Training Convolutional Neural Networks with Competitive Hebbian Learning Approaches*

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

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

Abstract. We explore competitive Hebbian learning strategies to train feature detectors in Convolutional Neural Networks (CNNs), without supervision. We consider variants of the Winner-Takes-All (WTA) strategy explored in previous works, i.e. k-WTA, e-soft-WTA and p-soft-WTA, performing experiments on different object recognition datasets. Results suggest that the Hebbian approaches are effective to train early feature extraction layers, or to re-train higher layers of a pre-trained network, with soft competition generally performing better than other Hebbian approaches explored in this work. Our findings encourage a path of cooperation between neuroscience and computer science towards a deeper investigation of biologically inspired learning principles.

Keywords: Neural networks · Machine learning · Hebbian learning · Competitive learning · Computer vision · Biologically inspired

1 Introduction

While deep learning has achieved outstanding results in a variety of domains, ranging from computer vision [9] to language processing [4], and reinforcement learning [24], there are still doubts about the biological plausibility of the learning algorithms in use, that are based on supervised end-to-end training with error backpropagation (*backprop*). This strategy lacks biological plausibility, according to neuroscientists [21]. This motivates investigation into different learning approaches, inspired by mammalian plasticity, which might eventually lead to improvements in machine learning models, as well as to a better understanding of how the brain works.

In this article, we consider the biologically plausible Hebbian learning principle [6, 8], coupled with different *competitive* learning strategies [7, 17, 18]. Specifically, we consider variants of the Winner-Takes-All (WTA) strategy, namely k-WTA, e-soft-WTA, and p-soft-WTA. In particular, the latter two strategies are novel variants of the soft-WTA approach [18], that we introduce in order

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

to make soft competition suitable in practical scenarios. The respective learning rules and details are described in the following sections. We provide an experimental evaluation of the proposed strategies in the context of Deep Neural Network (DNN) training on popular computer vision datasets, namely MNIST [16], CIFAR10, and CIFAR100 [12]

Hebbian learning was explored in previous works, to train network layers for computer vision tasks [2,14,22,25]. Nonetheless, only relatively shallow networks were considered. Deeper network architectures were also considered in [1], but still, a thorough investigation of the various competitive learning strategies is missing. Our experiments on different object recognition datasets show that the Hebbian approaches are effective to train early feature extraction layers, or to re-train higher layers of a pre-trained network, when compared to supervised backprop. Comparison with a popular unsupervised approach, the Variational Auto-Encoder (VAE), also based on backprop, suggests that Hebbian learning might represent a better unsupervised feature extraction strategy. Moreover, *soft* competition strategies (e-soft-WTA and p-soft-WTA) perform generally better than *sharp* variants (WTA and k-WTA).

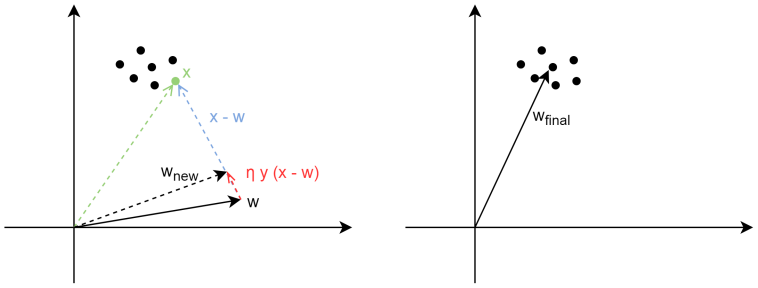
Our work is the results of a cooperation between neuroscience and computer science, suggesting that the collaboration between these two fields might bring a promising potential. Our contributions can be summarized as follows:

- We explore the different competitive learning strategies (WTA, k-WTA, e-soft-WTA, p-soft-WTA), to train Convolutional Neural Networks (CNNs) for feature extraction and classification;
- Among the approaches that we explore, we propose two novel strategies, namely e-soft-WTA and p-soft-WTA, inspired by the soft-WTA approach, but aiming to make soft competition suitable for practical tasks.
- Experimental evaluation of the various approaches on different object recognition datasets is performed.

