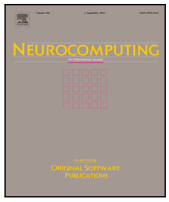
Neurocomputing 595 (2024) 127867

Contents lists available at ScienceDirect

Neurocomputing

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



Scalable bio-inspired training of Deep Neural Networks with FastHebb

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

[a] ISTI-CNR, Pisa, 56124, Italy
[b] Joanneum Research, Graz, 8010, Austria

ARTICLE INFO

Communicated by Q. Zheng

Keywords:
Hebbian learning
Deep learning
Neural networks
Biologically inspired

ABSTRACT

Recent work on sample efficient training of Deep Neural Networks (DNNs) proposed a semi-supervised methodology based on biologically inspired Hebbian learning, combined with traditional backprop-based training. Promising results were achieved on various computer vision benchmarks, in scenarios of scarce labeled data availability. However, current Hebbian learning solutions can hardly address large-scale scenarios due to their demanding computational cost. In order to tackle this limitation, in this contribution, we investigate a novel solution, named *FastHebb* (FH), based on the reformulation of Hebbian learning rules in terms of matrix multiplications, which can be executed more efficiently on GPU. Starting from Soft-Winner-Takes-All (SWTA) and Hebbian Principal Component Analysis (HPCA) learning rules, we formulate their improved FH versions: SWTA-FH and HPCA-FH. We experimentally show that the proposed approach accelerates training speed up to 70 times, allowing us to gracefully scale Hebbian learning experiments on large datasets and network architectures such as ImageNet and VGG.

1. Introduction

Recent efforts from the research community focused on the development of biologically plausible alternatives to the backpropagation algorithm for Deep Neural Network (DNN) training. Biological constraints require neurons to use only locally available information to compute the weight updates, and neuroscientific observations suggest that synaptic plasticity follows the *Hebbian* model [1,2]. In simple terms, the weight update should be proportional to the input on the respective synapse, and the neuron output at a given point in time. The study of biologically realistic learning models is interesting both because they are well suited for neuromorphic applications [3,4], and for the perspective to better understand the mechanisms behind biological intelligence and use them to enhance current Artificial Intelligence (AI) technologies.

Among the recently proposed bio-inspired learning approaches, Contrastive Hebbian Learning (CHL) [5] and Equilibrium Propagation (EP) [6] leverage recurrent architectures with Hebbian and anti-Hebbian phases, showing that the resulting update steps approximate backprop. More recently, the Forward–Forward (FF) approach has been proposed [7] for feedforward networks, which is also based on an alternation between two phases. While the approaches mentioned above focus on supervised learning solutions, a lot of attention on bio-inspired methods has converged on unsupervised learning. For example, the Similarity Matching criterion [8–11] or the Hebbian PCA rule [12,13] allow neurons to learn to extract the principal components from data. Similarly, Hebbian learning with Winner-Takes-All (WTA) competition allows neurons to find clusters in the data space [14–20]. This reveals interesting connections between the Hebbian theory of learning and data science.

In this work, we focus on a hybrid solution of unsupervised Hebbian learning and supervised backprop training, which are combined together in a semi-supervised fashion. In fact, supervised training alone has the disadvantage of requiring numerous training samples to achieve high performances, but the latter are often expensive to gather, requiring a consistent manual effort. To circumvent this issue, a possible direction is to pre-train the model on a large amount of unlabeled data, with an unsupervised algorithm, and then fine-tune with supervision on a small labeled dataset [21,22]. In this scenario, recent work has shown superior performance of Hebbian-based semi-supervised training, compared to other unsupervised methods for pre-training, such as Variational Auto-Encoders (VAE) [23], especially in scenarios where the available labeled data is scarce [13,17]. Due to the difficulty of collecting labeled data, these scenarios are of strong practical interest.

Despite their promising results, current Hebbian learning solutions can hardly be used to address large-scale problems, due to their demanding computational cost. In this perspective, the goal of our contribution is to address the performance limitations of Hebbian algorithms.

* Corresponding author.
E-mail addresses: gabriele.lagani@isti.cnr.it (G. Lagani), fabrizio.falchi@isti.cnr.it (F. Falchi), claudio.gennaro@isti.cnr.it (C. Gennaro), hannes.fassold@joanneum.at (H. Fassold), giuseppe.amato@isti.cnr.it (G. Amato).

https://doi.org/10.1016/j.neucom.2024.127867
Received 2 September 2023; Received in revised form 6 March 2024; Accepted 13 May 2024
Available online 24 May 2024

For this purpose, we developed a new Hebbian learning solution, named *FastHebb* (FH), which is designed to better take advantage of GPU acceleration. This is done in two steps. First, we notice that Hebbian learning with mini-batch processing evolves in two stages, one is the weight update computation for each sample in the mini-batch, and the other is the aggregation of updates over all the minibatch elements. These two phases can be merged together with a significant speedup. Second, the resulting Hebbian equations of synaptic updates can be translated in terms of matrix multiplications, which can be executed very efficiently on GPU. Specific FH improvements of established Soft-Winner-Takes-All (SWTA) [15,16,20,24–26] and Hebbian Principal Component Analysis (HPCA) [8,11,12,27–29] learning rules are formulated, leading to more performant SWTA-FH and HPCA-FH versions.

In order to provide an experimental evaluation of the proposed method, we used established computer vision benchmarks such as CIFAR10/100 [30], Tiny ImageNet [31] and ImageNet [32]. Besides the image classification evaluation, we also studied the performance of Hebbian neural features for Content-Based Image Retrieval (CBIR). We considered sample efficient learning scenarios, where label information is assumed to be available only for a certain percentage of the data used for training. Results confirm previous observation about the superior performance of Hebbian-based semi-supervised approaches, compared to alternative solutions, especially in label-scarce learning regimes. Moreover, the FastHebb solution exhibits a significant acceleration of training times, both compared to previous Hebbian learning solutions, and compared to backprop-based alternatives. In particular, FastHebb achieves a peak improvement in training up to 70 times faster than corresponding Hebbian approaches not leveraging FastHebb. This allowed to gracefully scale Hebbian algorithms to large datasets, on the scale of ImageNet, and architectures, on the scale of VGG [33]. Extending Hebbian learning to other types of architectures, such as residual networks [34] and transformers [35], in non-trivial and deserves to be explored in a separate work.

Some of the results on FastHebb were already presented in our recent conference publication [36]. However, those results were just preliminary, and the aim of this paper is to significantly extend previous work. Compared to [36], this paper performs a more comprehensive evaluation of the method by considering two types of test scenarios, image classification and CBIR, over four different computer vision benchmarks, including ImageNet. Moreover, we also extend the range of backbone architectures on which the approach is applied, pushing Hebbian learning to VGG-scale architectures. Code is available online.[1]

In summary, our contribution is twofold:

1. A scalable solution for Hebbian synaptic updates is proposed;
2. Extensive evaluation of Hebbian algorithms is presented, including new experiments on large-scale datasets (ImageNet) and architectures (VGG) which (to the best of our knowledge) have been out of reach for Hebbian algorithms so far.

Here is the structure of the following Sections: Section 2 illustrates related contributions; Section 3 describes the proposed FastHebb method more in detail; Section 4 delves into the details of our experimental scenarios in sample efficient and large-scale settings; in Section 5, the results of our experiments are described; conclusive remarks are presented in Section 6.

2. Background and related work

Some past contributions focused on addressing the biological plausibility problem of backpropagation by proposing solutions that can be shown to approximate backprop using Hebbian updates. Contrastive Hebbian Learning (CHL) [37] and Equilibrium Propagation (EP) [6] approaches do so by leveraging recurrent network architectures with two phases of activity. A free phase fixes the values of input neurons to represent a given sample, while output neurons and the remaining hidden units are left free. The recurrent dynamics lead the network to settle down into a steady state, where an anti-Hebbian update is performed. During the forced phase, the activations of output neurons are also fixed to a value closer to the desired target. Again, the recurrent dynamics will induce the network into another steady state, where a Hebbian update is performed. This combination of updates can be shown to approximate backprop at each neuron, using only local information. Another approach with strong biological support is Predictive Coding (PC) [38], in which a layer optimizes a local loss function that accounts for the error in predicting the next layer activations. Optimization is performed in a nested fashion. First, neuron activations are optimized to meet the objective, which leads to the emergence of recurrent interactions among neurons, followed by optimization of the weights. Again, the resulting updates can be shown to match backprop updates using only local information [39]. The PC approach has been successfully applied, in different flavors, to DNN training [40–42]. More recently, the Forward–Forward (FF) approach has been proposed [7]. This is based on standard feedforward architectures, but using an input vector composed of both sample and target. The approach alternates a positive phase, where sample and correct target are provided to the network, which is required to maximize its activations, and a negative phase, where the sample is paired with a randomly generated target, and the network is required to minimize its activations. In a previous work [43], we have used a similar method for training biologically realistic models of *in vitro* cultured neural networks, where sample and target are provided simultaneously to the network, and Hebbian plasticity reinforces the connection between the two, so that at test time, when a sample with no target is provided, the network can recall the association. A weight normalization mechanism plays the role of the negative phase in this case.

