Neural Networks 143 (2021) 719–731

Contents lists available at ScienceDirect

Neural Networks

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

Hebbian semi-supervised learning in a sample efficiency setting

Gabriele Lagani [a,*], Fabrizio Falchi [b], Claudio Gennaro [b], Giuseppe Amato [b]

[a] Computer Science Department, University of Pisa, Pisa, Italy
[b] ISTI - CNR, Pisa, Italy

ARTICLE INFO

Article history:
Received 26 February 2021
Received in revised form 27 July 2021
Accepted 2 August 2021
Available online 13 August 2021

Keywords:
Convolutional Neural Networks
Computer vision
Semi-supervised learning
Hebbian learning
Sample efficiency

ABSTRACT

We propose to address the issue of sample efficiency, in Deep Convolutional Neural Networks (DCNN), with a semi-supervised training strategy that combines Hebbian learning with gradient descent: all internal layers (both convolutional and fully connected) are pre-trained using an unsupervised approach based on Hebbian learning, and the last fully connected layer (the classification layer) is trained using Stochastic Gradient Descent (SGD). In fact, as Hebbian learning is an unsupervised learning method, its potential lies in the possibility of training the internal layers of a DCNN without labels. Only the final fully connected layer has to be trained with labeled examples.

We performed experiments on various object recognition datasets, in different regimes of sample efficiency, comparing our semi-supervised (Hebbian for internal layers + SGD for the final fully connected layer) approach with end-to-end supervised backprop training, and with semi-supervised learning based on Variational Auto-Encoder (VAE). The results show that, in regimes where the number of available labeled samples is low, our semi-supervised approach outperforms the other approaches in almost all the cases.

1. Introduction

In recent years, Deep Neural Networks (DNNs) have achieved impressive results in learning tasks such as computer vision (He, Zhang, Ren, & Sun, 2016), reinforcement learning (Silver et al., 2016), and natural language processing (Devlin, Chang, Lee, & Toutanova, 2018).

Today's neural networks are generally trained using Stochastic Gradient Descent (SGD) with the error backpropagation algorithm (backprop), which reaches high accuracy when a large number of labeled samples are available for training. However, gathering labeled samples is expensive, requires a significant amount of human work, and, in many applications, a large amount of training data is simply not available.

Therefore, researchers started to investigate strategies for sample efficient learning (Bengio, Lamblin, Popovici, & Larochelle, 2007; Chen, Kornblith, Norouzi, & Hinton, 2020; Kingma, Mohamed, Jimenez Rezende, & Welling, 2014; Larochelle, Bengio, Louradour, & Lamblin, 2009; Rasmus, Berglund, Honkala, Valpola, & Raiko, 2015; Weston, Ratle, Mobahi, & Collobert, 2012; Zhang, Lee, & Lee, 2016). In this setting, only a small number of labeled samples is assumed to be available. On the other hand, gathering unlabeled samples is relatively simple; therefore, these approaches exploit unlabeled samples to perform unsupervised training in addition to the supervised training process, leading to the so called *semi-supervised* learning technique. It is well known that unsupervised pre-training helps initializing the network weights in the neighborhood of a good local optimum (Bengio et al., 2007; Larochelle et al., 2009), thus easing convergence in a successive supervised fine-tuning phase. Current semi-supervised approaches leverage autoencoder architectures for the unsupervised part of the task (Kingma et al., 2014; Rasmus et al., 2015; Zhang et al., 2016), although they are still based on backprop. Another approach, SimCLR (Chen et al., 2020), exploits data augmentation and an unsupervised contrastive criterion.

In this work, we address the sample efficiency issue by proposing a semi-supervised learning approach, where an initial unsupervised learning step, using all available data – unlabeled and labeled (but without using label information) – is followed by a supervised learning step using a small amount of labeled data. To perform the unsupervised learning step we explore the use of the Hebbian learning paradigm, which mimics more closely the synaptic adaptation mechanisms found in biological brains, according to neuroscientists. Hebbian learning is a local learning rule (Gerstner & Kistler, 2002; Haykin, 2009), i.e. it does not require error backpropagation, and it does not require supervision. Moreover, the capabilities of biological brains to learn and generalize only from a limited number of labeled samples make this approach appealing for the sample efficient learning setting. Note also that backprop-based approaches are considered to be biologically implausible (O'Reilly & Munakata, 2000).

* Corresponding author.
E-mail address: gabriele.lagani@phd.unipi.it (G. Lagani).

https://doi.org/10.1016/j.neunet.2021.08.003
0893-6080/

The main contributions of this paper are the following:

- We propose a semi-supervised learning approach that combines Hebbian learning with SGD on object recognition tasks with Deep Convolutional Neural Networks (DCNNs).
  - All available training samples, unlabeled and labeled, are used for an unsupervised Hebbian pre-training phase (without using label information), where a nonlinear Hebbian Principal Component Analysis (HPCA) learning rule is used to train internal layers (both convolutional and fully connected);
  - Then, labeled training samples and SGD are used to train a classifier, obtained as a final fully connected layer, on the features extracted from previous layers;
- The results are compared from a sample efficiency perspective with those obtained by a baseline network trained end-to-end with backprop, on the labeled samples, and with semi-supervised learning based on Variational Auto-Encoder (VAE) (Kingma & Welling, 2013) unsupervised pre-training, the latter using all the available samples (VAE-based semi-supervised learning was also the approach considered in Kingma et al. (2014));
- Different datasets and different regimes of sample efficiency are explored, and it is shown that the proposed semi-supervised approach (Hebbian + SGD) outperforms the other approaches in almost all the cases where a limited number of labeled samples is available.

The remainder of this paper is structured as follows: Section 2 gives an overview on related work concerning semi-supervised training and Hebbian learning; Section 3 illustrates the sample efficiency problem. Section 4 defines our approach to sample efficiency based on semi-supervised Hebbian + SGD learning; Section 5 provides a background on the Hebbian learning rule that we used in this work; Section 6 delves into the details of our experimental setup; In Section 7, the results of our simulations are illustrated; Finally, Section 8 presents our conclusions and outlines possible future developments.

2. Related work

In this section, we present an overview of related work concerning both semi-supervised training and Hebbian learning.

2.1. Semi-supervised training and sample efficiency

Early work on deep learning had to face problems related to convergence to poor local minima during the training process, which led researchers to exploit a pre-training phase that allowed them to initialize network weights in a region near a good local optimum (Bengio et al., 2007; Larochelle et al., 2009). In these studies, greedy layerwise pre-training was performed by applying unsupervised autoencoder models layer by layer, thus training each layer to provide a compressed representation of the input for a successive decoding stage. It was shown that such pre-training was indeed helpful to obtain a good initialization for a successive supervised training stage. In successive works, the idea of enhancing neural network training with an unsupervised learning objective was considered (Kingma et al., 2014; Rasmus et al., 2015; Weston et al., 2012; Zhang et al., 2016). In Kingma et al. (2014), Variational Auto-Encoders (VAE) are considered to perform an unsupervised pre-training phase using a limited amount of labeled samples. Also Rasmus et al. (2015) and Zhang et al. (2016) relied on autoencoding architectures to augment supervised training with unsupervised reconstruction objectives, showing that joint optimization of supervised and unsupervised losses helped to regularize the learning process. In Weston, Chopra, and Bordes (2014), joint supervised and unsupervised training was again considered, but the unsupervised learning part was based on manifold learning techniques. Another approach, SimCLR (Chen et al., 2020), used a Contrastive Loss to perform the unsupervised learning part. The approach relies on data augmentation, in order to produce transformed variants of a given input, and the unsupervised loss basically imposes hidden representations to match for transformed variants generated from the same input.

2.2. Hebbian learning

Several variants of Hebbian learning rules were developed over the years, such as Hebbian learning with Winner-Takes-All (WTA) competition (Grossberg, 1976), Self-Organizing Maps (Kohonen, 1982), Hebbian learning for Principal Component Analysis (PCA) (Becker & Plumbley, 1996; Haykin, 2009; Karhunen & Joutsensalo, 1995; Sanger, 1989), Hebbian/anti-Hebbian learning (Pehlevan & Chklovskii, 2015; Pehlevan, Hu, & Chklovskii, 2015). A brief overview is given in Section 5. However, it was only recently that Hebbian learning started gaining attention in the context of DNN training (Amato, Carrara, Falchi, Gennaro, & Lagani, 2019; Bahroun & Soltoggio, 2017; Krotov & Hopfield, 2019; Lagani, 2019; Wadhwa & Madhow, 2016a, 2016b). In Krotov and Hopfield (2019), a Hebbian learning rule based on inhibitory competition was used to train a neural network composed of fully connected layers on object recognition tasks. Instead, the Hebbian/anti-Hebbian learning rule developed in Pehlevan et al. (2015) was applied in Bahroun and Soltoggio (2017) to extract convolutional features that were shown to be effective for classification. Convolutional layers were also considered in Wadhwa and Madhow (2016a, 2016b), where a Hebbian approach based on WTA competition was used to train the feature extractors. However, the previous approaches were based on relatively shallow network architectures (2–3 Layers). A further step was taken in Amato et al. (2019) and Lagani (2019), where a Hebbian WTA learning rule was investigated for training a 6-layer Convolutional Neural Network (CNN). Also, a supervised variant of Hebbian learning was proposed to train the final classification layer. Hybrid network models were also considered, in which some layers where trained using backprop and others using Hebbian learning. The results suggested that Hebbian learning is suitable for training early feature detectors, as well as higher network layers, but not very effective for training intermediate network layers. Furthermore, Hebbian learning was successfully used to retrain the higher layers of a pre-trained network, achieving results comparable to backprop, but requiring fewer training epochs, thus suggesting potential applications in the context of transfer learning (see also Canto, 2020; Magotra & kim, 2019; Magotra & Kim, 2020).