The remainder of this paper is structured as follows: Section 2 presents some related work on this field; Section 3 introduces the various competitive Hebbian learning strategies that we explored; Section 4 describes the scenarios in which we applied the above mentioned strategies; Section 5 goes into the details of our experiments; Section 6 provides the results of our evaluation; Finally, in Section 7, we present our conclusions and hints for future directions.

2 Related work

In previous work, Hebbian learning was used together with k-WTA competition on computer vision tasks, but only on relatively shallow networks [14, 25]. Nonetheless, results were comparable to those achieved by backprop on networks with similar structure, thus motivating further interest. In [2,22], a different approach based on Hebbian/anti-Hebbian learning was explored, which minimized an unsupervised similarity matching objective, equivalent to Principal Component Analysis (PCA) in the linear case. Hebbian PCA rules have also been widely



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

Fig. 1: Hebbian updates with weight decay.

studied in literature [3,23]. Still, the experiments are limited to relatively shallow networks. Deeper networks trained by Hebbian WTA were considered in [1], were it was confirmed that the WTA approach was effective for training early feature extraction layers, thus being suitable for relatively shallow networks, but also to retrain higher layers of a pre-trained network (including the final classifier, by a supervised Hebbian learning variant [15]), while requiring fewer training epochs than backprop, thus suggesting potential applications in the context of transfer learning [26]. Nonetheless, the results of this latter work were preliminary, and involved a single approach (WTA) and a single dataset for testing (CIFAR10).

3 Competitive 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. According to the Hebbian principle, in its most basic form, the latter term is computed as

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

where η is the learning rate. Basically, this rule states that the weight on a given synapse is potentiated when the input on that synapse and the output of the neuron are simultaneously high, thus reinforcing connections between neurons whose activations are correlated.

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

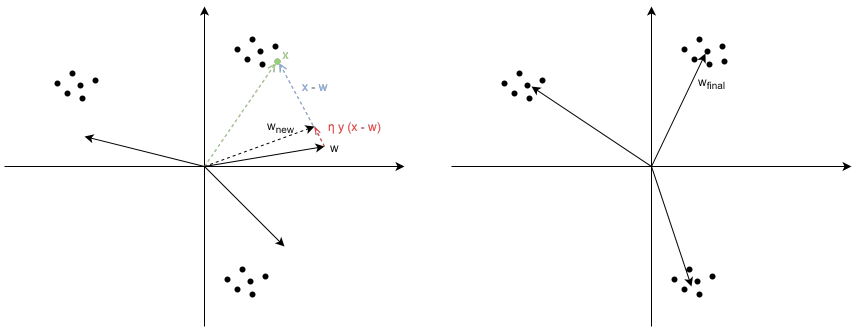
$$\Delta\mathbf{w} = \eta\, y\, \mathbf{x} - \eta\, y\, \mathbf{w} = \eta\, y\, (\mathbf{x} - \mathbf{w}) \tag{3}$$

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] 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).

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, actual neural codes exhibit a *sparse, distributed* representation, where multiple neurons activate combinatorially to encode different properties of the input, resulting in an improved coding power. The importance of sparse, distributed representations was also highlighted in [5, 20].

A more distributed coding scheme could be obtained by choosing more than one winner at a time. In the k-WTA strategy [17], the k top-activating neurons are selected as winners and allowed to perform the weight update. A *soft* form of competition was also proposed in literature [18]. In this soft-WTA approach, a reward is attributed to each neuron depending on the value of its activation, so that neurons with higher activation also receive a higher reward. Neurons perform update steps whose length is proportional to their reward. The reward



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

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

r_i for neuron i is computed as follows:

$$r_i = \frac{y_i}{\sum_j y_j} \tag{4}$$

So, basically, the reward is obtained as an L_1 normalization of the activations. We found that this formulation actually worked 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 distribute the reward:

$$r_i = \frac{e^{y_i/T}}{\sum_j e^{y_j/T}} \tag{5}$$

where T is the *temperature* hyperparameter. 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. We refer to this *exponential* form of soft competition simply as e-soft-WTA. We also explore a different formulation, based on L_p normalization, in which the reward is computed as:

$$r_i = \frac{y_i^p}{\sum_j y_j^p} \tag{6}$$

In this case, the value p acts as (inverse) temperature parameter. We refer to this *power* form of soft competition as p-soft-WTA.

