Nonlinear Direct Feedback Alignment for Scalable Backpropagation-Free Training

Wu Lin[1], **Yan Fan**[1], **Chenxiang Ma**[2,3*], **Yinglan Feng**[4,5],
Jia Wang[1], **Ka-Chun Wong**[6], **Qiuzhen Lin**[1]
[1] College of Computer Science and Software Engineering, Shenzhen University,
[2] School of Artificial Intelligence, Tianjin University,
[3] Institute for Embodied Intelligence, Tianjin University,
[4] School of Artificial Intelligence, Shenzhen University,
[5] Intelligent Computing Center, Shenzhen University,
[6] Department of Computer Science, City University of Hong Kong
{linwu, yinglfeng, jia.wang, qiuzhlin}@szu.edu.cn, 2500101043@mails.szu.edu.cn,
machenxiang@tju.edu.cn, kc.w@cityu.edu.hk

Abstract

Backpropagation (BP) remains the predominant algorithm for training deep neural networks, but it is biologically implausible and suffers from the backward locking problem, constraining parallelization across layers. Direct Feedback Alignment (DFA), which propagates the global output error directly to each hidden layer via fixed feedback matrices, has emerged as a promising alternative for addressing these limitations. However, a significant performance gap persists between DFA and BP, which we attribute to the limited ability of fixed feedback matrices to capture the nonlinear transformation from hidden representations to the network output. To address this limitation, we propose Nonlinear Direct Feedback Alignment (NDFA), which constructs each hidden-layer feedback using a lightweight nonlinear surrogate subnetwork. To enhance the effectiveness of the feedback throughout training, we further propose a learning strategy to update the surrogate subnetwork, thereby better approximating the underlying nonlinear transformation. Extensive experiments across residual networks and vision transformers on CIFAR-10, CIFAR-100, SVHN, and ImageNet datasets demonstrate that NDFA consistently outperforms existing DFA-based methods and substantially narrows the performance gap to BP, while maintaining efficient parallel training. The code is available at https://github.com/YanFan321/NDFA.

1 Introduction

Backpropagation (BP) [1] has long been the dominant algorithm for training deep neural networks, underpinning the success of deep learning across a wide range of applications [2–6]. Despite its remarkable effectiveness, BP relies on layer-by-layer gradient propagation, forcing each hidden layer to wait for gradients from subsequent layers before updating its parameters. This issue, known as the backward locking problem [7], hinders efficient parallelization of the training process. Additionally, this raises concerns about the biological plausibility of BP [8, 9], as synaptic plasticity in the brain is believed to operate locally and in parallel, without coordinated signals across distant layers [10, 11].

Direct feedback alignment (DFA) [12] has emerged as a promising alternative for training deep neural networks. Unlike BP, DFA propagates error signals directly from the output layer to each hidden layer through fixed random feedback matrices, eliminating the need for layer-by-layer gradient

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

propagation. Consequently, DFA addresses the backward locking problem, as each layer can be updated independently, enabling efficient parallelization of the training process. Additionally, DFA is considered more biologically plausible because it supports local and parallel parameter updates while the global output error resembles neuromodulators that influence local synaptic plasticity [13–15]. Furthermore, DFA is well-suited to emerging non-von Neumann hardware, where BP is challenging to implement [16–19]. For example, DFA can be efficiently implemented via multiple scattering of light in optical processing units, achieving substantially higher throughput and energy efficiency than BP on conventional hardware [20].

Although DFA can be applied to a variety of modern deep learning tasks and architectures [21–23], its performance still lags significantly behind BP [24–26], particularly in large-scale networks [27]. To narrow this gap, existing efforts have primarily focused on improving how the feedback matrices in DFA are updated. For instance, Direct Kolen–Pollack (DKP) [28] updates the feedback matrices to align with the corresponding forward weights using the outer product of locally available hidden representations and the global output error. Recent approaches, such as Forward Direct Feedback Alignment (FDFA) [29] and Gradient-Aligned Projected Error (GrAPE) [30], leverage forward-mode Jacobian–vector products to refine the feedback matrices. However, these methods still fail to match the accuracy of BP in deep networks.

In this paper, we first analyze the feedback mismatch between DFA and BP. For each hidden layer, the feedback in BP is determined by a nonlinear subnetwork transforming the hidden representation to the network output. In contrast, DFA uses a fixed feedback matrix, which corresponds to a linear transformation. We argue that this structural mismatch between the two feedback mechanisms is a key source of the performance gap and cannot be resolved by simply updating the feedback matrices.

To address this limitation, we propose Nonlinear Direct Feedback Alignment (NDFA), which introduces a lightweight nonlinear surrogate subnetwork to construct feedback for each hidden layer, as illustrated in Figure 1. This surrogate serves as an approximate transformation from the hidden representation to the network output, enabling the feedback to capture representation-dependent nonlinear structure. To encourage the surrogate to better approximate the true transformation, we optimize it using the same target as the true subnetwork. The feedback derived from the surrogate is therefore continuously changed throughout training, allowing it to more effectively guide the learning of each hidden layer. We extensively evaluate the proposed NDFA method on different image classification datasets, including CIFAR-10 [31], CIFAR-100 [31], SVHN [32], and ImageNet [33], using convolutional and transformer-based architectures, including AlexNet [34], ResNet-32 [35], and ViT-Tiny [36]. We further evaluate the neural data similarity of NDFA on the Mante 2013 dataset [37]. Our key contributions are summarized as follows:

- We identify a key limitation of DFA: its feedback matrices fail to capture the nonlinear transformations from hidden representations to the network output.
- We propose NDFA, which leverages a lightweight nonlinear surrogate subnetwork to construct feedback for each hidden layer, thereby addressing the identified limitation. We further propose a learning strategy to update the feedback during training.
- Extensive experiments across diverse datasets and architectures demonstrate that NDFA consistently outperforms existing DFA-based methods, substantially narrows the gap to BP while preserving parallel training efficiency, and achieves strong neural data similarity.

2 Related work

A growing body of research has explored alternatives to BP [8]. One line of work focuses on addressing the biological implausibility of BP, particularly its reliance on exactly symmetric forward and backward weights [38, 39]. Feedback Alignment [40] shows that propagating error signals through random feedback connections instead of symmetric weights can still support effective learning [41]. Other approaches avoid exact gradient propagation by directly projecting data labels to hidden layers [42] or by propagating layer-wise targets [43, 44]. Despite their biological motivation, these methods struggle to match the performance of BP on challenging tasks and deep architectures [24].

In addition to biological plausibility, another line of BP-free research promotes parallel learning across layers. Decoupled Neural Interfaces [7] introduce synthetic gradients that allow each layer to update independently. Similarly, SoftHebb [45] employs local Hebbian learning within soft winner-take-all

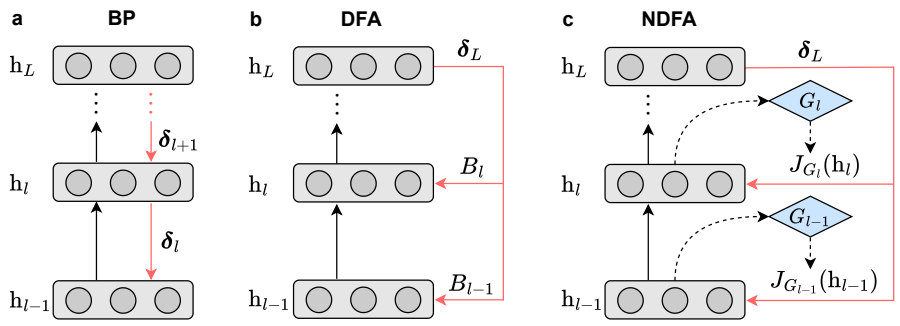


Figure 1: Comparison of (a) BP, (b) DFA, and (c) the proposed NDFA method. BP propagates errors sequentially from the top to bottom layers, leading to the backward locking problem. DFA directly projects the global output error δ_L to each hidden layer l using fixed random feedback matrices B_l. In contrast, NDFA constructs representation-dependent feedback through learnable nonlinear surrogate subnetworks G_l.

networks, while other local learning approaches [46–49] train each hidden layer using an auxiliary network and a local objective. Although these methods improve training parallelism, the lack of cross-layer coordination and global objective guidance often limits their overall performance [50].

Recent studies have attempted to eliminate backward passes through forward-only learning [51, 52]. Present the Error to Perturb the Input To modulate Activity (PEPITA) [51] uses an error-driven extra forward pass, while Forward-Forward [53] replaces BP with two forward passes over positive and negative samples and trains networks using local goodness functions. Forward gradient methods [54–56], in contrast, estimate gradients through directional derivatives computed via forward-mode automatic differentiation. While these approaches offer improved memory efficiency, they often suffer from high-variance gradient estimates and noticeable performance gaps compared with BP.