In addition to these attempts to model supervised learning from a biologically plausible perspective, other efforts have been focused on modeling bio-inspired unsupervised learning mechanisms. Past works used Hebbian learning with WTA competition models to train feature extractors in feedforward and/or convolutional CNNs [14–20], showing impressive convergence speed. In particular, a recent work [26] showed that soft-WTA training of DNNs allows the network to extract increasingly abstract representations, in the same vein as backprop training, but at the cost of using very wide layers. Indeed, while it is known that Hebbian learning alone is unable to improve performance in deeper networks [44,45], other factors such as network structure (as in SoftHebb [16,26]), or the conditions of label availability for training data [12] can result in interesting scenarios where Hebbian learning is helpful. The SoftHebb method is similar in spirit to FastHebb, as it reformulates Hebbian updates in terms of convolution operations for GPU-efficient computation, while our method leverages matrix multiplications. However, it is not clear whether SoftHebb can be extended also to other types of plasticity models, while FastHebb can be flexibly adapted to a variety of competitive learning and subspace learning principles, as we will show later. The authors also provide experimental results with Hebbian learning on ImageNet, although only for a single training epoch. The method that we propose allows us to run a full training session (20 epochs or more) even on ImageNet scale. Krotov's learning rule [15] is a variant of WTA-based learning, for which a Pytorch-optimized implementation has been provided in [46]. While the initial experimental analysis only focused on fully-connected networks, promising results suggest that there is potential toward extending this method to deeper convolutional architectures. Miconi [44] proposed translations of some Hebbian synaptic update equations into optimizable objective functions, which are more relatable to common frameworks for DL. A similar method based on surrogate losses is adopted in the Hebbian–anti-Hebbian (HaH) network models [47,48], as a mechanism to introduce local learning in DNN layers. The method

[1] https://github.com/GabrieleLagani/HebbianLearning/tree/fasthebb

is applied for training a VGG network model in a hybrid fashion, i.e. lower layers are trained by HaH learning, while higher layers are trained by backpropagation. Experimental analysis shows that the hybrid model is significantly more robust against adversarial noise [49]. In our contribution, we additionally show that FastHebb can scale even to train all layers of a VGG network. Another line of research explored the *Similarity Matching* objective as a possible direction to derive biologically plausible neural models for principal subspace extraction [8–11], with extensions also to the supervised end-to-end recurrent training case [50,51].

Other research on Hebbian learning has focused on different settings, such as domain adaptation [52], cross-modal retrieval [53], metalearning [54,55]. Moreover, recent studies investigated how biologically plausible features could enable the brain to support certain successful deep learning paradigms, such as weight sharing in convolutional layers [56], or attention-based mechanisms in Transformer architectures [57–59]. It has also been shown that networks trained with biologically inspired principles exhibit certain properties of neuronal selectivity observed also in primate brains [60].

In our past contributions, we took a hybrid approach, and explored Hebbian WTA and PCA training of DNNs in semi-supervised scenarios, using unsupervised Hebbian algorithms as a tool for pre-training [12,13,61]. Experiments showed promising results, compared to backprop-based alternative methods, especially in scarce data learning scenarios. Due to the difficulty of gathering manually labeled data, these scenarios are of strong practical interest. Given the promising results obtained in previous works, in this contribution we further enhance previous solutions towards achieving higher efficiency and scalability to more complex scenarios.

Other works have explored semi-supervised approaches exploiting unsupervised pre-training with backprop-based auto-encoding architectures [21,22,62,63]. A different direction towards semi-supervised learning is instead based on pseudo-labeling or consistency-based methods [64–67]. Since our approach belongs to the unsupervised pre-training category, we will focus our comparisons in this setting. However, it is worth mentioning that the other approaches are not mutually exclusive with unsupervised pre-training, and, indeed, these can be integrated together, as also suggested in Section 6

3. Speeding up Hebbian learning with FastHebb

In this Section we provide an overview of the FastHebb method. Compared to our previous work [36], this contribution is significantly extended with novel scenarios of experimental investigation: in addition to standard benchmarks for image recognition, we also consider more applicative scenarios of Content-Based Information Retrieval (CBIR). Moreover, we push our experiments to large-scale backbone network architectures that were prohibitive before, such as VGG [33].

In the following, we start by introducing a convenient notation, that will be used to translate Hebbian synaptic update equations into the FastHebb formulation. Then, we illustrate the learning rules that are analyzed in this work, and we derive their FastHebb-enhanced counterpart.

3.1. A convenient notation

When working with common packages for DL, such as Pytorch, data are typically represented as *tensors*. In this context, a tensor is simply a data array with multiple dimensions. We wish to introduce a notation for tensors that is better suited for relating mathematical formalism with the corresponding implementation in DL packages. For example, a tensor has a number of dimensions, whose interpretation lies in the mind of the programmer (e.g. batch, channel, height, and width dimensions for images). Given a tensor, packages such as Pytorch allow us to transpose or `permute` any of its dimensions, which corresponds to reordering indexes. We can also `unsqueeze` singleton dimensions or `squeeze` them out, which correspond to adding or removing *singleton* indexes, i.e. indexes of dimension 1. Therefore, in essence, the notation that we introduce is motivated by a more straightforward mapping to the programming formalism for working with tensors.

We consider tensors, denoted by capital letters, followed by one index per dimension. The symbol used to denote the index denotes its meaning. For example, we can use an index $p = 1...P$ to represent the *batch* dimension, an index $q = 1...Q$ to represent the *channel* dimension, and index $m = 1...M$ to represent the *weight vector* dimension. A 1 symbol used as an index represents a singleton. For example, a neural feature map at some layer can be denoted as $A_{p,q,1}$, which is a tensor with one element for each mini-batch element (dimension p) and for each neuron (dimension q). The last dimension is a singleton because the output of each neuron for each mini-batch element is a scalar. Similarly, a weight matrix can be denoted by $B_{1,q,m}$, because it does not extend along the mini-batch dimension, but it has a number of channels (one per neuron), and each corresponds to a weight vector. Concerning the input tensors, for example images, they have a mini-batch, channel, height, and width dimensions. However, due to the convolutional processing, a patch will be extracted from each horizontal and vertical location of each image, which is treated, for the Hebbian learning purposes, as a separate input. Therefore, our mini-batch is the collection of all patches extracted from all images. Each patch is flattened into a vector, whose size corresponds to the weight vector size of the next neural layer. On the other hand, this tensor does not extend along the neuron dimension. Overall, our input tensors can be represented as $C_{p,1,m}$.

This notation is convenient, because it allows us to easily swap indexes, or squeeze and unsqueeze singleton dimensions. If tensors have compatible dimensions, we can also perform element-wise operations (additions, multiplications, etc.). When a dimension is a singleton, it automatically undergoes *broadcasting*, i.e. the tensor is replicated along that direction until it matches the corresponding dimension of the other tensor involved in the operation.

Matrix multiplication plays an important role in DNN processing. We make the usage of matrix multiplications explicit in our notation, by writing $\mathrm{bmm}(\cdot,\cdot)$ (which stands for *batch matrix multiplication*):

$$\begin{aligned} Z_{i,j,l} &= \sum_k U_{i,j,k} V_{i,l,k} \\ &= \sum_k U_{i,j,k} V_{i,k,l} := \mathrm{bmm}(U_{i,j,k}, V_{i,k,l}). \end{aligned} \tag{1}$$

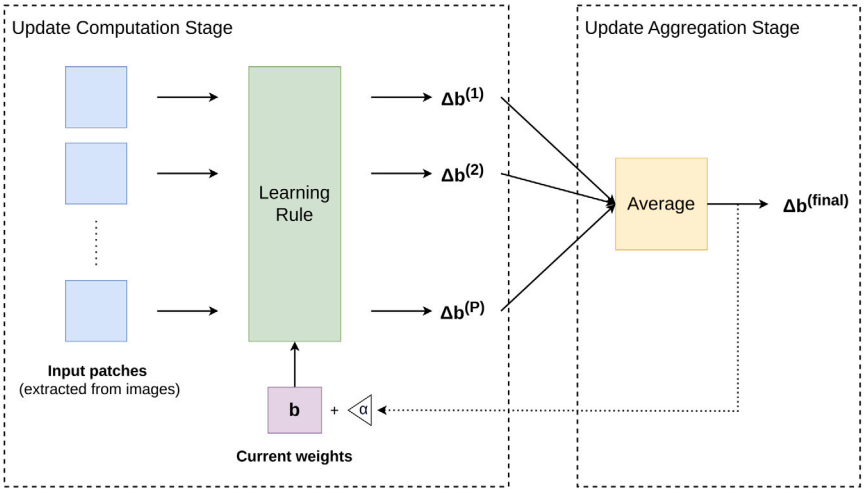
Notice that the matrix multiplication operation is equivalent to taking the element-wise product between tensors U and V, identifying the common index i and summing over (or *contracting*) index k. Specifically, index i represents a batch dimension, and the operation is a batch matrix multiplication between i matrix pairs. For each pair, the two matrices have indices (j,k) and (k,l), which are mapped to indices (j,l): $(j,k) \times (k,l) \rightarrow (j,l)$. The operation generalized to tensors with more that three dimensions as follows: all dimensions except the last two are considered as batch dimensions, while the last two dimensions represent rows and columns of the matrices being batch-multiplied. Batch dimensions must correspond between the two tensors, or be singleton (in which case, broadcasting takes place).

