OPT2025: 17th Annual Workshop on Optimization for Machine Learning

The Hebbian Forward-Forward Algorithm

Andrii Krutsylo ANDRII.KRUTSYLO@GMAIL.COM

Abstract

We introduce Hebbian Forward-Forward (HebbFF), a gradient-free alternative to the Forward-Forward (FF) algorithm. HebbFF replaces the local gradient computations in FF with classical Hebbian plasticity, which is modulated by a gating rule based on the "goodness" introduced in FF. This change eliminates the need for gradient calculations, reducing computational overhead and memory usage. On the MNIST and FashionMNIST datasets, HebbFF achieves the same level of accuracy as FF while training substantially faster and using less VRAM. Although HebbFF achieves lower predictive performance compared to backpropagation, it offers more resource-efficient training. Therefore, HebbFF establishes a stronger baseline than FF for exploring scalable, gradient-free learning in deep networks.

1. Introduction

Training deep neural networks usually depends on backpropagation, a process that updates network weights by recursively computing error gradients through each layer. Despite its success, backpropagation faces practical limitations, driving the exploration of alternative learning paradigms that avoid explicit gradient computations.

Replacing backpropagation would enable learning in settings where it is ineffective or impractical, such as with hardware containing unknown or variable analog nonlinearities, architectures with non-differentiable components, systems that must learn continuously without storing activations, and environments where energy costs or architectural constraints make forward-only training far more practical than full gradient propagation.

The Forward-Forward (FF) algorithm [6] provides one such biologically inspired alternative. Unlike backpropagation, the FF algorithm avoids global gradient propagation. It accomplishes this by independently training each layer using local gradients computed from two forward passes per training step. However, FF still requires local gradient computations, which are memory- and time-demanding.

Building upon these concepts, we introduce Hebbian Forward-Forward (HebbFF), which eliminates the need for gradients by replacing them with the classic Hebbian update [5]. HebbFF computes weight updates as the sum of the outer products of the normalized pre-synaptic inputs and post-synaptic activations. These updates are further simplified as the difference between the aggregated positive and negative sample updates, scaled by a learning rate.

Unlike FF, which still relies on local gradients, HebbFF is entirely gradient-free. HebbFF delivers the same predictive accuracy as FF, but it trains faster and uses substantially less memory. Therefore, HebbFF is not merely an alternative to FF; it is a significant improvement, setting a new standard for exploring scalable, biologically inspired, gradient-free learning rules.

Our findings highlight that integrating simple, biologically inspired mechanisms into modern architectures can provide a powerful, computationally efficient alternative to gradient-based training methods. This approach is particularly well-suited for dynamic environments where resource efficiency is essential.

All code and experiments are available at https://github.com/DentonJC/hebbff.

2. Previous Work

2.1. Hebbian Learning

According to Hebbian learning [5], synaptic strength increases when pre- and post-synaptic neurons co-activate, which is typically formalized as $\Delta w = \eta xy$. Although this rule is simple and biologically grounded, it suffers from instability due to unbounded weight growth. Extensions address this issue. Oja's rule [10] normalizes weights and aligns with principal component learning; the covariance rule [11] subtracts mean activities to handle c orrelated inputs; the BCM rule [1] introduces a dynamic threshold for potentiation and depression; and anti-Hebbian learning [4] supports inhibition and decorrelation.

Recent work adapts Hebbian mechanisms to deep learning. Hebbian-descent [8] stabilizes learning by centering activities and avoiding derivatives. HardWTA [9] integrates competition, inhibition, and BCM dynamics into convolutional models, achieving performance comparable to backpropagation. Neuron-centric Hebbian Learning (NcHL) [3] shifts from synapse-specific to neuron-specific updates, reducing parameters from $\mathcal{O}(W)$ to $\mathcal{O}(N)$ while remaining effective in reinforcement learning tasks.

These approaches show that Hebbian rules, when modified for stability and efficiency, can scale to modern architectures and serve as competitive, biologically inspired alternatives to gradient-based training.

2.2. Local and Biologically-Plausible Learning Rules

Several alternatives to backpropagation aim to preserve the representational power of deep networks while using only local information. Direct Feedback Alignment (DFA) [7] bypasses the weight transport problem with fixed random feedback, enabling each layer to minimize local errors without symmetric weights. Predictive coding networks [13] frame learning as iterative prediction error minimization, with Hebbian-like updates that approximate backprop.

2.3. The Forward-Forward Algorithm

The Forward-Forward (FF) algorithm [6] replaces forward-backward passes with two forward passes: one on positive (real) data and one on negative (synthetic or corrupted) data. Each layer is trained independently using a local *goodness* measure, typically the sum of squared activations. The objective is to increase goodness for positive inputs and decrease it for negative ones. Classification is performed by attaching each possible label to the input and selecting the label with highest cumulative goodness across layers. FF avoids backward error signals and supports non-differentiable components, though at some cost in efficiency and generalization relative to backpropagation.