3 Background

3.1 Backpropagation (BP)

Considering a fully connected neural network with L layers, an input $\mathbf{h}_0$ is propagated sequentially through the network. For each layer $l \in \{1, \dots, L\}$, the forward pass is given by:

$$\mathbf{a}_l = \mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l, \qquad \mathbf{h}_l = \sigma(\mathbf{a}_l), \tag{1}$$

where $\mathbf{W}_l \in \mathbb{R}^{d_l \times d_{l-1}}$ and $\mathbf{b}_l \in \mathbb{R}^{d_l}$ denote the weight and bias at layer l, respectively. $\mathbf{a}_l \in \mathbb{R}^{d_l}$ represents the pre-activation, $\sigma(\cdot)$ denotes a nonlinear activation function, and $\mathbf{h}_l \in \mathbb{R}^{d_l}$ is the layer's output representation. The network produces a prediction $\hat{\mathbf{y}} = \mathbf{a}_L \in \mathbb{R}^{d_y}$ in the output layer, and a loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ is then used to measure the difference between the prediction and the target $\mathbf{y}$.

The BP algorithm [1] computes the gradients of the loss function with respect to network parameters by sequentially propagating error signals from the output layer to each hidden layer. Let $\boldsymbol{\delta}_l = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_l}$ denote the hidden-layer error at layer l. The global output error, $\boldsymbol{\delta}_L = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_L} \in \mathbb{R}^{d_y}$, can be directly computed by the loss function. For hidden layers, the errors are propagated backward according to:

$$\boldsymbol{\delta}_l^{\mathrm{BP}} = \left(\mathbf{W}_{l+1}^\top \boldsymbol{\delta}_{l+1}^{\mathrm{BP}} \right) \odot \sigma'(\mathbf{a}_l), \quad l = L-1, \dots, 1, \tag{2}$$

where $\odot$ denotes the element-wise product and $\sigma'(\cdot)$ is the derivative of the activation function.

Using these errors, the gradients with respect to weight matrices of hidden layers are computed as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_l} = \boldsymbol{\delta}_l^{\mathrm{BP}} \mathbf{h}_{l-1}^\top, \qquad l = 1, \dots, L. \tag{3}$$

While BP is the most widely adopted learning algorithm, it suffers from the backward locking problem [7], where the computation of weight gradients at a hidden layer must wait until errors from all downstream layers have been computed. This dependency prevents parallelization across layers.

3.2 Direct feedback alignment (DFA)

DFA [12] is an alternative learning algorithm that addresses the backward locking problem in BP. Instead of propagating errors sequentially on a layer-by-layer basis, DFA computes the error at each hidden layer by directly projecting the global output error $\boldsymbol{\delta}_L$ using fixed random feedback matrices. Formally, let $\mathbf{B}_l \in \mathbb{R}^{d_l \times d_y}$ denote a fixed feedback matrix at layer l, and the error gradient is computed as:

$$\boldsymbol{\delta}_l^{\mathrm{DFA}} = (\mathbf{B}_l \boldsymbol{\delta}_L) \odot \sigma'(\mathbf{a}_l), \qquad l = 1, \ldots, L-1. \tag{4}$$

The corresponding weight gradients are then computed in the same manner as in Eq. (3). By directly broadcasting the global output error to all hidden layers, DFA eliminates the dependency on error propagation across layers, thereby enabling the hidden layers to be updated in parallel.

4 Method

We first analyze the feedback mismatch between DFA and BP in Section 4.1, showing that the feedback matrices in DFA are representation-independent and limited in capturing the nonlinear transformations from hidden representations to the network output. To address these limitations, we propose nonlinear direct feedback alignment (NDFA), which constructs each hidden-layer feedback via a learnable nonlinear surrogate subnetwork. Section 4.2 details the surrogate subnetwork and the resulting feedback, and Section 4.3 presents the learning strategy for the surrogate subnetwork.

4.1 Feedback mismatch between DFA and BP

We begin by analyzing the mismatch between DFA [12] and BP [1] in projecting the global output error to each hidden layer, which is referred to as feedback. For each hidden layer l, let $F_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_y}$ denote the downstream subnetwork induced by all subsequent layers, which transforms the hidden representation $\mathbf{h}_l$ into the network output $\hat{\mathbf{y}} = F_l(\mathbf{h}_l)$. The hidden-layer errors computed by BP in Eq. (2) can be reformulated as:

$$\boldsymbol{\delta}_l^{\mathrm{BP}} = (J_{F_l}(\mathbf{h}_l)\, \boldsymbol{\delta}_L) \odot \sigma'(\mathbf{a}_l), \quad l = L-1, \ldots, 1, \tag{5}$$

which propagates the global output error $\boldsymbol{\delta}_L$ through the Jacobian of the downstream subnetwork. In Eq. (5), $J_{F_l}(\mathbf{h}_l) = (\partial F_l(\mathbf{h}_l)/\partial \mathbf{h}_l)^\top \in \mathbb{R}^{d_l \times d_y}$ is the transposed Jacobian of F_l at $\mathbf{h}_l$. The term $J_{F_l}(\mathbf{h}_l)$ acts as the feedback, and it is a local linearization of the nonlinear subnetwork F_l around the current hidden representation $\mathbf{h}_l$. Importantly, the feedback depends dynamically on both $\mathbf{h}_l$ and F_l. DFA departs from this behavior by replacing the Jacobian with a fixed random matrix $\mathbf{B}_l \in \mathbb{R}^{d_l \times d_y}$ that is independent of both $\mathbf{h}_l$ and F_l.

Additionally, the feedback $\mathbf{B}_l$ in DFA implicitly induces a linear transformation $\mathbf{h}_l \mapsto \mathbf{B}_l^\top \mathbf{h}_l$ from the hidden representation space to the network output space. In contrast, the downstream subnetwork F_l in BP is highly nonlinear, as it consists of multiple layers and nonlinear activations. This mismatch between a simple linear transformation and a complex nonlinear subnetwork imposes an intrinsic limitation on DFA's ability to approximate BP. As network depth increases, the increasing complexity of F_l amplifies this mismatch.

The analysis of feedback mismatch between DFA and BP suggests two desirable properties for effective feedback. First, the feedback should be *representation-dependent*, allowing the hidden-layer error to reflect the state of the network at each hidden representation. Second, its induced transformation should be *nonlinear*, enabling it to better approximate F_l.

4.2 Representation-dependent feedback via nonlinear surrogate subnetworks

Guided by the above analysis, we aim to design the feedback that is representation-dependent and induced by a nonlinear transformation. Intuitively, this requires approximating the downstream subnetwork with a lightweight model, so that the feedback can vary with hidden representations without incurring excessive computational cost.

To formalize this idea, we generalize the feedback from a static matrix to a function of hidden representations. Specifically, for each hidden layer l, we introduce a function $A_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_l \times d_y}$, which produces the feedback $A_l(\mathbf{h}_l)$ dependent on $\mathbf{h}_l$. This formulation unifies both BP and DFA

as special cases: BP corresponds to the ideal but intractable choice $A_l(\mathbf{h}_l) = J_{F_l}(\mathbf{h}_l)$, while DFA corresponds to the degenerate static choice $A_l(\mathbf{h}_l) \equiv \mathbf{B}_l$. Our goal is to design tractable feedback that lies between these two extremes.

To this end, we introduce a nonlinear surrogate subnetwork that approximates the true downstream subnetwork, and construct feedback from its Jacobian. Concretely, we associate each hidden layer l with a lightweight surrogate subnetwork $G_l(\mathbf{h}_l; \boldsymbol{\theta}_l) : \mathbb{R}^{d_l} \to \mathbb{R}^{d_y}$, where $\boldsymbol{\theta}_l$ denotes its parameters. The feedback is expressed as $A_l(\mathbf{h}_l) = J_{G_l}(\mathbf{h}_l) = (\partial G_l(\mathbf{h}_l)/\partial \mathbf{h}_l)^\top$. The resulting hidden-layer errors in NDFA are computed as follows:

$$\boldsymbol{\delta}_l^{\mathrm{NDFA}} = (J_{G_l}(\mathbf{h}_l)\boldsymbol{\delta}_L) \odot \sigma'(\mathbf{a}_l), \qquad l = 1, \ldots, L-1. \tag{6}$$

To strike a balance between expressiveness and computational efficiency, we instantiate $G_l(\mathbf{h}_l; \boldsymbol{\theta}_l)$ as a multilayer perceptron (MLP), which provides a lightweight surrogate with strong empirical performance, as further discussed in Appendix E:

$$G_l(\mathbf{h}_l; \boldsymbol{\theta}_l) = \mathbf{M}_{l,2}\, \sigma\big(\mathbf{M}_{l,1}\mathbf{h}_l\big), \tag{7}$$

where $\boldsymbol{\theta}_l = \{\mathbf{M}_{l,1} \in \mathbb{R}^{r \times d_l},\ \mathbf{M}_{l,2} \in \mathbb{R}^{d_y \times r}\}$, and r controls the hidden width.

This design yields two key advantages. First, the surrogate subnetwork is nonlinear and better aligns with the structure of the downstream subnetwork. Second, the resulting feedback is representation-dependent, allowing the hidden-layer errors to adapt dynamically to hidden representations.

