Unlocking the Potential of Similarity Matching: Scalability, Supervision and Pre-training

Yanis Bahroun [*1] **Shagesh Sridharan** [*2] **Atithi Acharya** [2 3] **Dmitri Chklovskii** [1] **Anirvan M. Sengupta** [2 3 4]

Abstract

While effective, the backpropagation (BP) algorithm exhibits limitations in terms of biological plausibility, computational cost, and suitability for online learning. As a result, there has been a growing interest in developing alternative biologically plausible learning approaches that rely on local learning rules. This study focuses on the primarily unsupervised similarity matching (SM) framework, which aligns with observed mechanisms in biological systems and offers online, localized, and biologically plausible algorithms. i) To scale SM to large datasets, we propose an implementation of Convolutional Nonnegative SM using PyTorch. ii) We introduce a localized supervised SM objective reminiscent of canonical correlation analysis, facilitating stacking SM layers. iii) We leverage the PyTorch implementation for pre-training architectures such as LeNet and compare the evaluation of features against BP-trained models. This work combines biologically plausible algorithms with computational efficiency opening multiple avenues for further explorations.

1. Introduction

The backpropagation (BP) algorithm (Rumelhart et al., 1986) has proven highly effective; however, its reliance on computing gradients across all network layers raises concerns regarding its biological plausibility. Additionally, BP necessitates extensive computations and information exchange throughout the network, resulting in significant computational expense and energy consumption. Furthermore, the batch-based nature of BP makes it less suitable for online learning scenarios. Consequently, there has been a surge of interest in developing localized and biologically plausible alternatives to the widely adopted BP algorithm in recent years (Sacramento et al., 2018; Belilovsky et al., 2019; Duan & Príncipe, 2022; Golkar et al., 2022).

Biological systems rely on localized computations, demonstrate energy efficiency, and can train online. Therefore, developing algorithms that align with these principles is imperative. Such efforts can yield more efficient and scalable learning processes while facilitating continual learning. In the machine learning literature, localized learning is a specific area of research that explicitly addresses the trade-off between model performance, training speed, and scalability. Localized learning encompasses a class of machine learning methods that update model parameters based on local objectives, focusing on optimizing the performance of individual layers at a time, often referred to as greedy optimization.

In this study, we explore the potential of the similarity matching (SM) framework (Pehlevan & Chklovskii, 2019; Lipshutz et al., 2023) as a promising candidate for localized learning. The SM framework is a normative approach introduced to derive and comprehend the algorithmic basis of neural computation, with a particular emphasis on unsupervised problems. It involves deriving algorithms from computational objectives and evaluating their compatibility with anatomical and physiological observations. The resulting algorithms exhibit desirable properties such as locality, online trainability, and biological plausibility.

Previous studies suggested using nonnegative SM (NSM) for feature learning as individual or stacked layers (Bahroun & Soltoggio, 2017; Bahroun et al., 2017; Obeid et al., 2019). However, these fully unsupervised NSM models demonstrated limited performance when stacked together. To address these limitations, the contrastive NSM (CNSM) was proposed (Qin et al., 2021) to alleviate the issues above. While CNSM employs local learning rules, it lacks layer-wise training. Furthermore, the architecture of the NSM network involves lateral inhibition, which presents challenges in terms of implementation when scaling the model on GPUs and integrating it with convolution.

[*]Equal contribution [1]Center for Computational Neuroscience, Flatiron Institute, New York City, USA [2]Department of Physics and Astronomy, Rutgers University, New Brunswick, New Jersey, USA [3]Center for Computational Quantum Physics, Flatiron Institute, New York City, USA [4]Center for Computational Mathematics, Flatiron Institute, New York City, USA. Correspondence to: Yanis Bahroun <ybahroun@flatironinstitute.org>, Shagesh Sridharan <shagesh.sridharan@rutgers.edu>.

ICML 2023 Localized Learning Workshop, Honolulu, Hawaii, USA.