4 Deep competitive Hebbian learning

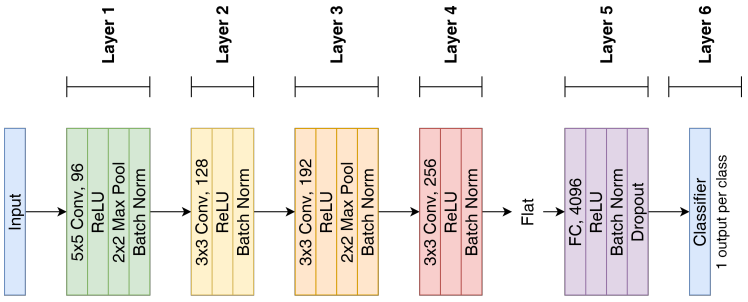


Fig. 3: The neural network used for the experiments.

The core part of our experiments consisted in training the deep layers of a neural network consisting of six layers: five deep layers plus a final linear classifier. The various layers were interleaved with other processing stages (such as ReLU nonlinearities, max pooling, etc.), as shown in Fig. 3. The architecture was inspired by AlexNet [13], but one of the fully connected layers was removed

and, in general, the number of neurons was slightly modified, in order to reduce the computational cost of the experiments. The network was trained using the k-WTA, e-soft-WTA and p-soft-WTA approaches, as well as backprop, in order to compare the results.

Since the comparisons mentioned so far involve supervised models (backprop-based) and unsupervised models (Hebbian), we also deemed interesting to compare the Hebbian models with other unsupervised (but still backprop-based) methods. Specifically, we considered a Variational Auto-Encoder (VAE) [11] : the network model in Fig. 3, 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.

In order to evaluate the quality of the features extracted from the various layers of the trained models for the image classification tasks, we placed a linear classifier on top of each already trained layer , and we evaluated the accuracy achieved by classifying the corresponding features. This was done both for the backprop trained network, for the Hebbian trained networks, and for the VAE network. The linear classifier was trained with Stochastic Gradient Descent (SGD) in all cases. Notice that this does not raise biological plausibility issues, because backpropagation is not required when SGD is used to train a single layer. Even if the Hebbian approach is unsupervised, it is also possible to apply a supervised variant [1, 15] for training the linear classifier, although, at this stage, we preferred to use SGD in all cases, in order to make comparisons on equal footings. Indeed, the SGD weight update can be considered as a form of supervised Hebbian update, modulated by a teacher signal. Later, we also present a comparison of trained with SGD and with the supervised Hebbian variant, placed on top of features extracted from various network layers, and evaluated on different datasets.

We also implemented hybrid network models, i.e. networks in which some layers were trained with backprop and other layers were trained with Hebbian approach , in order to asses up to which extent backprop layers in our model could be replaced with Hebbian equivalent without excessive impact on the accuracy. 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.

5 Details of training

We implemented our experiments using PyTorch. [3] We used the network architecture shown in Fig. 3. 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, and with the competitive Hebbian rules, in order to compare the results. For VAE training, we

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

used a network model with 256 gaussian latent variables and a specular decoder structure w.r.t. the encoder. The decoder part was removed at test time and the features extracted from encoder layers were used for classification. 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, Nesterov correction, and dropout rate 0.5. An L2 penalty was also used to improve regularization, with weight decay coefficient set to $5 \cdot 10^{-2}$ for MNIST and CIFAR10, and to 10^{-2} for CIFAR100. The VAE was trained in the same fashion but, obviously, in an unsupervised image encoding-decoding task, and no L2 penalty nor dropout was used in this case.

