Neural Networks 118 (2019) 90–101

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

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

Bernd Illing [*], Wulfram Gerstner, Johanni Brea

School of Computer and Communication Science & School of Life Science, EPFL, 1015 Lausanne, Switzerland

ARTICLE INFO

Article history:
Received 7 February 2019
Received in revised form 29 May 2019
Accepted 2 June 2019
Available online 20 June 2019

Keywords:
Deep learning
Local learning rules
Random projections
Unsupervised feature learning
Spiking networks
MNIST

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 data sets other than MNIST for testing the performance of future models of biologically plausible deep learning.

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, Bengio, & Hinton, 2015). DNNs and the cerebral cortex share that information is processed in multiple layers of many neurons (Kriegeskorte, 2015; Yamins & DiCarlo, 2016) 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, Wayne, & Kording, 2016; Whittington & 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, Voyles, & Shaheen, 2016).

Image recognition is a popular task to test the performance of neural networks. Because of its relative simplicity and popularity, the MNIST data set (28 × 28-pixel gray 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 and Table 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

[*] Corresponding author.
E-mail address: bernd.illing@epfl.ch (B. Illing).

https://doi.org/10.1016/j.neunet.2019.06.001
0893-6080/

Table 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−...	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

empirical signature of backpropagation-like deep learning (Lillicrap, Cownden, Tweed, & Akerman, 2016; Sacramento, Costa, Bengio, & Senn, 2017; Tavanaei, Ghodrati, Kheradpisheh, Masquelier, & Maida, 2018; Whittington & 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, Santoro, Richards, Hinton, & Lillicrap, 2018) or at least need major modifications to scale to deeper networks (Moskovitz, Litwin-kumar, & Abbott, 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, Zhu, & Siew, 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 & Oja, 1998) or algorithms in Brito and Gerstner (2016), Liu and Jia (2012), Olshausen and Field (1997), Rozell, Johnson, Baraniuk, and Olshausen (2008) 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 & Senn, 2014). Since straightforward stacking of multiple fully connected layers of unsupervised learning does not reveal more complex features (Olshausen & 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 (Section 3.1). Since sparse connectivity is sometimes superior to dense connectivity (Bartunov et al., 2018; Litwin-Kumar, Harris, Axel, Sompolinsky, & Abbott, 2017) 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 (Section 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 (Section 3.3). We discuss the performance and implications of this simplistic model with respect to current models of biologically plausible deep learning.

2. Related work

In recent years, many biologically plausible approaches to deep learning have been proposed, see e.g. (Marblestone et al., 2016; Tavanaei et al., 2018; Whittington & Bogacz, 2019) for reviews. Existing approaches usually use either involved architectures or elaborate mechanisms to approximate the backpropagation algorithm. Examples include the use of convolutional layers (Kheradpisheh et al., 2018; Lee, Srinivasan, Panda, & Roy, 2018; Tavanaei et al., 2018; Tavanaei & Maida, 2016) (and tables therein), dendritic computations (Guergiuev et al., 2016; Hussain et al., 2014; Sacramento et al., 2017) or backpropagation approximations such as feedback alignment (Baldi et al., 2016; Bartunov et al., 2018; Kohan et al., 2018; Lillicrap et al., 2016; Nøkland, 2016; Samadi et al., 2017) equilibrium propagation (Scellier & Bengio, 2017), membrane potential based backpropagation (Lee et al., 2016), restricted Boltzmann machines and deep belief networks (Neftci et al., 2014; O'Connor et al., 2013), (localized) difference target propagation (Bartunov et al., 2018; Lee et al., 2015), using reinforcement-signals (Pozzi et al., 2018; Rombouts, Bohte, & Roelfsema, 2015) or approaches using predictive coding (Whittington & Bogacz, 2017). Many models implement spiking neurons to stress bio-plausibility (Kulkarni & Rajendran, 2018; Liu, Pineda-Garcia, Stromatias, Serrano-Gotarredona, & Furber, 2016; Liu & Yue, 2018; Neftci et al., 2017; Tavanaei et al., 2018; Wu 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 (Bellec, Salaj, Subramoney, Legenstein, & Maass, 2018; Spoerer, McClure, & Kriegeskorte, 2017), starting directly from realistic circuits (Delahunt & Kutz, 2018), or combining unsupervised and supervised training (Krotov et al., 2019) as in this paper. We refer to Table 2 for an extensive list of current biologically plausible models tested on MNIST (see Table 1 for abbreviations).

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 $\mathbf{W}_2$ (Fig. 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 $\mathbf{W}_2^T$ (Fig. 1a). In the biologically plausible network considered in this paper (Fig. 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 Appendix A–Appendix D. Code for all (rate & spiking) models discussed below is publicly available at https://github.com/EPFL-LCN/pub-illing2019-nnetworks.