Our proposed approach involves refining the implementation of deep NSM and adapting it for large-scale datasets. By leveraging this enhanced deep NSM framework, we aim to uncover its potential for localized learning and address the limitations observed in earlier models. Through empirical evaluations and experiments, we demonstrate the effectiveness and scalability of our proposed methodology, shedding light on the capabilities and advantages of deep NSM for addressing complex learning tasks.

Our study makes several key contributions. Firstly, in Section 3, we present the implementation of Convolutional NSM using PyTorch. We employ tensor network methods to compute the neural dynamics and utilize auto differentiation to update the network parameters based on the energy function. This novel implementation enables efficient and scalable computation of Convolutional NSM.

Secondly, in Section 4, we propose a localized supervised objective for SM, which is trained greedily. This objective resembles canonical correlation analysis and allows us to incorporate supervisory signals at different layers of the network. This model can also be adapted as a supervised learning algorithm, leveraging labeled and unlabeled data to enhance performance.

Finally, in Section 5, leveraging the benefits of the PyTorch implementation, we propose pre-training architectures such as LeNet. We compare the learned features when the model is trained using BP. This investigation serves as a first step towards NSM-based pre-training of large-scale models, providing insights into the potential benefits and performance of Convolutional NSM compared to BP-trained models.

Overall, our contributions encompass the development of a PyTorch implementation for Convolutional NSM, introducing a localized supervised objective, and exploring pre-training using Convolutional NSM. Through our novel approaches and evaluations, we advance the understanding of localized learning methods and their potential for enhancing model training, feature learning, and pre-training strategies.

Notations. For positive integers n, m, let $\mathbb{R}^n$ denote n-dimensional Euclidean space, let $\mathbb{R}^n_+$ denote the n-dimensional positive Euclidean space, and let $\mathbb{R}^{n\times m}$ denote the set of $n \times m$ matrices equipped with the Frobenius norm $\|\cdot\|_F$ and the transpose operator $\cdot^\top$. Boldface lowercase letters (e.g., $\mathbf{x}_t$) denote vectors, and boldface uppercase letters (e.g., $\mathbf{M}$) denote matrices. For a set of inputs $\mathbf{x}_t \in \mathbb{R}^n$, $t = 1, \dots, T$, we denote by $\mathbf{X} \in \mathbb{R}^{n\times T}$ the input matrix where the columns of the matrix are the input vectors. Similarly, we define the label matrix $\mathbf{Y} \in \mathbb{R}^{c\times T} = [\mathbf{y}_1, \dots, \mathbf{y}_T]$ where $\mathbf{y}_i$ is the one hot encoding of c classes, and the k^{th} layer encoding matrix $\mathbf{Z}_k \in \mathbb{R}^{m_k\times T} = [\mathbf{z}_{k,1}, \dots, \mathbf{z}_{k,T}]$.

2. Background and related work

To build biologically plausible neural networks (NNs) from a normative approach, the authors of (Hu et al., 2014) considered an objective function reminiscent of classical multidimensional scaling (Cox & Cox, 2000). They departed from the standard reconstruction approach, which had led to the popular Oja's algorithms (Oja, 1989). Indeed, existing models have led to non-local learning rules, which contradict the fundamental Hebbian principle of plasticity, stating that the connections between two neurons are strengthened when activated simultaneously. The objective they considered is the following

$$\hat{\mathbf{Z}} = \underset{\mathbf{Z}\in\mathbb{R}^{m\times T}_+}{\arg\min} \|\mathbf{X}^\top\mathbf{X} - \mathbf{Z}^\top\mathbf{Z}\|_F^2. \tag{1}$$