During Hebbian 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. Images were preprocessed by a whitening transformation as described in [1, 15], although this step didn't have any significant effect for backprop training. A hyperparameter k is defined to control the behavior of k-WTA, e-soft-WTA, and p-soft-WTA. For the k-WTA approach, the parameter k is simply the number of neurons selected as winners. In e-soft-WTA, the parameter k is defined as the softmax temperature, i.e. $k = T$. For the p-soft-WTA approach, the parameter k is defined as the inverse of the exponent p used to compute the L_p normalization, i.e. $k = \frac{1}{p}$. In this way, a common parameter k controls how reward is distributed: roughly speaking, higher k corresponds to a reward distributed among more neurons, while lower k corresponds to reward distributed among fewer neurons, up to the special case of a single neuron being the only winner, corresponding to simple WTA. In our experiments, we set $k = 5$ for k-WTA, $k = 0.02$ for e-soft-WTA, $k = 0.05$ for p-soft-WTA.

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

All the above mentioned hyperparameters resulted from a parameter search to maximize the accuracy in the respective scenarios.

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.

Table 1: MNIST accuracy (top-1) and 95% confidence intervals on features extracted from convolutional network layers.

Layer	BP	VAE	WTA	5-WTA	e-soft-WTA	p-soft-WTA
1	95.80 ±0.02	**98.67** ±0.03	98.16 ±0.05	98.19 ±0.08	98.15 ±0.06	98.20 ±0.05
2	97.26 ±0.01	**98.90** ±0.03	98.52 ±0.06	98.45 ±0.07	98.47 ±0.08	98.47 ±0.08
3	98.77 ±0.01	98.30 ±0.02	98.55 ±0.02	98.38 ±0.08	**98.56** ±0.02	98.51 ±0.04
4	99.56 ±0.01	94.68 ±0.04	96.56 ±0.04	96.45 ±0.07	96.89 ±0.10	**97.07** ±0.04
5	99.59 ±0.02	90.32 ±0.06	**97.15** ±0.01	96.18 ±0.08	96.92 ±0.06	97.09 ±0.08

6 Results

In the following subsections, we present the experimental results on MNIST, CIFAR10, and CIFAR100 datasets. We performed five independent iterations of each experiment, using different seeds, averaging the results and computing 95% confidence intervals.

6.1 MNIST

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

In Tab. 1, we report the MNIST test accuracy obtained by classifiers placed on top of the various layers of the network. We compare the results obtained on the network trained with supervised backprop (BP), VAE, and competitive Hebbian approaches. We can observe that the 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.

Moreover, the Hebbian approaches seem to perform comparably to each other. They also perform comparably or better w.r.t. the unsupervised VAE approach, especially when higher level features are considered, with an improvement of almost 7% points on the fifth layer.

In Tab. 2, 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 baseline 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.

Tab. 2 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 our baseline for comparison, i.e. the network fully trained with

Table 2: MNIST accuracy (top-1) and 95% confidence intervals of hybrid network models.