Table 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 3. Models are ranked by MNIST test accuracy (rightmost column). Parts of this table are taken from Diehl and Cook (2015), Kheradpisheh, Ganjtabesh, Thorpe, and Masquelier (2018), Tavanaei et al. (2018). Models using convolutional layers (CNN) are marked in *italic*. See Table 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, Deng, Li, Zhu, & Shi, 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 & 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, Delbruck, & Pfeiffer, 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, Zhang, Fischer, & Bengio, 2015)	Rate	Supervised	Layer-wise AE, Target Prop.	98.5
Nonlin. Hebb + SGD (Krotov, Hopfield, & Lee, 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, Gavves, & Welling, 2017)	Pseudo-spike	Supervised	Sparse, discrete activities, STDP	98.3
Direct FA (Nøkland, 2016)	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, Bohté, & Roelfsema, 2018)	Rate	RL-like	RL-like BP-approx.	98.2
Forward propagation (FP) (Kohan, Rietman, & Siegelmann, 2018)	Rate	Supervised	FP: BP approximation	98.1
Spiking FA (Neftci, Augustine, Paul, & Detorakis, 2017)	Spikes	Supervised	Direct FA	98
Predictive coding (Whittington & Bogacz, 2017)	Rate	Supervised	BP approx. by pred. coding	98
Spiking CNN (Tavanaei & Maida, 2016)	Rate/Spikes	Unsupervised	Semi-online, STDP, ext. SVM	*98*
Equilibrium Prop. (Scellier & Bengio, 2017)	Rate	Supervised	1–3 hidden layers	97–98
Dendr. BP (Sacramento et al., 2017)	Spikes	Supervised	Dendritic comp. for BP approx.	97.5
Spiking FA (Samadi, Lillicrap, & Tweed, 2017)	Spikes	Supervised	3 hidden layers	97
Sparse/Skip FA (Baldi, Sadowski, & Lu, 2016)	Rate	Supervised	Sparse- & Skip-FA	96–97
Spiking CNN (Thiele, Bichler, & Dupret, 2018)	Spikes	Unsupervised	Recurrent Inhib., STDP	*96.6*
Spiking FA (Guergiuev, Lillicrap, & Richards, 2016)	Spikes	Supervised	Dendritic comp. for BP approx.	96.3
2 layer network (Diehl & Cook, 2015)	Spikes	Unsupervised	Recurrent Inhib., purely unsuperv.	95
Spiking RBM/DBN (O'Connor, Neil, Liu, Delbruck, & Pfeiffer, 2013)	Rate	Supervised	Conversion rate → spike	94.1
2 layer network (Querlioz, Bichler, Dollfus, & Gamrat, 2013)	Spikes	Unsupervised	Memristive device	93.5
Spiking HMAX/CNN (Liu & Yue, 2018)	Spikes	Supervised	STDP, HMAX preprocess.	*93*
Spiking RBM/DBN (Neftci, Das, Pedroni, Kreutz-Delgado, & Cauwenberghs, 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, Ding, Chen, Linares-Barranco, & Tang, 2015)	Spike	Supervised	Tempotron rule, sensor MNIST	*91.3*
Dendritic neurons (Hussain, Liu, & Basu, 2014)	Rate	Supervised	Nonlin. dendrites, neuromorphic appl.	90.3

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 Fig. 1 and methods Appendix A–Appendix C 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 Fig. 1b & blue curves in Fig. 2, or using one of the unsupervised methods Principal Component Analysis (**PCA**), Independent Component Analysis (**ICA**) or Sparse Coding (**SC**), see Fig. 1c & red curves in Fig. 2. All these methods can be implemented with local, biologically plausible learning rules (Hyvärinen & Oja, 1998; Oja, 1982; Olshausen & Field, 1997). We refer to the methods Appendix B for further details. As a reference, we train networks with the same architecture with standard backpropagation (**BP**, see Fig. 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 Appendix D 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 (Fig. 2). The PCA (red dashed) and ICA (red dotted) curves in Fig. 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.

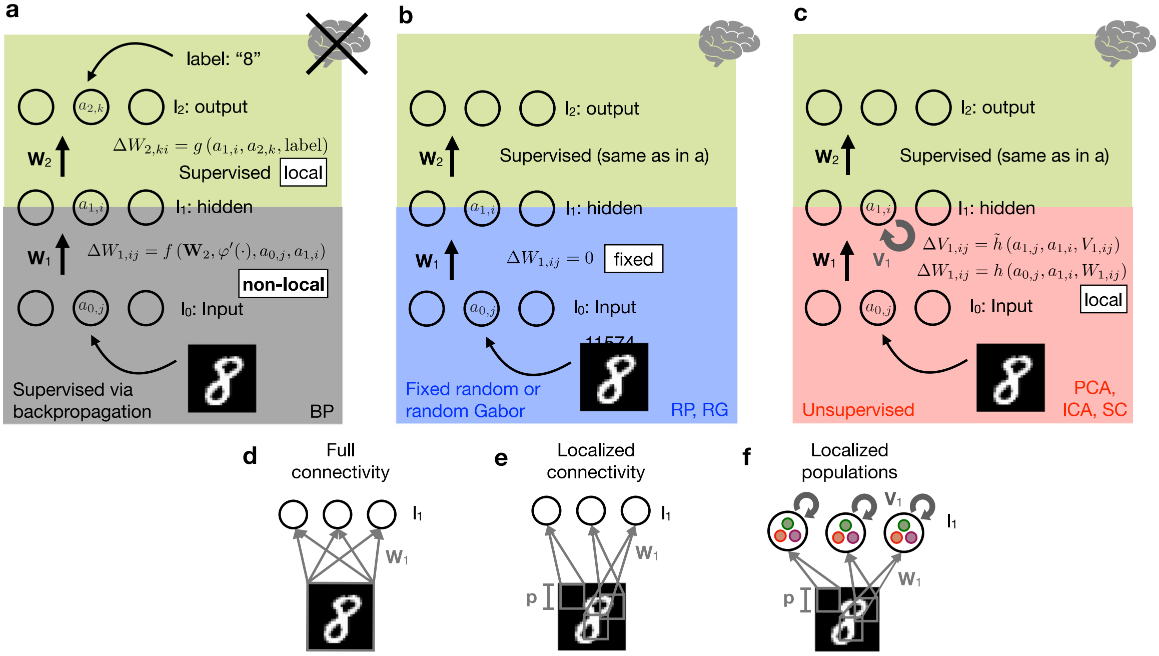
a
label: "8"
$a_{2,k}$
l2: output
W2
$\Delta W_{2,ki} = g\left(a_{1,i}, a_{2,k}, \text{label}\right)$
Supervised
local
$a_{1,i}$
l1: hidden
W1
$\Delta W_{1,ij} = f\left(\mathbf{W}_2, \varphi'(\cdot), a_{0,j}, a_{1,i}\right)$
non-local
$a_{0,j}$
l0: Input
Supervised via backpropagation
BP
b
l2: output
W2
Supervised (same as in a)
l1: hidden
W1
$\Delta W_{1,ij} = 0$
fixed
l0: Input
11574
Fixed random or random Gabor
RP, RG
c
l2: output
W2
Supervised (same as in a)
l1: hidden
W1
V1
$\Delta V_{1,ij} = \tilde{h}\left(a_{1,j}, a_{1,i}, V_{1,ij}\right)$
$\Delta W_{1,ij} = h\left(a_{0,j}, a_{1,i}, W_{1,ij}\right)$
local
l0: Input
Unsupervised
PCA, ICA, SC
d
Full connectivity
l1
W1
e
Localized connectivity
l1
W1
p
f
Localized populations
V1
l1
W1
p

