Evaluating Hebbian Learning in a Semi-Supervised Setting *

Gabriele Lagani[1], Fabrizio Falchi[2], Claudio Gennaro[2], and Giuseppe Amato[2]

[1] Dept. of Computer Science, University of Pisa, 56127, Pisa, Italy
gabriele.lagani@phd.unipi.it
[2] ISTI-CNR, 56124, Pisa, Italy
{giuseppe.amato,fabrizio.falchi,claudio.gennaro}@cnr.it

Abstract. We propose a semi-supervised learning strategy for deep Convolutional Neural Networks (CNNs) in which an unsupervised pre-training stage, performed using biologically inspired Hebbian learning algorithms, is followed by supervised end-to-end backprop fine-tuning. We explored two Hebbian learning rules for the unsupervised pre-training stage: soft-Winner-Takes-All (soft-WTA) and nonlinear Hebbian Principal Component Analysis (HPCA). Our approach was applied in sample efficiency scenarios, where the amount of available labeled training samples is very limited, and unsupervised pre-training is therefore beneficial. We performed experiments on CIFAR10, CIFAR100, and Tiny ImageNet datasets. Our results show that Hebbian outperforms Variational Auto-Encoder (VAE) pre-training in almost all the cases, with HPCA generally performing better than soft-WTA.

Keywords: Hebbian Learning · Deep Learning · Semi-Supervised · Sample Efficiency · Neural Networks · Bio-Inspired.

1 Introduction

While deep learning has achieved outstanding results in a variety of domains, ranging from computer vision [15] to language processing [10], and reinforcement learning [41], learning algorithms are typically based on supervised end-to-end Stochastic Gradient Descent (SGD) training with error backpropagation (*back-prop*), which needs a large number of labeled training samples in order to achieve high results. However, gathering labeled samples is expensive, as it requires a significant amount of human work. On the other hand, gathering unlabeled samples is relatively simple. Therefore, researchers started to investigate learning strategies to exploit large amounts of unlabeled data, in addition to the fewer labeled data, for sample efficient learning [5–7, 9, 18, 21, 27, 37, 40, 45, 47]. This led to the *semi-supervised* learning approach, in which an unsupervised pre-training stage is performed on all the available samples (but without using label information),

* This work was partially supported by the H2020 projects AI4EU (GA 825619) and AI4Media (GA 951911).

and then it is followed by a supervised fine-tuning stage on the few labeled samples only (in this case, the label information is used for supervision).

Note that backprop is not considered to be biologically plausible from the neuroscientific community [34]. On the other hand, a biologically motivated learning principle is represented by the Hebbian learning paradigm [12,14]. This approach does not require supervision, nor backpropagation. Since biological brains appear to be able to generalize from few samples, research on Hebbian learning algorithms seems a promising direction.

In this work, we propose a semi-supervised learning approach, in which the unsupervised pre-training step is performed by means of the Hebbian learning paradigm. Two Hebbian learning variants are considered: soft-Winner-Takes-All (soft-WTA) [31], and nonlinear Hebbian Principal Component Analysis (HPCA) [19]. We test our approach in sample efficiency scenarios, performing experiments on CIFAR10, CIFAR100 [23], and Tiny ImageNet [46] datasets. Different regimes of sample efficiency are considered, comparing the results with another popular unsupervised pre-training method, namely the Variational Auto-Encoder (VAE) [20]. The results show that our approach outperforms VAE pre-training in almost all the cases, especially when the number of labeled samples available for the successive supervised fine-tuning stage is low. Moreover, HPCA generally performs better than soft-WTA.

Integration of Hebbian learning and deep learning is still an emerging topic. However, our results are encouraging, motivating further interest in this direction.

The main contributions of this paper are the following:

- For the first time, Hebbian learning approaches are applied in a semi-supervised scenario, in which an unsupervised pre-training stage, based on Hebbian approach, is followed by a supervised end-to-end fine-tuning stage based on SGD and backprop;
- We provide extensive experimental evaluation of the approaches, from a sample efficiency perspective, on different object recognition datasets;

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 introduces the various Hebbian learning strategies that we explored; Section 4 illustrates the sample efficiency problem and defines our semi-supervised approach based on Hebbian learning; Section 5 delves into the details of our experimental setup; In Section 6, the results of our simulations are illustrated; Finally, Section 7 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. This led researchers to exploit a pre-training phase that allowed them to initialize network weights in a region near a good local optimum [5, 27]. In these studies, greedy layerwise pre-training was performed by applying unsupervised autoencoder models, layer by layer . 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 [21, 37, 45, 47]. In [21], Variational Auto-Encoders (VAE) were considered, in order to perform an unsupervised pre-training phase using a limited amount of labeled samples. Also [37] and [47] 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 [44], joint supervised and unsupervised training was again considered, but the unsupervised learning part was based on manifold learning techniques.

