IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 31 March 2024, accepted 25 May 2024, date of publication 5 June 2024, date of current version 17 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3409819

METHODS

Direct Feedback Learning With Local Alignment Support

HEESUNG YANG, SOHA LEE, AND HYEYOUNG PARK
School of Computer Science and Engineering, Kyungpook National University, Daegu 41566, South Korea

Corresponding author: Hyeyoung Park (hypark@knu.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant NRF-2020R1A2C1010020; and in part by the Human Resources Program in Energy Technology of Korea Institute of Energy Technology Evaluation and Planning (KETEP) through the Ministry of Trade, Industry and Energy, Republic of Korea, under Grant 20204010600060.

ABSTRACT While backpropagation (BP) algorithm has been pivotal in enabling the success of modern deep learning technologies, it encounters challenges related to computational inefficiency and biological implausibility. Especially, the sequential propagation of error signals using forward weights in BP is not biologically plausible and prevents efficient parallel updates of learning parameters. To solve these problems, the direct feedback alignment (DFA) method is proposed to directly propagate the error signal from output layer to each hidden layer through random feedback weight, but the performance of DFA is still not competent to BP, especially in complicate tasks with large number of outputs and the convolutional neural network models. In this paper, we propose a method to adjust the feedback weights in DFA using additional local modules that are connected to the hidden layers. The local module attached to each hidden layer has a single-layer structure and learns to mimic the final output of the network. Then, the weights of a local module behave like a direct path connecting each hidden layer to the network output, which has an inverse relationship to the direct feedback weights of DFA. We use this relationship to update the feedback weight of DFA. From the experimental investigation, we confirm that the proposed adaptive feedback weights improve the alignment of the error signal of DFA with that of BP. Furthermore, comparative experiments show that the proposed method significantly outperforms the original DFA on well-known benchmark datasets. The code used for the experiments is available at https://github.com/leibniz21c/direct-feedback-learning-with-local-alignment-support.

INDEX TERMS Backpropagation, biologically plausible learning, random feedback weight, direct feedback alignment, local alignment support module.

I. INTRODUCTION

Deep neural networks have recently achieved great success in various application fields including computer vision and natural language processing. The key to their success is a learning method called backpropagation (BP) [1], which was originally developed for training multilayer perceptrons [2], and has evolved along with various learning techniques such as dropout and stochastic optimization [3], [4], [5], [6]. Currently, it plays a crucial role in the learning of highly advanced deep networks. However, several problems have been identified with BP, such as computational inefficiency due to sequential feedback calculation and biological implausibility, and studies to address these issues are still ongoing [7].

A well-known issue regarding the biological implausibility of BP is that forward weight values are required during the backward propagation of error signals, commonly referred to as the *weight transport* problem [8]. In the BP algorithm, the weight update term of each layer is determined by the

The associate editor coordinating the review of this manuscript and approving it for publication was Ganesh Naik.

gradient vector of the output loss function. Since this gradient is calculated sequentially from the output layer to the input layer, the computational flow uses the same forward weights in the backward transmission. This perfectly symmetric structure of the forward and backward weight matrices cannot be found in biological neural networks, and this problem is also called the *weight symmetry* problem [9].

To address this problem, Lillicrap et al. [10] proposed a learning method called feedback alignment (FA), which uses fixed random weights for the feedback transmission of error signal. They experimentally showed that learning can be successful even if the forward weight matrix in the feedback propagation is replaced by a fixed random matrix, breaking the weight symmetry. They also provided some theoretical explanations for the success, insisting that the forward weights are aligned with the fixed feedback weights as they are updated, leading to the alignment of the error signal. This primary study led to the development of several variants of learning methods with random feedback weights and theoretical analyses of their properties [10], [11], [12], [13], [14].

In particular, the direct feedback alignment (DFA) method [12] is noteworthy in that it changes the error feedback path itself and also solves the *backward locking* problem [15]. In the sequential feedback learning such as BP and FA, the output error is backpropagated layer by layer, causing delays in updating the weights of each layer until the error signal is transmitted from the upper layer. This backward locking issue prevents parallel learning across layers, which is biologically implausible and computationally inefficient. In contrast, DFA can solve this problem, as well as the weight transport problem, by using direct feedback paths from the output to each hidden layer with randomly fixed weights.

The biological plausibility and practical advantages of DFA have led to a variety of subsequent studies, including performance improvements [16], [17], theoretical analyses [18], [19], and hardware implementations [16], [20], [21]. However, most of these studies have not been able to reach the performance of BP, and further improvement is needed for the widespread use of DFA in practice. To this end, this paper considers a method of updating the feedback weight of DFA.

In the case of FA, a method was proposed to update the backward weight of each layer to mimic the corresponding transposed forward weight matrix, enhancing their alignment and approaching the performance of BP [11]. In the case of DFA, however, it is difficult to obtain a similar updating rule for the backward weights because the layer-wise weight correspondence does not exist due to the structural difference between forward and feedback path.

To address this problem, we propose a new learning method with additional local modules that support the adjustment of backward weights in DFA. The additional single-layer local module is connected to each hidden layer and trained to produce outputs close to the network output. Thus, the weight of the local module behaves like a direct forward channel connecting each hidden layer to the final output, which provides an inverse correspondence to the direct feedback path of DFA. Based on this correspondence, we use the transposed local weights to update the direct feedback weights. The proposed update of the feedback weights allows the propagated error signal for forward weight learning to be well aligned with that of BP, as will be shown experimentally in Section III. We also experimentally confirm that the proposed method improves the learning performance compared to DFA in several benchmark tasks, especially for the convolutional neural network (CNN) models. The main contributions of this work are as follows:

- We propose a new learning method that updates direct feedback weights through the support of additional local modules.
- The proposed method is implemented in two different algorithms: the local alignment support (LAS) learning and the local target alignment support (LTAS) learning, which will be described in Section III.
- We verify the performance of the proposed method through experimental comparison with BP and DFA on several benchmark datasets and network models including CNN.