Fig. 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 Appendix D). **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 Appendix A–Appendix D. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

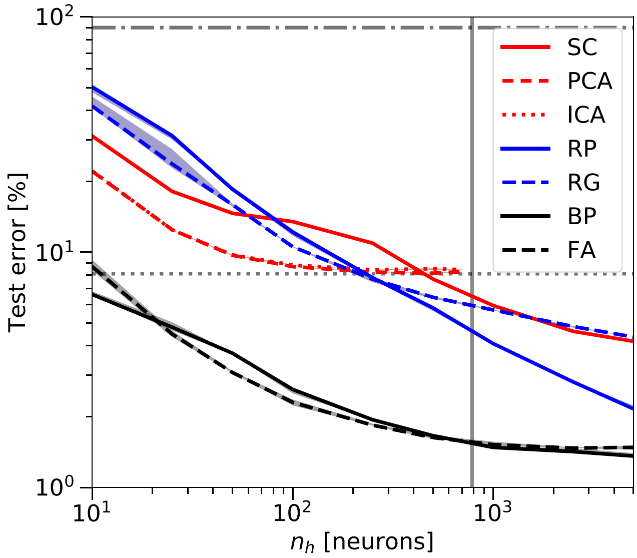

Fig. 2. MNIST classification with rate networks according to Fig. 1a-c with full connectivity (Fig. 1d). The test error decreases for increasing hidden layer size n_h for all methods, i.e. Principal/Independent Component Analysis (PCA/ICA, curves are highly overlapping), Sparse Coding (SC), fixed Random 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 Appendix C). 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 Fig. 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, 1993; Barron, Brandy, & Yale, 1994)). In the log–log-plot in Fig. 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 A). We find a much faster decay 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 Section 3.2 for further analysis on CIFAR10).

3.2. Localized receptive fields boost performance

There are good reasons to reduce the connectivity from all-to-all to localized receptive fields (Fig. 1e & f): local connectivity patterns are observed in real neural circuits (Hubel & 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 Fig. 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 Fig. 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 Appendix B).

We test *l*-RP for different patch sizes p and find an optimum around $p \approx 10$ (see Fig. 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 Fig. 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 generalize 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 & 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 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, Aggarwal, Huang, & Le, 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). 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.

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 Fig. 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 Section 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 Appendix E) 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 $\mathrm{tgt}_i(t)$

$$\tau_{\mathrm{tr}} \frac{d\mathrm{tr}_i(t)}{dt} = -\mathrm{tr}_i(t) + \sum_f \delta\left(t - t_i^f\right) \qquad (1)$$

$$\Delta w_{2,ij} = \alpha \cdot \left(\mathrm{tgt}_i^{\mathrm{post}}(t) - \mathrm{tr}_i^{\mathrm{post}}(t)\right) \delta\left(t - t_j^f\right),$$

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-synaptic neuron connected to one post-synaptic neuron is integrated for 650 ms. The pre- and post-synaptic membrane potentials show periodic spiking (Fig. 4a) which induces post-synaptic spike traces and corresponding weight changes (Fig. 4b), according to Eq. (1). For the MNIST task, Fig. 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 (see Naud, Marcille, & Clopath, 2008 and references therein). During the testing period, learning is shut off permanently (see methods Appendix E for more details). The LIF and STDP dynamics can be mapped to a rate model (see e.g. Diehl, et al., 2015 and Appendix E for details). However all following results are obtained with the fully spiking LIF/STDP model.

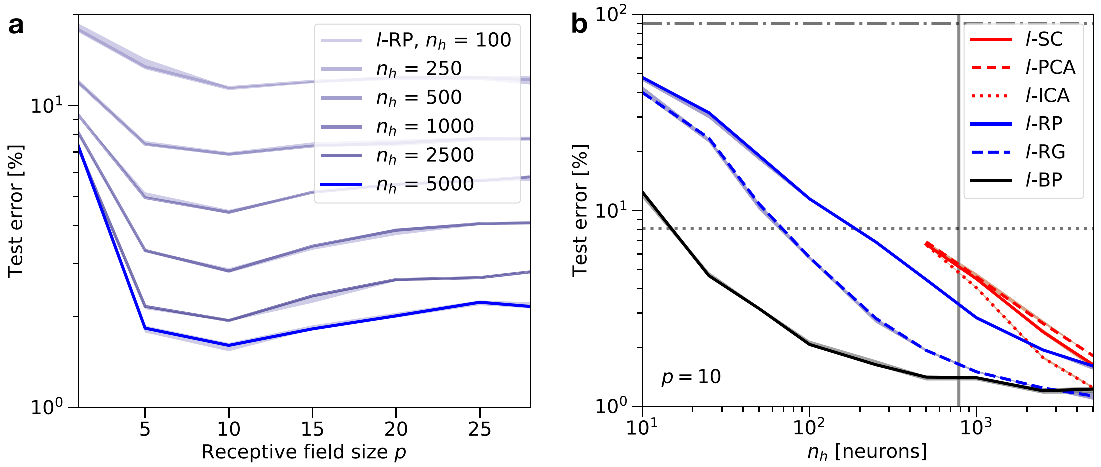

Fig. 3. Effect of localized connectivity on MNIST. **a** Test error for localized Random Projections (l-RP), dependent on receptive field size p for different hidden layer sizes n_h. The optimum for receptive field size p = 10 is more pronounced for large hidden layer sizes. Full connectivity is equivalent to p = 28. Note the log-lin scale. **b** Localized receptive fields decrease test errors for all tested networks (compare Fig. 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 Fig. 2. l-PCA/l-ICA & l-SC use 500 independent populations in the hidden layer (see Fig. 1f) which constrains the hidden layer size to $n_h \geq 500$. Note the log–log scale.

Table 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 $d = 32 \times 32 \times 3 = 3072$ input channels (the third factor is due to the color channels), MNIST only $d = 28 \times 28 = 784$. 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	–

When directly trained with the STDP rule of Eq. (1), the spiking LIF models closely approach the performance of their rate counterparts. Table 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 Section 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 and Table 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.

