Learning Dynamics in Competitive Hebbian Learning

Harry Liuson
University of Rochester
hliuson@u.rochester.edu

Abstract

Recent advancements in Hebbian local learning, utilizing winner-takes-all, sparse, and anti-Hebbian methods, have shown promising results in the field of computer vision, particularly when applied to convolutional neural networks. In this study, we examine the dynamics of these algorithms on multilayer perceptrons, uncovering a highly skewed distribution where a small subset of "live" neurons receives the majority of updates. These "live" neurons develop interpretable features, while the remaining "dead" neurons remain indistinguishable from random noise. We contend that relying solely on training accuracy as a success criterion may lead to misleading conclusions, as the effectiveness of these methods could be heavily influenced by the random projections contributed by the "dead" neurons.

1 Introduction

Large, deep neural networks trained with gradient-based optimizers have become the norm across the field of machine learning. As it stands today, there are no serious alternatives to backpropagation for explicit gradient calculation.

Gradient approximation schemes such as synthetic gradients [5], direct feedback alignment [10], and target propagation[11] seek to find ways to approximate the gradient while removing the need to explicitly backpropagate through the entire network at each training step. While some of these methods approach the performance of backpropagation, and may help to overcome its limitations, they are still within the same framework of gradient-based learning.

Hebbian learning is a biologically-inspired learning framework, which represents a dramatic break from gradient-based learning. Weight updates are specified by the rule

$$\Delta w_{ij} \propto x_i y_j$$

where $y = wx$, based on the principle that "neurons that fire together wire together." This immediately gives rise to two large issues. Firstly, this update rule is immediately and obviously unstable. This gives rise to Oja's rule [13]:

$$\Delta w_{ij} \propto y_j(x_i - y_j w_{ij})$$

which can be demonstrated to demonstrated to learn the first principal component of the data in the case of one neuron. The problem, however, is that Oja's rule and similar learning rules tend to only learn one principal component. Various attempts have been made to rectify this issue, such as using a hebbian update rule which explicitly performs a principal component analysis, [9].

Winner-takes-all anti-hebbian learning, allows neurons to "compete" to represent a given feature, or component. [3] [7] Weight updates of neurons which are not maximally activated are zeroed out. This method of hard thresholding is referred to as hard-WTA. Soft winner-takes-all methods use regularizers like a softmax over neuron activations to determine the size of the step taken by each neuron. These methods necessarily induce sparsity in updates, as by definition the majority of weights are being updated either not at all or minimally.

Preprint. Under review.

Related is the notion of anti-hebbian inhibition. This seeks to ensure that the loser of the winner-takes-all competition will not compete with the winning neuron to represent that same feature again. In this case, all but the maximally activated neuron(s) will have their updates negated.

We investigate the learning dynamics of such competitive learning methods. This was motivated by our investigation of alternative Hebbian learning methods. We found that algorithms with very low step sizes tended to preserve the random structure of the input weights. Investigated to what extent existing methods actually learned acros all of their neurons. We re-implement two specific methods, Hebbnet and Softhebb, and introduce a simple method.

We informally identify a dead neuron in Hebbian learning as one whose weights cannot meaningfully be distinguished from random noise, and a live neuron as one which has learned a feature and receives a high amount of the weight updates at each step. Such a distinction emerges due to two features which have been necessary to make Hebbian learning work: competitive learning / sparsity and anti-hebbian inhibition.

For first-layer neurons in an MLP, this can be pretty easily checked by visual inspection. We observe empirically that Softhebb and Hebbnet fail to properly train a high number of well-formed feature detectors, and exhibit a high proportion of dead neurons. We posit that this may be a reason for their high, but still near-random performance on MNIST in very wide networks.

To assist in assessing this imbalance qualitatively, we measure the Gini coefficient of the average step size over the duration of the training process. We measure the Gini coefficient as the mean absolute distance between all points, normalized to the population average. See the appendix for more details.

We introduce a new learning algorithm leveraging soft-WTA, top-1 anti-hebbian inhibition, and sparse updates which helps to exhibit more clearly the failure modes of competitive hebbian learning.