3.2. Hebbian synaptic updates: from computation to aggregation

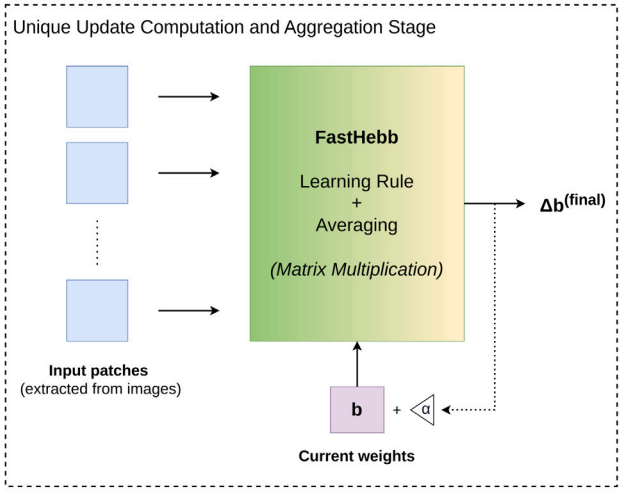
We consider two types of Hebbian learning rules: Hebbian PCA (HPCA) and soft-Winner-Takes-All (SWTA). In this paper, we just give the definition of these learning rules, but the interested reader can find more details in [12,13,19,20].

Given a layer of neurons whose activations are denoted by a_q (index q refers to the qth neuron in the layer), whose weight vectors are denoted by $\mathbf{b}_q$, and whose input vector is denoted by $\mathbf{c}$, the SWTA synaptic update equation is the following:

$$\Delta \mathbf{b}_q = \alpha\, s_q\, (\mathbf{c} - \mathbf{b}_q), \tag{2}$$



(a) Two phases of weight update: update computation for each patch, followed by aggregation of several updates.



(b) In FastHebb, the two phases can be reformulated in a single step, by means of matrix multiplications, which can be performed more efficiently on GPU.

Fig. 1. Phases of weight update computation and aggregation, before and after FastHebb.

where α stands for the learning rate and s_q is the softmax of the activations with temperature T [68]:

$$s_q = \frac{a_q/T}{\sum_k a_k/T}. \tag{3}$$

Essentially, this modulates the update steps so that neurons with higher activations will also 'win' larger updates.

The HPCA learning rule, instead, is the following:

$$\Delta \mathbf{b}_q = \alpha\, a_q\, (\mathbf{c} - \sum_{k=1}^{q} a_k \mathbf{b}_k). \tag{4}$$

This rule can be shown to induce neurons to extract the principal components from data [27,28].

When working with images and convolutional layers, these weight updates need to be computed for each patch extracted from a given image. However, due to the constraints of convolutional layers, neurons at different offsets need to maintain shared weights, hence they are bound to follow the same synaptic modifications. This can be achieved by aggregating the different weight updates, obtained from patches at different locations, into a unique update. Aggregation needs to be performed for all the images in a mini-batch as well. The overall two-phases approach is depicted in Fig. 1(a).

Aggregation is performed simply by averaging, in the HPCA case, or by a weighted average, where the weights are s_q coefficients, for SWTA.

In the following, we show that it is possible to reformulate the update computation and aggregation phases as a single stage, which leverages matrix multiplications for efficient GPU computation. This is depicted in Fig. 1(b).

3.3. From Hebbian synaptic updates to FastHebb

The Hebbian rules presented above can be rewritten in matrix form, including the aggregation step, using the notation outlined at the beginning of this Section:

$$\Delta B_{1,q,m} = \sum_p D_{p,q,1}\, \Delta B_{p,q,m} = \mathrm{bmm}(D_{q,1,p}, \Delta B_{q,p,m}), \tag{5}$$

where $\Delta B_{p,q,m}$ is the collection of all weight updates that need to be aggregated, and $D_{p,q,1}$ is the tensor of coefficients for the aggregation.

Now that the two phases of weight computation and aggregation are merged together. We proceed differently depending on the specific learning rule.

FastHebb for SWTA. Let us rewrite the SWTA update rule as follows:

$$\begin{aligned}\Delta B_{1,q,m} &= \alpha \sum_p D_{p,q,1}\, S_{p,q,1} \left(C_{p,1,m} - B_{1,q,m}\right) \\ &= \alpha \sum_p (D\,S)_{p,q,1}\, (C - B)_{p,q,m} \\ &= \alpha\, \mathrm{bmm}\Big((D\,S)_{q,1,p}, (C - B)_{q,p,m}\Big),\end{aligned}$$

where $D_{p,q,1} = \frac{S_{p,q,1}}{\sum_p S_{p,q,1}}$.

The computational complexity required by this algorithm is $O(P\,Q\,M)$ in time. Moreover, if we wish to exploit GPU parallelism, we need to keep a $P \times Q \times M$ tensor in memory, thus requiring $O(P\,Q\,M)$ space complexity as well, which can be prohibitive for large-scale scenarios.

However, it is possible to improve over these bounds by contracting the aggregation index p (which is typically the largest dimension) early:

$$\begin{aligned}\Delta B_{1,q,m} &= \alpha \sum_p D_{p,q,1}\, S_{p,q,1} \left(C_{p,1,m} - B_{1,q,m}\right) \\ &= \alpha \sum_p (D\,S)_{p,q,1}\, C_{p,1,m} - \alpha \sum_p (D\,S)_{p,q,1}\, B_{1,q,m} \\ &= \alpha\, \mathrm{bmm}\Big((D\,S)_{1,q,m}, C_{1,q,m}\Big) - \alpha \sum_p (D\,S)_{p,q,1}\, B_{1,q,m} \\ &= \alpha\, \mathrm{bmm}\Big((D\,S)_{1,q,p}, C_{1,p,m}\Big) - \alpha\, E_{1,q,1}\, B_{1,q,m},\end{aligned}$$

where $E_{1,q,1} = \sum_p (D\,S)_{p,q,1}$.

This requires only $O(Q\,(P + M))$ space complexity. Concerning the time complexity, this depends on the specific matrix multiplication algorithm adopted, but this can be made lower than $O(P\,Q\,M)$. This is the FastHebb formulation for SWTA.

FastHebb for HPCA. Similarly to the SWTA case, we can rewrite the HPCA equation, together with the aggregation phase (in this case, the coefficient $D_{p,q,1}$ is just $\frac{1}{P}$), with the proposed notation:

$$\begin{aligned}\Delta B_{1,q,m} &= \alpha\, \frac{1}{P} \sum_p A_{p,q,1} \left(C_{p,1,m} - \sum_{q'=1}^{q} A_{p,q',1}\, B_{1,q',m}\right) \\ &= \alpha\, \frac{1}{P} \sum_p A_{p,q,1} \left(C_{p,1,m} - \sum_{q'=1}^{Q} T_{q,q'}\, A_{p,q',1}\, B_{1,q',m}\right) \\ &= \alpha\, \frac{1}{P} \sum_p A_{p,q,1}\, F_{p,q,m} \\ &= \alpha\, \frac{1}{P} \mathrm{bmm}\Big(A_{q,1,p}, F_{q,p,m}\Big),\end{aligned}$$

where $F_{p,q,m} = \left(C_{p,1,m} - \sum_{q'=1}^{Q} T_{q,q'}\, A_{p,q',1}\, B_{1,q',m}\right)$, and $T_{q,q'}$ is simply a lower-triangular matrix with all ones on and below the main diagonal and all zeros above.

The computation of the HPCA equation is slightly more complex, requiring $O(PQ^2M)$ space and time, but this can be improved by reordering the sums:

$$\begin{aligned}\Delta B_{1,q,m} &= \alpha\, \frac{1}{P} \sum_p A_{p,q,1} \left(C_{p,1,m} - \sum_{q'=1}^{Q} T_{q,q'}\, A_{p,q',1}\, B_{1,q',m}\right) \\ &= \alpha\, \frac{1}{P} \sum_p A_{p,q,1}\, C_{p,1,m} \\ &\quad - \alpha\, \frac{1}{P} \sum_p A_{p,q,1} \sum_{q'=1}^{Q} T_{q,q'}\, A_{p,q',1}\, B_{1,q',m} \\ &= \alpha\, \frac{1}{P} \mathrm{bmm}\Big(A_{1,q,p}, C_{1,p,m}\Big) \\ &\quad - \alpha\, \frac{1}{P} \sum_{q'=1}^{Q} \sum_p A_{p,q,1}\, A_{p,q',1}\, T_{q,q'}\, B_{1,q',m} \\ &= \alpha\, \frac{1}{P} \mathrm{bmm}\Big(A_{1,q,p}, C_{1,p,m}\Big) \\ &\quad - \alpha\, \frac{1}{P} \sum_{q'=1}^{Q} \mathrm{bmm}\Big(A_{1,q,p}, A_{1,p,q'}\Big)\, T_{q,q'}\, B_{1,q',m} \\ &= \alpha\, \frac{1}{P} \mathrm{bmm}\Big(A_{1,q,p}, C_{1,p,m}\Big) \\ &\quad - \alpha\, \frac{1}{P} \sum_{q'=1}^{Q} G_{1,q,q'}\, B_{1,q',m} \\ &= \alpha\, \frac{1}{P} \mathrm{bmm}\Big(A_{1,q,p}, C_{1,p,m}\Big) \\ &\quad - \alpha\, \frac{1}{P} \mathrm{bmm}\Big(G_{1,q,q'}, B_{1,q',m}\Big),\end{aligned}$$

where, $G_{1,q,q'} = \mathrm{bmm}\Big(A_{1,q,p}, A_{1,p,q'}\Big)\, T_{q,q'}$.

