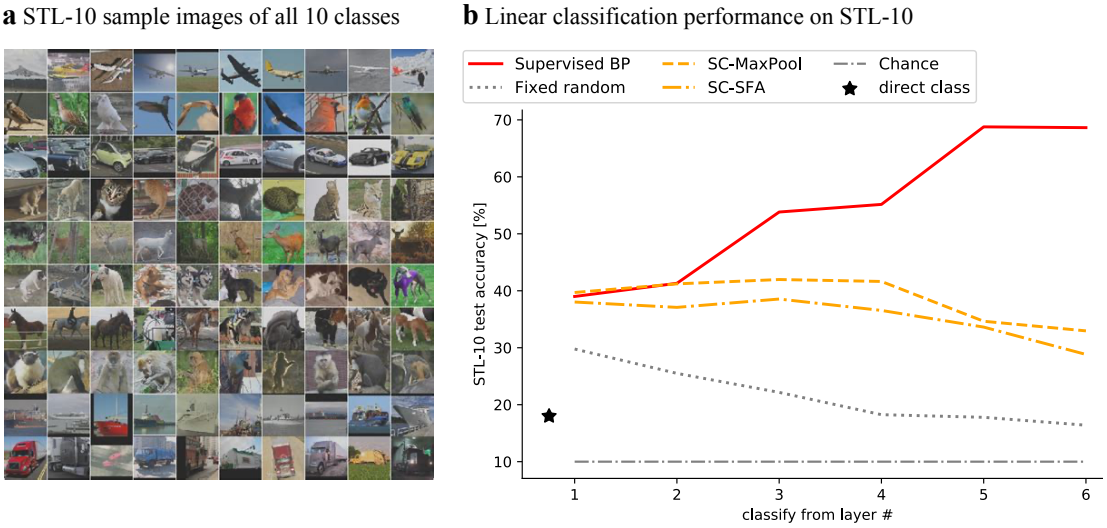


Figure 3.6: Evaluation of hierarchical competitive temporal Hebbian learning on STL-10 classification. **a** Sample images from the STL-10 dataset showing all 10 classes. Images taken from https://cs.stanford.edu/~acoates/stl10/ **b** We evaluate the quality of deep layers' representations by measuring test accuracy of a linear classification on STL-10 on this layer. Compared to chance level (dash-dotted), direct classification on pixels (black star) and a fixed random network (dotted), hierarchical temporal Hebbian learning (SC-SFA), as well as the control SC-MaxPool, yield useful representations. However, SC-SFA does *not* lead to hierarchical representations as performance does not increase with depth. The red line for supervised backpropagation (BP) shows that the architecture can in principle support hierarchical representations, if another learning rule is used.

Simulation results As described above, we evaluate the quality of the representation of a certain layer in the network by measuring test accuracy of a linear classification on STL-10 on the respective layer.

We consider two lower performance bounds, that any learning rule should surpass: The first one is chance level at 10 % classification accuracy (Figure 3.6 **b**, dash-dotted). The second one is direct classification on pixels which yields an accuracy of 18 % (black star). Interestingly, a network with fixed random weights (fixed random, dotted) already increases performance compared to direct classification, however, these random representation are *not* hierarchical as performance does not increase with depth.

The deep network of alternating simple and complex cell layers (SC-SFA, dash-dotted) yields useful representations in that performance increases compared to the network with fixed random weights. However, these representations are again not hierarchical as performance drops after layer 3. To investigate this finding further, we note that, at least concerning translations, the pooling that is learned by the complex cells, can at most be as good as a hard-coded MaxPool layer. We thus implement the control *SC-MaxPool* (dashed), which replaces the complex cell layers with standard MaxPool-layers. We find that the representations emerging

from SC-MaxPool are non-hierarchical as well, as performance saturates at layer 2 and starts to drop after layer 4. Similar observations hold if SC-SFA or SC-MaxPool are trained on patch videos extracted from whitened natural images (data set provided by Olshausen and Field (1996), as in subsection 3.3.1), see subsection 3.5.1

As an upper performance bound, we train the convolutional network on supervised image classification of STL-10, directly on the training labels and using end-to-end backpropagation (*Supervised BP*, red solid line). We see that performance increases with depth and that the architecture can support hierarchical representations.

Based on our evaluation method, linear classification on STL-10, we draw the preliminary conclusion that competitive temporal Hebbian learning (SC-SFA) in its current form does not lead to hierarchical representations in deep networks.

3.4 Discussion

This project aims at finding biologically plausible Hebbian learning rules that reproduce previous results on V1 simple and complex cells but, in addition, generalise to deeper architectures to build hierarchical representations. We propose a simple set of dynamics (Equation 3.1) and learning rules (Equation 3.3) for competitive Hebbian learning. Using a 2-layer model of simple and complex cells and videos of natural image patches as training data, we find typical receptive fields and response properties of V1 simple and complex cells after training. We applied the same learning rules to a deep convolutional model, where simple and complex cell layers alternate and receptive field sizes increase with depth. Although the deep model improves upon a network with fixed random weights, we did not find a deep hierarchy of features since the usefulness of representations decreases after layer 3, if tested on linear classification on STL-10.

Troubleshooting Currently we see two possible reasons why the model fails to build hierarchical representations:
1) Training a deep network with Hebbian learning rules brings *many design choices* with it. We motivated and justified our choices for the four most prominent ones about learning rules, architecture, training data and evaluation. However, we see many more options for all of these choices and the ad-hoc nature of most of them makes principled exploration hard (e.g. an architecture known to perform well with one learning rule can fail with another rule; or there might be an emergent hierarchy which we miss to detect with our evaluation method). We note here, that in terms of hyper-parameters of learning rule and architecture, neither manual nor automatic black-box optimisation significantly improved the results. We thus see more potential in more radical changes such as changing the training data (i.e. the task and dataset) or the evaluation method (e.g. direct analysis of deep receptive fields or neuronal representations in each layer). Furthermore, we note that several independent studies of deep Hebbian learning report mixed or negative results as well (Bahroun et al., 2017; Amato et al.,

2019; Lagani et al., 2021; Miconi, 2021), suggesting that design choices alone are unlikely to be the cause of failure.

2) We cannot exclude a *conceptual problem* with the idea of building deep representations from purely local, unsupervised learning rules as in Equation 3.3. The main motivation of the current model (SC-SFA) was to mimic the cascade of simple and complex cell layers of early models of the visual cortex (Riesenhuber and Poggio, 1999) or convolutional neural networks (Fukushima, 1980; LeCun, 1989; Patel and Baraniuk, 2016): extract useful features through sparse coding in the simple cell layers and implement local invariance ('pooling') through SFA in the complex layers. Our approach was to learin both steps by generalised Hebbian plasticity rules, and then repeat the process over severyl layers, i.e. stacking modules of sparse coding and SFA. Given this motivation, there are two questions that point towards a potential conceptual problem. First, how do we know that the learned features of higher layer simple cells are hierarchical, i.e. useful for evaluation tasks like classification? In contrast, supervised backpropagation assigns exact credit to each weight and automatically learns optimal features, whereas unsupervised learning with Hebbian plasticity does not. Second, it is hard to judge the usefulness of the invariances learnt by the complex cell layers. Concerning translations, the pooling that is learned by the complex cells, can at most be as good as a hard-coded MaxPool or MeanPool layer. For other transformations, such as rotation of an object or pose of limbs with respect to the torso, the flexible SFA approach seems superior. However, it is not obvious which information should be kept or discarded, and how we should evaluate performance.

Future directions Many attempts to make the right design choices and working on potential conceptual problems have led me to a multitude of potential future directions, some of which I already partially explored. In this concluding paragraph I gather these ideas as starting points of future work.

- **Learning rules** Changing the class of the learning rule from Hebbian to a less constrained framework could help. Options include a third factor, such as reward or surprise, that modulates the speed of learning Gerstner et al. (2018); Liakoni et al. (2021), or an extension of the framework to contrastive predicting learning as in Hinton (2002); Van den Oord et al. (2018); Löwe et al. (2019). Preliminary results with contrastive slow feature analysis, where a scalar third factor is used for contrastive updates, did not show significant improvement. However, I show in chapter 4, that contrasting, combined with a fourth, neuron-specific factor that implements dendritic prediction of future somatic activity, can indeed learn hierarchical representations.

- **Architecture** So far, I studied fully-connected and convolutional networks that operate in feedforward model, except for the recurrence implementing the competition. Introducing top-down connections from higher to lower layers could help solving the credit assignment problem to some extent: higher layers would have a channel to

communicate to lower layers what features are useful to improve downstream representations. This direction has seen some interest (Boutin et al., 2020) but remains largely unexplored.

- **Datasets** The temporal training data we use are not actual videos, in contrast to the data used in Einhäuser et al. (2002). We introduced a temporal dimension into our training data by sliding a patch window over static images. This procedure is somewhat artificial and, especially since we only use translations, fairly different from natural image transformations. A dataset like smallNORB (https://cs.nyu.edu/~yann/research/norb/), which features images of objects taken from many different perspectives, allows to generate sequences with much richer transformations. Surprisingly, preliminary experiments using such smallNORB sequences as input data do not significantly improve the quality of deep representations (measured by linear classification on smallNORB labels). Another step is using actual video data sets, such as UCF-101 (Soomro et al., 2012), which we have not explored yet. However, such experiments should be feasible since our `PyTorch` implementation of competitive temporal Hebbian learning supports efficient batch processing and GPU acceleration.

- **Task** Our unsupervised framework is generic and agnostic of the evaluation method, which we identified above as a potential conceptual problem. However, local Hebbian learning rules can often be linked to per-layer objective functions: standard examples are the link of Hebbian rules to principal (Oja, 1982) and independent component analysis (Hyvärinen and Oja, 1998) and sparse coding (Olshausen and Field, 1997; Brito and Gerstner, 2016). This leaves some freedom to choose the actual (unsupervised) objective, that a layer optimizes, and hence the learning rule it implements. So far, we only explored the case where every layer of a kind (simple or complex cell) optimises the same objective (sparse or slow features, respectively). However, this scheme can be generalised and different layers (even of the same kind) could be given different objectives. The objective could differ quantitatively (e.g. different time scales τ_{tr} of the complex cells at different layers in the network) or even qualitatively (e.g. presenting different transformations such as translations, rotations and zoom-ins, for training different layers of complex cells).

- **Evaluation** In this work, we looked at linear classification to evaluate the quality of deep representations. The interpretation of receptive fields or representations of deep hidden layers is known to be difficult (Hubel and Wiesel, 1962; Yosinski et al., 2015). There are known methods to visualise the representations deep inside neural networks, through feature visualisation (e.g. Yosinski et al. (2015)), low-dimensional embedding (e.g. Van der Maaten and Hinton (2008) and chapter 4) or probing preferred stimuli of single units (e.g. Löwe et al. (2019) and chapter 4), but all such methods are qualitative and ambiguous. We thus chose linear classification as our preferred method to evaluate deep representations, since it is quantitative and similar to common downstream tasks for which deep networks are often used. However, this method is blind to other features

of hierarchy, e.g. to the iterative composition of features, and future efforts should be put on finding better evaluation methods. A promising candidate is to compare learned deep representations to cortical representations from brain recordings (Yamins and DiCarlo, 2016; Zhuang et al., 2021).

- **Theory** A promising way to target the said conceptual problems is to gain theoretical insight into the deep representations we want to learn. The idea is an 'analysis by synthesis' by proposing a hierarchical generative model, that could underlie realistic data such as images, and then deriving dynamics and learning rules from it (Mumford, 2002). A non-hierarchical example of such a generative model is the linear reconstruction of sparse coding Olshausen and Field (1996) which leads to simple cell properties. The generalisation to hierarchical multi-layer models is a very promising, yet somehow stagnating field (Hurri and Hyvärinen, 2003; Hosoya, 2012; Patel and Baraniuk, 2016; Chen et al., 2018).

3.5 Supplemental information

3.5.1 Towards hierarchical Hebbian learning in CNNs trained on videos of whitened natural images patches

For this experiment, we choose the same data set of natural image patch videos (derived from Olshausen and Field (1997)) as in subsection 3.3.1, except using a bigger patch size to account for the larger receptive field of the whole convolutional network. Evaluation is still done via classification on the train and test set of the STL-10 data set. The results are summarized by the purple curves in Figure 3.7. All other curves are identical to Figure 3.6

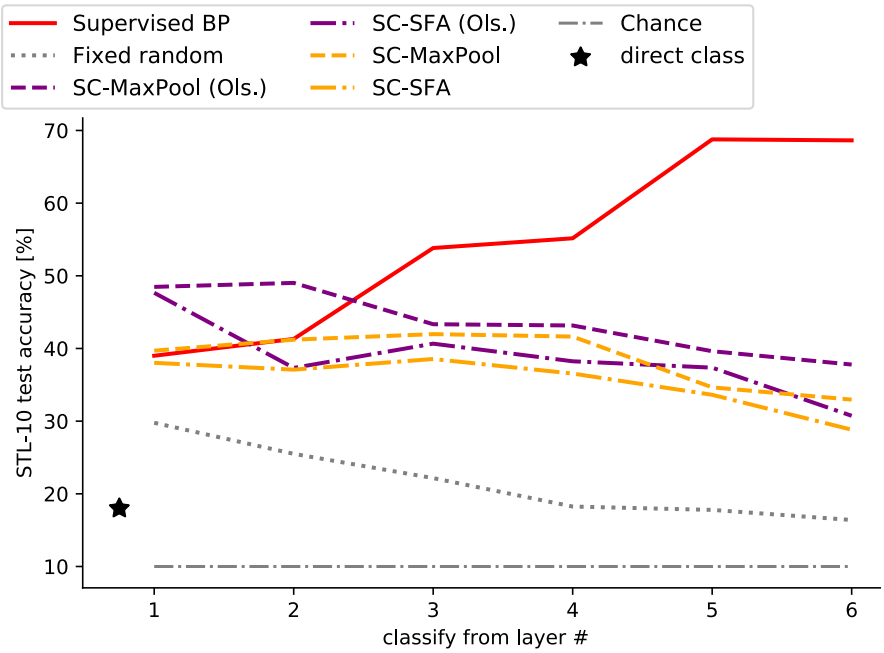


Figure 3.7: Evaluation of hierarchical competitive temporal Hebbian learning on STL-10 classification (as in Figure 3.6), but including curves for SC-MaxPool and SC-SFA trained on patch videos extracted from the dataset provided by Olshausen and Field (1996) (purple curves and the appendix '(Ols.)'). We evaluate the quality of deep layers' representations by measuring test accuracy of a linear classification on STL-10 on this layer. Compared to chance level (dash-dotted), direct classification on pixels (black star) and a fixed random network (dotted), hierarchical temporal Hebbian learning (SC-SFA), as well as the control SC-MaxPool, yield useful representations. However, it does *not* lead to hierarchical representations as performance does not increase with depth. The red line for supervised backpropagation (BP) shows that the architecture can support hierarchical representations.

3.5.2 k-winners-take-all competition

The convergence time to reach the fixed point was found to grow super-linearly with the number of neurons, which causes (i) a practical problem for simulating such circuits, and (ii)

a conceptual problem for a model of the cortex: long waiting times to reach a stable percept are prohibitive for a real biological agent. To solve the first issue, we also consider a simple k-winner-take-all (kWTA) circuit to implement competition, which algorithmically selects the k neurons with highest feedforward activity, in the first time step after the output has been presented, to stay active and shuts down all others:

$$\begin{aligned} \mathbf{u}(t) &= \mathbf{W} \cdot \mathbf{I} - \mathbf{b} \\ a_i(t) &= \begin{cases} u_i(t) & \text{if } i \in \text{k-argmax}(\mathbf{u}(t)) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3.7}$$

This alternative to Equation 3.1 solves problem (i) above, because the competition can be evaluated in a single time step, assuming that we run an efficient sorting algorithm within a single time step. Concerning problem (ii), we note that k-WTA competition can be implemented as a dynamical system with a single inhibitory neuron (Majani et al., 1989), which makes it a valid biological model. It remains to be investigated whether, in practice, the dynamics of the kWTA circuit of Equation 3.7 converge faster, i.e. after fewer time steps, compared to Equation 3.1. In the basic 2-layer network, we found both types of competition, Equation 3.1 and Equation 3.7, leading to qualitatively similar results in terms of neural activities and receptive field learning. In the deep convolutional setup, kWTA dynamics lead to qualitatively similar results as the original dynamics, i.e. non-hierarchical representations. However, for kWTA dynamics, the classification performance dropped significantly faster for higher layers than with standard dynamics, see Figure 3.8.

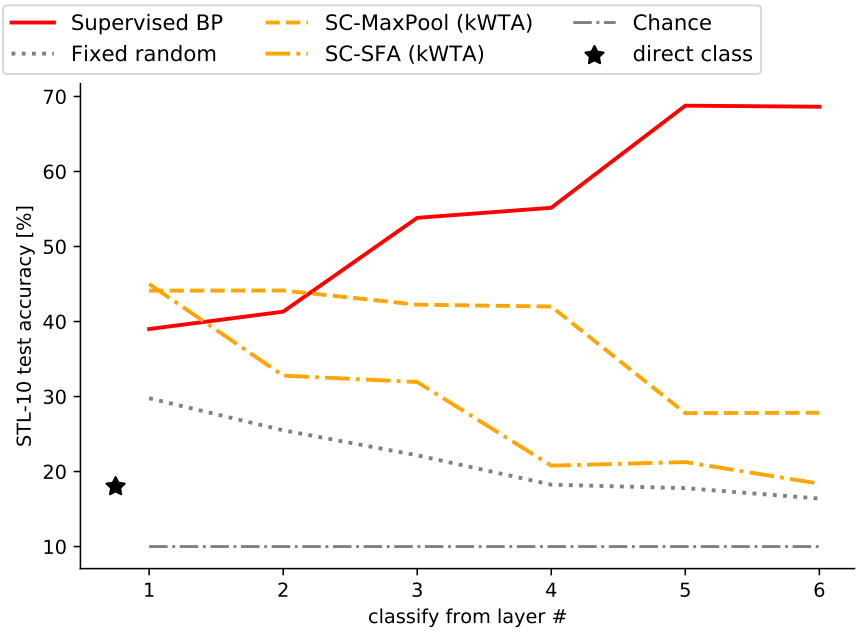


Figure 3.8: Evaluation of hierarchical competitive temporal Hebbian learning on STL-10 classification (as in Figure 3.6), but using kWTA dynamics as in Equation 3.7 for SC-MaxPool and SC-SFA. We evaluate the quality of deep layers' representations by measuring test accuracy of a linear classification on STL-10 on this layer. Compared to chance level (dash-dotted), direct classification on pixels (black star) and a fixed random network (dotted), hierarchical temporal Hebbian learning (SC-SFA), as well as the control SC-MaxPool, yield useful representations. However, it does *not* lead to hierarchical representations as performance does not increase with depth. The red line for supervised backpropagation (BP) shows that the architecture can support hierarchical representations.