2 Related Work

We use the Softhebb rule as introduced by Moraitis et al. (2021) and Journé et al. (2022) which demonstrates the viability of Soft-WTA Hebbian learning. The SoftHebb rule, instead of explicitly thresholding updates on-off, takes a softmax over each neuron's activation. To remain consistent with earlier notation, we slightly modify the notation as presented in Moraitis et al. Let $u = Wx$. Consider an activation function f, such as ReLU. Let $y = softmax(u)$. We then have our update rule

$$\Delta w_{ij} \propto y_j(x_i - u_j W_{ij})$$

which, under certain assumptions, is proven to minimize cross-entropy with respect to the labels (which are not provided). Softhebb's softmax regularization imposes sparsity and soft-WTA at the same time. Journé et al. (2022) also negate the updates for all but the non-maximally updated neuron, thus introducing anti-hebbian inhibition. [6]

We also use the Hebbnet rule defined as

$$\Delta w_{ij} \propto x_i(y_j - T_j)$$

where T is the average of previous values of y during training. This helps to stabilize weight updates and prevents unstable training behavior. Hebbnet also sorts the activations by magnitude and zeroes out the bottom p percentile. [3] In a sense, Hebbnet is a hard-WTA competitive learning algorithm, as only the maximally activated neurons are updated.

These learning rules, based purely on presynaptic and postsynaptic activity, are called two-factor hebbian rules, because their update rules have two factors. However, to continue the biological analogy, three factor Hebbian learning can incorporate additional information which might realistically be present in a brain. This third factor could encode "reward, punishment, surprise, or novelty" in biological neurons. [2] In artificial neural networks, it could be used to propagate error signals or otherwise help to assign credit in complex networks. [8]

Three factor Hebbian learning algorithms such as the Error Gated Hebbian Rule (EGHR), perform independent component analysis. A recent variant of this rule, EGHR-β, can interpolate between extracting independent components and principal components. [4]

Conscience mechanisms may help to better distribute learning across neurons in the competitive learning process. This is achieved by incorporating a conscience term into the learning algorithm,

which penalizes neurons that consistently win the competition and encourages the involvement of less active neurons. Research on conscience learning has shown that it can modify the competition dynamics such that each neuron can represent an equiprobable subset of the training data. [1] While we do not investigate conscience in this work, it may be a topic worth revisiting in light of recent success in competitive learning.

Krotov et al. (2019) also investigate the distinction between top-k and top-1 anti-hebbian inhibition, and find that as k approaches 1, the features learned converge to "pure" digit prototypes on MNIST. Experimenting with more complex forms of anit-hebbian inhibition may be a promising path forward for learning a diversity of features in competitive Hebbian learning.

3 Proposed Method

In this section, we elaborate on the neural network model and learning process. We consider a network with a hidden layer that processes input x through a weight matrix W and a nonlinearity function f. In our experiments, we do not include biases for simplicity. Given this setup, we can compute the output of the hidden layer as $y = f(Wx)$.

To incorporate competitive dynamics and achieve sparse activations in the hidden layer, we employ thresholding and soft-Winner-Take-All (WTA) mechanisms. At each step, we compute $y^* = \text{softmax}(y - y_{\text{EMA}})$, where y_{EMA} represents the per-neuron exponential moving average (EMA) of activations over previous epochs. This operation ensures that neurons compete against their historical average activation, promoting sparse and specialized activations.

Following the computation of y^*, we introduce sparsification and anti-hebbian inhibition to further refine the network dynamics. To achieve this, we calculate a mask $s(y^*)$ that takes different values depending on the activation level of each neuron. Specifically, the mask is set to 0 for neurons with $y^* > \frac{1}{n}$, where n denotes the number of neurons in the hidden layer. This operation effectively suppresses the influence of minimally activated neurons in the learning process. For maximally activated neurons, the mask is set to 1, ensuring that their role in learning is preserved. Lastly, for all other neurons, the mask takes a value of -1, thereby implementing anti-hebbian inhibition.

With these mechanisms in place, we can define the weight update rule for the network. The update is given by:

$$\Delta w_{ij} \propto s(y^*) y_j^* x_i$$

This update rule ensures that the network learns in a Hebbian manner, while also incorporating the effects of sparsification and anti-hebbian inhibition.