3. Sample efficiency scenario

Training neural networks with supervision requires gathering a large amount of labeled training samples. This can be quite expensive, since it requires a considerable amount of human intervention for manually labeling collected samples. On the other hand, gathering unlabeled samples is generally considerably cheaper. In real world scenarios, one might need to solve AI problems for which only a small amount of labeled samples is available. In some cases, it is possible to start from a neural network pre-trained on a similar task, for which a good number of labeled samples was available, and just fine-tune it on the target task using the available labeled samples (thus performing transfer learning (Yosinski, Clune, Bengio, & Lipson, 2014). However, in

many cases, the target task might be considerably different from the original task.

In such cases, it would be desirable to have a training algorithm that is capable of exploiting the latent information contained in large collections of unlabeled samples, while using only a small set of labeled samples for a successive stage of supervised training.

In fact, in applications where gathering large amounts of manually labeled data would be too expensive, it is often possible to collect a considerable amount of unlabeled samples at a relatively cheap cost. Therefore, it is desirable for an algorithm to be able to acquire knowledge about the data without using labels and learning to extract possibly useful features by, for example, learning to distinguish frequent shapes and patterns, or detecting rarely occurring anomalies. However, a fully supervised, backprop-based, approach, typically requires many labeled samples to achieve a good performance.

Formally, the sample efficiency learning scenario can be stated as follows: let $\mathcal{T}$ be our *training set*, let $s \in \mathcal{T}$ be an element (called *training sample*) of the training set. Elements of $\mathcal{T}$ are drawn from a statistical distribution with *probability density function (pdf)* $p(s)$:

$$s \sim p(s) \tag{1}$$

Let $N = |\mathcal{T}|$ be the number of training samples in $\mathcal{T}$ (where $|\cdot|$ denotes the cardinality of the set). Let $\mathcal{L}$ be another set, whose elements $l \in \mathcal{L}$ are called *labels*. Let

$$\phi : \mathcal{T} \rightarrow \mathcal{L} \tag{2}$$

be a function that maps training samples to a corresponding label. This function is unknown, except for a subset of $\mathcal{T}$ for which labels are given. Specifically, we define the *labeled set* as a subset $\mathcal{T}_L \subset \mathcal{T}$ of elements for which the image of the function ϕ is known:

$$\mathcal{T}_L = \{s \in \mathcal{T} \mid \phi(s) \text{ is known}\} \tag{3}$$

Let us define the *unlabeled set* $\mathcal{T}_U$ as the complement of $\mathcal{T}_L$ w.r.t. $\mathcal{T}$. Therefore, for the unlabeled set, label information is not available; nonetheless, samples from $\mathcal{T}_U$ are drawn from the same statistical distribution as the samples from $\mathcal{T}_L$ and $\mathcal{T}$, i.e. from $p(s)$. In a *sample efficiency* scenario, the number of samples in $\mathcal{T}_L$ is typically much smaller than the total number of samples N in $\mathcal{T}$, i.e.

$$|\mathcal{T}_L| \ll |\mathcal{T}| \tag{4}$$

In particular, an *r %-sample efficiency* scenario is characterized by

$$|\mathcal{T}_L| = \frac{r}{100}|\mathcal{T}| \tag{5}$$

i.e. the size of the labeled set is r % that of the whole training set (labeled plus unlabeled). A neural network is required to approximate, by a training process, the function ϕ. For a given $\mathcal{T}$ and a given r %-sample efficiency regime, we define a neural network to *perform better* than another if it reaches higher accuracy in mapping samples to correct labels (according to function ϕ), given that both networks are trained using $\mathcal{T}_L$ and $\mathcal{T}_U$ splits that are compliant with the considered r %-sample efficiency scenario, i.e. both networks have been trained with a number of labeled samples equal to r % of the total number of samples in $\mathcal{T}$. The training process can be *supervised*, in which case only samples from $\mathcal{T}_L$, and the associated labels, are used, or it can be *unsupervised*, in which case all samples from both $\mathcal{T}_U$ and $\mathcal{T}_L$ are used, but without using the labels for the latter.

4. Hebbian-SGD semi-supervised learning approach

Traditional supervised approaches based on backpropagation work well provided that the size of the labeled set is sufficiently large, but they do not exploit the unlabeled set. In the semi-supervised approach that we propose, we aim to learn an approximation of function ϕ by means of a two stage process: first, we run an unsupervised training stage in which we use samples from both $\mathcal{T}_U$ and $\mathcal{T}_L$ (but without using the labels for the latter), in order to learn latent representations for samples drawn from $p(s)$; subsequently, we use the small number of available labeled samples from $\mathcal{T}_L$ (with the corresponding labels) in a supervised training phase. During the first phase, latent representations are obtained from hidden layer of a DCNN, which are trained using an unsupervised, biologically plausible, Hebbian learning algorithm. During the second phase, supervised training is applied on a final linear classifier, by running a SGD optimization procedure using only the few labeled samples at our disposal (with the corresponding labels).

Specifically, we propose the following semi-supervised learning approach, which combines Hebbian learning with SGD, applied to a deep convolutional neural network architecture as the one in Fig. 1:

1. the internal layers of the neural network are trained without supervision using the Hebbian learning rule, given in Eq. (12) discussed in Section 5, and all the available samples from both $\mathcal{T}_U$ and $\mathcal{T}_L$ (but without using the labels for the latter), independently from the final fully connected layer, used as classifier;
2. the final fully connected layer, used as classifier, is trained with SGD, freezing the weights of previous layers, and using only the few labeled examples in $\mathcal{T}_L$ (with the corresponding labels);
3. We also considered the case in which, during the supervised phase, the weights of the previous layers are not frozen, but they are adapted together with the final classifier, thus performing end-to-end fine tuning on the Hebbian pre-trained network; we compare the results of the Hebbian approach with and without fine tuning, in order to be able to tell apart the contributions of Hebbian learning and end-to-end fine tuning on the final result.

As we will show in the following, we found that our semi-supervised approach achieves interesting results in low sample efficiency regimes. In fact, when only a small number of labeled samples is used (roughly from 1% to 5% of the total number of training samples contained in the considered datasets), our approach generally offers better results than full backprop training, and VAE-based semi-supervised learning. Thus it represents a promising solution to real life applications, where the number of available labeled samples is small.

In the experiments discussed below, we compared the performance of our approach with full backprop, and with VAE-based semi-supervised learning, considering various levels of sample efficiency. Moreover, we run different experiments in which the final classifier was placed on top of a different inner layer of the network, in order to evaluate the quality of the feature representations extracted at different network depths in the classification task. As we will show later, our approach performs better than the other methods in almost all the cases when we consider low sample efficiency scenarios, where the percentage of labeled samples is between 1% and 5% (i.e. r %-sample efficiency scenarios with $1 \leq r \leq 5$). Further improvements come when end-to-end fine tuning is used, in addition to Hebbian pre-training.

5. Background on Hebbian learning

For a given neuron, the weight updates according to the Hebbian learning rule, in its most basic form, can be expressed as (Haykin, 2009):

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \Delta\mathbf{w} \tag{6}$$

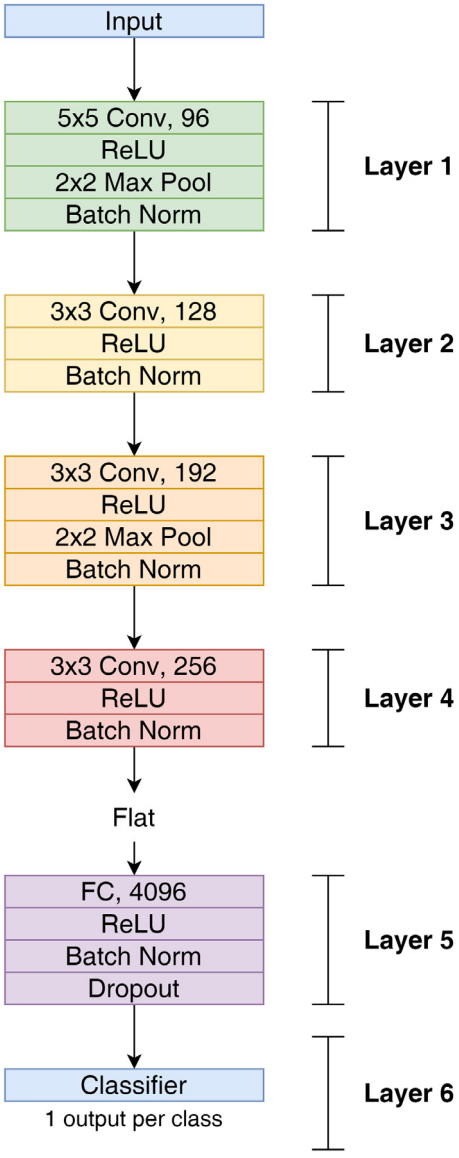


Fig. 1. The neural network used for the experiments.

where $\mathbf{w}_{new}$ is the updated weight vector, $\mathbf{w}_{old}$ is the old weight vector, and $\Delta\mathbf{w}$ is the weight update. The latter term is computed as follows:

$$\Delta\mathbf{w} = \eta\, y\, \mathbf{x} \tag{7}$$

Here, $\mathbf{x}$ is the input vector of the neuron, $\mathbf{w}$ is its weight vector, $y = \mathbf{w}^{\mathbf{T}}\mathbf{x}$ is its output, and η is the learning rate. According to this rule, the weight on a given synapse is increased when the input on that synapse and the output of the neuron are simultaneously high, so that correlation between simultaneously active neurons is reinforced.

The learning rule above is unstable, in that neuron weights are allowed to grow unbounded. To prevent this circumstance, a weight decay term is typically added: $\Delta\mathbf{w} = \eta\, y\, \mathbf{x} - \lambda\, \mathbf{w}$. In particular, the term λ can be chosen in the form $\lambda = \eta\, y$, as suggested, for example, in works on competitive learning (Grossberg, 1976; Kohonen, 1982), leading to the following expression for $\Delta\mathbf{w}$:

$$\Delta\mathbf{w} = \eta\, y\, \mathbf{x} - \eta\, y\, \mathbf{w} = \eta\, y\, (\mathbf{x} - \mathbf{w}) \tag{8}$$

The rule above can be easily interpreted intuitively: when an input vector is presented to the neuron, its vector of weights takes a small step toward the input, so that the neuron will exhibit a stronger response if a similar input is presented again in the future. When a series of inputs drawn from a cluster are presented to the neuron, the weight vector converges toward the cluster center (Fig. 2).