4.3 Learning nonlinear surrogate subnetworks for effective feedback

The effectiveness of the proposed feedback depends on how well the surrogate subnetwork G_l approximates the true subnetwork F_l. However, directly matching G_l to F_l is difficult, since F_l is determined by the subsequent layers that are constantly updated during training.

To address this, we train the surrogate G_l to approximate the input-output behavior of the true subnetwork F_l on the hidden representations encountered during training. This is motivated by the fact that, despite its evolving form, the role of F_l is to map hidden representations to the target space. By aligning G_l with this functional behavior, the surrogate could track F_l over the course of training.

Formally, for each hidden layer l, we define a feedback adaptation objective over the empirical distribution of hidden representations:

$$\mathcal{L}_l^{\mathrm{fb}} = \mathbb{E}_{\mathbf{h}_l \sim \mathcal{D}_l}\big[\, \mathcal{L}\big(G_l(\mathrm{sg}(\mathbf{h}_l)),\, \mathbf{y}\big)\,\big], \tag{8}$$

where $\mathcal{D}_l$ denotes the distribution of $\mathbf{h}_l$ on the training data, $\mathcal{L}(\cdot,\cdot)$ is the same loss function as the main network, and $\mathbf{y}$ is the corresponding ground-truth label. The operator $\mathrm{sg}(\cdot)$ denotes the stop-gradient operation, which prevents gradients from propagating into the main network. The parameters of the surrogate subnetwork are updated as follows:

$$\boldsymbol{\theta}_l \leftarrow \boldsymbol{\theta}_l - \eta\, \nabla_{\boldsymbol{\theta}_l} \mathcal{L}_l^{\mathrm{fb}}, \tag{9}$$

where η denotes the learning rate shared by the main network and surrogate subnetworks. Notably, the gradients of $\mathcal{L}_l^{\mathrm{fb}}$ are only used to update $\boldsymbol{\theta}_l$ and do not affect the parameters of the main network.

It is worth noting that, in NDFA, the feedback computation and surrogate subnetwork update for each hidden layer can be performed in parallel with the forward pass of subsequent layers. Once the forward pass is completed and the global output error becomes available, NDFA uses the precomputed feedback to directly project the global output error into each hidden layer, thereby enabling parallel training across layers. The pseudocode is provided in Appendix Algorithm 1.

5 Experiments

5.1 Experimental setup

Datasets. Our evaluation uses four commonly used image classification datasets, including CIFAR-10 [31], CIFAR-100 [31], SVHN [32], and ImageNet [33].

Architectures. We use four representative architectures, including a ConvNet with two convolutional layers and two fully connected layers, AlexNet [34], ResNet-32 [35], and ViT-Tiny [36].

Table 1: Comparison of different methods on CIFAR-10, CIFAR-100, and SVHN using ConvNet.

Method	CIFAR-10	CIFAR-100	SVHN
BP	81.36±0.32	51.40±0.25	93.41±0.12
DFA	71.59±0.77	36.03±0.95	89.09±1.15
DKP	73.04±0.13	37.53±0.71	88.34±1.62
PEPITA	56.80±0.56	22.93±1.62	78.71±0.63
FDFA	74.32±0.45	41.84±1.00	93.42±0.05
NDFA	**78.62**±0.46	**47.86**±0.54	**93.49**±0.09

Table 2: Comparison of different methods using AlexNet.

Method	CIFAR-10	CIFAR-100
BP	90.95±0.23	61.25±0.11
DFA	80.68±0.11	48.00±0.15
DKP	83.63±0.19	58.00±0.10
PEPITA	53.03±0.16	21.03±0.21
FDFA	84.65±0.12	60.70±0.31
NDFA	**87.38**±0.14	**61.74**±0.37

Table 3: Comparison of different methods using ResNet-32. K is the number of parallel modules, where $K = 16$ represents the layer-wise setting.

Method	$K = 4$			$K = 16$		
	CIFAR-10	CIFAR-100	SVHN	CIFAR-10	CIFAR-100	SVHN
BP (K=1)	92.23±0.22	67.98±0.22	96.71±0.09	92.23±0.22	67.98±0.22	96.71±0.09
DFA	72.73±0.31	27.17±0.59	90.85±0.33	60.62±2.19	15.24±0.46	80.93±1.07
DKP	63.91±0.56	21.84±0.78	90.02±0.70	51.04±0.61	18.59±0.77	65.92±1.32
FDFA	80.63±0.35	49.12±0.63	93.25±0.32	77.36±0.23	48.53±0.22	92.77±0.83
NDFA	**85.36**±0.48	**55.38**±0.38	**94.82**±0.04	**81.53**±0.60	**50.45**±0.25	**93.42**±0.19

Baselines. We compare NDFA with BP and existing DFA-based methods, including DFA [12], DKP [28], PEPITA [51], and FDFA [29]. DFA directly broadcasts the global output error to each hidden layer through fixed random feedback matrices. DKP updates the feedback matrices using information from the forward pathway. PEPITA perturbs the input using the global output error through an extra forward pass and can be viewed as a DFA-like algorithm. FDFA updates the feedback matrices using forward-mode gradient estimates.

Implementation details. For fair comparisons, we adopt consistent training settings across all methods, except for the learning rate, which is searched for each method. We compare NDFA with baselines in a module-wise setting (see Appendix A), where the network is divided into K modules. In this setting, DFA-based methods directly propagate the global output error to each module while allowing gradient propagation within individual modules. All reported results are averaged over three independent runs. Detailed training settings and module partitions are provided in Appendix A.

5.2 Main results

NDFA consistently outperforms DFA-based baselines across diverse datasets. As shown in Tables 1 and 2, NDFA achieves the best performance among all DFA-based methods across CIFAR-10, CIFAR-100, and SVHN on both ConvNet and AlexNet, substantially reducing the gap to BP. In particular, for ConvNet, NDFA surpasses FDFA by more than 4% and 6% accuracy on CIFAR-10 and CIFAR-100, respectively. These results demonstrate the effectiveness of the proposed NDFA method.

NDFA consistently outperforms DFA-based baselines on deep architectures. Table 3 reports the test accuracy of ResNet-32 on CIFAR-10, CIFAR-100, and SVHN under different numbers of parallel modules K. NDFA consistently achieves the strongest performance among all DFA-based methods across different datasets and parallelization granularities. Notably, its performance remains robust as the number of parallel modules increases, demonstrating the effectiveness of NDFA under more challenging parallel training settings. We further evaluate NDFA on ViT-Tiny to examine its scalability to transformer architectures. As shown in Table 4, NDFA substantially improves the performance of DFA on both CIFAR-10 and CIFAR-100. These results demonstrate that NDFA generalizes effectively across different architectures.

NDFA scales effectively to ImageNet. We further evaluate the scalability of NDFA on the large-scale ImageNet dataset using ResNet-32 and ViT-Tiny. As shown in Table 5, NDFA substantially outperforms DFA on both architectures. In particular, NDFA improves the accuracy of ResNet-32 by more than 17%. The improvement is even more significant for ViT-Tiny, where DFA fails to train effectively. These results demonstrate the strong scalability of NDFA to large-scale datasets

Table 4: Comparison of different methods on CIFAR-10/100 using ViT-Tiny.

Method	K	CIFAR-10	CIFAR-100
BP	1	92.26±0.23	66.08±0.13
DFA	5	28.76±0.63	7.00±0.26
NDFA	5	**91.15**±0.06	**65.22**±0.75
DFA	13	24.31±0.65	4.69±0.37
NDFA	13	**77.88**±0.80	**48.91**±0.75

Table 5: Results on ImageNet using ResNet-32 and ViT-Tiny.

Method	ResNet-32		ViT-Tiny	
	K	Acc.	K	Acc.
BP	1	69.22	1	73.58
DFA	3	44.20	3	0.72
NDFA	3	**61.28**	3	**66.27**

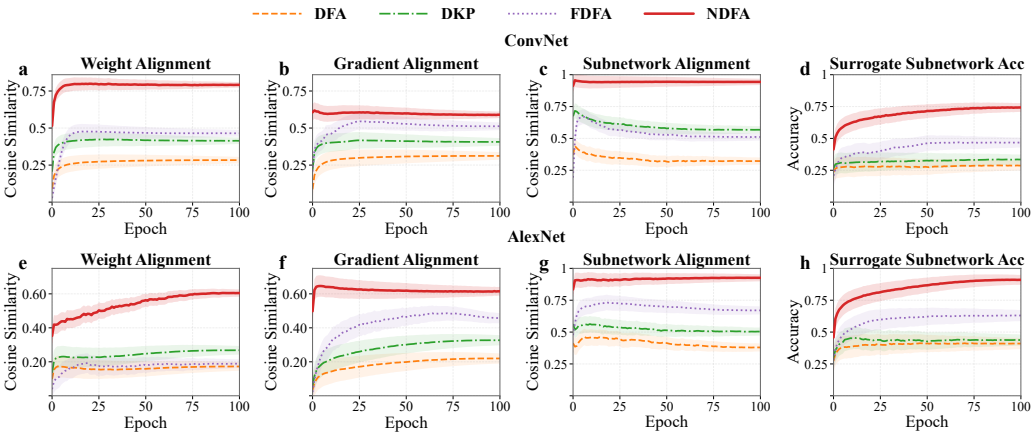


Figure 2: Comparison of feedback-related analyses between NDFA and other DFA-based methods on CIFAR-10 using ConvNet and AlexNet. Panels (a,e), (b,f), (c,g), and (d,h) show weight alignment, gradient alignment, subnetwork alignment, and surrogate subnetwork accuracy, respectively.

