Noname manuscript No.
(will be inserted by the editor)

Comparing the Performance of Hebbian against Backpropagation Learning using Convolutional Neural Networks

Gabriele Lagani* · Fabrizio Falchi · Claudio Gennaro · Giuseppe Amato

Received: date / Accepted: date

Abstract In this paper, we investigate Hebbian learning strategies applied to Convolutional Neural Network (CNN) training. We consider two unsupervised learning approaches, Hebbian Winner-Takes-All (HWTA) and Hebbian Principal Component Analysis (HPCA). The Hebbian learning rules are used to train the layers of a CNN in order to extract features that are then used for classification, without requiring backpropagation (backprop). Experimental comparisons are made with state-of-the-art unsupervised (but backprop-based) Variational Auto-Encoder (VAE) training. For completeness,we consider two supervised Hebbian learning variants (Supervised Hebbian Classifiers – SHC, and Contrastive Hebbian Learning – CHL), for training the final classification layer, which are compared to Stochastic Gradient Descent (SGD) training. We also investigate hybrid learning methodologies, where some network layers are trained following the Hebbian approach, and others are trained by backprop. We tested our approaches on MNIST, CIFAR10 and CIFAR100 datasets. Our

This work was partially supported by the H2020 project AI4EU under GA 825619 and by the H2020 project AI4Media under GA 951911.
Published version on Springer Neural Computing and Applications - doi: 10.1007/s00521-021-06701-4

G. Lagani
University of Pisa, Italy, 56124
E-mail: gabriele.lagani@phd.unipi.it
* Corresponding author

F. Falchi
ISTI-CNR Pisa, Italy, 56124
E-mail: fabrizio.falchi@cnr.it

C. Gennaro
ISTI-CNR Pisa, Italy, 56124
E-mail: claudio.gennaro@cnr.it

G. Amato
ISTI-CNR Pisa, Italy, 56124
E-mail: giuseppe.amato@cnr.it

results suggest that Hebbian learning is generally suitable for training early feature extraction layers, or to retrain higher network layers in fewer training epochs than backprop. Moreover, our experiments show that Hebbian learning outperforms VAE training, with HPCA performing generally better than HWTA.

Keywords Hebbian Learning · Deep Learning · Neural Networks · Biologically Inspired

1 Introduction

The error backpropagation algorithm (*backprop*) has been used with great success for training neural networks (e.g. [9,35]) on a variety of learning tasks. However, neuroscientists doubt that it is biologically plausible and that it models the real learning processes of the brain [27].

A possible biologically plausible learning mechanism could be based on the so-called *Hebbian* principle: "Neurons that fire together wire together". Starting from this simple principle, it is possible to formulate different variants of the Hebbian learning rule which are interesting also from the computer science point of view. For example, Hebbian learning with Winner-Takes-All (HWTA) competition [7] allows a group of neurons to learn to perform clustering on a set of data. Another interesting variant is Sanger's rule [33], which allows to perform Principal Component Analysis (PCA) on the data in an online fashion. In essence, Hebbian algorithms can be employed to extract features of interest from data and provide a biologically plausible, efficient and online solution for unsupervised learning tasks.

In the context of Convolutional Neural Networks (CNNs), the various network layers act as feature extractors, with lower layers extracting low-level features and next layers extracting progressively higher-level features. Therefore, Hebbian learning algorithms could represent a promising option for training such networks.

Previous works [37,36,2] already showed that Hebbian learning variants are suitable for training relatively shallow networks (with two or three layers), which are appealing for applications on constrained devices. For instance, in [1], preliminary results showed that HWTA competition was effective to retrain higher layers of a pre-trained network, achieving results comparable with backprop, but requiring fewer training epochs, thus suggesting potential applications in the context of transfer learning.

In this work, we take a step further and apply Hebbian learning on deeper network architectures. We perform a more detailed investigation of the HWTA learning rule, and we analyze the Hebbian Principal Component Analysis (HPCA) learning rule [33,13] to train deep CNNs.

We compared Hebbian algorithms, which are unsupervised, with another popular unsupervised (but backprop-based) approach, namely the Variational Auto-Encoder (VAE) [14]. We also deemed interesting to report the results obtained with supervised backprop training on an equivalent network, in order to give a more complete picture of the impact of different learning methodologies on the training process.

Specifically, a six layer *try-out* network was considered. The network was trained using the various learning approaches on the MNIST [20], CIFAR10, and CIFAR100 [17] datasets. We evaluated the quality of the features extracted from each layer by feeding these features to linear classifiers and evaluating the resulting accuracy. We decided to adopt a simplified network model because the focus of this work is not to evaluate the performance of a new complex network model, but rather to compare different learning approaches on an appropriate architecture. The six layer try-out network allows us to perform extensive experimentation, and to get insights on the effect of different learning paradigms on each network layer, evaluating the quality of the resulting feature extractors on a layer by layer basis.

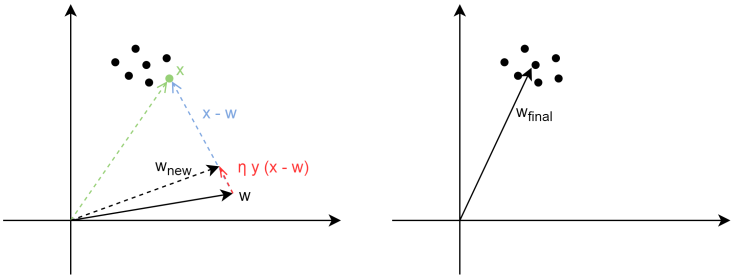
Furthermore, in order to assess the impact of switching from backprop to Hebbian training layer by layer, we also considered hybrid models in which some network layers are trained with backprop and others with Hebbian learning. Such hybrid models were also studied in [1], but only preliminary results where presented involving just the HWTA learning rule and just one dataset. In this work, we provide a more comprehensive evaluation of the HWTA rule, as well as the HPCA rule, using more datasets in our experiments.

Although Hebbian learning is an unsupervised approach, supervised variants were also proposed in literature. Some of these [30,34,19] are based on the concept of a *teacher neuron* coupled with a purely Hebbian learning rule. In the following, we will refer to classifiers trained with such approach as Supervised Hebbian Classifiers (SHCs). Other approaches [22,25] are based on the alternation between *Hebbian* and *anti-Hebbian* update phases, while also using a supervision signal. This kind of alternating strategy is called Contrastive Hebbian Learning (CHL). Another contribution of this paper is to provide an experimental evaluation of classifiers based on SHC and CHL on the various datasets.

Results in this paper confirm that Hebbian learning can be integrated with backprop, providing comparable accuracy when used to train lower or higher network layers, while requiring fewer training epochs. Moreover, they show that features learned by Hebbian training outperform VAE features in the classification task, with the HPCA variant performing generally better than HWTA.

The main contributions of this paper can be summarized as follows:

- Hebbian Winner-Takes-All (HWTA) and nonlinear Hebbian Principal Component Analysis (HPCA) learning rule variants, properly integrated with convolutional layers (Convolutional HWTA/HPCA), are applied to learn feature extractors in CNNs;
- The results on various datasets are compared with those obtained by unsupervised VAE, and the potentials and limitations of the methods are highlighted; we also deemed interesting to report the results of supervised backprop training in our discussion;



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

Fig. 1: Hebbian updates with weight decay.

- We also provide an experimental evaluation of hybrid neural network training (i.e. a scenario in which some network layers are trained with backprop and others with Hebbian approach) and supervised Hebbian learning variants on various datasets.

The remainder of this paper is structured as follows: Section 2 provides a background on the related literature; Section 3 describes our scenario of investigation, including how Hebbian learning is integrated with convolutional layers, hybrid network models, SHC and CHL classifiers; Section 4 delves into the details of our experimental setup; In Section 5, the results of our simulations are illustrated; Finally, Section 6 presents our conclusions and outlines possible future developments.

