LEARNING REPRESENTATIONS FOR BINARY-CLASSIFICATION WITHOUT BACKPROPAGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The family of feedback alignment (FA) algorithms aims to provide a more biologically motivated alternative to backpropagation (BP), by substituting the computations that are unrealistic to be implemented in physical brains. While FA algorithms have been shown to work well in practice, there is a lack of rigorous theory proofing their learning capabilities. Here we introduce the first feedback alignment algorithm with provable learning guarantees. In contrast to existing work, we do not require any assumption about the size or depth of the network except that it has a single output neuron, i.e., such as for binary classification tasks. We show that our FA algorithm can deliver its theoretical promises in practice, surpassing the learning performance of existing FA methods and matching backpropagation in binary classification tasks. Finally, we demonstrate the limits of our FA variant when the number of output neurons grows beyond a certain quantity.

1 INTRODUCTION

A key factor enabling the successes of Deep Learning is the backpropagation of error (BP) algorithm (Rumelhart et al., 1986). Since it has been introduced, BP has sparked several discussions on whether physical brains are realizing BP-like learning or not (Grossberg, 1987; Crick, 1989). Today, most researchers consent that two distinct characteristics of BP render the idea of a BP based learning in brains as implausible: 1) The usage of symmetric forward and backward connections and 2) the strict separation of activity and error propagation (Bartunov et al., 2018). These two objections have lead researchers to search for more biologically motivated alternatives to BP.

The three most influential families of BP alternatives distilled so far are Contrastive Hebbian Learing (CHL) (Movellan, 1991), target-propagation (TP) (LeCun, 1986; Hinton, 2007; Bengio, 2014) and feedback Alignment (FA) (Lillicrap et al., 2014).

The idea of CHL is to propagate the target activities, instead of the errors, backward through the network. For this reason, a temporal dimension is added to the neuron activities. Each neuron then adapts its parameters based on the temporal differences of its "forward" and "backward" activity. The two significant critic points of CHL are the requirement for symmetric "forward-backward" connections and the use of alternating "forward" and "backward" phases (Baldi & Pineda, 1991; Bartunov et al., 2018).

TP shares the idea with Contrastive Hebbian Learning of propagating target activities instead of errors. However, rather than keeping symmetric forward and backward paths, the reciprocal propagation of the activities are realized through learned connections. Consequently, each layer has assigned two objectives: Learning the inverse of the layer's forward function and minimizing the difference to the back-projected target activity. Variants of TP differ in how exactly the target activity is projected backward (LeCun, 1986; Bengio, 2014; Bartunov et al., 2018). Theoretical guarantees of TP rely on the assumption that each reciprocal connection implements the perfect inverse of the corresponding forward function. This issue of an imperfect inverse was also found to be the "bottleneck" of TP in practice (Bartunov et al., 2018). When the output of a layer has a significant lower dimension than its input, reconstructing the input from the output becomes challenging, resulting in poor learning performance.

Feedback alignment algorithms eliminate the weight sharing implausibility of BP by replacing the symmetric weights in the error backpropagation path by random matrices. The second objection, i.e., separate activity and error channels, is attenuated by Direct Feedback Alignment (Nøkland, 2016) which drastically reduces the number of channels carrying an error signal. While feedback alignment algorithms work well on small and medium-sized benchmarks, a recent study identified that they are unable to provide learning on more challenging datasets like ImageNet (Bartunov et al., 2018). Another criticism of FA algorithms is the lack of rigorous mathematical justification and convergence guarantees of the performed computations.

In this work, we investigate feed-forward networks where the weights of all, expect the first, layers are constrained to positive values. We prove that this constraint does not invalidate the universal approximation capabilities of neural networks. Next, we show that, in combination with monotonic activation functions, all layers from the second layer on realize monotonically increasing functions. The backpropagation of a scalar error signal through these layers only affects the magnitude of the error signal but does not change its sign. Consequently, we prove that algorithms that bypass the error backpropagation steps, such as Direct Feedback Alignment, can compute the sign of the true gradient with respect to the weights of our constraint networks without the need for backpropagation. Finally, we show that our algorithm, which we call monotone Direct Feedback Alignment, can deliver its theoretical promises in practice, by surpassing the learning performance of existing feedback alignment algorithms in binary classification task, i.e., when the error signal is scalar, and provide decent performance even when the error signal is not scalar.

We make the following key contributions:

- First FA algorithm that has provable learning capabilities for non-linear networks of arbitrary depth
- Experimental evaluation showing that our FA algorithm outperforms the learning performance of existing FA algorithms and match backpropagation in binary classification tasks
- We make an efficient TensorFlow implementation of all tested algorithms publicly available

2 Backpropagation and Feedback Alignment

We consider the feed-forward neural network

$$\begin{aligned} h_l(h_{l-1}) &:= \begin{cases} f(W_l h_{l-1} + b_l) & \text{if } l < L \\ W_l h_{l-1} + b_l & \text{if } l = L \end{cases} \\ h_0 &:= x \\ W_l &\in \mathbb{R}^{n_l \times n_{l-1}}, \\ b_l &\in \mathbb{R}^{n_l} \end{aligned} \tag{1}$$

where f is the non-linear activation function, x the input and h_L the output of the network. For classification tasks, h_L is usually transformed into a probability distribution with discrete support by a sigmoid or softmax function.

During training, the parameters $W_l, b_l, l = 1, \dots L$ are adjusted to minimize a loss function $\mathcal{L}(y, h_L)$ on samples of a giving training distribution $p(y, x)$. This is usually done by performing *gradient descent*

$$\theta_l \leftarrow \theta_l - \alpha \frac{d\mathcal{L}}{d\theta}, \qquad \alpha \in \mathbb{R}_+ \tag{2}$$

with respect to the parameters $\theta_l \in \{W_l, b_l\}, 1 \leq l \leq n$ of the network.