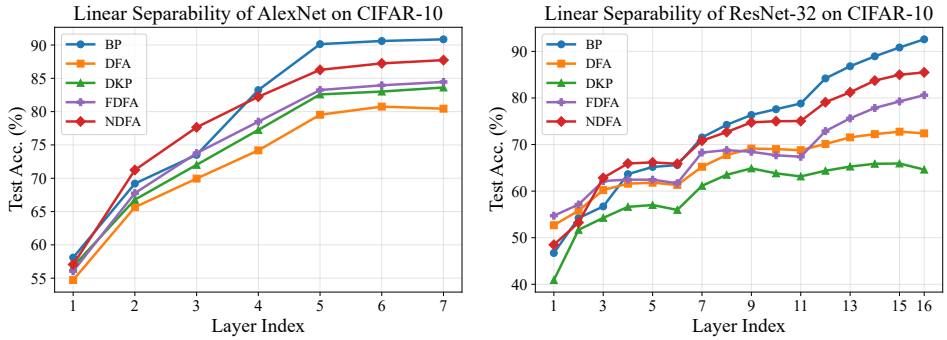


Figure 3: Comparison of layer-wise linear separability among different methods using AlexNet and ResNet-32 on CIFAR-10.

5.3 Analysis of feedback and representations

5.3.1 Feedback analysis

We compare the feedback in NDFA against other DFA-based methods using weight alignment, gradient alignment, subnetwork alignment, and surrogate subnetwork accuracy. Jacobian alignment is provided in Appendix C. ConvNet and AlexNet are trained on CIFAR-10.

Weight alignment measures the cosine similarity between the feedback and the corresponding forward weight matrix, following previous studies that assume the transformation from a hidden representation to the network output can be approximated by a linear mapping. As shown in Figures 2 (a,e), NDFA achieves substantially stronger weight alignment than existing DFA-based methods.

Moreover, the alignment consistently improves throughout training, indicating that the learned feedback in NDFA progressively aligns more closely with the corresponding forward weight.

Gradient alignment measures the cosine similarity between the hidden-layer errors produced by each DFA-based method and BP. As shown in Figure 2 (b, f), NDFA consistently achieves higher gradient alignment with BP than existing DFA-based methods throughout training. This is because NDFA provides a better structural approximation of the Jacobian of BP than baselines. This suggests that the feedback in NDFA produces hidden-layer update directions that more closely approximate those of BP, thereby contributing to its superior performance.

Subnetwork alignment evaluates how well the surrogate subnetwork G_l approximates the true downstream subnetwork F_l. Specifically, for the same hidden representation $\mathbf{h}_l$, we compute the cosine similarity between the surrogate output $G_l(\mathbf{h}_l)$ and the true output $F_l(\mathbf{h}_l)$. As shown in Figure 2 (c,g), NDFA achieves the highest subnetwork alignment among all methods. This indicates that the nonlinear surrogate subnetworks in NDFA closely approximate the input-output behavior of the true subnetworks. Note that a more accurate approximation of the downstream subnetwork F_l by its surrogate G_l does not necessarily imply better alignment between their Jacobians. This is shown in Figure 2 (f,g), where subnetwork alignment gradually increases across epochs, whereas gradient alignment remains stable under NDFA.

Surrogate subnetwork accuracy measures the prediction accuracy of the surrogate output $G_l(\mathbf{h}_l)$ with respect to the ground-truth label $\mathbf{y}$, reflecting how well the surrogate subnetwork learns the predictive function of the corresponding downstream subnetwork. As shown in Figure 2 (d,h), NDFA achieves substantially higher surrogate subnetwork accuracy than existing DFA-based methods, with the accuracy consistently improving throughout training. This indicates that the nonlinear surrogate subnetworks in NDFA can better capture task-relevant downstream transformations, thereby providing more informative feedback signals for learning hidden layers.

5.3.2 Representation analysis

We further evaluate the representation quality learned by NDFA using linear probing on CIFAR-10 with AlexNet and ResNet-32. Specifically, we freeze the trained network and train a linear classifier on the representation of each layer. The accuracy of the linear classifier is used to measure the layer-wise linear separability. Figure 3 shows that NDFA consistently learns more discriminative representations than existing DFA-based methods across both architectures. In particular, NDFA exhibits clear advantages in middle and deep layers, while exhibiting representation trends closer to BP. These results suggest that the feedback in NDFA provides more effective hidden-layer supervision, enabling the network to learn more linearly separable representations.

5.4 Parallel training efficiency

Unlike BP, DFA-based methods directly propagate the global output error to each hidden layer without backward locking. Consequently, the gradient computation of different hidden layers can be executed in parallel, potentially leading to significant training acceleration. To evaluate the parallel training efficiency of different methods, we estimate their backward computation cost by using the maximum backward time among all hidden layers. For NDFA, the surrogate subnetworks introduce additional computation overhead. As shown in Appendix Algorithm 1, these additional computations can in principle be executed concurrently with the forward pass of subsequent layers. However, to provide a clearer evaluation of the overhead introduced by NDFA, we conservatively include these computations in the measured backward time rather than overlapping them with the forward pass.

Table 6 compares the backward computation time of different methods on AlexNet and ResNet-32. Compared with BP, all DFA-based methods achieve substantial reductions in backward computation cost due to their parallel training property. Although NDFA introduces additional overhead compared with DFA, it still achieves significant acceleration over BP and FDFA while substantially outperforming existing DFA-based methods in accuracy. These results demonstrate that NDFA provides a favorable balance between parallel training efficiency and model performance.

Table 6: Per-iteration backward time (ms) of different methods on CIFAR-10. Values in parentheses denote the speedup relative to BP.

Method	AlexNet	ResNet-32
BP	2.34	3.45
DFA	0.25 (9.36×)	0.24 (14.4×)
DKP	0.33 (7.09×)	0.37 (9.32×)
FDFA	1.28 (1.83×)	1.11 (3.11×)
NDFA	0.92 (2.54×)	0.78 (4.42×)

Table 7: Ablation study of NDFA using ConvNet and ResNet-32 on CIFAR-10 and CIFAR-100.

Network	Learnable	Nonlinear	CIFAR-10	CIFAR-100
ConvNet	✗	✗	71.59±0.77	36.03±0.95
	✓	✗	73.17±0.35	39.47±0.67
	✗	✓	74.60±0.09	41.65±0.15
	✓	✓	**78.62**±0.46	**47.86**±0.54
ResNet-32	✗	✗	72.73±0.31	27.17±0.59
	✓	✗	73.78±0.67	36.40±0.31
	✗	✓	76.04±0.38	40.10±0.86
	✓	✓	**85.36**±0.48	**55.38**±0.38

5.5 Ablation study

We conduct ablation studies to investigate the effectiveness of the two key components of NDFA. Linear feedback is used to denote basic matrix as feedback. Table 7 summarizes the results on ConvNet and ResNet-32 over CIFAR-10 and CIFAR-100. First, introducing learnable feedback consistently improves performance over fixed feedback across both architectures and datasets, highlighting the importance of adapting the feedback to the evolving downstream subnetworks during training. Second, nonlinear feedback substantially outperforms linear feedback regardless of whether the feedback is learnable, demonstrating the importance of modeling nonlinear downstream transformations. Finally, combining both learnable and nonlinear feedback achieves the best performance across all settings, indicating that the two components are complementary for improving feedback quality. Overall, these results demonstrate that both learnable and nonlinear feedback are essential for effective NDFA training.

5.6 Neural data similarity

Biologically inspired learning rules are expected not only to achieve strong task performance, but also to produce representations that resemble biological neural recordings. To evaluate the neural data similarity of NDFA, we conduct experiments on the Mante 2013 dataset [37], a widely used neuroscience benchmark containing neural recordings from the prefrontal cortex during a context-dependent decision-making task. Following the experimental setup in [57], we train recurrent neural networks on this dataset using BPTT and NDFA. Implementation details are provided in Appendix D. To quantify neural similarity, we compare the representations of trained models with recorded neural data using Procrustes analysis, where a smaller Procrustes distance indicates stronger alignment with neural recordings. Results are averaged over three independent runs.

Figure 4 shows the relationship between normalized task accuracy and Procrustes distance throughout training. As task performance improves, both BPTT and NDFA progressively approach the neural data. Notably, in the high-accuracy regime, NDFA consistently achieves lower Procrustes distance than BPTT, indicating that the representations learned by NDFA more closely resemble the recorded neural data. These results suggest that NDFA can achieve competitive task performance while exhibiting stronger neural data similarity.