2 Background and related work

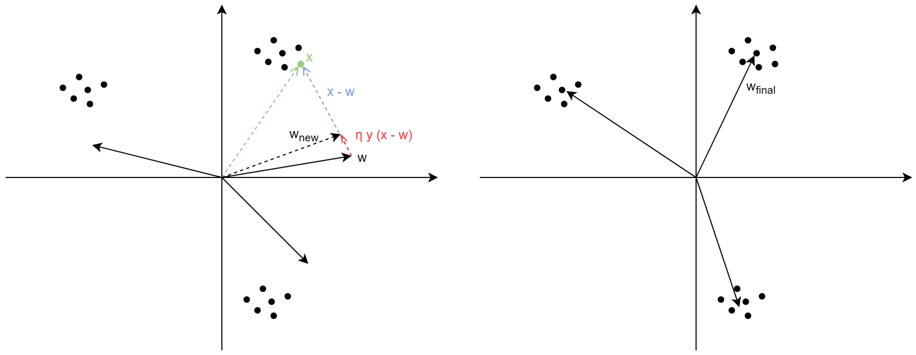
Consider a single neuron with weight vector $\mathbf{w}$ and input $\mathbf{x}$. Call $y = \mathbf{w}^T \mathbf{x}$ the neuron output. The Hebbian learning rule, in its most basic form, can be expressed mathematically as [8]:

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

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 latter term is computed, according to Hebbian learning, as follows:

$$\Delta\mathbf{w} = \eta\, y\, \mathbf{x} \tag{2}$$

where η is the learning rate. 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.



(a) Update step

(b) Final position after convergence

Fig. 2: Hebbian updates with Winner-Takes All competition.

2.1 Hebbian WTA

To prevent weights from growing unbounded, a weight decay term is generally added. In the context of competitive learning [7,32,15], this is obtained as follows:

$$\Delta\mathbf{w} = \eta\, y\, \mathbf{x} - \eta\, y\, \mathbf{w} = \eta\, y\, (\mathbf{x} - \mathbf{w}) \tag{3}$$

This rule has an intuitive interpretation: when an input vector is presented to the neuron, its vector of weights is updated in order to move it closer to the input, so that the neuron will respond more strongly when a similar input is presented. When several similar inputs are presented to the neuron, the weight vector converges to the center of the cluster formed by these inputs (Fig. 1).

When multiple neurons are involved in a complex network, the Winner-Takes-All (WTA) [7,32] strategy can be adopted to force different neurons to learn different patterns, corresponding to different clusters of inputs. When an input is presented to a WTA layer, the neuron whose weight vector is closest to the current input is elected as winner. Only the winner is allowed to perform a weight update, thus moving its weight vector closer to the current input (Fig. 2). If a similar input will be presented again in the future, the same neuron will be more likely to win again. This strategy allows a group of neurons to perform clustering on a set of data points (Fig. 2).

In recent works [37,36], WTA and the variant k-WTA (in which the k neurons with highest activations are elected as winners) were applied in the context of computer vision to train a three layer CNN to extract features from images, in order to perform classification. Similar paradigms were also studied in the context of Spiking Neural Networks (SNNs) [5,4]. These works showed that the approach is suitable to train relatively shallow networks (e.g. with two or three layers), achieving accuracy around 65-70% on CIFAR-10 and from 95% up to 98-99% on MNIST, which is comparable to backpropagation-based approaches on networks of the same depth.

In [1,19], the authors provided preliminary experiments on a single dataset (CIFAR10), by applying Hebbian-WTA learning to CNNs with up to six layers, comparing the results with those obtained by training the same network with backprop. The WTA approach, as it is, is unsupervised, but a supervised Hebbian learning variant was also proposed in order to train the final classification layer. The results confirmed that the approach was effective for training shallow networks. It was also found that the approach was effective for re-training the higher layers (including the final classifier) of a pre-trained network. In addition, the algorithm required much fewer epochs than backprop to converge.

The novel contributions of this work w.r.t. the previous one are that more extensive experimentation is performed using multiple datasets (MNIST, CIFAR10, CIFAR100), and a novel learning rule is also explored, in addition to Hebbian WTA. This is the Hebbian PCA learning rule, which is explained in the next sub-section. Moreover, we added experiments with VAE, for comparison with state-of-the-art backprop-based unsupervised learning. Finally, we performed experiments involving the supervised CHL and SHC methods, making comparisons between the two approaches and SGD training.

2.2 Hebbian PCA

According to the definition given above, WTA enforces a kind of *quantized* information encoding in layers of neural network. Only one neuron activates to encode the presence of a given pattern in the input. On the other hand, neural networks trained with backpropagation exhibit a *distributed* representation, where multiple neurons activate combinatorially to encode different properties of the input, resulting in an improved coding power. The importance of distributed representations was also highlighted in [6,24].

A more distributed coding scheme could be obtained by having neurons extract principal components from data, which can be achieved with Hebbian-type learning rules [33,3]. In order to perform Hebbian PCA, a set of weight vectors has to be determined, for the various neurons, that minimize the *representation error*, defined as:

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

where the subscript i refers to the i^{th} neuron in a given layer and $E[\cdot]$ is the mean value operator. It can be pointed out that, in the case of linear neurons and zero centered data, this reduces to the classical PCA objective of maximizing the output variance, with the weight vectors subject to orthonormality constraints [33,3,13]. From now on, we assume that the input data are centered around zero. If this is not true, we just need to subtract the average $E[x]$ from the inputs beforehand.

It can be shown that the following learning rule minimizes the objective in Eq. 4 [33]:

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

In case of nonlinear neurons, a solution to the problem can still be found [13]. Calling $f()$ the neuron activation function, the representation error

$$L(w_i) = E[(\mathbf{x} - \sum_{j=1}^{i} f(y_j)\, \mathbf{w_j})^2] \tag{6}$$

can be minimized with the following nonlinear version of the Hebbian PCA rule:

$$\varDelta \mathbf{w_i} = \eta f(y_i)(\mathbf{x} - \sum_{j=1}^{i} f(y_j) \mathbf{w_j}) \tag{7}$$

Several variants of the Hebbian PCA approach were explored in literature for the linear case [33,3,29,28], and applied in the context of computer vision [2], but only for relatively shallow networks. In our experiments, we applied the nonlinear version of the Hebbian PCA rule also on deeper networks, as explained in the following sections.

2.3 Supervised Hebbian learning

While the Hebbian approaches discussed so far are unsupervised, Hebbian learning can also be adapted to the supervised setting. We consider two approaches for doing so, the Supervised Hebbian Classifier (SHC) [19] and the Contrastive Hebbian Learning (CHL) [22] classifier.

The idea behind the SHC approache is based on the concept of a *teacher neuron* [30,34,37], which ideally provides the target signal to a trainable neuron. The teacher's signal replaces the actual output of the neuron so that, when the Hebbian principle is applied, it reinforces the correlation between the input and the teacher-provided output. In this way, when a similar input is presented again, the neuron tends to produce a similar response. The SHC is realized by applying this principle in combination with the learning rule in Eq. 3. More specifically, calling t the teacher signal, the learning rule becomes:

$$\varDelta \mathbf{w} = \eta\, t\, (\mathbf{x} - \mathbf{w}) \tag{8}$$

The teacher signal t should be 1 if the input's class correspond to that associated with the neuron, and 0 otherwise. The effect of this rule is that the neuron's weight vector will converge towards the centroid of the cluster formed by only those inputs associated with the target class that the neuron is supposed to detect.