To maintain training stability and avoid issues arising from the absence of the conventional Oja's rule $-Wy$ term, we employ a normalization step after each weight update. Specifically, we divide each neuron's weights by their L2 norm to set the neuron to unit length. This operation can be informally interpreted as a two-phase process analogous to the sleep-wake cycle in biological systems. In the first phase, the network learns features through Hebbian plasticity. In the second phase, the minimally activated weights are decayed, ensuring that only the most relevant connections are preserved.

While we do not conduct a formal ablation study, preliminary investigation seems to show that all components of this method are essential for the learning of diverse features which we will demonstrate in a later section.

4 Methodology

We observe empirically that both Hebbnet and Softhebb struggle to learn more than a small proportion of their neurons, particularly in wider networks. We operationalize this idea by measuring the Gini coefficient of the average step size for various neurons. The Gini coefficient is a method commonly used to measure wealth inequality within a country, but it can be applied to any probability distribution. The Gini coefficient ranges between zero and one with zero measuring total equality.

We chose to log the Gini coefficient for the running average of all updates, because while it is a given that at each step the updates will be highly imbalanced, we want to see whether these imbalanced updates average out over the course of training.

We train single-hidden-layer MLPs with 32, 128, 512, and 2048 hidden units.

In all experiments, we trained the networks for one epoch using a consistent learning rate of 0.05 across all methods, with the exception of CIFAR10 and Softhebb, where we applied a learning rate of 0.01 to enhance training stability. A softmax temperature of 1 was used where applicable. We used a sparsity of 0.8 for Hebbnet. To measure accuracy, we trained a linear probe for 10 epochs and recorded the maximum test accuracy achieved. A batch size of 32 was employed during the training phase.

Our network architecture included layer normalization prior to each fully-connected layer, as well as dropout applied between the hidden layer and the linear head. ReLU activation functions were utilized throughout the network.

In order to address the anomalously low performance of our method on CIFAR10, we also test our method using Triangle activation, which was introduced elsewhere. [12] [6] We do not otherwise measure the impact of specific interventions on the network.

We also forcibly regularized Hebbnet weights to unit norm after each step as we did in our method, as otherwise we found training to be highly unstable. No such regularization was necessary for Softhebb.

Data augmentation techniques were not employed in our experiments.

Despite not conducting an exhaustive exploration of hyperparameters and variations across different Hebbian learning algorithms, our initial investigations indicate that the paradoxical dead neuron phenomenon persists across a wide range of hyperparameters and algorithmic implementations.

5 Results

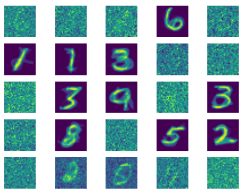
We find that Hebbnet obtains consistently high Gini coefficients (approximately 0.8) and Softhebb obtains consistently low Gini coefficients (approximately 0.2). However, their average updates seem to follow approximately the same bimodal distribution, with one large cluster near zero and another cluster much higher. Due to the explicit sparsity of Hebbnet, the near-zero mode is in fact exactly at zero, whereas the soft-WTA approach of Softhebb places a large mode slightly above zero, leading to a deceptively low Gini coefficient.

In our method and in Softhebb, Gini coefficients seem to rise as the width of the model increases. We do not discern any numerical laws governing these trends. In Hebbnet, we find that the Gini coefficient is more-or-less fixed to 0.8. This corresponds with Hebbnet updating the exact same set of neurons at each timestep (see appendix for more details), with 80% of neurons recieving no updates at all.

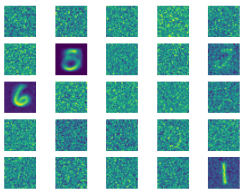
We find however that our method starts with a high Gini coefficient and drives it down over the course of training, a marked difference from the highly static coefficients measured for the aforementioned algorithms. This reflects our observations that our method tends to increase over the course of training the number of neurons which are "active" and have learned interpretable features via Hebbian plasticity. Over the course of one training epoch, in smaller networks it tends to drive the proportion of dead neurons down to a fairly low amount. Thus the interesting part of the measurement of the Gini coefficient is not the actual relative values, but how they change, or don't change, over the course of training.

This capability seems to break down between the 128-unit network and the 512-unit network, as we can see that the change in Gini coefficient falls off extremely quickly. A similar falloff is observed in bouth MNIST and CIFAR10, although CIFAR10 manages to learn a fairly high number of interpretable features even on the 512-hidden unit network. We conjecture that this is due to the increased complexity of CIFAR10, as the number of distinct patterns in MNIST is rather low.

