Efficient Target Propagation by Deriving Analytical Solution

Yanhao Bao[1], Tatsukichi Shibuya[1], Ikuro Sato[1,2], Rei Kawakami[1], Nakamasa Inoue[1]

[1] Tokyo Institute of Technology
[2] Denso IT Laboratory
bao.y.ab@m.titech.ac.jp, shibuya.t.ad@m.titech.ac.jp, isato@c.titech.ac.jp, reikawa@sc.e.titech.ac.jp, inoue@c.titech.ac.jp

Abstract

Exploring biologically plausible algorithms as alternatives to error backpropagation (BP) is a challenging research topic in artificial intelligence. It also provides insights into the brain's learning methods. Recently, when combined with well-designed feedback loss functions such as Local Difference Reconstruction Loss (LDRL) and through hierarchical training of feedback pathway synaptic weights, Target Propagation (TP) has achieved performance comparable to BP in image classification tasks. However, with an increase in the number of network layers, the tuning and training cost of feedback weights escalates. Drawing inspiration from the work of Ernoult et al., we propose a training method that seeks the optimal solution for feedback weights. This method enhances the efficiency of feedback training by analytically minimizing feedback loss, allowing the feedback layer to skip certain local training iterations. More specifically, we introduce the Jacobian matching loss (JML) for feedback training. We also proactively implement layers designed to derive analytical solutions that minimize JML. Through experiments, we have validated the effectiveness of this approach. Using the CIFAR-10 dataset, our method showcases accuracy levels comparable to state-of-the-art TP methods. Furthermore, we have explored its effectiveness in more intricate network architectures.

Introduction

Backpropagation (BP) (Rumelhart, Hinton, and Williams 1986) provides an efficient and feasible method for deep learning to train complex neural network models and has significantly contributed to the growth of the neural network domain. However, BP has been criticized since its widespread use as a biologically implausible algorithm because of its difficulty in making connections with the learning process of the human brain (Crick 1989; Roelfsema and Ooyen 2005). An issue with BP is pointed out that it lacks the utilization of local information (Lillicrap et al. 2020). Developing biologically plausible algorithms is important, not only to improve machine learning techniques but also to enhance our understanding of how the brain functions.

Former studies (Lillicrap et al. 2016; Nøkland 2016) have adjusted the backpropagation path, using a random matrix to propagate errors. These studies highlighted that a fixed random matrix can indeed support learning in neural networks, indicating that symmetric weight matrices in backpropagation might not be essential. However, follow-up studies (Bartunov et al. 2018; Launay et al. 2020) have suggested that, while this approach works for simpler tasks, it struggles when applied to more complex ones.

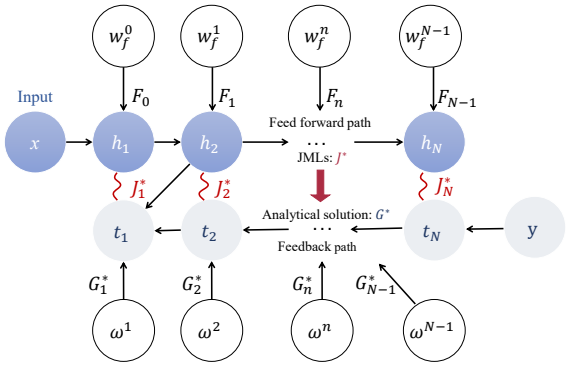

Figure 1. Illustration of the proposed model architecture. Minimizing JMLs provides G^*, the analytical feedback function, thereby avoiding the need for training the feedback network and enhancing the training efficiency of target propagation. See Section for variable definitions.

Target propagation (TP) (LeCun 1986; Le Cun, Galland, and Hinton 1988; Bengio 2014) is a biologically more plausible feedback algorithm than the error backpropagation which updates network parameters with layer-wise local loss. Many studies have shown that TP and its variants are effective for training deep neural networks (Bengio 2014; Lee et al. 2015; Ororbia and Mali 2019; Meulemans et al. 2020; Ernoult et al. 2022; Manchev and Spratling 2020). However, there is still room for improvement in the efficiency of training feedback networks. Similar to a series of work on Local Representation Alignment (LRA) (Ororbia et al. 2018; Ororbia and Mali 2019; Ororbia et al. 2020), the fixed-weight feedback network (Shibuya et al. 2023) that omits feedback parameter updates is the most efficient TP algorithm but it does not perform as well as BP.

In deep neural networks, a fundamental difficulty of feedback training is caused by the complexity of feedforward architectures. For example, even with simple convolutional networks such as VGGNet (Simonyan and Zisserman 2015)

and ResNet (He et al. 2016), feedback training for convolutional layers and pooling layers is not easy even with the difference target propagation (Lee et al. 2015) because their feedback architectures are nontrivial. Further, as the number of layers increases, feedback training with these layers tends to become unstable. This makes it difficult to scale TP algorithms.

In this paper, we propose a way to derive analytical solutions for feedback functions for TP algorithms. The key idea to improve efficiency is the use of the analytical solutions to the minimization problem with Jacobian matching losses (JMLs) to skip feedback training iterations. Our main contributions are summarized as follows.