3. Method

We introduce the Hebbian Forward-Forward (HebbFF) algorithm, a gradient-free learning method derived from the Forward-Forward (FF) framework. HebbFF integrates classical Hebbian plasticity with the FF paradigm while retaining the central idea of separate forward passes for positive and negative samples, each optimizing a local layer-wise objective based on squared activations.

Our implementation departs from the original FF protocol in three main respects. First, all layers are updated concurrently within each epoch: data are passed sequentially through the network, and every layer performs a single update using the same batch of positive and negative samples. In contrast, the original FF trains layers greedily, one at a time, with each layer trained to convergence before the next is introduced. Second, we employ mini-batch updates rather than full-batch updates. This improves memory efficiency and compatibility with modern accelerators but introduces stochasticity that can slow convergence. Third, instead of restricting label information to the input layer, we concatenate a scaled one-hot label vector to the activations at every layer. This design ensures that label information remains explicitly accessible throughout the hierarchy, providing a more persistent supervisory signal and altering the distribution of goodness values compared to the standard FF formulation. The model, training, and evaluation procedures are identical for FF and HebbFF. The only difference is the function that updates the layers using the pure or Hebbian-based FF algorithm.

We adopt the most basic FF, this variant is known to exhibit limitations such as sensitivity to static thresholds and loss function design. However, recent improvements such as the Trifecta framework [2] and the Symmetric Forward-Forward Algorithm [12] mitigate these issues while remaining compatible with HebbFF.

The defining feature of HebbFF is the elimination of gradient computations. Whereas FF calculates gradients from local loss functions, HebbFF applies Hebbian updates directly. Training proceeds as follows:

1. **Label Appending:** For each input batch, construct positive samples with true labels and negative samples with randomly chosen incorrect labels. Concatenate the one-hot label vector to the activations at every layer during both passes.

2. **Sequential Layer-wise Training:**

   (a) **Activation Computation:** For each layer, compute post-synaptic activations a separately for positive and negative samples using rectified linear units (ReLU).

   (b) **Goodness Scores:** For each sample, calculate a scalar goodness measure as the mean squared activation:

   $$g = \frac{1}{n} \sum_{k=1}^{n} a_k^2, \tag{1}$$

   where n is the number of neurons in the layer.

   (c) **Threshold-based Gating:** Assign weighting factors using a sigmoid function σ relative to a tunable threshold θ:

   $$w_{\text{pos}} = \sigma(\theta - g_{\text{pos}}), \quad w_{\text{neg}} = \sigma(g_{\text{neg}} - \theta). \tag{2}$$

Updates are down-weights or up-weights samples when positive samples have low goodness or negative samples have high goodness, ensuring that poorly represented examples drive learning.

(d) **Hebbian Updates:** Update weights using a Hebbian rule that contrasts positive and negative contributions:

$$\Delta W = \eta \left(\sum_{i \in \text{pos}} w_{\text{pos},i} a_i x_i^T - \sum_{j \in \text{neg}} w_{\text{neg},j} a_j x_j^T \right), \tag{3}$$

where η is the learning rate, a are post-synaptic activations, and x are normalized inputs. Updates to biases are done in the same way:

$$\Delta b = \eta \left(\sum_{i \in \text{pos}} w_{\text{pos},i} a_i - \sum_{j \in \text{neg}} w_{\text{neg},j} a_j \right). \tag{4}$$

A small weight decay term may optionally be applied to prevent unbounded parameter growth.

The sigmoid-based gating in HebbFF is directly adapted from the Forward-Forward objective. It assigns weights to updates based on whether positive samples achieve high goodness or negative samples achieve low goodness. This mechanism implements the FF training goal of maximizing positive goodness and minimizing negative goodness rather than acting as a stabilizer for Hebbian plasticity. Instead, stability in HebbFF arises from the subtraction between positive and negative contributions. Further stabilization could be achieved by incorporating classical Hebbian extensions, such as Oja's rule or the BCM rule. These extensions are explicitly designed to constrain weight growth and balance potentiation and depression.

4. Experiments

Experiments are conducted on MNIST and FashionMNIST datasets, with all input images flattened and normalized. We use a randomly selected 20% validation set from the training data for hyperparameters selection. The batch size for training is set to 128. Hyperparameters, such as learning rate and threshold values, are tuned, with learning rates explored in the range $[10^{-6}, 1.0]$ and thresholds between $[10^{-6}, 10]$, identical for all layers. The weight decay is set to zero for all experiments. The SGD optimizer is employed for backpropagation training. We are using two versions of MLP. The deeper MLP consists of layers with 784, 2048, 1024, 512, 256, and 10 units, where 784 corresponds to the flattened input dimension and 10 to the output classes. The wider MLP has layers with 784, 2000, 2000, and 10 units, following the same activation conventions as the deeper model.