The overall computation now has $O(Q^2 + QM)$ complexity in space, and up to $O(PQM + PQ^2 + Q^2M)$ in time. This is the FastHebb formulation for HPCA.

3.4. A variety of layers

Even though theoretical developments still need to be address in order to extend Hebbian learning to certain types of layers, such as residual or attention blocks, there is already a variety of neural network modules that can be leveraged with the currently available tools. In particular:

- *Convolutional* [69] can immediately rely on the FH formulation developed above, as it has been specifically designed for this type of processing.
- *Fully-Connected* (FC) [1] layers can be reduced to convolutional layers with kernel size equal to the whole input size, so that learning rules for convolutional layers can be reused in the FC case.
- *BatchNorm* [70] layers are already capable of learning without external feedback, computing running mean and variance parameters from batch statistics. As far as the affine parameters are concerned, this can be left untrained during unsupervised training, and fine-tuned in a successive supervised phase.
- *ReLU* [71] layers, or other activation functions, which do not have learnable parameters, do not require training.
- *Dropout* [72], again, does not have learnable parameters, hence training is not necessary.

4. Evaluation scenario

We evaluated the proposed methodology on a number of established computer vision benchmarks: CIFAR10/100 [30], Tiny ImageNet [31], and ImageNet [32]. We performed an evaluation of Hebbian-based approaches in semi-supervised learning settings, on a backbone network model described in the following, compared to a Variational Auto-Encoder (VAE) [21,23] baseline. In addition, we provide a FastHebb evaluation on VGG [33], to show the scalability of the proposed approach to large architectures. We evaluated the performance both in terms of classification accuracy, and in terms of training speedup achieved with FastHebb. We also provide an evaluation of Hebbian neural features for large-scale image retrieval tasks.

4.1. Neural network backbone for evaluation

In order to provide an evaluation for the proposed approach, we need to define a suitable backbone network architecture for our experiments. For this purpose, we need a network that presents the common architectural features of Convolutional Neural Networks (pooling and convolutional layers [69], batch normalization [70], etc.). On the other hand, we need to exclude more recent features such as residual connections [34] or attention layers [35], for Hebbian algorithms are not trivial to generalize to these cases, which deserve to be analyzed in a separate work. For a first experimentation stage, we do not need to consider a very large model; it is instead preferable to consider a more compact architecture, which enables faster experimentation, and easier analysis of deep features, also on a layer-by-layer basis. It also

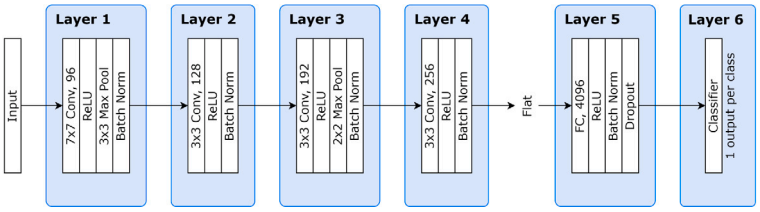


Fig. 2. Backbone neural network model used in our experiments.

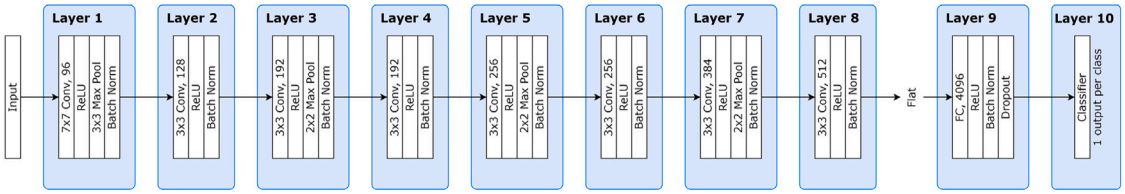


Fig. 3. Bigger neural network model for the ImageNet experiments.

makes reproducibility by other researchers more accessible. Therefore, we opted for an AlexNet-inspired [69] architecture shown in Fig. 2, with 6 layers, which is also consistent with previous works [20,36]. For larger-scale experiments on ImageNet we used an extended version of the previous model with 10 layers, shown in Fig. 3, as well as a VGG model [33].

Previous work also showed thorough analyses of the performance of various Hebbian learning methods, including HPCA and SWTA, varying the number of layers in the backbone architectures. Such results can be found in [12,13]. Note that the same results also apply to the proposed SWTA-FH and HPCA-FH approaches, since these are algebraically equivalent to the above learning rules (although more efficient in the execution).

4.2. Semi-supervised training protocol for sample-efficient learning

Since the proposed approaches are unsupervised learning paradigms, we aim at exploring whether they can be combined together with traditional supervised backprop training, in a semi-supervised fashion. Therefore, we investigate the combination of SWTA-FH and HPCA-FH methods with backprop, where SWTA-FH and HPCA-FH extrapolate knowledge from samples without using labels, while supervised backprop further empowers learned features with information coming from the supervision.

We evaluated the proposed approach assuming a condition of scarcity of available labeled training data. We define a *sample efficiency* regime as the percentage of available labeled samples, over the total number of training samples. For each of the considered datasets, we performed experiments in eight different sample efficiency regimes: 1%, 2%, 3%, 4%, 5%, 10%, 25%, and 100%.

In order to take advantage of both labeled and unlabeled training samples, for each sample efficiency regime, we followed a semi-supervised training protocol in two phases: first, the network is pre-trained using one of the proposed unsupervised Hebbian algorithms, exploiting all the available training samples; second, end-to-end fine-tuning is performed, using supervised backprop training on a cross-entropy loss, and exploiting the labeled samples only. Finally, both the resulting classification accuracy and the training time (in terms of epoch duration, number of epochs, and total duration) were recorded.

Note that, since the final classification layer of the neural network necessarily requires supervision to be trained, it is only trained during the supervised fine-tuning phase. However, since the final classifier has direct access to the error signal, it does not rely on backpropagation, but on direct Stochastic Gradient Descent (SGD) optimization. Instead, during the unsupervised training phase, SWTA-FH and HPCA-FH approaches are applied to train all the backbone architecture layers except the final classifier, without requiring supervision.

As a baseline for comparison, we used unsupervised pre-training based on the Variational Auto-Encoder (VAE) approach [73]. In this case, pre-training was performed by using the deep layers (excluding the final classifier) of the proposed architectures as encoder, mapping their output to 256 gaussian latent variables. This was augmented with a another network branch, acting as decoder, with a specular structure w.r.t. the encoder (i.e. pooling layers replaced with unpooling, and convolutions with transpose convolutions), mapping the latent variables to a decoded sample. The overall models were trained in the encoding–decoding task, optimizing the β-VAE Variational Lower Bound [74], in an end to end fashion, using all the available training samples. At this point, the decoder was dropped, a linear classifier was placed on top of the latent features, and supervised backprop-based end-to-end fine tuning was performed, using only the available labeled samples for the given sample efficiency regime. Notice that, in this case, the pre-training phase, even if unsupervised, is still backprop-based, while Hebbian algorithms enable pre-training without requiring backprop.

4.3. Retrieval with neural features

Deep features extracted from pre-trained networks were also used as vector descriptors for multimedia content indexing and retrieval [75–77]. The performance of the resulting feature representation was evaluated in Content-Based Image Retrieval (CBIR) tasks.

The CBIR systems architecture works as follows: in a first phase, feature representations are computed for all images in a given database, by extracting the deep representations from the convolutional part of the network. These feature representations are then mapped to a binary 256-dimensional descriptor which is then used for indexing the database images. This is done as in [76] by training another piece of network, with a 256 units hidden layer with tanh activations and a final classifier. This is trained in the classification task, so that the feature representation is mapped to the correct class, but passing through a compression stage into the desired 256 dimensional vector. The tanh activation is a "soft" proxy for the binarization operation, which does not block gradients from flowing backward during training. The 256 dimensional representation is then binarized by a thresholding operation: positive values are mapped to 1 and negative values are mapped to 0.

Test set images are used as sample queries: at test time, their 256-dimensional binary feature representation is computed as well,

Table 1
Analysis of algorithm performance on each dataset, for VAE, Hebbian PCA (HPCA), Hebbian PCA with FastHebb (HPCA-FH), soft-WTA (SWTA), and soft-WTA with FastHebb (SWTA-FH) methods, on the 10-layer (for ImageNet) and 6-layer (for the other datasets) networks.