In a complex neural network, several neurons are involved in representing the input. Ideally, we would like each neuron to learn to represent a different property of the inputs. Therefore, we need a strategy to prevent neurons from learning redundant information. The Winner-Takes-All (WTA) (Grossberg, 1976) strategy was proposed for this purpose. It was motivated by the observation that biological neural networks exhibit inhibitory competition, i.e. when a neuron fires, it provokes the inhibition of neighboring neurons. In the WTA strategy, when an input is presented to a neural network layer, a *winner* neuron is determined. This is the one whose weight vector is closest to the current input, and it is the only neuron allowed to perform a weight update according to Eq. (8). In this way, if a similar input is presented again in the future, the same neuron will be more likely to win again. At the same time, different neurons are induced to represent different properties of the inputs, namely the centroids of the clusters formed by the input data points (Fig. 3).

The WTA provides a *quantized* encoding scheme, in which one particular neuron activates to encode a particular input. However, it has been argued that a distributed type of encoding would be more powerful (Földiak, 1990; Olshausen & Field, 1996), in which different neurons simultaneously activate to encode different properties of the input.

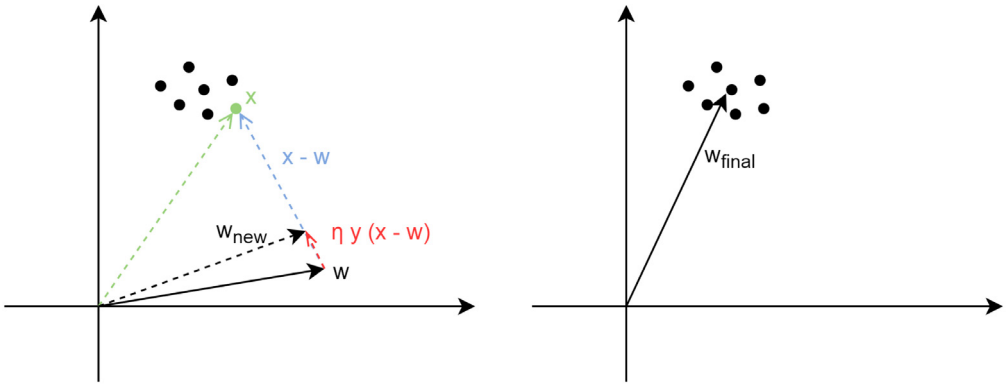
A more distributed coding scheme can be achieved if neuron weight vectors are trained to encode the principal components of the input data. This can be achieved by Hebbian Principal Component Analysis (HPCA) learning rules, in an efficient, online manner (Becker & Plumbley, 1996; Haykin, 2009; Karhunen & Joutsensalo, 1995; Sanger, 1989). The Hebbian PCA learning rule is obtained by minimizing the *representation error* loss function, defined as:

$$L(\mathbf{w_i}) = E[(\mathbf{x} - \sum_{j=1}^{i} y_j\, \mathbf{w_j})^2] \tag{9}$$

where, in this case, the subscript i is used to denote the ith neuron in the layer and $E[\cdot]$ is the mean value operator. Let us give an intuitive explanation to the latter equation. Each weight vector represents a pattern that is stored by a neuron. The neuron's output represents "how much" of that pattern was found in the current input. The summation in the equation represents a *reconstruction* of the current input, obtained as a linear combination of the weights of the neurons, with coefficients given by the corresponding neuron outputs. Finally, we compute the squared error between the actual input $\mathbf{x}$ and the reconstruction, which represents how much the current reconstruction is far from the actual input. Therefore, minimizing this loss corresponds to finding network weights that produce the most accurate reconstruction. The objective in Eq. (9) reduces to classical PCA, in the case of linear neurons and zero centered data, which requires to maximize output variance, while the weight vectors are subject to orthonormality constraints, as shown in Becker and Plumbley (1996), Haykin (2009), Karhunen and Joutsensalo (1995) and Sanger (1989). In the following, we assume that the input data are centered around zero. If this is not true, we just need to subtract the average $E[\mathbf{x}]$ from the inputs beforehand.

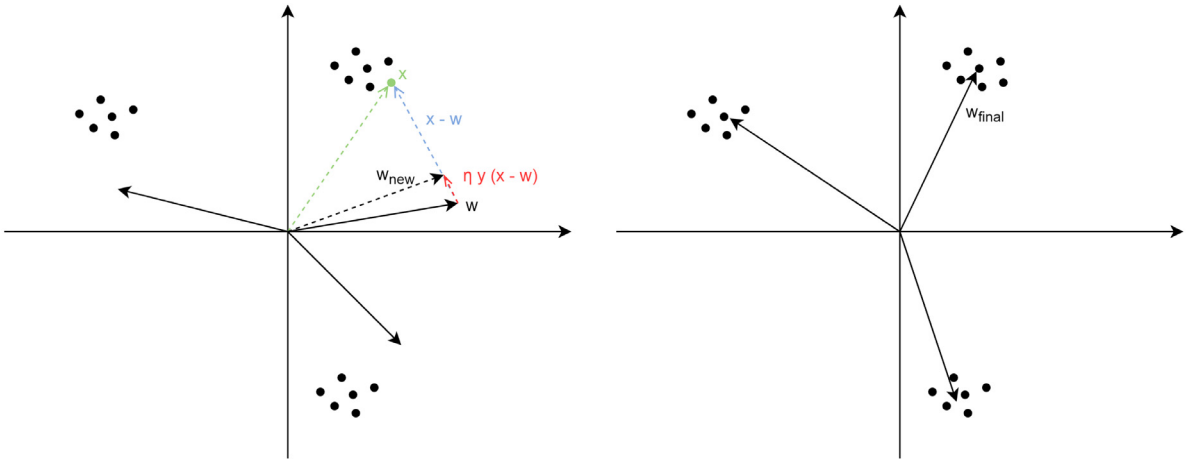
Minimizing the objective in Eq. (9) leads to the following Hebbian PCA learning rule (Becker & Plumbley, 1996; Haykin, 2009; Karhunen & Joutsensalo, 1995; Sanger, 1989):

$$\Delta\mathbf{w_i} = \eta y_i(\mathbf{x} - \sum_{j=1}^{i} y_j \mathbf{w_j}) \tag{10}$$



(a) Update step (b) Final position after convergence

Fig. 2. Hebbian updates with weight decay.



(a) Update step (b) Final position after convergence

Fig. 3. Hebbian updates with Winner-Takes All competition.

In particular, we consider the case in which neurons have non-linear activation functions $f()$, so that the representation error becomes:

$$L(\mathbf{w_i}) = E[(\mathbf{x} - \sum_{j=1}^{i} f(y_j)\,\mathbf{w_j})^2] \tag{11}$$

Minimization of this objective (Karhunen & Joutsensalo, 1995; Becker & Plumbley, 1996) leads to the nonlinear Hebbian PCA learning rule that we used in our experiments:

$$\Delta\mathbf{w_i} = \eta f(y_i)(\mathbf{x} - \sum_{j=1}^{i} f(y_j)\mathbf{w_j}) \tag{12}$$

6. Experimental setup

In the following, we describe the details of our experiments and comparisons, discussing the network architecture and the training procedure.[1]

[1] The code to reproduce the experiments described in this paper is available at: https://github.com/GabrieleLagani/HebbianPCA/tree/hebbpca.

6.1. Datasets used for the experiments

The experiments were performed on the following datasets: CIFAR10, CIFAR100 (Krizhevsky & Hinton, 2009) and Tiny ImageNet (Wu, Zhang, & Xu, 2017).

The CIFAR10 dataset contains 50,000 training images and 10,000 test images, belonging to 10 classes. Moreover, the training images were randomly split into a training set of 40,000 images and a validation set of 10,000 images.

The CIFAR100 dataset also contains 50,000 training images and 10,000 test images, belonging to 100 classes. Also in this case, the training images were randomly split into a training set of 40,000 images and a validation set of 10,000 images.

The Tiny ImageNet dataset contains 100,000 training images and 10,000 test images, belonging to 200 classes. Moreover, the training images were randomly split into a training set of 90,000 images and a validation set of 10,000 images.

We considered a range of sample efficiency regimes, i.e. we assumed that only a limited number of labeled samples was available for training. We considered the following sample-efficiency regimes: the amount of labeled samples were respectively 1%, 2%, 3%, 4%, 5%, 10%, 25% and 100% of the whole training set. This corresponds to 400, 800, 1200, 1600, 2000, 4000, 10,000, 40,000 labeled samples for the CIFAR10 and CIFAR100 datasets, and to 900, 1800, 2700, 3600, 4500, 9000, 22,500, 90,000 labeled samples for Tiny ImageNet.

6.2. Network architecture and training

We considered a six layer neural network as shown in Fig. 1: five deep layers plus a final linear classifier, obtained as a fully connected layer on top of previous layers. The various layers were interleaved with other processing stages (such as ReLU nonlinearities, max-pooling, etc.). The architecture was inspired by AlexNet (Krizhevsky, Sutskever, & Hinton, 2012), but with slight modifications in order to reduce the overall computational cost of training.

For each sample efficiency regime, we trained a deep network with our semi-supervised approach, using the Hebbian PCA (HPCA) rule in Eq. (12) (in which the nonlinearity was set to the ReLU function) in the internal layers, followed by the SGD training step in the classification layer only (plain HPCA), or also fine tuning previous network weights (HPCA plus Fine Tuning — HPCA+FT).

For each sample efficiency configuration we also created a baseline for comparison, training an identical network using SGD with error backpropagation in all layers, in a supervised, end-to-end fashion. We considered both a network trained from scratch, and a network pre-trained with unsupervised VAE method (Kingma & Welling, 2013) and then fine-tuned by supervised backprop. VAE-based semi-supervised learning was also the method considered in Kingma et al. (2014).