Another approach, SimCLR [9], used a Contrastive Loss to perform the unsupervised learning part. The approach relied on data augmentation, in order to produce transformed variants of a given input. The unsupervised loss basically encouraged hidden representations to match for transformed variants generated from the same input.

In this paper, we focus our comparisons on similar methods based on unsupervised pre-training, using VAE pre-training as baseline. Nonetheless, it is also worth mentioning that different approaches to semi-supervised learning were also proposed. For example, in [18, 40], graph-based methods were used to generate *pseudo-labels* for unlabeled samples, which were then used as target during training. In [6, 7, 40], the *mixup* approach was also used: convex combinations of pairs of input samples were generated, and a consistency criterion was imposed that pushed the prediction for the combination to match the corresponding combination of predictions. It should be noticed that our method is not in contrast with these other approaches, but rather they can be integrated together, as also suggested in Section 7.

2.2 Hebbian learning

Several variants of Hebbian learning rules were developed over the years. Some examples are: Hebbian learning with Winner-Takes-All (WTA) competition [13], Hebbian learning for Principal Component Analysis (PCA) [4,14,19,39], Hebbian/anti-Hebbian learning [35,36]. A brief overview is given in Section 3. However, it was only recently that Hebbian learning started gaining attention in the context of DNN training [2, 3, 25, 26, 42, 43].

In [25], a Hebbian learning rule based on inhibitory competition was used to train a neural network composed of fully connected layers. The approach was validated on object recognition tasks. Instead, the Hebbian/anti-Hebbian learning rule developed in [36] was applied in [3] to train convolutional feature extractors. The resulting features were shown to be effective for classification. Convolutional layers were also considered in [42, 43], where a Hebbian approach based on WTA competition was employed in this case.

However, the previous approaches were based on relatively shallow network architectures (2-3 layers). A further step was taken in [2, 26], where a Hebbian WTA learning rule was considered. The learning rule was applied for training a 6-layer Convolutional Neural Network (CNN). 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. The advantage was that Hebbian learning required fewer training epochs, thus suggesting potential applications in the context of transfer learning (see also [8, 28, 29]).

The novelty of our contribution w.r.t. previous work is that, for the first time, we investigate unsupervised Hebbian learning in combination with supervised backprop training, in a semi-supervised fashion. In addition, extensive experimental evaluation is performed.

3 Hebbian learning strategies

Consider a single neuron with weight vector $\mathbf{w}$ and input $\mathbf{x}$. Call $y = \mathbf{w}^T \mathbf{x}$ the neuron output. A learning rule defines a weight update as follows:

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \Delta \mathbf{w} \tag{1}$$

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 Hebbian learning rule, in its simplest form, can be expressed as $\Delta \mathbf{w} = \eta \, y \, \mathbf{x}$ (where η is the learning rate) [12, 14]. Basically, this rule states that the weight on a given synapse is reinforced when the input on that synapse and the output of the neuron are simultaneously high. Therefore, connections between neurons whose activations are correlated are reinforced. In order to prevent weights from growing unbounded, a weight decay term is generally added. In the context of competitive learning [13], this is obtained as follows:

$$\Delta \mathbf{w_i} = \eta \, y_i \, \mathbf{x} - \eta \, y_i \, \mathbf{w_i} = \eta \, y_i \, (\mathbf{x} - \mathbf{w_i}) \tag{2}$$

where the subscript i refers to the i'th neuron in a given network layer. Moreover, the output y_i can be replaced with the result r_i of a competitive nonlinearity, which allows to decorrelate the activity of different neurons. In the Winner-Takes-All (WTA) approach [13], at each training step, the neuron which produces the strongest activation for a given input is called the *winner*. In this case, $r_i = 1$ if the i'th neuron is the winner and 0 otherwise. In other words, only the winner is allowed to perform the weight update, so that it will be more likely for the same neuron to win again if a similar input is presented again in the future.

In this way different neurons are induced to specialize on different patterns. In soft-WTA [31], r_i is computed as $r_i = \frac{y_i}{\sum_j y_j}$. We found this formulation to work poorly in practice, because there is no tunable parameter to cope with the variance of activations. For this reason, we introduce a variant of this approach that uses a *softmax* operation in order to compute r_i:

$$r_i = \frac{e^{y_i/T}}{\sum_j e^{y_j/T}} \tag{3}$$

where T is called the *temperature* hyperparameter (the name comes from statistical mechanics, where this function was first introduced) [11]. The advantage of this formulation is that we can tune the temperature in order to obtain the best performance on a given task, depending on the distribution of the activations.

The Hebbian Principal Component Analysis (HPCA) learning rule, in the case of nonlinear neurons, is obtained by minimizing the so-called *representation error* [4,14,39]:

$$L(\mathbf{w_i}) = E[(x - \sum_{j=1}^{i} f(y_j)\,\mathbf{w_j})^2] \tag{4}$$

where $f()$ is the neuron activation function. Minimization of this objective leads to the nonlinear HPCA rule [19]:

$$\varDelta \mathbf{w_i} = \eta f(y_i)(x - \sum_{j=1}^{i} f(y_j)\mathbf{w_j}) \tag{5}$$

It can be noticed that these learning rules do not require supervision, and they are *local* for each network layer, i.e. they do not require backpropagation. In the next section, we discuss how Hebbian learning is integrated with backprop in a semi-supervised training approach.