4. Discussion

In contrast to biologically plausible deep learning algorithms that are derived from approximations of the backpropagation algorithm (Lillicrap et al., 2016; Pozzi et al., 2018; Sacramento et al., 2017; Whittington & Bogacz, 2019), 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, Sheehan, Stevens, and Navlakha (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 and Table 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 & 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

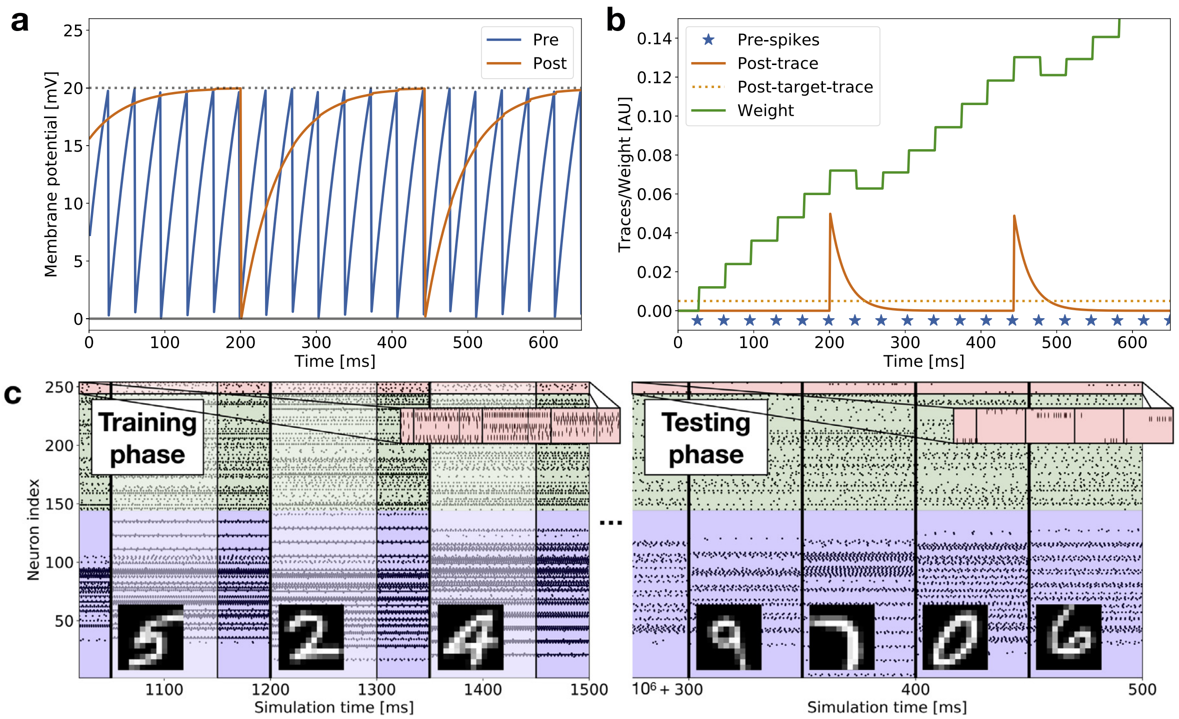

Fig. 4. Spiking LIF and STDP dynamics. **a** Dynamics of the pre- and postsynaptic membrane potentials, spike-traces and the weight value (**b**) of a toy example with two neurons and one interconnecting synapse. The weight decreases when the post-trace is above the post-target-trace (see Eq. (1) and Appendix E). Both neurons receive static supra-threshold 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: Behavior 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 & Roy, 2016), backpropagation approximations (Lee et al., 2016), multiple hidden layers (Lillicrap et al., 2016), dendritic neurons (Sacramento et al., 2017), recurrence (Diehl & 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 data set. 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 data sets, 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 and Table 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.

Acknowledgments

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

Appendix A. 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 Fig. 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

$$\begin{aligned}\mathbf{u}_{l+1} &= \mathbf{W}_{l+1}\mathbf{a}_l + \mathbf{b}_{l+1}\\ \mathbf{a}_{l+1} &= \varphi_{l+1}(\mathbf{u}_{l+1}).\end{aligned}\tag{2}$$

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 Figs. 2 and 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 data set 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.

Appendix B. 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 (Hyvärinen & Oja, 1998; Sanger, 1989). 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:

$$\mathbf{a}_1 = \mathbf{u}_1 = \mathbf{W}_1 \cdot \mathbf{a}_0 \quad \text{with } \mathbf{W}_1 = \mathbf{P} \tag{3}$$

Since the maximum number of principal/independent components that can be extracted is the dimensionality of the data, $n_{\text{max}} = d$, 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 Fig. 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:

$$\begin{aligned}\mathbf{W}^{opt}, \mathbf{a}_1^{opt} &= \text{argmin } \mathcal{L}(\mathbf{W}, \mathbf{a}_1) \\ \mathcal{L}(\mathbf{W}, \mathbf{a}_1) &= \frac{1}{2}\|\mathbf{a}_0 - \mathbf{W}^\top \mathbf{a}_1\|_2^2 + \lambda\|\mathbf{a}_1\|_1.\end{aligned} \tag{4}$$

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 Brito and Gerstner (2016), Olshausen and Field (1997), Pehlevan and Chklovskii (2015), Zylberberg, Murphy, and DeWeese (2011). 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 Brito and Gerstner (2016), Zylberberg et al. (2011):

   $$\begin{aligned}\tau_u \frac{d\mathbf{u}_1(t)}{dt} &= -\mathbf{u}_1(t) + (\mathbf{W}_1\mathbf{a}_0(t) - \mathbf{V}_1\mathbf{a}_1(t)) \\ \mathbf{a}_1(t) &= \varphi(\mathbf{u}_1(t))\end{aligned} \tag{5}$$

   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 & Gerstner, 2016):

   $$\begin{aligned}\Delta W_{1,ji} &= \alpha_w \cdot a_{0,i} \cdot a_{1,j} \\ \Delta V_{1,jk} &= \alpha_v \cdot a_{1,k} \cdot \left(a_{1,j} - \langle a_{1,j}\rangle\right)\end{aligned} \tag{6}$$

   $\langle\cdot\rangle$ 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, Eq. (5) is iterated for N_{iter} steps and the weights are updated according to Eq. (6).