As already stated, while supervised training only uses the labeled samples, the unsupervised training step (for the internal layers) uses the entire dataset, consisting of both labeled and unlabeled samples (although the label information is not used).

6.3. Testing sample efficiency at different layer depths

In our experiments, in addition to evaluate the entire network trained as discussed above, we also evaluated the sample efficiency capability on the various internal layers of the trained models. To this end, we cut the networks in correspondence to the output of the various layers and we trained a new linear classifier on top of each already pre-trained layer (for instance, Fig. 4 shows a classifier placed on top of the features extracted from the first layer), and the resulting accuracy was evaluated. These new classifiers were trained with supervision using SGD, hence using only the limited number of labeled training samples given by the sample efficiency regime considered. This process was done for the HPCA trained network with previous layers frozen, for the HPCA trained network with fine tuning of previous layers (HPCA+FT), for the SGD network trained from scratch with supervised backprop, and for the SGD network pre-trained by VAE and then fine tuned with supervised backprop, in order to make comparisons. More details are given below.

6.4. Details of training

We implemented our experiments using PyTorch. Training was performed in 20 epochs using mini-batches of size 64. Networks were fed input images of size 32×32 pixels. Experiments were performed using five different seeds for the Random Number Generator (RNG), averaging the results and computing 95% confidence intervals.

For SGD training of the baseline network, the initial learning rate was set to 10^{-3} and kept constant for the first ten epochs, while it was halved every two epochs for the remaining ten epochs. We also used momentum coefficient 0.9, Nesterov correction, dropout rate 0.5 and L2 weight decay penalty coefficient set to $5 \cdot 10^{-2}$ for CIFAR10, 10^{-2} for CIFAR100 and $5 \cdot 10^{-3}$ for Tiny ImageNet. Cross-entropy loss was used as optimization metric. When VAE pre-training was used, the network in Fig. 1, up to layer 5, acted as encoder, with an extra layer mapping Layer 5 output to 256 Gaussian latent variables, while a specular network branch acted as decoder (Fig. 5). VAE training was performed without supervision, in an end-to-end encoding–decoding task, optimizing the β-VAE Variational Lower Bound (Higgins et al., 2016), with coefficient $\beta = 0.5$.

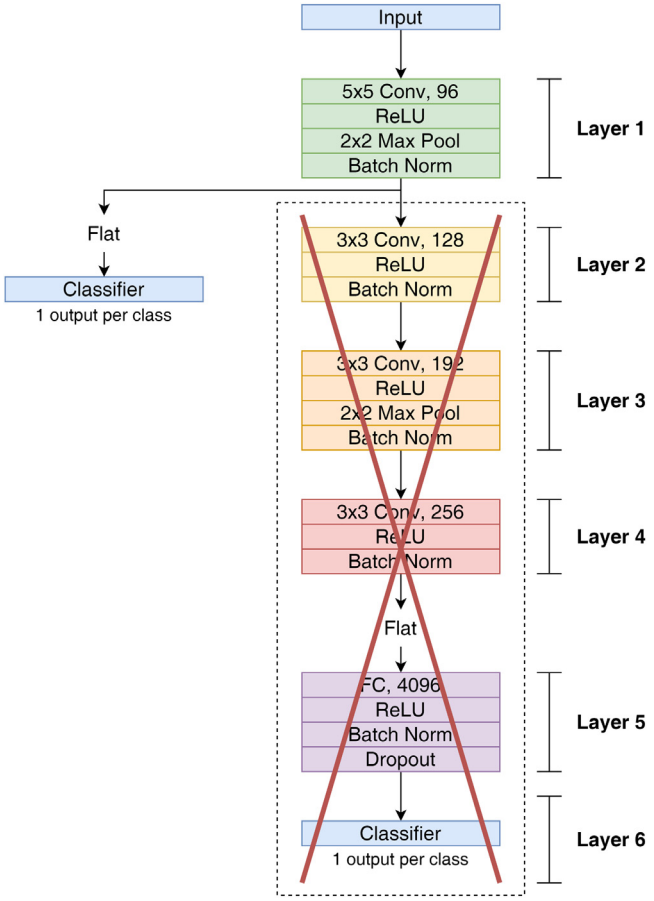


Fig. 4. A neural network is cut in correspondence to Layer 1, and a linear classifier is placed on top of the features extracted from that layer, in order to evaluate their quality in classification tasks.

In the HPCA training, the learning rate was set to 10^{-3}. No L2 regularization or dropout was used in this case, since the learning method did not present overfitting issues.

The linear classifiers placed on top of the various network layers were trained with supervision using SGD in the same way as we described above for training the whole network, but the L2 penalty term was reduced to $5 \cdot 10^{-4}$.

To obtain the best possible generalization, *early stopping* was used in each training session, i.e. we chose as final trained model the state of the network at the epoch when the highest validation accuracy was recorded.

All the above mentioned hyperparameters resulted from a parameter search aimed at maximizing the validation accuracy on the respective datasets.

6.5. Further details on HPCA training

When HPCA training was used, we noticed that BatchNorm (BN) on higher layers was disruptive. The reason is that HPCA uses the variance along input dimensions in order to understand which dimensions in the input space are more relevant. BN, however, normalizes the input distribution, so that each input

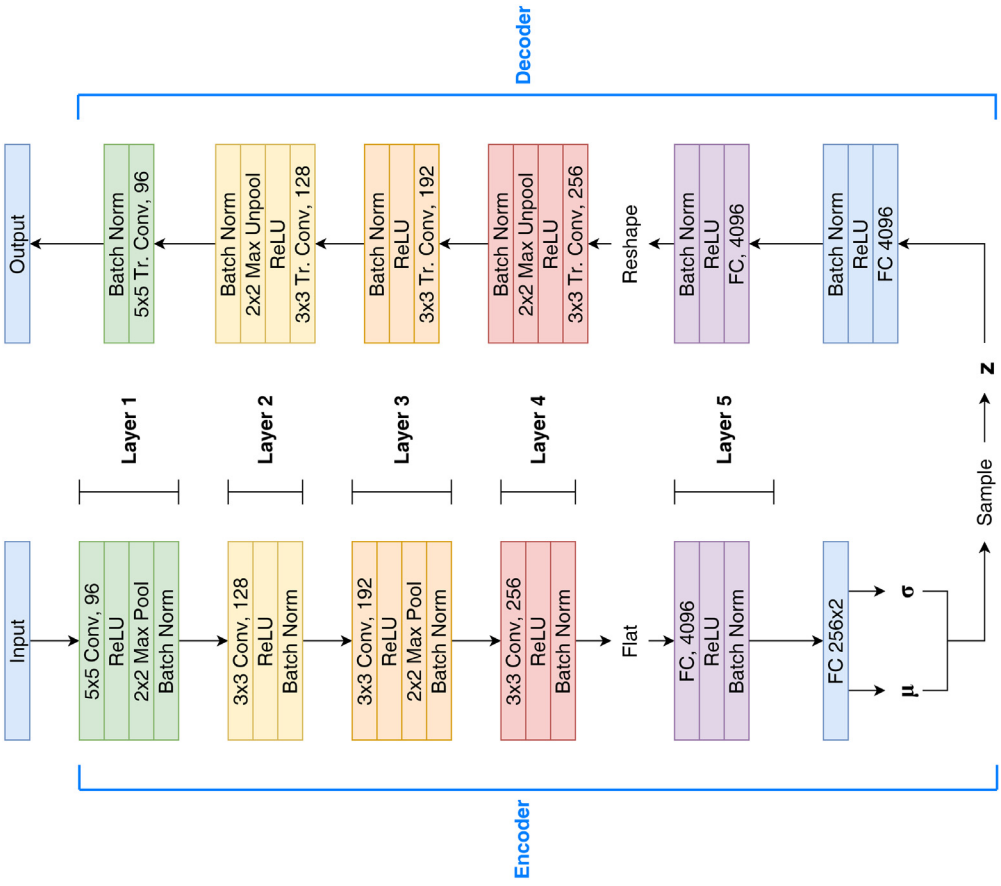


Fig. 5. VAE architecture used in our experiments.

dimension is rescaled to have unit variance, thus causing a loss of information that would have been useful to HPCA. For this reason, we defined a modified BN version as follows: instead of dividing each input dimension by the respective variance estimated from samples, we divided each input dimension by the average of all the variances estimated for all the input dimensions. This allowed us to rescale input dimensions in order to have a fixed variance, on average, while at the same time maintaining the relative order of each of the input dimensions in terms of their variances. The modified BN was applied to Layers 4 and 5 of the network during HPCA training. On the other hand, standard BN was found to be preferable for earlier layers, where feature detectors had not yet developed a task specificity.

In convolutional layers, the HPCA rule was applied to convolutional filters at each offset. In order to preserve weight sharing, the resulting weight updates were averaged along the height, width and batch dimensions (Fig. 6), and the result was the actual weight update that was applied to the convolutional filter.

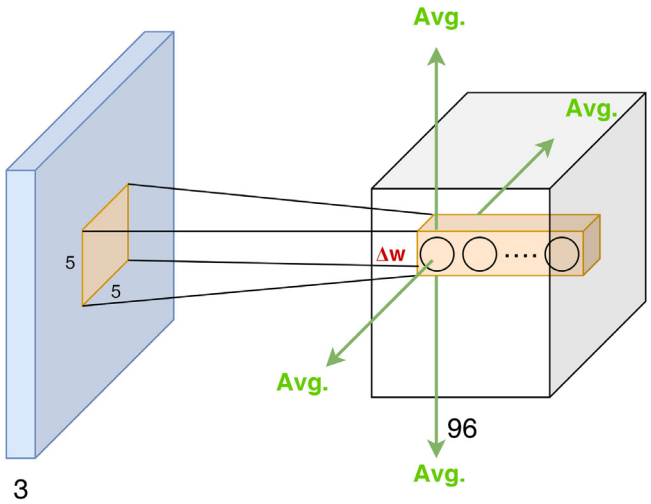


Fig. 6. Update averaging over horizontal and vertical dimensions.