4 Sample efficiency scenario and semi-supervised approach based on Hebbian learning

Let's define the *labeled set* $\mathcal{T}_L$ as a collection of elements for which the corresponding label is known. Conversely, the *unlabeled set* $\mathcal{T}_U$ is a collection of elements whose labels are unknown. The whole *training set* $\mathcal{T}$ is given by the union of $\mathcal{T}_L$ and $\mathcal{T}_U$. All the samples from $\mathcal{T}$ are assumed to be drawn from the same statistical distribution. In a *sample efficiency* scenario, the number of samples in $\mathcal{T}_L$ is typically much smaller than the total number of samples in $\mathcal{T}$. In particular, an $s\,\%$-sample efficiency *regime* is characterized by $|\mathcal{T}_L| = \frac{s}{100}|\mathcal{T}|$ (where $|\cdot|$ denotes the cardinality of a set, i.e. the number of elements inside the set). In other words, the size of the labeled set is $s\,\%$ that of the whole training set (labeled plus unlabeled).

Traditional supervised approaches based on SGD and backprop work well provided that the size of the labeled set is sufficiently large, but they do not exploit the unlabeled set. To tackle this limitation, we consider a semi-supervised approach in two phases. During the first phase, latent representations are obtained from hidden layers of a DCNN, which are trained using unsupervised Hebbian learning. Such approach, inspired by biology, has the advantage of being able to learn representations without requiring label information, nor back-propagation. This unsupervised pre-training is performed on all the available training samples, unlabeled and labeled (but without using label information in the latter case). During the second phase, a final linear classifier is placed on top of the features extracted from deep network layers. Classifier and deep layers are fine-tuned in a supervised training fashion, by running an end-to-end SGD optimization procedure using only the few labeled samples at our disposal (with the corresponding labels).

5 Experimental setup

In the following, we describe the details of our experiments and comparisons, discussing the network architecture and the training procedure[3].

5.1 Datasets used for the experiments

The experiments were performed on the following datasets: CIFAR10, CIFAR100 [23] and TinyImageNet [46].

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 TinyImageNet 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 sample efficiency regimes in which the amount of labeled samples was respectively 1%, 2%, 3%, 4%, 5%, 10%, 25% and 100% of the whole training set.

5.2 Network architecture and training

We considered a six layer neural network as shown in Fig. 1: five deep layers plus a final linear classifier. The various layers were interleaved with other processing stages (such as ReLU nonlinearities, max-pooling, etc.). The architecture was inspired by AlexNet [24], but with slight modifications in order to reduce the overall computational cost of training. We decided to adopt a simple network

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

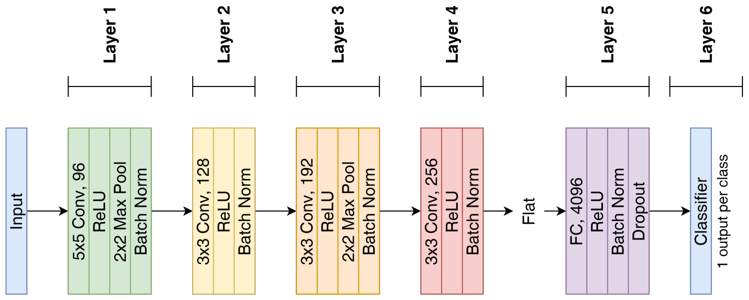

Fig. 1: The neural network used for the experiments.

model in order to be able to evaluate the effects of different learning approaches on a layer-by-layer basis. This choice also makes it more practical for other researchers to reproduce the experiments.

For each sample efficiency regime, we trained the network with our semi-supervised approach. First, we used the soft-WTA and the HPCA unsupervised pre-training in the internal layers. This was followed by the fine tuning stage with SGD training, involving the final classifier as well as the previous layers, in an end-to-end fashion.

For each sample efficiency configuration we also created a baseline for comparison. In this case, we used another popular unsupervised method, namely the Variational Auto-Encoder (VAE) [20], for the unsupervised pre-training stage. This was again followed by the supervised end-to-end fine tuning based on SGD. VAE-based semi-supervised learning was also the approach considered in [21].

5.3 Testing sample efficiency at different layer depths

In our experiments, in addition to evaluating 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 of the output of the various layers and we trained a new linear classifier on top of each already pre-trained layer . For each configuration, the supervised SGD training stage was performed using the labeled samples, thus fine tuning the classifier, as well as the previous network layers. Then, the resulting accuracy was evaluated. This process was done both for the Hebbian trained networks, and the VAE trained network, used as baseline, in order to make comparisons.