In CHL, the network alternates between two processing stages, a *free phase* and a *clamped phase*. During the free phase, ordinary processing occurs. Let us call denote the input and output of a neuron after the free phase as $\mathbf{x}^-$ and y^-, respectively. An *anti-Hebbian* update is computed after the free phase, according to the formula:

$$\varDelta \mathbf{w}^- = -\eta \, y^- \, \mathbf{x}^- \tag{9}$$

During the clamped phase, the neuron outputs are clamped to a desired value. Call $\mathbf{x}^+$ and y^+ the input and output of a neuron after the clamped phase. At this point, a regular Hebbian update is performed:

$$\varDelta \mathbf{w}^+ = \eta \, y^+ \, \mathbf{x}^+ \tag{10}$$

This approach was shown to be able to approximate backprop training under mild conditions [38], but in a biologically plausible and Hebbian fashion. CHL can be applied for training a linear classification layer by replacing the classifier's output y^+ with the teacher signal t during the clamped phase (while the inputs $\mathbf{x}^+ = \mathbf{x}^- = \mathbf{x}$ are the same for the two phases), thus leading to the total update:

$$\begin{aligned} \varDelta \mathbf{w} &= \varDelta \mathbf{w}^+ + \varDelta \mathbf{w}^- \\ &= \eta \, (y^+ \, \mathbf{x}^+ - y^- \, \mathbf{x}^-) \\ &= \eta \, \mathbf{x}(t - y^-) \end{aligned} \tag{11}$$

Note that this update is equivalent to a gradient descent update of a linear classifier on a Mean Squared Error (MSE) loss [8,22,25].

3 Hebbian learning on deep CNNs

In the following, we describe our approach to use Hebbian learning with deep CNNs. We introduce the strategy used for integrating Hebbian learning methods with convolutional layers, and the technique used extend the Hebbian learning approach to a supervised setting. In addition we introduce the *try-out* neural network architecture used to evaluate our approach, and the hybrid (Hebbian-backprop) learning modality.

3.1 Convolutional HWTA/HPCA

In order to be able to use the Hebbian rules with CNNs, we had to define a proper way to integrate these rules with convolutional layers. In particular, neurons at different horizontal and vertical offset of the convolutional layer are constrained to have shared weights.

Previous works [36,2] handled convolutions with Hebbian learning by extracting random patches from the images, or by processing patches sequentially, one at a time, and feeding each patch to a single column of convolutional filters. This approach is poorly parallelizable, and does not exploit all the information contained in the image.

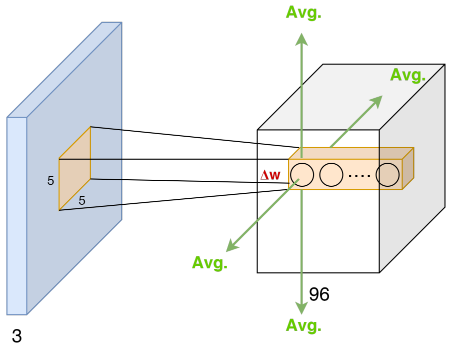


Fig. 3: Update averaging over horizontal and vertical dimensions.

In order to meet the convolutional constraints, we considered a different approach, in which the learning rule was adapted as follows: each set of neurons looking at the same portion of the image computed their updates by applying the desired rule, the input x being the patch extracted from the image at the specific horizontal and vertical position. We then averaged the updates over the horizontal and vertical dimensions (Fig. 3). The resulting update was applied to the kernel shared by all the neurons at different horizontal and vertical locations. When mini-batches of inputs were used during training, the update averaging was performed also over the mini-batch dimension.

3.2 SHC and CHL classifiers

In order to evaluate Hebbian learning also in teh supervised setting, we implemented SHC and CHL classifiers. These classifiers are trained on top of the features extracted from pre-trained networks, freezing the already trained network layers.

SHCs are trained using the learning rule in Eq. 8. The teacher signal was set to the target output that the neuron was required to produce for a given input. Similarly, CHL classifiers are trained according to Eq. 11, where the free phase output is the ordinary output provided by the classifier, and the clamped phase output was set to the target value.

3.3 Network architecture and evaluation

The focus of this work is not to evaluate the performance of complex network architecture. Rather we aim at evaluating and comparing the effects of Hebbian learning approaches, supervised backprop, and VAE under various settings.

Accordingly, we defined a *try-out* model, where it is possible to perform a large number of experiments and get insights about the effect of the learning approach on various network layers, by evaluating the quality of the features

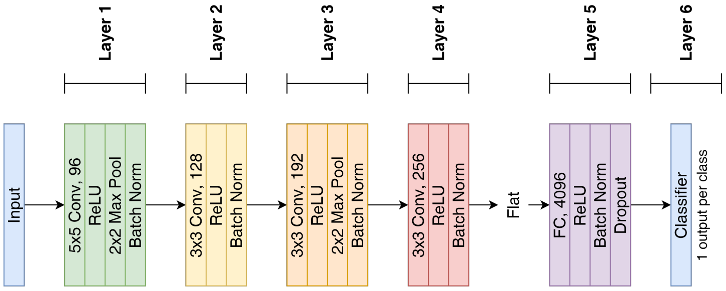


Fig. 4: The try-out neural network used for the experiments (image from [1]).

extracted from the network on a layer by layer basis. This architecture makes also the experiments more practical to be reproduced by other researchers. The following subsections illustrate the try-out network architecture and the evaluation procedure.

3.3.1 Try-out neural network architecture

The deep neural network used in this work consists of six layers: five layers plus a final linear classifier. The various layers are interleaved with other processing stages (such as ReLU nonlinearities, max pooling, etc.), as shown in Fig. 4. The architecture is inspired to the AlexNet [18], where one of the fully connected layers was removed and, in general, the number of neurons was slightly modified, to allow a finer grained analysis of the various learning approaches. In our experiments we compared both HWTA and HPCA learning approaches, with supervised backprop and VAE. Below, we also discuss more details of the VAE and supervised backprop training.

3.3.2 Variational Auto-Encoder for unsupervised learning

We compared the unsupervised Hebbian approaches with another popular unsupervised method, namely the Variational Auto-Encoder (VAE) [14]. We considered the VAE architecture shown in Fig. 5: the try-out network model in Fig. 4, up to layer 5, acted as encoder, with a fully connected layer mapping the output feature map to a 256 gaussian latent variable representation, while a specular network branch acted as decoder.

3.3.3 Backprop training for supervised learning

The first part of our experiments is mainly focused on comparing unsupervised learning approaches, i.e. Hebbian learning and VAE. Nonetheless, we also deemed interesting to include the results provided by supervised backprop learning in our discussion. For this purpose, we also report the results obtained by training a network with the same architecture as the try-out model

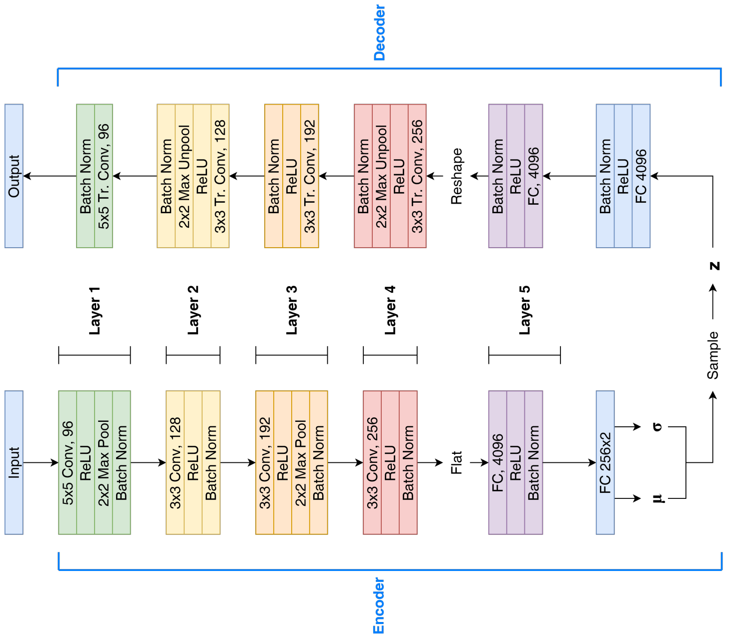


Fig. 5: The encoder-decoder architecture for the Variational Auto-Encoder (VAE) experiments.

shown in Fig. 4, by using supervised end-to-end Stochastic Gradient Descent (SGD) training on a cross-entropy loss metric.