7. Results and discussion

In this section, the experimental results obtained with each dataset are presented and analyzed. Results are reported in Tables 1 2, and 3 for CIFAR10, CIFAR100, and Tiny ImageNet, respectively. After training the entire network, independently, with supervised backprop (BP), VAE-based semi-supervised approach (VAE), Hebbian PCA (HPCA), and HPCA plus Fine Tuning (HPCA+FT), we placed and trained a linear classifier on top of the various layers. The linear classifier was trained, on top of the pre-trained networks, using supervised learning with a number of labeled samples corresponding to the various sample efficiency regimes considered. Tables report the classification accuracy, along with the 95% confidence intervals, for the various sample efficiency regimes, when the classification layer is placed on top of the various internal layers ($L1, \ldots, L5$). We report top-1 accuracy for CIFAR10, given that this dataset contains only 10 classes, and top-5 accuracy for CIFAR100 and Tiny ImageNet,

Table 1
CIFAR10 accuracy (top-1) and 95% confidence intervals, obtained with a linear classifier on top of various layers, for the various sample efficiency regimes. Results obtained with supervised backprop (BP), VAE-based semi-supervised approach (VAE), Hebbian PCA (HPCA), and HPCA plus Fine Tuning (HPCA + FT) are compared. It is possible to observe that, in regimes where the number of available samples is low (roughly between 1% and 5% of the total available samples), HPCA performs better than BP and VAE approaches in almost all the cases, leading to an improvement up to almost 5% (on Layer 3, in the 1% regime) w.r.t. non-Hebbian approaches. HPCA + FT helps to further boost accuracy.

Regimes	Method	L1	L2	L3	L4	L5
1%	BP	33.27 ±0.44	34.56 ±0.34	36.80 ±0.52	35.47 ±0.58	35.18 ±0.57
	VAE	33.54 ±0.27	34.41 ±0.84	29.92 ±1.25	24.91 ±0.66	22.54 ±0.60
	HPCA	36.78 ±0.46	37.26 ±0.14	41.31 ±0.57	39.33 ±0.72	38.46 ±0.44
	HPCA + FT	**37.01** ±0.42	**37.65** ±0.19	**41.88** ±0.53	**40.06** ±0.65	**39.75** ±0.50
2%	BP	37.33 ±0.25	39.01 ±0.19	42.34 ±0.51	41.50 ±0.32	41.10 ±0.39
	VAE	37.65 ±0.35	39.13 ±0.40	36.52 ±0.47	29.39 ±0.32	26.78 ±0.72
	HPCA	41.13 ±0.30	**41.63** ±0.18	45.76 ±0.41	44.70 ±0.45	43.15 ±0.45
	HPCA + FT	**41.60** ±0.28	42.12 ±0.24	**46.56** ±0.38	**45.61** ±0.19	**45.51** ±0.43
3%	BP	40.49 ±0.26	41.90 ±0.40	45.13 ±0.53	45.26 ±0.22	44.52 ±0.24
	VAE	41.22 ±0.27	43.16 ±0.44	42.60 ±0.87	31.91 ±0.44	29.00 ±0.33
	HPCA	44.16 ±0.42	44.84 ±0.08	48.92 ±0.17	47.70 ±0.57	45.60 ±0.27
	HPCA + FT	**44.74** ±0.08	**45.61** ±0.28	**49.75** ±0.41	**48.94** ±0.45	**48.80** ±0.27
4%	BP	43.38 ±0.22	45.43 ±0.18	49.51 ±0.49	48.96 ±0.48	48.80 ±0.24
	VAE	44.39 ±0.30	45.88 ±0.39	46.01 ±0.40	34.26 ±0.21	31.15 ±0.35
	HPCA	46.37 ±0.16	47.16 ±0.28	50.70 ±0.26	49.45 ±0.15	47.75 ±0.54
	HPCA + FT	**47.10** ±0.25	**48.26** ±0.09	**52.00** ±0.16	**51.05** ±0.29	**51.28** ±0.28
5%	BP	45.11 ±0.21	47.57 ±0.29	50.61 ±0.32	50.54 ±0.23	50.42 ±0.14
	VAE	46.31 ±0.39	48.21 ±0.21	48.98 ±0.34	36.32 ±0.35	32.75 ±0.32
	HPCA	47.51 ±0.65	48.69 ±0.37	51.69 ±0.56	50.44 ±0.43	48.51 ±0.32
	HPCA + FT	**48.49** ±0.44	**50.14** ±0.46	**53.33** ±0.52	**52.49** ±0.16	**52.20** ±0.37
10%	BP	51.60 ±0.40	54.60 ±0.31	57.97 ±0.28	**57.63** ±0.23	57.30 ±0.22
	VAE	53.83 ±0.26	**56.33** ±0.22	57.85 ±0.22	52.26 ±1.08	45.67 ±1.15
	HPCA	52.57 ±0.29	53.29 ±0.25	56.09 ±0.38	54.24 ±0.28	52.68 ±0.36
	HPCA + FT	**54.36** ±0.32	56.08 ±0.28	**58.46** ±0.15	56.54 ±0.23	**57.35** ±0.18
25%	BP	60.43 ±0.26	64.96 ±0.18	66.63 ±0.17	68.04 ±0.05	68.04 ±0.20
	VAE	**62.51** ±0.24	**67.26** ±0.32	**68.48** ±0.21	**68.79** ±0.29	**68.70** ±0.15
	HPCA	58.30 ±0.28	59.20 ±0.24	59.98 ±0.20	57.54 ±0.20	56.46 ±0.18
	HPCA + FT	61.45 ±0.26	65.25 ±0.16	64.71 ±0.17	62.43 ±0.13	64.77 ±0.22
100%	BP	61.59 ±0.08	67.67 ±0.11	73.87 ±0.15	83.88 ±0.04	84.71 ±0.02
	VAE	**67.53** ±0.22	**75.83** ±0.31	**80.78** ±0.28	**84.27** ±0.35	**85.23** ±0.26
	HPCA	64.69 ±0.29	65.92 ±0.14	64.43 ±0.21	61.24 ±0.22	61.16 ±0.33
	HPCA + FT	66.76 ±0.13	75.16 ±0.20	79.90 ±0.18	83.55 ±0.33	84.38 ±0.22

given that these datasets contain a much larger number of classes, i.e. 100 and 200, respectively.

7.1. CIFAR10

Table 1 reports the top-1 accuracy results obtained on the CIFAR10 dataset. We only report top-1 accuracy, given that CIFAR10 contains only 10 classes.

At a first glance, we see that in regimes where a limited number of labeled samples is available (between 1% and 5%), the HPCA achieves better results than the BP and VAE counterparts, in almost all the cases. On the other hand, when the number of available labeled samples becomes larger, BP and VAE approaches (which exploit end-to-end fine tuning in the supervised phase) are able to take advantage of supervision and improve over HPCA. When HPCA+FT is considered, we can observe that end-to-end fine tuning helps to boost accuracy. Still, VAE pre-training performs better in regimes where more labeled samples are available (beyond 10%), while HPCA and HPCA+FT are preferable in regimes with fewer labeled samples.

Comparing HPCA with the BP approach, we see that, for efficiency regimes up to 3%, HPCA is better than BP when tested in all layers of the network. We can observe that HPCA generally outperforms backprop by roughly 1–3 percent points, reaching a peak of almost 5 percent points on Layer 3, in the 1% sample efficiency regime. At 4% efficiency regime, we note that HPCA is still performing better than BP for the first 4 layers, while BP performs better than HPCA when the linear classifier is put on top of the fifth layer. This effect continues for higher efficiency regimes, where we see that increasing the amount of labeled samples, reduces the highest layer where HPCA has better performance than BP. So, we observe that for 5% efficiency regimes HPCA is better than BP up to Layer 3. Still, in low sample efficiency regimes (between 1% and 5%), HPCA outperforms backprop in almost all the cases. At 10% efficiency regimes HPCA is better only when the linear classifier is trained on top of Layer 1. BP always outperforms HPCA when 100% labeled examples are used. To explain this behavior we can observe that, on one hand, when the amount of labeled samples increases, BP is able to effectively take advantage of the supervised information and extract more useful knowledge from training data. This starts to be seen from the highest layers of the network, where the supervision signal is stronger. Increasing the amount of labeled training data brings this effect down up to the first network layer. On the other hand, unsupervised Hebbian learning signal, which is driven by the inputs, is stronger in the first layers, where coherence between input and output of neurons is more meaningful, and layers tend to adapt faster to the unsupervised stimuli.

Comparing HPCA with VAE approach, we see that, when low sample efficiency regimes are considered (between 1% and 5%) Hebbian approaches always achieve significantly higher results than VAE. Only when the number of available labeled samples increases (beyond 10%), VAE pre-training starts to become really competitive, obtaining results comparable to or higher than HPCA. In these scenarios, VAE pre-training also helps improving performance w.r.t. plain BP training from scratch. We can also observe that, in low sample efficiency regimes (10% or less), the VAE approach suffers from a decrease in performance when going deeper with the number of layers. This issue is common with unsupervised methods, because the lack of a supervision signal

Table 2
CIFAR100 accuracy (top-5) and 95% confidence intervals obtained with a linear classifier on top of various layers, for the various sample efficiency regimes. Results obtained with supervised backprop (BP), VAE-based semi-supervised approach (VAE), Hebbian PCA (HPCA), and HPCA plus Fine Tuning (HPCA + FT) are compared. It is possible to observe that, in regimes where the number of available samples is low (roughly between 1% and 5% of the total available samples), HPCA performs better than BP and VAE approaches in almost all the cases. Even though, in various cases, the improvement is small, it becomes significant in some scenarios, where peaks of improvement up to 2% are observed (on Layers 3 and 4) w.r.t. non-Hebbian approaches. HPCA + FT helps to further boost accuracy.