II. LEARNING WITH FEEDBACK ERROR SIGNAL

A. NETWORK FORMULATION

Learning methods using feedback error signal can be classified into two types, sequential feedback learning and direct feedback learning, depending on the feedback path. Figure 1 shows these two types of feedback learning. Before describing each of them in detail, let us first formulate the computation performed by the neural network. We consider a fully connected network with L layers with a weight matrix $\boldsymbol{W}_i$ and bias $\mathbf{b}_i$ in the i-th hidden layer. The output of the i-th layer $\mathbf{h}_i$ is computed using the output of the previous layer $\mathbf{h}_{i-1}$, the connection weights $\boldsymbol{W}_i$ and bias $\mathbf{b}_i$, and an activation function f_i, which can be written as

$$\mathbf{h}_i = f_i(\mathbf{a}_i), \tag{1}$$

$$\mathbf{a}_i = \boldsymbol{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i, \tag{2}$$

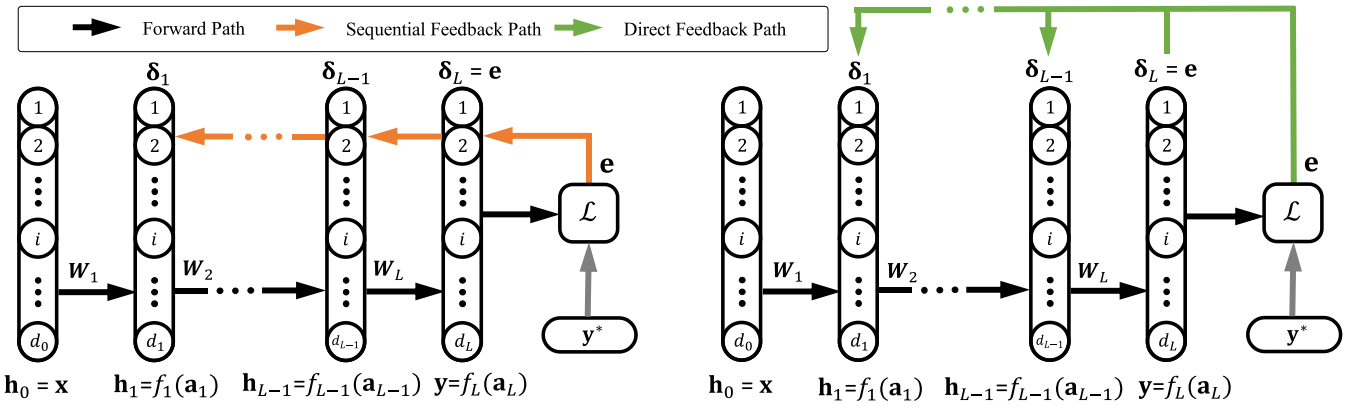
where $i = 1, \cdots, L$. For a given input sample $\mathbf{x} = \mathbf{h}_0$, the network output $\mathbf{y} = \mathbf{h}_L = f_L(\mathbf{a}_L)$ is obtained by sequential forward calculation in a layer-by-layer manner.

To train the network, we have a loss function $\mathcal{L}(\mathbf{y}, \mathbf{y}^*)$, where $\mathbf{y}^*$ is the target output, and the weights and biases of each layer are repeatedly updated to decrease the loss as follows:

$$\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i + \Delta \boldsymbol{W}_i, \tag{3}$$

$$\mathbf{b}_i \leftarrow \mathbf{b}_i + \Delta \mathbf{b}_i. \tag{4}$$

The update terms $\Delta \boldsymbol{W}_i$ and $\Delta \mathbf{b}_i$ for each layer are determined by rules that assign the responsibility for the current loss value to each layer. In error feedback learning, we first obtain the error vector $\mathbf{e}$ by taking the gradient of the loss function with



(a) Sequential feedback learning (b) Direct feedback learning

FIGURE 1. Two different feedback paths for learning with backward error signals. (a) In the sequential feedback learning, the error signal from the last layer is propagated sequentially, layer by layer, by the sequential feedback path (orange line). (b) In the direct feedback learning, he error signal from the last layer is transmitted directly to each layer via the direct feedback path (green line).

respect to $\mathbf{a}_L$, such as,

$$\mathbf{e} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_L}, \tag{5}$$

and use this to calculate the error signal $\boldsymbol{\delta}_i$ $(i = 1, \cdots, L)$ to assign the responsibility to each layer.

In the case of the output layer, the error vector $\mathbf{e}$ itself can be the error signal $\boldsymbol{\delta}_L$ to determine the update terms $\Delta \boldsymbol{W}_L$ and $\Delta \mathbf{b}_L$. Note that the specific form of $\mathbf{e}$ is determined by the loss function $\mathcal{L}$ and the activation function f_L. If we use the squared error loss and linear (identity) activation, the error vector is given as $\mathbf{e} = \mathbf{y} - \mathbf{y}^*$. We also have the same form of error in the case of the cross-entropy loss and the softmax activation function. On the other hand, for the i-th hidden layer, the error signal $\boldsymbol{\delta}_i$ needs to be computed by propagating the output error vectors $\mathbf{e}$ through a feedback path. There are many variants on how to define the specific formula of $\boldsymbol{\delta}_i$, and we try to categorize them according to the type of feedback path, sequential vs. direct.

B. SEQUENTIAL FEEDBACK LEARNING ALGORITHMS

The computational flow of the sequential feedback learning is shown in Figure 1 (a). The representative method of the sequential feedback learning is BP [1], in which the error signal $\boldsymbol{\delta}_i^{\mathrm{BP}}$ of the i-th layer is defined as the gradient of the loss function $\mathcal{L}$ with respect to $\mathbf{a}_i$, such as

$$\boldsymbol{\delta}_L^{\mathrm{BP}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_L} = \mathbf{e}, \tag{6}$$

$$\boldsymbol{\delta}_i^{\mathrm{BP}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_i} = \left(\boldsymbol{W}_{i+1}^T \boldsymbol{\delta}_{i+1}^{\mathrm{BP}} \right) \odot f_i' \left(\mathbf{a}_i \right), \tag{7}$$

where $i = 1, \cdots, L-1$ and $\odot$ denotes the element-wise multiplication. Using the error signal, the update terms of $\Delta \boldsymbol{W}_i^{\mathrm{BP}}$ and $\Delta \mathbf{b}_i^{\mathrm{BP}}$ are obtained as

$$\Delta \boldsymbol{W}_i^{\mathrm{BP}} = -\eta \boldsymbol{\delta}_i^{\mathrm{BP}} \mathbf{h}_{i-1}^T, \tag{8}$$

$$\Delta \mathbf{b}_i^{\mathrm{BP}} = -\eta \boldsymbol{\delta}_i^{\mathrm{BP}}. \tag{9}$$

Note that we need to use the forward weights $\boldsymbol{W}_{i+1}^T$ to compute the update terms $\Delta \boldsymbol{W}_i^{\mathrm{BP}}$ of the i-th hidden layers. Since the synaptic weights used in backward propagation are exactly the same as those used in forward calculation, BP has the weight transport problem. There is also the backward locking problem, which prevents efficient parallel updates, because we have to compute $\boldsymbol{\delta}_{i+1}^{\mathrm{BP}}$ first to get $\boldsymbol{\delta}_i^{\mathrm{BP}}$. Despite these biological implausibility and computational inefficiency, BP is the most widely used learning algorithm due to its high performance.