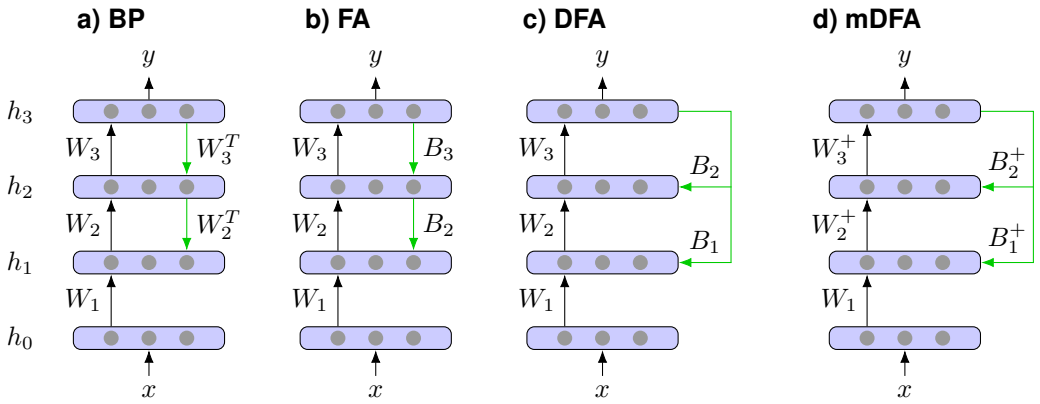

Figure 1: Graphical overview of various error transport methods, adapted from Nøkland (2016). Black arrows represent the forward path, whereas green arrows represent the error feedback. Weights that are learned are denoted as W_i and fixed weights are denoted by B_i. Weights that are restricted to only positive values are annotated by a $^+$. **a)** Backpropagation, **b)** Feedback alignment, **c)** Direct Feedback Alignment, **d)** monotone Direct Feedback Alignment.

2.1 Backpropagation

Backpropagation (Rumelhart et al., 1986) is the primary method to compute the gradients needed by the updates in equation (2) by iteratively applying the chain-rule

$$\frac{d\mathcal{L}}{d\theta_l} = \left(\frac{dh_l}{d\theta_l}\right)^T \frac{d\mathcal{L}}{dh_l} \tag{3}$$

$$\frac{d\mathcal{L}}{dh_l} = \left(\frac{dh_{l+1}}{dh_l}\right)^T \frac{d\mathcal{L}}{dh_{l+1}} \tag{4}$$

$$\frac{dh_{l+1}}{dh_l} = W_{l+1}\text{diag}(f'(W_{l+1}h_l + b_{l+1})). \tag{5}$$

A graphical representation of how information first flow forward and then backward in BP through each layer is shown in Figure (1) a.

Two major concerns argue against the idea that biological neural networks are implementing BP-based learning. I) The weight matrix W_l of the forward path is reused in the backward path in the form of W_l^T (*weight sharing*), and II) the strict separation of activity carrying forward and error carrying backward connections (*reciprocal error transport*).

2.2 Feedback Alignment algorithms

Feedback alignment addresses the implausibility of reusing W_l^T in the backward path by replacing W_l^T by a fixed random matrix B_l. Lillicrap et al. (2014) showed that this somewhat counterintuitive approach works remarkably well in practice. The term "feedback alignment" originates from the observations made by Lillicrap et al. (2014) that the angle between the FA update vector and the true gradient starts to decrease, i.e., *align*, after a few epochs of the training algorithm. Theoretical groundwork on this alignment principle of FA relies on strong assumptions such as a linearized network with one hidden layer (Lillicrap et al., 2014).

FA avoids any *weight sharing* but does not address the *reciprocal error transport* implausibility, due to its strict separation of forward and backward pathways, as shown in Figure (1) b. Direct Feedback Alignment (DFA) (Nøkland, 2016) relaxes this issue by replacing all backward paths with a direct feed from the output layer error-gradient $\frac{d\mathcal{L}}{dh_L}$. Consequently, there is only a single error signal that is distributed across the entire network, which is arguably more biologically plausible than reciprocal error connections. The resulting parameter updates of DFA are of the form

$$\delta\theta_l := \begin{cases} \frac{d\mathcal{L}}{dh_L}\frac{h_L}{\theta_l} & \text{if } l = L \\ \frac{d\mathcal{L}}{dh_L}B_l\frac{h_l}{\theta_l} & \text{if } l < L \end{cases} \tag{6}$$

, where $B_l \in \mathbb{R}^{n_L \times n_l}$ is a random matrix. A graphical schematic of DFA is shown in Figure (1) c. Similar to FA, DFA shows a decent learning performance in mid sized classification tasks (Nøkland, 2016), but fails on more complex datasets such as ImageNet (Bartunov et al., 2018). Theory on adapting the alignment principle to DFA shows that under the strong assumptions of constant DFA update directions and a layer-wise criterion minimization, the DFA update vector will align with the true gradient (Nøkland, 2016; Gilmer et al., 2017).

Recently, Frenkel et al. (2019) proposed to combine ideas from feedback alignment and target-propagation in their Direct Random Target Projection (DRTP) algorithm. While DRTP shows decent empirical performance, theoretical guarantees about DRTP rely on linearized networks.

3 MONOTONE DIRECT FEEDBACK ALIGNMENT

In this section, we first introduce a new class of feed-forward networks, where all, except the first, layers are constrained to realize monotone functions. We call such networks *mono-nets* and show that they are as expressive as unconstrained feed-forward networks. Next, we prove that for our mono-nets with single output tasks, e.g., binary-classification, feedback alignment algorithms provide the sign of the gradient. The sign of the gradient is interesting for learning, as it tells us if the value of a parameter should be increased or decreased in order to reduce the loss. At the end of this section, we will highlight similarities to algorithms from literature, which can provide resilient learning by only relying on the sign of the gradient.

Neural networks with monotonic constraints have been already studied in literature (You et al., 2017), however not in the context of learning algorithms.

Definition 1 (mono-net)**.** *A mono-net is a feed-forward neural network with L layers $h_1, \dots h_L$, each layer l composted of n_l units and the semantics*

$$h_l(h_{l-1}) := \begin{cases} f(W_l h_{l-1} + b_l) & \text{if } l < L \\ W_l h_{l-1} + b_l & \text{if } l = L \end{cases} \tag{7}$$

$$h_0 := x \tag{8}$$

$$W_1 \in \mathbb{R}^{n_1 \times n_0}, \tag{9}$$

$$W_l \in \mathbb{R}_+^{n_l \times n_{l-1}}, \text{ for } l > 1 \tag{10}$$

$$b_l \in \mathbb{R}^{n_l} \tag{11}$$

where $\mathbb{R}_+$ are the positive reals, i.e. $\mathbb{R}_+ = \{x \in \mathbb{R} | x \geq 0\}$, f is a non-linear monotonic increasing activation function, x the input and h_L the output of the network.

The major difference between mono-nets and general feed-forward neural networks is the restriction to only positive weight values in layers from the second layer on. Combined with the monotonic increasing activation function, this means that each layer $h_l(h_{l-1}), l \geq 2$ realizes a monotone increasing function. Because functional composition preserves monotonicity, the complete network up to the first layer

$$h_l \circ h_{l-1} \circ \cdots \circ h_2(h_1) \tag{12}$$

implements a monotone increasing function.

mono-nets are Universal Approximators At first glance, this restriction seems counterproductive, as it might interfere with the expressiveness of the networks. However, we proof in Theorem 1 that our mono-nets with tangent hyperbolic activation are universal approximators, meaning that we can approximate any continuous function arbitrarily close. A potential drawback of the monotonicity constraint is that we might need a larger number of units in the hidden layers to achieve the same expressiveness as a general feed-forward network, as illustrated in our proof of Theorem 1.