1. We introduce JMLs between the feedforward and feedback functions and Jacobian matching conditions (JMCs) that help to analyze TP algorithms. The computational flow of feedforward and feedback functions is illustrated in Figure 1. We discuss the relation between JMLs/JMCs and TP algorithms.
2. We derive the analytical solutions to the JML minimization problems for commonly used layers such as convolutional layers, fully-connected layers, LayerNorm, etc.
3. We conduct experiments with LeNet (LeCun et al. 1989), Simplified-VGG (a simplified version of VGGNet (Simonyan and Zisserman 2015)), MLPMixer (Tolstikhin et al. 2021) on MNIST, Fashion-MNIST and CIFAR-10 datasets. We show that our proposal improves training efficiency by using the analytical solution for the feedback networks, avoiding the iterative training process. The accuracy of our method is on par with the state-of-the-art TP methods.

Related Work

Improving the biological plausibility of training algorithms is a challenging and fundamental research direction in the field of artificial intelligence and machine learning. In the past decade, Target propagation (TP) algorithms have been one of the most successful approaches from both biological and practical perspectives with their applications to deep neural networks.

The basic idea of TP that propagate targets was proposed in 1980s (LeCun 1986; Rohwer 1989). To make TP work with deep neural networks, difference target propagation (DTP) (Lee et al. 2015) has been a breakthrough approach, in which the difference correction mechanism is introduced to resolve the interference of reconstruction errors with target updates. There have been many variants of DTP. For instance, DTP-Sigma (Ororbia and Mali 2019), SDTP (Bartunov et al. 2018), GaitProp (Ahmad, van Gerven, and Ambrogioni 2020), DRL (Meulemans et al. 2020; Bengio 2020) have been proposed; however, it remains a question whether the feedback network can be properly learned.

Most recently, Ernoult et al. (2022) have shown that DTP with local difference reconstruction loss (LDRL) performs comparable with BP, achieving 89% in classification accuracy on the CIFAR-10 dataset with a VGG-like architecture. However, feedback training is computationally expensive. FWDTP (Shibuya et al. 2023) that fixes feedback weights is an efficient counterpart but it does not perform as well as BP in terms of image classification accuracy. In contrast to these previous studies focusing on training algorithms, this work mainly focuses on the feedback architectures that improve the efficiency of DTP.

Target Propagation Algorithms

Given an input vector x, we define a feedforward architecture F as follows:

$$F(x) = F_{N-1} \circ F_{N-2} \circ \cdots F_0(x), \tag{1}$$

where N is the number of layers and $F_n : \mathbb{R}^{d_n} \to \mathbb{R}^{d_{n+1}}$ is the feedforward function at the n^{th} layer. We denote by $\theta_n \in \mathbb{R}^{p_n}$, where θ_n is the flatten parameter vector of F_n and p_n is a number of the parameters. Feedforward activations are inductively defined as follows:

$$h_n = \begin{cases} x & (n = 0) \\ F_{n-1}(h_{n-1}) & (n = 1, 2, \cdots, N) \end{cases}. \tag{2}$$

Note that we have two Jacobian matrices[1]:

$$\partial_{h_n} F_n \in \mathbb{R}^{d_{n+1} \times d_n}, \ \ \partial_{\theta_n} F_n \in \mathbb{R}^{d_{n+1} \times p_n}. \tag{3}$$

This study considers supervised learning with a global loss function $\mathcal{L}_N(h_N, y)$ where y is the target (ground truth). The goal is to find feedforward parameters $\theta = \{\theta_n\}_{n=0}^{N-1}$ that minimize the global loss.

Target Propagation (Bengio 2014) For feedback propagation, TP uses the following targets:

$$t_n = \begin{cases} h_N - \beta \partial_{h_N} \mathcal{L} & (n = N) \\ G_n(t_{n+1}) & (n = N-1, \cdots, 0) \end{cases}, \tag{4}$$

where $G_n : \mathbb{R}^{d_{n+1}} \to \mathbb{R}^{d_n}$ is a feedback function and $\beta \in \mathbb{R}$ is a hyperparameter. We denote by ω_n the flattened parameter vector of the function G_n. With TP, the feedforward parameters θ_n are updated by minimizing the local mean squared error (MSE) loss:

$$\mathcal{L}_n(\theta_n) = \frac{1}{2\beta} \|t_{n+1} - h_{n+1}\|^2, \tag{5}$$

where the target t_{n+1} is treated as a constant when computing the gradient. The feedback parameters ω_n are updated by minimizing the local reconstruction loss (LRL):

$$\hat{\mathcal{L}}^{\text{LRL}}_{n,\epsilon}(\omega_n) = \frac{1}{2} \|r_{n,\epsilon} - h_{n,\epsilon}\|^2, \tag{6}$$

where $h_{n,\epsilon} = h_n + \epsilon$ is a noisy activation, $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$ is a Gaussian noise, and $r_{n,\epsilon} = G_n \circ F_n(h_{n,\epsilon})$ is the locally reconstructed activation.