While the NSM objective (1) can be minimized by taking gradient descent steps with respect to $\mathbf{Z}$, this would not lead to an online algorithm because such computation requires combining data from different time steps. Rather, the authors of (Pehlevan et al., 2017) introduced auxiliary matrix variables, $\mathbf{W}$ and $\mathbf{M}$ allowing for the NSM computation using solely contemporary inputs and will correspond to synaptic weights in the network implementation and rewrite the minimization problem (1) as the following min-max problem

$$\min_{\mathbf{Z}\in\mathbb{R}^{m\times T}_+,\mathbf{W}} \max_{\mathbf{M}} -4\,\mathrm{Tr}(\mathbf{X}^\top\mathbf{W}^\top\mathbf{Z} - \tfrac{1}{2}\mathbf{Z}^\top\mathbf{M}^\top\mathbf{Z}) + 2\,\mathrm{Tr}(\mathbf{W}^\top\mathbf{W}) - \mathrm{Tr}(\mathbf{M}^\top\mathbf{M}). \tag{2}$$

The optimal solution of Eq. (2), $\hat{\mathbf{Z}}$ can be obtained by running the following dynamics until convergence

$$\tfrac{d\mathbf{Z}(\gamma)}{d\gamma} = [\mathbf{W}\mathbf{X} - \mathbf{M}\mathbf{Z}(\gamma)]_+ \ , \tag{3}$$

where $[\cdot]_+$ is the point-wise ReLU. The update rules for the parameters $\mathbf{W}, \mathbf{M}$ admit closed-form solutions as follows

$$\Delta\mathbf{W} = \mathbf{X}\hat{\mathbf{Z}}^\top \ , \ \Delta\mathbf{M} = -\hat{\mathbf{Z}}\hat{\mathbf{Z}}^\top. \tag{4}$$

3. Scaling NSM: PyTorch implementation

To ensure the scalability of the NSM framework, we leverage the powerful capabilities of the PyTorch library. We can significantly enhance computational performance by utilizing tools that enable efficient GPU utilization. Our implementation includes a versatile pipeline that can accommodate various similarity-matching objective functions, as extensively explored in prior publications (Bahroun & Soltoggio, 2017; Sengupta et al., 2018). In particular, our approach is well-suited for loss functions incorporating local error signals, and we effectively approximate the solution using a convolutional architecture. This tailored implementation allows for robust and efficient computations, enabling us to address a wide range of similarity-matching objectives within the NSM framework.

3.1. Implementation

Neural dynamics. Each incoming input image, denoted as $\mathbf{X}$, can be represented as a 3-dimensional tensor with dimensions for channel, width, and height. To process each input, we execute the dynamics described in Eq. 3 until convergence. For efficient and repeated calculations and updates of the tensor $\mathbf{MZ}$, we employ the EinOps library (Rogozhnikov, 2022). EinOps leverages Einstein notations, a mathematical representation for expressing tensor operations, as visualized in Figure 1. This notation allows us to match tensor axes based on labels, such as ν between $\mathbf{M}$ and $\mathbf{Z}$, as depicted in Figure 1. By utilizing EinOps within the PyTorch framework, we can preserve the tensor structure, avoid the need to compute permutations of axes and achieve the computational speedup offered by PyTorch.

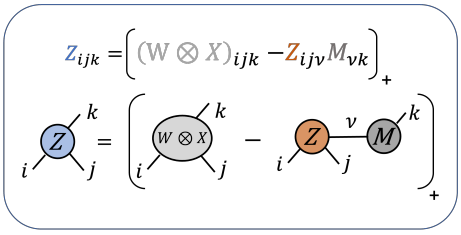

Figure 1. Graphical notation of tensor operations for visualization underlying the efficient computations. Each colored node denotes a tensor, and each outgoing line indicates a dimension of the tensor. We assume implicit summation over indices shared by two tensors, e.g., the index ν shared by $\mathbf{Z}$ and auxiliary variable $\mathbf{M}$ here.

Gradient updates. We evaluate the loss function defined in Eq. 2 using the current values of the auxiliary variables $\mathbf{W}$, $\mathbf{M}$, and the previously computed $\mathbf{Z}$. Next, we utilize the automatic gradients provided by PyTorch to perform updates to the auxiliary variables according to Eq. 4.