L1	**L2**	**L3**	**L4**	**L5**	**L6**	**Accuracy (%)**			
B	B	B	B	B	G	99.59 ±0.02			
WTA approach						**WTA**	**5-WTA**	**e-soft-WTA**	**p-soft-WTA**
H	B	B	B	B	G	99.48 ±0.03	**99.53** ±0.05	99.45 ±0.03	99.49 ±0.04
B	H	B	B	B	G	99.48 ±0.05	99.42 ±0.02	99.52 ±0.03	**99.53** ±0.03
B	B	H	B	B	G	**99.55** ±0.02	99.54 ±0.03	**99.55** ±0.02	99.54 ±0.04
B	B	B	H	B	G	**99.61** ±0.02	99.59 ±0.02	99.58 ±0.03	99.58 ±0.02
B	B	B	B	H	G	**99.66** ±0.02	99.61 ±0.01	99.65 ±0.03	99.64 ±0.04
H	H	B	B	B	G	99.35 ±0.02	99.30 ±0.03	**99.36** ±0.03	99.34 ±0.03
B	H	H	B	B	G	99.29 ±0.02	99.28 ±0.09	99.31 ±0.05	**99.34** ±0.03
B	B	H	H	B	G	**99.42** ±0.02	99.34 ±0.02	99.37 ±0.07	99.35 ±0.04
B	B	B	H	H	G	99.51 ±0.01	99.37 ±0.02	**99.58** ±0.02	**99.58** ±0.02
H	H	H	B	B	G	**99.22** ±0.05	99.12 ±0.03	99.20 ±0.04	99.19 ±0.02
B	H	H	H	B	G	98.99 ±0.03	98.92 ±0.03	99.04 ±0.05	**99.07** ±0.02
B	B	H	H	H	G	99.08 ±0.02	98.51 ±0.03	**99.25** ±0.01	98.98 ±0.02
H	H	H	H	B	G	98.45 ±0.04	98.27 ±0.08	98.45 ±0.07	**98.47** ±0.06
B	H	H	H	H	G	98.25 ±0.06	97.28 ±0.05	98.43 ±0.07	**98.46** ±0.04
H	H	H	H	H	G	**97.15** ±0.01	96.18 ±0.08	96.92 ±0.06	97.09 ±0.08

backprop. In the next rows we can observe the results of a network in which a single layer was switched. The Hebbian approaches exhibit comparable results w.r.t. the baseline. A result slightly higher than the baseline is observed when layer 5 is replaced, suggesting that some combinations of layers might actually be helpful. In the successive rows, more layers are switched from backprop to Hebbian training, and a slight performance drop is observed. The Hebbian approaches appear to perform comparably to each other, although it seems that soft approaches (e-soft-WTA and p-soft-WTA) tend to behave better when applied to higher layers, while sharp approaches (WTA and 5-WTA)) seem to be preferable for lower layers.

Tab. 3 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 [1, 15] 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 [26].

6.2 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.

Table 3: MNIST accuracy (top-1), 95% confidence intervals, and convergence epochs obtained by retraining higher layers of a pre-trained network.

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	WTA	99.55 ±0.02	**1**
						5-WTA	99.54 ±0.03	**1**
						e-soft-WTA	99.53 ±0.03	**1**
						p-soft-WTA	99.53 ±0.02	**1**

Table 4: CIFAR10 accuracy (top-1) and 95% confidence intervals on features extracted from convolutional network layers.

Layer	BP	VAE	WTA	5-WTA	e-soft-WTA	p-soft-WTA
1	61.59 ±0.08	60.71 ±0.16	64.79 ±0.34	63.36 ±0.20	**65.08** ±0.41	65.00 ±0.43
2	67.67 ±0.11	56.32 ±0.31	64.35 ±0.35	60.88 ±0.16	**66.27** ±0.42	66.20 ±0.43
3	73.87 ±0.15	41.31 ±0.16	59.69 ±0.16	55.28 ±0.10	**63.94** ±0.11	63.50 ±0.29
4	83.88 ±0.04	29.58 ±0.07	48.56 ±0.17	43.51 ±0.26	54.94 ±0.15	**54.99** ±0.17
5	84.95 ±0.25	26.95 ±0.12	46.88 ±0.23	42.40 ±0.11	**52.31** ±0.15	51.99 ±0.28

In Tab. 4, we report the CIFAR10 test accuracy obtained by classifiers placed on top of the various convolutional layers of the network. We compare the results obtained on the network trained with supervised backprop (BP), VAE, and competitive Hebbian approaches. We can observe that the Hebbian approaches reach 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.

Moreover, soft Hebbian approaches seem to perform comparably to each other, and better than sharp approaches. The most prominent difference appears on layer 4, where soft Hebbian approaches reach an improvement of almost 7% points over sharp approaches. Still, further research is needed in order to close the gap with backprop also when more layers are added, in order to make the Hebbian approach suitable as a biologically plausible alternative to backprop for training deep networks. In fact, Hebbian approaches seem to suffer from a decrease in performance when going further on with the number of layers. The same holds also for the unsupervised VAE approach, although Hebbian features appear to behave better than unsupervised VAE features, especially on higher layers, with an improvement up to 25% points on the fifth layer.