Difference Target Propagation (Lee et al. 2015) To stabilize training with non-invertible feedforward functions, DTP introduces *difference correction* by which targets are propagated as follows:

$$t_n = \begin{cases} h_N - \beta \partial_{h_N} \mathcal{L} & (n = N) \\ \tilde{G}_n(t_{n+1}; h_n) & (n = N-1, \cdots, 0) \end{cases}, \tag{7}$$

[1] $\partial_{h_n} F_n$ indicates $\frac{\partial}{\partial h_n} F_n(h_n)$.

Algorithm 1: Local Difference Reconstruction Loss (LDRL)

for $n = 1$ to L **do**
  Training feedback mapping of layer n
  for $i = 1$ to N **do**
    $h_{n+1} = F_n(h_n)$
    $\epsilon, \eta \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$
    $\hat{r}_{n,\epsilon} = G_n(F_n(h_n + \epsilon)) - G_n \circ F_n(h_n) + h_n$
    $\tilde{r}_{n,\eta} = G_n(F_n(h_n) + \eta) - G_n \circ F_n(h_n) + h_n$
    Update feedback weights by using $\hat{\mathcal{L}}^{\mathrm{LDRL}}_{n,\epsilon,\eta}$.
  end for
end for

where

$$\tilde{G}_n(t_{n+1}; h_n) = G_n(t_{n+1}) + h_n - G_n \circ F_n(h_n) \tag{8}$$

is the difference correction function. To update parameters, the local losses in Eqs. (5, 6) are used in the same way as TP.

Fixed-Weight Difference Target Propagation (Shibuya et al. 2023) Fixed-Weight Difference Target Propagation (FWDTP) is defined as an algorithm that does not use reconstruction loss for updating feedback weights in DTP. All feedback weights are first randomly initialized and then kept constant during training. This research has shown that FWDTP is effective in propagating target values and eliminates the need for the feedback network training process.

Local Difference Reconstruction Loss (Ernoult et al. 2022) To bridge the gap between TP and BP, local difference reconstruction loss (LDRL) has been found helpful (see Algorithm 1). It is defined by the sum of two loss functions:

$$\hat{\mathcal{L}}^{\mathrm{LDRL}}_{n,\epsilon,\eta} = \hat{\mathcal{L}}^{\mathrm{IP}}_{n,\epsilon} + \hat{\mathcal{L}}^{\mathrm{RR}}_{n,\eta}, \tag{9}$$

where

$$\hat{\mathcal{L}}^{\mathrm{IP}}_{n,\epsilon}(\omega_n) = -\epsilon^\top \cdot (\hat{r}_{n,\epsilon} - h_n), \tag{10}$$

$$\hat{\mathcal{L}}^{\mathrm{RR}}_{n,\eta}(\omega_n) = \frac{1}{2}\|\tilde{r}_{n,\eta} - h_n\|^2. \tag{11}$$

Note that LDRL is used with DTP targets in Eq. (7). Feedback parameters are updated to minimize the layer-wise LDRL. Minimizing the LDRL ensures that the Jacobian of the feedback network converges to the transposed Jacobian of the feedforward network; thus, it approximates backpropagation gradients.

Although the LDRL algorithm shows an accuracy comparable to BP on the CIFAR-10 dataset, during the feedback path training process, to ensure the correct relationship between the Jacobian matrix of forward and backpropagation, the algorithm needs to traverse each layer multiple times. This approach may increase training time and complexity of parameter tuning.

Assigning Analytical Solution Instead of Training the Feedback Parameters

In this section, we define Jacobian matching losses (JMLs) between the feedforward function F_n and feedback function G_n for feedback training. We also define Jacobian matching conditions (JMCs) that help to analyze TP algorithms, and show the relation between JMLs/JMCs and conventional feedback loss functions we described in Section 3. Based on the different JMLs defined below, to address the lengthy training time of feedback path, we propose a method to update the parameters of each layer using an analytical solution, thereby bypassing traditional training methods for the feedback network.

Jacobian Matching Loss

Definition 1 (IJML) *Suppose that Jacobian $\partial_{h_n} F_n$ is a regular matrix. We define the inverse Jacobian matching loss (IJML) as follows:*

$$\mathcal{J}^I(G_n, F_n) \triangleq \left\| \partial_{h_{n+1}} G_n - (\partial_{h_n} F_n)^{-1} \right\|_{\mathrm{F}}, \tag{12}$$

where $\|A\|_{\mathrm{F}}$ is the Frobenius norm of a matrix A. We say that the feedback function G_n satisfies inverse Jacobian matching condition (IJMC) w.r.t F_n iff $\mathcal{J}^I(G_n, F_n) = 0$.

Definition 2 (PJML) *We define the pseudo-inverse Jacobian matching loss (PJML) as follows:*

$$\mathcal{J}^P(G_n, F_n) \triangleq \left\| \partial_{h_{n+1}} G_n - (\partial_{h_n} F_n)^{\dagger} \right\|_{\mathrm{F}}, \tag{13}$$

where $A^\dagger$ is the Moore-Penrose inverse of A. We say that G_n satisfies the pseudo-inverse Jacobian matching condition (PJMC) w.r.t F_n iff $\mathcal{J}^P(G_n, F_n) = 0$.