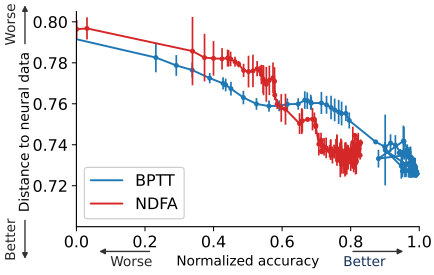


Figure 4: Comparison of neural data similarity between BPTT and NDFA on the Mante 2013 dataset [37].

6 Conclusion

In this paper, we revisit the feedback mismatch between DFA and BP and show that the fixed feedback matrices in DFA are limited in capturing the nonlinear downstream transformations of deep networks. Motivated by this analysis, we propose NDFA, which constructs hidden-layer feedback through learnable nonlinear surrogate subnetworks. Extensive experiments across convolutional networks, residual networks, and vision transformers demonstrate that NDFA consistently outperforms existing

DFA-based methods while maintaining parallel training across layers. Further analyses demonstrate that NDFA achieves stronger alignment with BP and learns more discriminative intermediate representations. In addition, experiments on neural data similarity show that NDFA produces representations that more closely resemble biological neural recordings.

Limitations. Although NDFA substantially narrows the performance gap between DFA and BP, an observable gap to BP still remains on very deep architectures. Future work could investigate more expressive surrogate subnetworks to further improve feedback quality. In addition, NDFA introduces extra computational overhead due to the surrogate subnetworks. While these computations can in principle be executed concurrently with the forward pass, the practical training efficiency of NDFA depends on efficient parallel implementations and hardware support. Exploring asynchronous implementations and system-level acceleration strategies is an important direction for future work. Furthermore, a rigorous theoretical analysis of convergence would provide a deeper understanding of the proposed method, and we leave it for future work. Exploring alternatives to lightweight surrogate subnetworks for constructing the feedback $A_l(\mathbf{h}_l)$ is also a promising future direction.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China under Grants 62606328, 62602434, 62476170, and 62376163; and in part by the Shenzhen Science and Technology Program under Grant JCYJ20250604181716022. This work is supported by the Intelligent Computing Center of Shenzhen University.

References

[1] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[3] Hongzhi Wang, Xiubo Liang, Tao Zhang, Gu Yue, and Weidong Geng. PSSD-transformer: Powerful sparse spike-driven transformer for image semantic segmentation. In *ACM Multimedia 2024*, 2024.

[4] Hongzhi Wang, Xiubo Liang, Jinxing Han, and Weidong Geng. Spike-retinexformer: Rethinking low-light image enhancement with spiking neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[5] Xiubo Liang, Jinxing Han, Yuke Li, Haoqi Zhu, Yu Zhao, and Hongzhi Wang. Spike-HTR: Spiking neural transformer for handwritten text recognition. In *Forty-third International Conference on Machine Learning*, 2026.

[6] Xiubo Liang, Jinxing Han, Yuke Li, Haoqi Zhu, Yu Zhao, and Hongzhi Wang. SMM transformer: Leveraging spiking neural networks for multimodal tasks. In *Forty-third International Conference on Machine Learning*, 2026.

[7] 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, 2017.

[8] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[9] Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, et al. A deep learning framework for neuroscience. *Nature Neuroscience*, 22(11):1761–1770, 2019.

[10] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[11] Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: a Hebbian learning rule. *Annual Review of Neuroscience*, 31(1):25–46, 2008.

[12] Arild Nø kland. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[13] Pieter R Roelfsema and Anthony Holtmaat. Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19(3):166–180, 2018.

[14] David Clark, LF Abbott, and SueYeon Chung. Credit assignment through broadcasting a global error vector. *Advances in Neural Information Processing Systems*, 34:10053–10066, 2021.

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

[16] Mitsumasa Nakajima, Katsuma Inoue, Kenji Tanaka, Yasuo Kuniyoshi, Toshikazu Hashimoto, and Kohei Nakajima. Physical deep learning with biologically inspired training method: gradient-free approach for physical hardware. *Nature Communications*, 13(1):7847, 2022.

[17] Matthew J Filipovich, Zhimu Guo, Mohammed Al-Qadasi, Bicky A Marquez, Hugh D Morison, Volker J Sorger, Paul R Prucnal, Sudip Shekhar, and Bhavin J Shastri. Silicon photonic architecture for training deep neural networks with direct feedback alignment. *Optica*, 9(12): 1323–1332, 2022.

[18] Jeongjun Lee, Renqian Zhang, Wenrui Zhang, Yu Liu, and Peng Li. Spike-train level direct feedback alignment: Sidestepping backpropagation for on-chip training of spiking neural nets. *Frontiers in Neuroscience*, 14:143, 2020.

[19] Haoxiong Ren, Yangu He, Kwunhang Wong, Rui Bao, Ning Lin, Zhongrui Wang, and Dashan Shang. When pipelined in-memory accelerators meet spiking direct feedback alignment: A co-design for neuromorphic edge computing. In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2025.

[20] Ziao Wang, Kilian Müller, Matthew Filipovich, Julien Launay, Ruben Ohana, Gustave Pariente, Safa Mokaadi, Charles Brossollet, Fabien Moreau, Alessandro Cappelli, et al. Streamlined optical training of large-scale modern deep learning architectures with direct feedback alignment. *arXiv preprint arXiv:2409.12965*, 2024.

[21] Julien Launay, Iacopo Poli, François Boniface, and Florent Krzakala. Direct feedback alignment scales to modern deep learning tasks and architectures. In *Advances in Neural Information Processing Systems*, volume 33, pages 9346–9360, 2020.

[22] Maria Refinetti, Stéphane D'Ascoli, Ruben Ohana, and Sebastian Goldt. Align, then memorise: the dynamics of learning with feedback alignment. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8925–8935, 18–24 Jul 2021.

[23] Yunseok Lee and Seulki Lee. Enabling fine-tuning of direct feedback alignment via feedback-weight matching. In *The Fourteenth International Conference on Learning Representations*, 2026.

[24] 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*, volume 31, 2018.

[25] Brian Crafton, Abhinav Parihar, Evan Gebhardt, and Arijit Raychowdhury. Direct feedback alignment with sparse connections for local learning. *Frontiers in Neuroscience*, 13:525, 2019.

[26] Julien Launay, Iacopo Poli, and Florent Krzakala. Principled training of neural networks with direct feedback alignment. *arXiv preprint arXiv:1906.04554*, 2019.

[27] Theodore H Moskovitz, Ashok Litwin-Kumar, and LF Abbott. Feedback alignment in deep convolutional networks. *arXiv preprint arXiv:1812.06488*, 2018.

[28] Matthew Bailey Webster, Jonghyun Choi, and Changwook Ahn. Learning the connections in direct feedback alignment, 2021.

[29] Florian Bacho and Dominique Chu. Low-variance forward gradients using direct feedback alignment and momentum. *Neural Networks*, 169:572–583, 2024. doi: https://doi.org/10.1016/j.neunet.2023.10.051.

[30] Paul Caillon, Erwan Fagnou, Blaise Delattre, and Alexandre Allauzen. Scaling direct feedback learning with Jacobian alignment guarantees. In *The Fourteenth International Conference on Learning Representations*, 2026.

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[32] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011(2):4, 2011.

[33] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[36] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[37] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, 2013.

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

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

[40] 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(1):13276, 2016.

[41] Mohamed Akrout, Collin Wilson, Peter Humphreys, Timothy Lillicrap, and Douglas B Tweed. Deep learning without weight transport. *Advances in Neural Information Processing Systems*, 32, 2019.

[42] Charlotte Frenkel, Martin Lefebvre, and David Bol. Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks. *Frontiers in Neuroscience*, 15:629892, 2021.

[43] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 498–515, 2015.

[44] Maxence M Ernoult, Fabrice Normandin, Abhinav Moudgil, Sean Spinney, Eugene Belilovsky, Irina Rish, Blake Richards, and Yoshua Bengio. Towards scaling difference target propagation by learning backprop targets. In *International Conference on Machine Learning*, pages 5968–5987. PMLR, 2022.

[45] Adrien Journé, Hector Garcia Rodriguez, Qinghai Guo, and Timoleon Moraitis. Hebbian deep learning without feedback. In *The Eleventh International Conference on Learning Representations*, 2023.

[46] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *International Conference on Machine Learning*, pages 583–593, 2019.

[47] Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. In *International Conference on Machine Learning*, pages 4839–4850, 2019.

[48] Chenxiang Ma, Rui Yan, Zhaofei Yu, and Qiang Yu. Deep spike learning with local classifiers. *IEEE Transactions on Cybernetics*, 53(5):3363–3375, 2023. doi: 10.1109/TCYB.2022.3188015.

[49] Chenxiang Ma, Xinyi Chen, Kay Chen Tan, and Jibin Wu. Spatiotemporal decoupled learning for spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 37 (9):4597–4611, 2026. doi: 10.1109/TNNLS.2026.3671461.