Similar to localized PCA/ICA, the localized version of SC uses independent populations in the hidden layer (see Fig. 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.

Appendix C. 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

$$\mathbf{W}_1 \sim \mathcal{N}(0, \sigma^2) \;\; \sigma^2 = \frac{1}{100\, n_0}$$
$$\mathbf{b}_1 \sim \mathcal{U}([0, 0.1]) \tag{7}$$

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 neighboring 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:

$$\mathbf{W}_{1,\mathrm{rf}} \sim \mathcal{N}(0, \sigma^2_{\mathrm{rf}}) \;\; \sigma^2_{\mathrm{rf}} = \frac{c}{100\, p} \tag{8}$$
$$\mathbf{b}_1 \sim \mathcal{U}([0, 0.1]) \tag{9}$$

where the parameter $c = 3$ was found empirically through a grid-search optimization of classification performance. For exact parameter values, see Table 4.

The (localized) random Gabor filters in RG have the same receptive field structure as in l-RP (see Appendix C) but instead of choosing the weights within the receptive field as random values, they are chosen 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:

$$g\,(x, y; \lambda, \Theta, \psi, \sigma, \gamma) = \tag{10}$$
$$\exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cdot \cos\left(2\pi \frac{x'}{\lambda} + \psi\right)$$
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

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.

Appendix D. 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 (Fig. 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 (Fig. 1a). The forward pass of the model reads:

$$\mathbf{u}_{l+1} = \mathbf{W}_{l+1}\mathbf{a}_l + \mathbf{b}_{l+1}$$
$$\mathbf{a}_{l+1} = \varphi_{l+1}(\mathbf{u}_{l+1}) \tag{11}$$

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

$$\tilde{\mathbf{e}}_L = \mathbf{tgt} - \mathbf{a}_L \tag{12}$$

when minimizing mean squared error (MSE)

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{2}\|\mathbf{tgt} - \mathbf{a}_L\|_2^2 \tag{13}$$

or

$$\mathbf{p} = \mathrm{softmax}\,(\mathbf{a}_L)$$
$$\tilde{\mathbf{e}}_L = \mathbf{tgt} - \mathbf{p} \tag{14}$$

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

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

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. $\Delta W_{l,ij} \propto \frac{\partial \mathcal{L}}{\partial W_{l,ij}}$. The full BP algorithm for deep networks reads (Rumelhart, Hinton, & Williams, 1986):

$$\mathbf{e}_L = \varphi'_L(\mathbf{u}_L) \odot \tilde{\mathbf{e}}_L$$
$$\mathbf{e}_{l-1} = \varphi'_{l-1}(\mathbf{u}_l) \odot \mathbf{W}_l^\top \mathbf{e}_l$$
$$\Delta\mathbf{W}_l = \alpha \cdot \mathbf{e}_l \otimes \mathbf{a}_{l-1}$$
$$\Delta\mathbf{b}_l = \alpha \cdot \mathbf{e}_l \tag{15}$$

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 Eq. (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 Eq. (15) was applied to the weights within the receptive fields. The exact parameter values can be found in Table 4.

Appendix E. 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 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:

$$\tau_m \frac{du_i(t)}{dt} = -u_i(t) + RI_i(t)$$
$$\text{with } \; I_i(t) = I_i^{ff}(t) + I_i^{ext}(t)$$
$$= \sum_{j,f} w_{ij}\epsilon\left(t - t_j^f\right) + I_i^{ext}(t) \tag{16}$$

and the spiking condition: $u_i(t) \geq \vartheta_i$: $u_i \to u_{\mathrm{reset}}$, 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)

Table 4
(Hyper-)Parameters for (l-) BP, FA, RP, RG (apart from weight initialization, see Appendix C) & 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 (≈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 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 Eq. (5)	50
N	Number of iterations for SC	1e5
$\mathbf{W}_l^{\text{init}}$	Feed-forward weight initialization	$W_{l,ij} \sim \mathcal{N}(0,1)/(10\sqrt{n_{l-1}})$
$\mathbf{V}_1^{\text{init}}$	Recurrent weight initialization	**0**
$\mathbf{b}_1^{\text{init}}$	Bias initialization	**0** (and kept fixed)
$\varphi_1(\cdot)$	nonlinearity of hidden SC units	ReLU max(0, · − λ)

Table 6
(Hyper-)Parameters for the spiking LIF l-RP & l-RG models (apart from weight initialization, Appendix C). Input and target amplitudes are implausibly high due to the arbitrary convention $R = 1$ Ω. 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$_{\text{inp}}$	Input amplitude	500 mA
amp$_{\text{tgt}}$	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 (≈117/167 epochs)
$\mathbf{W}_l^{\text{init}}$	Feed-forward weight initialization	$W_{l,ij} \sim \mathcal{N}(0,1) \cdot 20/\sqrt{n_{l-1}}$
$\tilde{\mathbf{W}}_l^{\text{init}}$	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	≤5e−2 ms

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 $\text{tgt}_i(t)$.

$$\tau_{\text{tr}} \frac{d\text{tr}_i(t)}{dt} = -\text{tr}_i(t) + \sum_f \delta\left(t - t_i^f\right) \tag{17}$$

$$\Delta w_{ij} = g\left(\text{tr}_i^{\text{post}}(t), \text{tgt}_i^{\text{post}}(t)\right) \delta\left(t - t_j^f\right)$$

with the plasticity function

$$g\left(\text{tr}_i^{\text{post}}(t), \text{tgt}_i(t)\right) = \alpha \cdot \left(\text{tgt}_i^{\text{post}}(t) - \text{tr}_i^{\text{post}}(t)\right). \tag{18}$$

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$_{\text{inp}}$, the targets **tgt**(t) of the output layer are the one-hot-coded classes, scaled by the target amplitude amp$_{\text{tgt}}$. 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 Eqs. (16) and (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 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 Fig. 4a& b we used static input and target with the parameters amp$_{\text{inp}}$ = 40, 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 Fig. 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:

$$\begin{aligned} \mathbf{u}_l &= \mathbf{W}_l \mathbf{u}_{l-1} + R\mathbf{I}^{ext} \\ \mathbf{a}_l &= \varphi_{\text{LIF}}\left(\mathbf{u}_l\right) \\ \Delta w_{ij} &= \tilde{g}\left(a_j^{\text{pre}}, a_i^{\text{post}}, \text{tgt}_i^{\text{post}}\right) \end{aligned} \tag{19}$$

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

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

The latter can be obtained by integrating the STDP rule of Eq. (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 6). The learning rate 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

$$\alpha = \frac{1000 \text{ ms}}{T_{\text{pat}} \text{ [ms]}} \cdot 1000 \cdot \tilde{\alpha}, \tag{20}$$

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.