3.2. Numerical evaluation

We compare our novel implementation with K-means (Lloyd, 1982) and Manifold tiling (Sengupta et al., 2018) techniques using the CIFAR-10 dataset (Krizhevsky et al., 2009). Our experimental setup follows the methodology outlined in (Bahroun & Soltoggio, 2017). In our approach, we employ a convolutional NSM, where the input is composed of patches represented as $\mathbf{X}$ in Eq. (3). For image classification, we utilize an SVM and employ a simple pooling technique on the feature vectors $\mathbf{Z}$.

By leveraging the computational power of GPUs, our single-layer NN achieves significantly faster convergence to a steady state. To provide a quantitative comparison, we present the training times for processing 10,000 images using different algorithms in Table 1. Specifically, we include the computation times for a conventional implementation executed on a CPU, as well as the CPU and GPU implementations of our novel approach. The conventional implementation extracts patches from images and streams patches to the network in an online fashion after initial processing. PyTorch speeds up this computation using convolution layers, and EinOps enables it to happen on GPUs.

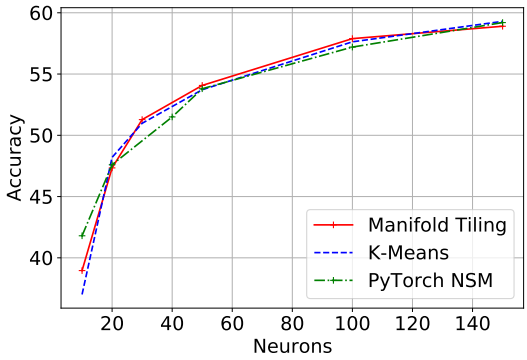

Figure 2. Our algorithm achieves the same accuracy as the solution implemented in (Bahroun & Soltoggio, 2017; Sengupta et al., 2018) despite significantly lower time complexity (c.f. Table 1).

Algorithm	Conventional	CPU[1]	GPU[2]
10k images	2399s	93.85s	13.54s

Table 1. Training times for processing 10,000 images using different algorithms on CPU and GPU.

The PyTorch and conventional implementations yield similar results, accounting for potential variations due to initialization, as shown in Fig. 2. The comparable performance between the two implementations supports the effectiveness and reliability of our novel GPU-accelerated approach, further validating its potential for efficient training of NNs.

4. Including supervision in NSM

In practice, it has been observed that stacking layers of unsupervised learning algorithms offers limited improvements beyond a certain depth (He et al., 2014; Bahroun & Soltoggio, 2017; Amato et al., 2019). Recent studies investigating feature alignment in deep NNs shed light on this phenomenon. It has been demonstrated that as the depth of the layer increases, the alignment between the learned features and the input samples decreases, while the alignment with the corresponding labels becomes inversely related (Canatar & Pehlevan, 2022). In this section, we propose incorporating supervision within the SM framework while preserving the ability to train locally and greedily. By introducing this localized supervision, we aim to address the challenges associated with feature alignment in deep NNs and enhance the performance of the SM framework.

[1]CPU: Intel Xeon Platinum 8358 Processors (48MB cache, 2.60GHz): 3200 MHz DDR4 memory.

[2]GPU: 1xTesla T4 GPU, 2560 CUDA cores, compute 3.7, 15GB GDDR6 VRAM.

4.1. Supervised objective

Armed with the novel implementation of NSM, we propose a new type of supervised SM (S^{2}M) defined as follows for $k \in \{1, L\}$ where L is the number of layers,

$$\hat{\mathbf{Z}}_k = \underset{\mathbf{Z}_k \geq 0}{\arg\min} \left\| \left[\hat{\mathbf{Z}}_{k-1}^\top \hat{\mathbf{Z}}_{k-1} + \alpha_k \mathbf{Y}^\top \mathbf{Y} \right] - \mathbf{Z}_k^\top \mathbf{Z}_k \right\|_F^2, \tag{5}$$