[50] Chenxiang Ma, Jibin Wu, Chenyang Si, and KC Tan. Scaling supervised local learning with augmented auxiliary networks. In *International Conference on Learning Representations*, volume 2024, pages 17334–17351, 2024.

[51] Giorgia Dellaferrera and Gabriel Kreiman. Error-driven input modulation: Solving the credit assignment problem without a backward pass. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 4937–4955, 17–23 Jul 2022.

[52] Ravi Francesco Srinivasan, Francesca Mignacco, Martino Sorbaro, Maria Refinetti, Avi Cooper, Gabriel Kreiman, and Giorgia Dellaferrera. Forward learning with top-down feedback: Empirical and analytical characterization. In *The Twelfth International Conference on Learning Representations*, 2024.

[53] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2(3):5, 2022.

[54] Atılım Güneş Baydin, Barak A Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*, 2022.

[55] Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling forward gradient with local losses. In *The Eleventh International Conference on Learning Representations*, 2023.

[56] Louis Fournier, Stéphane Rivaud, Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Can forward gradient match backpropagation? In *International Conference on Machine Learning*, pages 10249–10264, 2023.

[57] Yuhan Helena Liu, Guangyu Robert Yang, and Christopher J Cueva. Can biologically plausible temporal credit assignment rules match BPTT for neural similarity? e-prop as an example. In *Forty-second International Conference on Machine Learning*, 2025.

Algorithm 1 Nonlinear Direct Feedback Alignment (NDFA)

Require: Main network $\{f_l(\cdot;\mathbf{W}_l)\}_{l=1}^{L}$; surrogate subnetworks $\{G_l(\cdot;\boldsymbol{\theta}_l)\}_{l=1}^{L-1}$; learning rate η

for each training iteration **do**
  Sample a mini-batch $(\mathbf{x}, \mathbf{y})$
  $\mathbf{h}_0 = \mathbf{x}$
  for $l = 1$ to L **do** ▷ Forward pass
    $\mathbf{a}_l = f_l(\mathbf{h}_{l-1}; \mathbf{W}_l)$
    if $l < L$ **then**
      $\mathbf{h}_l = \sigma_l(\mathbf{a}_l)$
      Compute $\mathcal{L}_l^{\text{fb}} = \mathcal{L}(G_l(\text{sg}(\mathbf{h}_l)), \mathbf{y})$ ▷ can run in parallel with the forward pass
      Compute $J_{G_l}(\mathbf{h}_l)$ and $\nabla_{\boldsymbol{\theta}_l}\mathcal{L}_l^{\text{fb}}$ ▷ can run in parallel with the forward pass
    else
      $\hat{\mathbf{y}} = \mathbf{a}_L$
    end if
  end for
  $\mathcal{L} = \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$, $\boldsymbol{\delta}_L = \nabla_{\hat{\mathbf{y}}}\mathcal{L}$
  for $l = 1$ to L **in parallel do**
    if $l < L$ **then**
      $\boldsymbol{\delta}_l = (J_{G_l}(\mathbf{h}_l)\boldsymbol{\delta}_L) \odot \sigma_l'(\mathbf{a}_l)$
      $\nabla_{\mathbf{W}_l}\mathcal{L} = \boldsymbol{\delta}_l \mathbf{h}_{l-1}^\top$
      $\mathbf{W}_l = \mathbf{W}_l - \eta\nabla_{\mathbf{W}_l}\mathcal{L}$
      $\boldsymbol{\theta}_l = \boldsymbol{\theta}_l - \eta\nabla_{\boldsymbol{\theta}_l}\mathcal{L}_l^{\text{fb}}$
    else
      $\nabla_{\mathbf{W}_L}\mathcal{L} = \boldsymbol{\delta}_L \mathbf{h}_{L-1}^\top$
      $\mathbf{W}_L = \mathbf{W}_L - \eta\nabla_{\mathbf{W}_L}\mathcal{L}$
    end if
  end for
end for

A Implementation details for image classification experiments

Datasets. We adopt four image classification datasets, including CIFAR-10 [31], CIFAR-100 [31], SVHN [32], and ImageNet [33]. CIFAR-10 is a widely used small-scale natural image classification benchmark consisting of 60,000 color images of size 32×32. It contains 10 classes, with 50,000 training images and 10,000 test images. Following standard practice, we use random crop with 4-pixel padding and random horizontal flip for training. CIFAR-100 has the same image format and data split as CIFAR-10, but contains 100 classes. This makes it more challenging due to its finer semantic granularity and higher demand on representation learning. SVHN is a real-world digit recognition benchmark composed of RGB images cropped from natural street-view scenes. The dataset contains 32×32 images from 10 digit classes. Although SVHN has limited semantic diversity, it remains challenging due to cluttered backgrounds, illumination changes, and appearance variations in natural scenes. We adopt the standard split with 73,257 training images and 26,032 test images. ImageNet is used as the large-scale image classification benchmark. It contains 1,000 classes, with approximately 1.2 million training images and 50,000 validation images. Training images are cropped to 224×224 and randomly horizontally flipped, while validation images are resized and center-cropped for evaluation.

Models. We evaluate NDFA on a diverse set of architectures, including a ConvNet, AlexNet [34], ResNet-32 [35], and ViT-Tiny [36]. The ConvNet architecture is 15C5-40C5-MP2-128-10, where 15C5 and 40C5 denote convolutional layers with 15 and 40 output channels and 5×5 kernels, MP2 denotes a 2×2 max-pooling layer, 128 denotes a fully connected hidden layer with 128 units, and 10 denotes the output dimension. For a convolutional layer, its output feature map is pooled and flattened before being passed to the first linear layer of the corresponding surrogate subnetwork.

Training setups. We compare NDFA with BP [1] and DFA-based methods, including DFA [12], DKP [28], PEPITA [51], and FDFA [29]. All experiments are conducted on 4 NVIDIA RTX A6000 GPUs. The hyperparameters for different methods and architectures are summarized in Table 8. Since

Table 8: Training hyperparameters used for different methods and architectures.

Architecture	Method	Learning Rate	Optimizer	Batch Size	Epochs
ConvNet	BP	5e-4	AdamW	256	100
	DFA	5e-4	AdamW	256	100
	DKP	8e-4	AdamW	256	100
	PEPITA	5e-2	SGD	256	100
	FDFA	1e-4	AdamW	256	100
	NDFA	1e-3	AdamW	256	100
AlexNet	BP	5e-4	AdamW	256	100
	DFA	1e-4	AdamW	256	100
	DKP	1e-4	AdamW	256	100
	PEPITA	3e-3	SGD	256	100
	FDFA	1e-4	AdamW	256	100
	NDFA	5e-4	AdamW	256	100
ResNet-32	BP	2e-1	SGD	256	200
	DFA	5e-4	AdamW	256	200
	DKP	4e-4	AdamW	256	200
	FDFA	5e-4	AdamW	256	200
	NDFA	1e-3	AdamW	256	200
ViT-Tiny	BP	1e-3	AdamW	256	300
	DFA	1e-3	AdamW	256	300
	NDFA	1e-3	AdamW	256	300

the optimal learning rate varies across different learning rules, we tune the learning rate separately for each method to ensure fair comparisons.

Module-wise settings. In module-wise experiments, the network is partitioned into K parallel modules. For ResNet-32 on CIFAR-10 and CIFAR-100, we evaluate $K = 4$ and $K = 16$. For $K = 4$, the network is divided into the convolutional stem and three residual stages, whereas for $K = 16$, a finer partition is adopted consisting of the convolutional stem and 15 individual residual blocks.

For ViT-Tiny, we evaluate $K = 5$ and $K = 13$. For $K = 5$, the model is divided into the patch embedding module and four groups of three transformer blocks. For $K = 13$, the patch embedding module and each of the 12 transformer blocks are treated as separate modules.

For ImageNet experiments, both ResNet-32 and ViT-Tiny are evaluated with $K = 3$, where the backbone is partitioned into three consecutive macro modules to maintain consistent settings across architectures.

B Additional experiments

Evaluation on different tasks. We evaluate NDFA on a time-series classification dataset (FordA) and a language-modeling dataset (WikiText-2). ViT-Tiny is trained using each evaluated method for 100 epochs, with a batch size of 256 and a learning rate of 0.001. As shown in Table 9, NDFA consistently outperforms DFA and achieves performance comparable to BP, demonstrating its effectiveness across different tasks.

Table 9: Results on FordA and WikiText-2 using ViT-Tiny.