Regimes	Method	L1	L2	L3	L4	L5
1%	BP	**22.56** ±0.53	**22.73** ±0.28	23.41 ±0.44	20.85 ±0.58	21.88 ±0.30
	VAE	21.69 ±0.10	21.70 ±0.30	17.61 ±0.54	13.45 ±0.54	12.28 ±0.50
	HPCA	21.90 ±0.33	22.23 ±0.50	22.98 ±0.18	20.88 ±0.43	21.90 ±0.55
	HPCA + FT	22.30 ±0.38	22.28 ±0.63	**23.58** ±0.21	**21.70** ±0.61	**22.63** ±0.55
2%	BP	28.99 ±0.49	29.07 ±0.68	30.75 ±0.34	27.67 ±0.37	28.18 ±0.35
	VAE	28.24 ±0.13	28.42 ±0.31	23.56 ±0.73	17.01 ±0.37	15.25 ±0.63
	HPCA	29.08 ±0.31	**29.40** ±0.23	32.22 ±0.28	29.20 ±0.46	28.95 ±0.35
	HPCA + FT	**29.65** ±0.52	26.57 ±0.26	**33.20** ±0.20	**30.21** ±0.54	**30.83** ±0.35
3%	BP	31.77 ±0.42	32.56 ±0.51	34.06 ±0.41	31.81 ±0.33	32.45 ±0.23
	VAE	31.28 ±0.54	31.71 ±0.27	27.46 ±1.23	18.26 ±0.24	16.44 ±0.12
	HPCA	32.07 ±0.46	33.04 ±0.30	36.41 ±0.15	33.67 ±0.39	32.61 ±0.51
	HPCA + FT	**32.81** ±0.18	**33.08** ±0.55	**37.75** ±0.38	**35.02** ±0.36	**35.04** ±0.17
4%	BP	34.74 ±0.29	35.88 ±0.30	37.63 ±0.19	35.92 ±0.35	36.52 ±0.37
	VAE	34.60 ±0.10	35.44 ±0.31	32.34 ±0.79	19.68 ±0.32	17.89 ±0.27
	HPCA	35.34 ±0.40	35.97 ±0.27	39.85 ±0.35	37.23 ±0.19	36.05 ±0.37
	HPCA + FT	**36.13** ±0.39	**36.23** ±0.20	**41.21** ±0.39	**39.16** ±0.32	**38.89** ±0.15
5%	BP	36.84 ±0.23	37.70 ±0.32	39.70 ±0.21	38.42 ±0.32	39.21 ±0.65
	VAE	36.68 ±0.17	37.26 ±0.26	35.33 ±0.81	20.55 ±0.44	18.48 ±0.26
	HPCA	37.28 ±0.40	37.75 ±0.24	42.12 ±0.49	39.37 ±0.18	37.84 ±0.22
	HPCA + FT	**38.03** ±0.20	**38.02** ±0.25	**43.76** ±0.33	**41.66** ±0.20	**41.42** ±0.23
10%	BP	42.04 ±0.24	**44.98** ±0.23	48.39 ±0.22	48.98 ±0.35	**49.84** ±0.34
	VAE	42.64 ±0.34	44.84 ±0.48	46.04 ±0.44	27.81 ±0.13	23.80 ±0.60
	HPCA	43.05 ±0.36	43.93 ±0.23	48.68 ±0.27	46.05 ±0.24	43.87 ±0.28
	HPCA + FT	**43.51** ±0.34	44.84 ±0.26	**50.84** ±0.22	**49.53** ±0.19	48.93 ±0.38
25%	BP	53.36 ±0.10	**59.11** ±0.21	60.94 ±0.15	**64.57** ±0.26	**67.17** ±0.16
	VAE	**53.53** ±0.12	57.63 ±0.52	**62.16** ±0.57	55.29 ±0.68	52.59 ±1.02
	HPCA	49.62 ±0.36	51.30 ±0.25	56.14 ±0.29	53.46 ±0.28	51.29 ±0.15
	HPCA + FT	51.51 ±0.31	54.22 ±0.23	59.60 ±0.44	58.29 ±0.29	58.70 ±0.18
100%	BP	51.67 ±0.10	60.84 ±0.19	67.01 ±0.13	78.85 ±0.10	**80.74** ±0.05
	VAE	**67.51** ±0.11	**73.83** ±0.30	**78.70** ±0.23	**79.58** ±0.18	79.97 ±0.14
	HPCA	60.94 ±0.09	62.24 ±0.15	64.17 ±0.22	61.27 ±0.24	59.51 ±0.20
	HPCA + FT	65.61 ±0.12	70.38 ±0.23	74.10 ±0.12	73.38 ±0.18	74.42 ±0.14

(or still its scarcity, in case of semi-supervised scenarios) makes it more difficult to develop task-specific feature detectors on higher layers, which are essential to reach higher performances, as also previous studies on deep CNNs reveal (Agrawal, Girshick, & Malik, 2014). With HPCA, this problem seems to alleviate, and the accuracy remains pretty much constant with the number of layers, meaning that the features produced by this approach are more meaningful for the classification task. Only when the amount of supervision, i.e. the number of labeled samples, becomes large enough (above 10%), the end-to-end supervised training phase, that follows VAE pre-training in the semi-supervised approach, manages to transform VAE feature detectors to task-specific features that perform even better than those obtained by BP training from scratch. Overall, these results suggest that VAE-based semi-supervised learning is better suited in sample efficiency regimes where the labeled portion of the dataset is still relatively large (10% or more), while our method is preferable to address sample efficiency regimes in which the number of available labeled samples is *very* small (5% or less).

The HPCA+FT strategy is still preferable in low sample efficiency regimes (between 1% and 5%), where it helps to further increase accuracy w.r.t. plain HPCA. In particular, in these regimes, we can observe a further increase in accuracy up to 2% points on network Layers 3 and 4, and up to 4% points on Layer 5 (in the 4% and 5% regimes). Fine tuning also helps increasing accuracy in successive sample efficiency regimes, especially on higher layers.

Please note that in some configurations, the accuracy of pure HPCA (even without fine-tuning on deep layers) is higher than the accuracy obtained with BP. Consider, for instance, the accuracy obtained at L5 for configurations up to 3% sample efficiency regimes. This might appear strange since, in principle, label information used by BP should help to achieve higher accuracy. The explanation is that BP training tends to generalize poorly with low sample efficiency regimes, when just a limited number of labeled samples is available. On the other hand, unsupervised HPCA training exploits a large number of unlabeled samples, and this allows us to achieve generally higher accuracy in the aforementioned cases.

7.2. CIFAR100

Since CIFAR10 contained just 10 different classes, to validate our observations with a similar, yet more difficult scenario, we also performed tests with CIFAR100, containing 100 classes. In Table 2 the top-5 accuracy results obtained on the CIFAR100 dataset are shown. In this case, we report top-5 accuracy instead of top-1, given that CIFAR100 contains a much larger number of classes than the previous dataset.

These experiments confirm our previous observations. The results show that, in regimes where a limited number of labeled samples is available (between 1% and 5%), our semi-supervised approach, based on Hebbian learning, achieves better results than BP and VAE counterparts in almost all the cases. On the other hand, when the number of available labeled samples becomes larger, BP and VAE approaches (which exploit end-to-end fine tuning in the supervised phase) are able to take advantage of supervision and improve over HPCA. Also in this case, we can observe that the end-to-end fine tuning in HPCA+FT helps to further boost accuracy.

Table 3
Tiny ImageNet accuracy (top-5) and 95% confidence intervals obtained with a linear classifier on top of various layers, for the various sample efficiency regimes. Results obtained with supervised backprop (BP), VAE-based semi-supervised approach (VAE), Hebbian PCA (HPCA), and HPCA plus Fine Tuning (HPCA + FT) are compared. It is possible to observe that, in regimes where the number of available samples is low (roughly between 1% and 5% of the total available samples), HPCA performs better than BP and VAE approaches in almost all the cases, leading to an improvement up to almost 3% (on Layer 3, in the 4% regime) w.r.t. non-Hebbian approaches. HPCA + FT helps to further boost accuracy.