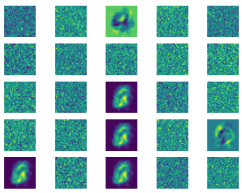
We also observe that the Gini coefficient of our method when training a 32-hidden unit network on CIFAR10 (see Figure 3) dips and then stabilizes to a higher value. We observe that while the Gini



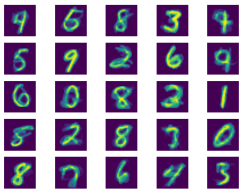
(a) Ours, 100 steps



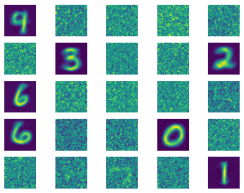
(b) Softhebb, 100 steps



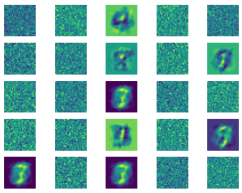
(c) Hebbnet, 100 steps



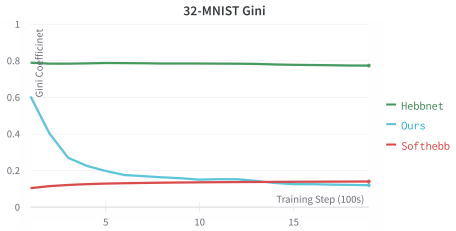
(d) Ours, 1 epoch



(e) Softhebb, 1 epoch



(f) Hebbnet, 1 epoch



(g) Our method activates more neurons over training

(h) Performance correlates with sparsity of active neurons

Figure 1: Experimental results for 32 hidden-unit MLP

Table 1: Accuracy of linear probe after training

	MNIST				CIFAR10			
Hidden Units	32	128	512	2048	32	128	512	2048
Random	67.5	84.4	91.6	84.0	21.9	30.0	34.2	16.3
Ours	52.6	54.8	81.8	93.6	10.0	10.0	10.0	10.1
Softhebb	55.4	77.7	90.6	82.3	19.0	31.8	27.0	21.1
Hebbnet	49.0	82.8	91.4	76.3	21.4	28.0	33.2	28.1
Ours + Triangle	-	-	-	-	22.9	12.4	10.0	10.5

coefficient is lowering rapidly, there are still dead neurons in the network. It's unclear why we see this phenomenon in some places but not others.

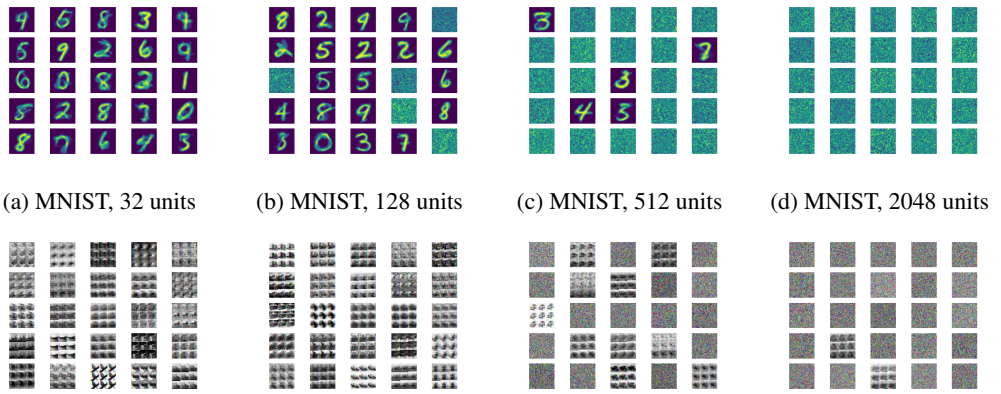
However, despite this increased ability to learn interpretable features, our method performs consistently worse than both the random benchmark and the other methods we compared against. In fact, we find that the only regime in which our method is competitive with Hebbnet and Softhebb is in the case of wider networks where our method also suffers from a high proportion of dead neurons. We note that in the case of 2048 hidden units on MNIST, where our method outperforms all others and achieves above-random performance, our method has a end-of-training Gini coefficient of 98.5, corresponding to almost all training being allocated to a very small subset of neurons. Performance also starts to pick up fairly significantly on MNIST around 512 hidden units, which is also when our method's Gini coefficient begins to stay fixed at a very high level.