On the other hand, the feedback alignment (FA) [10] learning method replaces $\boldsymbol{W}_{i+1}^T$ in the calculation of $\boldsymbol{\delta}_i^{\mathrm{BP}}$ with a fixed random matrix $\mathbf{B}_{i+1}$, and obtains the error signal for each layer such as

$$\boldsymbol{\delta}_L^{\mathrm{FA}} = \boldsymbol{\delta}_L^{\mathrm{BP}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_L} = \mathbf{e}, \tag{10}$$

$$\boldsymbol{\delta}_i^{\mathrm{FA}} = \left(\boldsymbol{B}_{i+1} \boldsymbol{\delta}_{i+1}^{\mathrm{FA}} \right) \odot f_i' \left(\mathbf{a}_i \right). \tag{11}$$

The update terms $\Delta \boldsymbol{W}_i^{\mathrm{FA}}$ and $\Delta \mathbf{b}_i^{\mathrm{FA}}$ can then be given as

$$\Delta \boldsymbol{W}_i^{\mathrm{FA}} = -\eta \boldsymbol{\delta}_i^{\mathrm{FA}} \mathbf{h}_{i-1}^T, \tag{12}$$

$$\Delta \mathbf{b}_i^{\mathrm{FA}} = -\eta \boldsymbol{\delta}_i^{\mathrm{FA}}. \tag{13}$$

As shown in the equations, the backward locking problem still exists because the computation of $\boldsymbol{\delta}_i^{\mathrm{FA}}$ is not possible until the computation of $\boldsymbol{\delta}_{i+1}^{\mathrm{FA}}$ is complete. However, the weight transport problem can be solved by using $\boldsymbol{B}_{i+1}$ instead of the forward weights $\boldsymbol{W}_{i+1}^T$ in the computation of $\boldsymbol{\delta}_i^{\mathrm{FA}}$. Through computational experiments using FA on benchmark datasets, it was confirmed that the exact forward weights are not required to train a multilayer neural network, and that it can be trained even with fixed random weights [10]. In addition, it was observed that each forward weight align with its corresponding feedback weight as learning progress [10], which is an important condition for successful training with fixed random backward weights.

Furthermore, Liao et al. [9] investigated which factors are important in the symmetry of forward and backward weights, and experimentally showed that the magnitude of the weight does not matter for the performance and the sign concordance between forward and backward weights is important. They also proposed a variant of FA called fixed random magnitude sign concordance (frSF) that uses sign concordant feedback weights with random magnitude. This method can be written as

$$\boldsymbol{\delta}_i^{\mathrm{frSF}} = \left(\left(\boldsymbol{M}_{i+1} \odot sign\left(\boldsymbol{W}_{i+1}^T\right)\right)\boldsymbol{\delta}_{i+1}^{\mathrm{frSF}}\right) \odot f_i'\left(\mathbf{a}_i\right), \quad (14)$$

where $\boldsymbol{M}_{i+1}$ is a random fixed matrix with positive elements. Although these variants show some improvement over FA, the performance gap with BP has not yet been overcome.

In order to alleviate the performance gap, Akrout et al. [11] proposed two methods using adjustable feedback weights: the weight mirror algorithm and a modified version of Kolen and Pollak's backward weight updates [14]. The two methods begin with random feedback weights $\boldsymbol{B}_i$ and update the values to align well with the transpose of forward weight matrix $\boldsymbol{W}_i^T$. By updating the feedback weights, the modified Kolen-Pollak (KP) algorithm and the weight mirror (WM) algorithm outperformed FA and came close to BP, especially for the complicated tasks. They are free from the weight transport problem and their possible implementation in real neural networks have also been suggested in [14]. However, the sequential feedback learning methods still suffer from the backward locking problem, which is crucial for efficient learning of deep network models with a large number of layers.

C. DIRECT FEEDBACK LEARNING ALGORITHMS

Unlike the sequential feedback learning, the direct feedback learning methods propagate the error signals through direct paths from output to every layer in parallel, as shown in Figure 1 (b). In the direct feedback alignment (DFA) method [12], the error vector of the output layer is propagated directly to all hidden layers through random feedback weights. The error signal $\boldsymbol{\delta}_i^{\mathrm{DFA}}$ for ith hidden layer and the update terms of weight $\boldsymbol{W}_i$ and bias $\mathbf{b}_i$ are given by

$$\boldsymbol{\delta}_L^{\mathrm{DFA}} = \boldsymbol{\delta}_L^{\mathrm{BP}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_L} = \mathbf{e}, \quad (15)$$

$$\boldsymbol{\delta}_i^{\mathrm{DFA}} = \left(\boldsymbol{B}_{i+1}\mathbf{e}\right) \odot f_i'\left(\mathbf{a}_i\right), \quad (16)$$

$$\Delta \boldsymbol{W}_i^{\mathrm{DFA}} = -\eta \boldsymbol{\delta}_i^{\mathrm{DFA}} \mathbf{h}_{i-1}^T, \quad (17)$$

$$\Delta \mathbf{b}_i^{\mathrm{DFA}} = -\eta \boldsymbol{\delta}_i^{\mathrm{DFA}}. \quad (18)$$

Note that the error signal $\boldsymbol{\delta}_i^{\mathrm{DFA}}$ for i-th hidden layer is computed by the direct transition of the error vector $\mathbf{e}$ of output layer, which does not require the error signal $\boldsymbol{\delta}_{i+1}^{\mathrm{DFA}}$ of the upper layer. Therefore, it is free from the backward locking problem and can be efficiently trained in parallel. In addition, it can solve the weight transport problem as well, because the forward weight is not necessary in the error feedback path.

Focusing on the advantages of parallel updates in DFA learning, several attempts have been made to increase the practical usability of DFA by improving its performance and implementation efficiency [16], [17], [19], [20]. The sparse direct feedback alignment (SDFA) [16] and the binary direct feedback alignment (BDFA) [20] reduce the memory of feedback weights used in conventional DFA, and Launay et al. [17] proposed a method to share a feedback weight matrix for multiple layers for memory efficiency in the implementation of DFA. Furthermore, the direct random target projection (DRTP) [19] showed that a random projection of target vector (one-hot encoded label) can be used a proxy of the error signal of DFA.

In spite of the active works on the modification of DFA, their performances still lag behind BP, especially in the advanced deep learning models such as CNN. To overcome the performance degradation, we need to understand the nature of DFA training. As an attempt to do so, Refinetti et al. [18] showed that the forward weights align with the fixed feedback weights in the early phase of DFA learning, and then the alignment is sacrificed in the later phase to minimize the loss. Consequently, the fixed feedback weights of DFA limit the search space for optimal forward weights, leading to the performance gap with BP. In addition, they also showed that the alignment cannot be achieved in CNN training due to the structural difference between the fixed random feedback weights.

Based on the observations, it would be possible to consider updating the backward weight like KP and WM in the FA learning. However, it is difficult to find where $\boldsymbol{B}_i$ should be aligned due to the structural difference between forward and backward paths. As an alternative, Baldi et al. [22] updated the feedback weight of the i-th layer using the product of the forward weights after the i-th layer to achieve good alignment, but its performance was lower than DFA. In this paper, we try to find an appropriate update direction of feedback weight by adopting additional local modules.