with $\mathbf{Y} \in \mathbb{R}^{c \times T}$, the matrix of labels, with one-hot encoding of c classes and α_k controls the influence of the label matrix. For simplicity, we absorb α_k into $\mathbf{Y}^\top \mathbf{Y}$. The PyTorch implementation of S^{2}M is similar to that of the model defined in the previous section. We introduce auxiliary variables $\mathbf{W}_k$, $\mathbf{M}_k$, and $\mathbf{V}_k$ and obtain the neural activity using EinOps by running the following dynamics until convergence

$$\frac{d\mathbf{Z}_k(\gamma)}{d\gamma} = [\mathbf{W}_k \hat{\mathbf{Z}}_{k-1} + \mathbf{V}_k \mathbf{Y} - \mathbf{M}_k \mathbf{Z}_k(\gamma)]_+ \ . \tag{6}$$

The update rules are implemented using auto differentiation of the following min-max objective

$$\max_{\mathbf{M}_k} \min_{\mathbf{W}_k, \mathbf{V}_k} -4 \operatorname{Tr}\left(\left[\mathbf{Z}_{k-1}^\top \mathbf{W}_k^\top + \mathbf{Y}^\top \mathbf{V}_k^\top - \tfrac{1}{2} \mathbf{Z}_k^\top \mathbf{M}_k^\top \right] \mathbf{Z}_k \right) + 2 \operatorname{Tr}(\mathbf{W}_k^\top \mathbf{W}_k + \mathbf{V}_k^\top \mathbf{V}_k) - \operatorname{Tr}(\mathbf{M}_k^\top \mathbf{M}_k). \tag{7}$$

Each layer takes as input the previous layer's outputs and a scaled version of the label. The model contains a set of L tunable parameters, determining how much we force the neural variables to align with the samples or the labels.

Although we use autograd to compute the gradients, the update rules also admit closed-form solutions as follows

$$\Delta \mathbf{V}_k = \mathbf{Y} \hat{\mathbf{Z}}_k, \Delta \mathbf{W}_k = \hat{\mathbf{Z}}_{k-1} \hat{\mathbf{Z}}_k, \Delta \mathbf{M}_k = -\hat{\mathbf{Z}}_k \hat{\mathbf{Z}}_k^\top. \tag{8}$$

The objective Eq. (5) can also be used for semi-supervised learning, where we can set α_k to 0 for unavailable labels and 1 for available labels, resembling (Genkin et al., 2019).

Other objective functions, which directly consider the notion of similarities for optimizing NNs as in (Pogodin & Latham, 2020; Ma et al., 2020), have been proposed and, as part of the framework, lead to local learning rules or alternatives to BP. These models are nonetheless not greedy. Our model is also reminiscent of the work of Nøkland & Eidnes (2019).

4.2. Numerical evaluation

We evaluated a single layer S^{2}M on CIFAR-10 dataset (Krizhevsky et al., 2009) with different values of $\alpha_1 \in \{10^{-5}, \ldots, 10^1\}$.The results are shown in Fig. 3. We observe that the accuracy reaches its peak at an optimal α_1. This indicates that the model achieves the highest accuracy at an optimal value of α_1 due to a balanced combination of aligning with labels and sample patches crucial for supervised feature learning.

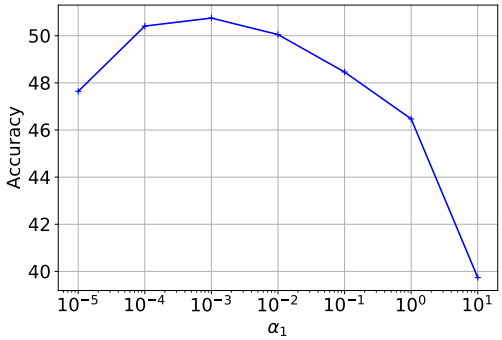

Figure 3. Evaluation of S^{2}M. We show classification on CIFAR10 as a function of α_1 for a single-layer network of 10 neurons.