Dataset	Method	Epoch duration	Num. epochs	Total duration
CIFAR10	VAE	14 s	17	3 min 58 s
	SWTA	4 min 14 s	1	4 min 14 s
	SWTA-FH	18 s	1	**18 s**
	HPCA	6 min 23 s	12	1 h 16 min 36 s
	HPCA-FH	19 s	12	3 min 48 s
CIFAR100	VAE	15 s	15	3 min 45 s
	SWTA	4 min 16 s	1	4 min 16 s
	SWTA-FH	18 s	1	**18 s**
	HPCA	6 min 25 s	7	44 min 55 s
	HPCA-FH	19 s	7	2 min 13 s
Tiny ImageNet	VAE	33 s	20	11 min
	SWTA	9 min 41 s	1	9 min 41 s
	SWTA-FH	41 s	1	**41 s**
	HPCA	14 min 20 s	14	3 h 20 min 40 s
	HPCA-FH	43 s	14	10 min 2 s
ImageNet	VAE	2 h 59 min 19 s	16	47 h 49 min 4 s
	SWTA	105 h 13 min 24 s	3	315 h 40 min 12 s
	SWTA-FH	3 h 38 min 6 s	3	**10 h 54 min 18 s**
	HPCA	155 h 41 min 39 s	3	467 h 4 min 57 s
	HPCA-FH	3 h 39 min 18 s	3	10 h 57 min 54 s

Table 2
Accuracy results on each dataset (top-1 for CIFAR10, and top-5 for the other datasets, since they have many more classes), for the various approaches explored, on the 10-layer (for ImageNet) and 6-layer (for the other datasets) networks.

Regime	Method	CIFAR10	CIFAR100	Tiny ImageNet	ImageNet
1%	VAE	22.54	12.28	5.55	2.72
	SWTA-FH	30.23	15.30	6.20	6.69
	HPCA-FH	**39.75**	**22.63**	**11.38**	**8.65**
2%	VAE	26.78	15.25	6.74	6.14
	SWTA-FH	36.59	20.76	8.56	11.52
	HPCA-FH	**45.51**	**30.83**	**15.71**	**13.64**
3%	VAE	29.00	16.44	7.74	15.35
	SWTA-FH	41.54	23.69	10.26	15.67
	HPCA-FH	**48.80**	**35.04**	**18.23**	**17.28**
4%	VAE	31.15	17.89	8.45	**23.97**
	SWTA-FH	45.31	26.91	11.52	19.95
	HPCA-FH	**51.28**	**38.89**	**20.55**	20.39
5%	VAE	32.75	18.48	9.29	**29.04**
	SWTA-FH	48.35	29.57	12.55	24.87
	HPCA-FH	**52.20**	**41.42**	**22.46**	23.28
10%	VAE	45.67	23.80	13.51	**43.73**
	SWTA-FH	**58.00**	38.26	16.70	41.54
	HPCA-FH	57.35	**48.93**	**28.13**	34.27
25%	VAE	68.70	52.59	**37.89**	**61.33**
	SWTA-FH	**69.85**	56.26	24.96	59.34
	HPCA-FH	64.77	**58.70**	37.10	56.92
100%	VAE	85.23	**79.97**	**60.23**	76.84
	SWTA-FH	**85.37**	79.80	54.94	76.10
	HPCA-FH	84.38	74.42	53.96	**77.28**

and the database images are ranked against the query based on the Hamming distance between feature representations. Retrieved images are considered to be a correct match if they belong to the same class as the query.

The evaluation measure used for the CBIR task is the Average Precision Score (APS) :

$$APS = \sum_{i=1}^{K} P_i \left(R_i - R_{i-1}\right) \tag{6}$$

where P_i is the precision at the ith retrieved item, R_i is the corresponding recall. This score is renormalized (so that its maximum value is always 1) and averaged over all the queries, thus obtaining the mean Average Precision (mAP).

4.4. Implementation details

The experiments, implemented in Pytorch, depend on a number of hyperparameters, whose search was pursued by Coordinate Descent (CD) [78], optimizing, for each dataset, the accuracy results of the trained models on the respective validation set. In the following, the resulting parameters and implementation details are illustrated.

All training sessions were performed over 20 epochs (which were enough for the models to converge). Data were processed in mini-batches of 64 samples each, and each sample was an RGB image of 32 pixels in height and width for the 6-layer CIFAR10/100 and Tiny ImageNet network, 210 pixels for the 10-layer ImageNet network, and 224 pixels for VGG (specifically, the VGG-11 model was used), pre-normalized to zero mean and unit variance.

Concerning Hebbian pre-training, the learning rate parameter was set to 10^{-3}. For ImageNet training, we also introduced an adaptive learning rate mechanism to cope with the high variance of weight updates due to the high dimensionality of the feature maps (causing instability during training), which divides the learning rate by the square root of the input size (this corresponds to normalizing the output variance, assuming the inputs are normalized). For SWTA training only, whitening pre-processing was also necessary, as in [30,79], although this step did not show any benefit on the other approaches. SWTA uses 0.02 as inverse temperature parameter $1/T$.

Batch-norm layers used momentum 0.9.

Backprop-based training (i.e. both fine-tuning and VAE pre-training) leveraged Stochastic Gradient Descent (SGD) optimization with learning rate 10^{-3}, and momentum 0.9, with Nesterov acceleration [80]. After 10 training epochs, learning rate was reduced by half every 2 epochs until the end of the training session. The best training epoch in terms of validation results was then selected as final model (early stopping).

β-VAE training used coefficient $\beta = 0.5$.

Supervised fine-tuning was regularized by dropout with 0.5 rate, and L_2 weight decay with penalty equal to $5{\cdot}10^{-2}$, 10^{-2}, $5{\cdot}10^{-3}$, $1{\cdot}10^{-3}$, for CIFAR10, CIFAR100, Tiny ImageNet, ImageNet, respectively.

The implementation used Pytorch version 1.8.1 and Python 3.7, with an Ubuntu 20.4 system running on an I7 series 10700K Intel Processor, 32 GB RAM, and 12 GB NVidia Geforce 3060 GPU.

5. Results

The results of the experiments described in the previous Section are illustrated hereafter. First, we report the recorded training speed, in

Table 3
mAP results on each dataset, for the various approaches explored, on the 10-layer (for ImageNet) and 6-layer (for the other datasets) networks.

Regime	Method	CIFAR10	CIFAR100	Tiny ImageNet	ImageNet
1%	VAE	21.90	6.10	3.18	0.95
	SWTA-FH	28.29	8.16	3.80	1.94
	HPCA-FH	**37.39**	**15.39**	**6.61**	**4.88**
2%	VAE	23.13	6.40	3.20	1.39
	SWTA-FH	31.71	8.82	3.95	2.54
	HPCA-FH	**39.80**	**15.98**	**7.39**	**5.72**
3%	VAE	24.27	6.36	3.52	2.02
	SWTA-FH	34.54	9.13	4.16	3.21
	HPCA-FH	**41.64**	**16.40**	**7.57**	**6.56**
4%	VAE	24.36	6.36	3.57	3.83
	SWTA-FH	36.54	9.41	4.32	3.77
	HPCA-FH	**43.22**	**16.72**	**7.61**	**7.19**
5%	VAE	24.65	6.49	3.57	5.49
	SWTA-FH	38.69	9.66	4.39	4.19
	HPCA-FH	**44.92**	**17.10**	**7.79**	**7.75**
10%	VAE	28.26	7.09	3.76	**13.27**
	SWTA-FH	48.55	11.27	5.01	9.14
	HPCA-FH	**49.99**	**18.42**	**8.45**	10.56
25%	VAE	62.30	13.69	7.59	**24.50**
	SWTA-FH	**63.95**	16.37	6.98	20.75
	HPCA-FH	58.81	**21.49**	**10.13**	21.24
100%	VAE	**84.67**	42.83	**22.64**	**44.21**
	SWTA-FH	84.54	**43.95**	20.86	39.60
	HPCA-FH	81.80	36.19	17.99	43.81

terms of epoch duration, number of epochs for convergence, and total duration, on CIFAR10/100, Tiny ImageNet, and ImageNet datasets, comparing VAE pre-training, ordinary Hebbian learning, and FastHebb. Second, we report the classification and retrieval results of the various approaches in the label-scarcity scenarios described earlier. Third, we provide a performance comparison of different publicly available Hebbian learning implementations on a standard setting, where the epoch duration is measured on a single convolutional layer, while training on CIFAR-10. Finally, we report the results on the VGG, with convolutional layers enhanced by FastHebb pre-training, architecture as well. The results were obtained from averaging five independent experiment iterations, and t-testing confirmed the observed differences to be statistically significant with p-values below 0.05.

5.1. Training speed analysis

Table 1 shows a comparison between the considered approaches in terms of computational performance of training, on the 10-layer (for ImageNet) and 6-layer (for the other datasets) architectures. The Table shows the single epoch duration, the number of epochs until convergence (measured as the point after which validation results stop improving), and the total training duration.[2] These results are specifically focused on the pre-training duration, while we observed no statistically significant difference in the duration of the successive fine-tuning phase for different pre-training approaches.