III. PROPOSED METHODS

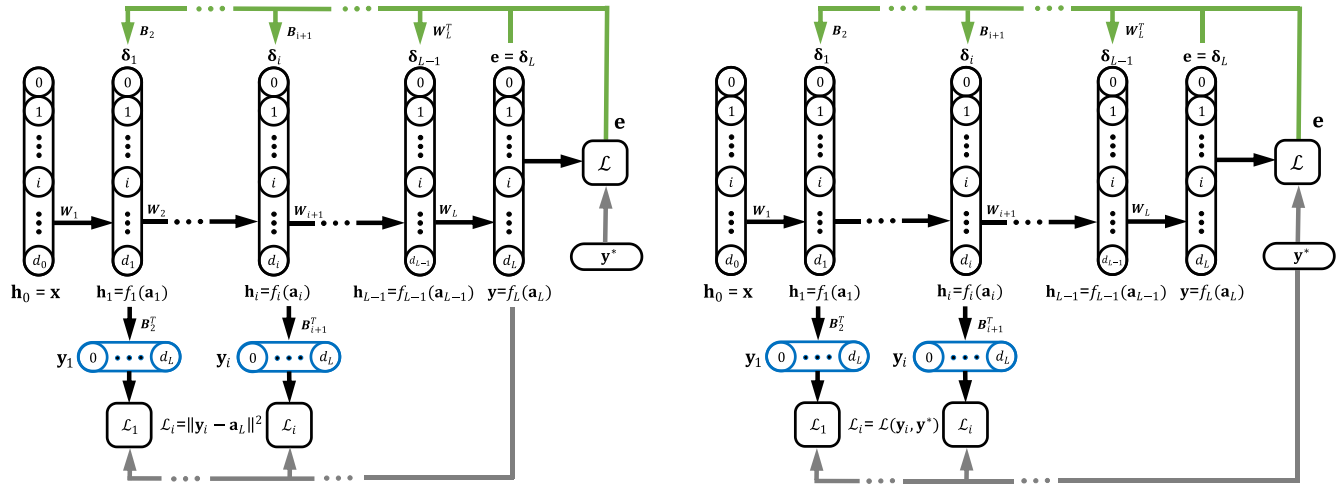
A. LEARNING WITH LOCAL ALIGNMENT SUPPORT

In order to get appropriate alignment directions for updating feedback weights in DFA, we proposed to use additional local modules, which are called local alignment support (LAS) modules. The LAS module is a single layer network with linear output, which is attached to each hidden layer to make an output directly from the hidden layer, as shown in Figure 2 (a). The local output from the LAS module attached to i-th hidden layer is then written by

$$\mathbf{y}_i = \boldsymbol{B}_{i+1}^T \mathbf{h}_i. \quad (19)$$

The LAS module is trained to mimic the output of the network, $\mathbf{a}_L$, using the loss function defined as

$$\mathcal{L}_i = \left\|\mathbf{y}_i - \mathbf{a}_L\right\|^2. \quad (20)$$



(a) Direct feedback learning with LAS module

(b) Direct feedback learning with LTAS module

FIGURE 2. Direct feedback learning with local modules. (a) Each local alignment support (LAS) module is trained to approximate the linear network output a_L, and the trained weight B_i, is used for updating weights in the direct feedback paths. (b) Each local target alignment support (LTAS) module is trained to predict y^*, and the trained weight B_i, is used for updating the direct feedback weights.

Then the update term of the weight is given by

$$\Delta \boldsymbol{B}_{i+1} = -\eta \mathbf{h}_i \left(\mathbf{a}_L - \mathbf{y}_i\right)^T . \tag{21}$$

Through the updates, the weight of the LAS module acts as the direct forward connection from each hidden layer to the output layer, which corresponds the direct forward weights of LAS module and the direct feedback connection of DFA. Based on the correspondence, we assign the transpose of $\boldsymbol{B}_i$ in LAS module to the fixed random weight of DFA. Thus, the global error vector $\mathbf{e}$ is propagated directly through $\boldsymbol{B}_{i+1}$ to each layer as in conventional DFA, and the error signal is given as

$$\boldsymbol{\delta}_L^{\mathrm{LAS}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_L} = \mathbf{e} = \mathbf{y}^* - \mathbf{y}, \tag{22}$$

$$\boldsymbol{\delta}_{L-1}^{\mathrm{LAS}} = \boldsymbol{\delta}_{L-1}^{\mathrm{BP}} = \left(\boldsymbol{W}_L^T \mathbf{e}\right) \odot f'_{L-1}\left(\mathbf{a}_{L-1}\right), \tag{23}$$

$$\boldsymbol{\delta}_i^{\mathrm{LAS}} = \left(\boldsymbol{B}_{i+1}\mathbf{e}\right) \odot f'_i\left(\mathbf{a}_i\right). \tag{24}$$

Figure 2 (a) shows the architecture of a fully connected network with the direct feedback connection and the proposed LAS modules. Note that the LAS module is not attached to the last hidden layer because the output layer can play the same role. The proposed approach is inspired from the WM method [11] and KP method [14], which update the backward weights in sequential feedback learning to align them with the transpose of forward weight matrix. The LAS module is additionally introduced to make the similar alignment possible in DFA.

Algorithm 1 shows how to update the whole network with the LAS modules. Though it is important that $\mathbf{y}_i$ and $\mathbf{a}_L$ are close enough for training with LAS, we have found empirically that the iterative training of LAS modules and main networks works well enough. It is noteworthy that the proposed method can solve the backward locking problem because the weights $\boldsymbol{B}_{i+1}$ of each layer can be updated in parallel.

Algorithm 1 Learning with LAS module

Input: Weight parameter ($\boldsymbol{W}_i$, $\mathbf{b}_i$, $\boldsymbol{B}_i$) of L-layer network $\boldsymbol{F}$
Input data $\mathbf{x}$, target data $\mathbf{y}^*$, and learning rate η
Output: Updated weight parameter $\boldsymbol{W}_i$, $\mathbf{b}_i$, and $\boldsymbol{B}_i$
0: For input $\mathbf{x} = \mathbf{h}_0$,
 # Forward calculation
1: **for** $i = 1$ **to** $L - 1$
2: $\mathbf{a}_i = \boldsymbol{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i$
3: $\mathbf{h}_i = f_i\left(\mathbf{a}_i\right)$ *# hidden layer output*
4: $\mathbf{y}_i = \boldsymbol{B}_{i+1}^T \mathbf{h}_i$ *# local module output*
5: $\mathbf{a}_L = \boldsymbol{W}_L \mathbf{h}_{L-1} + \mathbf{b}_L$
6: $\mathbf{y} = f_L\left(\mathbf{a}_L\right)$ *# network output*
7: $\mathcal{L} = \mathrm{Loss}\left(\mathbf{y}, \mathbf{y}^*\right)$ *# Loss value*
8: $\boldsymbol{\delta}_L = \mathbf{e} = \mathbf{y}^* - \mathbf{y}$ *# error signal*
 # Parallel update of weight parameters $\boldsymbol{B}_i$, $\mathbf{W}_i$, $\mathbf{b}_i$.
9: **do in parallel for** $i = 1$ **to** $L - 1$
10: **if** $i = L - 1$ **then**
11: $\boldsymbol{\delta}_i = \left(\boldsymbol{W}_{i+1}^T \mathbf{e}\right) \odot f'_i\left(\mathbf{a}_i\right)$
12: **else**
13: $\boldsymbol{B}_{i+1} \leftarrow \boldsymbol{B}_{i+1} - \eta \mathbf{h}_i \left(\mathbf{a}_L - \mathbf{y}_i\right)^T$
14: $\boldsymbol{\delta}_i = \left(\boldsymbol{B}_{i+1}\mathbf{e}\right) \odot f'_i\left(\mathbf{a}_i\right)$
15: $\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i - \eta \boldsymbol{\delta}_i \mathbf{h}_{i-1}^T$
16: $\mathbf{b}_i \leftarrow \mathbf{b}_i - \eta \boldsymbol{\delta}_i$