3.3.4 Evaluating internal network layers

As we will also discuss in Section 5, we aim at evaluating how the Hebbian approach affects the capability of learning feature extractors in the various layers of the try-out neural network, on a layer by layer basis. In order to evaluate the quality of the features extracted from the various layers of the trained models, we cut the try-out network, in correspondence of the various layers, and we placed a linear classifier on top of each already trained layer (for example, Fig. 6 shows a classifier on top of the first network layer). Then, we evaluated the accuracy achieved by classifying the corresponding features. This was done for the Hebbian-trained networks and for the VAE network, in order to compare the results, and also for the supervised backprop-trained network, as we also deemed interesting to include these results in our discussion.

3.3.5 Hybrid network models

We also implemented hybrid network learning, i.e. scenarios in which some network layers were trained with backprop and others were trained with Hebbian approach (Fig. 7), in order to asses the impact on accuracy when replacing

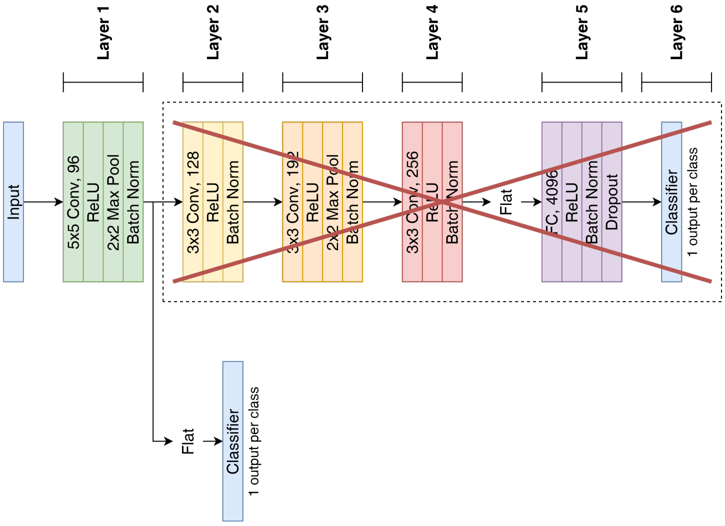


Fig. 6: Classifier placed on top of the first layer of the network.

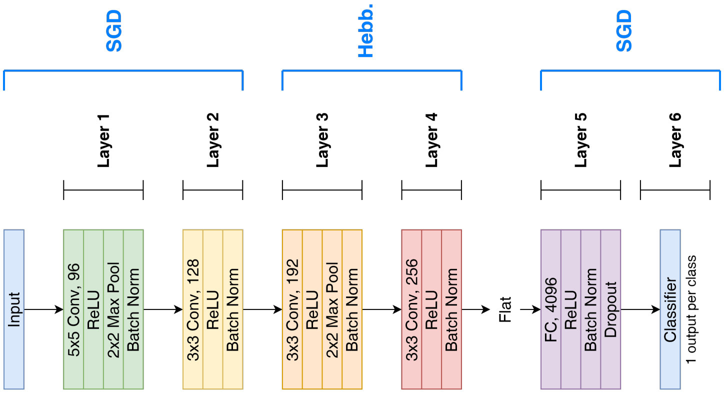


Fig. 7: An example of hybrid network model.

backprop layers with Hebbian equivalent. The models were constructed by replacing the upper layers of a pre-trained network with new ones, and training from scratch using different learning algorithms. Meanwhile, the lower layers remained frozen, in order to avoid adaptation to the new upper layers. Various configurations of layers were considered.

4 Details of training

We implemented our experiments using PyTorch. [1]

All the hyperparameters discussed below, resulted from a parameter search, based on Coordinate Descent (CD) [16], to maximize the validation accuracy in the respective scenarios. CD works as follows: starting from an initially selected point in hyperparameter space, one coordinate (i.e. hyperparameter) at a time is perturbed, and the resulting hyperparameter configuration is evaluated. Hyperparameters are updated in the direction of the perturbation that leads to an improvement in the result. The steps are the following: 1) get hyperparameter set according to CD based on previous validation results; 2) train the model with the given hyperparameters and record the resulting validation accuracy; 3) repeat from point 1 until no further improvement is obtained.

Concerning the datasets that we used, the MNIST dataset contains 60,000 training samples and 10,000 test samples, divided in 10 classes representing handwritten digits from 0 to 9. In our experiments, we further divided the training samples into 50,000 samples that were actually used for training, and 10,000 for validation. The CIFAR10 and CIFAR100 datasets contain 50,000 training samples and 10,000 test samples, divided in 10 and 100 classes, respectively, representing natural images. In our experiments, we further divided the training samples into 40,000 samples that were actually used for training, and 10,000 for validation. In order 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.

4.1 Training the try-out network

We used the try-out network architecture shown in Fig. 4. The model was fed with RGB images of size 32x32 pixels as inputs. The network was trained using Stochastic Gradient Descent (SGD) with error backpropagation and cross-entropy loss, with the HPCA rule in Eq. 7 (in which the nonlinearity was set to the ReLU function), and with the HWTA rule. During Hebbian training, the final classifier was trained using the SHC approach, according to Eq. 8.

Training was performed in 20 epochs (although, for the Hebbian approach, convergence was typically achieved in much fewer epochs) using mini-batches of size 64.

For SGD training, 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, and Nesterov correction [10].

Contrarily to standard momentum (which first corrects the accumulated momentum with the current gradient estimate and then updates the weight

[1] The code to reproduce the experiments is available at:
`github.com/GabrieleLagani/HebbianPCA/tree/hebbpca`.

in the resulting direction), Nesterov method first updates the weights in the momentum direction, and then applies a correction to the accumulated momentum given by the gradient estimate at the new location. This look-ahead strategy helps correcting optimization trajectories and improves convergence.

Dropout rate was set to 0.5. L2 penalty was also used to improve regularization. We recall that this is a regularization term in the form $\lambda \, |\mathbf{w}|^2$ that is added to the loss function, in order to penalize large weights. Here, λ is the weight decay coefficient, which was set to $5 \cdot 10^{-2}$ for MNIST and CIFAR10, and to 10^{-2} for CIFAR100.

In the HPCA and HWTA 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. In case of HWTA training, images were preprocessed by a whitening transformation as described in [17], although this step didn't have any significant effect for the other training methods.

4.2 VAE training

VAE training of the network in Fig. 5 was performed in the same fashion as for the try-out network training but, obviously, in an unsupervised image encoding-decoding task. Specifically, the model was trained using the β-VAE [11] Variational Lower-Bound unsupervised criterion, with coefficient $\beta = 0.5$. No L2 penalty nor dropout was used in this case. Note that the decoder part was removed at test time and the features extracted from encoder layers were used for classification.

4.3 Training of classifiers on top of internal layers

The SGD linear classifiers placed on top of the various network layers, as shown in Fig. 6, were trained with supervision, in the same way as we described above for training the whole try-out network. Learning rate was set to 10^{-3} and the L2 penalty term was reduced to $5 \cdot 10^{-4}$.

CHL classifiers were also trained as above, using the desired target as teacher signal, with learning rate set to 10^{-3} and L2 penalty $5 \cdot 10^{-4}$.

The SHC linear classifiers placed on top of the various network layers were trained with learning rate set to 10^{-3}, but no learning rate scheduling nor L2 regularization was needed in this case.