In Tab. 5, 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 subsection, 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

Table 5: CIFAR10 accuracy (top-1) and 95% confidence intervals of hybrid network models.

L1	**L2**	**L3**	**L4**	**L5**	**L6**	**Accuracy (%)**			
B	B	B	B	B	G	84.95 ±0.25			
WTA approach						**WTA**	**5-WTA**	**e-soft-WTA**	**p-soft-WTA**
H	B	B	B	B	G	**84.30** ±0.26	82.75 ±0.22	84.07 ±0.32	84.07 ±0.31
B	H	B	B	B	G	**81.40** ±0.14	81.02 ±0.15	80.74 ±0.40	81.07 ±0.21
B	B	H	B	B	G	**80.88** ±0.02	79.39 ±0.17	78.30 ±0.35	78.36 ±0.49
B	B	B	H	B	G	**81.09** ±0.16	80.61 ±0.24	76.92 ±0.25	76.98 ±0.17
B	B	B	B	H	G	**84.46** ±0.07	84.42 ±0.09	84.36 ±0.07	84.32 ±0.15
H	H	B	B	B	G	**79.97** ±0.46	77.75 ±0.40	78.87 ±0.19	79.04 ±0.29
B	H	H	B	B	G	68.13 ±0.19	66.26 ±0.30	74.20 ±0.26	**74.41** ±0.15
B	B	H	H	B	G	73.43 ±0.17	71.39 ±0.22	**74.10** ±0.21	73.85 ±0.25
B	B	B	H	H	G	**78.53** ±0.12	76.29 ±0.22	74.88 ±0.26	74.92 ±0.24
H	H	H	B	B	G	68.71 ±0.18	64.50 ±0.21	**71.75** ±0.20	**71.75** ±0.23
B	H	H	H	B	G	49.22 ±0.21	50.53 ±0.21	61.45 ±0.22	**62.65** ±0.31
B	B	H	H	H	G	68.26 ±0.14	64.72 ±0.21	67.96 ±0.18	**68.57** ±0.21
H	H	H	H	B	G	52.53 ±0.18	48.64 ±0.35	**59.48** ±0.29	59.27 ±0.16
B	H	H	H	H	G	45.29 ±0.05	44.47 ±0.28	54.91 ±0.13	**55.87** ±0.19
H	H	H	H	H	G	46.88 ±0.23	42.40 ±0.11	**52.31** ±0.15	51.99 ±0.28

row represents our baseline for comparison, i.e. 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. Hebbian approaches exhibit comparable results w.r.t. the baseline 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 learning. In the successive rows, more layers are switched from backprop to Hebbian training, and a higher performance drop is observed. Still, sharp approaches seem to be preferable when few layers are switched, but soft approaches seem to perform better when more Hebbian layers are involved. The most prominent difference appears when layers 2 to 4 are replaced with Hebbian equivalent, in which case soft approaches show an increase of almost 12% points over sharp approaches.

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 [1, 15] is used to train the final classifier), achieving accuracy comparable to backprop, but requiring fewer training epochs (1 vs 12, respectively). This suggests potential applications in the context of transfer learning [26].

6.3 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,

Table 6: CIFAR10 accuracy (top-1), 95% confidence intervals, and convergence epochs obtained by retraining higher layers of a pre-trained network.

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	WTA	82.48 ±0.14	**1**
						5-WTA	82.42 ±0.11	**1**
						e-soft-WTA	82.67 ±0.16	**1**
						p-soft-WTA	82.65 ±0.14	**1**

Table 7: CIFAR100 accuracy (top-5) and 95% confidence intervals on features extracted from convolutional network layers.