B. LEARNING WITH LOCAL TARGET ALIGNMENT SUPPORT

As a variant of the proposed learning with LAS module, we introduce the local target alignment support (LTAS) module, which is trained by using the target output values $\mathbf{y}^*$ instead of the network output $\mathbf{a}_L$. The proposed LTAS module has the same single-layer structure as LAS, except for the activation function of the output nodes. In the case of regression problem where the target has continuous real

values, the activation function is identity, which is the same as the LAS module. However, in the case of classification problem, the target output vector $\mathbf{y}^*$ is given as a one-hot vector and the output of LTAS module $\mathbf{y}_i$ needs to be obtained by applying the softmax activation function to the linear output $\boldsymbol{B}_{i+1}^T \mathbf{h}_i$.

Since the proposed LTAS module is trained to fit the target output, instead to mimic the network output, the loss function should also be defined for the objectives. For the regression problem, the loss function is defined by using the squared error, such as

$$\mathcal{L}_i = \left\| \mathbf{y}_i - \mathbf{y}^* \right\|^2 . \tag{25}$$

For the classification problem, the cross-entropy loss function is used, which can be written as

$$\mathcal{L}_i = \mathcal{L}_{\text{cross-entropy}} \left(\mathbf{y}_i, \mathbf{y}^* \right) . \tag{26}$$

For both of the two different loss functions, the update term for LTAS module is commonly given as

$$\Delta \boldsymbol{B}_{i+1} = -\eta \mathbf{h}_i \left(\mathbf{y}^* - \mathbf{y}_i \right)^T . \tag{27}$$

The proposed LAS and LTAS modules commonly make the direct forward connection from each hidden layer to final output. The difference is in the learning objective. By training the LAS module to mimic the network output, we eventually expect them to act as direct connections between the network output and the i-th hidden layer. On the other hand, by training the LTAS module to get closer to the desired target output, we want to get information about the error credit of each hidden layer through the weight of the LTAS module.

Figure 2 (b) shows the structure and information flow of the learning with LTAS module, and Algorithm 2 shows how to train the whole network with the LTAS modules. Compared to Algorithm 1, one can see that the LTAS modules can be trained in the middle of the forward path. This is possible due to that LTAS learning does not need the network output and only uses target values that can be directly obtained from training data. Once the LTAS module is updated, the update of the whole network can be done in parallel, in the same way of DFA and LAS.

Algorithm 2 Learning with LTAS module

Input: Weight parameter ($\boldsymbol{W}_i$, $\mathbf{b}_i$, $\mathbf{B}_i$) of L-layer network $\mathbf{F}$
Input data $\mathbf{x}$, target data $\mathbf{y}^*$, and learning rate η
Output: Updated weight parameter $\boldsymbol{W}_i$, $\mathbf{b}_i$, and $\mathbf{B}_i$
0: For input $\mathbf{x} = \mathbf{h}_0$,
 # Forward calculation and update of $\mathbf{B}_i$
1: **for** $i = 1$ **to** $L - 1$ do
2: $\quad \mathbf{a}_i = \boldsymbol{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i$
3: $\quad \mathbf{h}_i = f_i(\mathbf{a}_i)$ *# hidden layer output*
4: $\quad \mathbf{y}_i = \mathbf{B}_{i+1}^T \mathbf{h}_i$ *# local module output*
 # Update LTAS module
5: $\quad \mathbf{B}_{i+1} \leftarrow \mathbf{B}_{i+1} - \eta \mathbf{h}_i \left(\mathbf{y}^* - \mathbf{y}_i \right)^T$
6: $\mathbf{a}_L = \boldsymbol{W}_L \mathbf{h}_{L-1} + \mathbf{b}_L$
7: $\mathbf{y} = f_L(\mathbf{a}_L)$ *# network output*
8: $\mathcal{L} = \text{Loss}(\mathbf{y}, \mathbf{y}^*)$ *# Loss value*
9: $\boldsymbol{\delta}_L = \mathbf{e} = \mathbf{y}^* - \mathbf{y}$
 # Parallel update weight of parameters $\boldsymbol{W}_i$ and $\mathbf{b}_i$
10: **do in parallel for** $i = 1$ **to** L
11: $\quad$ **if** $i = L - 1$ **then**
12: $\quad\quad \boldsymbol{\delta}_i = \left(\boldsymbol{W}_{i+1}^T \mathbf{e} \right) \odot f_i'(\mathbf{a}_i)$
13: $\quad$ **else**
14: $\quad\quad \boldsymbol{\delta}_i = \left(\mathbf{B}_{i+1} \mathbf{e} \right) \odot f_i'(\mathbf{a}_i)$
15: $\quad \boldsymbol{W}_i \leftarrow \boldsymbol{W}_i - \eta \boldsymbol{\delta}_i \mathbf{h}_{i-1}^T$
16: $\quad \mathbf{b}_i \leftarrow \mathbf{b}_i - \eta \boldsymbol{\delta}_i$

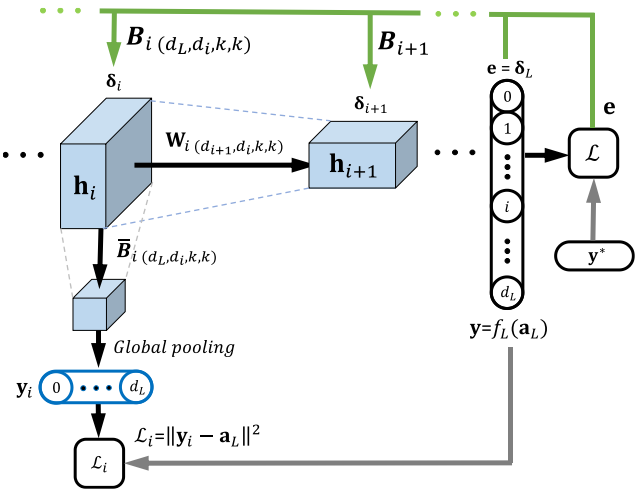


FIGURE 3. Convolution block with LAS module. The LAS module attached to a convolution block has single layer convolution with d_L planes and global pooling to make d_L dimensional local output.