Theorem 1 (mono-nets are Universal Approximators)**.** *Let I_n be the n-dimensional unit hypercube $[0,1]^n$ and $C(I_n)$ denote the set of continuous functions $f : I_n \to \mathbb{R}$. We define $\|f\|_\infty$ as the supremum norm of $f \in C(I_n)$ over its domain I_n. For any given $f \in C(I_n)$ and $\varepsilon > 0$, there exist a function $m : I_n \to \mathbb{R}$ of the form*

$$m(x) := \sum_{i=1}^{M} \bar{v}_i \tanh(\bar{w}_i{}^T x + \hat{w}_i{}^T(-x) + \bar{b}_i) + c \tag{13}$$

with $\bar{v} \in {\mathbb{R}_+}^M, \bar{w}_i \in {\mathbb{R}_+}^n, \hat{w}_i \in {\mathbb{R}_+}^n, \bar{b} \in \mathbb{R}^M, c \in \mathbb{R}$ *and* $M < \infty$ *such that*

$$\|m(x) - f(x)\|_\infty < \varepsilon.$$

In essence, the set of functions $m(x)$ *of the form given in (19) is* dense *in* $C(I_n)$.

Proof. See supplementary materials 3. □

mDFA provides the sign of the gradient Here, we prove that for 1-dimensional outputs DFA applied on a mono-net, which we will call simply mDFA, provides the sign of the true gradient. Note that we focus our methods on DFA instead of "vanilla" FA, due to the superiority of DFA in terms of biological plausibility and empirical performance (Nøkland, 2016; Bartunov et al., 2018).

Theorem 2 (For 1-dimensional outputs mDFA computes the sign of the gradient)**.** *Let* $\mathcal{L}(y, h_L)$ *be a loss function,* $m(x) := h_L \circ h_{L-1} \circ \cdots \circ h_2 \circ h_1 \circ h_0(x)$ *be a mono-net according to Definition 1 with parameters* $\Theta := \{W_l, b_l \big| l = 1, \dots L\}$. *We denote* $\delta\theta$ *the update value computed by mDFA and* $\nabla\theta$ *as the gradient* $\frac{\partial \mathcal{L}}{\partial \theta}$ *for any* $\theta \in \{W_l, b_l\}$ *with* $1 \leq l \leq L$. *If* $n_L = 1$, *it follows that*

$$\Big(\delta\theta\Big)_{i,j} \cdot \Big(\nabla\theta\Big)_{i,j} \geq 0, \tag{14}$$

for each coordinate (i, j) *of* θ.

Proof. See supplementary materials. □

A graphical illustration of how activities and errors propagate in mDFA is shown in Figure (1) d.

Literature on learning by relying only on the sign of the gradient Two learning concepts related to mDFA are RPROP (Riedmiller & Braun, 1993; Riedmiller, 1994) and signSGD (Bernstein et al., 2018). RPROP aims to build a more resilient alternative to gradient descent by decoupling the amplitude of the gradient from the step size of parameter updates. In essence, for each coordinate RPROP adapts the step size based on the sign of the most recent gradients computed. Riedmiller & Braun (1993) showed that their approach could stabilize the training of a neural network compared to standard gradient descent.

Performing gradient descent with taking the sign of each gradient coordinate is on an algorithmic level equivalent to the well-known *steepest descent method* with L_∞ norm (Boyd & Vandenberghe, 2004; Bernstein et al., 2018). signSGD (Bernstein et al., 2018) studies convergence properties of the stochastic approximation of this algorithm.

What about networks with more than one output neuron? Theorem 2 applies only to networks with scalar output. As a natural consequence, one may ask whether such theoretical guarantees can be extended to more dimensional output variables. The simple answer is, unfortunately not. In the supplementary materials section A.3 we provide a counterexample showing that Theorem 2 naively extended to two output neurons does not hold anymore.

4 Experiments

So far, we have only proven the learning capabilities of mDFA. What remains unclear is whether mDFA can deliver its theoretical promises in practice. In this section, we experimentally evaluate the learning performance of mDFA on a set of empirical benchmarks. We aim to answer the following two questions:

- How well does mDFA perform compared to DFA, FA, and backpropagation in "natural conditions," i.e., in binary classification tasks, and
- how much does the performance of mDFA degrade in multi-class classification tasks?

Our performance measure is the achieved classification accuracy of a network trained by a particular method. First, we report the highest achieved accuracy on the training set, which tells us how well the algorithm could fit the model to the training data. Secondly, for each method, we tuned the hyperparameters on a separate validation set and selected the best performing configuration to be

evaluated on the test data. The obtained test accuracy tells us how well the model generalizes to data outside the training set.

We evaluate fully-connected networks (FC) and Convolutional networks (CNNs) in form of modified *all-convolutional-nets* (Springenberg et al., 2015) with tangent hyperbolic, ReLU (Nair & Hinton, 2010), and a third "sc-ReLU" non-linearity. Our sc-ReLU is a symmetric version of the capped ReLU-n activation function proposed by Krizhevsky & Hinton (2010)

$$\text{sc-ReLU}(x) := \min(\max(x, -1), 1). \tag{15}$$

Hyperparameters For all training methods, we fixed the batch size to 64, applied no regularization, no normalization, and no data augmentation. Optimizer, i.e., {"Vanilla" Gradient Descent, Adam (Kingma & Ba, 2014), Rmsprop (Tieleman & Hinton, 2012) }, learning rate, training epochs, and weight initialization method were tuned on the validation set. We tested three different weight initialization schemes; all zeros, a scaled uniform, and a normal distribution. Note that *all zeros* was only tested on the forward weights. Our uniform initialization followed the methodology of Nøkland (2016), i.e., scaling the bounds of the distribution inversely by the square-root of the number of incoming connections of a neuron. In order to comply with the weight constraints of mono-nets, for mDFA the lower bound of the uniform distribution was set to $\varepsilon = 10^{-3}$. Moreover, for mDFA we post-processed the initial weights of the normal distribution by taking their absolute values. Input variables are scaled to [0,1]. Detailed network architectures and a brief discussion about the best performing hyperparameter configurations are listed in the supplementary materials in section B.1 and section B.3.

4.1 Binary classification

We created a series of binary classification benchmarks by randomly sampling two classes from the well-studied CIFAR-100 and ImageNet datasets. We then train and test a 1-vs-1 classifier on the samples of the two classes. For each dataset, we create five such "binarized" sub-datasets and report the mean and standard deviation over these experiments. CIFAR-100 (Krizhevsky et al., 2014) poses a challenging classification dataset, consisting of 32-by-32 RGB images in 100 real-world object categories. ImageNet (Russakovsky et al., 2015) is a large-scale object classification benchmark and de-facto standard to asses new deep learning methods. Each sample is a high-resolution image representing one out of 1000 possible object classes. We pre-processed all samples by cropping and resizing each image to 224-by-224 pixels. Because ImageNet lacks a public test set, we will report the validation accuracy, i.e., as it was reported by Bartunov et al. (2018).