We present a comparative analysis of the performance of our novel algorithm on the CIFAR-10 dataset with Contrastive Similarity Matching (CSM) variants (Qin et al., 2021) and Equilibrium Propagation (EP) (Scellier & Bengio, 2017). The evaluation of accuracy is presented in Table 2, where logistic regression is employed for S^{2}M, instead of SVM, to ensure a fair and meaningful comparison among the approaches.

Algorithm	Accuracy
CSM-Structured	50.50%
S^{2}M (Logistic Regression)	49.11%
EP beta-positive	42.40%
CSM	40.79%

Table 2. Comparative analysis of the validation accuracy achieved by fully connected networks using Equilibrium Propagation (EP) with positive beta, Contrastive Similarity Matching (CSM), and our novel algorithm on the CIFAR-10 dataset. In all cases, the networks consist of a single hidden layer.

5. Pre-training

The introduction of pre-training of NNs was a paradigm shift for deep learning. Empirical (Erhan et al., 2010) and theoretical works such as for sample complexity (Tripuraneni et al., 2020; Du et al., 2020) or the out-of-distribution risk (Kumar et al., 2022) tried to understand the mechanisms.

Although various candidates for pre-training models exist, we claim that NSM is better suited for the following reasons. Unlike sparse coding models, NSM produces ReLU activation instead of soft thresholding (Fadili & Starck, 2006; Teti et al., 2022). Unlike Non-negative Matrix Factorization (NMF) models (Hoyer, 2004), NSM does not enforce the nonnegativity of the weights. Unlike autoencoders (Goodfellow et al., 2016), NSM does not require decoder stages, reducing by half the number of parameters.

5.1. Experimental setup

Step 1. Pre-training. We initialize a single-layer NSM network with the same number of neurons as the filters in the first layer of LeNet. Subsequently, we train the NSM by executing the dynamics Eq. (6) until convergence is achieved. We then initialize the first layer of LeNet, with the learned weights $\mathbf{W}$ obtained from the NSM, while decoupling $\mathbf{M}$ and $\mathbf{V}$. We initialize the other layers of LeNet randomly.

Step 2. Fine-tuning with BP. We proceed with supervised fine-tuning of the LeNet layer through BP for all layers.

5.2. Numerical evaluation

In this section, we present the numerical evaluation of our approach. First, we examine the results of unsupervised pre-training of a CNN architecture on the MNIST dataset, as shown in Figure 4. To measure the effectiveness of pre-training, we compute the cosine similarity distance between the pre-trained weights and the evolving filters of the CNN during BP training.

We observe that multiple filters exhibit minimal rotation during BP training, indicating their stability and retention of initial orientations from pre-training. This stability suggests that the pre-trained features align well with the final BP-trained features, demonstrating the effective utilization of unsupervised pre-trained knowledge.

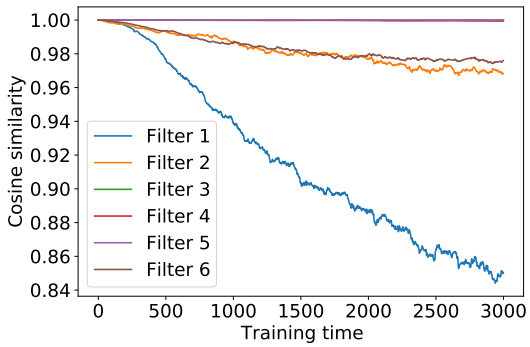

Figure 4. Evaluation of unsupervised NSM used for pre-training of LeNet. We measure the cosine similarity between the weights pre-trained with unsupervised NSM and those at each training step of BP. Three of the six pre-trained filters show minimal rotation during BP training.

Next, in Figure 5, we analyze the median evolution of filters during BP training for different values of α_1, while the shaded regions represent the 25th and 75th percentiles of errors. The median rotation of filters during BP training is observed to be lowest for an optimum α_1. This finding suggests that the filters are most stable and retain their initial orientations from pre-training at the optimum α_1.