C. EXTENSION TO CONVOLUTIONAL NEURAL NETWORKS

The proposed method can be extended to train CNN models. It is known that DFA does not work well in CNN training due to the local connection and weight sharing structure of convolutional layers. To overcome these limitations, we try to find an appropriate direct feedback signal by utilizing the LAS module. However, we should note that the simple fully connected layer attached to each convolutional layer is practically intractable, because we need much more number of parameters proportional to the number of neurons in all planes of the convolutional blocks.

To address this inefficiency, we design a new local module with a convolutional layer and global pooling. For the i-th convolutional block with d_i planes of kernel size (k, k), the attached LAS module has a convolutional layer with the same kernel size and d_L planes. Then, by applying the global pooling to each plane, the module computes a d_L dimensional output vector $\mathbf{y}_i$. Figure 3 shows a convolutional block with the LAS module. Obviously, the LTAS module has the same structure.

The training of parameter $\bar{\boldsymbol{B}}_i$ in i-th LAS and LTAS modules can be done in the same manner described in Section III.A and 3.B, respectively. Once the local modules are trained, the weights are used for adapting the direct feedback weights: The error signal $\boldsymbol{\delta}_i^{\text{LAS}}$ for i-th convolutional block is obtained by using the global error vector $\mathbf{e}$ and $\bar{\boldsymbol{B}}_i$, where $\bar{\boldsymbol{B}}_i$ is the weight tensor obtained by applying flipped kernel operation to $\boldsymbol{B}_i$.

D. ALIGNMENT PROPERTIES

In this section, we discuss the alignment property, which is **the motivation of the proposed method and the core** characteristic of FA and DFA learning. When the FA method was proposed by Lillicrap [13], it was suggested that FA works because the forward weight $\boldsymbol{W}^T$ tends to align with the fixed feedback weight $\boldsymbol{B}$ during learning. This weight alignment leads to the gradient alignment, which means that the error signal $\boldsymbol{\delta}_i^{\mathrm{FA}}$ aligns with that of the BP, $\boldsymbol{\delta}_i^{\mathrm{BP}}$. Based on these alignment properties, Akrout et al. [11] proposed to update the backward weights $\boldsymbol{B}_i$ to be aligned with the corresponding forward weight $\boldsymbol{W}_i^T$ in their proposed WM method and the modified KP method, and improved the learning performance.

In the case of DFA, the exact alignment between forward weight $\boldsymbol{W}_i^T$ and the corresponding direct feedback weight $\boldsymbol{B}_i$ is impossible due to their structural discrepancy. However, Refinetti et al. [18] experimentally showed that the weight and gradient alignment still occur in DFA learning. Also, in the experiment of [16], it is shown that there is a strong correlation between the prediction accuracy and the amount of alignment between $\boldsymbol{B}_i$ and $\boldsymbol{W}_i^T \cdots \boldsymbol{W}_L^T$. Based on these observations, we try to find an appropriate backward matrix $\boldsymbol{B}_i$ through learning of the proposed local module.

Using the LAS module with a single layer, we try to approximate of the output of the whole network $\mathbf{a}_L$. This can be exactly realized when the activation functions of hidden layers are linear. For example, for a fully connected L-layer network with linear activations, the final output $\mathbf{a}_L$ can be written by a linear transformation, such as

$$\mathbf{a}_L = \boldsymbol{W}_L \boldsymbol{W}_{L-1} \cdots \boldsymbol{W}_{i+1} \mathbf{h}_i. \tag{28}$$

Therefore, the LAS module of i-th hidden layer can exactly predict $\mathbf{a}_L$ when we have $\boldsymbol{B}_i = \boldsymbol{W}_i^T \cdots \boldsymbol{W}_L^T$, which implies the exact alignment discussed in [16] and [22]. In addition, the error signal of BP in the linear network is given as

$$\boldsymbol{\delta}_i^{\mathrm{BP}} = \boldsymbol{W}_{i+1}^T \boldsymbol{W}_{i+2}^T \cdots \boldsymbol{W}_L^T \mathbf{e}, \tag{29}$$

and the error signal of DFA also becomes equal to $\boldsymbol{\delta}_i^{\mathrm{BP}}$ when $\boldsymbol{B}_i = \boldsymbol{W}_i^T \cdots \boldsymbol{W}_L^T$. In a linear network, this exact alignment can be achieved by learning of LAS module because the weight update is given as

$$\Delta \boldsymbol{B}_i = -\eta \mathbf{h}_{i-1} \mathbf{h}_{i-1}^T \left(\boldsymbol{W}_i^T \cdots \boldsymbol{W}_L^T - \boldsymbol{B}_i \right), \tag{30}$$

which has an equilibrium at $\boldsymbol{B}_i = \boldsymbol{W}_i^T \cdots \boldsymbol{W}_L^T$.

However, this exact alignment condition is only applicable to the linear networks. In the case of nonlinear networks, the adaptive feedback weight method [22], which uses this alignment condition, does not work well and even shows lower performance than the original DFA with fixed random weights. We speculate that the performance gap between DFA and BP comes from the lack of alignment by the use of the condition that does not account for the nonlinearity of the network. Therefore, instead of using the specific alignment condition, we try to find appropriate weights that give a good alignment of the error signal by training the local module.

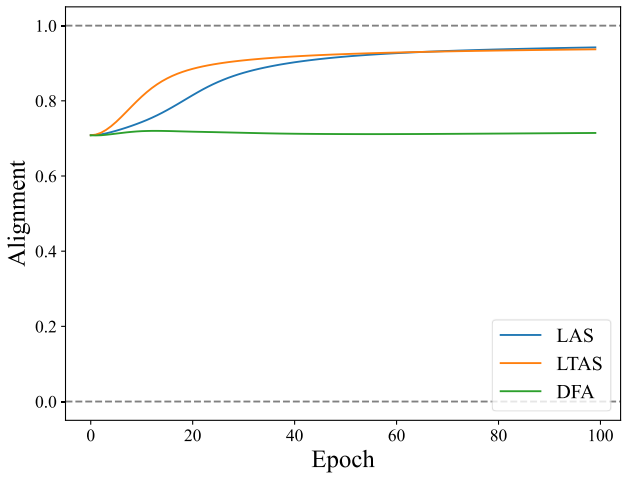


FIGURE 4. Change of alignment during training MNIST dataset. The alignment is measure by cosine similarity between δ_1^{BP} and δ_1^{LAS} (–), δ_1^{LTAS} (–), δ_1^{DFA} (–), respectively.