It seems, based off of our observed data, that in fact the main regimes in which Hebbian learning methods are competitive are when they have a large amount of dead, uninterpretable neurons. While

Table 2: Gini Coefficients at start and end of training

	MNIST			CIFAR10		
Units	Start	End	Diff.	Start	End	Diff.
			Ours			
32	.61	.12	-.49	.62	.42	-.20
128	.92	.52	-.40	.93	.28	-.65
512	.98	.91	-.07	.99	.78	-.21
2048	1.00	0.90	-.01	1.00	.96	-.04
			Softhebb			
32	.10	.14	+.04	.06	.11	+.05
128	.09	.15	+.06	.05	.12	+.07
512	.12	.17	+.05	.05	.13	+.08
2048	.40	.48	+.08	.08	.14	+.06
			Hebbnet			
32	.79	.73	-.06	.79	.79	-.00
128	.80	.79	-.01	.80	.80	-.00
512	.80	.80	-.00	.80	.80	-.00
2048	.80	.80	-.00	.80	.80	-.00

these dead neurons augmented with a small amount of feature classifiers do sometimes eke out some extra performance over random initialization, it would seem that a high number of dead neurons is a necessary condition for good performance on the tasks tested.



(a) MNIST, 32 units (b) MNIST, 128 units (c) MNIST, 512 units (d) MNIST, 2048 units

(e) CIFAR10, 32 units (f) CIFAR10, 128 units (g) CIFAR10, 512 units (h) CIFAR10, 2048 units

Figure 2: Visualizations of learned features of our method. See Figure 5 for full size example

6 Discussion

In this study, we have presented a novel Hebbian learning algorithm that incorporates competitive dynamics, sparsification, and anti-hebbian inhibition to promote sparse and specialized activations in the hidden layer. We have compared this algorithm against two recently published methods in competitive Hebbian learning.

We have employed the Gini coefficient, a measure commonly used for assessing wealth inequality, to analyze the distribution of average step sizes for various neurons. We observed that our proposed method demonstrates a dynamic Gini coefficient that decreases over the course of training, reflecting an increase in the number of "active" neurons learning interpretable features through Hebbian plasticity. This contrasts with the relatively static Gini coefficients observed for Hebbnet and Softhebb.

While we observe that the Gini coefficient is informative in understanding network dynamics, it does not seem to be a perfect measure of what exactly we were hoping to quantify, which was the number of dead neurons in the network. This raises a limitation in our work, in that it is highly dependent on manual inspection of learned neurons, and is difficult to quantify.

Our method is able to more effectively learn a large number of interpretable features and obtain a high equality of weight updates in relatively narrower networks. However we also note that the features learned by our method tend to be somewhat unstable, a phenomenon which is exhibited by Hebbnet as well, and to a lesser extend Softhebb. By this we mean that the features tend to shift rapidly over the course of training. Further investigation is needed to determine if this is just an artifact of hyperparameters or a deeper underlying issue.

While Hebbnet is able to learn a greater amount of its neurons, it does not learn as well-formed features as the other learning algorithms do, and it seems to always activate the exact same neurons. We note that this may be an artifact of the exponential moving average we used in place of the original paper's average over all training.

Additionally, while we have posited above a theory about the learning dynamics of Hebbnet, we do not posit any particular explanation for the difference in learning dynamics between our method and Softhebb. Understanding this difference is of particular import because these methods seem to learn more well-formed feature detectors.

Further work is necessary to understand the reasons behind the observed learning mechanics in order to develop competitive learning algorithms which can extract a large number of meaningful features in wide neural networks.