In summary, both our unsupervised and supervised pre-training approaches demonstrate the stability of filters and their alignment with the final BP-trained features. These findings provide evidence of the effective utilization of pre-trained knowledge for improving the performance of CNN architectures.

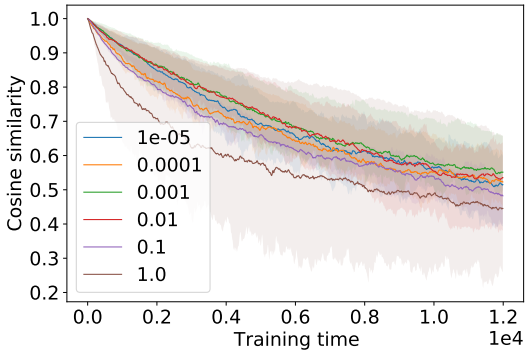

Figure 5. Evaluation of supervised NSM used for pre-training of LeNet. We measure the cosine similarity between the weights pre-trained with NSM and those at each training step of BP for different α_1. The solid line shows the median value for the six filters averaged over 100 trials. The shaded region represents the 25 and 75 error percentiles.

6. Conclusion and future work

This study represents an initial investigation into the potential of NSM as a localized learning alternative to BP. Our contributions encompass various aspects, including the development of a scalable convolutional NSM implementation using PyTorch, the introduction of a localized supervised objective, and the exploration of NSM-based pre-training for models such as LeNet. These proposed models enhance overall model performance and facilitate efficient learning processes. Future work will focus on further exploring and refining the SM framework to uncover its full potential in the realm of localized learning.

7. Acknowledgement

We acknowledge the support of the Simons Foundation, in particular, the grant SF 626323 to AS, which partly funded this research. We thank our Flatiron Institute and Rutgers University colleagues for their valuable feedback. YB would like to thank Abdul Canatar for insightful discussions. Also, we would like to thank Tiberiu Tesileanu for his assistance in creating the Python package pynsm. The package is available at https://pypi.org/project/pynsm/. We recommend it to anyone interested in implementing SM-based algorithms in PyTorch.

References

Amato, G., Carrara, F., Falchi, F., Gennaro, C., and Lagani, G. Hebbian learning meets deep convolutional neural

networks. In *Image Analysis and Processing–ICIAP 2019: 20th International Conference, Trento, Italy, September 9–13, 2019, Proceedings, Part I 20*, pp. 324–334. Springer, 2019.

Bahroun, Y. and Soltoggio, A. Online representation learning with single and multi-layer hebbian networks for image classification. In *Artificial Neural Networks and Machine Learning–ICANN 2017: 26th International Conference on Artificial Neural Networks, Alghero, Italy, September 11-14, 2017, Proceedings, Part I 26*, pp. 354–363. Springer, 2017.

Bahroun, Y., Hunsicker, E., and Soltoggio, A. Building efficient deep hebbian networks for image classification tasks. In *Artificial Neural Networks and Machine Learning–ICANN 2017: 26th International Conference on Artificial Neural Networks, Alghero, Italy, September 11-14, 2017, Proceedings, Part I 26*, pp. 364–372. Springer, 2017.

Belilovsky, E., Eickenberg, M., and Oyallon, E. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pp. 583–593. PMLR, 2019.

Canatar, A. and Pehlevan, C. A kernel analysis of feature learning in deep neural networks. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1–8. IEEE, 2022.

Cox, T. F. and Cox, M. *Multidimensional Scaling*. Chapman and Hall/CRC, 2000. ISBN 1584880945.

Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., and Lei, Q. Few-shot learning via learning the representation, provably. *arXiv preprint arXiv:2002.09434*, 2020.

Duan, S. and Príncipe, J. C. Training deep architectures without end-to-end backpropagation: A survey on the provably optimal methods. *IEEE Computational Intelligence Magazine*, 17(4):39–51, 2022.