Definition 3 (TJML) *We define the transposed Jacobian matching loss (TJML) as follows:*

$$\mathcal{J}^T(G_n, F_n) \triangleq \left\| \partial_{h_{n+1}} G_n - (\partial_{h_n} F_n)^{\top} \right\|_{\mathrm{F}}. \tag{14}$$

We say that G_n satisfies the transposed Jacobian matching condition (TJMC) w.r.t F_n iff $\mathcal{J}^T(G_n, F_n) = 0$.

The relation between JMCs and feedback losses used in TP algorithms can be explained by the following propositions. Note that Prop. 2 and Prop. 3 are inspired by the works of DRL (Meulemans et al. 2020) and LDRL (Ernoult et al. 2022).

Proposition 1 *Suppose that the constant terms of F_n and G_n are zero. IJMC is satisfied iff LRL is zero, i.e.,*

$$\forall \epsilon \;\; \hat{\mathcal{L}}^{\mathrm{RL}}_{n,\epsilon} = 0 \iff \mathcal{J}^I(G_n, F_n) = 0. \tag{15}$$

Proposition 2 *PJMC is satisfied iff the expectation of the DRL converges to zero as the variance of noise approaches zero, i.e.,*

$$\lim_{\sigma \to 0} \frac{1}{\sigma^2} \mathbb{E}_{\epsilon,\eta} \left[\hat{\mathcal{L}}^{\mathrm{DRL}}_n \right] = 0 \iff \mathcal{J}^P_n(\omega_n) = 0, \tag{16}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

Proposition 3 *TJMC is satisfied iff the expectation of the LDRL converges to zero as the variance of noise approaches zero, i.e.,*

$$\lim_{\sigma \to 0} \frac{1}{\sigma^2} \mathbb{E}_{\epsilon,\eta} \left[\hat{\mathcal{L}}^{\mathrm{LDRL}}_n \right] = 0 \iff \mathcal{J}^T_n(\omega_n) = 0, \tag{17}$$

where $\epsilon, \eta \sim \mathcal{N}(0, \sigma^2)$.

Algorithm 2: Analytical Feedback Function

Require: Learning Rate α_f, Stepsize β, Loss $\mathcal{L}$, Parameters of feedforward function W_f

Compute all activations
for $n = 1$ to L **do**
  $h_{n+1} = F_n(h_n)$
end for
Solving analytical solution for feedback function
for $n = L - 1$ to 2 **do**
  $G_n^* = \underset{H}{\operatorname{argmin}}\, \mathcal{J}^*(H, F_n)$
end for
Compute first target $t_L \leftarrow h_L - \beta \nabla_{h_L} \mathcal{L}_(h_L, y)$
for $n = L - 1$ to 1 **do**
  $t_n \leftarrow G_{n+1}^*(t_{n+1}) + h_n - G_{n+1}^*(h_{n+1})$
end for
Training feedforward function
for $n = 1$ to L **do**
  $\mathcal{L} = \|h_n - t_n\|_2^2$
  $W_f \leftarrow W_f - \alpha_f \nabla_{W_f} \mathcal{L}$
end for

Proposed Method

To improve the training efficiency of the feedback network, the key idea lies in how to optimize the training process. Specifically, we define a training rule that uses the analytical feedback function G_n^* to update the feedback parameters, allowing us to bypass certain training steps. This rule uses analytical solutions obtained by solving the minimization problem with JMLs. We define analytical feedback functions as follows.

Definition 4 (Analytical feedback function) *Given a feedforward function F_n, feedback function H, we define a feedback function that minimizes JMLs as follows:*

$$G_n^* = \underset{H}{\operatorname{argmin}}\, \mathcal{J}^*(H, F_n), \tag{18}$$

where $* \in \{I, P, T\}$ and $\mathcal{J}^*$ is JML (Definitions 1 to 3).

Algorithm 2 represents our proposed method. Our procedure is divided into the following steps: Initially, we calculate the activation for each layer. Subsequently, we establish the feedback network that minimizes the JML as described by the analytical feedback function in Eq. (18). Finally, we compute the target for each layer and update the forward network parameters. Since each layer of the feedback network updates the parameters based on the solutions of Eq. (18), we no longer need to compute feedback loss. Therefore, the efficiency of network parameter training is greatly improved. However, solving Eq. (18) is not always straightforward.

Limitations of Analytical Feedback Update

The time complexity of matrix inversion and the requirement of matrix invertibility will impose many constraints on the architecture of the network.

Time Complexity Inverting large matrices is computationally expensive. PyTorch uses SVD for matrix inversion, and the time complexity depends on the matrix size and the algorithm used. When n is close to m in an $m \times n$ matrix, the complexity can reach $O(n^3)$. IJMC and PJMC methods involve frequent matrix inversions, significantly increasing training time. Figure 2 shows the matrix dimension effects on the inverse, pseudo-inverse, and transpose matrix calculations. Square matrices $n \times n$ ($100 \leq n \leq 500$) were used. As can be seen, the transpose matrix calculation was fast, while inverse and pseudo-inverse computation times grew exponentially. This result highlights the time-consuming nature of matrix inversion in PJMC and IJMC methods, which use pseudo-inverse and inverse solutions, respectively.