The results on the binarized benchmarks are shown in Table 1 for CIFAR-100 and Table 2 for ImageNet. mDFA could bring the training error to zero for fully-connected networks, and match the test/validation accuracy of backpropagation for convolutional networks.

Poor learning performance for ReLU networks One surprising characteristic in our results is that mDFA fails to provide the same level of learning performance as backpropagation for ReLU net-

Activation	Model	Training accuracy (FC)	Test accuracy (FC)	Training accuracy (CNN)	Test accuracy (CNN)
sc-ReLU	BP	**100.0** $\pm$ 0.0%	**81.3** $\pm$ 11.4%	**100.0** $\pm$ 0.0%	**80.9** $\pm$ 10.4%
	DFA	84.1 $\pm$ 8.2%	76.8 $\pm$ 11.5%	84.4 $\pm$ 7.9%	80.0 $\pm$ 10.3%
	FA	85.3 $\pm$ 8.0%	76.9 $\pm$ 12.0%	82.5 $\pm$ 10.2%	77.5 $\pm$ 12.2%
	mDFA	**100.0** $\pm$ 0.0%	77.3 $\pm$ 12.2%	87.4 $\pm$ 7.0%	**80.9** $\pm$ 9.2%
ReLU	BP	**100.0** $\pm$ 0.0%	**80.0** $\pm$ 10.8%	**100.0** $\pm$ 0.0%	**82.2** $\pm$ 9.5%
	DFA	76.2 $\pm$ 9.6%	72.9 $\pm$ 9.2%	73.0 $\pm$ 15.5%	72.0 $\pm$ 16.2%
	FA	77.9 $\pm$ 10.2%	71.0 $\pm$ 12.0%	68.6 $\pm$ 15.9%	69.8 $\pm$ 15.7%
	mDFA	78.2 $\pm$ 9.9%	62.1 $\pm$ 12.7%	78.0 $\pm$ 11.7%	75.5 $\pm$ 11.7%
tanh	BP	**100.0** $\pm$ 0.0%	**81.4** $\pm$ 10.5%	**100.0** $\pm$ 0.0%	**81.1** $\pm$ 11.9%
	DFA	84.2 $\pm$ 8.2%	77.5 $\pm$ 11.2%	81.6 $\pm$ 9.3%	79.7 $\pm$ 10.5%
	FA	84.9 $\pm$ 8.0%	77.6 $\pm$ 12.2%	94.2 $\pm$ 3.5%	**83.4** $\pm$ 8.7%
	mDFA	**100.0** $\pm$ 0.0%	77.2 $\pm$ 12.2%	89.0 $\pm$ 6.0%	**81.1** $\pm$ 9.8%

Table 1: Accuracies on binarized-CIFAR-100. Mean and standard deviation

Activation	Model	Training acc. (FC)	Validation acc. (FC)	Training acc. (CNN)	Validation acc. (CNN)
sc-ReLU	BP	**98.4** $\pm$ 3.6%	77.3 $\pm$ 6.8%	**99.8** $\pm$ 0.5%	81.7 $\pm$ 11.6%
	DFA	93.8 $\pm$ 13.0%	74.0 $\pm$ 8.5%	**100.0** $\pm$ 0.0%	83.0 $\pm$ 8.6%
	FA	97.5 $\pm$ 0.8%	74.7 $\pm$ 7.7%	51.9 $\pm$ 0.5%	54.7 $\pm$ 4.6%
	mDFA	95.3 $\pm$ 10.4%	**79.5** $\pm$ 9.8%	79.5 $\pm$ 9.8%	**83.8** $\pm$ 7.2%
ReLU	BP	**100.0** $\pm$ 0.0%	**78.3** $\pm$ 7.7%	**90.0** $\pm$ 18.1%	**72.5** $\pm$ 13.7%
	DFA	75.1 $\pm$ 8.1%	73.2 $\pm$ 8.0%	50.4 $\pm$ 0.9%	52.7 $\pm$ 2.4%
	FA	68.3 $\pm$ 7.9%	71.0 $\pm$ 6.7%	52.3 $\pm$ 0.9%	54.8 $\pm$ 2.1%
	mDFA	56.9 $\pm$ 0.9%	59.7 $\pm$ 2.4%	52.6 $\pm$ 1.6%	58.4 $\pm$ 4.5%
tanh	BP	**99.9** $\pm$ 0.2%	75.8 $\pm$ 7.7%	**99.9** $\pm$ 0.1%	76.2 $\pm$ 9.8%
	DFA	97.9 $\pm$ 2.2%	76.8 $\pm$ 7.2%	**99.3** $\pm$ 1.5%	**83.2** $\pm$ 8.8%
	FA	98.1 $\pm$ 2.3%	76.5 $\pm$ 7.2%	62.4 $\pm$ 17.3%	60.8 $\pm$ 13.8%
	mDFA	**99.9** $\pm$ 0.1%	**78.0** $\pm$ 6.8%	78.2 $\pm$ 9.4%	**83.0** $\pm$ 6.8%

Table 2: Accuracies on binarized-ImageNet. Mean and standard deviation

works. Recall that Theorem 1 proves the universal approximation capabilities only for mono-net with tanh activation function. A mono-net with ReLU non-linearity restricts both, the activation and the weight values, to positive values. These constraints arguably limit the approximation capabilities of mono-net in combination with ReLU and thus explain the poor performance of mDFA for ReLU networks. We could validate our hypothesis by testing a rectifier non-linearity that also contains negative values in its image. Coincidentally, our sc-ReLU function matches exactly this criterion. Therefore, the decent learning performance of mDFA for sc-ReLU networks confirms our hypothesis. Note that also FA and DFA expressed an improvement in performance when switching from ReLU to sc-ReLU activation. This observation suggests that feedback alignment algorithms in general benefit from symmetric activation functions.

Discrepancy between fully-connected and convolutional networks Though mDFA achieved the same training accuracy as backpropagation for fully-connected networks, the generalization ability, i.e., test and validation accuracy, slightly lacks behind BP in our binarized-CIFAR-100 benchmark. This observation aligns with the studies by Nøkland (2016); Bartunov et al. (2018). For convolutional neural networks, this effect is reversed, i.e., a decent test/validation accuracy but a higher training error than BP. We speculate that the two tested initialization schemes for the feedback weights caused this discrepancy. Both backpropagation and feedback alignment have been shown to be sensitive to the employed initialization method (Zhang et al., 2019; Bartunov et al., 2018). The restriction to positive values of the weights used in mDFA, requires a re-thinking of the initialization methods examined in the literature. However, we will leave this study open for future work.