We can observe that FastHebb methods are significantly faster (up to 50 times for HPCA and HPCA-FH on ImageNet) than the traditional Hebbian counterparts, with an epoch duration becoming comparable to backprop-based VAE training. This enables Hebbian approaches to scale gracefully to complex datasets such as ImageNet, where the best speed-up by a factor of 50, in terms of epoch duration, is observed for HPCA. Moreover, the overall training duration of Hebbian approaches becomes faster (up to 5 times on ImageNet) than VAE, thanks to the lower number of epochs required to convergence. Among the Hebbian approaches, soft-WTA has lower time complexity, and it is in fact faster.

5.2. Label scarcity results

Table 2 illustrates the classification results, in terms of accuracy (top-1 for CIFAR10, and top-5 for the other datasets, since they contain many more classes), in various sample efficiency regimes, comparing the alternative approaches. Notice that in the results for HPCA and SWTA there is no difference between using FastHebb or not. In fact, despite the computational speedup, from the algebraic point of view FastHebb is equivalent to ordinary Hebbian learning, leading to the same results. Therefore, we show these results just once.

The results show that, in conditions of label scarcity (sample efficiency regimes below 4%–5%), Hebbian approaches perform significantly better than VAE. On the other hand, it is only when far more labels are available for the supervised fine-tuning phase that VAE-based pre-training really kicks in. In these scenarios, however, the performance of Hebbian approaches is comparable or only slightly lower, but this is compensated by the speedup in training time observed before. Comparing HPCA and SWTA, it appears that the former performs typically better.

5.3. Retrieval experiments

Table 3 shows the retrieval mAP results obtained on the various datasets, for each of the considered approaches, on the 10-layer (for ImageNet) and 6-layer (for the other datasets) architectures.

This second scenario confirms the previous observations that, in conditions of extreme label scarcity (below 10%), Hebbian-based neural features achieve better results than VAE counterparts. Again, VAE-based pre-training improves in higher regimes, but, as observed before, this is fairly compensated by the training time advantage of Hebbian approaches. Comparing HPCA and SWTA, also in this case it appears that the former performs typically better.

5.4. Comparison with other Hebbian learning implementations

Table 4 provides a performance comparison of various publicly available Hebbian learning implementations. We considered the surrogate loss formulation of Hebbian learning from [44],[3] the Hebbian–anti-Hebbian (HaH) model [47,48],[4] the Pytorch implementation of Krotov's learning rule [15] illustrated in [46],[5] and SoftHebb [26].[6] We also consider the standard HPCA and SWTA learning rules from previous works [12,13,20],[7] and the respective FastHebb formulations from this work.

In order to compare the considered methods on equal footings, we measure their performance in a common scenario, where a single convolutional layer with 5×5 filters, 3 input channels (corresponding to RGB planes), and 96 output channels is considered. The layer is trained using a given Hebbian learning methodology on the CIFAR-10 dataset, and the average epoch duration is measured and reported in the Table. The Table also shows the best results for each method, taken from the respective papers.

From these results, it is possible to observe that FastHebb enables efficient training of convolutional Hebbian layers compared to previous solutions. On the other hand, among the competing approaches, a solution that provides comparable efficiency to FastHebb is represented by SoftHebb [26]. While FastHebb leverages matrix multiplications,

[2] For Hebbian approaches not using FastHebb, the training duration would be unfeasible to measure explicitly; instead, it was estimated by multiplying the single epoch duration by the number of epochs.

[3] https://github.com/ThomasMiconi/HebbianCNNPyTorch
[4] https://github.com/metehancekic/HaH
[5] https://github.com/julestalloen/pytorch-hebbian
[6] https://github.com/NeuromorphicComputing/SoftHebb
[7] https://github.com/GabrieleLagani/HebbianPCA

Table 4
Single-layer training performance (in terms of epoch duration) for different publicly available Hebbian learning repositories, compared to our solution based on FastHebb, on CIFAR-10. Accuracy results of the overall models from the respective papers on CIFAR-10 are also reported.

Method	Epoch duration (single layer)	Acc. (%)
HebbGrad-WTA	8 s	65.55 [44]
HaH	11 s	**87.30** [48]
Pytorch-Hebbian Krotov	9 s	50.75 [15]
SoftHebb	**4 s**	80.30 [26]
SWTA	56 s	85.37 [13]
HPCA	1 min 23 s	84.38 [13]
SWTA-FH (this work)	**4 s**	**85.37**
HPCA-FH (this work)	**4 s**	**84.38**

Table 5
Comparison of ImageNet training times, for Hebbian PCA (HPCA), Hebbian PCA with FastHebb (HPCA-FH), soft-WTA (SWTA), and soft-WTA with FastHebb (SWTA-FH) methods, on VGG-11.

Dataset	Method	Epoch duration	Num. epochs	Total duration
ImageNet	SWTA	290 h 44 min	5	1454 h
	SWTA-FH	7 h 7 min	5	**35 h 35 min**
	HPCA	453 h 32 min	13	5896 h
	HPCA-FH	6 h 25 min	13	83 h 25 min

Table 6
Accuracy results on ImageNet (top-5), and retrieval mean Average Precision (mAP) for the various approaches explored, on the VGG network.

Regime	Pre-train	Accuracy (%)	mAP (%)
1%	None	14.71	4.79
	SWTA-FH	**19.40**	3.63
	HPCA-FH	15.53	**5.16**
2%	None	26.88	6.42
	SWTA-FH	**31.91**	5.18
	HPCA-FH	27.24	**6.70**
3%	None	36.68	**8.05**
	SWTA-FH	**40.24**	7.08
	HPCA-FH	36.74	**8.05**
4%	None	44.01	9.34
	SWTA-FH	**46.98**	8.95
	HPCA-FH	43.69	**9.68**
5%	None	49,37	10.80
	SWTA-FH	**51.36**	10.47
	HPCA-FH	50.40	**10.92**
10%	None	**65.61**	17.44
	SWTA-FH	65.09	17.53
	HPCA-FH	65.49	**17.65**
25%	None	**78.71**	29.01
	SWTA-FH	78.17	**29.52**
	HPCA-FH	78.53	29.20
100%	None	**90.03**	**50.58**
	SWTA-FH	88.00	49.93
	HPCA-FH	88.54	50.06

SoftHebb achieves efficient computation by reformulating part of the Hebbian update as a convolution. However, it is not clear whether this approach, designed for a soft-WTA type of learning principle, can be extended also to other forms of weight updates such as PCA-type plasticity rules. On the other hand, FastHebb is very flexible, and can be adapted to both competitive learning and subspace learning types of plasticity rules, as shown in this work. However, a reformulation of more general Hebbian updates in terms of convolutions could also be an interesting direction of future research (see Section 6).

The HaH approach [47,48] has been deployed in hybrid VGG architectures, where only the bottom layers where trained with Hebbian learning. In our experiments, instead, we were able to apply FastHebb to a VGG model in its entirety, as will be shown in the next subsection.

5.5. Experiments on VGG

In Table 5, we report the training times required for the pre-training phase of VGG models using the different approaches considered so far. We do not consider VAE-type training of the VGG model, because that requires a large decoder, making the overall model very deep, which we found to be untrainable due to vanishing gradients [81,82]. On the other hand, Hebbian pre-training was straightforward to apply in this case, as it requires no gradient backpropagation. Instead, as a baseline for comparison, we used Xavier initialization [83] (note that, since this is not properly a training method, it is not included in Table 5). In fact, it is known that appropriate initialization methods can achieve competitive results compared to end-to-end pre-training [83,84].

Training times show once more the effectiveness of FastHebb methods in training large scale architectures, while using ordinary Hebbian learning would be unfeasible. In the best case, a speedup of almost 70 times is reached, comparing HPCA-FH with HPCA.

Finally, in Table 6, we report the results, both in terms of classification accuracy and retrieval mAP, achieved by training the VGG model in the semi-supervised task. We show the results achieved with Xavier initialization, HPCA pre-training, and SWTA pre-training.

When the scale of the architecture increases, it appears that SWTA approach improves over HPCA. Previous observations about Hebbian methods performing better in low sample efficiency regimes (5% and below) are confirmed. In particular, SWTA outperforms the network with no pre-training by a margin up to 5 percent points in accuracy, in the 1%–2% sample efficiency regimes. In terms of mAP, Hebbian pre-training is still slightly superior, although the difference is not statistically significant.

6. Conclusions and future work

In this article, we have illustrated the FastHebb approach for Hebbian learning, which leverages a matrix multiplication formulation of Hebbian synaptic updates to achieve higher efficiency. This makes Hebbian learning more scalable, enabling the use of Hebbian neural features also on large datasets (ImageNet) and architectures (VGG), which (to the best of our knowledge) have been computationally prohibitive for Hebbian learning so far. Experimental scenarios of label scarcity show promising results of the proposed FH learning rules, namely SWTA-FH and HPCA-FH, for pre-training – compared to backprop-based alternatives such as VAE – considering classification accuracy, retrieval mAP, and training time.

Even though, in this paper, we have shown that it is possible to scale Hebbian training to large models such as VGG, further work needs to be done to adapt Hebbian approaches to more recent architectures, such as residual networks [34] or Transformers [35]. At the current stage, Hebbian learning models still lack a theoretical background to support and guide their application to such types of computation blocks. This represent an interesting open challenge for future research directions. While the FastHebb solution leverages matrix multiplications to achieve efficient GPU computation, recent work suggests that similar improvements could be achieved by directly leveraging convolutions [26]. This could represent an interesting direction of future investigation.