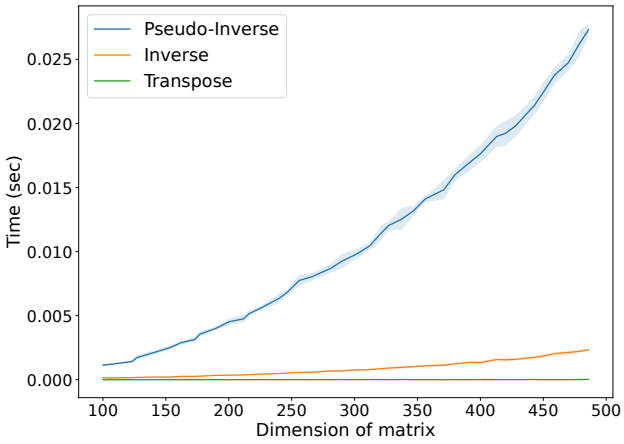

Figure 2. The relationship between the dimensions of the input matrix and the calculation time for solving the corresponding inverse, pseudo-inverse and transpose matrices.

Matrix Invertibility An invertible matrix must have a non-zero determinant, which constrains the alignment of input and output dimensions in neural networks. Networks with matching dimensions are often suited for tasks like dimensionality reduction and restoration, potentially limiting their ability to capture complex data patterns. Furthermore, changing input and output dimensions poses challenges in designing the feedback function. Therefore, our method exclusively focuses on the TJMC case in subsequent sections.

Deriving Analytical Solutions for Feedback Functions of TP

In this section, we explore the feedback network architectures of common network layers under the TJMC for target propagation. We specifically choose LeNet (LeCun et al. 1989), Simplified-VGG, and MLPMixer (Tolstikhin et al. 2021) to verify the effectiveness of our newly introduced method. The layers in these networks cover the most common layers used in image classification. Below, we individually analyze the architecture of each network, as well as the analytical solutions for their corresponding feedback paths. Note that skip connection is omitted following previous studies of TP algorithms (Lee et al. 2015; Meulemans et al. 2020; Ernoult et al. 2022) but we delve into an empirical analysis of them in the experiments.

Analytical Solutions of Feedback Path for Common Layers

Let x be the input to the network layer. The function of the feedforward layer is denoted as F. Given an analytical solution that satisfies TJMC, the corresponding feedback network layer can be defined in G^*.

Convolution Layer The feedforward and feedback paths of the convolution layer can be defined as follows:

$$F_{\text{Conv}}(x) = W_{\text{Conv}} * x, \tag{19}$$

$$G^*_{\text{Conv}}(x) = \text{ROT}(W_{\text{Conv}}) * x, \tag{20}$$

where $*$ represents the convolution operation, and ROT denotes the rotation of the convolution kernel by 180 degrees. Note that when the number of input channels and the number of output channels are different, we need to rearrange the relative positions of each channel of convolution kernels.

MaxPooling Layer Due to the different dimensions of the feedforward and feedback Jacobian matrices, finding an analytical solution for the weights of the feedback network is not straightforward. To reduce the discrepancy between the input matrix and its reconstructed counterpart following the feedback function, we utilize the MaxUnpool function:

$$F_{\text{MaxPool}}(x) = \text{MaxPool}(x), \tag{21}$$

$$G^*_{\text{MaxPool}}(x) = \text{MaxUnpool}(x). \tag{22}$$

The MaxUnpool function uses the output of MaxPool and the index of the maximum value, to reconstruct the original matrix (Badrinarayanan, Kendall, and Cipolla 2017).

Fully-connected Layer The feedforward and feedback paths of fully-connected (FC) layer can be defined as

$$F_{\text{FC}}(x) = Wx, \tag{23}$$

$$G^*_{\text{FC}}(x) = W^\top x, \tag{24}$$

where the parameters of the feedback network layer are defined as the transpose of the parameters of the forward layer.

Activation Function The feedforward and feedback paths of element-wise activation functions can be defined as

$$F_{\text{Act}}(x) = \sigma(x), \tag{25}$$

$$G^*_{\text{Act}}(x) = \varsigma(x), \tag{26}$$

where

$$\varsigma(x) = \int \sigma'(\sigma^{-1}(x))dx. \tag{27}$$

LeakyReLU (Xu et al. 2015) is used as the default activation function because GELU (Hendrycks and Gimpel 2016), used in the original MLP-Mixer, is not invertible.

Hadamard Product of Activation Function To enhance stability, we modified the feedback process of the original activation function in MLP-Mixer. We added an Hadamard Product to achieve better performance in the feedback path:

$$F_{\text{Act}}(x) = \sigma(x), \tag{28}$$

$$G^*_{\text{Act}}(x) = \sigma(\sigma^{'}(x) \odot t), \tag{29}$$

where t represents the target input for the feedback path, and x denotes the input for the feedforward path.