4 Local plasticity rules can learn deep representations using self-supervised contrastive predictions

Paper information

Authors **Bernd Illing**, Jean Ventura, Guillaume Bellec, Wulfram Gerstner

Abstract Learning in the brain is poorly understood and learning rules that respect biological constraints, yet yield deep hierarchical representations, are still unknown. Here, we propose a learning rule that takes inspiration from neuroscience and recent advances in self-supervised deep learning. Learning minimizes a simple layer-specific loss function and does not need to back-propagate error signals within or between layers. Instead, weight updates follow a local, Hebbian, learning rule that only depends on pre- and post-synaptic neuronal activity, predictive dendritic input and widely broadcasted modulation factors which are identical for large groups of neurons. The learning rule applies contrastive predictive learning to a causal, biological setting using saccades (i.e. rapid shifts in gaze direction). We find that networks trained with this self-supervised and local rule build deep hierarchical representations of images, speech and video.

Keywords Hebbian learning, deep learning, unsupervised learning, contrastive predictive coding, saccades, dendrites, STL-10, LibriSpeech, UCF-101

Author contributions The project was designed by GB and BI. Theoretical derivations were done by GB and BI, simulations by BI, GB and JV. The paper was written by BI, GB and WG.

Publication This chapter is accepted as a conference paper at NeurIPS 2021; arXiv version: (Illing et al., 2020)

Funding This research was supported by the Swiss National Science Foundation (no. 200020_184615), by the Intel Neuromorphic Research Lab and by the European Union Horizon 2020 Framework Program under grant agreement no. 785907 (Human Brain Project, SGA2).

4.1 Introduction

Synaptic connection strengths in the brain are thought to change according to 'Hebbian' plasticity rules (Hebb, 1949). Such rules are local and depend only on the recent state of the pre- and post-synaptic neurons (Sjöström et al., 2001; Caporale and Dan, 2008; Markram et al., 2011), potentially modulated by a third factor related to reward, attention or other high-level signals (Kuśmierz et al., 2017; Gerstner et al., 2018). Therefore, one appealing hypothesis is that *representation learning in sensory cortices emerges from local and unsupervised plasticity rules*.

Following a common definition in the field (Fukushima, 1988; Riesenhuber and Poggio, 1999; LeCun, 2012; Lillicrap et al., 2020), a hierarchical representation (i) builds higher-level features out of lower-level ones, and (ii) provides more useful features in higher layers. Now there seems to be a substantial gap between the rich hierarchical representations observed in the cortex and the representations emerging from local plasticity rules implementing principal/independent component analysis (Oja, 1982; Hyvärinen and Oja, 1998), sparse coding (Olshausen and Field, 1997; Rozell et al., 2008; Kohonen, 2012) or slow-feature analysis (Földiák, 1991; Wiskott and Sejnowski, 2002; Sprekeler et al., 2007). Hebbian rules seem to struggle especially when 'stacked', i.e. when asked to learn deep, hierarchical representations.

This performance gap is puzzling because there are learning rules, relying on back-propagation (BP), that *can* build hierarchical representations similar to those found in visual cortex (Yamins et al., 2014; Zhuang et al., 2021). Although some progress towards biologically plausible implementations of back-propagation has been made (Lillicrap et al., 2016; Guerguiev et al., 2017; Sacramento et al., 2018; Payeur et al., 2021), most models rely either on a neuron-specific error signal that needs to be transmitted by a separate error network (Crick, 1989; Amit, 2019; Kunin et al., 2020), or time-multiplexing feedforward and error signals (Lillicrap et al., 2020; Payeur et al., 2021). Algorithms like contrastive divergence (Hinton, 2002), contrastive Hebbian learning (Xie and Seung, 2003) or equilibrium propagation (Scellier and Bengio, 2017) use local activity exclusively to calculate updates, but they require to wait for convergence to an equilibrium which is not appropriate for online learning from quickly varying inputs.

The present paper demonstrates that deep representations can emerge from a local, biologically plausible and unsupervised learning rule, by integrating two important insights from neuroscience: First, we focus on self-supervised learning from temporal data – as opposed to supervised learning from labelled examples – because this comes closest to natural data, perceived by real biological agents, and because the temporal structure of natural stimuli is a rich source of information. In particular, we exploit the typical, self-generated changes of gaze direction ('*saccades*') to distinguish input from a moving object during fixation from input arriving after a saccade towards a new object. Second, we notice that electrical signals stemming from segregated apical dendrites can modulate synaptic plasticity in biological neurons (Major et al., 2013; Körding and König, 2001). Our model still relies on some multiplexing, however, and opposed to models of biologically plausible backpropagation (Lillicrap et al.,

2020), multiplexing is only used to calculate a scalar, layer-wide plasticity modulator within layers.

Algorithmically, our approach takes inspiration from deep self-supervised learning algorithms that seek to contrast, cluster or predict stimuli in the context of BP (Van den Oord et al., 2018; Caron et al., 2018; Zhuang et al., 2019; Löwe et al., 2019). Interestingly, Löwe et al. (2019) demonstrated that such methods even work if end-to-end BP is partially interrupted. We build upon this body of work and suggest the *Contrastive, Local And Predictive Plasticity* (CLAPP) model which avoids BP completely, yet still builds hierarchical representations.[1]

4.2 Main goals and related work

In this paper, we propose a local plasticity rule that learns deep representations. To describe our model of synaptic plasticity, we represent a cortical area by the layer l of a deep neural network. The neural activity of this layer at time t is represented by the vector $\boldsymbol{z}^{t,l} = \rho(\boldsymbol{a}^{t,l})$, where ρ is a non-linearity and $\boldsymbol{a}^{t,l} = \boldsymbol{W}^l \boldsymbol{z}^{t,l-1}$ is the vector of the respective summed inputs to the neurons through their basal dendrites $\boldsymbol{W}^l$ (the bias is absorbed into $\boldsymbol{W}^l$). To simplify notation, we write the pre-synaptic input as $\boldsymbol{x}^{t,l} = \boldsymbol{z}^{t,l-1}$ and we only specify the layer index l when it is necessary.

Our plasticity rule exploits the fact that the temporal structure of natural inputs affects representation learning (Li and DiCarlo, 2008). Specifically, we consider a scenario where an agent first perceives a moving object at time t (e.g. a flying eagle in Figure 4.1 a), and then spontaneously decides to change gaze direction towards another moving object at time $t + \delta t$ (e.g. *saccade* towards the elephant in Figure 4.1 a). We further assume that the visual pathway is 'self-aware' of saccades due to saccade-specific modulation of processing (Ross et al., 2001).

Following classical models of synaptic plasticity, we assume that a weight change is implemented by a biologically plausible, *Hebbian*-like learning rule (Hebb, 1949; Markram et al., 2011) which is local in space and time: updates ΔW_{ji}^t of a synapse, connecting neurons i and j, can only depend on the current activity of the pre-synaptic and post-synaptic neurons at time t, or slightly earlier at time $t - \delta t$, and one or several widely broadcasted modulating factors (Urbanczik and Senn, 2009; Gerstner et al., 2018).

Furthermore, we allow the activity of another neuron k to influence the weight update ΔW_{ji}, as long as there is an explicit connection W_{jk}^{pred} from k to j. The idea is to overcome the representational limitations of classical Hebbian learning by including dendritic inputs, which are thought to predict the future somatic activity (Körding and König, 2001; Urbanczik and Senn, 2014) and take part in the plasticity of the post-synaptic neuron (Larkum et al., 1999; Dudman et al., 2007; Major et al., 2013). Hence we assume that each neuron j in a layer l may receive dendritic inputs $(\boldsymbol{W}^{\text{pred}} \boldsymbol{c}^{t,l})_j$ coming either from the layer above ($\boldsymbol{c}^{t,l} = \boldsymbol{z}^{t,l+1}$) or from lateral connections in the same layer ($\boldsymbol{c}^{t,l} = \boldsymbol{z}^{t,l}$).

[1] Our code is available at https://github.com/EPFL-LCN/pub-illing2021-neurips

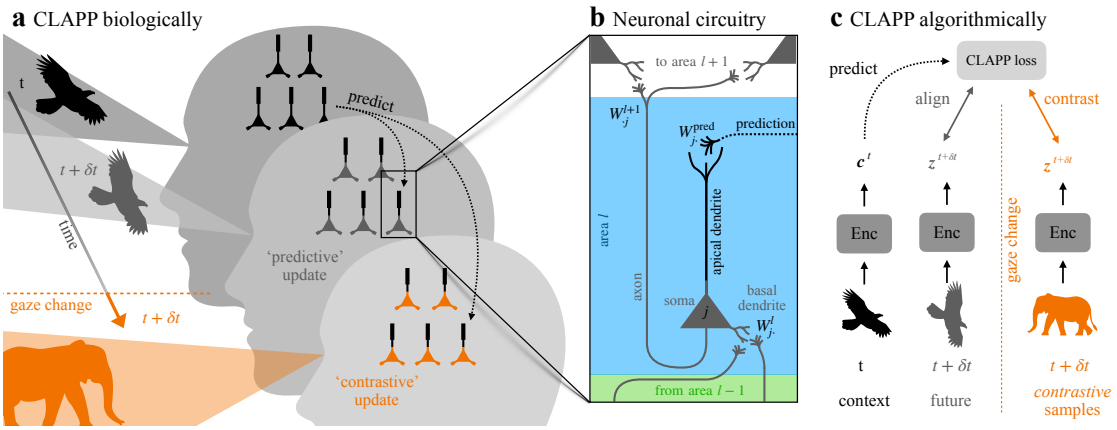


Figure 4.1: Contrastive, local and predictive plasticity (CLAPP). **a** Perceiving a moving object (e.g. an eagle) at times t and $t+\delta t$ leads to neural responses in the visual cortex. After a gaze change ('saccade'), a different object (elephant) is seen. **b** (zoom) At each time step, pyramidal neurons integrate input activity at the basal dendrites (matrix $\boldsymbol{W}^l$ of feedforward weights) and pass on their response to downstream areas ($\boldsymbol{W}^{l+1}$). At any point in time, neurons predict future neural responses through recurrent connections $\boldsymbol{W}^{\text{pred}}$. These inputs target the apical dendrites and modulate ongoing synaptic plasticity through 'predictive' updates. Information about a saccade is transmitted by a broadcast signal triggered at the moment of saccade initiation, which leads to 'contrastive' updates. As no external supervision or reward signals are provided, learning is self-supervised and local in time and space ('Hebbian'). **c** Algorithmically, an encoder network (Enc) produces a 'context' representation $\boldsymbol{c}^t$ at time t. Given $\boldsymbol{c}^t$, CLAPP tries to *predict* the encoding of the future input $\boldsymbol{z}^{t+\delta t}$. In case of a gaze change between t and $t+\delta t$, CLAPP seeks to keep the prediction as different as possible from the encoding of the upcoming *contrastive* sample.

For algorithmic reasons, that we detail in section 4.3, we assume that the dendritic input $(\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^{t,l})_j$ influences the weight updates ΔW_{ji} of the post-synaptic neuron j, but not its activity z_j^t. This assumption is justified by neuroscientific findings that the inputs to basal and apical dendrites affect the neural activity and plasticity in different ways (Larkum et al., 1999; Dudman et al., 2007; Major et al., 2013; Urbanczik and Senn, 2014). In general, we do not rule out influence of dendritic activity on somatic activity in later phases of cortical processing, but see this beyond the scope of the current work.

Given these insights from neuroscience, we gather the essential factors that influence synaptic plasticity in the following learning rule prototype:

$$\Delta W_{ji} \propto \underbrace{\text{modulators}}_{\text{broadcast factors}} \cdot \underbrace{(\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^{t_1})_j}_{\text{dendritic prediction}} \underbrace{\cdot \text{post}_j^{t_2} \cdot \text{pre}_i^{t_2}}_{\text{local-activity}} . \tag{4.1}$$

The modulating broadcast factors are the same for large groups of neurons, for example all neurons in the same area, or even all neurons in the whole network. $\text{post}_j^{t_2}$ and $\text{pre}_i^{t_2}$ are functions of the pre- and post- synaptic activities. At this point, we do not specify the exact timing between t_1 and t_2, as this will be determined by our algorithm in section 4.3.

Related work Many recent models of synaptic plasticity fit an apparently similar learning rule prototype (Lillicrap et al., 2016; Nøkland, 2016; Roelfsema and Holtmaat, 2018; Nøkland and Eidnes, 2019; Lillicrap et al., 2020; Pozzi et al., 2020) if we interpret the top-down signals emerging from the BP algorithm as the dendritic signal. However, top-down error signals in BP are not directly related to the activity $\boldsymbol{c}^t$ of the neurons in the main network during processing of sensory input. Rather, they require a separate linear network mirroring the initial network and feeding back error signals (see Figure 4.2 a and Lillicrap et al. (2020)), or involved time-multiplexing of feedforward and error signals in the main network (Lillicrap et al., 2020; Payeur et al., 2021). Our model is fundamentally different, because in our case, the dendritic signal onto neuron j is strictly $(\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^t)_j$ which is a weighted sum of the main network activity and there is no need of a (linear) feedback network transmitting exact error values across many layers. In our model, some multiplexing is still needed, however, this multiplexing happens within single layers and is only used to calculate a scalar, layer-wide modulator.

Moreover, we show in simulations in section 4.4, that the dendritic signal does not have to come from a layer above but that the prediction fed to layer l may come from the same layer. This shows that our learning rule works even in the complete absence of downward signaling from $l+1$ to l. This last point is a significant difference to other methods that also calculate updates using only activities of the main network, but require tuned top-down connections to propagate signals downwards in the network hierarchy (Kunin et al., 2020), such as methods in the difference target propagation family (Lee et al., 2015; Bartunov et al., 2018; Golkar et al., 2020), contrastive divergence (Hinton, 2002) and equilibrium propagation (Scellier and Bengio, 2017). Furthermore, the latter two require convergence to an equilibrium state for each input (Laborieux et al., 2021). Our model does not require this convergence because it uses the recurrent dendritic signal $(\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^t)_j$ only for synaptic plasticity and not for inference.

Most previous learning rules which include global modulating factors interpret it as a reward prediction error (Schultz et al., 1997; Gerstner et al., 2018; Pozzi et al., 2020). In this paper, we address self-supervised learning and view global modulating factors as broadcasting signals, modeling the self-awareness that something has changed in the stimulus (e.g. because of a saccade). Hence, the main function of the broadcast factor in our model is to identify contrastive inputs, which avoids a common pitfall for self-supervised learning models: 'trivial' or 'collapsed' solutions, where the model produces a constant output, which is easily predictable, but useless for downstream tasks. In vision, we use a broadcast factor to model the strong, saccade-specific activity patterns identified throughout the visual pathway (Kowler et al., 1995; Leopold and Logothetis, 1998; Ross et al., 2001; McFarland et al., 2015). In other sensory pathways, like audition, this broadcast factor may model attention signals arising when changing focus on a new input source (Fritz et al., 2007), cross-modal input indicating a change in head or gaze direction, or signal/speaker-identity inferred from blind source separation, which can be done on low-level representation with biologically plausible learning rules (Hyvärinen and Oja, 1997; Ziehe and Müller, 1998; Molgedey and Schuster, 1994).

Our theory takes inspiration from the substantial progress seen in unsupervised machine learning in recent years and specifically from contrastive predictive coding (CPC) (Van den Oord et al., 2018). CPC trains a network (called *encoder*) to make *predictions* of its own responses to future inputs, while keeping this prediction as different as possible to its responses to *fake* inputs (*contrasting*). A key feature of CPC is that predicting and contrasting happens in latent space, i.e. on the output representation of the encoder network. This avoids modeling a generative model for perfect reconstruction of the input and all its details (e.g. green, spiky). Instead the model is forced to focus on extracting high-level information (e.g. cactus). In our notation, CPC evaluates a prediction $\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^t$ such that a score function $u_t^\tau = \boldsymbol{z}^{\tau\top}\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^t$ becomes larger for the true future $\tau = t + \delta t$ (referred to as positive sample) than for any other vector $\boldsymbol{z}^{t'}$ taken at arbitrary time points t' elsewhere in the entire training set (referred to as negative samples in CPC). This means, that the prediction should align with the future activity $\boldsymbol{z}^{t+\delta t}$ but not with the negative samples. Van den Oord et al. (2018) formalizes this as a softmax cross-entropy classification, which leads to the traditional CPC loss:

$$\mathscr{L}_{\text{CPC}}^t = -\log \frac{\exp u_t^{t+\delta t}}{\sum_{\tau\in\mathscr{T}} \exp u_t^\tau} \,, \tag{4.2}$$

where $\mathscr{T} = \{t^{t+\delta t}, t'_1 \dots t'_N\}$ comprises the positive sample and N negative samples. The learned model parameters are the elements of the matrix $\boldsymbol{W}^{\text{pred}}$, as well as the weights of the encoder network. The loss function $\mathscr{L}_{\text{CPC}}^t$ is then minimized by stochastic gradient descent on these parameters using BP. Amongst numerous recent variants of contrastive learning (He et al., 2019; Chen et al., 2020; Xiong et al., 2020), we focus here on CPC (Van den Oord et al., 2018), for which a more local variant, Greedy InfoMax, was recently proposed by Löwe et al. (2019).

Greedy InfoMax (GIM) (Löwe et al., 2019) is a variant of CPC which makes a step towards local, BP-free learning: the main idea is to split the encoder network into a few gradient-isolated modules to avoid back-propagation between these modules. As the authors mention in their conclusion, "*the biological plausibility of GIM is limited by the use of negative samples and within-module back-propagation*". This within-module back-propagation still requires a separate feedback network to propagate prediction errors (Figure 4.2 a), but can be avoided in the most extreme version of GIM, where each gradient-isolated module contains a single layer (*layer-wise GIM*). However, the gradients of layer-wise GIM, derived from Equation 4.2, still cannot be interpreted as synaptic plasticity rules because the gradient computation requires (1) the transmission of information other than the network activity (see Figure 4.2 b), and (2) perfect memory to replay the negative samples $\boldsymbol{z}^{t'}$, as mentioned in the above quote (see subsection 4.6.1 for details). Overall it is not clear how this weight update of layer-wise GIM could be implemented with realistic neuronal circuits. The CLAPP rule, proposed in this paper, solves the above mentioned implausibilities and allows a truly local implementation in space and time.