5.4 Details of training

We implemented our experiments using PyTorch. All the hyperparameters mentioned below resulted from a parameter search aimed at maximizing the validation accuracy on the respective datasets, following the Coordinate Descent (CD) approach [22].

Training was performed in 20 epochs using mini-batches of size 64. No more epochs were necessary, since the models had already reached convergence at that point. Networks were fed input images of size 32x32 pixels. Experiments were performed using five different seeds for the Random Number Generator (RNG), averaging the results and computing 95% confidence intervals.

In the Hebbian training, the learning rate was set to 10^{-3}. No L2 regularization or dropout was used, since the learning method did not present overfitting issues. For soft-WTA training, images were preprocessed by a whitening transformation as described in [23], although this step didn't have any significant effect for other training methods. The *temperature* parameter T of the softmax operation used in soft-WTA was set to $T = 0.02$.

For VAE training, 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. VAE training was performed without supervision, in an end-to-end encoding-decoding task, optimizing the β-VAE Variational Lower Bound [16], with coefficient $\beta = 0.5$

For the supervised training stage, based on SGD, 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 TinyImageNet. Cross-entropy loss was used as optimization metric.

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.

6 Results and discussion

In this section, the experimental results obtained with each dataset are presented and analyzed. We report the classification accuracy, along with the 95% confidence intervals, in the various sample efficiency regimes, for the CIFAR10, CIFAR100 and Tiny ImageNet datasets.

6.1 CIFAR10

Tab. 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.

As we can observe, Hebbian approaches perform better than VAE in almost all the cases. In particular, when low sample efficiency regimes are considered (between 1% and 5%) Hebbian approaches achieve significantly higher results than VAE. Only when the number of available labeled samples increases (beyond 10%), VAE pre-training starts to become competitive, obtaining results comparable to Hebbian training. Overall, Hebbian pre-training appears to be more effective than VAE, in particular when the number of available labeled samples is relatively low (5% or less). The maximum improvement of Hebbian approaches

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 VAE and Hebbian pre-training are compared.

Regime	**Pre-Train**	L1	L2	L3	L4	L5
	VAE	33.54 ±0.27	34.41 ±0.84	29.92 ±1.25	24.91 ±0.66	22.54 ±0.60
1%	soft-WTA	35.47 ±0.19	35.75 ±0.65	36.09 ±0.27	30.57 ±0.36	30.23 ±0.37
	HPCA	**37.01** ±0.42	**37.65** ±0.19	**41.88** ±0.53	**40.06** ±0.65	**39.75** ±0.50
	VAE	37.65 ±0.35	39.13 ±0.40	36.52 ±0.47	29.39 ±0.32	26.78 ±0.72
2%	soft-WTA	41.05 ±0.39	42.09 ±0.34	43.48 ±0.36	37.85 ±0.28	36.59 ±0.23
	HPCA	**41.60** ±0.28	**42.12** ±0.24	**46.56** ±0.38	**45.61** ±0.19	**45.51** ±0.43
	VAE	41.22 ±0.27	43.16 ±0.44	42.60 ±0.87	31.91 ±0.44	29.00 ±0.33
3%	soft-WTA	44.67 ±0.37	**46.12** ±0.27	48.08 ±0.42	43.22 ±0.31	41.54 ±0.50
	HPCA	**44.74** ±0.08	45.61 ±0.28	**49.75** ±0.41	**48.94** ±0.45	**48.80** ±0.27
	VAE	44.39 ±0.30	45.88 ±0.39	46.01 ±0.40	34.26 ±0.21	31.15 ±0.35
4%	soft-WTA	46.77 ±0.36	**49.24** ±0.40	51.23 ±0.37	46.90 ±0.27	45.31 ±0.18
	HPCA	**47.10** ±0.25	48.26 ±0.09	**52.00** ±0.16	**51.05** ±0.29	**51.28** ±0.28
	VAE	46.31 ±0.39	48.21 ±0.21	48.98 ±0.34	36.32 ±0.35	32.75 ±0.32
5%	soft-WTA	48.34 ±0.27	**52.90** ±0.28	**54.01** ±0.24	49.80 ±0.16	48.35 ±0.26
	HPCA	**48.49** ±0.44	50.14 ±0.46	53.33 ±0.52	**52.49** ±0.16	**52.20** ±0.37
	VAE	53.83 ±0.26	56.33 ±0.22	57.85 ±0.22	52.26 ±1.08	45.67 ±1.15
10%	soft-WTA	54.23 ±0.18	**59.40** ±0.20	**61.27** ±0.24	**58.33** ±0.35	**58.00** ±0.26
	HPCA	**54.36** ±0.32	56.08 ±0.28	58.46 ±0.15	56.54 ±0.23	57.35 ±0.18
	VAE	**62.51** ±0.24	67.26 ±0.32	68.48 ±0.21	68.79 ±0.29	68.70 ±0.15
25%	soft-WTA	61.29 ±0.23	**68.23** ±0.31	**70.09** ±0.41	**70.01** ±0.17	**69.85** ±0.37
	HPCA	61.45 ±0.26	65.25 ±0.16	64.71 ±0.17	62.43 ±0.13	64.77 ±0.22
	VAE	**67.53** ±0.22	75.83 ±0.31	80.78 ±0.28	84.27 ±0.35	85.23 ±0.26
100%	soft-WTA	67.37 ±0.16	**77.39** ±0.04	**81.83** ±0.47	**84.42** ±0.15	**85.37** ±0.03
	HPCA	66.76 ±0.13	75.16 ±0.20	79.90 ±0.18	83.55 ±0.33	84.38 ±0.22