LayerNorm Layer Here we formulate LayerNorm by decomposing it into mean normalization F_{Mean} and standard-deviation (SD) normalization F_{SD} as follows:

$$F_{\text{Norm}}(x) = F_{\text{SD}} \circ F_{\text{Mean}}(x). \tag{30}$$

Mean normalization is defined as:

$$F_{\text{Mean}}(x) = Wx, \;\; W = I - \frac{1}{d_n}\mathbf{1}, \tag{31}$$

where I is the identity matrix and $\mathbf{1}$ is the square matrix with all elements equal to one. The feedback path of mean normalization can be defined as:

$$G^*_{\text{Mean}}(x) = Wx. \tag{32}$$

SD normalization is defined as:

$$F_{\text{SD}}(x) = \frac{1}{\sqrt{s}}x, \;\; s = \frac{1}{d_2}x^\top x. \tag{33}$$

The feedback function for SD normalization is defined as:

$$G^*_{\text{SD}}(x; s) = \frac{1}{\sqrt{s}}x. \tag{34}$$

Here, s is obtained from Eq. (33) and needs to be frozen; that is, the standard deviation s is computed in the feedforward path and reused in the feedback path. This technique is similar to difference correction (Lee et al. 2015) in Eq. (7). Therefore, the feedback path of LayerNorm can be defined as follows:

$$G^*_{\text{Norm}}(x) = G^*_{\text{Mean}}(x) \circ G^*_{\text{SD}}. \tag{35}$$

Global Average Pooling The feedforward and feedback paths of the global average pooling (GAP) layer can be defined as follows:

$$F_{\text{GAP}}(x) = \frac{1}{d_n}\mathbf{1}^\top x, \tag{36}$$

$$G_{\text{GAP}}(x) = \frac{1}{d_n}\mathbf{1}x, \tag{37}$$

where $\mathbf{1}$ is the square matrix with all elements equal to one.

Skip Connection The feedforward and feedback paths of the skip connection layer can be defined as follows:

$$F_{\text{Skip}}(x) = f(x) + x \tag{38}$$

$$G^*_{\text{Skip}}(x) = g^*(x) + x \tag{39}$$

where f and g^* satisfy the TJMC.

Composite Function Suppose the Composite Function in the feedforward path can be written as follows:

$$F_{\text{Com}}(x) = f_2 \circ f_1(x), \tag{40}$$

its corresponding feedback network can be written as follows:

$$G^*_{\text{Com}}(x) = g^*_1 \circ g^*_2(x), \tag{41}$$

where g^*_1 and g^*_2 satisfy TJMC w.r.t. f_1 and f_2 respectively.

Block Architecture

In LeNet and Simplified-VGG, the basic block consists of only two functions:

$$F_{\text{Block}} = F_{\text{MaxPool}} \circ F_{\text{Conv}}, \tag{42}$$

where F_{Conv} represents the Convolution Layer, F_{MaxPool} is MaxPooling layer. Similarly, its corresponding feedback block also consists of two functions:

$$G^*_{\text{Block}} = G^*_{\text{Conv}} \circ G^*_{\text{MaxPool}}, \tag{43}$$

where G^*_{Block} serves as a fundamental unit for propagating the target forward through the DTP algorithm.

In MLPMixer, the basic block comprises two parts: Channel MLP and Spatial MLP, each with its corresponding skip connection. The skip connection also plays a fundamental role within the DTP algorithm.

Channel MLP The block composed of Channel MLP consists of four feedforward functions, and it can be defined as follows:

$$F_{\text{Channel}}(x) = F_{\text{FC}} \circ F_{\text{Act}} \circ F_{\text{FC}} \circ F_{\text{Norm}}(x) + x, \tag{44}$$

and its corresponding feedback block can be defined as follows:

$$G^*_{\text{Channel}}(x) = G^*_{\text{Norm}} \circ G^*_{\text{FC}} \circ G^*_{\text{Act}} \circ G^*_{\text{FC}}(x) + x. \tag{45}$$

Spatial MLP In addition to the transpose operation, the block composed of Spatial MLP consists of three feedforward functions, and it can be defined as follows:

$$F_{\text{Spatial}}(x) = F_{\text{Act}} \circ F_{\text{FC}} \circ F_{\text{Norm}}(x) + x, \tag{46}$$

here the expression of transpose has been omitted. Its corresponding feedback block can be defined as follows:

$$G^*_{\text{Spatial}}(x) = G^*_{\text{Norm}} \circ G^*_{\text{FC}} \circ G^*_{\text{Act}}(x) + x, \tag{47}$$

where G^*_{Channel} and G^*_{Spatial} serve as two fundamental units for propagating the target forward through the DTP algorithm.

Experiments

In this section, we evaluate the effectiveness of our proposal using the LeNet, Simplified-VGG, and MLPMixer models on the MNIST, Fashion-MNIST (FMNIST), and CIFAR-10 datasets. In terms of training time, we plan to benchmark our proposal against the LDRL method (Ernoult et al. 2022).