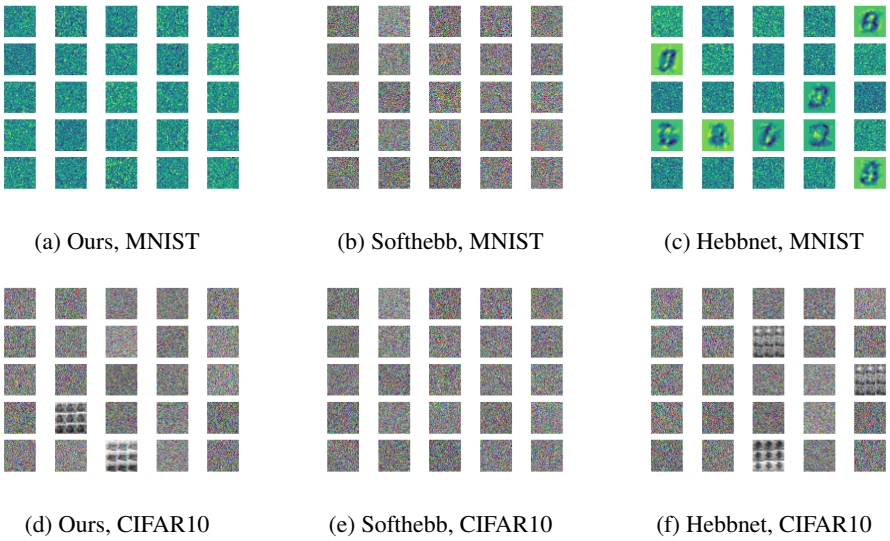
Our empirical results indicate that existing Hebbian learning methods, such as Hebbnet and Softhebb, achieve competitive performance primarily in wider networks where a high proportion of neurons maintain their randomly-initialized weights, leading to a large number of dead, uninterpretable neurons.

Interestingly, our method appears to be competitive with Hebbnet and Softhebb in scenarios where it also experiences a high proportion of dead neurons, suggesting that the presence of these uninterpretable neurons may be a necessary condition for achieving good performance with Hebbian learning algorithms on the tasks tested. This observation raises questions about the underlying principles governing the success of these learning methods as they scale to wider networks.

The use of triangle activation boosts performance of our method to marginally above-random. Analysis of its training dynamics also suggests that it does in fact learn meaningful feature detectors. We conjecture that Triangle activation assists in Hebbian based methods because many of them omit learned biases. This, alongside promising work elsewhere, suggests that while Hebbian learning has struggled as a field, that learned Hebbian weights can still extract useful features, even in MLPs, which are not particularly well-suited to computer vision tasks.

Our work also does not address the learning dynamics of these methods on Convolutional Neural Networks, and we also only address performance on a limited set of computer vision dataset.

Figure 3: Gini coefficient over training on CIFAR10 on 32 hidden unit network. We find a large dip and then raise in Gini coefficient.



(a) Ours, MNIST (b) Softhebb, MNIST (c) Hebbnet, MNIST

(d) Ours, CIFAR10 (e) Softhebb, CIFAR10 (f) Hebbnet, CIFAR10

Figure 4: Visualizations of learned features on network with 2048 hidden units. Features are highly sparse in competitive learning algorithms.

7 Conclusion

In this study, we conducted a comparative analysis of two recent competitive Hebbian learning algorithms and one algorithm of our own design. We underscored that the methods we tested, Softhebb and Hebbnet, exhibit highly imbalanced weight updates. We introduce a method incorporating soft WTA competitive learning, anti-hebbian inhibition, and update sparsity, which circumvents this in narrow networks but exhibits a similar failure mode in wider networks.

We additionally highlighted a misleading phenomenon where performance on tested benchmarks of wide, shallow MLPs on simple computer vision tasks does not necessarily indicate the learning of meaningful features. Instead, these networks achieve high performance by having a high proportion of effectively random weights. Their performance is, in fact, closely tied to that of randomly-initialized weights.

This finding emphasizes the importance of understanding the underlying mechanisms and learning dynamics of such algorithms, as they may not achieve high performance for the desired or anticipated reasons. We underscore the need for more sophisticated competitive learning algorithms which are able to effectively learn a diversity of features.

Further research should address this issue by more rigorously investigating the correspondence between performance on benchmarks and the learning of genuinely meaningful features. Future work could focus on the development of better benchmarks and metrics that rely less on human evaluation and more effectively capture the true learning capabilities of Hebbian learning algorithms. This is a challenging endeavor, as the definition of "meaningful" is somewhat subjective and difficult to quantify. Nonetheless, overcoming these challenges could pave the way for more robust and insightful evaluations of learning algorithms and their potential applications.

References

[1] Duane DeSieno. Adding a conscience to competitive learning. In *ICNN*, volume 1, 1988.