over VAE is achieved in the 5% sample efficiency regime, in correspondence of network layer 5, where a gap of almost 16% points is observed between VAE and soft-WTA, and a gap of almost 20% points is observed between VAE and HPCA. Moreover, in low sample efficiency regimes (10 % or less) it is possible to notice that VAE and soft-WTA approaches suffer from a decrease in performance when going deeper with the number of layers. This issue is common with unsupervised methods, because the absence of a supervision signal (or still its scarcity, in case of semi-supervised training) makes it harder to develop task-specific features on higher layers, which is essential to achieve higher performances, as it emerges from previous studies on deep CNNs [1]. With HPCA, this problem seems to alleviate, and the accuracy remains pretty much constant with the number of layers, again in the low sample efficiency regimes (10% or less), meaning that the features produced by this approach are more meaningful for the classification

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 VAE and Hebbian pre-training are compared.

Regime	**Pre-Train**	L1	L2	L3	L4	L5
1%	VAE	21.69 ±0.10	21.70 ±0.30	17.61 ±0.54	13.45 ±0.54	12.28 ±0.50
	soft-WTA	19.60 ±0.23	20.08 ±0.32	18.50 ±0.44	15.21 ±0.36	15.30 ±0.28
	HPCA	**22.30** ±0.38	**22.28** ±0.63	**23.58** ±0.21	**21.70** ±0.61	**22.63** ±0.55
2%	VAE	28.24 ±0.13	**28.42** ±0.31	23.56 ±0.73	17.01 ±0.37	15.25 ±0.63
	soft-WTA	26.73 ±0.34	26.67 ±0.20	25.48 ±0.20	20.22 ±0.32	20.76 ±0.24
	HPCA	**29.65** ±0.52	26.57 ±0.26	**33.20** ±0.20	**30.21** ±0.54	**30.83** ±0.35
3%	VAE	31.28 ±0.54	31.71 ±0.27	27.46 ±1.23	18.26 ±0.24	16.44 ±0.12
	soft-WTA	30.53 ±0.37	30.81 ±0.52	29.99 ±0.44	23.22 ±0.25	23.69 ±0.49
	HPCA	**32.81** ±0.18	**33.08** ±0.55	**37.75** ±0.38	**35.02** ±0.36	**35.04** ±0.17
4%	VAE	34.60 ±0.10	35.44 ±0.31	32.34 ±0.79	19.68 ±0.32	17.89 ±0.27
	soft-WTA	33.51 ±0.26	34.15 ±0.21	32.85 ±0.18	25.78 ±0.21	26.91 ±0.24
	HPCA	**36.13** ±0.39	**36.23** ±0.20	**41.21** ±0.39	**39.16** ±0.32	**38.89** ±0.15
5%	VAE	36.68 ±0.17	37.26 ±0.26	35.33 ±0.81	20.55 ±0.44	18.48 ±0.26
	soft-WTA	35.71 ±0.29	36.83 ±0.37	35.80 ±0.18	28.39 ±0.43	29.57 ±0.13
	HPCA	**38.03** ±0.20	**38.02** ±0.25	**43.76** ±0.33	**41.66** ±0.20	**41.42** ±0.23
10%	VAE	42.64 ±0.34	44.84 ±0.48	46.04 ±0.44	27.81 ±0.13	23.80 ±0.60
	soft-WTA	41.91 ±0.27	**45.61** ±0.29	44.98 ±0.28	36.39 ±0.27	38.26 ±0.46
	HPCA	**43.51** ±0.34	44.84 ±0.26	**50.84** ±0.22	**49.53** ±0.19	**48.93** ±0.38
25%	VAE	**53.53** ±0.12	57.63 ±0.52	**62.16** ±0.57	55.29 ±0.68	52.59 ±1.02
	soft-WTA	50.60 ±0.34	**57.84** ±0.26	59.94 ±0.15	51.26 ±0.41	56.26 ±0.34
	HPCA	51.51 ±0.31	54.22 ±0.23	59.60 ±0.44	**58.29** ±0.29	**58.70** ±0.18
100%	VAE	**67.51** ±0.11	**73.83** ±0.30	**78.70** ±0.23	**79.58** ±0.18	**79.97** ±0.14
	soft-WTA	64.00 ±0.23	73.06 ±0.20	76.39 ±0.12	76.07 ±0.12	79.80 ±0.11
	HPCA	65.61 ±0.12	70.38 ±0.23	74.10 ±0.12	73.38 ±0.18	74.42 ±0.14

task. Furthermore, HPCA seems to perform generally better than soft-WTA, especially on higher layers, when low sample efficiency regimes are considered (5% or less). The maximum improvement of HPCA over soft-WTA is achieved in the 1% sample efficiency regime, in correspondence of network layers 5 and 4, where a gap of almost 9-10% points is observed.