Layer	**BP**	**VAE**	**WTA**	**5-WTA**	**e-soft-WTA**	**p-soft-WTA**
1	66.57 ±0.06	58.46 ±0.12	59.56 ±0.13	59.01 ±0.25	**60.77** ±0.26	60.46 ±0.22
2	71.75 ±0.19	54.63 ±0.20	58.49 ±0.20	57.08 ±0.28	**62.98** ±0.16	62.65 ±0.30
3	75.05 ±0.28	39.46 ±0.15	52.97 ±0.22	52.07 ±0.12	57.89 ±0.25	**59.05** ±0.30
4	78.84 ±0.18	26.42 ±0.21	37.38 ±0.12	38.20 ±0.14	44.02 ±0.29	**45.98** ±0.13
5	78.53 ±0.38	23.03 ±0.12	37.87 ±0.21	34.33 ±0.18	43.45 ±0.26	**44.89** ±0.19

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.

In Tab. 7, we report the CIFAR100 top-5 test accuracy obtained by classifiers placed on top of the various convolutional layers of the network. We compare the results obtained on the network trained with supervised backprop (BP), VAE, and competitive Hebbian approaches. We can observe that Hebbian approaches reach competitive 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.

Moreover, soft Hebbian approaches seem to perform comparably to each other, and better than sharp approaches. The most prominent difference appears on layer 4, where soft Hebbian approaches reach an improvement of almost 8% points over sharp approaches. Still, Hebbian approaches seem to suffer from a decrease in performance when going further on with the number of layers. The same holds also for the unsupervised VAE approach, although Hebbian features appear to behave better than unsupervised VAE features, especially on higher layers, with an improvement up to 21% points on the fifth layer.

In Tab. 8, 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 from backprop to Hebbian training. The first row represents our baseline for comparison, i.e. the network fully trained

Table 8: CIFAR100 accuracy (top-5) and 95% confidence intervals of hybrid network models.

L1	**L2**	**L3**	**L4**	**L5**	**L6**	**Accuracy (%)**			
B	B	B	B	B	G	78.53 ±0.38			
WTA approach						**WTA**	**5-WTA**	**e-soft-WTA**	**p-soft-WTA**
H	B	B	B	B	G	76.84 ±0.41	76.58 ±0.27	**77.81** ±0.25	77.07 ±0.37
B	H	B	B	B	G	**75.80** ±0.31	73.82 ±0.22	75.30 ±0.34	75.36 ±0.53
B	B	H	B	B	G	**77.29** ±0.27	76.15 ±0.35	76.68 ±0.23	76.50 ±0.28
B	B	B	H	B	G	**74.42** ±0.12	73.36 ±0.21	70.68 ±0.38	71.56 ±0.20
B	B	B	B	H	G	77.42 ±0.07	**77.77** ±0.19	76.99 ±0.18	77.01 ±0.15
H	H	B	B	B	G	72.81 ±0.28	72.22 ±0.26	**74.50** ±0.33	73.98 ±0.43
B	H	H	B	B	G	**77.10** ±0.24	65.15 ±0.19	71.79 ±0.17	71.67 ±0.22
B	B	H	H	B	G	65.89 ±0.05	63.16 ±0.17	67.71 ±0.33	**67.90** ±0.20
B	B	B	H	H	G	**70.09** ±0.13	65.61 ±0.15	68.90 ±0.17	69.77 ±0.20
H	H	H	B	B	G	66.49 ±0.42	62.99 ±0.30	69.21 ±0.24	**70.16** ±0.30
B	H	H	H	B	G	51.85 ±0.24	51.10 ±0.24	**58.80** ±0.12	58.61 ±0.13
B	B	H	H	H	G	57.61 ±0.29	53.80 ±0.33	60.71 ±0.20	**60.77** ±0.10
H	H	H	H	B	G	42.88 ±0.32	43.72 ±0.28	52.49 ±0.31	**55.09** ±0.32
B	H	H	H	H	G	41.42 ±0.13	40.13 ±0.14	**51.63** ±0.26	51.21 ±0.17
H	H	H	H	H	G	37.87 ±0.21	34.33 ±0.18	43.45 ±0.26	**44.89** ±0.19