[2] Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Frontiers in neural circuits*, 12:53, 2018.

[3] Manas Gupta, ArulMurugan Ambikapathi, and Savitha Ramasamy. Hebbnet: A simplified hebbian learning framework to do biologically plausible learning. In *ICASSP 2021 - 2021*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3115–3119, 2021.

[4] Takuya Isomura and Taro Toyoizumi. Error-gated hebbian rule: A local learning rule for principal and independent component analysis. *Scientific reports*, 8(1):1835, 2018.

[5] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *International conference on machine learning*, pages 1627–1635. PMLR, 2017.

[6] Adrien Journé, Hector Garcia Rodriguez, Qinghai Guo, and Timoleon Moraitis. Hebbian deep learning without feedback. arXiv, 2022.

[7] Dmitry Krotov and John J Hopfield. Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences*, 116(16):7723–7731, 2019.

[8] Łukasz Kuśmierz, Takuya Isomura, and Taro Toyoizumi. Learning with three factors: modulating hebbian plasticity with errors. *Current opinion in neurobiology*, 46:170–177, 2017.

[9] Gabriele Lagani, Giuseppe Amato, Fabrizio Falchi, and Claudio Gennaro. Training convolutional neural networks with hebbian principal component analysis. arXiv, 2020.

[10] Julien Launay, Iacopo Poli, François Boniface, and Florent Krzakala. Direct feedback alignment scales to modern deep learning tasks and architectures. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9346–9360. Curran Associates, Inc., 2020.

[11] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15*, pages 498–515. Springer, 2015.

[12] Thomas Miconi. Hebbian learning with gradients: Hebbian convolutional neural networks with modern deep learning frameworks. *arXiv preprint arXiv:2107.01729*, 2021.

[13] Erkki Oja. Simplified neuron model as a principal component analyzer. 1982.

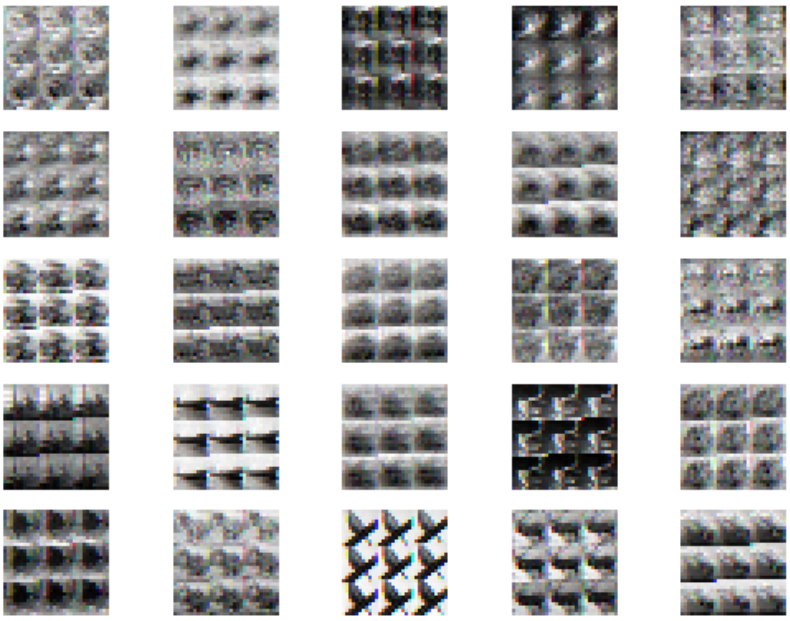


Figure 5: Learned weights by our method on CIFAR-10 with 32 hidden units, full size. Some shapes such as ships, birds, and horses can be discerned.

A Gini Coefficient

The Gini coefficient of a set of n samples x can be evaluated as $\frac{\sum_{i,j=1}^{n} \|x_i - x_j\|}{n^2 \bar{x}}$, where $\bar{x} = \frac{1}{n} \sum_i^n x_i$, the mean of the set.

We observe that if p of the population has value zero and $1 - p$ of the population has some constant value c, then:

$$\frac{\sum_{i,j=1}^{n} \|x_i - x_j\|}{2n^2 \bar{x}} = \frac{2cp(1-p)n^2}{2n^2 c(1-p)} = p$$