6.2 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 Tab. 2 the top-5 accuracy results obtained on the CIFAR100 dataset are shown.

As we can observe, in low sample efficiency regimes (10% or less), Hebbian approaches perform better than VAE in almost all the cases. In particular, soft-

WTA generally performs better than VAE on higher network layers, and HPCA generally performs better than both soft-WTA and VAE on all network layers. Only when the number of available labeled samples increases (beyond 10%), VAE pre-training starts to really kick in, obtaining higher results than Hebbian training in almost all the cases. The maximum improvement of Hebbian approaches over VAE is achieved in the 10% sample efficiency regime, in correspondence of network layer 5, where a gap of almost 15% points is observed between VAE and soft-WTA, and a gap of over 25% points is observed between VAE and HPCA. Moreover, in low sample efficiency regimes (10 % or less) it is possible to notice that VAE and soft-WTA approaches suffer from a decrease in performance when going deeper with the number of layers. As already observed on the previous dataset, this is likely due to the lack of task-specificity of higher layer features provided by unsupervised training. With HPCA, this problem seems to alleviate, and the accuracy remains pretty much constant with the number of layers, again in the low sample efficiency regimes (10% or less), meaning that the features produced by this approach are more meaningful for the classification task. Furthermore, HPCA seems to perform generally better than soft-WTA, especially on higher layers (except for the 100% regime). The maximum improvement of HPCA over soft-WTA is achieved in the 4-5% sample efficiency regimes, in correspondence of network layers 5 and 4, where a gap of almost 12-13% points is observed. Overall, the results suggest that HPCA scales better than other approaches with the complexity of the dataset, especially for low sample efficiency regimes (10% or less), while VAE is generally preferable in regimes when more labeled samples are available (25% or higher).

6.3 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). In Tab. 3 the top-5 accuracy results obtained on the Tiny ImageNet dataset are shown.

As we can observe, in low sample efficiency regimes (10% or less), Hebbian approaches perform better than VAE. In particular, soft-WTA generally performs better than VAE on higher network layers, and HPCA performs better than both soft-WTA and VAE on all network layers. Only when the number of available labeled samples increases (beyond 10%), VAE pre-training starts to really kick in, obtaining higher results than Hebbian training. The maximum improvement of Hebbian approaches over VAE is achieved in the 10% sample efficiency regime, in correspondence of network layer 5, where a gap of over 3% points is observed between VAE and soft-WTA, and a gap of almost 15% points is observed between VAE and HPCA. Moreover, in low sample efficiency regimes (10 % or less) it is possible to notice that VAE and soft-WTA approaches suffer from a decrease in performance when going deeper with the number of layers. As already observed on previous datasets, this is likely due to the lack of task-specificity of higher layer features provided by unsupervised training. With HPCA, this

Table 3: TinyImageNet 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 VAE and Hebbian pre-training are compared.