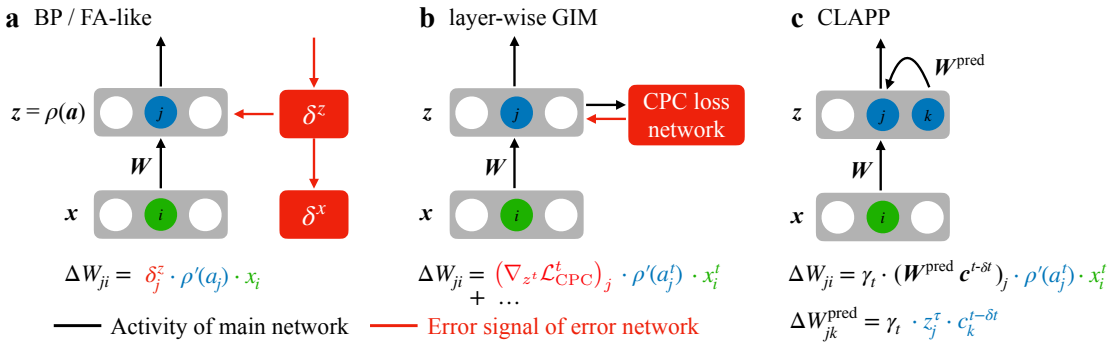


Figure 4.2: Comparison of weight updates **a** Networks trained with back-propagation (BP) or Feedback Alignment (FA)-like methods require separate error networks (red) for computing weight updates. **b** Layer-wise GIM, with one layer per gradient-isolated module, does not transmit error signals across layers (i.e. modules) but requires (1) the transmission of information other than the network activity (red) and (2) a perfect replay of negative samples. Thus, the resulting update computation needs a separate loss network and cannot be interpreted as a local learning rule. **c** Contrastive Local and Predictive Plasticity (CLAPP) calculates updates using locally and temporally available information: pre- and post-synaptic activity and predictive recurrent input onto the apical dendrite $\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^{t-\delta t}$. Global broadcasting factors γ_t modulate plasticity depending on the presence or absence of a saccade.

4.3 Derivation of the CLAPP rule: contrastive, local and predictive plasticity

We now suggest a simpler contrastive learning algorithm which solves the issues encountered with layer-wise GIM and for which a gradient descent update is naturally compatible with the learning rule prototype from Equation 4.1. The most essential difference compared to CPC or GIM is, that we do not require the network to simultaneously access the true future activity $\boldsymbol{z}^{t+\delta t}$ and recall (or imagine) the network activity $\boldsymbol{z}^{t'}$ seen at some other time. Rather, we consider the naturalistic time-flow illustrated in Figure 4.1 a, where an agent fixates on a moving animal for a while and then changes gaze spontaneously. In this way, the prediction $\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^{t}$ is expected to be meaningful during fixation, but inappropriate right after a saccade. In our simulations, we model this by feeding the network with subsequent frames from the same sample (e.g. different views of an eagle), and then abruptly changing to frames from another sample (e.g. different views of an elephant).

We note that the future activity $\boldsymbol{z}^{t+\delta t}$ and the context $\boldsymbol{c}^{t}$ are *always* taken from the main feedforward encoder network. We focus on the case where the context stems from the same layer as the future activity ($\boldsymbol{c}^{t,l} = \boldsymbol{z}^{t,l}$), however, the model allows for the more general case, where the context stems from another layer (e.g. the layer above $\boldsymbol{c}^{t,l} = \boldsymbol{z}^{t,l+1}$).

Derivation of the CLAPP rule from a self-supervised learning principle Rather than using a global loss function for multi-class classification to separate the true future from multiple negative samples, as in CPC, we consider here a binary classification problem at every layer l:

we interpret the score function $u_t^{t+\delta t,l} = \boldsymbol{z}^{t+\delta t,l\top} \boldsymbol{W}^{\text{pred},l} \boldsymbol{c}^{t,l}$ as the layer's 'guess' whether the agent performed a fixation or a saccade. In subsection 4.6.3, we discuss how $u_t^{t+\delta t,l}$ could be (approximately) computed in real neuronal circuits. In short, every neuron i has access to its 'own' dendritic prediction $\hat{z}_i^{t,l} = \sum_j W_{ij}^{\text{pred},l} c_j^{t,l}$ of somatic activity (Urbanczik and Senn, 2014), and the product $z_i^{t+\delta t,l}\, \hat{z}_i^{t,l}$ can be seen as a coincidence detector of dendritic and somatic activity, communicated by specific burst signals (Larkum et al., 1999). These burst signals allow time-multiplexed communication (Payeur et al., 2021) of the products $z_i^{t+\delta t,l}\, \hat{z}_i^{t,l}$ of many neurons, which can then be summed by an interneuron representing $u_t^{t+\delta t,l}$.

As mentioned in section 4.2, information about the presence or absence of a saccade between two time points is available in the visual processing stream and is modeled here by the variable $y^t = -1$ and $y^t = +1$, respectively. We interpret y^t as the label of a binary classification problem, characterized by the Hinge loss, and define the CLAPP loss at layer l as:

$$\mathscr{L}_{CLAPP}^{t,l} = \max\left(0, 1 - y^t \cdot u_t^{t+\delta t,l}\right) \text{ with } \begin{cases} y^t = +1 & \text{for fixation} \\ y^t = -1 & \text{for saccade} \end{cases} \tag{4.3}$$

We now derive the gradients of Equation 4.3 with respect to the feedforward weights and show that gradient descent on this loss function is compatible with the learning rule prototype suggested in Equation 4.1. Note that CLAPP optimises Equation 4.3 for each layer l independently, without any gradient flow between layers. That being said, the following derivation is the same for every layer l, which is why we omit the layer index l from here on.

Since we chose to formalize the binary classification with a Hinge loss, the gradient vanishes when the classification is already correct: high score $u_t^{t+\delta t} > 1$ during fixation ($y^t = +1$), or a low score $u_t^{t+\delta t} < -1$ after a saccade ($y^t = -1$). Otherwise, it is $-\nabla u_t^{t+\delta t}$ during a fixation or $\nabla u_t^{t+\delta t}$ after a saccade. In the 'predicted layer' $\boldsymbol{z}$, i.e. the target of the prediction, let W_{ji} denote the feedforward weight from neuron i in the previous layer (with activity x_i^t) to neuron j, with summed input a_j^t and activity z_j^t. Similarly, in the 'predicting layer' $\boldsymbol{c}$, i.e. the source of the prediction, let $W_{kl}^{\boldsymbol{c}}$ denote the feedforward weight between the neuron l in the previous layer (with activity $x_l^{\boldsymbol{c},t}$) and neuron k, with summed input $a_k^{\boldsymbol{c},t}$ and activity c_k^t. Therefore, $\cdot^{\boldsymbol{c}}$ as an upper index refers to the context layer, whereas $\boldsymbol{c}$ as a full-size letter refers to the respective neuronal activity. We then find the gradients with respect to these weights as:

$$\frac{\partial \mathscr{L}_{CLAPP}^t}{\partial W_{ji}} = \pm(\boldsymbol{W}^{\text{pred}} \boldsymbol{c}^t)_j\, \rho'(a_j^{t+\delta t})\, x_i^{t+\delta t} \tag{4.4}$$

$$\frac{\partial \mathscr{L}_{CLAPP}^t}{\partial W_{km}^{\boldsymbol{c}}} = \pm(\boldsymbol{W}^{\text{pred}\top} \boldsymbol{z}^{t+\delta t})_k\, \rho'(a_k^{\boldsymbol{c},t})\, x_m^{\boldsymbol{c},t}\,, \tag{4.5}$$

where the sign is negative during fixation and positive after a saccade. To change these equa-

tions into online weight updates, we consider the gradient descent update delayed by δt, such that $\Delta W_{ji}^t = -\eta \frac{\partial \mathscr{L}_{CLAPP}^{t-\delta t}}{\partial W_{ji}}$, where η is the learning rate. Let us define a modulating factor $\gamma_t = y^t \cdot H^t$, where $y^t = \pm 1$ is a network-wide broadcast signal (self-awareness) indicating a saccade (-1) or a fixation ($+1$) and $H^t \in \{0, \eta\}$ is a layer-wide broadcast signal indicating whether the saccade or fixation was correctly classified as such. In this way, Equation 4.4 becomes a weight update which follows strictly the ideal learning rule prototype from Equation 4.1:

$$\Delta W_{ji}^t = \underbrace{\gamma_t}_{\text{broadcast factors}} \cdot \underbrace{(\boldsymbol{W}^{\text{pred}} \boldsymbol{c}^{t-\delta t})_j}_{\text{dendritic prediction}} \cdot \underbrace{\rho'(a_j^t) x_i^t}_{\text{local activity}} . \tag{4.6}$$

For the updates of the connections onto the neuron c_k^t, which emits the prediction rather than receiving it, our theory in Equation 4.5 requires the opposite temporal order and the transmission of the information in the opposite direction: from $\boldsymbol{z}^t$ back to $\boldsymbol{c}^t$. Since connections in the brain are unidirectional (Lillicrap et al., 2016), we introduce another matrix $\boldsymbol{W}^{\text{retro}}$ which replaces $\boldsymbol{W}^{\text{pred}\top}$ in the final weight update. Given the inverse temporal order, we interpret $\boldsymbol{W}^{\text{retro}} \boldsymbol{z}^t$ as a retrodiction rather than a prediction. In subsection 4.6.3, we show that using $\boldsymbol{W}^{\text{retro}}$ minimises a loss function of the same form as Equation 4.3, and empirically performs as well as using $\boldsymbol{W}^{\text{pred}\top}$. The resulting weight update satisfies the learning rule prototype from Equation 4.1, as it can be written:

$$\Delta W_{km}^{\boldsymbol{c},t} = \underbrace{\gamma_t}_{\text{broadcast factors}} \cdot \underbrace{(\boldsymbol{W}^{\text{retro}} \boldsymbol{z}^t)_k}_{\text{dendritic retrodiction}} \cdot \underbrace{\rho'(a_k^{\boldsymbol{c},t-\delta t}) x_m^{\boldsymbol{c},t-\delta t}}_{\text{local activity}} . \tag{4.7}$$

In the (standard) case, where context and predicted activity are from the same layer ($\boldsymbol{c}^{t,l} = \boldsymbol{z}^{t,l}$), $\boldsymbol{W}$ and $\boldsymbol{W}^{\boldsymbol{c}}$ are the same weights and the updates Equation 4.6 and Equation 4.7 are added up linearly.

The prediction and retrodiction weights, $\boldsymbol{W}^{\text{pred}}$ and $\boldsymbol{W}^{\text{retro}}$, respectively, are also plastic. By deriving the gradients of $\mathscr{L}_{CLAPP}^t$ with respect to $\boldsymbol{W}^{\text{pred}}$, we find an even simpler Hebbian learning rule for these weights:

$$\Delta W_{jk}^{\text{pred}} = \Delta W_{kj}^{\text{retro}} = \underbrace{\gamma_t}_{\text{broadcast factors}} \cdot \underbrace{z_j^t \cdot c_k^{t-\delta t}}_{\text{pre and post}} , \tag{4.8}$$

where neuron k in the predicting layer $\boldsymbol{c}$ is pre-synaptic (post-synaptic) and neuron j in the predicted layer $\boldsymbol{z}$ is post-synaptic (pre-synaptic) for the prediction weights W_{jk}^{pred} (retrodiction weights W_{kj}^{retro}). Note that the update rules for W_{jk}^{pred} and W_{kj}^{retro} are reciprocal, a method that leads to mirrored connections, given small enough initialisation (Burbank, 2015; Amit, 2019; Pozzi et al., 2020).

We emphasize that all information needed to calculate the above CLAPP updates (Equations 4.6 – 4.8) is spatially and temporally available, either as neuronal activity at time t, or as traces

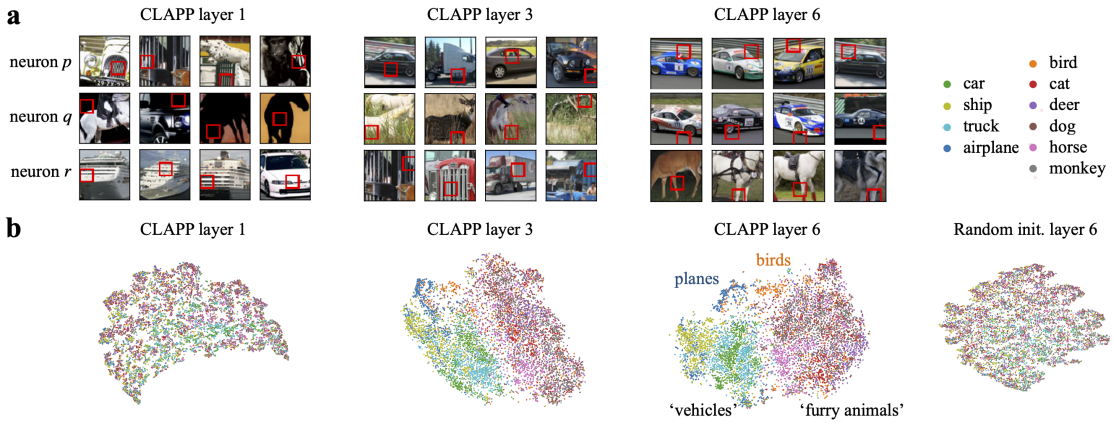


Figure 4.3: Hierarchical representations learned by CLAPP. **a** Red boxes in STL-10 images indicate patches that best activate a specific neuron (rows) in a network trained with CLAPP. Layer 1 extracts simple features like gratings or uniform patches, higher layers extract richer features like parts of objects. **b** 2-dimensional t-SNE projection of neuronal activities at different layers unveils increasing representational structure in higher layers (every dot represents one input image). Note that CLAPP has not seen any class labels during training.

of recent activity $(t - \delta t)$ (Gerstner et al., 2018). In order to implement Equation 4.6, the dendritic prediction has to be retained during δt. However, we argue that dendritic activity can outlast (50-100 ms) somatic neuronal activity (2-10 ms) (Major et al., 2013), which makes predictive input from several time steps in the past $(t - \delta t)$ available at time t.

Generalizations While the above derivation considers fully-connected feedforward networks, we apply analogous learning rules to convolutional neural networks (CNN) and recurrent neural networks (RNN). Analyzing the biological plausibility of the standard spatial weight sharing and spatial MeanPooling operations in CNNs is beyond the scope of the current work. Furthermore, we allow gradient flow through *single* MaxPooling layers, as discussed in subsection 4.6.3.

To obtain local learning rules even for RNNs, we combine CLAPP with the e-prop theory (Bellec et al., 2020), which provides a biologically plausible alternative to BP through time: gradients can be propagated forward in time through the intrinsic neural dynamics of a neuron using eligibility traces. The propagation of gradients across recurrently connected units is forbidden and disabled. This yields a biologically plausible and self-supervised plasticity update in GRU units, as explained in subsection 4.6.3.

4.4 Empirical results

Building hierarchical representations We first demonstrate numerically, that CLAPP yields deep hierarchical representations, despite using a local plasticity rule compatible with Equation 4.1. We report here the results for $\boldsymbol{c}^{t,l} = \boldsymbol{z}^{t,l}$, i.e. the dendritic prediction in Equation 4.1

is generated from lateral connections and the representations in the same layer. We note, however, that we obtained qualitatively similar results with $\boldsymbol{c}^{t,l} = \boldsymbol{z}^{t,l+1}$ (i.e. the dendritic prediction is generated from one layer above), suggesting that top-down signaling is neither necessary for, nor incompatible with, our algorithm (also see subsection 4.6.3).

We first consider the STL-10 image dataset (Coates et al., 2011). To simulate a time dimension in these static images, we follow Hénaff et al. (2019) and Löwe et al. (2019): each image is split into 16×16 patches and the patches are viewed one after the other in a vertical order (one time step is one patch). Other hyper-parameters and data-augmentation are taken from Löwe et al. (2019), see subsection 4.6.2. We then train a 6-layer VGG-like (Simonyan and Zisserman, 2015) encoder (VGG-6) using the CLAPP rule (Equations 4.6 – 4.8). Training is performed on the unlabelled part of the STL-10 dataset for 300 epochs. We use 4 GPUs (`NVIDIA Tesla V100-SXM2 32 GB`) for data-parallel training, resulting in a simulation time of around 4 days per run.

In order to study how neuronal selectivity changes over layers, we select neurons randomly and show image patches which best activate these neurons the most (rows in Figure 4.3 a). As expected for a visual hierarchy, first-layer neurons (first column in Figure 4.3 a) are selective to horizontal or vertical gratings, or homogeneous colors. In the third layer of the network (second column), neurons start to be selective to more semantic features like grass, or parts of vehicles. Neurons in the last layer (third column) are selective to specific object parts (e.g. a wheel touching the road). The same analysis for a random, untrained encoder does not reveal a clear hierarchy across layers, see subsection 4.6.3.

To get a qualitative idea of the learned representation manifold, we use the non-linear dimension reduction technique t-SNE (Van der Maaten and Hinton, 2008) to visualise the encodings of the (labeled) STL-10 test set in Figure 4.3 b. We see that the representation in the first layer is mostly unrelated to the underlying class. In the third and sixth layers' representation, a coherent clustering emerges, yielding an almost perfect separation between furry animals and vehicles. This clustered representation is remarkable since the network has never seen class labels, and was never instructed to separate classes, during CLAPP training The representation of the same architecture, but without training (Random init.), shows that a convolutional architecture alone does not yield semantic features.

To produce a more quantitative measurement of the quality of learned representations, we follow the methodology of Van den Oord et al. (2018) and Löwe et al. (2019): we freeze the trained encoder weights and train a linear classifier to recognize the class labels from each individual layer (Figure 4.4). As expected for a deep representation, the classification accuracy increases monotonically with the layer number and only saturates at layers 5 and 6. The accuracies obtained with layer-wise GIM are almost indistinguishable from those obtained with CLAPP. It is only at the last two layers, that layer-wise GIM performs slightly better than CLAPP; yet GIM has multiple biologically implausible features that are removed by CLAPP. As a further benchmark, we also plot the accuracies obtained with an encoder trained with

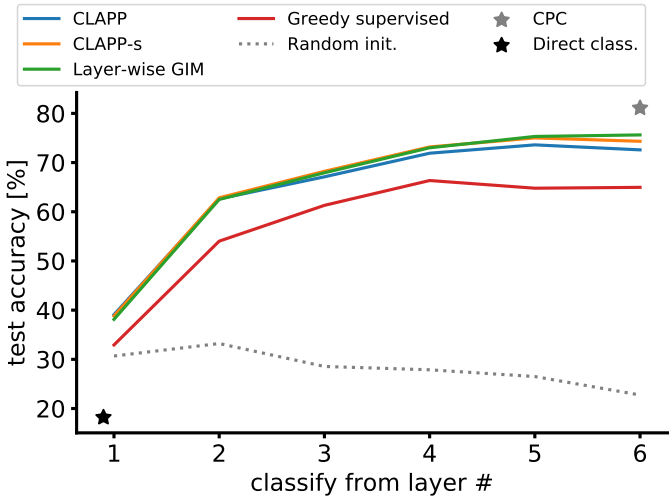


Figure 4.4: CLAPP stacks well: representations after stacking up to 5 layers increase performance of a linear classifier on STL-10, despite the local learning rule (blue and orange lines), while performance decreases for convolutional network with weights fixed at random initialisation (dotted). Greedy supervised training (see subsection 4.6.2) also stacks, but already saturates at layer 4 and shows overall lower performance. Direct linear classification on image pixels (black star) and CPC performance after 6 layers (gray star) serve as upper and lower performance bounds, respectively.

greedy supervised training. This method trains each layer independently using a supervised classifier at each layer, without BP between layers, which results in an almost local update (see Löwe et al. (2019) and subsection 4.6.2). We find that accuracy is overall lower and saturates already at layer 4. On this dataset, with many more unlabelled than labelled images, greedy supervised accuracy is almost 10% below the accuracy obtained with CLAPP. Again, we see that a convolutional architecture alone does not yield hierarchical representations, as performance decreases at higher layers for a fixed random encoder.