Moreover, additional performance improvements might come from the combination of Hebbian-based pre-training with pseudo-labeling and consistency-based semi-supervised methods [67,85]. Finally, in line with recent efforts towards backprop-free learning (such as the Forward–Forward algorithm [7]), we plan to explore strategies to combine Hebbian approaches with local supervision signals.

CRediT authorship contribution statement

Gabriele Lagani: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Fabrizio Falchi:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Conceptualization. **Claudio Gennaro:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Conceptualization. **Hannes Fassold:** Validation, Supervision, Conceptualization. **Giuseppe Amato:** Writing – review & editing, Writing – original draft, Supervision, Conceptualization, Funding acquisition, Methodology, Project administration, Resources.

Declaration of competing interest

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

Data availability

The authors are unable or have chosen not to specify which data has been used.

Acknowledgments

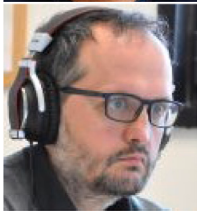
This work was partially supported by:
- Tuscany Health Ecosystem (THE) Project (CUP I53C22000780001), funded by the National Recovery and Resilience Plan (NRPP), within the NextGeneration Europe (NGEU) Program;
- AICult project (code 2022M95RC7), funded by the Italian Ministry for University and Research (MUR), under the PRIN 2022 program;
- AI4Media project, funded by the EC (H2020 - Contract n. 951911);
- INAROS (INtelligenza ARtificiale per il mOnitoraggio e Supporto agli anziani) project co-funded by Tuscany Region POR FSE CUP B53D21008060008.

References

[1] S. Haykin, Neural Networks and Learning Machines, Third ed., Pearson, 2009.
[2] W. Gerstner, W.M. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity, Cambridge University Press, 2002.
[3] C. Gamrat, O. Bichler, D. Roclin, Memristive based device arrays combined with spike based coding can enable efficient implementations of embedded neuromorphic circuits, IEDM, in: 2015 IEEE International Electron Devices Meeting, vol. 2016, Institute of Electrical and Electronics Engineers Inc, 2015, pp. 4–5.
[4] X. Wu, V. Saxena, K. Zhu, S. Balagopal, A CMOS spiking neuron for brain-inspired neural networks with resistive synapses andin SituLearning, IEEE Trans. Circuits Syst. II 62 (11) (2015) 1088–1092.
[5] J.R. Movellan, Contrastive Hebbian learning in the continuous Hopfield model, in: Connectionist Models, Elsevier, 1991, pp. 10–17.
[6] B. Scellier, Y. Bengio, Equilibrium propagation: Bridging the gap between energy-based models and backpropagation, Front. Comput. Neurosci. 11 (2017) 24.
[7] G. Hinton, The forward-forward algorithm: Some preliminary investigations, 2022, arXiv preprint arXiv:2212.13345.
[8] C. Pehlevan, T. Hu, D.B. Chklovskii, A Hebbian/anti-Hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data, Neural Comput. 27 (7) (2015) 1461–1495.
[9] C. Pehlevan, D. Chklovskii, A normative theory of adaptive dimensionality reduction in neural networks, in: Advances in Neural Information Processing Systems, 2015, pp. 2269–2277.
[10] C. Pehlevan, D.B. Chklovskii, Optimization theory of Hebbian/anti-Hebbian networks for pca and whitening, in: 2015 53rd Annual Allerton Conference on Communication, Control, and Computing, Allerton, IEEE, 2015, pp. 1458–1465.
[11] Y. Bahroun, A. Soltoggio, Online representation learning with single and multi-layer Hebbian networks for image classification, in: International Conference on Artificial Neural Networks, Springer, 2017, pp. 354–363.
[12] G. Lagani, F. Falchi, C. Gennaro, G. Amato, Hebbian semi-supervised learning in a sample efficiency setting, Neural Netw. 143 (2021) 719–731.
[13] G. Lagani, F. Falchi, C. Gennaro, G. Amato, Evaluating Hebbian learning in a semi-supervised setting, in: International Conference on Machine Learning, Optimization, and Data Science, Springer, 2021, pp. 365–379.
[14] A. Wadhwa, U. Madhow, Bottom-up deep learning using the Hebbian principle, 2016.
[15] D. Krotov, J.J. Hopfield, Unsupervised learning by competing hidden units, Proc. Natl. Acad. Sci. 116 (16) (2019) 7723–7731.
[16] T. Moraitis, D. Toichkin, Y. Chua, Q. Guo, SoftHebb: Bayesian inference in unsupervised Hebbian soft winner-take-all networks, 2021, arXiv preprint arXiv:2107.05747.
[17] M. Gupta, S.K. Modi, H. Zhang, J.H. Lee, J.H. Lim, Is bio-inspired learning better than backprop? Benchmarking bio learning vs. Backprop, 2022, arXiv preprint arXiv:2212.04614.
[18] G. Amato, F. Carrara, F. Falchi, C. Gennaro, G. Lagani, Hebbian learning meets deep convolutional neural networks, in: International Conference on Image Analysis and Processing, Springer, 2019, pp. 324–334.
[19] G. Lagani, F. Falchi, C. Gennaro, G. Amato, Training convolutional neural networks with competitive Hebbian learning approaches, in: International Conference on Machine Learning, Optimization, and Data Science, Springer, 2021, pp. 25–40.
[20] G. Lagani, F. Falchi, C. Gennaro, G. Amato, Comparing the performance of Hebbian against backpropagation learning using convolutional neural networks, Neural Comput. Appl. 34 (8) (2022) 6503–6519.
[21] D.P. Kingma, S. Mohamed, D. Jimenez Rezende, M. Welling, Semi-supervised learning with deep generative models, in: Advances in Neural Information Processing Systems, vol. 27, 2014, pp. 3581–3589.
[22] Y. Zhang, K. Lee, H. Lee, Augmenting supervised neural networks with unsupervised objectives for large-scale image classification, in: International Conference on Machine Learning, 2016, pp. 612–621.
[23] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[24] S. Grossberg, Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors, Biol. Cybern. 23 (3) (1976) 121–134.
[25] D.E. Rumelhart, D. Zipser, Feature discovery by competitive learning, Cogn. Sci. 9 (1) (1985) 75–112.
[26] A. Journé, H.G. Rodriguez, Q. Guo, T. Moraitis, Hebbian deep learning without feedback, 2022, arXiv preprint arXiv:2209.11883.
[27] J. Karhunen, J. Joutsensalo, Generalizations of principal component analysis, optimization problems, and neural networks, Neural Netw. 8 (4) (1995) 549–562.
[28] S. Becker, M. Plumbley, Unsupervised neural network learning procedures for feature extraction and classification, Appl. Intell. 6 (3) (1996) 185–203.
[29] B. Illing, W. Gerstner, J. Brea, Biologically plausible deep learning—But how far can we go with shallow networks? Neural Netw. 118 (2019) 90–101.
[30] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, 2009.
[31] J. Wu, Q. Zhang, G. Xu, Tiny ImageNet Challenge, Tech. rep., Stanford University, 2017.
[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, Ieee, 2009, pp. 248–255.
[33] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
[34] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[36] G. Lagani, C. Gennaro, H. Fassold, G. Amato, FastHebb: Scaling hebbian training of deep neural networks to ImageNet level, in: International Conference on Similarity Search and Applications, Springer, 2022, pp. 251–264.
[37] X. Xie, H.S. Seung, Equivalence of backpropagation and contrastive Hebbian learning in a layered network, Neural Comput. 15 (2) (2003) 441–454.
[38] R.P. Rao, D.H. Ballard, Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects, Nature Neurosci. 2 (1) (1999) 79–87.
[39] J.C. Whittington, R. Bogacz, An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity, Neural Comput. 29 (5) (2017) 1229–1262.
[40] H. Wen, K. Han, J. Shi, Y. Zhang, E. Culurciello, Z. Liu, Deep predictive coding network for object recognition, 2018, arXiv preprint arXiv:1802.04762.
[41] K. Han, H. Wen, Y. Zhang, D. Fu, E. Culurciello, Z. Liu, Deep predictive coding network with local recurrent processing for object recognition, in: Advances in Neural Information Processing Systems, 2018, pp. 9201–9213.
[42] V. Boutin, A. Franciosini, F. Ruffier, L. Perrinet, Meaningful representations emerge from sparse deep predictive coding, 2019, arXiv preprint arXiv:1902.07651.