Regime	**Pre-Train**	L1	L2	L3	L4	L5
1%	VAE	9.63 ±0.26	9.49 ±0.39	7.58 ±0.28	5.99 ±0.19	5.55 ±0.23
	soft-WTA	9.25 ±0.20	9.63 ±0.29	8.71 ±0.12	6.54 ±0.29	6.20 ±0.20
	HPCA	**10.81** ±0.27	**10.99** ±0.36	**12.15** ±0.46	**11.05** ±0.27	**11.38** ±0.41
2%	VAE	12.94 ±0.37	13.06 ±0.23	10.86 ±0.28	7.40 ±0.27	6.74 ±0.20
	soft-WTA	12.67 ±0.26	12.56 ±0.30	11.36 ±0.18	8.57 ±0.21	8.56 ±0.29
	HPCA	**14.12** ±0.23	**14.32** ±0.31	**16.89** ±0.61	**15.28** ±0.28	**15.71** ±0.47
3%	VAE	14.31 ±0.18	15.17 ±0.20	13.67 ±0.36	8.35 ±0.29	7.74 ±0.19
	soft-WTA	14.66 ±0.17	14.50 ±0.33	13.71 ±0.18	9.95 ±0.25	10.26 ±0.18
	HPCA	**16.25** ±0.21	**16.54** ±0.28	**19.78** ±0.47	**18.31** ±0.24	**18.23** ±0.33
4%	VAE	16.09 ±0.20	17.05 ±0.20	16.83 ±0.51	8.86 ±0.11	8.45 ±0.21
	soft-WTA	16.20 ±0.31	16.51 ±0.26	15.70 ±0.17	11.04 ±0.29	11.52 ±0.07
	HPCA	**17.70** ±0.44	**18.33** ±0.24	**21.95** ±0.57	**20.86** ±0.32	**20.55** ±0.28
5%	VAE	17.44 ±0.26	18.62 ±0.32	19.16 ±0.52	9.92 ±0.24	9.29 ±0.17
	soft-WTA	17.72 ±0.17	18.06 ±0.49	17.03 ±0.30	12.15 ±0.19	12.55 ±0.15
	HPCA	**19.26** ±0.41	**19.93** ±0.41	**23.97** ±0.52	**22.95** ±0.26	**22.46** ±0.17
10%	VAE	21.62 ±0.25	23.83 ±0.19	27.42 ±0.18	16.69 ±0.18	13.51 ±0.34
	soft-WTA	21.22 ±0.43	23.08 ±0.21	21.90 ±0.15	16.21 ±0.27	16.70 ±0.17
	HPCA	**22.82** ±0.33	**24.34** ±0.29	**28.69** ±0.36	**28.79** ±0.26	**28.13** ±0.38
25%	VAE	**29.40** ±0.31	**32.42** ±0.29	**39.93** ±0.31	**37.97** ±0.62	**37.89** ±0.54
	soft-WTA	26.36 ±0.48	31.31 ±0.28	32.54 ±0.13	22.39 ±0.11	24.96 ±0.23
	HPCA	28.01 ±0.75	30.63 ±0.16	35.87 ±0.53	36.98 ±0.26	37.10 ±0.23
100%	VAE	**42.32** ±0.16	**48.54** ±0.53	**58.31** ±0.12	**59.60** ±0.23	**60.23** ±0.65
	soft-WTA	38.55 ±0.20	46.82 ±0.33	48.91 ±0.24	42.35 ±0.24	54.94 ±0.10
	HPCA	40.34 ±0.31	45.00 ±0.40	53.12 ±0.26	52.95 ±0.28	53.96 ±0.43

problem seems to alleviate, and the accuracy remains pretty much constant or slightly increases with the number of layers, again in the low sample efficiency regimes (10% or less), meaning that the features produced by this approach are more meaningful for the classification task. Furthermore, HPCA seems to perform generally better than soft-WTA, especially on higher layers (except for the 100% regime). The maximum improvement of HPCA over soft-WTA is achieved in the 25% sample efficiency regime, in correspondence of network layers 5 and 4, where a gap of almost 13-14% points is observed. Overall, the results suggest that HPCA scales better than other approaches with the complexity of the dataset, especially for low sample efficiency regimes (10% or less), while VAE is preferable in regimes when more labeled samples are available (25% or higher).

7 Conclusions and future work

In summary, our results suggest that our semi-supervised approach based on unsupervised Hebbian pre-training performs generally better than VAE pre-training, especially in low sample efficiency regimes, in which only a small portion of the training set (between 1% and 10%) is assumed to be labeled. In particular, the HPCA approach appears to perform generally better than soft-WTA. Moreover, HPCA seems to scale better than other approaches when the complexity of the dataset increases, especially when low sample efficiency regimes are considered. On the other hand, VAE pre-training seems to become more effective in regimes where a larger portion of the training set (25% or higher) is labeled. 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 works, further improvements might come from exploring more complex feature extraction strategies, which can also be formulated as Hebbian learning variants, such as Independent Component Analysis (ICA) [17] and sparse coding [32, 33, 38]. Moreover, Hebbian approaches can also be combined with pseudo-labeling and consistency methods mentioned in Section 2 [6, 7, 18, 40]. In addition to the semi-supervised learning scenario considered in this paper, it would also be interesting to investigate Hebbian approaches in a meta-learning scenario. Hebbian learning already found application in the context of meta-learning, with the *differentiable plasticity* model [30]. In this case, the simple Hebbian learning rule, $\Delta \mathbf{w} = \eta \, y \, \mathbf{x}$, was used, but further improvements might come from applying more advanced Hebbian rules, such as those studied in this paper. Finally, an exploration on the behavior of such algorithms w.r.t. adversarial examples also deserves attention.