Appendix F. Parameter tables

For all simulations, we scaled the learning rate proportional to $1/n_h$ for $n_h > 5000$ to ensure convergence (see Tables 4–6).

References

Baldi, Pierre, Sadowski, Peter, & Lu, Zhiqin (2016). Learning in the machine: random backpropagation and the learning channel. (pp. 1–57). arXiv Prepr., URL http://arxiv.org/abs/1612.02734.

Barron, Andrew R. (1993). Universal approximation bounds for superposition of a sigmoid function. *IEEE Transaction on Information Theory*, [ISSN: 09205691] *39*(3), 930–945. http://dx.doi.org/10.1007/s11263-010-0390-2.

Barron, Andrew R., Brandy, Barron, & Yale, Stat (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning*, *14*, 115–133.

Bartunov, Sergey, Santoro, Adam, Richards, Blake A, Hinton, Geoffrey E, & Lillicrap, Timothy P (2018). Assessing the scalability of biologically-motivated deep learning algorithms and architectures. arXiv Prepr. URL https://arxiv.org/abs/1807.04587.

Bellec, Guillaume, Salaj, Darjan, Subramoney, Anand, Legenstein, Robert, & Maass, Wolfgang (2018). Long short-term memory and learning-to-learn in networks of spiking neurons. (pp. 1–17). arXiv Prepr., URL http://arxiv.org/abs/1803.09574.

Brito, Carlos S. N., & Gerstner, Wulfram (2016). Nonlinear hebbian learning as a unifying principle in receptive field formation. *PLoS Computational Biology*, [ISSN: 15537358] *12*(9), 1–24. http://dx.doi.org/10.1371/journal.pcbi.1005070.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, [ISSN: 0028-0836] *337*(6203), 129–132. http://dx.doi.org/10.1038/337129a0, URL https://www.nature.com/articles/337129a0.pdf.

Dale, H. (1935). Pharmacology and nerve-endings (walter ernest dixon memorial lecture): (section of therapeutics and pharmacology). *Proceedings of the Royal Society of Medicine*, [ISSN: 0035-9157] *28*(3), 319–332.

Dasgupta, Sanjoy, Sheehan, Timothy C., Stevens, Charles F., & Navlakha, Saket (2018). A neural data structure for novelty detection. *Proceedings of the National Academy of Sciences*, *115*(51), 13093–13098. http://dx.doi.org/10.1073/pnas.1814448115.

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

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, & Li, Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. *IEEE Conf. Comput. Vis. pattern Recognit.*, [ISSN: 1063-6919] 248–255. http://dx.doi.org/10.1109/CVPRW.2009.5206848, URL http://www.image-net.org.

Diehl, Peter U., & Cook, Matthew (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, [ISSN: 1662-5188] *9*(August), 1–9. http://dx.doi.org/10.3389/fncom.2015.00099, http://journal.frontiersin.org/Article/10.3389/fncom.2015.00099/abstract.

Diehl, Peter U., Neil, Daniel, Binas, Jonathan, Cook, Matthew, Liu, Shih-Chii, & Pfeiffer, Michael (2015). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. *International Journal of Conference in Neural Networks*.

Guergiuev, Jordan, Lillicrap, Timothy P., & Richards, Blake A. (2016). Deep learning with segregated dendrites 1610 (00161), 1–29. arXiv Prepr., URL https://arxiv.org/abs/1610.00161.

Hayashi-Takagi, Akiko, Yagishita, Sho, Nakamura, Mayumi, Shirai, Fukutoshi, Wu, Yi I., Loshbaugh, Amanda L., et al. (2015). Labelling and optical erasure of synaptic memory traces in the motor cortex. *Nature*, [ISSN: 14764687] *525*(7569), 333–338. http://dx.doi.org/10.1038/nature15257.

Hebb, D. O. (1949). *The organization of behavior (Vol. 911)* (p. 335). New York Wiley, [ISSN: 03619230] ISBN: 0805843000.

Huang, Guang Bin, Zhu, Qin Yu, & Siew, Chee Kheong (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, [ISSN: 09252312] *70*(1–3), 489–501. http://dx.doi.org/10.1016/j.neucom.2005.12.126.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal Physiology*, [ISSN: 00223751] *160*(1), 106–154. http://dx.doi.org/10.1113/jphysiol.1962.sp006837.

Hussain, Shaista, Liu, Shih Chii, & Basu, Arindam (2014). Improved margin multi-class classification using dendritic neurons with morphological learning. In *Proc. - IEEE Int. Symp. Circuits Syst.* (pp. 2640–2643). ISSN: 02714310. http://dx.doi.org/10.1109/ISCAS.2014.6865715.

Hyvärinen, Aapo, & Oja, Erkki (1998). Independent component analysis by general nonlinear Hebbian-like learning rules. *Signal Processing*, *64*, 301–313.

Kheradpisheh, Saeed Reza, Ganjtabesh, Mohammad, Thorpe, Simon J., & Masquelier, Timothée (2018). STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, [ISSN: 08936080] *99*, 56–67. http://dx.doi.org/10.1016/j.neunet.2017.12.005, URL http://linkinghub.elsevier.com/retrieve/pii/S0893608017302903.

Kohan, Adam A., Rietman, Edward A., & Siegelmann, Hava T. (2018). Error forward-propagation: Reusing feedforward connections to propagate errors in deep learning. arXiv Prepr., URL http://arxiv.org/abs/1808.03357.