Comparing CPC and CLAPP Since CLAPP can be seen as a simplification of CPC (or GIM) we study four algorithmic differences between CPC and CLAPP individually. They are: (1) Gradients in CLAPP (layer-wise GIM) cannot flow from a layer to the next one, as opposed to BP in CPC, (2) CLAPP performs a binary comparison (fixation vs. saccade) with the Hinge loss, whereas CPC does multi-class classification with the cross entropy loss, (3) CLAPP processes a single input at a time, whereas CPC uses many positive and negative samples synchronously, and (4) we introduced $\boldsymbol{W}^{\text{retro}}$ to avoid the weight transport problem in $\boldsymbol{W}^{\text{pred}}$.

We first study features (1) and (2) but relax the constraints from (3) and (4). That means, in this paragraph, we allow a fixation and $N = 16$ synchronous saccades and set $\boldsymbol{W}^{\text{retro}} = \boldsymbol{W}^{\text{pred},\top}$. We refer to *Hinge Loss CPC* as the algorithm minimizing the CLAPP loss (Equation 4.3) but using end-to-end BP. *CLAPP-s* (for *synchronous*) applies the Hinge Loss to every layer, but with gradients blocked between layer. We find that the difference between the CPC loss and Hinge

Loss CPC is less than 1%, see Table 4.1. In contrast, additional blocking of gradients between layers causes a performance drop of almost 5% for both loss functions. We investigate how gradient blocking influences performance with a series of simulations, splitting the 6 layers of the network into two or three gradient isolated modules, exploring the transition from Hinge CPC to CLAPP-s. Performance drops monotonously but not catastrophically, as the number of gradient blocks increases (Table 4.1).

CLAPP's temporal locality allows the interpretation that an agent alternates between fixations and saccades, rather than perfectly recalling negative samples and processing synchronously, as required by CPC. To study the effect of temporal locality, we now apply features (2) and (3) and relax the constraints (1) and (4). The algorithm combining temporal locality and the CLAPP loss function is referred to as *time-local Hinge CPC*. We find that the temporal locality constraint decreases accuracy by 1.2% compared to Hinge Loss CPC. The last feature introduced for biological plausibility is using the matrix $\boldsymbol{W}^{\text{retro}}$ and we observe almost no difference in classification accuracy with this alternative (the accuracy decreases by 0.1%). Conversely, omitting the update in Equation 4.7 entirely, i.e. setting the retrodiction $\boldsymbol{W}^{\text{retro}} = 0$, compromises accuracy by 2% compared to vanilla Hinge Loss CPC.

When combining all features (1) to (4), we find that the fully local CLAPP learning rule leads to an accuracy of 73.6% at layer 5. We conclude from the analysis above, that the feature with the biggest impact on performance is (1): blocking the gradients between each layer. However, despite the performance drop caused by blocking the gradients, CLAPP still stacks well and leverages the depth of the network (Figure 4.4). All other features (2) - (4), introduced to derive a weight update compatible with our prototype (Equation 4.1), only caused a minor performance loss.

Applying CLAPP to speech and video We now demonstrate that CLAPP is applicable to other modalities like the LibriSpeech dataset of spoken speech (Panayotov et al., 2015) and the UCF-101 dataset containing short videos of human actions (Soomro et al., 2012). When applying CLAPP to auditory signals, we do not explicitly model the contrasting mechanism (saccades in the vision task; see discussion in section 4.2 for the auditory pathway) and hence consider the application of CLAPP as benchmark application, rather than a neuroscientifically exact study. To increase computational efficiency, we study CLAPP-s on speech and video. Based on the image experiments, we expect similar results for CLAPP, given enough time to converge. We use the same performance criteria as for the image experiments and summarize our results in Table 4.1, for details see subsection 4.6.2.

For the audio example, we use the same architecture as Van den Oord et al. (2018) and Löwe et al. (2019): multiple temporal 1d-convolution layers and one recurrent GRU layer on top. As in the feedforward case, CLAPP still optimises the objective of Equation 4.3. For the 1d-convolution layers, the context $\boldsymbol{c}^t$ is computed as for the image task, for the last layer, $\boldsymbol{c}^t$ is the output of the recurrent GRU layer. We compare the performance of the algorithms on phoneme classification (41 classes) using labels provided by Van den Oord et al. (2018). In

Table 4.1: CLAPP performs best among methods that are local in space and time. Linear classification test accuracy [%] on STL-10, phone classification on LibriSpeech, and video human action recognition on UCF-101 using features from the encoder trained with different methods. On STL-10, performance degrades gracefully with the number of gradient-isolated modules in the VGG-6 encoder (at fixed number of encoder layers). Greedy supervised training uses BP in auxiliary classifier networks ('almost' local in space). For LibriSpeech, BP through time is used (can be avoided, see subsection 4.6.3). Values with * are taken from Löwe et al. (2019). For simulation details, see subsection 4.6.2.

Method	local in ... space?	local in ... time?	STL-10	LibriSpeech	UCF-101
Chance performance			10.0	2.4	0.99
Random init.	✓	✓	21.8	27.7*	30.5
MFCC	✓	✓	-	39.7*	-
Greedy supervised	(✓)	✓	66.3	73.4*	-
Supervised	✗	✓	73.2	77.7*	51.5
CPC	✗	✗	81.1	64.3	35.7
Layer-wise GIM	✗	✗	75.6	63.9	41.2
Hinge Loss CPC (ours)	✗	✗	80.3	62.8	36.1
CLAPP-s (2 modules of 3 layers)	✗	✗	77.6	-	-
CLAPP-s (3 modules of 2 layers)	✗	✗	77.4	-	-
CLAPP-s (ours)	✓	✗	75.0	61.7	41.6
time-local Hinge CPC (ours)	✗	✓	79.1	-	-
CLAPP (ours)	✓	✓	73.6	-	-

this setting, layer-wise training lowers performance by only 0.4% for layer-wise GIM, and by 1.1% for CLAPP-s. Implemented as such, CLAPP-s still relies on BP through time (BPTT) to train the GRU layer. Using CLAPP-s with biologically plausible e-prop (Bellec et al., 2020), instead of non-local BPTT, reduces performance by only 3.1 %, whereas omitting the GRU layer compromises performance by 9.3 %, see subsection 4.6.3.

Applying CLAPP to videos is especially interesting because their temporal sequence of images perfectly fits the scenario of Figure 4.1 a. In this setting, we take inspiration from Han et al. (2019), and use a VGG-like stack of 2D and 3D convolutions to process video frames over time. On this task (101 classes), we found layer-wise GIM and CLAPP-s to achieve higher downstream classification accuracy than their end-to-end counterparts CPC and Hinge Loss CPC (see Table 4.1), in line with the findings on STL-10 in Löwe et al. (2019). On the other hand, we found that CLAPP-s requires more negative samples (i.e. more simultaneous comparisons of positive and negative samples) on videos than on STL-10 and LibriSpeech. Under the constraint of temporal locality in fully local CLAPP, this leads to prohibitively long convergence times in the current setup. However, since CLAPP linearly combines updates stemming from multiple negative and positive samples, we eventually expect the same final performance, if we run the online CLAPP algorithm for a sufficiently long time.

4.5 Discussion

We introduced CLAPP, a self-supervised and biologically plausible learning rule that yields deep hierarchical representations in neural networks. CLAPP integrates neuroscientific evidence on the dendritic morphology of neurons and takes the temporal structure of natural data into account. Algorithmically, CLAPP minimises a layer-wise contrastive predictive loss function and stacks well on different task domains like images, speech and video – despite the locality in space and time.

While the performance loss due to layer-wise training is a limitation of the current model, the stacking property is preserved and preliminary results suggest improved versions that stack even better (e.g. using adaptive encoding patch sizes). Note that CLAPP models self-supervised learning of cortical hierarchies and does *not* provide a general credit assignment method, such as BP. However, the representation learned with CLAPP could serve as an initialisation for transfer learning, where the encoder is fine-tuned later with standard BP. Alternatively, fine-tuning could even start already during CLAPP training. CLAPP in its current form is data- and compute-intensive, however, it runs on unlabelled data with quasi infinite supply, and is eligible for neuromorphic hardware, which could decrease energy consumption dramatically (Wunderlich et al., 2019).

Classical predictive coding models alter neural activity at inference time, e.g. by cancelling predicted future activity (Rao and Ballard, 1999; Keller and Mrsic-Flogel, 2018). Here, we suggest a different, perhaps complementary, role of predictive coding in synaptic plasticity, where dendritic activity predicts future neural activity, but directly enters the learning rule (Körding and König, 2001; Urbanczik and Senn, 2014). CLAPP currently does not model certain features of biological neurons, e.g. spiking activity or long range feedback, and requires neurons to transmit signals with precise value and timing. We plan to address these topics in future work.

4.6 Supplemental information

Notation in appendices In all appendices, and in line with (Van den Oord et al., 2018; Löwe et al., 2019), the context vector, from which the prediction is performed, is denoted $\boldsymbol{c}^t$ and the feature vector being predicted is denoted $\boldsymbol{z}^{t+\delta t}$ (or $\boldsymbol{z}^{t'}$ for negative samples). In general, the loss function of CPC and CLAPP are therefore defined with the score functions $u_t^\tau = \boldsymbol{z}^{\tau\top} \boldsymbol{W}^{\text{pred}} \boldsymbol{c}^t$.

Throughout the vision experiments and when training the temporal convolutions of the audio processing network, it happens that $\boldsymbol{c}$ and $\boldsymbol{z}$ denote the same layer (see subsection 4.6.2 for details). However, when processing audio, the highest loss uses the last layer as the context layer $\boldsymbol{c}$ and the one before last for $\boldsymbol{z}$.

To cover the most general case, we introduce different notations for the parameters and the variables of the context layer $\boldsymbol{c}$ and the feature layer $\boldsymbol{z}$. For simplicity our analysis considers

standard, fully-connected networks – even if the reasoning generalises easily to other architectures. Hence, with a non-linearity ρ, the feature layer produces the activity $\boldsymbol{z}^t = \rho(\boldsymbol{a}^{z,t})$ with $\boldsymbol{a}^{z,t} = \boldsymbol{W}^z \boldsymbol{x}^{z,t} + \boldsymbol{b}^z$ where $\boldsymbol{x}^{z,t}$, $\boldsymbol{W}^z$ and $\boldsymbol{b}^z$ are the input vector (at time t), weight matrix and bias respectively (the layer index l is omitted for simplicity). The notation naturally extends to the context layer $\boldsymbol{c}$ and we use $\boldsymbol{x}^{c,t}$, $\boldsymbol{W}^c$ and $\boldsymbol{b}^c$ to denote its input and its parameters. Note that when the context and feature layer are the same layer $\boldsymbol{z} = \boldsymbol{c}$, the two parameters $\boldsymbol{W}^c$ and $\boldsymbol{W}^z$ are actually only one single parameter $\boldsymbol{W}$ and the weight update is given by $\Delta \boldsymbol{W} = \Delta \boldsymbol{W}^c + \Delta \boldsymbol{W}^z$.

For the gradient computations in the appendices we assume that the gradient cannot propagate further than one layer. Hence, $\boldsymbol{x}^z$ and $\boldsymbol{x}^c$ are always considered as constants with respect to all parameters, even though this is technically not true, for instance with $\boldsymbol{c}^l = \boldsymbol{z}^{l+1}$. In this case we would have $\boldsymbol{z} = \boldsymbol{x}^c$ and thus $\nabla_{\boldsymbol{W}^z} \boldsymbol{x}^c \neq \boldsymbol{0}$, but we use the convention $\nabla_{\boldsymbol{W}^z} \boldsymbol{x}^c = \boldsymbol{0}$ to obtain local learning rules. Gradients are computed accordingly by stopping gradient propagation in all our experiments.

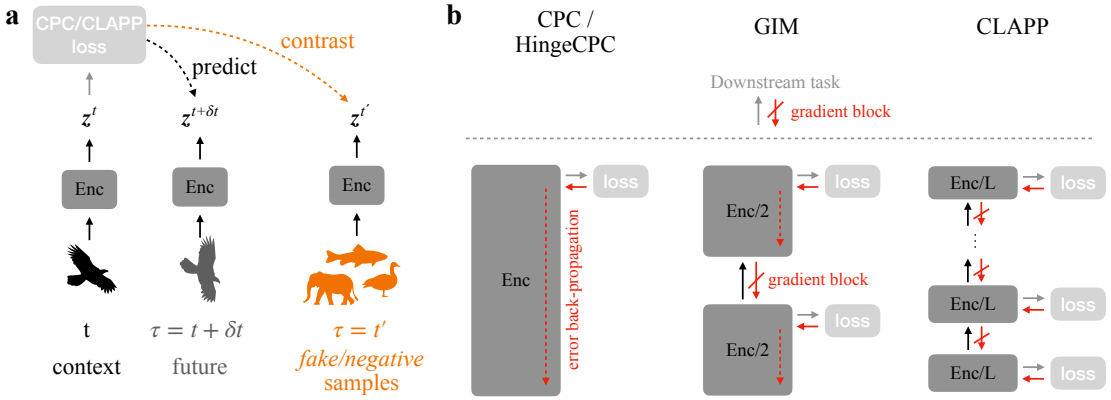


Figure 4.5: **a** In Contrastive Predictive Coding (CPC) and CLAPP(-s), an encoder network (Enc) produces a representation $\boldsymbol{z}^t$ at time t (sometimes more generally called 'context'). Given $\boldsymbol{z}^t$, the encoding of the future input $\boldsymbol{z}^{t+\delta t}$ should be *predicted* while keeping the prediction as different as possible from encoded *fake* or *negative* samples $\boldsymbol{z}^{t'}$ (*contrasting*). The loss function implementing this contrasting depends on the method: CPC uses cross-entropy classification, CLAPP uses a Hinge-loss. **b** CPC trains the encoder network end-to-end using gradient back-propagation (red arrows). In Greedy InfoMax (GIM), the encoder network is split into several, gradient-isolated modules and the loss (CPC or Hinge) is applied separately to each module. Gradient back-propagation still occurs within modules (red, dashed arrows) but is blocked between modules. In CLAPP, every module contains only a single trainable layer of the L-layer encoder. This avoids any back-propagation and makes CLAPP layer-local.

4.6.1 Analysis of the original CPC gradient

Even after preventing gradients to flow from a layer to the next, we argue that parts of the gradient computation in CPC and GIM are hard to implement with the type of information processing that is possible in neural circuits. For this reason we analyse the actual gradients computed by layer-wise GIM. We further discuss the bio-plausibility of the resulting gradient computation in this section.

To derive the loss gradient we define the probability π_t^{t*} that the sample $\boldsymbol{z}^{t*}$ is predicted as the true future given the context layer $\boldsymbol{c}^t$: $\pi_t^{t*} \stackrel{\text{def}}{=} \frac{1}{\mathcal{Z}} \exp u_t^{t*}$ with $\mathcal{Z} \stackrel{\text{def}}{=} \sum_{\tau \in \mathcal{T}} \exp u_t^{\tau}$. The set $\mathcal{T} = \left\{t^{t+\delta t}, t'_1 \dots t'_N\right\}$ comprises the positive and N negative samples. We have in particular $\mathcal{L}_{CPC}^t = -\log \pi_t^{t+\delta t}$ and for any parameter θ the (negative) loss gradient is given by:

$$\nabla_\theta \log \pi_t^{t+\delta t} \quad = \quad \nabla_\theta u_t^{t+\delta t} - \sum_{\tau \in \mathcal{T}} \pi_t^\tau \, \nabla_\theta u_t^\tau \, . \tag{4.9}$$

We consider only three types of parameters: the weights $\boldsymbol{W}^c$ onto the context vector $\boldsymbol{c}^t$, the weights $\boldsymbol{W}^z$ onto the feature vector $\boldsymbol{z}^{t*}$ and the weights $\boldsymbol{W}^{\text{pred}}$ defining the scalar score $u_t^{t*} = \boldsymbol{z}^{t*\top} \boldsymbol{W}^{\text{pred}} \boldsymbol{c}^t$ (the biases are absorbed in the weight matrices for simplicity).

Let's first analyze the gradient with respect to $\boldsymbol{W}^{\text{pred}}$. Using the conventions that k is the index of the context unit c_k and j is the index of the feature unit z_j, we have:

$$\nabla_{W_{jk}^{\text{pred}}} \log \pi_t^{t+\delta t} \quad = \quad c_k^t \left(z_j^{t+\delta t} - \sum_{\tau \in \mathcal{T}} \pi_t^\tau \, z_j^\tau \right) \tag{4.10}$$

Viewing a gradient descent weight update of that parameter as a model of synaptic plasticity in the brain raises essential questions. If $z_j^{t+\delta t} - \sum_{\tau \in \mathcal{T}} \pi_t^\tau z_j^\tau$ was the activity of the unit j, it would boil down to a Hebbian learning rule, well supported experimentally, but the activity of unit j is considered to be the vector element z_j since it is transmitted to the layer above during inference. Hence, the unit j would have to transmit two distinct quantities at the same time, which is unrealistic when modelling real neurons. On top of that, it is unclear how the term $\sum_{\tau \in \mathcal{T}} \pi_t^\tau z_i^\tau$ would be computed.

We now compute the gradient with respect to $\boldsymbol{W}^{\boldsymbol{c}}$ and $\boldsymbol{W}^{\boldsymbol{z}}$. The update of these parameters raises an extra complication because it involves the activity of more than two units. For the parameters of the layer $\boldsymbol{z}$ we denote j a neuron in this layer, and i a neuron from its input layer $\boldsymbol{x}$. Then the loss gradient is given by:

$$\nabla_{W_{ji}^z} \log \pi_t^{t+\delta t} \quad = \quad (\boldsymbol{W}^{\text{pred}} \boldsymbol{c}^t)_j \left(\rho'(a_j^{\boldsymbol{z}})^{t+\delta t} x_i^{\boldsymbol{z},t+\delta t} - \sum_{\tau \in \mathcal{T}} \pi_t^\tau \, \rho'(a_j^{\boldsymbol{z}})^\tau x_i^{\boldsymbol{z},\tau} \right) . \tag{4.11}$$

Similarly, for the parameters of a neuron $\boldsymbol{c}_j^t$:

$$\nabla_{W_{ji}^c} \log \pi^{t+\delta t} \quad = \quad \left(\boldsymbol{W}^{\text{pred},\top} \left(\boldsymbol{z}^{t+\delta t} - \sum_{\tau \in \mathcal{T}} \pi_t^\tau \, \boldsymbol{z}^\tau \right) \right)_j \rho'(a_j^{\boldsymbol{c}})^t x_i^{\boldsymbol{c},t} \, . \tag{4.12}$$

These gradients raise the same essential problems as the computation of the gradients with

respect to $\boldsymbol{W}^{pred}$ and even involve other complex computations.