4.4 Hybrid network training

Hybrid network models were trained using various combinations of Hebbian and backprop layers, as in Fig. 7. Training was performed in a bottom-up approach, i.e. we first started by training the base try-out network with backprop, then we split the network at a desired point, removing all the layers on top, and replacing them with new Hebbian layers. The new Hebbian layers were trained using HWTA or HPCA, as described above, while the bottom layers remained frozen. This process produces a network whose bottom layers are trained with backprop, and top layers are trained with Hebbian. Again, a new splitting point can be chosen among the Hebbian layers, in order to remove all the Hebbian layers on top of the desired point, replacing them with backprop layers. Retraining the new layers with SGD, while the bottom layers are kept frozen, produces a network alternating backprop-Hebbian-backprop layers, as in Fig. 7. SGD training for the first or the last part of the hybrid networks (i.e. bottom layers or top layers) was performed as described above, but using L2 penalty $5 \cdot 10^{-4}$ for the top layers, when the last splitting point was right before the ultimate or penultimate layer (hence, for retraining the last or the last two layers), and $5 \cdot 10^{-}2$ in all the other cases.

5 Results

In the following subsections, we present the experimental results on MNIST, CIFAR10, and CIFAR100 datasets. For each of these datasets, we present Tables 1, 3, 5, showing the accuracy obtained by a linear classifier trained on top of the features extracted from each network layer, in order to asses the quality of the respective features in the classification task. We compare the results of unsupervised HPCA, HWTA and VAE training. Even though we mainly focus on comparing unsupervised methods, we also deemed interesting to report the results of supervised backprop (BP) training in our discussion. We also report, in Tables 2, 4, 6, the results obtained when retraining higher layers of a network pre-trained with backprop, together with the required number of epochs to convergence, in order to assess the potential of Hebbian approaches to tasks that involve retraining of higher network layers. In these cases, the final classification layer was trained by SHC, because, as we observed from other experiments (see Appendix A), this method performed better than CHL on higher network layers, in terms of trade-off between accuracy and training epochs.

Supplementary results, included in Appendix A, show the results of hybrid training, and the comparison between SHC, CHL, and SGD classifiers.

We performed five independent iterations of each experiment, using different seeds, averaging the results and computing 95% confidence intervals.

5.1 MNIST

In this sub-section we analyze the behavior of Hebbian learning approaches in a simple scenario of digit recognition on the MNIST dataset.

5.1.1 Classifiers on top of internal layers

In Tab. 1, we report the MNIST test accuracy obtained by classifiers placed on top of the various layers of the try-out network. We report the results obtained on the network trained with, respectively, supervised backprop (BP), VAE, HPCA, and HWTA.

Unsupervised approaches typically suffer from a decrease in performance when going deeper with the number of layers. The reason is that they are not able to exploit a supervision signal that enables the formation of task-specific features that are essential to boost the performance on higher layers. This can be observed both for HWTA and VAE training. With the HPCA approach, the problem seems to alleviate, and the accuracy remains pretty much constant when we move to deeper layers. In particular, the HPCA approach exhibits an increase of almost 2% points w.r.t. HWTA on the features extracted from the fourth convolutional layer. The Hebbian features appear to behave comparably or better than VAE features, especially on higher layers, with an improvement up to 8% points on the fifth layer. Moreover, we can observe that both Hebbian approaches reach higher performance w.r.t. backprop for the features extracted from the first two layers, suggesting possible applications of Hebbian learning for training relatively shallow networks.

5.1.2 Re-training higher network layers

Tab. 2 aims to show that it is possible to replace the last two network layers (including the final classifier) with new ones, and re-train them with Hebbian approach (in this case, the supervised Hebbian algorithm is used to train the final classifier), achieving accuracy comparable to backprop, but requiring fewer training epochs (1 vs 15, respectively). This suggests potential applications in the context of transfer learning [39].

5.2 CIFAR10

In the previous sub-section, we considered a relatively simple image recognition task involving digits. In this section, we aim at analysing Hebbian learning approaches in a slightly more complex task involving natural image recognition on the CIFAR10 dataset.

5.2.1 Classifiers on top of internal layers

In Tab. 3, we report the CIFAR10 test accuracy obtained by classifiers placed on top of the various layers of the network. We report the results obtained on the try-out network trained with, respectively, supervised backprop (BP), VAE, HPCA, and HWTA.

Also in this case, the HWTA and VAE approaches suffer from a decrease in performance when going deeper with the number of layers. With the HPCA approach, this problem seems to alleviate, and the accuracy remains pretty much constant when we move to deeper layers. In particular, the HPCA approach exhibits an increase of almost 5% points w.r.t. HWTA on the features extracted from the fifth layer. Still, further research is needed in order to close the gap

Table 1: MNIST accuracy (top-1) and 95% confidence intervals on features extracted from convolutional network layers. Underline represents best overall result. Bold represents best result among unsupervised methods. The Hebbian approaches appear to be comparable or superior to VAE, especially when higher layer features are considered. Moreover, HPCA improves over HWTA on higher layer features. It is also possible to observe that Hebbian training achieves higher results than supervised backprop when lower layer features are concerned.

Layer	BP Acc.(%)	VAE Acc. (%)	HPCA Acc.(%)	HWTA Acc.(%)
1	95.80 ±0.02	**98.67** ±0.03	98.23 ±0.09	98.16 ±0.05
2	97.26 ±0.01	**98.90** ±0.03	98.47 ±0.09	98.52 ±0.06
3	98.77 ±0.01	98.30 ±0.02	98.47 ±0.09	**98.55** ±0.02
4	99.56 ±0.01	94.68 ±0.04	**98.48** ±0.08	96.58 ±0.04
5	99.59 ±0.02	90.32 ±0.06	**98.53** ±0.08	97.15 ±0.01

Table 2: MNIST accuracy (top-1), 95% confidence intervals, and convergence epochs obtained by retraining higher layers of a pre-trained network. Supervised backprop (BP), the HPCA approach, and the HWTA approach are compared. It can be observed that Hebbian learning achieves comparable results to BP, but in fewer training epochs.

L1	L2	L3	L4	L5	L6	Method	Acc.(%)	Num. Epochs
B	B	B	B	B	G	BP	99.59 ±0.02	15
B	B	B	B	B	H	SHC	**99.62** ±0.01	**1**
B	B	B	B	H	H	HPCA + SHC	99.55 ±0.03	1
						HWTA + SHC	99.55 ±0.02	1

Table 3: CIFAR10 accuracy (top-1) and 95% confidence intervals on features extracted from convolutional network layers. Underline represents best overall result. Bold represents best result among unsupervised methods. The Hebbian approaches appear to perform better than VAE, especially when higher layer features are considered. Moreover HPCA improves over HWTA on higher layer features. It is also possible to observe that Hebbian training achieves comparable results with backprop when lower layer features are concerned.

Layer	BP Acc.(%)	VAE Acc. (%)	HPCA Acc.(%)	HWTA Acc.(%)
1	61.59 ±0.08	60.71 ±0.16	64.69 ±0.29	**64.79** ±0.34
2	67.67 ±0.11	56.32 ±0.31	**65.92** ±0.14	64.35 ±0.35
3	73.87 ±0.15	41.31 ±0.16	**64.43** ±0.21	59.69 ±0.16
4	83.88 ±0.04	29.58 ±0.07	**61.24** ±0.21	48.56 ±0.17
5	84.95 ±0.25	26.95 ±0.12	**61.12** ±0.33	46.88 ±0.23

Table 4: CIFAR10 accuracy (top-1), 95% confidence intervals, and convergence epochs obtained by retraining higher layers of a pre-trained network. Supervised backprop (BP), the HPCA approach, and the HWTA approach are compared. It can be observed that Hebbian learning achieves competitive results w.r.t. BP, but in fewer training epochs.