Kriegeskorte, Nikolaus (2015). Deep neural networks: A new framework for modeling biological vision and brain information processing. *Annual Review of Visualization in Science*, [ISSN: 2374-4642] *1*(1), 417–446. http://dx.doi.org/10.1146/annurev-vision-082114-035447.

Krizhevsky, Alex (2013). URL https://www.cs.toronto.edu/~kriz/cifar.html.

Krotov, Dmitry, Hopfield, John J., & Lee, Daniel D. (2019). Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences*, *116*(16), 7723–7731. http://dx.doi.org/10.1073/pnas.1820458116.

Kulkarni, Shruti R., & Rajendran, Bipin (2018). Spiking neural networks for handwritten digit recognition-supervised learning and network optimization. *Neural Networks*, [ISSN: 18792782] *103*, 118–127, http://dx.doi.org/S0893608018301126.

LeCun, Yann (1998). http://yann.lecun.com/exdb/mnist/.

LeCun, Yann, Bengio, Yoshua, & Hinton, Geoffrey (2015). Deep learning. *Nature Review*, [ISSN: 0028-0836] *521*, 436–444. http://dx.doi.org/10.1038/nature14539.

Lee, Jun Haeng, Delbruck, Tobi, & Pfeiffer, Michael (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, [ISSN: 1662453X] *10*(NOV), http://dx.doi.org/10.3389/fnins.2016.00508.

Lee, Chankyu, Srinivasan, Gopalakrishnan, Panda, Priyadarshini, & Roy, Kaushik (2018). Deep spiking convolutional neural network trained with unsupervised spike timing dependent plasticity. *IEEE Transactions in Cognnitive Development System*, [ISSN: 2379-8920] *8920*(c), 1. http://dx.doi.org/10.1109/TCDS.2018.2833071, URL https://ieeexplore.ieee.org/document/8354825/.

Lee, Dong Hyun, Zhang, Saizheng, Fischer, Asja, & Bengio, Yoshua (2015). Difference target propagation. *Lecture Notes in Computer Science*, [ISSN: 16113349] *9284*(3), 498–515, (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). http://dx.doi.org/10.1007/978-3-319-23528-8_31.

Lillicrap, Timothy P., Cownden, Daniel, Tweed, Douglas B., & Akerman, Colin J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communication*, [ISSN: 2041-1723] *7*, 13276. http://dx.doi.org/10.1038/ncomms13276.

Lin, Zhouhan, & Memisevic, Roland (2016). How far Can we go without convolution: Improving fully - connected networks. In *Work. Track - ICLR 2016* (pp. 1–10).

Litwin-Kumar, Ashok, Harris, Kameron Decker, Axel, Richard, Sompolinsky, Haim, & Abbott, L. F. (2017). Optimal degrees of synaptic connectivity. *Neuron*, [ISSN: 10974199] *93*(5), 1153–1164.e7. http://dx.doi.org/10.1016/j.neuron.2017.01.030.

Liu, Jiqian, & Jia, Yunde (2012). A lateral inhibitory spiking neural network for sparse representation in visual cortex. *Advances in Brain Inspired Cognitive Systems*, *7366*, 259–267. http://dx.doi.org/10.1007/978-3-642-31561-9.

Liu, Qian, Pineda-Garcia, Garibaldi, Stromatias, Evangelos, Serrano-Gotarredona, Teresa, & Furber, Steve B. (2016). Benchmarking spike-based visual recognition: A dataset and evaluation. *Frontiers in Neuroscience*, [ISSN: 1662453X] *10*(NOV), http://dx.doi.org/10.3389/fnins.2016.00496.

Liu, D., & Yue, S. (2018). Event-driven continuous STDP learning with deep structure for visual pattern recognition. *IEEE Transactions in Cybernetics*, [ISSN: 21682267] 1–14. http://dx.doi.org/10.1109/TCYB.2018.2801476.

Marblestone, Adam Henry, Wayne, Greg, & Kording, Konrad P. (2016). Towards an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, [ISSN: 1662-5188] *10*(September), 1–61. http://dx.doi.org/10.1101/058545.

Moskovitz, Theodore H., Litwin-kumar, Ashok, & Abbott, L. f. (2018). Feedback alignment in deep convolutional networks. *Neural Evolution Computation*, 1–10, arXiv:1812.06488.

Naud, Richard, Marcille, Nicolas, & Clopath, Claudia (2008). Firing patterns in the adaptive exponential integrate-and-fire model. *Biological Cybernetics*, 335–347. http://dx.doi.org/10.1007/s00422-008-0264-7.

Nawrocki, Robert A., Voyles, Richard M., & Shaheen, Sean E. (2016). A mini review of neuromorphic architectures and implementations. *IEEE Transactions in Electron Devices*, [ISSN: 00189383] *63*(10), 3819–3829. http://dx.doi.org/10.1109/TED.2016.2598413.

Neftci, Emre O., Augustine, Charles, Paul, Somnath, & Detorakis, Georgios (2017). Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in Neuroscience*, [ISSN: 1662453X] *11*(JUN), 1–18. http://dx.doi.org/10.3389/fnins.2017.00324.

Neftci, Emre, Das, Srinjoy, Pedroni, Bruno, Kreutz-Delgado, Kenneth, & Cauwenberghs, Gert (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience*, [ISSN: 1662453X] *7*(8 JAN), 1–14. http://dx.doi.org/10.3389/fnins.2014.00272.

Nøkland, Arild (2016). Direct feedback alignment provides learning in deep neural networks. *Neural Information Processing Series*.

O'Connor, Peter, Gavves, Efstratios, & Welling, Max (2017). Temporally efficient deep learning with spikes. arXiv Prepr., URL http://arxiv.org/abs/1706.04159.

O'Connor, Peter, Neil, Daniel, Liu, Shih Chii, Delbruck, Tobi, & Pfeiffer, Michael (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, [ISSN: 16624548] *7*(7 OCT), 1–13. http://dx.doi.org/10.3389/fnins.2013.00178.

Oja, Erkki (1982). A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, *1*, 267–273.

Olshausen, Bruno A., & Field, David J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, [ISSN: 00426989] *37*(23), 3311–3325. http://dx.doi.org/10.1016/S0042-6989(97)00169-7.