4.6.2 Simulation details

We use pytorch (Paszke et al., 2017) for our implementation and base it on the code base of the GIM paper (Löwe et al., 2019) [2]. Unless mentioned otherwise we adopt their setup, data sets, data handling and (hyper-)parameters.

Vision experiments

General procedure We use the STL-10 dataset, designed for unsupervised learning algorithms (Coates et al., 2011), which contains 100,000 unlabeled color images of 96×96 pixels. Additionally, STL-10 contains a much smaller labeled training set with 10 classes and 500 training images per class and 800 labeled test images per class. Since CPC-like methods rely on sequences of data we have to introduce an artificial 'temporal' dimension in the case of vision data sets. To simulate a time dimension in these static images we represent the motion of the visual scene by splitting the image into partially overlapping tiles. Then, vertical slices of patches define a temporal order, as in Hénaff et al. (2019) and Löwe et al. (2019): the patches are viewed one after the other in a vertical order (one time step is one patch). The hyperparameters of this procedure and of any other image preprocessing and data augmentation steps are as in Löwe et al. (2019).

This results in a time varying input stimulus which is fed into the encoder network and the weights of this network are updated using the CLAPP rule equation 4.6 and equation 4.7 (or reference algorithms, respectively). CLAPP represents saccades towards a new object by changing the next input image to a different one at any time step with probability 0.5. Note that this practice reduces the number of training data by 50% compared to CLAPP-s, GIM and CPC, which are updated with positive and negative sample synchronously at every step. Since this slows down convergence, we grant CLAPP double the amount of training epochs to yield a fair comparison (1% improvement for CLAPP). We leverage common practices from deep learning to accelerate the simulation: the weight changes are averaged and applied after going through a batch of 32 images so that the images can be processed in parallel. We accumulate the gradient updates and use the Adam optimiser with fixed learning rate 0.0002.

We then freeze the encoder network and train a linear downstream classifier on representations created by the encoder using held-out, labeled data from 10 different classes from the STL-10 dataset. The accuracy of that classification serves as a measure to evaluate the quality of the learned encoder representations.

[2] https://github.com/loeweX/Greedy_InfoMax

Encoder architecture We use VGG-6, a custom 6-layer VGG-like (Simonyan and Zisserman, 2015) encoder with 6 trainable layers (6 convolutional, 4 MaxPool, 0 fully-connected, see Table 4.2). The architecture choice was inspired by the condensed VGG-like architectures successfully applied in Nøkland and Eidnes (2019). The main motivation was to work with an architecture that allows pure layer-wise training which is impossible in e.g. ResNet-50 due to skip-connections. Surprisingly we find that the transition from ResNet-50 to VGG-6 does neither compromise CPC losses nor downstream classification performance for almost all training methods, see Table 4.5.

# of trainable layer	layer type
1	3×3 conv128, ReLU
2	3×3 conv256, ReLU 2×2 MaxPool
3	3×3 conv256, ReLU
4	3×3 conv512, ReLU 2×2 MaxPool
5	3×3 conv1024, ReLU 2×2 MaxPool
6	3×3 conv1024, ReLU 2×2 MaxPool

Table 4.2: Architecture of the VGG-6 encoder network. Convolutional layers (conv) have stride (1, 1), Pooling layers use stride (2, 2). The architecture is inspired by the VGG-like networks used in Nøkland and Eidnes (2019).

In GIM and CLAPP, the encoder is split into several, gradient-isolated modules. Depending on the number of such modules, each module contains a different number of layers. In CPC we do not use any gradient blocking and consequently the encoder consists only of one module containing layers 1-6. In layer-wise GIM and CLAPP each of the 6 modules contains exactly on layer (and potentially another MaxPooling layer). Table 4.3 shows the distribution of layers into modules for the cases in between.

Table 4.3: Distribution of layers into modules as done for the simulations in Table 4.1 and Table 4.4. The layer numbers refer to Table 4.2.

# of modules	layer distribution
1 (CPC)	(1,2,3,4,5,6)
2	(1,2,3), (4,5,6)
3	(1,2), (3,4), (5,6)
4	(1,2,3),(4),(5),(6) or (1),(2),(3),(4,5,6)
6	(1),(2),(3),(4),(5),(6)

Table 4.4: Linear classification test accuracy (%) on STL-10 with features from a VGG-6 encoder trained with CLAPP-s for different sizes of gradient-isolated modules.

# modules	# layers per module	Test accuracy (%)
6, i.e. layer-wise (CLAPP)	1	74.0
4 modules upper	3,1,1,1	75.4
4 modules lower	1,1,1,3	76.2
3 modules	2	77.4
2 modules	3	77.6
1 module (end-to-end) (see Table 4.1)	6	80.3

Table 4.5: Linear classification test accuracy (%) on STL-10 with features coming from two different encoder models: ResNet-50 as in Löwe et al. (2019) and a 6-layer VGG-like encoder (VGG-6). Values for ResNet-50 are taken from Löwe et al. (2019).

	ResNet-50	VGG-6
Random init	27.0	21.8
Greedy Supervised	65.2	65.0
Supervised	71.4	73.2
CPC	80.5	81.1
GIM (3 modules)	**81.9**	78.3

Reference algorithms *Random init* refers to the random initialisation of the encoder network. It thus represents an untrained network with random weight matrices. This 'method' serves as a lower bound on performance and as a sanity check for other algorithms.

In classic *supervised* training, we add a fully-connected layer with as many output dimensions as classes in the data set to the encoder architecture. Then the whole stack is trained end-to-end using a standard supervised loss and back-propagation. For data sets offering many labels this serves as an upper bound on performance of unsupervised methods. In the case of sparsely labeled data, unsupervised learners could, or even should, outperform supervised learning.

The *greedy supervised* method trains every gradient-isolated module of the encoder separately. For that, one fully-connected layer is added on top of each module. Then, for every module, the stack consisting of the module and the added fully-connected layer is trained with a standard supervised loss requiring labels. Gradients are back-propagated within the module but blocked between modules. This layer-wise training makes the method quasi layer-local, however, BP through the added fully-connected layer is still required.

Audio experiments

We follow most of the implementation methods used in Löwe et al. (2019). The model is trained without supervision on 100 hours of clean spoken sentences from the LibriSpeech data set (Panayotov et al., 2015) without any data augmentation. For feature evaluation, a linear classifier is used to extract the phonemes divided into 41 classes. This classifier is trained on the test split of the same dataset, along with the phoneme annotations computed with a software from Van den Oord et al. (2018).

The audio stream is first processed with four 1D convolutional layers and one recurrent layer of Gated Recurrent Units (GRU). The hyperparameters of this architecture are the same as the ones used in Löwe et al. (2019).

All convolutional layers are assigned a CPC or a CLAPP loss as described in the main text and the gradients are blocked between them. To train the last layer – the recurrent layer –, we add one variant of the CLAPP and CPC losses where the score function is defined by $u_t^\tau = \boldsymbol{z}^{\boldsymbol{\tau}\top} \boldsymbol{W}^{\text{pred}} \boldsymbol{c}^t$ where $\boldsymbol{c}^t$ is the activity of the GRU layer and $\boldsymbol{z}^\tau$ is the activity of the last layer of convolutions. This loss is minimized with respect to the parameters of $\boldsymbol{c}$ and $\boldsymbol{z}$, and the gradients cannot flow to the layers below (hence $\nabla_{\boldsymbol{W}^z} \boldsymbol{c}^t = \boldsymbol{0}$ even if $\boldsymbol{z}$ is implicitly the input to $\boldsymbol{c}$ with this architecture).

Within the GRU layer the usual implementation of gradient descent with pytorch involves back-propagation through time (BPTT), even if we avoided BP between layers. To avoid all usage of back-propagation and obtain a more plausible learning rule we used e-prop (Bellec et al., 2020) instead of BPTT. The details of this implementation are provided in the next section (subsection 4.6.3) in the paragraph 'Combining e-prop and CLAPP'.

Video experiments

General procedure We use the UCF-101 dataset (Soomro et al., 2012), an action recognition dataset containing 13,000 videos representing 101 actions. The original clips have a frequency of 30 frames per second and were downsampled by a factor 3. Videos were cut into clips of respectively 54 frames (5.4 seconds) for self-supervised learning and 72 (7.2 seconds) for the following classification. Frames in a clip were randomly grayed and jittered following the procedure of Han et al. (2019). Cropping and horizontal flipping were applied per clip.

Architecture and training For our network, we use a VGG-like network with 5 trainable layers presented in table 4.6. The architecture is decomposed into spatial convolutions processing frames individually and additional temporal convolutions accounting for the temporal component of a clip. The first convolution uses no padding and all others have padding (0, 1, 1). The stride used for the spatial convolutions is, respectively, (1, 2, 2), (1, 2, 2) and (1, 1, 1) whereas the temporal convolutions both have stride (3, 1, 1) to prevent temporal overlap between successive encodings.

Table 4.6: Architecture of the VGG-5 encoder network. Pooling layers use stride (1, 2, 2).

# of trainable layer	layer type
1	1×7×7 conv96, BN, ReLU 1×3×3 MaxPool
2	1×5×5 conv256, BN, ReLU 1×3×3 MaxPool
3	1×3×3 conv512, BN, ReLU
4	3×3×3 conv512, BN, ReLU
5	3×3×3 conv512, BN, ReLU 1×3×3 MaxPool

Whereas Löwe et al. (2019) applies pooling to the feature maps outputted by a layer to obtain the encoding, we flatten them to preserve spatial information necessary to understand and predict the spatial flow and structure from movements related to an action.

For the training procedure, we use a batch size of 8 and train for 300 epochs with a fixed learning rate of 0.001. We use as many negative samples as available in the batch, for the spatial convolutions this leads to 429 negatives and the two temporal convolutions respectively have 141 and 45. This decrease is due to the temporal reductions occurring, aimed at preventing information leakage between sequences.

4.6.3 Additional material

Weight transport in $\boldsymbol{W}^{\text{pred}}$ The update of the encoder weights **W** in CPC, GIM and CLAPP (before introducing $\boldsymbol{W}^{\text{retro}}$) relies on weight transport in W^{pred}, as seen in Equation 4.12 or Equation 4.5.

The activity of $\boldsymbol{c}^t$ is propagated with the matrix $\boldsymbol{W}^{\text{pred}}$ and $\boldsymbol{z}^\tau$ with its transpose. This is problematic because typical synapses in the brain transmit information only in a single direction. The existence of a symmetric reverse connection matrix would solve this problem but raises the issue that connection strengths would have to be synchronised (hence the word *weight transport*) between $\boldsymbol{W}^{\text{pred}}$ and the reverse connections.

One first naive solution is to block the gradient at the layer $\boldsymbol{c}$ in the definition of the score $u_t^\tau = \boldsymbol{z}^{\tau\top}\boldsymbol{W}^{\text{pred}}\text{block\_grad}(\boldsymbol{c}^t)$, with the definition:

$$\begin{aligned} \text{block\_grad}(x) &= x \\ \nabla_x \text{block\_grad}(x) &= 0\,. \end{aligned} \tag{4.13}$$

In this way, no information needs to be transmitted through the transpose of $\boldsymbol{W}^{\text{pred}}$. However this results in a relatively large drop in performance on STL-10 for Hinge Loss CPC (78.0 %)

and CLAPP (70 %).

A better option – and as done in the main paper – is to split the original $\boldsymbol{W}^{\text{pred}}_{\text{orig}}$ into two matrices $\boldsymbol{W}^{\text{pred}}$ and $\boldsymbol{W}^{\text{retro}}$ (for 'retrodiction') which are independent and which allow information flow only in a single direction (as in actual biological synapses). To this end, we split the loss function into two parts: one part receives the activity $\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^t$ coming from $\boldsymbol{c}^t$ and only updates the parameters of $\boldsymbol{z}$; and the other part receives the activity $\boldsymbol{W}^{\text{retro}}\boldsymbol{z}^\tau$ coming from $\boldsymbol{z}^\tau$ and updates the parameters of $\boldsymbol{c}$. Like this information is transmitted through $\boldsymbol{W}^{\text{pred}}$ and $\boldsymbol{W}^{\text{retro}}$ instead of $\boldsymbol{W}^{\text{pred}}$ and its transpose matrix and hence solves the weight transport problem.

More formally, let us write F to summarize the definition of the usual CLAPP loss function in Equation 4.3 such that $\mathscr{L}^t_{\text{CLAPP}} = F(\boldsymbol{c}^t, \boldsymbol{z}^\tau, \boldsymbol{W}^{\text{pred}}_{\text{orig}})$. We then introduce a modified version of the CLAPP loss function:

$$\tilde{\mathscr{L}}^t_{\text{CLAPP}} = \frac{1}{2}\left(\tilde{\mathscr{L}}^{t,\boldsymbol{z}}_{\text{CLAPP}} + \tilde{\mathscr{L}}^{t,\boldsymbol{c}}_{\text{CLAPP}}\right), \tag{4.14}$$

with $\tilde{\mathscr{L}}^{t,\boldsymbol{z}}_{\text{CLAPP}} = F(\text{block\_grad}(\boldsymbol{c}^t), \boldsymbol{z}^\tau, \boldsymbol{W}^{\text{pred}})$ and $\mathscr{L}^{t,\boldsymbol{c}}_{\text{CLAPP}} = F(\boldsymbol{c}^t, \text{block\_grad}(\boldsymbol{z}^\tau), \boldsymbol{W}^{\text{retro}})$. Similarly, we define the corresponding scores as $u^{\tau,\boldsymbol{z}}_t = \boldsymbol{z}^{\tau\top}\boldsymbol{W}^{\text{pred}}\text{block\_grad}(\boldsymbol{c}^t)$ and $u^{\tau,\boldsymbol{c}}_t = \text{block\_grad}(\boldsymbol{z}^\tau)^\top \boldsymbol{W}^{\text{retro},\top}\boldsymbol{c}^t$. With this, the gradients with respect to the weight parameters $W^{\boldsymbol{z}}_{ji}$ (encoding $\boldsymbol{z}^\tau$) are:

$$\frac{\partial u^{\tau,\boldsymbol{z}}_t}{\partial W^{\boldsymbol{z}}_{ji}} = x^{\tau,\boldsymbol{z}}_i \rho'(a^{\tau,\boldsymbol{z}}_j)(\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^t)_j \qquad \text{and} \qquad \frac{\partial u^{\tau,\boldsymbol{c}}_t}{\partial W^{\boldsymbol{z}}_{ji}} = 0\,, \tag{4.15}$$

and the gradients with respect to the weights $W^{\boldsymbol{c}}_{ji}$ (encoding $\boldsymbol{c}^t$) become:

$$\frac{\partial u^{\tau,\boldsymbol{z}}_t}{\partial W^{\boldsymbol{c}}_{ji}} = 0 \qquad \text{and} \qquad \frac{\partial u^{\tau,\boldsymbol{c}}_t}{\partial W^{\boldsymbol{c}}_{ji}} = x^{t,\boldsymbol{c}}_i \rho'(a^{t,\boldsymbol{c}}_j)(\boldsymbol{W}^{\text{retro}}\boldsymbol{z}^\tau)_j\,. \tag{4.16}$$

The final plasticity rule combines those terms and recovers the original CLAPP rule Equation 4.6 and Equation 4.7:

$$\begin{aligned}
\Delta W^\tau_{ji} &= \gamma_\tau\left[\Delta W^{\tau,\boldsymbol{z}}_{ji} + \Delta W^{\tau,\boldsymbol{c}}_{ji}\right] \\
\Delta W^{\tau,\boldsymbol{z}}_{ji} &= \left(\boldsymbol{W}^{\text{pred}}\boldsymbol{c}^t\right)_j \rho'(a^{\tau,\boldsymbol{z}}_j)\, x^{\tau,\boldsymbol{z}}_i \\
\Delta W^{\tau,\boldsymbol{c}}_{ji} &= \left(\boldsymbol{W}^{\text{retro}}\boldsymbol{z}^\tau\right)_j \rho'(a^{t,\boldsymbol{c}}_j)\, x^{t,\boldsymbol{c}}_i,
\end{aligned} \tag{4.17}$$

under the assumption of having only one gating factor γ_τ. This is approximately the case when $\boldsymbol{W}^{\text{pred}}$ and $\boldsymbol{W}^{\text{retro}}$ align since then $u^\tau_t = u^{\tau,\boldsymbol{c}}_t = u^{\tau,\boldsymbol{z}}_t$. We consider this assumption realistic since $\boldsymbol{W}^{\text{pred}}$ and $\boldsymbol{W}^{\text{retro}}$ share the same update rule Equation 4.8. We see that the propagation of the activity through the independent weights $\boldsymbol{W}^{\text{pred}}$ and $\boldsymbol{W}^{\text{retro}}$ is always unidirectional.

It turns out that, using the modified loss $\tilde{\mathscr{L}}^t_{\text{CLAPP}}$, Equation 4.14, instead of the original CLAPP loss $\mathscr{L}^t_{\text{CLAPP}}$, Equation 4.3, the performance on STL-10 (linear classification on last layer) is unchanged for Hinge Loss CPC (80.2 %) and CLAPP-s (74.1 %).

Combining e-prop and CLAPP CLAPP avoids the usage of back-propagation through the depth of the network, but when using a recurrent GRU layer in the audio task, gradients are still back-propagated through time inside the layer. A more plausible alternative algorithm has been suggested in Bellec et al. (2020): synaptic eligibility traces compute local gradients forward in time using the activity of pre- and post-synaptic units, then these traces are merged with the learning signal (here $\boldsymbol{W}\boldsymbol{z}^{t+\delta t}$) to form the weight update. It is simple to implement e-prop with an auto-differentiation software such as pytorch by introducing a block_grad function in the update of the recurrent network. With GRU, we implement a custom recurrent network as follows (the notations are consistent with the pytorch tutorial on GRU networks[3] and unrelated to the rest of the paper):

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_{ir}\boldsymbol{x}_t + \boldsymbol{b}_{ir} + \boldsymbol{W}_{hr}\text{block\_grad}(\boldsymbol{h}_{t-1}) + \boldsymbol{b}_{hr}) \tag{4.18}$$

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_{iz}\boldsymbol{x}_t + \boldsymbol{b}_{iz} + \boldsymbol{W}_{hz}\text{block\_grad}(\boldsymbol{h}_{t-1}) + \boldsymbol{b}_{hz}) \tag{4.19}$$

$$\boldsymbol{n}_t = \tanh\big(\boldsymbol{W}_{in}\boldsymbol{x}_t + \boldsymbol{b}_{in} + \boldsymbol{r}_t \star (\boldsymbol{W}_{hn}\text{block\_grad}(\boldsymbol{h}_{t-1}) + \boldsymbol{b}_{hn})\big) \tag{4.20}$$

$$\boldsymbol{h}_t = (1-\boldsymbol{z}_t) \star \boldsymbol{n}_t + \boldsymbol{z}_t \star \boldsymbol{h}_{t-1} \tag{4.21}$$

In summary we use $\boldsymbol{h}_t$ as the hidden state of the recurrent network, $\boldsymbol{r}_t$, $\boldsymbol{z}_t$ and $\boldsymbol{n}_t$ as the network gates, $\star$ as the term-by-term product, and $\boldsymbol{W}_.$ and $\boldsymbol{b}_.$ as the weights and bias respectively. One can show that applying e-prop in a classical GRU network is mathematically equivalent to applying BPTT in the network above.