To empirically verify the effect of the local module, we conducted some experiments using MNIST [25] dataset. We trained a fully connected network with two hidden layers. Each hidden layer has 1024 neurons with ReLU nonlinearity. The network is trained using four learning methods - BP, DFA, LAS, LTAS - to measure the alignment of the error signal between BP and the others (DFA, LAS, LTAS). For DFA learning, we set the feedback weight as $\boldsymbol{B}_i = \boldsymbol{W}_L \cdots \boldsymbol{W}_i$ using the initial values of the forward weights, and the same values are used to initialize the feedback weight of the LAS and LTAS modules.

We perform learning for 100 epochs using stochastic gradient descent (SGD) with a learning rate of 0.0001 and a batch size of 64. At each iteration, we measure the alignment using the cosine similarity of $\boldsymbol{\delta}_1^{\mathrm{BP}}$ and the error signal of the other methods ($\boldsymbol{\delta}_1^{\mathrm{DFA}}$, $\boldsymbol{\delta}_1^{\mathrm{LAS}}$, and $\boldsymbol{\delta}_1^{\mathrm{LTAS}}$). Note that an alignment value of 1 means that the learning direction is equal to BP.

Figure 4 shows the change in the average alignment for the minibatches in each epoch. The results show that the error signals of LAS and LTAS become closer to that of BP as learning progresses, while there is no significant increase in the alignment for DFA. From the result, we argue that the proposed local module can provide meaningful information to approximate the error signal of BP.

IV. EXPERIMENTAL RESULTS

A. RESULTS ON FULLY CONNECTED NETWORKS

We experimentally demonstrate the performances of LAS and LTAS by using the benchmark datasets MNIST, CIFAR-10, and CIFAR-100 [27]. We first use a fully connected network with two 1024-dimensional ReLU nonlinearity hidden layers for the MNIST dataset, and three 4096-dimensional ReLU nonlinearity hidden layers for CIFAR-10 and CIFAR-100.

TABLE 1. Test accuracy on convolutional network models. For each data setting, the best result is written in bold, and the second runner is denoted with under bar.

Dataset	Augmentation	Model	BP	Shallow	DFA	LAS	LTAS
CIFAR-10	flip	Conv3	**0.8451**	0.6668	0.7831	0.7960	0.8191
		Conv3+BN	**0.8476**	-	0.7838	0.8189	0.8167
		ResNet18	**0.8262**	-	0.7697	0.7765	0.7772
	random+cutout+flip	Conv3	**0.9100**	-	0.7925	0.8499	0.8682
		Conv3+BN	**0.9147**	-	0.8467	0.8578	0.8575
		ResNet18	**0.9339**	-	0.8790	0.8685	0.8726
CIFAR-100	flip	Conv3	0.5407	0.3886	0.4628	0.5400	0.5581
		Conv3+BN	**0.6425**	-	0.5042	0.5797	0.5816
		ResNet18	**0.5325**	-	0.4538	0.5026	0.4882
	random+cutout+flip	Conv3	**0.6433**	-	0.3979	0.6139	0.5631
		Conv3+BN	**0.7143**	-	0.5616	0.6367	0.6407
		ResNet18	**0.6960**	-	0.5427	0.6038	0.6030
TinyImageNet	random+cutout+flip	ResNet18	**0.5997**	-	0.3884	0.4678	0.4566

To train the networks, we set 64 batch sizes and perform a grid search on the learning rates for the Adam [5] optimizer. We use 100, 300, and 500 epochs for MNIST, CIFAR-10, and CIFAR-100, respectively, and random horizontal flip augmentation with 0.5 probability. Since the updates of the last two layers in the proposed methods are the same as in BP, we also compared the performance of a shallow network where only the last two layers are trained and others are fixed.

Table 1 shows test accuracies of the different learning methods for each dataset. For the MNIST dataset, all methods do not show much difference, but LAS performs best. For the CIFAR-10 data, all methods except shallow learning achieve a test accuracy of 0.6 or higher. In the case of CIFAR-100 data, however, we can see the performance degradation of DFA, which is similar to that of shallow learning. This is consistent with a previous study [18], which found that DFA performs poorly as the dimensionality of the target value increases due to poor alignment. It is noteworthy that the proposed methods overcome the limitation of DFA and show similar performance to BP.

Furthermore, for all tasks, LAS and LTAS show large performance differences with the shallow, which only learns the last two layers. This shows that the high performance of LAS and LTAS is not due to learning the last two layers in the same way as BP, but rather due to sending the appropriate learning signals to the earlier layers. Although there is still a small performance gap between BP and the proposed methods, it is consistently confirmed that the use of the local module can improve the performance of direct feedback learning while maintaining layer-wise parallel processing.

B. RESULT ON CONVOLUTIONAL NEURAL NETWORKS

In the previous works, it is known that DFA does not work well in CNN models due to the unique architecture of local connection and weight sharing [18]. To see the effect of the proposed local module on CNN learning, we trained three different CNN models (Conv3, Conv3+BN, and ResNet18) using CIFAR-10, CIFAR-100, and TinyImageNet [31] datasets.

The Conv3 model, used in Crafton et al. [16], has three convolution blocks and three fully connected layers. All convolution layers in this model have (5, 5) kernel size, (2, 2) stride, and (2, 2) zero padding. The Conv3+BN model has the same structure as the Conv3 model except for the addition of the batch normalization (BN) [6] layers after the convolution operation in each block. The ResNet18 [4] with skip connection is a model that is a common basis for modern CNN architectures. For this model, we use the block-wise direct feedback used in Launay et al. [23]. This feedback method propagates the error signal to convolutional blocks in a block-wise fashion and computes the same as BP within a block.

In training the CNN models, we performed a grid search to empirically optimize the learning rate and the weight decay term. We first trained CIFAR-10 and CIFAR-100 data with the random horizontal flip augmentation, which is the same as the fully connected network, and also tried an additional random augmentation [29] and a cutout augmentation [30].

Table 2 shows the test accuracies on CIFAR-10, CIFAR-100, and TinyImageNet. For learning with only random horizontal flip augmentation, the overall experimental results of DFA are similar to those reported by Crafton et al. in [16]. Compared to DFA, the proposed methods (LAS and LTAS) show slight improvements for CIFAR-10 dataset. For CIFAR-100 data, however, they significantly outperform DFA and even better than BP in the case of the Conv3 model. Notably, the ResNet18 shows lower performances than the simpler models, which is consistently observed for all learning methods (BP, DFA, LAS, and LTAS). This

TABLE 2. Test accuracy on fully connected networks. For each data set, the best result is written in bold fonts, and the second runner is written with under bar.

Method	MNIST	CIFAR-10	CIFAR-100
BP	0.9862	**0.6144**	**0.3376**
Shallow	0.9862	0.5831	0.2965
DFA	0.9850	0.6008	0.2997
LAS	**0.9871**	<u>0.6057</u>	<u>0.3253</u>
LTAS	<u>0.9864</u>	0.6054	0.3203

could be due to overfitting phenomena of the large model, and we performed additional augmentation to obtain better performance.