[43] G. Lagani, R. Mazziotti, F. Falchi, C. Gennaro, G.M. Cicchini, T. Pizzorusso, F. Cremisi, G. Amato, Assessing pattern recognition performance of neuronal cultures through accurate simulation, in: 2021 10th International IEEE/EMBS Conference on Neural Engineering, NER, IEEE, 2021, pp. 726–729.
[44] T. Miconi, Multi-layer Hebbian networks with modern deep learning frameworks, 2021, arXiv preprint arXiv:2107.01729.
[45] T. Morfeldt Gadler, Comparison of Hebbian learning and backpropagation for image classification in convolutional neural networks, 2023.
[46] J. Talloen, J. Dambre, A. Vandesompele, Pytorch-Hebbian: Facilitating local learning in a deep learning framework, 2021, arXiv preprint arXiv:2102.00428.
[47] M. Cekic, C. Bakiskan, U. Madhow, Towards robust, interpretable neural networks via Hebbian/anti-Hebbian learning: A software framework for training with feature-based costs, Softw. Impacts 13 (2022) 100347.
[48] M. Cekic, C. Bakiskan, U. Madhow, Neuro-inspired deep neural networks with sparse, strong activations, in: 2022 IEEE International Conference on Image Processing, ICIP, IEEE, 2022, pp. 3843–3847.
[49] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015.
[50] D. Obeid, H. Ramambason, C. Pehlevan, Structured and deep similarity matching via structured and deep Hebbian networks, in: Advances in Neural Information Processing Systems, 2019, pp. 15403–15412.
[51] S. Qin, N. Mudur, C. Pehlevan, Supervised deep similarity matching, 2020, arXiv preprint arXiv:2002.10378.
[52] Y. Tang, C. Zhang, H. Xu, S. Chen, J. Cheng, L. Leng, Q. Guo, Z. He, Neuro-modulated Hebbian learning for fully test-time adaptation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 3728–3738.
[53] P. Kaur, A.K. Malhi, H.S. Pannu, Hybrid SOM based cross-modal retrieval exploiting Hebbian learning, Knowl.-Based Syst. (2021) 108014.
[54] T. Miconi, J. Clune, K.O. Stanley, Differentiable plasticity: Training plastic neural networks with backpropagation, ICML 2018, in: 35th International Conference on Machine Learning, vol. 8, 2018, pp. 5728–5739.
[55] T. Munkhdalai, A. Trischler, Metalearning with Hebbian fast weights, 2018, arXiv preprint arXiv:1807.05076.
[56] R. Pogodin, Y. Mehta, T. Lillicrap, P.E. Latham, Towards biologically plausible convolutional networks, Adv. Neural Inf. Process. Syst. 34 (2021) 13924–13936.
[57] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G.K. Sandve, et al., Hopfield networks is all you need, in: ICLR 2021 - 9th International Conference on Learning Representations, 2021.
[58] I.T. Ellwood, Short-term Hebbian learning can implement transformer-like attention, in: bioRxiv, Cold Spring Harbor Laboratory, 2023.
[59] L. Kozachkov, K.V. Kastanenka, D. Krotov, Building transformers from neurons and astrocytes, Proc. Natl. Acad. Sci. 120 (34) (2023) e2219150120.
[60] M.S. Halvagal, F. Zenke, The combination of Hebbian and predictive plasticity learns invariant object representations in deep sensory networks, Nature Neurosci. (2023) 1–10.
[61] G. Lagani, D. Bacciu, C. Gallicchio, F. Falchi, C. Gennaro, G. Amato, Deep features for CBIR with scarce data using Hebbian learning, 2022, arXiv preprint arXiv:2205.08935.
[62] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in Neural Information Processing Systems, 2007, pp. 153–160.
[63] H. Larochelle, Y. Bengio, J. Louradour, P. Lamblin, Exploring strategies for training deep neural networks, J. Mach. Learn. Res. 10 (1) (2009).
[64] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, Q.V. Le, Unsupervised data augmentation for consistency training, 2019, arXiv preprint arXiv:1904.12848.
[65] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, C. Raffel, Mixmatch: A holistic approach to semi-supervised learning, 2019, arXiv preprint arXiv:1905.02249.
[66] D. Berthelot, N. Carlini, E.D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, C. Raffel, Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring, 2019, arXiv preprint arXiv:1911.09785.
[67] P. Sellars, A.I. Aviles-Rivero, C.-B. Schönlieb, LaplaceNet: A hybrid energy-neural model for deep semi-supervised classification, 2021, arXiv preprint arXiv:2106.04527.
[68] B. Gao, L. Pavel, On the properties of the softmax function with application in game theory and reinforcement learning, 2017, arXiv preprint arXiv:1704.00805.
[69] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, vol. 25, 2012, pp. 1097–1105.
[70] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, arXiv preprint arXiv:1502.03167.
[71] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Icml, 2010.
[72] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[73] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, 2013, arXiv preprint arXiv:1312.6114.
[74] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, Beta-vae: Learning basic visual concepts with a constrained variational framework, 2016.
[75] J. Wan, D. Wang, S.C.H. Hoi, P. Wu, J. Zhu, Y. Zhang, J. Li, Deep learning for content-based image retrieval: A comprehensive study, in: Proceedings of the 22nd ACM International Conference on Multimedia, 2014, pp. 157–166.
[76] C. Bai, L. Huang, X. Pan, J. Zheng, S. Chen, Optimization of deep convolutional neural network for large scale image retrieval, Neurocomputing 303 (2018) 60–67.
[77] G. Amato, F. Falchi, C. Gennaro, F. Rabitti, YFCC100M-HNfc6: A large-scale deep features benchmark for similarity search, in: L. Amsaleg, M.E. Houle, E. Schubert (Eds.), Similarity Search and Applications, Springer International Publishing, Cham, 2016, pp. 196–209.
[78] T.G. Kolda, R.M. Lewis, V. Torczon, Optimization by direct search: New perspectives on some classical and modern methods, SIAM Rev. 45 (3) (2003) 385–482.
[79] G. Lagani, 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/.
[80] M. Assran, M. Rabbat, On the convergence of Nesterov's accelerated gradient method in stochastic settings, 2020, arXiv preprint arXiv:2002.12414.
[81] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166.
[82] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International Conference on Machine Learning, PMLR, 2013, pp. 1310–1318.
[83] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
[84] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
[85] A. Iscen, G. Tolias, Y. Avrithis, O. Chum, Label propagation for deep semi-supervised learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5070–5079.

Gabriele Lagani received his Master Degree in Computer Engineering from University of Pisa in May, 2019. At the same University, he completed his Ph.D. in Computer Science in May, 2023. He is currently a research fellow at ISTI-CNR, Pisa. His research interests involve biologically inspired methods for deep learning.



Fabrizio Falchi is researcher of the Artificial Intelligence for Media and Humanities lab of ISTI-CNR. He has a Ph.D. in Information Engineering from University of Pisa (Italy), and a Ph.D. in Informatics from Faculty of Informatics of Masaryk University of Brno (Czech Republic). He also received an M.B.A. from Scuola Superiore Sant'Anna in Pisa. His research interests include deep learning, convolutional neural network, deep features, similarity search, distributed indexes, multimedia information retrieval, computer vision, peer-to-peer systems.

Claudio Gennaro is a first researcher at CNR-ISTI. He received a bachelor's degree in Electronic Engineering from the University of Pisa in 1994 and a master's degree in Information Technology from CEFRIEL in Milan. He received his Ph.D. in Computer and Automatic Engineering in 1999 from the Polytechnic University of Milan. He has published more than 150 papers in international journals and conferences. He is Project Manager for ISTI of the SUN (Social and hUman ceNtered XR) Project funded by the European Commission under the Horizon Europe research and innovation program. He is Principal Investigator of the research project for the analysis, study and implementation of an image search and recognition system containing works of art funded by CY4GATE S.p.A. and of the INAROS ("INtelligence ARtificiale per il mOnitoraggio e Supporto agli anziani") project funded by the Tuscany Region. He has participated in a number of European and national projects including the FP5 ECHO (European Cultural Heritage Online) project, the FP7 SAPIR (Research on audiovisual content using peer-to-peer information retrieval) project, the FIRB NeP4B (Networked Peers for Business) project, and the FP7 Rubicon (Robotic UBIquitous COgnitive Network) project.

Hannes Fassold received a M.Sc. degree in Applied Mathematics from Graz University of Technology in 2004. Since then he works at JOANNEUM RESEARCH, where he is currently a senior researcher at the Machine Vision Applications Group of the DIGITAL institute. His main research interests are how to employ machine vision and AI methods successfully to solve real-world problems like image and video enhancement, defect inspection, object detection and tracking and so on. He is presenting regularly in renowned computer vision, multimedia and AI conferences like ACM Multimedia, ICIP, ICME, AIVR, MVA, MMSP, ASPAI, ISVC, EUSIPCO, GTC etc. Furthermore, he is doing paper reviews for several of these conferences and has been also in the program committee. He coordinates the machine learning workflow as well as the dedicated ML hardware and software infrastructure for the DIGITAL institute.

Giuseppe Amato graduated in Computer Science at the University of Pisa, Italy, in 1992 and was awarded a Ph.D. in Computer Science at the University of Dortmund, Germany, in 2002. Since 1994 he is a member of the research staff at CNR in Pisa, and he is currently a research director, leading the Artificial Intelligence for Multimedia and Humanity (AIMH) laboratory at ISTI-CNR. His main research interests are content based retrieval of multimedia documents, access methods for similarity search of multimedia documents, smart camera networks. He has published in international journals and conferences in the areas of information systems and multimedia information retrieval. He has participated in several EC and national funded research actions in the areas of content based retrieval of multimedia data and artificial intelligence.