In simulations, we evaluate the performance as the phoneme classification accuracy on the test set. We find that CLAPP-s achieves 61.7% with BPTT and 58.6% with e-prop; but the latter can be implemented with purely local learning rules by relying on eligibility traces (Bellec et al., 2020). In comparison, phoneme classification from the last feedforward layer before the RNN only yields 52.4% accuracy.

Biologically plausible computation of the score $u_t^{t+\delta t}$ We think of the loss $\mathscr{L}^t_{CLAPP}$ in Equation 4.3 as a surprise signal that is positive if the prediction is wrong, either because a fixation has been wrongly predicted as a saccade or vice-versa. Surprising events are indicated by physiological markers of brain activity such as the EEG or pupil dilation. Moreover, the activity of neuromodulators such as norepinephrine, acetylcholine, and partially also dopamine is correlated with surprising events; an active sub-field of computational neuroscience attempts to relate neuro-modulators to surprise and uncertainty (Angela and Dayan, 2005; Nassar et al., 2012; Ostwald et al., 2012; Schwartenbeck et al., 2013; Heilbron and Meyniel, 2019; Liakoni et al., 2021).

[3] https://pytorch.org/docs/stable/generated/torch.nn.GRU.html

In analogy to the theory of reinforcement learning, where abstract models have been successfully correlated with brain activity and dopamine signals well before the precise brain circuitry necessary to calculate the dopamine signal was known (Dabney et al., 2020), we take the view that surprise signals exist and can be used in the models, even if we have not yet identified a circuit to calculate them. The neuromodulator signal in our model would be 1 if $\mathscr{L}^t_{CLAPP} > 0$ and zero otherwise. Thus the exact value of $\mathscr{L}^t_{CLAPP}$ is not needed.

Nevertheless, let us try to sketch a mechanism to compute this signal. Every neuron i has access to its 'own' internal dendritic signal $\hat{z}^t_i = \sum_j W^{pred}_{ij} c^t_j$ interpretable as the dendritic prediction of somatic activity (Urbanczik and Senn, 2014). What we need is the product $z^{t+\delta t}_i \hat{z}_i{}^t$ and then we need to sum over all neurons. Four insights are important. First, a potential problem is that the dendritic prediction $\hat{z}^t_i$ is *different* from the actual activity $z^{t+\delta t}_i$ that is driven in our model by feedforward input. However, the work of Larkum et al. (1999) has shown that neurons emit specific burst-like signals if both dendrite and soma are activated. The product $z^{t+\delta t}_i \hat{z}_i{}^t$ can be seen as a detector of such coincident events. Second, if bursts indicate such coincident events, then the burst signals of many neurons need to be summed together, which could be done either by an interneuron in the same area (same layer of the model) or by neurons in a deep nucleus located below the cortex. The activity of this nucleus would serve as one of the inputs of the nucleus that actually calculates surprise. Third, following ideas on time-multiplexing in Payeur et al. (2021), the burst signal can be considered as a communication channel that is *separate* from the single-spike communication channel for the feedforward network used for inference. Fourth, as often in neuroscience, positive and negative signals must be treated in different pathways (the standard example is ON and OFF cells in the visual system), before they would be finally combined with the saccade signal y to emit the binary surprise signal $\gamma_t = y^t H^t$ that is broadcasted to the area corresponding to one layer of our network. An empirical test of this suggested circuitry is out of scope for the present paper but will be addressed in future work.

Preferred patch visualisation for random encoder As a control, we repeat the preferred patch visualisation analysis, as in Figure 4.3 a, for the random encoder, i.e. a network with random fixed weights. The result is shown in Figure 4.6 b, in comparison with the analysis of an encoder trained with CLAPP. For CLAPP, higher layers extract higher-level features creating a hierarchy, whereas for the random encoder, no clear hierarchy is apparent across layers. Together with the non-informative t-SNE embedding of the random encoder (Figure 4.3 b), this suggest that a convolutional architecture alone does not yield hierarchical representations.

Gradient flow through MaxPooling layers In fact, MaxPooling can be viewed as a simple model of lateral inhibition which provides a learning rule compatible with Equation 4.1, without introducing approximations and without blocking gradients below the MaxPool operator.

The idea is that 2×2 MaxPooling can be viewed as a simple model of lateral inhibition between

Figure 4.6: (As Figure 4.3 a) Red boxes in STL-10 images indicate patches that best activate a specific neuron (rows) in **a** a network trained with CLAPP or **b** a random encoder with random weights fixed at initialisation. For CLAPP, layer 1 extracts simple features like gratings or uniform patches, whereas higher layers extract richer features like object parts. For the random encoder, no clear hierarchy is apparent across layers.

the 4 neurons involved. During inference, this inhibition enforces only one of the four neurons to be active. We use the following notation for the output of the MaxPool operator $z'^{t}_{j'} = \max\{z^t_{j_0}, z^t_{j_1}, z^t_{j_2}, z^t_{j_3}\}$, where z^t_i is defined as in the main paper. We define $\boldsymbol{c}'^{t}$ accordingly for the context layer, if it includes a MaxPool operator.

Then, following the derivation from the main text, the learning rule is proportional to the gradient $\frac{\partial u_t^{t+\delta t}}{\partial W_{ij}}$, but now $u_t^{t+\delta t}$ is defined using the output of the pooling operators: $u_t^{t+\delta t} = \sum_{k',j'} z'^{t+\delta t}_{j'} W^{\text{pred}}_{j'k'} c'^{t}_{k'}$. Since the partial derivative over the MaxPool operator is either 1 (the neurons is active), or 0 (for the other three neurons, which are inhibited), $\frac{\partial u_t^{t+\delta t}}{\partial W_{ij}}$ is either $\left(\sum_{k'} W^{\text{pred}}_{j'k'} c'^{t}_{k'}\right) \cdot \sigma'(a^{t+\delta t}_{j'}) x^{t+\delta t}_i$ if $j = j'$ (the neuron is active), or 0 if $j \neq j'$ (the neuron is inhibited). Hence, and without further approximation, the learning rule is only applied if the

neuron is active, in which case $\frac{\partial u_i^{t+\delta t}}{\partial W_{ij}}$ takes the form 'dendritic signal × post × pre', and the resulting learning rule is compatible with Equation 4.1.

Predicting from higher layers We ran CLAPP-s with the context representation c^t coming from one layer above the predicted layer $z^{t+\delta t}$ (except for the last layer, where c^t and $z^{t+\delta t}$ come from the same layer). Linear classification performance on STL-10 still grows over layers but only yields 72.4 % test accuracy when classifying from the last layer. In comparison defining $\boldsymbol{c}^t$ to be the same layer as $\boldsymbol{z}^t$ reached 75.0%.

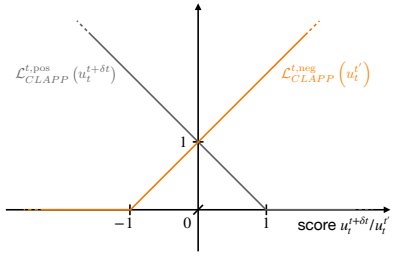


Figure 4.7: Illustration of the positive ($y = +1$, gray) and negative ($y = -1$, orange) part of the CLAPP loss, see Equation 4.3

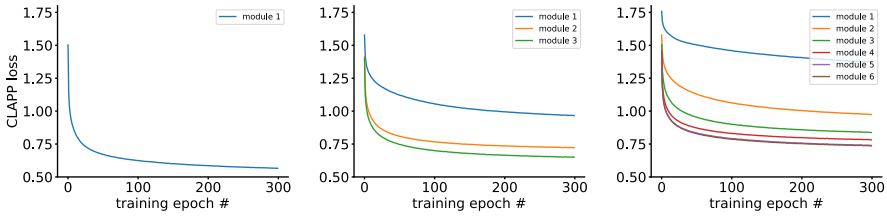


Figure 4.8: CLAPP-s training losses for encoders split into 1, 3 or 6 gradient-isolated modules.

5 Conclusion

This thesis is placed at the intersection of computational neuroscience and AI, and investigates biologically plausible models of learning hierarchical representations. More specifically, chapter 2, chapter 3 & chapter 4 cover three main questions in this context: (i) What are useful neural representations and when are deep hierarchical representations needed? (chapter 2), (ii) Can such representations evolve in the brain with known biologically plausible learning rules? (chapter 3), and (iii) Can AI inspire novel models of biological learning, that build deep representations, yet are still biologically plausible? (chapter 4).

In chapter 2, we investigated biologically plausible learning in shallow networks, i.e. networks with only one hidden layer of learned features. This study aimed at testing the limits of feature learning in such shallow networks on digit (MNIST) and object (CIFAR10) classification, before addressing deeper networks in the next two chapters. We found that, for large hidden layers, random features can support surprisingly high performance. When combined with localized receptive fields, such random features even reach the performance of supervised backpropagation on MNIST, but not on CIFAR10. We conclude that models of biologically plausible learning in shallow and deep networks should significantly surpass such random feature benchmarks on MNIST, or, preferably, that the quality and capability of such models should be evaluated in different ways.

The main question of chapter 3 was how hierarchical properties of neurons in the visual cortex, e.g. selectivity and invariance, could evolve from Hebbian learning and realistic input. We followed a *bottom-up* approach to build a model from basic principles such as Hebbian learning rules. Inspired by common models of simple and complex cells found in primary visual cortex (V1), that are based on sparse coding and invariance learning, we proposed a simple unifying model, *competitive temporal Hebbian learning*. In a shallow 2-layer network, our simple model learned simple and complex cell receptive fields as found in V1. However, when applied to deeper networks by stacking multiple layers, Hebbian learning did not learn hierarchical representations of increasing usefulness. From this, it seems that standard Hebbian learning is too constrained to build increasingly useful and invariant representations, as observed in the visual cortex or deep artificial neural networks.

In chapter 4, we asked the same question as in chapter 3, but this time pursuing a *top-down* strategy: We sought inspiration from recent breakthroughs in unsupervised deep learning to design a *neo-Hebbian* learning rule that applies contrastive predictive learning to a causal, biological setting using saccades. The learning rule is neo-Hebbian because it comprises more factors than the classical pre- and post-synaptic neuronal activity. However, all factors are locally available in form of predictive dendritic input and layer-wise modulation factors. We found that this model of biological learning builds deep hierarchical representations of images, speech and video—just as its AI counterpart.

Understanding the brain and creating intelligent machines—the two challenges that opened this thesis—remain pressing questions for neuroscience and AI. With this thesis, we hope to contribute to answering these questions by addressing the topics of *neural representations* and *learning*. We see our modeling as a potential starting point for new hypotheses in neuroscience, that can be tested experimentally. Furthermore, our research facilitates adding more biological realism to AI models. Including biological features like locality, self-supervision and on-line processing could help overcome many shortcomings of current AI, such as high power consumption and the need for large amounts of labeled data.

A Additional publications

A.1 Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape

Authors Johanni Brea, Berfin Simsek, **Bernd Illing**, Wulfram Gerstner

Abstract The permutation symmetry of neurons in each layer of a deep neural network gives rise not only to multiple equivalent global minima of the loss function, but also to first-order saddle points located on the path between the global minima. In a network of $d-1$ hidden layers with n_k neurons in layers $k = 1, \ldots, d$, we construct smooth paths between equivalent global minima that lead through a 'permutation point' where the input and output weight vectors of two neurons in the same hidden layer k collide and interchange. We show that such permutation points are critical points with at least n_{k+1} vanishing eigenvalues of the Hessian matrix of second derivatives indicating a *local* plateau of the loss function. We find that a permutation point for the exchange of neurons i and j transits into a flat valley (or generally, an *extended* plateau of n_{k+1} flat dimensions) that enables all $n_k!$ permutations of neurons in a given layer k at the same loss value. Moreover, we introduce high-order permutation points by exploiting the recursive structure in neural network functions, and find that the number of K^{th}-order permutation points is at least by a factor $\sum_{k=1}^{d-1} \frac{1}{2!^K} \binom{n_k - K}{K}$ larger than the (already huge) number of equivalent global minima. In two tasks, we illustrate numerically that some of the permutation points correspond to first-order saddles ('permutation saddles'): first, in a toy network with a single hidden layer on a function approximation task and, second, in a multilayer network on the MNIST task. Our geometric approach yields a lower bound on the number of critical points generated by weight-space symmetries and provides a simple intuitive link between previous mathematical results and numerical observations.

Author contributions The project was designed by WG, BS and JB. Theory was done by BS and WG, simulations by JB and BI. The paper was written by all authors.

Publication Brea et al. (2019)

Bibliography

E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Josa a*, 2(2):284–299, 1985.

G. Amato, F. C. B, F. Falchi, C. Gennaro, and G. Lagani. Hebbian Learning Meets Deep Convolutional Neural Networks. In *ICIAP*, pages 324–334. Springer International Publishing, 2019. ISBN 9783030306427. doi: 10.1007/978-3-030-30642-7.

Y. Amit. Deep learning with asymmetric connections and hebbian updates. *Frontiers in computational neuroscience*, 13:18, 2019.

J. Y. Angela and P. Dayan. Uncertainty, neuromodulation, and attention. *Neuron*, 46(4):681–692, 2005.

J. Antolik and J. A. Bednar. Development of maps of simple and complex cells in the primary visual cortex. *Frontiers in computational neuroscience*, 5:17, 2011.

Y. Bahroun, E. Hunsicker, and A. Soltoggio. Building efficient deep hebbian networks for image classification tasks. In *International Conference on Artificial Neural Networks*, pages 364–372. Springer, 2017.

P. Baldi, P. Sadowski, and Z. Lu. Learning in the Machine: Random Backpropagation and the Learning Channel. *arXiv Prepr.*, pages 1–57, 2016. URL http://arxiv.org/abs/1612.02734.

A. R. Barron. Universal Approximation Bounds for Superposition of a Sigmoid Function. *IEEE Trans. Inf. Theory*, 39(3):930–945, 1993. ISSN 09205691. doi: 10.1007/s11263-010-0390-2.

A. R. Barron, B. Brandy, and S. Yale. Approximation and Estimation Bounds for Artificial Neural Networks. *Mach. Learn.*, 14:115–133, 1994.

S. Bartunov, A. Santoro, B. A. Richards, L. Marris, G. E. Hinton, and T. Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *arXiv preprint arXiv:1807.04587*, 2018.

G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *arXiv Prepr.*, pages 1–17, 2018. URL http://arxiv.org/abs/1803.09574.

G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass. A solution to the learning dilemma for recurrent networks. *Nature communications*, 11, 2020.

P. Berkes and L. Wiskott. Slow feature analysis yields a rich repertoire of complex cell properties. *J. Vis.*, 5(6):9–9, 2005. ISSN 1534-7362. doi: 10.1167/5.6.9. URL http://jov.arvojournals.org/Article.aspx?doi=10.1167/5.6.9.

V. Boutin, A. Franciosini, F. Chavane, F. Ruffier, and L. Perrinet. Sparse Deep Predictive Coding captures contour integration capabilities of the early visual system. pages 1–35, 2019. URL http://arxiv.org/abs/1902.07651.

V. Boutin, A. Franciosini, F. Ruffier, and L. Perrinet. Effect of Top-Down Connections in Hierarchical Sparse Coding. *Neural Comput.*, pages 1–31, 2020. ISSN 0899-7667. doi: 10.1162/neco_a_01325.

J. Brea, B. Simsek, B. Illing, and W. Gerstner. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*, 2019.

C. S. Brito and W. Gerstner. Nonlinear hebbian learning as a unifying principle in receptive field formation. *PLoS computational biology*, 12(9):e1005070, 2016.

D. V. Buonomano and M. M. Merzenich. Cortical plasticity: from synapses to maps. *Annual review of neuroscience*, 21(1):149–186, 1998.

K. S. Burbank. Mirrored stdp implements autoencoder learning in a network of spiking neurons. *PLoS computational biology*, 11(12):e1004566, 2015.

N. Caporale and Y. Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

T. N. Chandrapala and B. E. Shi. Learning slowness in a sparse model of invariant feature detection. *Neural Computation*, 27(7):1496–1529, 2015.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Int. Conf. Mach. Learn.*, 2020. URL https://github.com/google-research/simclr.

Y. Chen, D. M. Paiton, and B. A. Olshausen. The Sparse Manifold Transform. *Advances in Neural Information Processing Systems*, pages 1–17, 2018. URL https://arxiv.org/pdf/1806.08887.pdf.

D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. Flexible, High Performance Convolutional Neural Networks for Image Classification. In *Int. Jt. Conf. Artif. Intell.*, 2011.

A. Citri and R. C. Malenka. Synaptic plasticity: multiple forms, functions, and mechanisms. *Neuropsychopharmacology*, 33(1):18–41, 2008.

A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

M. Concetta Morrone and D. Burr. Feature detection in human vision: A phase-dependent energy model. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 235 (1280):221–245, 1988.

F. Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.

W. Dabney, Z. Kurth-Nelson, N. Uchida, C. K. Starkweather, D. Hassabis, R. Munos, and M. Botvinick. A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792):671–675, 2020.

H. Dale. Pharmacology and Nerve-endings (Walter Ernest Dixon Memorial Lecture): (Section of Therapeutics and Pharmacology). *Proc. R. Soc. Med.*, 28(3):319–332, 1935. ISSN 0035-9157.

S. Dasgupta, T. C. Sheehan, C. F. Stevens, and S. Navlakha. A neural data structure for novelty detection. *Proc. Natl. Acad. Sci.*, 115(51):13093–13098, 2018. doi: 10.1073/pnas.1814448115.

K. Deisseroth. Optogenetics. *Nature methods*, 8(1):26–29, 2011.

C. B. Delahunt and J. N. Kutz. Putting a bug in ML: The moth olfactory network learns to read MNIST. *arXiv Prepr.*, (i):1–16, 2018. URL http://arxiv.org/abs/1802.05405.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li. ImageNet: A large-scale hierarchical image database. *IEEE Conf. Comput. Vis. pattern Recognit.*, pages 248–255, 2009. ISSN 1063-6919. doi: 10.1109/CVPRW.2009.5206848. URL http://www.image-net.org.

J. J. DiCarlo and D. D. Cox. Untangling invariant object recognition. *Trends Cogn. Sci.*, 11(8): 333–341, 2007. ISSN 13646613. doi: 10.1016/j.tics.2007.06.010.

P. U. Diehl and M. Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.*, 9(August):1–9, 2015. ISSN 1662-5188. doi: 10.3389/fncom.2015.00099. URL http://journal.frontiersin.org/Article/10.3389/fncom.2015.00099/abstract.