4.2 Multi-class classification

Here we modify the classification benchmark creation procedure used above, by randomly sampling n classes from the datasets instead of just two. Due to their compelling results in our binary classification benchmark, we restrict our evaluation to networks with tanh activation. Table 3 and Table 4 show the results on our n-class CIFAR-100 benchmark for fully-connected and CNNs respectively. The results on n-class ImageNet can be found in the supplementary materials in Table 5 and 6.

mDFA can provide learning for networks with more than one output neuron Though we do not have a complete theory on mDFA for multi-dimensional outputs, our experiments indicate that mDFA can provide learning for networks with more than one output neuron. In particular, the achieved accuracies of mDFA outperforms other feedback alignment methods for convolutional networks with ten or fewer output neurons. However, mDFA falls behind standard DFA for networks with more than ten output neurons. This observation suggests that our restriction to positive weights can be beneficial even for multi-class tasks but eventually hurts learning performance when the number of classes grows larger.

Feedback alignment algorithms, in general, tend to struggle with increasing output dimension Our results express the trend that with an increase in the number of classes, feedback alignment algorithms struggle to fit the training data. While BP can reduce the training error to almost zero

Set	Classes	BP	DFA	FA	mDFA
Training	3	**100.0** ± 0.0%	50.5 ± 9.5%	79.0 ± 6.1%	53.3 ± 8.9%
	5	**100.0** ± 0.0%	33.8 ± 9.7%	68.4 ± 6.0%	34.1 ± 9.8%
	10	**100.0** ± 0.0%	24.7 ± 1.8%	52.4 ± 4.0%	25.9 ± 1.2%
	25	**99.9** ± 0.1%	13.4 ± 1.7%	31.3 ± 1.6%	13.1 ± 1.4%
	50	**99.7** ± 0.1%	8.5 ± 0.1%	20.0 ± 1.1%	7.9 ± 0.2%
Test	3	**77.4** ± 8.7%	49.1 ± 7.2%	**74.2** ± 8.1%	50.9 ± 7.2%
	5	**68.5** ± 8.1%	31.2 ± 11.6%	**60.7** ± 8.4%	33.9 ± 9.8%
	10	**57.3** ± 4.9%	25.0 ± 3.2%	46.9 ± 3.8%	26.3 ± 2.3%
	25	**41.2** ± 2.6%	13.4 ± 1.9%	28.2 ± 1.4%	13.1 ± 1.3%
	50	**31.8** ± 1.8%	8.8 ± 0.4%	18.1 ± 0.9%	8.0 ± 0.4%

Table 3: Mulit-class classification accuracies of fully-connected network with tanh activation on n-class CIFAR-100. Mean and standard deviation.

Set	Classes	BP	DFA	FA	mDFA
Training	3	**100.0** ± 0.0%	86.3 ± 5.0%	84.1 ± 10.2%	**100.0** ± 0.0%
	5	**100.0** ± 0.0%	84.6 ± 4.6%	74.3 ± 8.6%	**100.0** ± 0.0%
	10	**100.0** ± 0.0%	83.0 ± 3.5%	57.9 ± 3.4%	94.4 ± 1.7%
	25	**99.9** ± 0.0%	81.0 ± 1.5%	36.3 ± 1.8%	64.8 ± 1.4%
	50	**98.9** ± 0.0%	80.5 ± 1.0%	22.7 ± 1.4%	44.5 ± 0.9%
Test	3	76.2 ± 10.1%	79.8 ± 6.2%	79.6 ± 9.7%	**82.0** ± 7.6%
	5	65.4 ± 9.8%	71.8 ± 7.6%	70.1 ± 9.0%	**73.3** ± 8.8%
	10	57.5 ± 5.2%	**65.0** ± 4.8%	55.1 ± 4.1%	63.9 ± 4.2%
	25	41.8 ± 2.2%	**48.3** ± 2.2%	35.0 ± 1.4%	44.0 ± 2.5%
	50	33.3 ± 1.5%	**37.1** ± 1.5%	22.3 ± 1.1%	34.5 ± 1.5%

Table 4: Mulit-class classification accuracies of Convolutional Neural Network with tanh activation on n-class CIFAR-100. Mean and standard deviation.

independently of the output dimension, the training errors achieved by FA algorithms are significantly higher and correlate with the number of output neurons. Our observations suggest that the dimension of the error signal affects the training convergence for FA algorithms, in contrast to BP, which appears to be less affected by the dimension of the error signal. This relation potentially provides a step towards understanding why FA algorithms fail on challenging datasets as described by Bartunov et al. (2018).

5 Conclusion

Feedback alignment algorithms are promising biologically motivated alternatives to backpropagation. While existing literature provides empirical evidence that FA algorithms can work well in practice, there is still a lack of rigorous theory formalizing their learning capabilities. Here we contributed to the field of biologically motivated learning algorithms, by introducing the first feedback alignment algorithm that provable provides learning for non-linear networks of arbitrary depth and single output neuron. We showed that our FA algorithm outperforms existing FA algorithms in binary classification tasks, and even provide decent learning on multi-class problems.

Is this really useful? In terms of scientific significance, this work provided theoretical contributions toward understanding the capabilities and limits of feedback alignment algorithms. From a practical point of view, our mDFA algorithm has an advantage over backpropagation concerning training latency, as all weight updates can be computed in parallel, i.e., see Figure 1. Furthermore, we have shown that mDFA is superior to other feedback alignment algorithms in binary classification tasks. Consequently, mDFA provides an effective solution for binary classification problems with training latency constraints. *Dynamic branch prediction* in microprocessor pipelines poses such problem instance, where program specific binary branch outcomes, i.e., branch taken/not taken, need to be learned in real-time. Because of this real-time constraint, existing branch predictors often employ only shallow perceptron modules (Jiménez & Lin, 2002; Egan et al., 2003). mDFA could enable deeper branch predictor networks to be learned in real-time.

REFERENCES

Pierre Baldi and Fernando Pineda. Contrastive learning and neural oscillations. *Neural Computation*, 3(4):526–545, 1991.

Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in Neural Information Processing Systems*, pp. 9389–9399, 2018.

Yoshua Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*, 2014.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 560–569, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Francis Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.

Colin Egan, Gordon Steven, Patrick Quick, Rubén Anguera, Fleur Steven, and Lucian Vintan. Two-level branch prediction using neural networks. *Journal of Systems Architecture*, 49(12-15):557–570, 2003.

Charlotte Frenkel, Martin Lefebvre, and David Bol. Learning without feedback: Direct random target projection as a feedback-alignment algorithm with layerwise feedforward training. *arXiv preprint arXiv:1909.01311*, 2019.