L1	**L2**	**L3**	**L4**	**L5**	**L6**	**Method**	**Acc.(%)**	**Num. Epochs**
B	B	B	B	B	G	BP	**84.95** ±0.25	12
B	B	B	B	B	H	SHC	84.59 ±0.01	**1**
B	B	B	B	H	H	HPCA + SHC	81.48 ±0.16	**1**
						HWTA + SHC	82.48 ±0.14	**1**

with backprop also when more layers are added, as it would be desirable to make the Hebbian approach suitable as a biologically plausible alternative to backprop for training deep networks. The Hebbian features appear to behave better than VAE features, especially on higher layers, with an improvement up to 24% points on the fifth layer. Moreover, we can observe that both Hebbian approaches reach higher or comparable performance w.r.t. backprop for the features extracted from the first two layers, suggesting possible applications of Hebbian learning for training relatively shallow networks.

5.2.2 Re-training higher network layers

Tab. 4 aims to show that it is possible to replace the last two network layers (including the final classifier) with new ones, and re-train them with Hebbian approach (in this case, the supervised Hebbian algorithm is used to train the final classifier), achieving accuracy comparable to backprop (with a peak performance drop of just 2-3% points when the last two layers are replaced), but requiring fewer training epochs (1 vs 12, respectively). This suggests potential applications in the context of transfer learning [39].

5.3 CIFAR100

In this sub-section, we want to further analyse the scalability of Hebbian learning to a more complex task of natural image recognition involving more classes, namely CIFAR100. In this case, we evaluated the top-5 accuracy, given that CIFAR100 contains a much larger number of classes than the previous datasets.

5.3.1 Classifiers on top of internal layers

In Tab. 5, we report the CIFAR100 top-5 test accuracy obtained by classifiers placed on top of the various layers of the try-out network. We report the results obtained on the network trained with, respectively, supervised backprop (BP), VAE, HPCA, and HWTA.

Table 5: CIFAR100 accuracy (top-5) and 95% confidence intervals on features extracted from convolutional network layers. Underline represents best overall result. Bold represents best result among unsupervised methods. The Hebbian approaches appear to perform better than VAE, especially when higher layer features are considered. Moreover HPCA improves over HWTA on higher layer features. It is also possible to observe that Hebbian training achieves competitive results w.r.t. backprop when lower layer features are concerned.

Layer	BP Acc.(%)	VAE Acc. (%)	HPCA Acc.(%)	HWTA Acc.(%)
1	51.67 ±0.10	58.46 ±0.12	**60.94** ±0.09	59.56 ±0.13
2	60.84 ±0.19	54.63 ±0.20	**62.24** ±0.15	58.49 ±0.20
3	67.01 ±0.13	39.46 ±0.15	**64.17** ±0.22	52.97 ±0.22
4	78.85 ±0.10	26.42 ±0.21	**61.27** ±0.24	37.38 ±0.12
5	80.74 ±0.05	23.03 ±0.12	**59.51** ±0.20	37.87 ±0.21

Table 6: CIFAR100 accuracy (top-5), 95% confidence intervals, and convergence epochs obtained by retraining higher layers of a pre-trained network. The network fully trained with backprop (BP), the HPCA approach, and the HWTA approach are compared. It can be observed that HPCA performs better than HWTA, and achieves competitive results w.r.t. BP, but in fewer training epochs.

L1	L2	L3	L4	L5	L6	Method	Acc.(%)	Num. Epochs
B	B	B	B	B	G	BP	**80.74** ±0.05	7
B	B	B	B	B	H	SHC	79.45 ±0.02	**1**
B	B	B	B	H	H	HPCA + SHC	77.66 ±0.09	**1**
						HWTA + SHC	63.62 ±0.27	**1**

Again, VAE and HWTA approaches suffer from a decrease in performance when going deeper with the number of layers. With the HPCA approach, this problem seems to alleviate, and the accuracy remains pretty much constant when we move to deeper layers. In particular, the HPCA approach exhibits an increase of almost 24% points w.r.t. HWTA on the features extracted from the fourth convolutional layer. The Hebbian features appear to behave comparably or better than VAE features, especially on higher layers, with an improvement of up to 36% points on the fifth layer. Moreover, we can observe that both Hebbian approaches reach competitive performance w.r.t. backprop for the features extracted from the first three layers, with HPCA in particular improving by 9% points over BP on the first layer, suggesting possible applications of Hebbian learning for training relatively shallow networks.

5.3.2 Re-training higher network layers

Tab. 6 aims to show that it is possible to replace the last two network layers (including the final classifier) with new ones, and re-train them with Hebbian approach (in this case, the supervised Hebbian algorithm is used to train the final classifier), achieving accuracy comparable to backprop (with just a performance drop smaller than 3% points when the last two layers are re-trained with HPCA), but requiring fewer training epochs (1 vs 7, respectively). This suggests potential applications in the context of transfer learning [39]. Moreover, it can be observed that HPCA performs better than HWTA.

5.4 Pros and cons of Hebbian learning

We conclude this Section with a list of pros and cons of Hebbian learning approaches, emerging from the observed results.

Pros of Hebbian learning:
– Effective for training low-level feature extractors;
– Produces better features than VAE for the classification task;
– Effective for re-training higher network layers in fewer epochs than other approaches;
– Some hybrid combinations of Hebbian and backprop help improving performance in some cases, as can be observed in Appendix A;

Cons of Hebbian learning:
– Not effective for training intermediate layers;
– Even though HPCA provides a reduction in the gap between unsupervised and supervised methods, the latter are still preferable for end-to-end network training;
– Finding the best combination of Hebbian and backprop layers is not immediate and requires exploring various network configurations.

6 Conclusions and future work

In summary, our results suggest that the Hebbian approach is suitable for training early feature extraction layers or to re-train the final layers of a pre-trained deep neural network, requiring fewer training epochs than other methods. This suggests potential applications in the context of transfer learning, where an experimenter wants to re-train or fine-tune higher network layers of a pre-trained model on a new task.

Hebbian approaches outperform VAE training, reducing the gap between unsupervised methods and supervised backprop training. Moreover, the HPCA methods seems to perform generally better than HWTA.

Moreover, supplementary results in Appendix A also show that some hybrid combinations of backprop and Hebbian layers appear to be helpful in some cases, offering performance higher than either Hebbian or supervised backprop alone.

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

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) [12] and sparse coding [24,23,31]. It might be promising also to apply Hebbian learning to enhance current state-of-the-art network architectures, either as a stand-alone learning algorithm, or in combination with backprop, as an inductive bias for regularization [26], in a semi-supervised fashion.

Hebbian learning already found application in the context of meta-learning, with the *differentiable plasticity* model [21]. In this case, the simple Hebbian learning rule, $\Delta w = \eta \, y \, x$, 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.

Conflict of interest

The authors declare that they have no conflict of interest.

Appendix

A Supplementary Results

In this Appendix, we present the additional results on MNIST, CIFAR10, and CIFAR100 datasets. Tables 7, 9, 11, show the results of hybrid training, in which part of the network layers are trained by supervised backprop training, and part with the Hebbian approach. Tables 8, 10, 12, show the results of SHC and CHL classifiers, compared with SGD classifiers, trained on the features extracted from the various layers of a pre-trained network.

A.1 MNIST

A.1.1 Hybrid network models

In Tab. 7, we report the results obtained on the MNIST test set with hybrid networks. In each row, we reported the results for a network with a different combination of Hebbian and backprop layers (the first row below the header represent the network fully trained with backprop). We used the letter ”H” to denote layers trained using the Hebbian approach, and the letter ”B” for layers trained using backprop. The letter ”G” is used for the final classifier (corresponding to the sixth layer) trained with gradient descent. The final classifier (corresponding to the sixth layer) was trained with SGD in all the cases, in order to make comparisons on equal footings. The last two columns show the resulting accuracy obtained with the corresponding combination of layers.

Tab. 7 allows us to understand what is the effect of switching a specific layer (or group of layers) in a network from backprop to Hebbian training. The first row represents the network fully trained with backprop. In the next rows we can observe the results of a network in which a single layer was switched. Both HPCA and HWTA exhibit comparable results w.r.t. full backprop training. A result slightly higher than full backprop is observed when layer 5 is replaced, suggesting that some combinations of layers might actually be helpful to increase

Table 7: MNIST accuracy (top-1) and 95% confidence intervals of hybrid network models. The first six columns describe the network configuration: H denotes a Hebbian layer, B denotes a backprop layer, G is used for the final classifier, trained by Gradient Descent but without the need for backpropagation. The first row refers to the network fully trained with backprop, the other rows compare HPCA and HWTA approaches.