P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer. Fast-Classifying, High-Accuracy Spiking Deep Networks Through Weight and Threshold Balancing. *Int. Jt. Conf. Neural Networks*, 2015.

J. T. Dudman, D. Tsay, and S. A. Siegelbaum. A Role for Synaptic Inputs at Distal Dendrites: Instructive Signals for Hippocampal Long-Term Plasticity. *Neuron*, 56(5):866–879, dec 2007. ISSN 08966273. doi: 10.1016/j.neuron.2007.10.020.

W. Einhäuser, C. Kayser, P. König, and K. Körding. Learning the invariance properties of complex cells from their responses to natural stimuli. *European Journal of Neuroscience*, 15 (3):475–486, 2002.

M. Fagiolini, T. Pizzorusso, N. Berardi, L. Domenici, and L. Maffei. Functional postnatal development of the rat primary visual cortex and the role of visual experience: Dark rearing and monocular deprivation. *Vision Res.*, 34(6):709–720, mar 1994. ISSN 00426989. doi: 10.1016/0042-6989(94)90210-0.

P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biol. Cybern.*, 64(2): 165–170, 1990. ISSN 03401200. doi: 10.1007/BF02331346.

P. Földiák. Learning Invariance from Transformation Sequences. *Neural Comput.*, 3(2):194–200, 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.2.194. URL http://www.mitpressjournals.org/doi/10.1162/neco.1991.3.2.194.

J. B. Fritz, M. Elhilali, S. V. David, and S. A. Shamma. Auditory attention - focusing the searchlight on sound. *Curr. Opin. Neurobiol.*, 17(4):437–455, aug 2007. ISSN 09594388. doi: 10.1016/j.conb.2007.07.011.

K. Fukushima. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biol. Cybern.*, 36:193–202, 1980.

K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988. ISSN 08936080. doi: 10.1016/0893-6080(88)90014-7.

W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

W. Gerstner, M. Lehmann, V. Liakoni, D. Corneil, and J. Brea. Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Frontiers in neural circuits*, 12:53, 2018.

S. Golkar, D. Lipshutz, Y. Bahroun, A. M. Sengupta, and D. B. Chklovskii. A biologically plausible neural network for local supervision in cortical microcircuits. *arXiv preprint arXiv:2011.15031*, 2020.

J. Guerguiev, T. P. Lillicrap, and B. A. Richards. Towards deep learning with segregated dendrites. *Elife*, 6:e22901, 2017.

T. Han, W. Xie, and A. Zisserman. Video representation learning by dense predictive coding. *arXiv preprint*, 2019.

D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

A. Hayashi-Takagi, S. Yagishita, M. Nakamura, F. Shirai, Y. I. Wu, A. L. Loshbaugh, B. Kuhlman, K. M. Hahn, and H. Kasai. Labelling and optical erasure of synaptic memory traces in the motor cortex. *Nature*, 525(7569):333–338, 2015. ISSN 14764687. doi: 10.1038/nature15257.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Conf. Comput. Vis. Pattern Recognit.*, pages 9726–9735. IEEE Computer Society, nov 2019. URL http://arxiv.org/abs/1911.05722.

Y. He, K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi. Unsupervised feature learning by Deep Sparse Coding. *SIAM Int. Conf. Data Min. 2014, SDM 2014*, 2:902–910, 2014. doi: 10.1137/1.9781611973440.103.

D. O. Hebb. *The Organization of Behavior*. 1949. ISBN 0805843000.

D. J. Heeger. Normalization of cell responses in cat striate cortex. *Visual neuroscience*, 9(2): 181–197, 1992.

M. Heilbron and F. Meyniel. Confidence resets reveal hierarchical adaptive learning in humans. *PLoS computational biology*, 15(4):e1006972, 2019.

O. J. Hénaff, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord. Data-Efficient Image Recognition with Contrastive Predictive Coding. *arXiv Prepr.*, 2019.

G. Hinton. How to do backpropagation in a brain. In *Invited talk at the NIPS'2007 deep learning workshop*, volume 656, 2007.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, 79:2554–2558, 1982.

J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

H. Hosoya. Multinomial bayesian learning for modeling classical and nonclassical receptive field properties. *Neural computation*, 24(8):2119–2150, 2012.

G. B. Huang, Q. Y. Zhu, and C. K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006. ISSN 09252312. doi: 10.1016/j.neucom.2005.12.126.

D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol.*, 160(1):106–154, 1962. ISSN 00223751. doi: 10.1113/jphysiol.1962.sp006837. URL http://doi.wiley.com/10.1113/jphysiol.1962.sp006837.

D. H. Hubel and T. N. Wiesel. Receptive fields of cells in striate cortex of very young, visually inexperienced kittens. *Journal of neurophysiology*, 26(6):994–1002, 1963.

J. Hurri and A. Hyvärinen. A two-layer temporal generative model of natural video exhibits complex-cell-like pooling of simple cell outputs. *Neurocomputing*, 52:553–559, 2003.

S. Hussain, S. C. Liu, and A. Basu. Improved margin multi-class classification using dendritic neurons with morphological learning. *Proc. - IEEE Int. Symp. Circuits Syst.*, pages 2640–2643, 2014. ISSN 02714310. doi: 10.1109/ISCAS.2014.6865715.

A. Hyvärinen and P. O. Hoyer. A two-layer sparse coding model learns simple and complex cell receptive fields and topography from natural images. *Vision Res.*, 41(18):2413–2423, 2001. ISSN 00426989. doi: 10.1016/S0042-6989(01)00114-6.

A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural computation*, 9(7):1483–1492, 1997.

A. Hyvärinen and E. Oja. Independent component analysis by general nonlinear Hebbian-like learning rules. *Signal Processing*, 64:301–313, 1998.

B. Illing, W. Gerstner, and J. Brea. Biologically plausible deep learning—but how far can we go with shallow networks? *Neural Networks*, 118:90–101, 2019.

B. Illing, W. Gerstner, and G. Bellec. Towards truly local gradients with clapp: Contrastive, local and predictive plasticity. *arXiv preprint arXiv:2010.08262*, 2020.

T. Isomura and T. Toyoizumi. Error-Gated Hebbian Rule: A Local Learning Rule for Principal and Independent Component Analysis. *Sci. Rep.*, 8(1):1–11, 2018. ISSN 20452322. doi: 10.1038/s41598-018-20082-0. URL http://dx.doi.org/10.1038/s41598-018-20082-0.

E. Jonas and K. P. Kording. Could a Neuroscientist Understand a Microprocessor ? *PLoS Comput. Biol.*, 13:1–24, 2017. doi: 10.1371/journal.pcbi.1005268.

G. B. Keller and T. D. Mrsic-Flogel. Predictive Processing: A Canonical Cortical Computation. *Neuron*, 100(2):424–435, 2018. ISSN 10974199. doi: 10.1016/j.neuron.2018.10.003. URL https://doi.org/10.1016/j.neuron.2018.10.003.

S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018. ISSN 08936080. doi: 10.1016/j.neunet.2017.12.005. URL http://linkinghub.elsevier.com/retrieve/pii/S0893608017302903.

D. Kingma and J. L. Ba. ADAM: A method for stochastic optimization. *ICLR 2015*, 2015. ISSN 15517616. doi: 10.1063/1.4902458.

A. A. Kohan, E. A. Rietman, and H. T. Siegelmann. Error Forward-Propagation: Reusing Feedforward Connections to Propagate Errors in Deep Learning. *arXiv Prepr.*, 2018. URL http://arxiv.org/abs/1808.03357.

T. Kohonen. Self-Organized Formation of Topologically Correct Feature Maps. *Biol. Cybern.*, 43:59–69, 1982.

T. Kohonen. *Self-organizing maps*, volume 30. Springer Science & Business Media, 2012.

K. P. Körding, C. Kayser, W. Einhäuser, and P. König. How Are Complex Cell Properties Adapted to the Statistics of Natural Stimuli ? How Are Complex Cell Properties Adapted to the Statistics of Natural Stimuli ? *J. Neurophysiol.*, 91(August 2003):206–212, 2004. ISSN 0022-3077. doi: 10.1152/jn.00149.2003. URL http://www.ncbi.nlm.nih.gov/pubmed/12904330.

P. K. Körding and P. König. Neurons with Two Sites of Synaptic Integration Learn Invariant Representations. *Neural Comput.*, 13:2823–2849, 2001.

E. Kowler, E. Anderson, B. Dosher, and E. Blaser. The role of attention in the programming of saccades. *Vision Res.*, 35(13):1897–1916, 1995. ISSN 00426989. doi: 10.1016/0042-6989(94)00279-U. URL https://pubmed.ncbi.nlm.nih.gov/7660596/.

N. Kriegeskorte. Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing. *Annu. Rev. Vis. Sci.*, 1(1):417–446, 2015. ISSN 2374-4642. doi: 10.1146/annurev-vision-082114-035447.

A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *Sci. Dep. Univ. Toronto, Tech.*, pages 1–60, 2009. ISSN 1098-6596. doi: 10.1.1.222.9220.

A. Krizhevsky. https://www.cs.toronto.edu/~kriz/cifar.html, 2013. URL https://www.cs.toronto.edu/{~}kriz/cifar.html.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.*, pages 1–9, 2012. ISSN 10495258. doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

D. Krotov, J. J. Hopfield, and D. D. Lee. Unsupervised learning by competing hidden units. *Proc. Natl. Acad. Sci.*, 116(16):7723–7731, 2019. doi: 10.1073/pnas.1820458116.

S. R. Kulkarni and B. Rajendran. Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization. *Neural Networks*, 103:118–127, 2018. ISSN 18792782. doi: S0893608018301126.

D. Kunin, A. Nayebi, J. Sagastuy-Brena, S. Ganguli, J. M. Bloom, and D. L. K. Yamins. Two Routes to Scalable Credit Assignment without Weight Symmetry. In *ICML*, 2020. URL http://arxiv.org/abs/2003.01513.

Ł. Kuśmierz, T. Isomura, and T. Toyoizumi. Learning with three factors: modulating Hebbian plasticity with errors. *Curr. Opin. Neurobiol.*, 46:170–177, oct 2017. ISSN 18736882. doi: 10.1016/j.conb.2017.08.020. URL http://dx.doi.org/10.1016/j.conb.2017.08.020.

A. Laborieux, M. Ernoult, B. Scellier, Y. Bengio, J. Grollier, and D. Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:129, 2021.

G. Lagani, F. Falchi, C. Gennaro, and G. Amato. Hebbian Semi-Supervised Learning in a Sample Efficiency Setting. *arXiv Prepr.*, 2021.

M. E. Larkum, J. J. Zhu, and B. Sakmann. A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, 398(6725):338–341, 1999. ISSN 00280836. doi: 10.1038/18686.

Y. LeCun. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1: 541–551, 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541.

Y. LeCun. http://yann.lecun.com/exdb/mnist/, 1998. URL http://yann.lecun.com/exdb/mnist/.

Y. LeCun. Learning invariant feature hierarchies. In *European conference on computer vision*, pages 496–505. Springer, 2012.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nat. Rev.*, 521:436 – 444, 2015. ISSN 0028-0836. doi: 10.1038/nature14539.

C. Lee, G. Srinivasan, P. Panda, and K. Roy. Deep Spiking Convolutional Neural Network Trained with Unsupervised Spike Timing Dependent Plasticity. *IEEE Trans. Cogn. Dev. Syst.*, 8920(c):1–1, 2018. ISSN 2379-8920. doi: 10.1109/TCDS.2018.2833071. URL https://ieeexplore.ieee.org/document/8354825/.

D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 498–515. Springer, 2015.

J. H. Lee, T. Delbruck, and M. Pfeiffer. Training deep spiking neural networks using backpropagation. *Front. Neurosci.*, 10(NOV), 2016. ISSN 1662453X. doi: 10.3389/fnins.2016.00508.

D. A. Leopold and N. K. Logothetis. Microsaccades differentially modulate neural activity in the striate and extrastriate visual cortex. *Experimental Brain Research*, 123(3):341–345, 1998.

N. Li and J. J. DiCarlo. Unsupervised natural experience rapidly alters invariant object representation in visual cortex. *science*, 321(5895):1502–1507, 2008.

V. Liakoni, A. Modirshanechi, W. Gerstner, and J. Brea. Learning in volatile environments with the bayes factor surprise. *Neural Computation*, 33(2):269–340, 2021.

J.-P. Lies, R. M. Häfner, and M. Bethge. Slowness and Sparseness Have Diverging Effects on Complex Cell Learning. *PLoS Comput Biol*, 10(3):1003468, 2014. doi: 10.1371/journal.pcbi.1003468. URL www.ploscompbiol.org.

T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):1–10, 2016.

T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Z. Lin and R. Memisevic. How Far Can We Go Without Convolution : Improving Fully - Connected Networks. In *Work. track - ICLR 2016*, pages 1–10, 2016.

R. Linsker. From basic network principles to neural architecture: Emergence of spatial-opponent cells. *Proceedings of the National Academy of Sciences*, 83(19):7508–7512, 1986a.

R. Linsker. From basic network principles to neural architecture: Emergence of orientation-selective cells. *Proceedings of the National Academy of Sciences*, 83(21):8390–8394, 1986b.

R. Linsker. From basic network principles to neural architecture: Emergence of orientation columns. *Proceedings of the National Academy of Sciences*, 83(22):8779–8783, 1986c.

A. Litwin-Kumar, K. D. Harris, R. Axel, H. Sompolinsky, and L. F. Abbott. Optimal Degrees of Synaptic Connectivity. *Neuron*, 93(5):1153–1164.e7, 2017. ISSN 10974199. doi: 10.1016/j.neuron.2017.01.030. URL http://dx.doi.org/10.1016/j.neuron.2017.01.030.

D. Liu and S. Yue. Event-Driven Continuous STDP Learning With Deep Structure for Visual Pattern Recognition. *IEEE Trans. Cybern.*, pages 1–14, 2018. ISSN 21682267. doi: 10.1109/TCYB.2018.2801476.

J. Liu and Y. Jia. A Lateral Inhibitory Spiking Neural Network for Sparse Representation in Visual Cortex. *Adv. Brain Inspired Cogn. Syst.*, 7366:259–267, 2012. doi: 10.1007/978-3-642-31561-9.

Q. Liu, G. Pineda-Garcia, E. Stromatias, T. Serrano-Gotarredona, and S. B. Furber. Benchmarking spike-based visual recognition: A dataset and evaluation. *Front. Neurosci.*, 10(NOV), 2016. ISSN 1662453X. doi: 10.3389/fnins.2016.00496.

S. Löwe, P. O'Connor, and B. S. Veeling. Putting An End to End-to-End: Gradient-Isolated Learning of Representations. *Advances in neural information processing systems*, 2019.

E. Majani, R. Erlarson, and Y. Abu-Mostafa. On the k-winners-takes-all network. *Adv. Neural Inf. Process. Syst. I*, pages 634–642, 1989. ISSN 978-0262561457.

G. Major, M. E. Larkum, and J. Schiller. Active properties of neocortical pyramidal neuron dendrites. *Annual review of neuroscience*, 36:1–24, jul 2013. ISSN 1545-4126. doi: 10.1146/annurev-neuro-062111-150343. URL http://www.ncbi.nlm.nih.gov/pubmed/23841837.

A. H. Marblestone, G. Wayne, and K. P. Kording. Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 10, 2016. URL http://dx.doi.org/10.3389/fncom.2016.00094.

H. Markram, W. Gerstner, and P. J. Sjöström. A history of spike-timing-dependent plasticity. *Frontiers in synaptic neuroscience*, 3:4, 2011.

T. Masquelier and S. J. Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Comput Biol*, 3(2):e31, 2007.

T. Masquelier, T. Serre, S. Thorpe, and T. Poggio. Learning complex cell invariance from natural videos: A plausibility proof. Technical report, Technical report of the MIT computer science and artificial intelligence lab, 2007.

A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience*, 21(9):1281–1289, 2018.

G. Matteucci and D. Zoccolan. Unsupervised experience with temporal continuity of the visual environment is causally involved in the development of v1 complex cells. *Science advances*, 6(22):eaba3742, 2020.

J. M. McFarland, A. G. Bondy, R. C. Saunders, B. G. Cumming, and D. A. Butts. Saccadic modulation of stimulus processing in primary visual cortex. *Nature communications*, 6(1): 1–14, 2015.

M. M. Merzenich, R. J. Nelson, M. P. Stryker, M. S. Cynader, A. Schoppmann, and J. M. Zook. Somatosensory cortical map changes following digit amputation in adult monkeys. *Journal of comparative Neurology*, 224(4):591–605, 1984.

T. Miconi. Multi-layer hebbian networks with modern deep learning frameworks. *arXiv preprint arXiv:2107.01729*, 2021.

K. D. Miller. A model for the development of simple cell receptive fields and the ordered arrangement of orientation columns through activity-dependent competition between on-and off-center inputs. *Journal of Neuroscience*, 14(1):409–441, 1994.

K. D. Miller and F. Fumarola. Mathematical Equivalence of Two Common Forms of Firing-Rate Models of Neural Networks. *Neural Comput*, 24(1):25–31, 2012. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3237837/pdf/nihms313491.pdf.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. URL http://dx.doi.org/10.1038/nature14236.

L. Molgedey and H. G. Schuster. Separation of a mixture of independent signals using time delayed correlations. *Physical review letters*, 72(23):3634, 1994.

T. H. Moskovitz, A. Litwin-kumar, and L. Abbott. Feedback alignment in deep convolutional networks. *arXiv Neural Evol. Comput.*, pages 1–10, 2018. doi: arXiv:1812.06488v1. URL http://arxiv.org/abs/1812.06488.

D. Mumford. Pattern theory: the mathematics of perception. I, 2002. doi: arXiv:math/0212400. URL http://arxiv.org/abs/math/0212400.

M. R. Nassar, K. M. Rumsey, R. C. Wilson, K. Parikh, B. Heasly, and J. I. Gold. Rational regulation of learning dynamics by pupil-linked arousal systems. *Nature neuroscience*, 15(7):1040–1046, 2012.

R. Naud, N. Marcille, and C. Clopath. Firing patterns in the adaptive exponential integrate-and-fire model. *Biol. Cybern.*, pages 335–347, 2008. doi: 10.1007/s00422-008-0264-7.

R. A. Nawrocki, R. M. Voyles, and S. E. Shaheen. A Mini Review of Neuromorphic Architectures and Implementations. *IEEE Trans. Electron Devices*, 63(10):3819–3829, 2016. ISSN 00189383. doi: 10.1109/TED.2016.2598413.

E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs. Event-driven contrastive divergence for spiking neuromorphic systems. *Front. Neurosci.*, 7(8 JAN):1–14, 2014. ISSN 1662453X. doi: 10.3389/fnins.2014.00272.

E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Front. Neurosci.*, 11(JUN):1–18, 2017. ISSN 1662453X. doi: 10.3389/fnins.2017.00324.

A. Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in neural information processing systems*, pages 1037–1045, 2016.