Regimes	Method	L1	L2	L3	L4	L5
1%	BP	9.89 ±0.15	10.10 ±0.26	9.99 ±0.17	9.15 ±0.23	9.53 ±0.21
	VAE	9.63 ±0.26	9.49 ±0.39	7.58 ±0.28	5.99 ±0.19	5.55 ±0.23
	HPCA	**10.83** ±0.28	10.87 ±0.26	11.85 ±0.19	10.84 ±0.26	10.86 ±0.23
	HPCA+FT	10.81 ±0.27	**10.99** ±0.36	**12.15** ±0.46	**11.05** ±0.27	**11.38** ±0.41
2%	BP	12.76 ±0.27	12.84 ±0.14	13.95 ±0.34	13.04 ±0.15	13.48 ±0.39
	VAE	12.94 ±0.37	13.06 ±0.23	10.86 ±0.28	7.40 ±0.27	6.74 ±0.20
	HPCA	13.84 ±0.17	**14.35** ±0.15	16.18 ±0.15	14.52 ±0.32	14.03 ±0.15
	HPCA + FT	**14.12** ±0.23	14.32 ±0.31	**16.89** ±0.61	**15.28** ±0.28	**15.71** ±0.47
3%	BP	14.12 ±0.20	14.65 ±0.57	16.50 ±0.32	15.76 ±0.27	15.99 ±0.38
	VAE	14.31 ±0.18	15.17 ±0.20	13.67 ±0.36	8.35 ±0.29	7.74 ±0.19
	HPCA	16.13 ±0.14	16.32 ±0.33	18.87 ±0.29	17.04 ±0.26	16.38 ±0.25
	HPCA + FT	**16.25** ±0.21	**16.54** ±0.28	**19.78** ±0.47	**18.31** ±0.24	**18.23** ±0.33
4%	BP	15.44 ±0.42	16.72 ±0.31	18.36 ±0.22	17.85 ±0.16	17.84 ±0.19
	VAE	16.09 ±0.20	17.05 ±0.20	16.83 ±0.51	8.86 ±0.11	8.45 ±0.21
	HPCA	17.64 ±0.49	18.27 ±0.34	21.07 ±0.17	19.16 ±0.33	18.13 ±0.39
	HPCA + FT	**17.70** ±0.44	**18.33** ±0.24	**21.95** ±0.57	**20.86** ±0.32	**20.55** ±0.28
5%	BP	16.75 ±0.25	17.94 ±0.25	20.26 ±0.21	20.15 ±0.35	19.84 ±0.36
	VAE	17.44 ±0.26	18.62 ±0.32	19.16 ±0.52	9.92 ±0.24	9.29 ±0.17
	HPCA	18.93 ±0.14	19.67 ±0.36	22.65 ±0.35	21.01 ±0.38	19.57 ±0.15
	HPCA + FT	**19.26** ±0.41	**19.93** ±0.41	**23.97** ±0.52	**22.95** ±0.26	**22.46** ±0.17
10%	BP	20.26 ±0.18	23.12 ±0.14	27.05 ±0.20	27.30 ±0.20	27.21 ±0.29
	VAE	21.62 ±0.25	23.83 ±0.19	27.42 ±0.18	16.69 ±0.18	13.51 ±0.34
	HPCA	22.15 ±0.43	23.69 ±0.24	27.02 ±0.24	25.73 ±0.34	23.08 ±0.17
	HPCA + FT	**22.82** ±0.33	**24.34** ±0.29	**28.69** ±0.36	**28.79** ±0.26	**28.13** ±0.38
25%	BP	28.97 ±0.26	**32.63** ±0.36	37.38 ±0.13	**38.81** ±0.20	**38.80** ±0.39
	VAE	**29.40** ±0.31	32.42 ±0.29	**39.93** ±0.31	37.97 ±0.62	37.89 ±0.54
	HPCA	27.05 ±0.47	28.39 ±0.34	32.08 ±0.19	31.30 ±0.26	29.51 ±0.23
	HPCA + FT	28.01 ±0.75	30.63 ±0.16	35.87 ±0.53	36.98 ±0.26	37.10 ±0.23
100%	BP	**42.89** ±0.13	**49.94** ±0.13	54.54 ±0.27	57.00 ±0.16	57.50 ±0.16
	VAE	42.32 ±0.16	48.54 ±0.53	**58.31** ±0.12	**59.60** ±0.23	**60.23** ±0.65
	HPCA	35.74 ±0.15	38.29 ±0.19	38.78 ±0.07	38.61 ±0.21	36.99 ±0.36
	HPCA + FT	40.34 ±0.31	45.00 ±0.40	53.12 ±0.26	52.95 ±0.28	53.96 ±0.43

Except for 1% sample efficiency regime, where the difference in the results is not really significant, for regimes up to 3% HPCA is always better than BP. In particular, the improvement becomes significant in correspondence to network Layers 3 and 4 in the sample efficiency regimes between 2% and 5%. Note also that Layer 3 generally offers absolute highest accuracy for all efficiency regimes, when HPCA is used. In these cases, we observe peaks of improvement over 2 percent points. As before, for efficiency regimes higher than 4% BP starts to provide better accuracy in the higher layers of the network. However, HPCA is still better than BP, with 100% efficiency regime, when tested on Layers 1 and 2. This can be explained by the fact that CIFAR100 offers a more difficult scenario than CIFAR10, and BP has more problems than before in backpropagating the error signal to the initial layers of the network. Differently, unsupervised Hebbian learning has most of its effects in the very first layers of the network.

Also in this case, we observe that HPCA always performs better than VAE method when low sample efficiency regimes are considered (between 1% and 5%), especially for higher network layers. Again, VAE pre-training seems to be more effective in regimes where more labeled samples are available (beyond 10%).

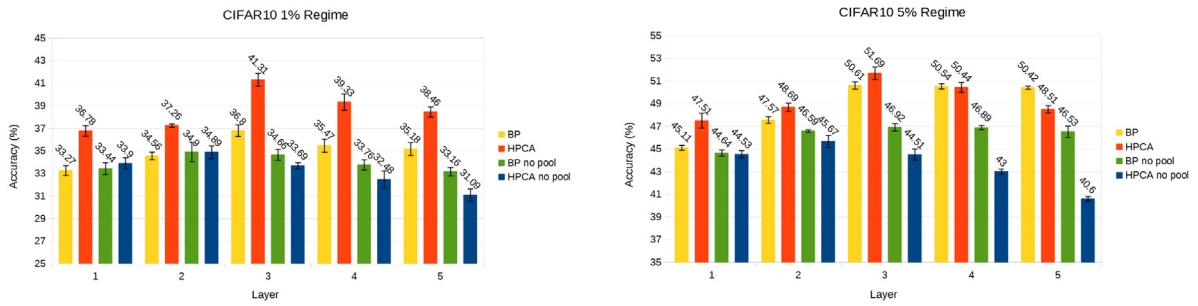
The HPCA+FT strategy is still preferable in low sample efficiency regimes (between 1% and 5%), where it helps to further increase accuracy w.r.t. plain HPCA. In particular, in these regimes, we can observe a further increase in accuracy up to 4% points on Layer 5 (in the 5% regime). Fine tuning also helps increasing accuracy in successive sample efficiency regimes, especially on higher layers.

7.3. Tiny ImageNet

Further experiments on Tiny ImageNet allowed us to validate the scalability of our previous observations to larger datasets. Tiny ImageNet has 200 classes and the training set consists of 100,000 samples (90,000 of which are used for training and 10,000 for validation). Results are reported in Table 3, where the top-5 accuracy measures are shown, along with their 95% confidence interval. Also in this case, we report top-5 accuracy instead of top-1, given that Tiny ImageNet contains a large number of classes, as opposed to other datasets such as CIFAR10.

Again, results confirm our observations. In regimes where a limited number of labeled samples is available (between 1% and 5%), the Hebbian approach outperforms BP and VAE counterparts, in almost all the cases. On the other hand, when the number of available labeled samples becomes larger, BP and VAE approaches (which exploit end-to-end fine tuning in the supervised phase) are able to take advantage of supervision and improve over HPCA. Also in this case, we can observe that the end-to-end fine tuning in HPCA+FT helps to further boost accuracy.

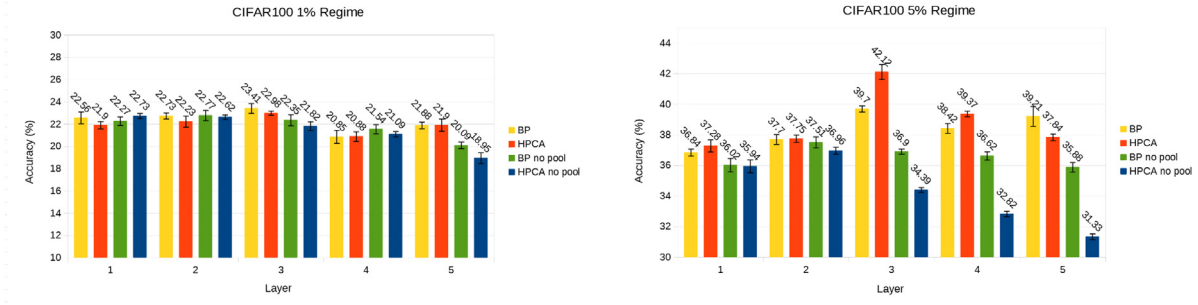
Specifically, HPCA outperforms BP in all layers up to 4% sample efficiency regime. In fact, with CIFAR10 and CIFAR100, BP started to outperform HPCA on Layer 5 at 4% regimes. Here, with 4% sample efficiency regime, HPCA is still better than BP in all layers. This is probably due to the fact that the number of classes in Tiny ImageNet is higher and using just a few samples does not allow back propagation to correctly adapt the behavior of network layers. In addition, we can observe that HPCA generally outperforms backprop by roughly 1–2 percent points, reaching a peak of almost 3 percent points on Layer 3, in the 4% sample



(a) Accuracy results (top-1) in the 1% sample efficiency regime.

(b) Accuracy results (top-1) in the 5% sample efficiency regime.

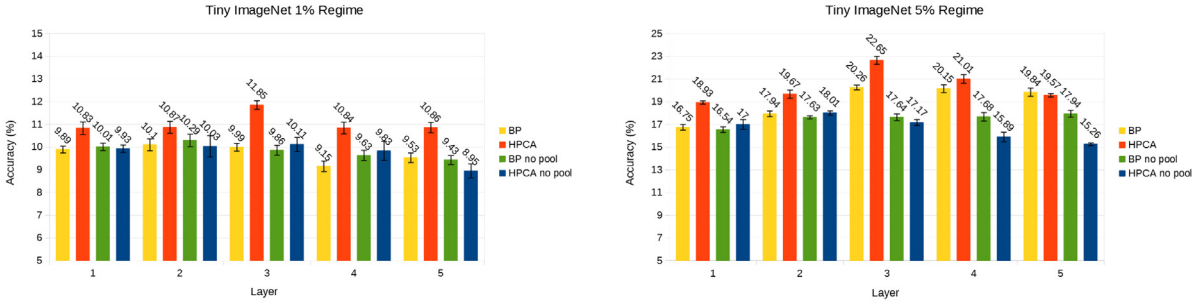
Fig. 7. Comparison of network architectures with and without pooling on the CIFAR10 dataset. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Accuracy results (top-5) in the 1% sample efficiency regime.

(b) Accuracy results (top-5) in the 5% sample efficiency regime.

Fig. 8. Comparison of network architectures with and without pooling on the CIFAR100 dataset. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Accuracy results (top-5) in the 1% sample efficiency regime.

(b) Accuracy results (top-5) in the 5% sample efficiency regime.

Fig. 9. Comparison of network architectures with and without pooling on the Tiny ImageNet dataset. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

efficiency regime. With higher efficiency regimes, BP begins to outperform HPCA, starting from the higher layers. At 100% sample efficiency regime, BP outperforms HPCA on all layers. This is probably due to the fact that 90,000 labeled training samples are sufficient for BP to correctly train all network layers, exploiting the supervised information.