Method	FordA Acc. (↑)	WikiText-2 PPL (↓)
BP	83.46 ± 0.31	84.02 ± 0.58
DFA	72.18 ± 0.74	229.64 ± 4.27
NDFA	$\mathbf{82.71 \pm 0.45}$	$\mathbf{87.96 \pm 1.17}$

Comparison with different BP-free methods. We compare NDFA with Forward-Forward [53], Local Greedy Forward Gradient (LG-FG-A) [55], Decoupled Neural Interfaces (DNI) [7], and PredSim [47] on CIFAR-10 using the ConvNet. We reproduce Forward-Forward, DNI, and PredSim using their code and compare them with NDFA under a consistent experimental setting, using the same architecture, data preprocessing, training protocol, and hyperparameters. Specifically, all methods are trained for 100 epochs using AdamW with a learning rate of 0.001 and a batch size of 256 using the ConvNet architecture. We also provide the results for LG-FG-A, Forward-Forward, and DNI reported in their respective original papers, as denoted by * in Table 10. Forward-Forward was originally evaluated using a four-layer fully connected network, while our experiments show that it performs less effectively on convolutional architectures. LG-FG-A relies on its specially designed LocalMixer architecture and is trained for 200 epochs, twice our training budget. Despite these specialized architectures and larger training budgets, NDFA still achieves higher accuracy than these baselines. As shown in Table 10, NDFA achieves the highest accuracy among all compared methods, outperforming PredSim and the other BP-free baselines.

Table 10: Comparison with different BP-free methods on CIFAR-10 using ConvNet. Results marked with $*$ are copied from their original papers.

Method	Accuracy (%)
Forward-Forward	27.68 ± 0.21
Forward-Forward*	59.00
LG-FG-A*	69.32
DNI	62.68 ± 0.10
DNI*	64.86
PredSim	76.08 ± 0.18
NDFA	$\mathbf{78.62 \pm 0.46}$

C Additional analyses

Jacobian alignment. We evaluate the layer-wise cosine similarity between the Jacobians used by NDFA and BP, i.e., $J_{G_l}(\mathbf{h}_l)$ and $J_{F_l}(\mathbf{h}_l)$, on CIFAR-10 using ConvNet. We also compare NDFA with DFA under the same setting. As shown in Figure 5, NDFA consistently achieves substantially higher Jacobian cosine similarity than DFA across different layers and training epochs. This stronger Jacobian alignment helps explain the considerably higher accuracy achieved by NDFA. We also observe that the Jacobian alignment of NDFA increases for layers closer to the network output. This result is expected because earlier layers are followed by deeper and more complex downstream subnetworks, which are more difficult to approximate using lightweight MLP-based surrogate subnetworks. Note that the first-layer alignment for NDFA decreases substantially after reaching an early peak, because the downstream subnetwork associated with the first layer is deeper than other layers. By contrast, the average Jacobian cosine similarity across all layers declines more moderately and remains at a relatively high level. Moreover, both the layer-wise and average Jacobian cosine similarities of NDFA consistently exceed those of DFA throughout training.

Convergence analysis. We conduct a convergence analysis by reporting the losses of the main network and three surrogate subnetworks on CIFAR-10 using ConvNet. As shown in Table 11, all losses gradually decrease over the course of training, and their standard deviations remain very small, indicating stable optimization across runs. Moreover, the loss decreases for deeper surrogate subnetwork, suggesting that NDFA can exploit network depth to improve performance.

Table 11: Convergence analysis on CIFAR-10 using ConvNet.

Loss	Epoch 1	Epoch 30	Epoch 50	Epoch 80	Epoch 100
Main Network	1.69 ± 0.02	0.74 ± 0.02	0.66 ± 0.02	0.58 ± 0.02	0.57 ± 0.02
1st Surrogate	3.64 ± 0.07	2.26 ± 0.06	2.10 ± 0.05	1.98 ± 0.05	1.96 ± 0.05
2nd Surrogate	3.51 ± 0.04	1.81 ± 0.04	1.63 ± 0.04	1.48 ± 0.04	1.46 ± 0.04
3rd Surrogate	3.42 ± 0.04	1.47 ± 0.04	1.28 ± 0.04	1.11 ± 0.03	1.08 ± 0.03

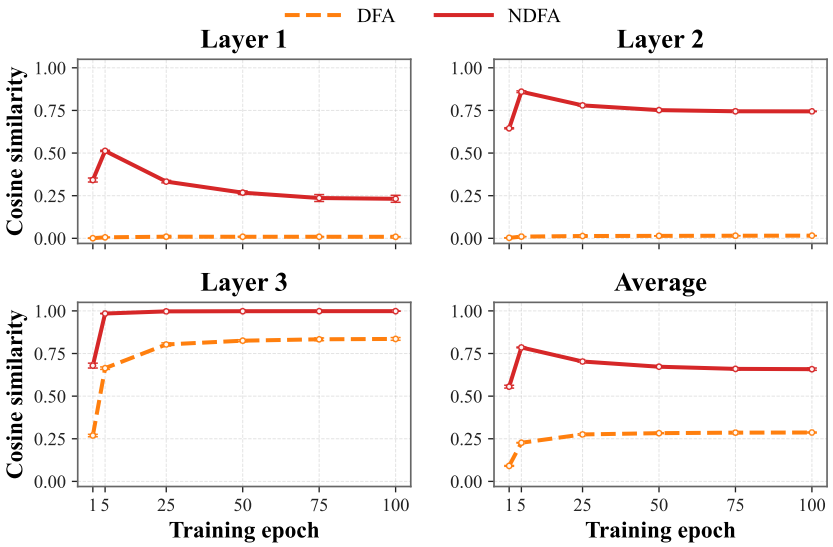


Figure 5: Comparison of layer-wise Jacobian alignment with BP for NDFA and DFA on CIFAR-10 using a ConvNet.

Influence of learning rate. We conduct experiments on CIFAR-10 with ConvNet by varying the learning rate of surrogate subnetworks. As shown in Table 12, NDFA remains stable across different learning rates. Moreover, using a larger learning rate for surrogate subnetworks yields improved accuracy, suggesting that further tuning their learning rates could lead to additional accuracy gains.

Table 12: Results of learning rate (LR) on CIFAR-10 using ConvNet.

LR of Main Net	LR of Surrogate	Accuracy (%)
1×10^{-3}	1×10^{-3}	78.62 ± 0.46
1×10^{-3}	3×10^{-3}	78.74 ± 0.47
1×10^{-3}	5×10^{-3}	$\mathbf{79.34 \pm 0.50}$
1×10^{-3}	5×10^{-4}	78.58 ± 0.42
1×10^{-3}	3×10^{-4}	78.45 ± 0.45

Effect of nonlinearity in the surrogate subnetworks. We conduct additional ablation studies to isolate the effect of nonlinearity in the surrogate subnetworks. Specifically, we compare learnable NDFA with nonlinear surrogate subnetworks against its linear counterpart, while matching the number of parameters between the two variants. We also perform the same comparison for fixed NDFA, again using parameter-matched nonlinear and linear surrogate subnetworks. These experiments are conducted on both CIFAR-10 and CIFAR-100 using ConvNet and ResNet-32.

As shown in Table 13, both the learnable and fixed versions of NDFA with nonlinear surrogate subnetworks consistently outperform their corresponding linear variants. These results demonstrate the effectiveness of the nonlinearity of the surrogate subnetworks in NDFA.

Table 13: Effect of nonlinearity in the surrogate subnetworks.

Network	Method	CIFAR-10	CIFAR-100
ConvNet	Fixed NDFA w/ Linear Surrogate Subnets	73.31 ± 0.26	40.29 ± 0.35
	Fixed NDFA	74.60 ± 0.09	41.65 ± 0.15
	Learnable NDFA w/ Linear Surrogate Subnets	76.23 ± 0.34	43.42 ± 0.58
	NDFA	$\mathbf{78.62 \pm 0.46}$	$\mathbf{47.86 \pm 0.54}$
ResNet-32	Fixed NDFA w/ Linear Surrogate Subnets	73.88 ± 0.32	37.79 ± 0.37
	Fixed NDFA	76.04 ± 0.38	40.10 ± 0.86
	Learnable NDFA w/ Linear Surrogate Subnets	81.03 ± 0.25	46.64 ± 0.47
	NDFA	$\mathbf{85.36 \pm 0.48}$	$\mathbf{55.38 \pm 0.38}$

Table 14: Effects of surrogate subnetwork depth and hidden dimension.

Architecture	Dimension=128			Surrogate Subnetwork Depth=2		
	Depth	CIFAR-10	CIFAR-100	Dimension	CIFAR-10	CIFAR-100
ConvNet	2	78.62±0.46	47.86±0.54	128	78.62±0.46	47.86±0.54
	3	81.30±0.28	49.04±0.44	256	79.48±0.23	48.16±0.47
	4	82.95±0.22	51.11±0.49	512	79.82±0.26	48.25±0.38
ResNet-32	2	85.36±0.48	55.38±0.38	128	85.36±0.48	55.38±0.38
	3	86.76±0.15	58.17±0.29	256	85.69±0.18	55.78±0.35
	4	87.82±0.14	59.35±0.31	512	84.81±0.20	55.94±0.42

D Implementation details for neural data similarity

The Mante 2013 dataset [37]. It is a primate neural recording dataset for context-dependent decision-making. In this task, the model must selectively integrate task-relevant sensory inputs according to contextual cues, capturing the core computational requirement of flexible behavior. The dataset enables evaluation of both task performance and the biological plausibility of learned internal representations. In addition, the task involves temporally extended trials consisting of fixation, target presentation, motion or color context cues, a delay period, and a saccadic response.