A. Nøkland and L. H. Eidnes. Training Neural Networks with Local Error Signals. *International Conference on Machine Learning*, 2019. doi: arXiv:1901.06656v1.

D. Obeid, H. Ramambason, and C. Pehlevan. Structured and Deep Similarity Matching via Structured and Deep Hebbian Networks. In *NeurIPS*, 2019. URL http://arxiv.org/abs/1910.04958.

P. O'Connor, D. Neil, S. C. Liu, T. Delbruck, and M. Pfeiffer. Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci.*, 7(7 OCT):1–13, 2013. ISSN 16624548. doi: 10.3389/fnins.2013.00178.

P. O'Connor, E. Gavves, and M. Welling. Temporally Efficient Deep Learning with Spikes. *arXiv Prepr.*, (NIPS), 2017. URL http://arxiv.org/abs/1706.04159.

E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 1:267–273, 1982.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996. ISSN 0028-0836. doi: 10.1038/381607a0. URL http://dx.doi.org/10.1038/381607a0.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Res.*, 37(23):3311–3325, 1997. ISSN 00426989. doi: 10.1016/S0042-6989(97)00169-7.

D. Ostwald, B. Spitzer, M. Guggenmos, T. T. Schmidt, S. J. Kiebel, and F. Blankenburg. Evidence for neural encoding of bayesian surprise in human somatosensation. *NeuroImage*, 62(1):177–188, 2012.

V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

P. Panda and K. Roy. Unsupervised Regenerative Learning of Hierarchical Features in Spiking Deep Networks for Object Recognition. *arXiv Prepr.*, 2016. URL https://arxiv.org/abs/1602.01510.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

A. B. Patel and R. G. Baraniuk. A Probabilistic Framework for Deep Learning. *Conf. Neural Inf. Process. Syst. NIPS*, (Nips), 2016.

A. Payeur, J. Guerguiev, F. Zenke, B. A. Richards, and R. Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature neuroscience*, pages 1–10, 2021.

C. Pehlevan and D. B. Chklovskii. A Hebbian/Anti-Hebbian network derived from online non-negative matrix factorization can cluster and discover sparse features. *Conf. Rec. - Asilomar Conf. Signals, Syst. Comput.*, 2015-April:769–775, 2015. ISSN 10586393. doi: 10.1109/ACSSC.2014.7094553.

C. Pehlevan, A. M. Sengupta, and D. B. Chklovskii. Why do similarity matching objectives lead to hebbian/anti-hebbian networks? *Neural computation*, 30(1):84–124, 2018.

R. A. Poldrack. The future of fmri in cognitive neuroscience. *Neuroimage*, 62(2):1216–1220, 2012.

J. Poort, A. G. Khan, M. Pachitariu, A. Nemri, I. Orsolic, J. Krupic, M. Bauza, M. Sahani, G. B. Keller, T. D. Mrsic-Flogel, et al. Learning enhances sensory and multiple non-sensory representations in primary visual cortex. *Neuron*, 86(6):1478–1490, 2015.

I. Pozzi, S. M. Bohté, and P. R. Roelfsema. A Biologically Plausible Learning Rule For Deep Learning In The Brain. *arXiv Prepr.*, pages 1–14, 2018.

I. Pozzi, S. M. Bohté, and P. R. Roelfsema. Attention-Gated Brain Propagation: How the brain can implement reward-based error backpropagation. In *NeurIPS*, 2020.

D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat. Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Trans. Nanotechnol.*, 12(3):288–295, 2013. ISSN 1536125X. doi: 10.1109/TNANO.2013.2250995.

R. P. Rao and D. H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.*, 2(1), 1999. URL http://neurosci.nature.com.

E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized Evolution for Image Classifier Architecture Search. *arXiv Prepr.*, 2018. URL http://arxiv.org/abs/1802.01548.

G. H. Recanzone, C. E. Schreiner, and M. M. Merzenich. Plasticity in the frequency representation of primary auditory cortex following discrimination training in adult owl monkeys. *Journal of Neuroscience*, 13(1):87–103, 1993.

M. Rehn and F. T. Sommer. A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. *Journal of computational neuroscience*, 22(2):135–146, 2007.

B. A. Richards, T. P. Lillicrap, P. Beaudoin, Y. Bengio, R. Bogacz, A. Christensen, C. Clopath, R. P. Costa, A. de Berker, S. Ganguli, et al. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770, 2019.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nat. Neurosci.*, 2(11):1019–25, 1999. ISSN 1097-6256. doi: 10.1038/14819. URL http://www.ncbi.nlm.nih.gov/pubmed/10526343.

L. Robinson and E. T. Rolls. Invariant visual object recognition: biologically plausible approaches. *Biol. Cybern.*, 109(4-5):505–535, 2015. ISSN 14320770. doi: 10.1007/s00422-015-0658-2.

P. R. Roelfsema and A. Holtmaat. Control of synaptic plasticity in deep cortical networks. *Nat. Rev. Neurosci.*, 19, 2018. doi: 10.1038/nrn.2018.6. URL www.nature.com/nrn.

E. T. Rolls and T. Milward. A model of invariant object recognition in the visual system: learning rules, activation functions, lateral inhibition, and information-based performance measures. *Neural computation*, 12(11):2547–2572, 2000.

J. O. Rombouts, S. M. Bohte, and P. R. Roelfsema. How Attention Can Create Synaptic Tags for the Learning of Working Memories in Sequential Tasks. *PLoS Comput. Biol.*, 11(3):1–34, 2015. ISSN 15537358. doi: 10.1371/journal.pcbi.1004060.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, 65(6):386–408, 1958. ISSN 0033-295X. doi: 10.1037/h0042519.

J. Ross, M. C. Morrone, M. E. Goldberg, and D. C. Burr. Changes in visual perception at the time of saccades. *Trends in neurosciences*, 24(2):113–121, 2001.

C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10):2526–2563, 2008.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. ISSN 0028-0836. doi: 10.1038/323533a0. URL http://www.nature.com/doifinder/10.1038/323533a0.

S. Russell. *Human compatible: Artificial intelligence and the problem of control*. Penguin, 2019.

S. Russell and P. Norvig. Artificial intelligence: a modern approach. 2002.

J. Sacramento, R. P. Costa, Y. Bengio, and W. Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in neural information processing systems*, pages 8721–8732, 2018.

A. Samadi, T. P. Lillicrap, and D. B. Tweed. Deep Learning with Dynamic Spiking Neurons and Fixed Feedback Weights. *Neural Comput.*, 29:578–602, 2017. ISSN 1530888X. doi: 10.1162/NECO.

J. N. Sanes and J. P. Donoghue. Plasticity and primary motor cortex. *Annual review of neuroscience*, 23(1):393–415, 2000.

T. D. Sanger. Optimal unsupervised learning in a single-layered linear feedforward network. *Neural Networks*, 2:459–473, 1989.

B. Scellier and Y. Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

W. Schultz, P. Dayan, and P. R. Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

P. Schwartenbeck, T. FitzGerald, R. Dolan, and K. Friston. Exploration, novelty, surprise, and free energy minimization. *Frontiers in psychology*, 4:710, 2013.

T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3):411–426, 2007. ISSN 01628828. doi: 10.1109/TPAMI.2007.56.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. URL http://www.robots.ox.ac.uk/.

P. J. Sjöström, G. G. Turrigiano, and S. B. Nelson. Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron*, 32(6):1149–1164, 2001.

K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

C. J. Spoerer, P. McClure, and N. Kriegeskorte. Recurrent convolutional neural networks: A better model of biological object recognition. *Front. Psychol.*, 8(SEP):1–14, 2017. ISSN 16641078. doi: 10.3389/fpsyg.2017.01551.

H. Sprekeler and L. Wiskott. A theory of slow feature analysis for transformation-based input signals with an application to complex cells. *Neural Comput.*, 23(2):303–35, 2011. ISSN 1530-888X. doi: 10.1162/NECO_a_00072. URL http://www.ncbi.nlm.nih.gov/pubmed/21105830.

H. Sprekeler, C. Michaelis, and L. Wiskott. Slowness: An Objective for Spike-Timing–Dependent Plasticity? *PLoS Comput. Biol.*, 3(6):e112, 2007. ISSN 1553-734X. doi: 10.1371/journal.pcbi.0030112. URL http://dx.plos.org/10.1371/journal.pcbi.0030112.

N. A. Steinmetz, C. Koch, K. D. Harris, and M. Carandini. Challenges and opportunities for large-scale electrophysiology with neuropixels probes. *Current opinion in neurobiology*, 50: 92–100, 2018.

G. S. Stent. A physiological mechanism for hebb's postulate of learning. *Proceedings of the National Academy of Sciences*, 70(4):997–1001, 1973.

M. P. Stryker and W. A. Harris. Binocular impulse blockade prevents the formation of ocular dominance columns in cat visual cortex. *Journal of Neuroscience*, 6(8):2117–2133, 1986.

B. Stuhr and J. Brauer. CSNNs: Unsupervised, backpropagation-free convolutional neural networks for representation learning. *Proc. - 18th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2019*, pages 1613–1620, 2019. doi: 10.1109/ICMLA.2019.00265.

T. J. Sullivan and V. R. de Sa. A temporal trace and som-based model of complex cell development. *Neurocomputing*, 58:827–833, 2004.

J. Talloen, J. Dambre, and A. Vandesompele. Pytorch-hebbian: facilitating local learning in a deep learning framework. *arXiv preprint arXiv:2102.00428*, 2021.

A. Tavanaei and A. S. Maida. Bio-Inspired Spiking Convolutional Neural Network using Layer-wise Sparse Coding and STDP Learning. *arXiv Prepr.*, (1611.03000v2):1–20, 2016. URL http://arxiv.org/abs/1611.03000.

A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. S. Maida. Deep Learning in Spiking Neural Networks. *Neural Networks*, 111:47–63, 2018. ISSN 18792782. doi: 10.1016/j.neunet.2014.09.003. URL http://arxiv.org/abs/1804.08150.

J. C. Thiele, O. Bichler, and A. Dupret. Event-based, timescale invariant unsupervised online deep learning with STDP. *Front. Comput. Neurosci.*, 12(June):46, 2018. ISSN 1662-5188. doi: 10.3389/FNCOM.2018.00046. URL https://www.frontiersin.org/articles/10.3389/fncom.2018.00046/abstract.

A. M. Turing and J. Haugeland. *Computing machinery and intelligence*. MIT Press Cambridge, MA, 1950.

G. G. Turrigiano, K. R. Leslie, N. S. Desai, L. C. Rutherford, and S. B. Nelson. Activity-dependent scaling of quantal amplitude in neocortical neurons. *Nature*, 391(6670):892–896, 1998.

R. Urbanczik and W. Senn. Reinforcement learning in populations of spiking neurons. *Nature neuroscience*, 12(3):250–252, 2009.

R. Urbanczik and W. Senn. Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528, 2014.

A. Van den Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv Prepr.*, 2018.

L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

C. Von der Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14(2):85–100, 1973.

G. Wallis and E. T. Rolls. Invariant face and object recognition in the visual system. *Progress in neurobiology*, 51(2):167–194, 1997.

J. C. Whittington and R. Bogacz. An Approximation of the Error Backpropagation Algorithm in a Predictive Coding Network with Local Hebbian Synaptic Plasticity. *Neural Comput.*, 29: 1229–1262, 2017. ISSN 1530888X. doi: 10.1162/NECO.

J. C. R. Whittington and R. Bogacz. Theories of Error Back-Propagation in the Brain. *Trends Cogn. Sci.*, xx:1–16, 2019. ISSN 1364-6613. doi: 10.1016/j.tics.2018.12.005. URL https://doi.org/10.1016/j.tics.2018.12.005.

T. N. Wiesel and D. H. Hubel. Single-cell responses in striate cortex of kittens deprived of vision in one eye. *Journal of neurophysiology*, 26(6):1003–1017, 1963.

L. Wiskott and T. J. Sejnowski. Slow Feature Analysis : Unsupervised Learning of Invariances. *Neural Comput.*, 770:715–770, 2002.

Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi. Direct Training for Spiking Neural Networks: Faster, Larger, Better. *arXiv Prepr.*, 2018. URL http://arxiv.org/abs/1809.05793.

T. Wunderlich, A. F. Kungl, E. Müller, A. Hartel, Y. Stradmann, S. A. Aamir, A. Grübl, A. Heimbrecht, K. Schreiber, D. Stöckel, et al. Demonstrating advantages of neuromorphic computation: a pilot study. *Frontiers in neuroscience*, 13:260, 2019.

X. Xie and H. S. Seung. Equivalence of backpropagation and contrastive hebbian learning in a layered network. *Neural computation*, 15(2):441–454, 2003.

Y. Xiong, M. Ren, and R. Urtasun. LoCo: Local Contrastive Representation Learning. *Advances in neural information processing systems*, 2020.

D. L. Yamins and J. J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nat. Neurosci.*, 19(3):356–365, 2016. ISSN 15461726. doi: 10.1038/nn.4244.

D. L. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014.

J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding Neural Networks Through Deep Visualization. 2015. URL http://arxiv.org/abs/1506.06579.

A. M. Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):1–7, 2019.

B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang. Feedforward Categorization on AER Motion Events Using Cortex-Like Features in a Spiking Neural Network. *IEEE Trans. Neural Networks Learn. Syst.*, 26(9):1963–1978, 2015. ISSN 21622388. doi: 10.1109/TNNLS.2014.2362542.

C. Zhuang, A. L. Zhai, and D. Yamins. Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6002–6012, 2019.

C. Zhuang, S. Yan, A. Nayebi, M. Schrimpf, M. C. Frank, J. J. DiCarlo, and D. L. Yamins. Unsupervised neural network models of the ventral visual stream. *Proceedings of the National Academy of Sciences*, 118(3), 2021.

A. Ziehe and K.-R. Müller. Tdsep—an efficient algorithm for blind separation using time structure. In *International Conference on Artificial Neural Networks*, pages 675–680. Springer, 1998.

J. Zylberberg, J. T. Murphy, and M. R. DeWeese. A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields. *PLoS Comput. Biol.*, 7(10), 2011. ISSN 1553734X. doi: 10.1371/journal.pcbi.1002250.

BERND ILLING

bernd.illing@protonmail.com

Profile

Hi! I am a PhD student at EPFL (Gerstner lab), working at the intersection of computational neuroscience and AI, with a background in theoretical and computational physics. I enjoy the interdisciplinary challenge of advancing research in neuroscience with AI methods and vice versa: I develop biologically plausible learning methods for deep neural networks that scale to AI tasks. My faithful accomplices on this mission are: mathematical modelling, Python, PyTorch & Julia, spiking neuron simulators (custom or brian2) and scientific computing clusters. I am excited about working in interdisciplinary teams where people learn from each other every day.

Education

- **EPFL, Switzerland—PhD Program in Neuroscience, supervised by Prof. Gerstner, 2016—2021**
- **Computer Vision Summer School (CVSS) 2019,** Freudenstadt, Germany, 1-week International computer vision intensive training programme
- **CAJAL Advanced Computational Neuroscience Training Programme 2018,** 3-week course, Champalimaud Centre, Lisbon
- **University of Constance, Germany—B.S. & M.S. in Physics, 2010—2015**

Human speak

English (fluent) — **French** (conversational) — **German** (native) — **Arabic** (basic)

Computer speak

Python: PyTorch, Brian2, TensorFlow, scikit-learn — **Julia:** custom-built event and rate-based spiking neuron simulator (available on GitHub)— **Matlab** — **C++** **Unix shell essentials** — **scientific computing:** GPU cluster, docker, kubernetes, runai — **software development:** git, unit testing, continuous integration, packaging (pip, conda) — **presentation:** Latex, matplotlib, office

Research experience

- **PhD, Gerstner lab, EPFL — 2016—2021, thesis title:** *Biologically plausible unsupervised learning in shallow and deep neural networks*

  Main methods: Computer simulations of learning and dynamics of rate-based and spiking neural networks (Python, PyTorch, brian2, Julia), statistical learning (PCA, sparse coding, clustering), deep learning (CNN, RNN, backprop, optimisation, regularisation), mathematical modelling of synaptic plasticity (Hebbian rules, Spike-timing dependent plasticity (STDP)), dynamical systems (fixed points, phase plane)
- **Research Assistent, University of Constance — 2016 Main methods:** data analysis, signal processing, theoretical and numerical modelling

- **Master project, University of Constance — 2015** *Strain and stress correlations in colloidal glass formers* **Main methods:** Statistical and computational physics, object (colloid) detection, data analysis
- **Bachelor project, University of Constance — 2013** *Graphene nano-flakes on Ag(111) surfaces studied by scanning tunnelling microscopy*

Publications (Google Scholar)

- [under review] B. Illing, J. Ventura, G. Bellec & W. Gerstner, *Local plasticity rules can learn deep representations using self-supervised contrastive predictions* (2021)
- [under review] W. Wybo, M. Tsai, B. Illing, J. Jordan, A. Morrison & W. Senn *Biologically plausible and data efficient multi-task learning through contextual bias adaptation* (2021)
- B. Illing, W. Gerstner & G. Bellec, *Towards truly local gradients with CLAPP: Contrastive, Local And Predictive Plasticity.* **arXiv,** (2020).
- B. Illing, W. Gerstner & J. Brea, *Biologically plausible deep learning – but how far can we go with shallow networks?*, **Neural Networks** 118 (2019)
- J. Brea, B. Simsek, B. Illing & W. Gerstner, *Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape*, **arXiv,** (2019)
- B. Illing et al., *Mermin–Wagner fluctuations in 2D amorphous solids*, **PNAS** 114 8 (2017)
- B. Illing et al., *Strain Pattern in Supercooled Liquids*, **PRL** 117 208002 (2016)

Presentations

- Poster @ NeurIPS workshop *Beyond Backpropagation* 2020, virtual
- Poster @ EPFL Symposium *Neuroscience meets deep learning* 2019
- Poster @ Computer Vision Summer School (CVSS) 2019, Black Forest
- Poster @ Cosyne 2019, Lisbon
- Talk @ Human Brain Project SP4 meeting 2019, EITN Paris
- Poster @ Lemanic Neuroscience Annual Meeting (LNAM) 2017

Reviewing for the journals **Neural Computation, Neural Networks, JMLR**

Awards

- Firmenich Stanford-EPFL fellowship (2020 - canceled due to pandemic)
- Premium for content creating of two MOOCs on the platform edX (2018)
- Award for the best master thesis in physics, University of Constance (2016)

Teaching

- Supervision of one master thesis and three semester project students
- Co-designer of two MOOC courses (2017—2018): Neuronal Dynamics & Computational Neuroscience: Neuronal Dynamics of Cognition
- Teaching assistant at EPFL (Gerstner), courses: *Artificial Neural Networks* (2019—2020) and *Biological Modelling of Neuronal Networks* (2017—2018)
- Teaching assistant at the University of Constance for physics and mathematics courses for biologists (2013—2016)

Hors catégorie

If it's not about work, I like to talk mountains, bikes, music and coffee ;)