As shown in Table 1, deeper and wider MLPs trained with backpropagation consistently achieve the highest accuracies, reaching approximately 97–98% on MNIST and 88% on FashionMNIST. In contrast, both the original Forward-Forward and HebbFF methods yield substantially lower performance, with accuracies of around 89–90% on MNIST and 81% on FashionMNIST. The accuracy differences between FF and HebbFF are small, typically within one standard deviation.

Table 2 shows that backpropagation is the fastest method, requiring between 108–173 seconds depending on architecture and dataset. In contrast, the FF and HebbFF methods are substantially

Model	Dataset	Backprop	FF	HebbFF
Deep	FMNIST	88.03 ± 0.61	80.79 ± 0.64	80.65 ± 0.46
Deep	MNIST	97.62 ± 0.24	89.79 ± 0.43	90.03 ± 0.27
Wide	FMNIST	88.09 ± 0.64	81.20 ± 0.34	80.87 ± 0.32
Wide	MNIST	97.79 ± 0.21	89.41 ± 0.34	89.16 ± 0.51

Table 1: Test accuracy (mean $\pm$ standard deviation) for different methods on MNIST and Fashion-MNIST.

Model	Dataset	Backprop	FF	HebbFF
Deep	FMNIST	166s, 117.9MB	570s, 123.3MB	427s, 71.5MB
Deep	MNIST	108s, 117.9MB	462s, 123.3MB	369s, 71.5MB
Wide	FMNIST	173s, 150.4MB	905s, 164.5MB	626s, 106.2MB
Wide	MNIST	125s, 150.4MB	659s, 164.5MB	331s, 106.2MB

Table 2: The average training time (in seconds) and peak VRAM usage (in MB) for different methods on MNIST and FashionMNIST.

slower, with FF taking the longest time overall (462–905 seconds) and HebbFF performing in between (331–626 seconds). Memory usage follows the opposite trend: HebbFF consistently consumes the least VRAM (71–106 MB), while FF and backpropagation require more, with the wider architecture pushing peak usage above 160 MB.

This highlights a clear trade-off, where backpropagation is time-efficient but more memory-intensive than HebbFF, while FF is both slower and heavier in memory use.

The results highlight a clear trade-off between accuracy, runtime, and memory. Backpropagation remains best for accuracy and speed, but HebbFF dominates FF on the Pareto frontier of resource efficiency. HebbFF matches FF in accuracy yet requires up to 50% less training time and 35–40% less memory, making it a computationally efficient gradient-free method. From a systems perspective, this efficiency makes HebbFF a stronger practical baseline for environments constrained by VRAM or runtime budgets.

5. Conclusion

We presented Hebbian Forward-Forward (HebbFF), a fully gradient-free learning algorithm that integrates Hebbian plasticity into the Forward-Forward framework. Our experiments demonstrate that HebbFF achieves accuracy on par with FF but trains faster and with significantly lower memory usage. This makes HebbFF a strict improvement over FF: it preserves predictive power while providing clear computational and resource advantages. Although both HebbFF and FF lag behind backpropagation in accuracy and training speed, HebbFF represents a more efficient foundation for future work.

References

[1] Elie L Bienenstock, Leon N Cooper, and Paul W Munro. Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2(1):32–48, 1982.

[2] Thomas Dooms, Ing Jyh Tsang, and Jose Oramas. The trifecta: Three simple techniques for training deeper forward-forward networks, 2023. URL https://arxiv.org/abs/2311.18130.

[3] Andrea Ferigo, Elia Cunegatti, and Giovanni Iacca. Neuron-centric hebbian learning, 2024. URL https://arxiv.org/abs/2403.12076.

[4] Péter Földiák. Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, 64(2):165–170, 1990.

[5] Donald O Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, 1949.

[6] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

[7] Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, 2016. doi: 10.1038/ncomms13276.

[8] Jan Melchior and Laurenz Wiskott. Hebbian-descent, 2019. URL https://arxiv.org/abs/1905.10585.

[9] Julian Jimenez Nimmo and Esther Mondragon. Advancing the biological plausibility and efficacy of hebbian convolutional neural networks, 2025. URL https://arxiv.org/abs/2501.17266.

[10] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

[11] Terrence J. Sejnowski. Storing covariance with nonlinearly interacting neurons. *Journal of Mathematical Biology*, 4(4):303–321, 1977.

[12] Erik B. Terres-Escudero, Javier Del Ser, and Pablo Garcia Bringas. A contrastive symmetric forward-forward algorithm (sffa) for continual learning tasks, 2025. URL https://arxiv.org/abs/2409.07387.

[13] James C. R. Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural Computation*, 29(5):1229–1262, 2017. doi: 10.1162/NECO_a_00949.