Justin Gilmer, Colin Raffel, Samuel S Schoenholz, Maithra Raghu, and Jascha Sohl-Dickstein. Explaining the learning dynamics of direct feedback alignment. *ICLR - Workshop track*, 2017.

Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

Geoffrey Hinton. How to do backpropagation in a brain. In *Invited talk at the NIPS2007 Deep Learning Workshop*, volume 656, 2007.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Daniel A Jiménez and Calvin Lin. Neural methods for dynamic branch prediction. *ACM Transactions on Computer Systems (TOCS)*, 20(4):369–397, 2002.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 2010.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-100 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

Yann LeCun. Learning process in an asymmetric threshold network. In *Disordered systems and biological organization*, pp. 233–240. Springer, 1986.

Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*, 2014.

Javier R Movellan. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist models*, pp. 10–17. Elsevier, 1991.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in neural information processing systems*, pp. 1037–1045, 2016.

Martin Riedmiller. Rprop-description and implementation details. *Citeseer*, 1994.

Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proceedings of the IEEE international conference on neural networks*, volume 1993, pp. 586–591. San Francisco, 1993.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *International Conference on Learning Representations (ICLR)*, 2015.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya Gupta. Deep lattice networks and partial monotonic functions. In *Advances in Neural Information Processing Systems*, pp. 2981–2989, 2017.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Residual learning without normalization via better initialization. *International Conference on Learning Representations*, 2019.

A APPENDIX

A.1 MONO-NETS ARE UNIVERSAL APPROXIMATORS

We denote

$$\sigma(x) := \frac{1}{1+\exp(-x)}$$

the *sigmoid* function,

$$\tanh(x) := \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

the *hyperbolic tangent* and

$$\mathbb{1}[expr] := \begin{cases} 1 & \text{if } expr \text{ is true} \\ 0 & \text{if } expr \text{ is false} \end{cases}$$

the *indicator function*.

From the definitions above we can derive the equalities

$$\sigma(x) = \frac{1}{2}\Big(1 + \tanh(\frac{x}{2})\Big) \tag{16}$$

$$\tanh(x) = -\tanh(-x) \tag{17}$$

, which we will need for our proof.

Lemma 1 (Universal Approximation Theorem)**.** *For any given $f \in C(I_n)$ and $\varepsilon > 0$, there exist a function $g : I_n \to \mathbb{R}$ of the form*

$$g(x) := \sum_{i=1}^{N} v_i \sigma(w_i^T x + b_i) \tag{18}$$

with $v \in \mathbb{R}^N, w_i \in \mathbb{R}^n, b \in \mathbb{R}^N$ and $N < \infty$ such that

$$\|g(x) - f(x)\|_\infty < \varepsilon.$$

In essence, the set of functions $g(x)$ of the form given in (18) is dense *in $C(I_n)$.*

Proof. See Hornik et al. (1989) □

Theorem 3 (mono-nets are Universal Approximators)**.** *For any given $f \in C(I_n)$ and $\varepsilon > 0$, there exist a function $m : I_n \to \mathbb{R}$ of the form*

$$m(x) := \sum_{i=1}^{M} v_i \tanh(w_i^T x + b_i) + c \tag{19}$$

with $v \in {\mathbb{R}_+}^M, w_i \in \mathbb{R}^n, \bar{b} \in \mathbb{R}^M, c \in \mathbb{R}$ and $M < \infty$ such that

$$\|m(x) - f(x)\|_\infty < \varepsilon.$$

In essence, the set of functions $m(x)$ of the form given in (19) is dense *in $C(I_n)$.*

Proof. By Lemma 1 we know that there exist a $g(x)$ with $\|g(x) - f(x)\|_\infty < \varepsilon$ that has the form

$$g(x) = \sum_{i=1}^{N} v_i \sigma(w_i^T x + b_i)$$

We will show that we can reformulate $g(x)$ to the form in equation (19). Our basic idea is to propagate all negative weight entries into to the first layer where negative weight values are allowed.

$$
\begin{aligned}
g(x) &= \sum_{i=1}^{N} v_i \sigma(w_i^T x + b_i) \\
&= \sum_{i=1}^{N} v_i \frac{1}{2}\Big(1 + \tanh(w_i^T x + b_i)\Big) \\
&= \sum_{i=1}^{N} v_i \frac{1}{2} \tanh\Big(\sum_{j=1}^{n} w_{i,j} x_j + b_i\Big) + \frac{1}{2}\sum_{i=1}^{N} v_i \\
&= \sum_{i=1}^{N} \mathbb{1}[v_i \geq 0] v_i \frac{1}{2} \tanh\Big(\sum_{j=1}^{n} w_{i,j} x_j + b_i\Big) \\
&\quad + \sum_{i=1}^{N} \mathbb{1}[v_i < 0] v_i \frac{1}{2} \tanh\Big(\sum_{j=1}^{n} w_{i,j} x_j + b_i\Big) \\
&\quad + \frac{1}{2}\sum_{i=1}^{N} v_i \\
&= \sum_{i=1}^{N} \mathbb{1}[v_i \geq 0] v_i \frac{1}{2} \tanh\Big(\sum_{j=1}^{n} w_{i,j} x_j + b_i\Big) \\
&\quad + \sum_{i=1}^{N} -\mathbb{1}[v_i < 0] v_i \frac{1}{2} \tanh\Big(-\sum_{j=1}^{n} w_{i,j} x_j - b_i\Big) \\
&\quad + \frac{1}{2}\sum_{i=1}^{N} v_i \\
&= \sum_{i=1}^{N} \bar{v_i} \tanh\Big({w_i}^T x + b_i\Big) \\
&\quad + \sum_{i=1}^{N} \bar{v_i'} \tanh\Big(\bar{w_i}^T x + \bar{b_i}\Big) + c
\end{aligned}
$$

with

$$
\begin{aligned}
\bar{v_i} &= \mathbb{1}[v_i \geq 0] v_i \frac{1}{2} \geq 0 \\
\bar{v_i'} &= -\mathbb{1}[v_i < 0] v_i \frac{1}{2} \geq 0 \\
\bar{w_i} &= -{w_i}^T \\
\bar{b_i} &= -b_i \\
c &= \frac{1}{2}\sum_{i=1}^{N} v_i
\end{aligned}
$$

Therefore, we showed that there exist a $m(x)$ with a form as in equation (19) that satisfy

$$\|m(x) - f(x)\|_\infty < \varepsilon.$$

□

This proof showed us how to translate any neural network with sigmoid activation function and one hidden layer of size N into a mono-net with $2N$ hidden units.