Network Architecture In the experiments, we used LeNet, Simplified-VGG, and MLPMixer to validate our proposal. In Section 5, we have introduced the fundamental unit of the DTP algorithm for LeNet and Simplified-VGG. This unit, crucial for delivering the target value, combines a convolution layer with MaxPooling. The LeNet architecture comprises two sets of fundamental units followed by fully connected layers. The Simplified-VGG architecture comprises five groups of these basic units. For these architectures, the convolution kernel size in LeNet is 5×5, while in Simplified-VGG, it is 3×3. The MLPMixer architecture comprises an embedding layer, along with two fundamental units and the mean layer.

Note that the mean layer in MLPMixer serves the same function as the previously analyzed Global Average Pooling (GAP) layer in Section 5. Thus, its feedforward and feedback functions align with those of GAP. Furthermore, the Mean layer also acts as a fundamental unit for propagating the target forward through the DTP algorithm. However, a key distinction is the absence of learnable parameters in the mean layer, allowing the training step to be skipped.

Comparison with SOTA on LeNet and Simplified-VGG This experiment compares our method with state-of-the-art TP algorithms, LDRL (Ernoult et al. 2022) and FWDTP (Shibuya et al. 2023) on LeNet and Simplified-VGG. We used MNIST, FMNIST, and CIFAR-10 datasets to report image classification accuracy and speed-up ratio of feedback training.

Tables 1 and 2 detail the performance metrics and experimental results of our proposal, respectively. When validated with the LeNet, our proposal achieves accuracy on par with and even surpasses the state-of-the-art TP methods, and reduces the training time of the network by approximately 60% while achieving 93% on MNIST dataset. For the results of Simplified-VGG, in Table 2, our proposal outperforms LDRL in terms of training accuracy. The higher training accuracy can be attributed to the analytical solution of the feedback network we defined, which promotes faster convergence. However, comparing the results with test accuracy, our method maintains a competitive performance, but overfitting may have occurred. When compared with LDRL in terms of training speed, our method greatly reduces the training time by approximately 75%.

Comparison with SOTA on MLPMixer This experiment compares our method with other TP algorithms on MLPMixer. We used CIFAR-10 dataset to report image classification accuracy and the speed-up ratio of feedback training.

In our experiments with the LDRL method, we utilized the skip connection as the fundamental unit, as defined in Section 5.4, for target propagation with DTP. We observed that its accuracy was lower than the results presented in Table 3. Therefore, exclusively in the LDRL method, we employed the feedback block as a unit for reverse target value transfer, which comprises two skip connections.

The results presented in Table 3 also clearly verify the effectiveness of our proposal. In terms of accuracy, our method outperforms other state-of-the-art target propagation methods with fast training speed, reducing the training time by approximately 75%.

Conclusion and Discussion

In this study, our main goal is to improve the training efficiency of target propagation feedback networks, i.e., to reduce the training duration while maintaining or surpassing the performance of state-of-the-art target propagation algorithms. To achieve this goal, we have presented a method to update the parameters of the feedback network using the analytical solution. We analyze the network layer by layer and provide solutions for common layers. Our experiments verify that the proposed TJMC method satisfies these criteria.

Method	MNIST			FMNIST			CIFAR-10		
	Train	Test	Speed	Train	Test	Speed	Train	Test	Speed
LDRL (Ernoult 22)	$100.0^{\pm 0.00}$	$\mathbf{99.3^{\pm 0.05}}$	$\times 1.00$	$100.0^{\pm 0.00}$	$90.7^{\pm 0.26}$	$\times 1.00$	$98.2^{\pm 2.48}$	$74.7^{\pm 0.60}$	$\times 1.00$
FWDTP (Shibuya 23)	$100.0^{\pm 0.00}$	$98.6^{\pm 0.04}$	$\times 3.57$	$98.1^{\pm 1.47}$	$88.3^{\pm 0.18}$	$\times 25.00$	$80.1^{\pm 1.92}$	$66.5^{\pm 0.63}$	$\times 10.00$
TJML-DTP (Ours)	$100.0^{\pm 0.00}$	$\mathbf{99.3^{\pm 0.05}}$	$\times 3.45$	$100.0^{\pm 0.00}$	$\mathbf{91.4^{\pm 0.07}}$	$\times 14.29$	$100.0^{\pm 0.00}$	$\mathbf{76.7^{\pm 0.22}}$	$\times 3.45$
Back Propagation	$100.0^{\pm 0.00}$	$99.3^{\pm 0.02}$	-	$100.0^{\pm 0.00}$	$92.7^{\pm 0.17}$	-	$100.0^{\pm 0.00}$	$81.4^{\pm 0.13}$	-

Table 1. Comparison on LeNet. Accuracies and relative training speed on MNIST, FMNIST and CIFAR-10 are reported. We report results with LDRL (Ernoult et al. 2022), FWDTP (Shibuya et al. 2023) and our analytical feedback method (TJML-DTP). All results are averaged over five different seeds.