Panda, Priyadarshini, & Roy, Kaushik (2016). Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition. arXiv Prepr., URL https://arxiv.org/abs/1602.01510.

Pehlevan, Cengiz, & Chklovskii, Dmitri B. (2015). A Hebbian/Anti-Hebbian network derived from online non-negative matrix factorization can cluster and discover sparse features. In *Conf. Rec. - Asilomar Conf. Signals, Syst. Comput. (Vol. 2015-April)* (pp. 769–775). ISSN: 10586393. http://dx.doi.org/10.1109/ACSSC.2014.7094553.

Pozzi, Isabella, Bohté, Sander M., & Roelfsema, Pieter R. (2018). A biologically plausible learning rule for deep learning in the brain. (pp. 1–14). arXiv Prepr.

Querlioz, Damien, Bichler, Olivier, Dollfus, Philippe, & Gamrat, Christian (2013). Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Transactions in Nanotechnology*, [ISSN: 1536125X] *12*(3), 288–295. http://dx.doi.org/10.1109/TNANO.2013.2250995.

Real, Esteban, Aggarwal, Alok, Huang, Yanping, & Le, Quoc V. (2018). Regularized evolution for image classifier architecture search. arXiv Prepr., URL http://arxiv.org/abs/1802.01548.

Rombouts, Jaldert O., Bohte, Sander M., & Roelfsema, Pieter R. (2015). How attention can create synaptic tags for the learning of working memories in sequential tasks. *PLoS Computational Biology*, [ISSN: 15537358] *11*(3), 1–34. http://dx.doi.org/10.1371/journal.pcbi.1004060.

Rozell, Christopher J, Johnson, Don H, Baraniuk, Richard G, & Olshausen, Bruno A (2008). Sparse coding via thresholding and local competition in neural circuits. *Neural Computation*, [ISSN: 08997667] *20*(10), 2526–2563. http://dx.doi.org/10.1162/neco.2008.03-07-486.

Rumelhart, David E., Hinton, Geoffrey E., & Williams, Ronald J. (1986). Learning representations by back-propagating errors. *Nature*, [ISSN: 0028-0836] *323*(6088), 533–536. http://dx.doi.org/10.1038/323533a0.

Sacramento, João, Costa, Rui Ponte, Bengio, Yoshua, & Senn, Walter (2017). Dendritic error backpropagation in deep cortical microcircuits. (pp. 1–37). arXiv Prepr. http://arxiv.org/abs/1801.00062.

Samadi, Arash, Lillicrap, Timothy P., & Tweed, Douglas B. (2017). Deep learning with dynamic spiking neurons and fixed feedback weights. *Neural Computation*, [ISSN: 1530888X] *29*, 578–602. http://dx.doi.org/10.1162/NECO.

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

Scellier, Benjamin, & Bengio, Yoshua (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, [ISSN: 1662-5188] *11*(May), 1–13. http://dx.doi.org/10.3389/fncom.2017.00024.

Spoerer, Courtney J., McClure, Patrick, & Kriegeskorte, Nikolaus (2017). Recurrent convolutional neural networks: A better model of biological object recognition. *Frontiers in Psychology*, [ISSN: 16641078] *8*(SEP), 1–14. http://dx.doi.org/10.3389/fpsyg.2017.01551.

Tavanaei, Amirhossein, Ghodrati, Masoud, Kheradpisheh, Saeed Reza, Masquelier, Timothee, & Maida, Anthony S. (2018). Deep learning in spiking neural networks. *Neural Networks*, [ISSN: 18792782] *111*, 47–63. http://dx.doi.org/10.1016/j.neunet.2014.09.003, http://arxiv.org/abs/1804.08150.

Tavanaei, Amirhossein, & Maida, Anthony S. (2016). Bio-inspired spiking convolutional neural network using layer-wise sparse coding and STDP learning, 1611.03000v2. (pp. 1–20). arXiv Prepr. URL http://arxiv.org/abs/1611.03000.

Thiele, Johannes Christian, Bichler, Olivier, & Dupret, Antoine (2018). Event-based, timescale invariant unsupervised online deep learning with STDP. *Frontiers in Computational Neuroscience*, [ISSN: 1662-5188] *12*(June), 46. http://dx.doi.org/10.3389/FNCOM.2018.00046, URL https://www.frontiersin.org/articles/10.3389/fncom.2018.00046/abstract.

Urbanczik, Robert, & Senn, Walter (2014). Learning by the dendritic prediction of somatic spiking. *Neuron*, *81*(3), 521–528. http://dx.doi.org/10.1016/j.neuron.2013.11.030.

Whittington, James C. R., & Bogacz, Rafal (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural Computation*, [ISSN: 1530888X] *29*, 1229–1262. http://dx.doi.org/10.1162/NECO.

Whittington, James C. R., & Bogacz, Rafal (2019). Theories of error backpropagation in the brain. *Trends in Cognitive Science*, [ISSN: 1364-6613] *xx*, 1–16. http://dx.doi.org/10.1016/j.tics.2018.12.005.

Wu, Yujie, Deng, Lei, Li, Guoqi, Zhu, Jun, & Shi, Luping (2018). Direct training for spiking neural networks: Faster, larger, better. arXiv Prepr., URL http://arxiv.org/abs/1809.05793.

Yamins, Daniel L. K., & DiCarlo, James J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, [ISSN: 15461726] *19*(3), 356–365. http://dx.doi.org/10.1038/nn.4244.

Zhao, Bo, Ding, Ruoxi, Chen, Shoushun, Linares-Barranco, Bernabe, & Tang, Huajin (2015). Feedforward Categorization on AER motion events using cortex-like features in a spiking neural network. *IEEE Transactions in Neural Networks Learning System*, [ISSN: 21622388] *26*(9), 1963–1978. http://dx.doi.org/10.1109/TNNLS.2014.2362542.

Zylberberg, Joel, Murphy, Jason Timothy, & DeWeese, Michael Robert (2011). A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields. *PLoS Computational Biology*, [ISSN: 1553734X] *7*(10), http://dx.doi.org/10.1371/journal.pcbi.1002250.