CTRNN architecture. For the Mante 2013 dataset, we employ a continuous-time recurrent neural network (CTRNN). The model consists of an input-to-hidden projection, recurrent hidden-state dynamics, and a hidden-to-output readout layer. The hidden states evolve continuously over time through recurrent dynamics with nonlinear activations.

Training protocol. In this recurrent setting, the BP baseline corresponds to backpropagation through time (BPTT), where errors are propagated backward through the unfolded temporal computation graph. NDFA is extended to the CTRNN by applying the same DFA-based principle to recurrent hidden states. Specifically, at each relevant time step, the global output error is directly projected to the recurrent hidden state through a learned nonlinear surrogate pathway. The surrogate subnetwork maps the current hidden state to the output space and generates a feedback signal, which is then used to compute the hidden-state error for updating the recurrent parameters.

Neural similarity evaluation. We follow the evaluation setup in [57]. During training, we evaluate both normalized task accuracy and neural similarity at intermediate training checkpoints. Normalized accuracy measures task performance, where higher values indicate better performance. Neural similarity is quantified using the Procrustes distance between the CTRNN hidden representations and recorded neural activity.

Let $H \in \mathbb{R}^{BT\times N}$ denote the hidden representation of the CTRNN and $\tilde{H} \in \mathbb{R}^{BT\times N'}$ denote the recorded neural responses, where B is the number of trials or experimental conditions, T is the number of time steps, N is the number of model hidden units, and N' is the number of recorded neurons. The Procrustes distance is defined as

$$\theta(H,\tilde{H}) = \min_{Q\in\mathcal{O}} \arccos\left(\frac{\langle H_\phi, \tilde{H}_\phi Q\rangle}{\|H_\phi\|\|\tilde{H}_\phi\|}\right),$$

where $\mathcal{O}$ denotes the set of orthogonal transformations. A lower Procrustes distance indicates that the learned population activity is more closely aligned with the neural recordings after optimal orthogonal alignment. This evaluation enables comparison between BPTT and NDFA at matched task performance and assesses whether NDFA produces recurrent representations that more closely resemble biological neural activity.

Table 15: Per-iteration backward computation time (ms) of NDFA with different surrogate subnetwork depths d on CIFAR-10 using AlexNet and ResNet-32. Values in parentheses indicate the speedup relative to BP.

Method	AlexNet	ResNet-32
BP	2.34 (1.00×)	3.45 (1.00×)
NDFA ($d = 2$)	0.92 (2.54×)	0.78 (4.42×)
NDFA ($d = 3$)	2.23 (1.05×)	2.90 (1.19×)
NDFA ($d = 4$)	3.29 (0.71×)	4.87 (0.71×)

E Influence of surrogate subnetwork architecture

In all experiments of the main paper, the surrogate subnetwork is implemented as a lightweight two-layer MLP with hidden dimension 128. Here, we further investigate how the surrogate subnetwork depth and hidden dimension affect the performance and efficiency of NDFA.

Table 14 summarizes the classification performance with different surrogate architectures on ConvNet and ResNet-32 over CIFAR-10 and CIFAR-100. We first examine the effect of the hidden dimension while fixing the surrogate depth to two layers. Increasing the hidden dimension from 128 to 512 leads to only marginal performance changes across architectures and datasets, suggesting that a relatively small hidden dimension is already sufficient for learning effective feedback. Therefore, we adopt the lightweight setting with hidden dimension 128 throughout the main experiments.

We next analyze the effect of surrogate subnetwork depth while fixing the hidden dimension to 128. Increasing the depth consistently improves classification accuracy on both architectures and datasets, indicating that deeper surrogate subnetworks can better approximate the nonlinear downstream transformations and provide more informative feedback signals. However, as shown in Table 15, increasing the depth also introduces substantially higher backward computation cost and reduces the training acceleration of NDFA. In particular, deeper surrogate subnetworks gradually diminish the parallel training advantage over BP.

These results suggest that lightweight MLP surrogate subnetworks provide a favorable balance between feedback quality and computational efficiency. Consequently, we employ a two-layer MLP with hidden dimension 128 in all experiments as an effective and efficient surrogate subnetwork.

F Surrogate subnetworks with layer-dependent capacities

We conduct experiments to investigate surrogate subnetworks with layer-dependent capacities. Specifically, we use AlexNet with six surrogate subnetworks on CIFAR-10 and evaluate a range of configurations in which the surrogate depth gradually increases or decreases across network layers. For each configuration, we report both classification accuracy and backward-pass time (values in parentheses denote the speedup relative to BP).

As shown in Table 16, in most cases, assigning deeper surrogate subnetworks to earlier layers and shallower ones to later layers substantially improves accuracy, with only a modest reduction in training efficiency. In contrast, assigning shallower surrogate subnetworks to earlier layers and deeper ones to later layers not only increases the computational cost but also leads to lower accuracy. Note that for the highest-capacity setting (6→5→4→3→2→1), our method substantially improves accuracy compared with the uniform two-layer surrogate setting and achieves performance close to that of BP, while achieving a $1.10\times$ backward-pass speedup. The efficiency advantage becomes smaller as the surrogate depth approaches that of the corresponding BP path. The AlexNet architecture used in this experiment consists of eight layers. The first layer has seven downstream layers, whereas the associated surrogate subnetwork in this configuration has a depth of six. Therefore, the computational depth of the highest-capacity surrogate is already close to that of the corresponding BP path, which explains the limited backward-time improvement.

These results suggest that allocating greater surrogate capacity to earlier layers is a more effective design choice for improving performance, despite a slight reduction in training efficiency. Nevertheless, our method with these settings remains much faster than BP due to its parallel training capability. In

future work, we will investigate more principled approaches for designing layer-dependent surrogate subnetworks.

Table 16: Results of surrogate subnetworks with layer-dependent capacities using AlexNet on CIFAR-10. The surrogate depths are listed from the earliest to the latest network layer. Time is per-iteration backward time, and values in parentheses report speedup relative to BP.

Depths (early $\rightarrow$ late)	Acc. (%)	Time (ms)	FLOPs (G)	Params (M)
$2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2$	87.38 ± 0.14	0.92 (2.54×)	140.30	45.82
$3 \rightarrow 3 \rightarrow 2 \rightarrow 2 \rightarrow 1 \rightarrow 1$	88.32 ± 0.34	1.63 (1.44×)	173.08	43.93
$3 \rightarrow 3 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2$	88.88 ± 0.19	1.69 (1.38×)	175.06	44.90
$4 \rightarrow 4 \rightarrow 3 \rightarrow 3 \rightarrow 2 \rightarrow 2$	89.95 ± 0.16	1.90 (1.23×)	242.04	42.36
$6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$	$\mathbf{90.54 \pm 0.15}$	2.13 (1.10×)	277.15	39.65
$2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 3 \rightarrow 3$	87.43 ± 0.20	1.52 (1.54×)	209.56	79.39
$1 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 3 \rightarrow 3$	87.03 ± 0.15	1.34 (1.75×)	202.62	76.00

G Computational overhead

We compare NDFA with BP in terms of per-iteration wall-clock time, FLOPs, and peak GPU memory on CIFAR-10 using ConvNet, AlexNet, and ResNet-32. The reported wall-clock time for NDFA does not account for parallel speedup. As shown in Table 17, NDFA introduces negligible overhead in FLOPs due to its lightweight surrogate subnetworks. Moreover, NDFA requires less GPU memory than BP because it does not need to store the end-to-end computational graph.

We further conduct a scaling analysis by progressively increasing the number of convolutional layers in ConvNet. The results are reported in Table 17, where CXF2 denotes a ConvNet with (X) convolutional layers and two fully connected layers, evaluated on CIFAR-10. As the network depth increases, the additional computational overhead introduced by NDFA remains negligible relative to BP, owing to the lightweight MLP-based surrogate subnetworks.

Table 17: Comparison of computational overhead on CIFAR-10.

Network	BP/NDFA Time (ms)	BP/NDFA FLOPs (G)	BP/NDFA GPU Memory (MB)
C2F2	7.29 / 6.89	5.25 / 6.04	263 / 263
C3F2	10.42 / 9.37	13.33 / 14.26	455 / 456
C5F2	12.31 / 14.05	21.35 / 22.65	454 / 455
C7F2	15.90 / 18.29	22.91 / 24.65	454 / 455
AlexNet	13.64 / 18.35	137.07 / 140.31	1894 / 1509
ResNet-32	37.34 / 46.88	35.39 / 38.80	880 / 831

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We do not claim any contribution out of scope. All the contributions mentioned are supported in the later sections, see Section 1

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of the work in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: Our work primarily focuses on the experimental perspective.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The details to reproduce our experimental results are described in the Appendix A

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: : The code is publicly available at https://github.com/YanFan321/NDFA.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all details in Appendix A

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the mean and standard deviation values by running each result for 3 times with different seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the GPU type in the Appendix A

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We obey the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: No societal impact is concerned by our work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all the papers whose datasets are used in our experiment.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No experiment involves crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No experiment involves crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We only use LLMs for writing.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.