In the case of learning with three different types of data augmentation, we can observe a clear difference between DFA and the proposed methods. In particular, for CIFAR-100, the proposed methods show greater effects of data augmentation than DFA. It is noteworthy that the performance improvement is consistently observed for all models and tasks, including ResNet18 model and TinyImageNet data, which have not been investigated much in previous works on DFA. These promising results make it clear that the feedback weights learned by the proposed LAS and LTAS are more meaningful learning signals than the fixed random feedback weights.

Furthermore, it is shown that the performance gap between LAS and DFA increases as the difficulty of the task increases, which could be explained by the advantage of using adaptive feedback weights. The adaptive feedback weights can expand the search area in the parameter space to find a better solution than the fixed random feedback weight. This is related to the limitation of DFA addressed in the previous work [18].

While studies have shown that batch normalization is helpful in learning DFA [10], [20], others have shown that it is not necessary [17]. However, from the results in Table 2, it seems that batch normalization plays an important role in learning of DFA, LAS, and LTAS. These results suggest that other normalization techniques, such as layer normalization [32], can also be applied for further improvement.

V. CONCLUSION

In this paper, we proposed two direct feedback learning methods to improve the performance of DFA learning method, which can solve the weight transport problem and the backward locking problem. The proposed local modules, called LAS and LTAS, find a linear approximation of the nonlinear mapping from each hidden layer to the network output. The weight of the trained local module can give appropriate direction to which the direct feedback weight should be aligned. As a result, the proposed methods outperform traditional DFA on several benchmark datasets. In particular, for CNN models, the proposed method shows promising results, giving the possibility to apply direct feedback learning to convolutional blocks, which is known to be difficult. Nevertheless, the performance gap with BP still exists, and this may be due to the difficulty of approximating the complicated mapping of the convolutional layer by a simple linear module. In order to handle this problem, it would be possible to use more sophisticated local modules and loss functions. In addition, to increase the practical usability of the proposed method, further extension to more advanced networks such as MLP-mixer [33] will be done as a future work.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
[2] A. Meulemans, M. T. Farinha, J. G. Ordonez, P. V. Aceituno, J. Sacramento, and B. F. Grewe, "Credit assignment in neural networks through deep feedback control," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 4674–4687.
[3] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 1–12.
[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1–19.
[7] M. J. Filipovich, A. Cappelli, D. Hesslow, and J. Launay, "Scaling laws beyond backpropagation," 2022, *arXiv:2210.14593*.
[8] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognit. Sci.*, vol. 11, no. 1, pp. 23–63, Jan. 1987.
[9] Q. Liao, J. Leibo, and T. Poggio, "How important is weight symmetry in backpropagation?" in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 1–6.
[10] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Commun.*, vol. 7, no. 1, pp. 1–17, Nov. 2016.
[11] M. Akrout, C. Wilson, P. Humphreys, T. Lillicrap, and D. B. Tweed, "Deep learning without weight transport," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1–19.
[12] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1–9.
[13] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," *Nature Reviews Neurosci.*, vol. 21, pp. 335–346, Apr. 2020.
[14] J. F. Kolen and J. B. Pollack, "Backpropagation without weight transport," in *Proc. IEEE Int. Conf. Neural Netw. (ICNN)*, vol. 3, Jul. 1994, pp. 1375–1380.
[15] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu, "Decoupled neural interfaces using synthetic gradients," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1627–1635.
[16] B. Crafton, A. Parihar, E. Gebhardt, and A. Raychowdhury, "Direct feedback alignment with sparse connections for local learning," *Frontiers Neurosci.*, vol. 13, pp. 1–12, May 2019.
[17] J. Launay, I. Poli, and F. Krzakala, "Principled training of neural networks with direct feedback alignment," 2019, *arXiv:1906.04554*.
[18] M. Refinetti, S. d'Ascoli, R. Ohana, and S. Goldt, "Align, then memorise: The dynamics of learning with feedback alignment," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2021, pp. 8925–8935.
[19] C. Frenkel, M. Lefebvre, and D. Bol, "Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks," *Frontiers Neurosci.*, vol. 15, Feb. 2021, Art. no. 629892.

[20] D. Han and H.-j. Yoo, "Direct feedback alignment based convolutional neural network training for low-power online learning processor," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 2445–2452.
[21] D. Han and H.-J. Yoo, "DF-LNPU: A pipelined direct feedback alignment based deep neural network learning processor for fast online learning," in *On-Chip Training NPU-Algorithm, Architecture and SoC Design*. Cham, Switzerland: Springer, 2023, pp. 95–119.
[22] P. Baldi, P. Sadowski, and Z. Lu, "Learning in the machine: Random backpropagation and the deep learning channel," *Artif. Intell.*, vol. 260, pp. 1–35, Jul. 2018.
[23] J. Launay, I. Poli, F. Boniface, and F. Krzakala, "Direct feedback alignment scales to modern deep learning tasks and architectures," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 9346–9360.
[24] K. Jung, I. Baek, S. Kim, and Y. Dohn Chung, "LAFD: Local-differentially private and asynchronous federated learning with direct feedback alignment," *IEEE Access*, vol. 11, pp. 86754–86769, 2023.
[25] L. Deng, "The MNIST database of handwritten digit images for machine learning research [Best of the Web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[27] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Tech. Rep., 2009.
[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[29] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 3008–3017.
[30] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.
[31] Y. Le and X. Tiny, *Imagenet Visual Recognition Challenge*, document CS 231N, 2015.
[32] J. Lei Ba, J. Ryan Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[33] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "MLP-Mixer: An all-MLP architecture for vision," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 24261–24272.

HEESUNG YANG received the B.S. degree from the School of Computer Science and Engineering, Kyungpook National University, Daegu, South Korea, in 2022, where he is currently pursuing the M.S. degree with the School of Computer Science and Engineering. His current research interests include computational learning theory, deep learning, and their applications, such as medical image processing and protein density map sharpening.

SOHA LEE received the B.S. degree in mathematics and the M.S. degree in artificial intelligence from Kyungpook National University, Daegu, South Korea, in 2019 and 2021, respectively, where she is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. Her current research interests include deep learning and computational learning theory, such as predictive coding.

HYEYOUNG PARK received the B.S. degree (summa cum laude) and the M.S. and Ph.D. degrees in computer science from Yonsei University, Seoul, South Korea, in 1994, 1996, and 2000, respectively. From 2000 to 2004, she was a member of Research Staff with the Brain Science Institute, RIKEN, Japan. She is currently a Professor with the School of Computer Science and Engineering, Kyungpook National University, Daegu, South Korea. Her current research interests include computational learning theory, machine learning theory, and their application to various fields, such as pattern recognition, image processing, and data mining.