L1	L2	L3	L4	L5	L6	HPCA Acc.(%)	HWTA Acc.(%)
B	B	B	B	B	G	99.59 ±0.02	99.59 ±0.02
H	B	B	B	B	G	**99.61** ±0.02	99.48 ±0.03
B	H	B	B	B	G	**99.51** ±0.03	99.48 ±0.05
B	B	H	B	B	G	**99.58** ±0.02	99.55 ±0.02
B	B	B	H	B	G	99.60 ±0.02	**99.61** ±0.02
B	B	B	B	H	G	99.61 ±0.02	**99.66** ±0.02
H	H	B	B	B	G	**99.42** ±0.02	99.35 ±0.02
B	H	H	B	B	G	**99.35** ±0.06	99.29 ±0.02
B	B	H	H	B	G	**99.50** ±0.03	99.42 ±0.02
B	B	B	H	H	G	**99.54** ±0.02	99.51 ±0.01
H	H	H	B	B	G	**99.23** ±0.04	99.22 ±0.05
B	H	H	H	B	G	**99.16** ±0.07	98.99 ±0.03
B	B	H	H	H	G	**99.30** ±0.04	99.08 ±0.02
H	H	H	H	B	G	**99.04** ±0.06	98.45 ±0.04
B	H	H	H	H	G	**98.63** ±0.03	98.25 ±0.06
H	H	H	H	H	G	**98.53** ±0.08	97.15 ±0.01

performance. In the successive rows, more layers are switched from backprop to Hebbian training, and a slight performance drop is observed, but the HPCA approach seems to perform generally better than HWTA when more Hebbian layers are involved. The most prominent difference appears when we finally replace all the network layers with Hebbian equivalent, in which case the HPCA approach shows an increase of more than 2% points over HWTA.

A.1.2 Comparison of SHC and SGD

Tab. 8 shows a comparison between SHC and SGD classifiers placed on the various layers of a network pre-trained with backprop. The results suggest that SHC is effective in classifying high-level features, achieving comparable accuracy as SGD, but requiring fewer training epochs. On the other hand, SHC is not so effective on lower layer features, although the convergence time is still fast, suggesting that the supervised Hebbian approach benefits from the use of more abstract latent representations. CHL appears to perform comparably to SGD training.

A.2 CIFAR10

A.2.1 Hybrid network models

In Tab. 9, we report the results obtained on the CIFAR10 test set with hybrid networks. The table, which has the same structure as that of the previous sub-section, allows us to understand what is the effect of switching a specific layer (or group of layers) in a network from backprop to Hebbian training. The first row represents the network fully trained with

Table 8: MNIST accuracy (top-1), 95% confidence intervals, and convergence epochs of SHC, CHL, and SGD classifiers on top of various network layer features. It can be observed that SHC achieves comparable classification accuracy as an SGD classifier, when placed on top of higher layer features, while requiring fewer training epochs. CHL performs comparably to SGD.

Layer	Method	Acc. (%)	Num. Epochs
1	SGD	95.80 ±0.02	14
	SHC	89.06 ±0.04	**10**
	CHL	**95.82** ±0.07	17
2	SGD	97.26 ±0.01	13
	SHC	95.08 ±0.03	**11**
	CHL	**97.32** ±0.05	14
3	SGD	**98.77** ±0.01	13
	SHC	98.47 ±0.01	**3**
	CHL	98.63 ±0.01	15
4	SGD	99.56 ±0.01	**5**
	SHC	99.56 ±0.01	6
	CHL	**99.57** ±0.01	7
5	SGD	99.59 ±0.02	15
	SHC	**99.62** ±0.01	**1**
	CHL	**99.62** ±0.01	**1**

Table 9: CIFAR10 accuracy (top-1) and 95% confidence intervals of hybrid network models. The first six columns describe the network configuration: H denotes a Hebbian layer, B denotes a backprop layer, G is used for the final classifier, trained by Gradient Descent but without the need for backpropagation. The first row refers to the network fully trained with backprop, the other rows compare HPCA and HWTA approaches.

L1	L2	L3	L4	L5	L6	HPCA Acc.(%)	HWTA Acc.(%)
B	B	B	B	B	G	84.95 ±0.25	84.95 ±0.25
H	B	B	B	B	G	82.84 ±0.17	**84.30** ±0.26
B	H	B	B	B	G	**81.91** ±0.10	81.40 ±0.14
B	B	H	B	B	G	79.01 ±0.29	**80.88** ±0.02
B	B	B	H	B	G	79.20 ±0.24	**81.09** ±0.16
B	B	B	B	H	G	**84.69** ±0.09	84.46 ±0.07
H	H	B	B	B	G	77.29 ±0.45	**79.97** ±0.46
B	H	H	B	B	G	**76.54** ±0.27	68.13 ±0.19
B	B	H	H	B	G	**75.53** ±0.24	73.43 ±0.17
B	B	B	H	H	G	74.49 ±0.19	**78.53** ±0.12
H	H	H	B	B	G	**72.30** ±0.28	68.71 ±0.18
B	H	H	H	B	G	**71.00** ±0.17	49.22 ±0.21
B	B	H	H	H	G	**69.53** ±0.23	68.26 ±0.14
H	H	H	H	B	G	**68.17** ±0.15	52.53 ±0.18
B	H	H	H	H	G	**63.40** ±0.27	45.29 ±0.05
H	H	H	H	H	G	**61.12** ±0.33	46.88 ±0.23

Table 10: CIFAR10 accuracy (top-1), 95% confidence intervals, and convergence epochs of SHC, CHL, and SGD classifiers on top of various network layer features. It can be observed that SHC achieves comparable classification accuracy as an SGD classifier, when placed on top of higher layer features, while requiring fewer training epochs. CHL performs comparably to SGD.

Layer	**Method**	**Acc. (%)**	**Num. Epochs**
1	SGD	**61.59** ±0.08	16
	SHC	48.36 ±0.17	**1**
	CHL	61.42 ±0.25	8
2	SGD	**67.67** ±0.11	17
	SHC	58.87 ±0.08	**1**
	CHL	67.06 ±0.20	8
3	SGD	**73.87** ±0.15	15
	SHC	70.94 ±0.05	**2**
	CHL	72.28 ±0.38	8
4	SGD	83.88 ±0.04	12
	SHC	82.78 ±0.03	**1**
	CHL	**84.10** ±0.12	3
5	SGD	84.95 ±0.25	12
	SHC	84.59 ±0.01	**1**
	CHL	**85.22** ±0.09	**1**

backprop. In the next rows we can observe the results of a network in which a single layer was switched. Both HPCA and HWTA exhibit competitive results w.r.t. full backprop training, when they are used to train the first or the fifth network layer. A small, but more significant drop is observed when inner layers are switched from backprop to Hebbian. In the successive rows, more layers are switched from backprop to Hebbian training, and a higher performance drop is observed, but the HPCA approach seems to perform better than HWTA when more Hebbian layers are involved. The most prominent difference appears when we finally replace all the deep network layers with Hebbian equivalent, in which case the HPCA approach shows an increase of 15% points over HWTA.

A.2.2 Comparison of SHC and SGD

Tab. 10 shows a comparison between SHC, CHL, and SGD classifiers placed on the various layers of a network pre-trained with backprop. The results suggest that SHC is effective in classifying high-level features, achieving comparable accuracy as SGD, but requiring fewer training epochs. On the other hand, SHC is not so effective on lower layer features, although the convergence time is still fast, suggesting that the supervised Hebbian approach benefits from the use of more abstract latent representations. CHL appears to perform comparably to SGD training.