with backprop. In the next rows we can observe the results of a network in which a single layer was switched. The Hebbian approaches exhibit comparable results with the baseline when they are used used to train the first, third, or fifth network layer. A small, but more significant drop is observed when other layers are switched from backprop to Hebbian learning. In the successive rows, more layers are switched from backprop to Hebbian training, and a higher performance drop is observed. Still, sharp approaches seem to be preferable when few layers are switched, but soft approaches seem to perform better when more Hebbian layers are involved. The most prominent difference appears when layers 1 to 4 are replaced with Hebbian equivalent, in which case soft approaches show an increase of almost 13% points over sharp approaches.

Tab. 9 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 [1, 15] is used to train the final classifier), achieving accuracy comparable to backprop, but requiring fewer training epochs (1 vs 7, respectively). This suggests potential applications in the context of transfer learning [26].

7 Conclusions and future work

In conclusion, our results suggest that competitive learning approaches are effective for training early feature extraction layers, or to re-train higher layers of a

Table 9: CIFAR100 accuracy (top-5), 95% confidence intervals, and convergence epochs obtained by retraining higher layers of a pre-trained network.

L1	**L2**	**L3**	**L4**	**L5**	**L6**	**Method**	**Acc.(%)**	**Num. Epochs**
B	B	B	B	B	G	BP	78.53 ±0.38	7
B	B	B	B	B	H	SHC	**79.45** ±0.02	**1**
B	B	B	B	H	H	WTA	63.62 ±0.27	**1**
						5-WTA	59.76 ±0.38	**1**
						e-soft-WTA	70.44 ±0.23	**1**
						p-soft-WTA	70.36 ±0.26	2

pre-trained network, while requiring fewer training epochs than backprop, suggesting potential applications in transfer learning [26]. In particular, Hebbian approaches seem to produce better features than unsupervised VAE training for the classification tasks, with soft competitive approaches (e-soft-WTA and p-soft-WTA) generally performing better than sharp competitive learning variants (WTA and k-WTA).

In future works, we plan to explore other Hebbian approaches that are based on sparse coding [19, 20] and Independent Component Analysis (ICA) [10]. It is also interesting to investigate strategies to combine Hebbian updates with gradient descent updates, in a semi-supervised fashion, in order to combine the task-specific knowledge given by supervised backprop training with the general knowledge extracted by unsupervised Hebbian learning. Finally, an exploration of competitive approaches w.r.t. adversarial examples also deserves attention.

We hope that our work can stimulate further interest and cooperation between the computer science and neuroscience communities towards this field.

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. 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)
5. 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)
6. Gerstner, W., Kistler, W.M.: Spiking neuron models: Single neurons, populations, plasticity. Cambridge university press (2002)

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. Pearson, 3 edn. (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. Hyvarinen, A., Karhunen, J., Oja, E.: Independent component analysis. Studies in informatics and control **11**(2), 205–207 (2002)
11. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
12. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems (2012)
14. Krotov, D., Hopfield, J.J.: Unsupervised learning by competing hidden units. Proceedings of the National Academy of Sciences **116**(16), 7723–7731 (2019)
15. 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/
16. 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)
17. Majani, E., Erlanson, R., Abu-Mostafa, Y.S.: On the k-winners-take-all network. In: Advances in neural information processing systems. pp. 634–642 (1989)
18. Nowlan, S.J.: Maximum likelihood competitive learning. In: Advances in neural information processing systems. pp. 574–582 (1990)
19. Olshausen, B.A.: Learning linear, sparse, factorial codes. Massachusetts Institute of Technology, AIM-1580 (1996)
20. 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)
21. O'Reilly, R.C., Munakata, Y.: Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain. MIT press (2000)
22. 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)
23. Sanger, T.D.: Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural networks **2**(6), 459–473 (1989)
24. 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)
25. Wadhwa, A., Madhow, U.: Bottom-up deep learning using the hebbian principle (2016)
26. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? arXiv preprint arXiv:1411.1792 (2014)