A.2 FOR 1-DIMENSIONAL OUTPUTS MDFA COMPUTES THE SIGN OF THE GRADIENT

Lemma 2 (The gradient of a monotone layer is non-negative)**.** *Let $m : \mathbb{R}^{n_0} \to \mathbb{R}$ with $m(x) := h_L \circ h_{L-1} \circ \cdots \circ h_2 \circ h_1 \circ h_0(x)$ be a mono-net according to Definition 1. Then*

$$\Big(\frac{dh_l}{dh_{l-1}}\Big)_{i,j} \geq 0, \tag{20}$$

for any l, i, j with $2 \leq l \leq L$ and $1 \leq i \leq n_l$ and $1 \leq j \leq n_{l-1}$

Proof. We have to distinguish two cases:
Case 1: $l = L$, i.e. there is no activation function.
We have

$$\Big(\frac{dh_l}{dh_{l-1}}\Big)_{i,j} = \Big(W_l\Big)_{i,j} \geq 0, \tag{21}$$

according to the definition in Equation (10).
Case 2: $l < L$, i.e. there is an activation function.
We have

$$\Big(\frac{dh_l}{dh_{l-1}}\Big)_{i,j} = \Big(W_l \text{diag}(f'(W_l h_{l-1} + b_l))\Big)_{i,j} \tag{22}$$

$$= \sum_{k=1}^{n_l} \Big(W_l\Big)_{i,k} \Big(\text{diag}(f'(W_l h_{l-1} + b_l))\Big)_{k,j}, \tag{23}$$

where f' is the derivative of the activation function. Because f is a monotonic function, its derivative is non-negative everywhere. As a result we have a sum of a product of non-negative values. Ergo

$$\Big(\frac{dh_l}{dh_{l-1}}\Big)_{i,j} \geq 0 \tag{24}$$

□

Lemma 3 (The gradient of a composition of monotone layers is non-negative)**.** *Let $m : \mathbb{R}^{n_0} \to \mathbb{R}$ with $m(x) := h_L \circ h_{L-1} \circ \cdots \circ h_2 \circ h_1 \circ h_0(x)$ be a mono-net according to Definition 1. Then*

$$\Big(\frac{dh_l}{dh_k}\Big)_{i,j} \geq 0, \tag{25}$$

for any l, k, i, j with $2 \leq l \leq L$ and $1 \leq k < l$ and $1 \leq i \leq n_l$ and $1 \leq j \leq n_k$

Proof. By applying the chain rule we get

$$\Big(\frac{dh_l}{dh_k}\Big)_{i,j} = \Big(\prod_{m=k+1}^{l} \frac{dh_m}{dh_{m-1}}\Big)_{i,j} \tag{26}$$

According to Lemma 2 we have a product of non-negative values. Because a product of non-negative values is non-negative itself, we have

$$\Big(\frac{dh_l}{dh_k}\Big)_{i,j} \geq 0 \tag{27}$$

□

Theorem 4 (For 1-dimensional outputs mDFA computes the sign of the gradient)**.** *Let $\mathcal{L}(y, h_L)$ be a loss function, $m(x) := h_L \circ h_{L-1} \circ \cdots \circ h_2 \circ h_1 \circ h_0(x)$ be a mono-net according to Definition 1 with parameters $\Theta := \{W_l, b_l \big| l = 1, \ldots L\}$. We denote $\delta\theta$ the update value computed by mDFA and $\nabla\theta$ as the gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ for any $\theta \in \{W_l, b_l\}$ with $1 \leq l \leq L$. If $n_L = 1$, it follows that*

$$\Big(\delta\theta\Big)_{i,j} \cdot \Big(\nabla\theta\Big)_{i,j} \geq 0, \tag{28}$$

for each coordinate (i, j) of θ.

Proof. We distinguish two cases:
Case 1: $l = L$, i.e. θ is a parameter of the last layer. From the definition of mDFA we have

$$\delta\theta := \frac{d\mathcal{L}}{dh_L}\frac{dh_L}{d\theta}. \tag{29}$$

For the gradient by applying the chain rule we get

$$\nabla\theta = \frac{d\mathcal{L}}{d\theta} = \frac{dL}{dh_L}\frac{dh_L}{d\theta} \tag{30}$$

Thus, in the last layer the mDFA update equals the gradient.
Case 2: $l < L$, i.e θ is a parameter of a hidden layer. From the definition of mDFA we have

$$\Big(\delta\theta\Big)_{i,j} := \Big(\frac{d\mathcal{L}}{dh_L}B\frac{dh_l}{d\theta}\Big)_{i,j} \tag{31}$$

with $B \in {\mathbb{R}_+}^{n_L \times n_l}$. Next, we expand the multiplication,

$$\Big(\delta\theta\Big)_{i,j} = \sum_{k=1}^{n_L}(\frac{d\mathcal{L}}{dh_L})_k B_{k,i}\frac{d(h_l)_i}{d\theta_{i,j}} \tag{32}$$

$$= \frac{d(h_l)_i}{d\theta_{i,j}}\sum_{k=1}^{n_L}(\frac{d\mathcal{L}}{dh_L})_k B_{k,i}. \tag{33}$$

We assumed that the output dimension is 1, i.e. $n_L = 1$. Therefore,

$$\Big(\delta\theta\Big)_{i,j} = \frac{d(h_l)_i}{d\theta_{i,j}}(\frac{d\mathcal{L}}{dh_L})_1 B_{1,i}. \tag{34}$$

For the gradient by applying the chain rule we get

$$\nabla\theta = \frac{d\mathcal{L}}{d\theta} = \frac{dL}{dh_L}\frac{dh_L}{d\theta} \tag{35}$$

$$= \frac{d\mathcal{L}}{dh_L}\frac{dh_L}{dh_l}\frac{dh_l}{d\theta} \tag{36}$$

Like above, we expand the multiplication,

$$\Big(\nabla\theta\Big)_{i,j} = \sum_{k=1}^{n_L}(\frac{d\mathcal{L}}{dh_L})_k(\frac{dh_L}{dh_l})_{k,i}\frac{d(h_l)_i}{d\theta_{i,j}} \tag{37}$$

$$= \frac{d(h_l)_i}{d\theta_{i,j}}\sum_{k=1}^{n_L}(\frac{d\mathcal{L}}{dh_L})_k(\frac{dh_L}{dh_l})_{k,i}. \tag{38}$$

We assumed that the output dimension is 1, i.e. $n_L = 1$. Therefore,

$$\Big(\nabla\theta\Big)_{i,j} = \frac{d(h_l)_i}{d\theta_{i,j}}(\frac{d\mathcal{L}}{dh_L})_1(\frac{dh_L}{dh_l})_{1,i}. \tag{39}$$