A.3 CIFAR100

A.3.1 Hybrid network models

In Tab. 11, we report the results obtained on the CIFAR100 test set with hybrid networks. The table, which has the same structure as those of the previous sub-sections, allows us to understand what is the effect of switching a specific layer (or group of layers) in a network

Table 11: CIFAR100 accuracy (top-5) and 95% confidence intervals of hybrid network models. The first six columns describe the network configuration: H denotes a Hebbian layer, B denotes a backprop layer, G is used for the final classifier, trained by Gradient Descent but without the need for backpropagation. The first row refers to the network fully trained with backprop, the other rows compare HPCA and HWTA approaches.

L1	**L2**	**L3**	**L4**	**L5**	**L6**	**HPCA Acc.(%)**	**HWTA Acc.(%)**
B	B	B	B	B	G	80.74 ±0.05	80.74 ±0.05
H	B	B	B	B	G	76.46 ±0.34	**76.84** ±0.41
B	H	B	B	B	G	**77.41** ±0.30	75.80 ±0.31
B	B	H	B	B	G	**78.44** ±0.18	77.29 ±0.27
B	B	B	H	B	G	**77.97** ±0.17	74.42 ±0.12
B	B	B	B	H	G	**82.46** ±0.11	77.42 ±0.07
H	H	B	B	B	G	72.32 ±0.34	**72.81** ±0.28
B	H	H	B	B	G	75.41 ±0.29	**77.10** ±0.24
B	B	H	H	B	G	**75.12** ±0.26	65.89 ±0.05
B	B	B	H	H	G	**76.03** ±0.15	70.09 ±0.13
H	H	H	B	B	G	**70.26** ±0.20	66.49 ±0.42
B	H	H	H	B	G	**69.13** ±0.22	51.85 ±0.24
B	B	H	H	H	G	**69.61** ±0.13	57.61 ±0.29
H	H	H	H	B	G	**66.34** ±0.21	42.88 ±0.32
B	H	H	H	H	G	**62.27** ±0.12	41.42 ±0.13
H	H	H	H	H	G	**59.51** ±0.20	37.87 ±0.21

from backprop to Hebbian training. The first row represents our network fully trained with backprop. In the next rows we can observe the results of a network in which a single layer was switched. HWTA exhibits competitive results w.r.t. full backprop when it is used to train the first or the fifth network layer. A small, but more significant drop is observed when inner layers are switched from backprop to HWTA. On the other hand, the HPCA approach seems to perform generally better than HWTA. In particular, it slightly outperforms full backprop (by 2% points), when used to train the fifth network layer, suggesting that this kind of hybrid combinations might be useful when more complex tasks are involved. In the successive rows, more layers are switched from backprop to Hebbian training, and a higher performance drop is observed, but still, the HPCA approach exhibits a better behavior than HWTA. The most prominent difference appears when we finally replace all the network layers with Hebbian equivalent, in which case the HPCA approach shows an increase of 22% points over HWTA.

A.3.2 Comparison of SHC and SGD

Tab. 12 shows a comparison between SHC, CHL, and SGD classifiers placed on the various layers of a network pre-trained with backprop. In this case, SHC achieves comparable accuracy as SGD (even with a slight improvement of 6% points on layer 3), but requiring fewer training epochs, suggesting that the approach might be especially useful when more complex tasks are involved. On the other hand, in this case, lower performance is observed when CHL is used, suggesting that this approach has more difficulties in scaling to more complex datasets.

Table 12: CIFAR100 accuracy (top-5), 95% confidence intervals, and convergence epochs of SHC, CHL, and SGD classifiers on top of various network layer features. It can be observed that SHC achieves comparable classification accuracy as an SGD classifier, while requiring fewer training epochs. CHL seems to perform worse on this dataset.

Layer	**Method**	**Acc. (%)**	**Num. Epochs**
1	SGD	51.67 ±0.10	14
	SHC	**51.70** ±0.12	**1**
	CHL	29.12 ±0.54	13
2	SGD	60.84 ±0.19	11
	SHC	**63.67** ±0.06	**1**
	CHL	31.94 ±0.31	12
3	SGD	67.01 ±0.13	15
	SHC	**73.99** ±0.30	**1**
	CHL	25.34 ±0.31	13
4	SGD	78.85 ±0.10	15
	SHC	**79.98** ±0.04	**1**
	CHL	27.34 ±0.77	13
5	SGD	**80.74** ±0.05	7
	SHC	79.45 ±0.02	**1**
	CHL	41.32 ±0.43	12

References

1. 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)
2. 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)
3. Becker, S., Plumbley, M.: Unsupervised neural network learning procedures for feature extraction and classification. Applied Intelligence **6**(3), 185–203 (1996)
4. Diehl, P.U., Cook, M.: Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Frontiers in computational neuroscience **9**, 99 (2015)
5. Ferré, P., Mamalet, F., Thorpe, S.J.: Unsupervised feature learning with winner-takes-all based stdp. Frontiers in computational neuroscience **12**, 24 (2018)
6. Földiak, P.: Adaptive network for optimal linear feature extraction. In: Proceedings of IEEE/INNS Int. Joint. Conf. Neural Networks, vol. 1, pp. 401–405 (1989)
7. Grossberg, S.: Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. Biological cybernetics **23**(3), 121–134 (1976)
8. Haykin, S.: Neural networks and learning machines, 3 edn. Pearson (2009)
9. 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)
10. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 558–567 (2019)
11. 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)
12. Hyvarinen, A., Karhunen, J., Oja, E.: Independent component analysis. Studies in informatics and control **11**(2), 205–207 (2002)

13. Karhunen, J., Joutsensalo, J.: Generalizations of principal component analysis, optimization problems, and neural networks. Neural Networks **8**(4), 549–562 (1995)
14. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2013)
15. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological cybernetics **43**(1), 59–69 (1982)
16. 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)
17. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems (2012)
19. Lagani, G.: Hebbian learning algorithms for training convolutional neural networks. Master's thesis, School of Engineering, University of Pisa, Italy (2019). URL https://etd.adm.unipi.it/theses/available/etd-03292019-220853/
20. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
21. Miconi, T., Clune, J., Stanley, K.O.: Differentiable plasticity: training plastic neural networks with backpropagation (2018)
22. Movellan, J.R.: Contrastive hebbian learning in the continuous hopfield model. In: Connectionist models, pp. 10–17. Elsevier (1991)
23. Olshausen, B.A.: Learning linear, sparse, factorial codes (1996)
24. 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)
25. O'Reilly, R.C.: Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. Neural computation **8**(5), 895–938 (1996)
26. O'reilly, R.C.: Generalization in interactive networks: The benefits of inhibitory competition and hebbian learning. Neural computation **13**(6), 1199–1241 (2001)
27. O'Reilly, R.C., Munakata, Y.: Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain. MIT press (2000)
28. 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)
29. 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)
30. Ponulak, F.: Resume-new supervised learning method for spiking neural networks. technical report. In: Institute of Control and Information Engineering, Poznan University of Technology (2005)
31. 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)
32. Rumelhart, D.E., Zipser, D.: Feature discovery by competitive learning. Cognitive science **9**(1), 75–112 (1985)
33. Sanger, T.D.: Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural networks **2**(6), 459–473 (1989)
34. Shrestha, A., Ahmed, K., Wang, Y., Qiu, Q.: Stable spike-timing dependent plasticity rule for multilayer unsupervised and supervised learning. In: 2017 international joint conference on neural networks (IJCNN), pp. 1999–2006. IEEE (2017)
35. 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)
36. Wadhwa, A., Madhow, U.: Bottom-up deep learning using the hebbian principle (2016)
37. Wadhwa, A., Madhow, U.: Learning sparse, distributed representations using the hebbian principle (2016)
38. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: International conference on machine learning, pp. 478–487 (2016)
39. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? (2014)