Erhan, D., Courville, A., Bengio, Y., and Vincent, P. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 201–208. JMLR Workshop and Conference Proceedings, 2010.

Fadili, J. M. and Starck, J.-L. Sparse representation-based image deconvolution by iterative thresholding. In *Astronomical Data Analysis ADA'06*, 2006.

Genkin, A., Sengupta, A. M., and Chklovskii, D. A neural network for semi-supervised learning on manifolds. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Theoretical Neural Computation: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part I 28*, pp. 375–386. Springer, 2019.

Golkar, S., Tesileanu, T., Bahroun, Y., Sengupta, A., and Chklovskii, D. Constrained predictive coding as a biologically plausible model of the cortical hierarchy. *Advances in Neural Information Processing Systems*, 35: 14155–14169, 2022.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

He, Y., Kavukcuoglu, K., Wang, Y., Szlam, A., and Qi, Y. Unsupervised feature learning by deep sparse coding. In *Proceedings of the 2014 SIAM international conference on data mining*, pp. 902–910. SIAM, 2014.

Hoyer, P. O. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5 (9), 2004.

Hu, T., Pehlevan, C., and Chklovskii, D. B. A Hebbian/anti-Hebbian network for online sparse dictionary learning derived from symmetric matrix factorization. In *2014 48th Asilomar Conference on Signals, Systems and Computers*, pp. 613–619. IEEE, 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kumar, A., Raghunathan, A., Jones, R., Ma, T., and Liang, P. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

Lipshutz, D., Bahroun, Y., Golkar, S., Sengupta, A. M., and Chklovskii, D. B. A normative framework for deriving neural networks with multi-compartmental neurons and non-hebbian plasticity. *arXiv preprint arXiv:2302.10051*, 2023.

Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Ma, W.-D. K., Lewis, J., and Kleijn, W. B. The hsic bottleneck: Deep learning without back-propagation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5085–5092, 2020.

Nøkland, A. and Eidnes, L. H. Training neural networks with local error signals. In *International conference on machine learning*, pp. 4839–4850. PMLR, 2019.

Obeid, D., Ramambason, H., and Pehlevan, C. Structured and deep similarity matching via structured and deep hebbian networks. *Advances in neural information processing systems*, 32, 2019.

Oja, E. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, Vol. 01, No. 01:61–68, 1989.

Pehlevan, C. and Chklovskii, D. B. Neuroscience-inspired online unsupervised learning algorithms: Artificial neural networks. *IEEE Signal Processing Magazine*, 36(6):88–96, 2019.

Pehlevan, C., Sengupta, A. M., and Chklovskii, D. B. Why do similarity matching objectives lead to hebbian/anti-hebbian networks? *Neural computation*, 30(1):84–124, 2017.

Pogodin, R. and Latham, P. Kernelized information bottleneck leads to biologically plausible 3-factor hebbian learning in deep networks. *Advances in Neural Information Processing Systems*, 33:7296–7307, 2020.

Qin, S., Mudur, N., and Pehlevan, C. Contrastive similarity matching for supervised learning. *Neural computation*, 33(5):1300–1328, 2021.

Rogozhnikov, A. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=oapKSVM2bcj.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Sacramento, J., Ponte Costa, R., Bengio, Y., and Senn, W. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31, 2018.

Scellier, B. and Bengio, Y. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

Sengupta, A., Pehlevan, C., Tepper, M., Genkin, A., and Chklovskii, D. Manifold-tiling localized receptive fields are optimal in similarity-preserving neural networks. *Advances in neural information processing systems*, 31, 2018.

Teti, M., Kenyon, G., Migliori, B., and Moore, J. Lcanets: Lateral competition improves robustness against corruption and attack. In *International Conference on Machine Learning*, pp. 21232–21252. PMLR, 2022.

Tripuraneni, N., Jordan, M., and Jin, C. On the theory of transfer learning: The importance of task diversity. *Advances in neural information processing systems*, 33: 7852–7862, 2020.