References

1. Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. arXiv preprint arXiv:1407.1610 (2014)
2. Amato, G., Carrara, F., Falchi, F., Gennaro, C., Lagani, G.: Hebbian learning meets deep convolutional neural networks. In: International Conference on Image Analysis and Processing. pp. 324–334. Springer (2019)
3. Bahroun, Y., Soltoggio, A.: Online representation learning with single and multi-layer hebbian networks for image classification. In: International Conference on Artificial Neural Networks. pp. 354–363. Springer (2017)
4. Becker, S., Plumbley, M.: Unsupervised neural network learning procedures for feature extraction and classification. Applied Intelligence **6**(3), 185–203 (1996)
5. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in neural information processing systems. pp. 153–160 (2007)
6. Berthelot, D., Carlini, N., Cubuk, E.D., Kurakin, A., Sohn, K., Zhang, H., Raffel, C.: Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. arXiv preprint arXiv:1911.09785 (2019)

7. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C.: Mixmatch: A holistic approach to semi-supervised learning. arXiv preprint arXiv:1905.02249 (2019)
8. Canto, F.J.A.: Convolutional neural networks with hebbian-based rules in online transfer learning. In: Mexican International Conference on Artificial Intelligence. pp. 35–49. Springer (2020)
9. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020)
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
11. Gao, B., Pavel, L.: On the properties of the softmax function with application in game theory and reinforcement learning. arXiv preprint arXiv:1704.00805 (2017)
12. Gerstner, W., Kistler, W.M.: Spiking neuron models: Single neurons, populations, plasticity. Cambridge university press (2002)
13. Grossberg, S.: Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. Biological cybernetics **23**(3), 121–134 (1976)
14. Haykin, S.: Neural networks and learning machines. Pearson, 3 edn. (2009)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
16. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework (2016)
17. Hyvarinen, A., Karhunen, J., Oja, E.: Independent component analysis. Studies in informatics and control **11**(2), 205–207 (2002)
18. Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Label propagation for deep semi-supervised learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5070–5079 (2019)
19. Karhunen, J., Joutsensalo, J.: Generalizations of principal component analysis, optimization problems, and neural networks. Neural Networks **8**(4), 549–562 (1995)
20. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
21. Kingma, D.P., Mohamed, S., Jimenez Rezende, D., Welling, M.: Semi-supervised learning with deep generative models. Advances in neural information processing systems **27**, 3581–3589 (2014)
22. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: New perspectives on some classical and modern methods. SIAM review **45**(3), 385–482 (2003)
23. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems (2012)
25. Krotov, D., Hopfield, J.J.: Unsupervised learning by competing hidden units. Proceedings of the National Academy of Sciences **116**(16), 7723–7731 (2019)
26. Lagani, G.: Hebbian learning algorithms for training convolutional neural networks. Master's thesis, School of Engineering, University of Pisa, Italy (2019), https://etd.adm.unipi.it/theses/available/etd-03292019-220853/

27. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. Journal of machine learning research **10**(1) (2009)
28. Magotra, A., kim, J.: 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 (2019)
29. Magotra, A., Kim, J.: Improvement of heterogeneous transfer learning efficiency by using hebbian learning principle. Applied Sciences **10**(16), 5631 (2020)
30. Miconi, T., Clune, J., Stanley, K.O.: Differentiable plasticity: training plastic neural networks with backpropagation. arXiv preprint arXiv:1804.02464 (2018)
31. Nowlan, S.J.: Maximum likelihood competitive learning. In: Advances in neural information processing systems. pp. 574–582 (1990)
32. Olshausen, B.A.: Learning linear, sparse, factorial codes. Massachusetts Institute of Technology, AIM-1580 (1996)
33. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature **381**(6583), 607 (1996)
34. O'Reilly, R.C., Munakata, Y.: Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain. MIT press (2000)
35. Pehlevan, C., Chklovskii, D.B.: 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 (2015)
36. Pehlevan, C., Hu, T., Chklovskii, D.B.: A hebbian/anti-hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data. Neural computation **27**(7), 1461–1495 (2015)
37. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., Raiko, T.: Semi-supervised learning with ladder networks. In: Advances in neural information processing systems. pp. 3546–3554 (2015)
38. Rozell, C.J., Johnson, D.H., Baraniuk, R.G., Olshausen, B.A.: Sparse coding via thresholding and local competition in neural circuits. Neural computation **20**(10), 2526–2563 (2008)
39. Sanger, T.D.: Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural networks **2**(6), 459–473 (1989)
40. Sellars, P., Aviles-Rivero, A.I., Schönlieb, C.B.: Laplacenet: A hybrid energy-neural model for deep semi-supervised classification. arXiv preprint arXiv:2106.04527 (2021)
41. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484 (2016)
42. Wadhwa, A., Madhow, U.: Bottom-up deep learning using the hebbian principle (2016)
43. Wadhwa, A., Madhow, U.: Learning sparse, distributed representations using the hebbian principle. arXiv preprint arXiv:1611.04228 (2016)
44. Weston, J., Chopra, S., Bordes, A.: Memory networks. arXiv preprint arXiv:1410.3916 (2014)
45. Weston, J., Ratle, F., Mobahi, H., Collobert, R.: Deep learning via semi-supervised embedding. In: Neural networks: Tricks of the trade, pp. 639–655. Springer (2012)
46. Wu, J., Zhang, Q., Xu, G.: Tiny imagenet challenge. Tech. rep., Stanford University (2017)
47. Zhang, Y., Lee, K., Lee, H.: Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In: International conference on machine learning. pp. 612–621 (2016)