Also in this case, we observe that HPCA always performs better than VAE method when low sample efficiency regimes are considered (between 1% and 5%), especially for higher network layers. Again, VAE pre-training seems to be more effective in regimes where more labeled samples are available (beyond 10%).

The HPCA+FT strategy is still preferable in low sample efficiency regimes (between 1% and 5%), where it helps to further increase accuracy w.r.t. plain HPCA. In particular, in these regimes, we can observe a further increase in accuracy up to 3% points on Layer 5 (in the 5% regime). Fine tuning also helps increasing accuracy in successive sample efficiency regimes, especially on higher layers.

7.4. Effect of pooling layers

From the results presented so far, it is possible to observe that higher results for low sample efficiency regimes (1%–5%) are generally achieved in correspondence to Layer 3. We note also that in Layers 1 and 3 of our network, as show in Fig. 1, we have max-pooling operations. We performed further experiments in order to evaluate the impact of pooling layers on the final results. In this subsection, we discuss the results of previous experiments executed in a network where all max-pooling operations were eliminated. In Figs. 7, 8, 9, we show the accuracy obtained for the 1% and 5% sample efficiency regimes, on the CIFAR10, CIFAR100 and Tiny ImageNet datasets. Yellow and red bars correspond to experiments executed with the original network, trained with BP and HPCA, respectively. Green and blue bars correspond to experiments executed with the network where max-pooling was eliminated, also trained with BP and HPCA respectively. We show

the results for BP and HPCA, because these are the scenarios in which the aforementioned effect is more prominent.

It can be clearly seen the peak of accuracy, occurring at Layer 3, for all experiments executed with the original network. Similarly, we can see that the peak disappeared with the experiments executed without max-pooling. In fact, it seems that the peak moved at Layer 2. However, it is within the reported confidence interval of the adjacent layers, so the difference is not statistically significant. In addition, when max-pooling is not used, we report a general decrease of performance. This confirms that max-pooling is a relevant element and its use significantly helps our semi-supervised approach. Note that also in case of BP experiments, without max-pooling, significantly lower results were obtained w.r.t. the highest accuracy, which was achieved with HPCA training in conjunction with pooling layers.

Max-pooling clearly helps the network to produce better feature maps to be used by the linear classifier. In fact, adding consecutive convolutional layers produces neurons with increasingly larger receptive field size. However neurons activations corresponding to adjacent areas, in a given feature map, will be highly correlated. The effect of pooling is to reduce this correlation. This turns out to be very helpful for the final classifiers, which can better handle feature maps of lower dimension and with less redundant information, making it easier to discover relationships between features and target classes.

It can also be observed that, when the pooling layer is removed, the accuracy drop of HPCA is larger than that of BP. Again this can be justified by considering that, without pooling, neuron activations corresponding to adjacent areas will be highly correlated. Nonetheless, when backprop training is used, the supervision signal can drive network weights in order to produce feature representations that reduce such correlation. With unsupervised Hebbian training, this is not possible, and therefore the resulting model suffers a higher performance drop when correlations are incentivized due to the removal of the pooling layers. So our conclusion is that pooling operations play a relevant role in Hebbian training.

8. Conclusions and future work

In summary, our results suggest that our semi-supervised approach leveraging Hebbian learning is preferable w.r.t. backprop training (from scratch or with VAE pre-training) to perform training in low sample efficiency regimes where only a limited number of labeled samples is available. Specifically, our results on CIFAR10, CIFAR100, and Tiny ImageNet show that HPCA performs better than backprop training (from scratch or with VAE pre-training) in sample efficiency regimes in which only a small portion of the training set (between 1% and 5%) is assumed to be labeled. In addition, the HPCA+FT approach helps to further improve performance. Therefore, our method is preferable in scenarios in which manually labeling a large number of training samples would be too expensive, while gathering unlabeled samples is relatively cheap.

In future work, further improvements might come from exploring more complex feature extraction strategy, which can also be formulated as Hebbian learning variants, such as Kernel-PCA (Schölkopf, Smola, & Müller, 1998) and Independent Component Analysis (ICA) (Hyvarinen, Karhunen, & Oja, 2002). In addition, it would be interesting to replicate this work also to the context of Spiking Neural Networks (SNNs), where the Hebbian principle is implemented by the Spike Timing Dependent Plasticity (STDP) learning rule (Gerstner & Kistler, 2002). SNNs are more realistic models of biological neural computation, which use pulses (called *spikes*) to encode signals, rather than continuous values. This communication paradigm is the key toward energy-efficient computation in the brain (Javed et al., 2010), and is being currently implemented in *neuromorphic hardware* (Furber, Galluppi, Temple, & Plana, 2014; Wu, Saxena, Zhu, & Balagopal, 2015). In this scenario, it is necessary to map the variants of the Hebbian rule to corresponding STDP variants and test their effectiveness for SNN training. Finally, an exploration on the behavior of such algorithms w.r.t. adversarial examples also deserves attention.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by the H2020 project AI4EU, Italy under GA 825619 and by the H2020 project AI4Media, Italy under GA 951911.

References

Agrawal, P., Girshick, R., & Malik, J. (2014). Analyzing the performance of multilayer neural networks for object recognition. arXiv preprint arXiv:1407.1610.

Amato, G., Carrara, F., Falchi, F., Gennaro, C., & Lagani, G. (2019). Hebbian learning meets deep convolutional neural networks. In *International conference on image analysis and processing* (pp. 324–334). Springer.

Bahroun, Y., & Soltoggio, A. (2017). Online representation learning with single and multi-layer hebbian networks for image classification. In *International conference on artificial neural networks* (pp. 354–363). Springer.

Becker, S., & Plumbley, M. (1996). Unsupervised neural network learning procedures for feature extraction and classification. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *6*(3), 185–203.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems* (pp. 153–160).

Canto, F. J. A. (2020). Convolutional neural networks with hebbian-based rules in online transfer learning. In *Mexican international conference on artificial intelligence* (pp. 35–49). Springer.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607). PMLR.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Földiak, P. (1990). Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, *64*(2), 165–170.

Furber, S. B., Galluppi, F., Temple, S., & Plana, L. A. (2014). The spinnaker project. *Proceedings of the IEEE*, *102*(5), 652–665.

Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models: single neurons, populations, plasticity*. Cambridge university press.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. *Biological Cybernetics*, *23*(3), 121–134.

Haykin, S. (2009). *Neural networks and learning machines* (3rd ed.). Pearson.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., et al. (2016). Beta-vae: Learning basic visual concepts with a constrained variational framework.

Hyvarinen, A., Karhunen, J., & Oja, E. (2002). Independent component analysis. *Studies in Informatics and Control*, *11*(2), 205–207.

Javed, F., He, Q., Davidson, L. E., Thornton, J. C., Albu, J., & Boxt, L. others (2010). Brain and high metabolic rate organ mass: contributions to resting energy expenditure beyond fat-free mass. *The American Journal of Clinical Nutrition*, *91*(4), 907–912.

Karhunen, J., & Joutsensalo, J. (1995). Generalizations of principal component analysis, optimization problems, and neural networks. *Neural Networks*, *8*(4), 549–562.

Kingma, D. P., Mohamed, S., Jimenez Rezende, D., & Welling, M. (2014). Semi-supervised learning with deep generative models. *Advances in Neural Information Processing Systems*, *27*, 3581–3589.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*(1), 59–69.
Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*.
Krotov, D., & Hopfield, J. J. (2019). Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences*, *116*(16), 7723–7731.
Lagani, G. (2019). *Hebbian learning algorithms for training convolutional neural networks* (Master's thesis), Italy: School of Engineering, University of Pisa.
Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks.. *Journal of Machine Learning Research*, *10*(1).
Magotra, A., & kim, J. (2019). Transfer learning for image classification using hebbian plasticity principles. In *Proceedings of the 2019 3rd international conference on computer science and artificial intelligence* (pp. 233–238).
Magotra, A., & Kim, J. (2020). Improvement of heterogeneous transfer learning efficiency by using hebbian learning principle. *Applied Sciences*, *10*(16), 5631.
Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*(6583), 607.
O'Reilly, R. C., & Munakata, Y. (2000). *Computational explorations in cognitive neuroscience: understanding the mind by simulating the brain*. MIT press.
Pehlevan, C., & Chklovskii, D. B. (2015). Optimization theory of hebbian/anti-hebbian networks for pca and whitening. In *2015 53rd annual allerton conference on communication, control, and computing (allerton)* (pp. 1458–1465). IEEE.
Pehlevan, C., Hu, T., & Chklovskii, D. B. (2015). A hebbian/anti-hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data. *Neural Computation*, *27*(7), 1461–1495.
Rasmus, A., Berglund, M., Honkala, M., Valpola, H., & Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in neural information processing systems* (pp. 3546–3554).
Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, *2*(6), 459–473.
Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319.
Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., & Van Den Driessche, G. others (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484.
Wadhwa, A., & Madhow, U. (2016a). Bottom-up deep learning using the hebbian principle.
Wadhwa, A., & Madhow, U. (2016b). Learning sparse, distributed representations using the hebbian principle. arXiv preprint arXiv:1611.04228.
Weston, J., Chopra, S., & Bordes, A. (2014). Memory networks. arXiv preprint arXiv:1410.3916.
Weston, J., Ratle, F., Mobahi, H., & Collobert, R. (2012). Eep learning via semi-supervised embedding. In *Neural networks: tricks of the trade* (pp. 639–655). Springer.
Wu, X., Saxena, V., Zhu, K., & Balagopal, S. (2015). A cmos spiking neuron for brain-inspired neural networks with resistive synapses andin situlearning. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *62*(11), 1088–1092.
Wu, J., Zhang, Q., & Xu, G. (2017). *Tiny imagenet challenge*: *Tech. rep. technical report*, Stanford University, Available online at http …(2017).
Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. arXiv preprint arXiv:1411.1792.
Zhang, Y., Lee, K., & Lee, H. (2016). Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *International conference on machine learning* (pp. 612–621).