For Equation (28) we get by applying Lemma 3,

$$\Big(\delta\theta\Big)_{i,j}\cdot\Big(\nabla\theta\Big)_{i,j} = \Big(\frac{d(h_l)_i}{d\theta_{i,j}}(\frac{d\mathcal{L}}{dh_L})_1 B_{1,i}\Big)\cdot\Big(\frac{d(h_l)_i}{d\theta_{i,j}}(\frac{d\mathcal{L}}{dh_L})_1(\frac{dh_L}{dh_l})_{1,i}\Big) \tag{40}$$

$$= \underbrace{\Big(\frac{d(h_l)_i}{d\theta_{i,j}}(\frac{d\mathcal{L}}{dh_L})_1\Big)^2}_{\geq 0}\underbrace{B_{1,i}}_{\geq 0}\underbrace{(\frac{dh_L}{dh_l})_{1,i}}_{\geq 0} \tag{41}$$

$$\geq 0. \tag{42}$$

□

A.3 FOR $k \geq 2$-DIMENSIONAL OUTPUTS MDFA UPDATES MAY NOT ALIGN WITH THE GRADIENT

Theorem 5 (For $k \geq 2$-dimensional outputs mDFA updates may not align with the gradient)**.** *Let $\mathcal{L}(y, h_L)$ be a loss function, $m(x) := h_L \circ h_{L-1} \circ \cdots \circ h_2 \circ h_1 \circ h_0(x)$ be a mono-net according to Definition 1 with parameters $\Theta := \{W_l, b_l \big| l = 1, \ldots L\}$. We denote $\delta\theta$ the update value computed by mDFA and $\nabla\theta$ as the gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ for any $\theta \in \{W_l, b_l\}$ with $1 \leq l < L$. If $n_L \geq 2$, there exists the possibility that*

$$\Big(\delta\theta\Big)_{i,j} \cdot \Big(\nabla\theta\Big)_{i,j} < 0, \tag{43}$$

for at least one coordinate (i, j) of θ.

Proof. We construct a minimal counterexample consisting of a network with two outputs and one hidden layer of two neurons.

Let

$$W_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{44}$$

$$B_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{45}$$

$$\frac{\partial \mathcal{L}}{\partial h_L} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{46}$$

$$\frac{\partial h_1}{\partial \theta} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{47}$$

$$\tag{48}$$

Then we have

$$\nabla\theta = \frac{d\mathcal{L}}{dh_L} W_2^T \frac{dh_l}{d\theta} \tag{49}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{50}$$

$$= \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \tag{51}$$

and

$$\delta\theta = \frac{d\mathcal{L}}{dh_L} B_2 \frac{dh_l}{d\theta} \tag{52}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{53}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{54}$$

$$\tag{55}$$

, which are orthogonal. □

B EXPERIMENT SETUP

B.1 NETWORK ARCHITECTURES

Dataset	Fully-connected	Convolutional
CIFAR-100	1024,1024	(96,5,2),(96,3,2),(96,3,1)
ImageNet	1024,1024,1024,1024	(96,3,2),(96,5,1),(128,3,2), (192,3,1),(192,3,2),(384,3,1)

Network architectures, layers are separated by commas. Fully-connected column specifies the number of neurons of each layer. Convolutional column specifies the number of filters, kernel size, and stride for each layer.

B.2 n-CLASS IMAGENET

Set	Classes	BP	DFA	FA	mDFA
Training	3	**100.0** $\pm$ 0.0%	59.6 $\pm$ 7.2%	50.4 $\pm$ 5.9%	51.4 $\pm$ 4.4%
	5	**100.0** $\pm$ 0.0%	48.9 $\pm$ 5.0%	41.4 $\pm$ 5.5%	41.0 $\pm$ 7.1%
	10	**99.7** $\pm$ 0.5%	33.4 $\pm$ 4.5%	24.7 $\pm$ 3.3%	38.7 $\pm$ 10.1%
Validation	3	**74.5** $\pm$ 8.8%	62.8 $\pm$ 11.9%	64.2 $\pm$ 10.9%	62.0 $\pm$ 9.6%
	5	**63.0** $\pm$ 7.4%	43.8 $\pm$ 6.2%	42.3 $\pm$ 7.1%	37.3 $\pm$ 5.2%
	10	**41.2** $\pm$ 5.0%	30.8 $\pm$ 4.9%	26.0 $\pm$ 4.5%	33.3 $\pm$ 8.0%

Table 5: Mulit-class classification accuracy of fully-connected network with tanh activation on n-class ImageNet. Mean and standard deviation.

Set	Classes	BP	DFA	FA	mDFA
Training	3	**100.0** $\pm$ 0.0%	100.0 $\pm$ 0.0%	35.8 $\pm$ 0.7%	68.1 $\pm$ 8.2%
	5	**100.0** $\pm$ 0.0%	100.0 $\pm$ 0.0%	21.4 $\pm$ 0.4%	60.4 $\pm$ 6.2%
	10	**100.0** $\pm$ 0.0%	90.4 $\pm$ 10.0%	11.3 $\pm$ 0.7%	42.9 $\pm$ 3.9%
Validation	3	68.5 $\pm$ 13.0%	**76.5** $\pm$ 9.9%	53.3 $\pm$ 1.7%	**74.8** $\pm$ 8.8%
	5	61.6 $\pm$ 7.3%	**68.7** $\pm$ 6.3%	29.0 $\pm$ 2.0%	61.9 $\pm$ 5.3%
	10	**52.8** $\pm$ 4.6%	**53.6** $\pm$ 5.3%	10.6 $\pm$ 0.8%	44.1 $\pm$ 4.4%

Table 6: Mulit-class classification accuracy of Convolutional Neural Network with tanh activation on n-class ImageNet. Mean and standard deviation.

B.3 DISCUSSION ON HYPERPARAMETERS

We observed that all FA algorithms yield a more stable convergence with "vanilla" stochastic gradient descent, i.e., no post-processing of the FA updates, than with Adam (Kingma & Ba, 2014) or Rmsprop (Tieleman & Hinton, 2012). This may be non-surprising as these acceleration methods have been developed for gradient-based optimization, whereas FA updates roughly align with the gradients at best.

Furthermore, we observed that FA algorithms achieve a descent validation accuracy after the first few training epochs. However, in contrast to BP, these methods may require over a hundred training epochs to converge fully. This "fast-start-slow-convergence" aligns with the observations made by Lillicrap et al. (2014).

We found that FA, DFA, and mDFA perform best when all forward weights are initialized to zero. Notable exceptions are networks with ReLU activation function which express poor performance with the all-zeros initialization scheme. This poor performance with all-zeros initialization partially explains the poor observed performance of FA and DFA for ReLU networks.

In contrast to Nøkland (2016), we observed that backward weights initialized by a normal distribution perform slightly better than the scaled uniform proposed by Nøkland (2016).