Method	Train	Test	Speed
LDRL (Ernoult 22)	$98.7^{\pm 0.65}$	$\mathbf{89.5^{\pm 0.54}}$	$\times 1.00$
FWDTP (Shibuya 23)	$77.5^{\pm 1.72}$	$62.4^{\pm 0.56}$	$\times 5.88$
TJML-DTP (Ours)	$100.0^{\pm 0.00}$	$85.2^{\pm 0.21}$	$\times 5.00$
Back Propagation	$100.0^{\pm 0.00}$	$89.0^{\pm 0.22}$	-

Table 2. Comparison on Simplified-VGG. Accuracies and training speed on CIFAR-10 are reported. All results are averaged over five different seeds.

Method	Train	Test	Speed
LDRL (Ernoult 22)	$55.9^{\pm 1.13}$	$37.3^{\pm 1.48}$	$\times 1.00$
FWDTP (Shibuya 23)	$81.6^{\pm 1.30}$	$56.1^{\pm 0.35}$	$\times 4.17$
TJML-DTP (Ours)	$80.2^{\pm 3.17}$	$\mathbf{56.9^{\pm 0.49}}$	$\times 4.00$
Back Propagation	$81.8^{\pm 0.71}$	$69.1^{\pm 0.52}$	-

Table 3. Comparison on MLPMixer. Accuracies and training speed on CIFAR-10 are reported. All results are averaged over five different seeds.

Method	Test
Reversible Structure	$42.8^{\pm 0.18}$
TJML-DTP (Ours)	$\mathbf{56.9^{\pm 0.49}}$

Table 4. Comparison with reversible structure. An MLPMixer architecture is used. CIFAR-10 test accuracies are reported.

Below, we summarize the effectiveness of our proposal as well as other possible approaches and future work on speeding up the feedback function.

Effectiveness of JMCs We demonstrate how our studies relate to previous research, highlighting our method's effectiveness. IJMC stems from TP's basic definition, using an ideally perfect inverse feedback network to inform its design. PJMC, to a certain extent, references the research conclusions of DRL. Meulemans et al. (Meulemans et al. 2020) noted that when utilizing DRL functions to train feedback networks and using DTP to transfer the target and train the feedforward function parameters, the parameter update can meet the Gauss-Newton optimization. PJMC synthesized these conclusions. TJMC is drawn from the findings of LDRL. Ernoult et al. (Ernoult et al. 2022) explored the connection between TP and BP based on the work of Meulemans et al., and pointed out that LDRL is proposed to train the feedback network parameters, ensuring that the Jacobian of the feedforward function and the feedback function satisfy the transposition relationship. When using DTP to transfer the target and train the parameters of the feedforward function, the parameter update is then linked to BP. TJMC is derived from these conclusions and demonstrated the best results.

Relation between TJMC and BP Backpropagation involves a two-step process. It first obtains the output of the network through feedforward function, then it uses this output in the backpropagation step to calculate gradients and subsequently update the parameters of each layer. On the other hand, the rationale behind target propagation is to generate a target for each layer instead of directly computing the gradient of each individual weight. But the weights of each layer are updated based on its corresponding target using backpropagation so that the output of the layer is consistent with that target. Our proposed TJMC method is designed to bypass the training of the feedback function, thereby accelerating the overall training process. While the feedforward and feedback function aligns with the TJMC, the resultant activation updates might seem akin to those in the backpropagation process. However, at its core, it adheres to the fundamental principle of target propagation, which is to update the weights according to the target value of each layer, rather than directly computing the gradient of each weight.

Further exploration We have explored replacing the forward pass with a reversible structure (Dinh, Krueger, and Bengio 2014; Dinh, Sohl-Dickstein, and Bengio 2016; Kingma and Dhariwal 2018) to skip feedback training. Ensuring the consistency of input and output dimensions within this reversible structure is crucial. We conducted an experiment using a network with four MLPMixer feedforward blocks, transforming it into a reversible structure. The experimental results are shown in Table 4. The performance has not reached the state of the art, but the use of reversible structure with our method is an interesting future direction.

Acknowledgements

This work was an outcome of a research project, Development of Quality Foundation for Machine-Learning Applications, supported by DENSO IT LAB Recognition and Learning Algorithm Collaborative Research Chair (Tokyo Tech.). This work was also supported by JSPS KAKENHI Grant Number JP22H03642.

References

Ahmad, N.; van Gerven, M.; and Ambrogioni, L. 2020. GAIT-prop: A biologically plausible learning rule derived from backpropagation of error. In *NeurIPS*.

Badrinarayanan, V.; Kendall, A.; and Cipolla, R. 2017. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12): 2481–2495.

Bartunov, S.; Santoro, A.; Richards, B.; Marris, L.; Hinton, G.; and Lillicrap, T. 2018. Assessing the Scalability of Biologically-Motivated Deep Learning Algorithms and Architectures. In *NeurIPS*.

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

Bengio, Y. 2020. Deriving Differential Target Propagation from Iterating Approximate Inverses. *arXiv preprint arXiv:2007.15139*.

Crick, F. 1989. The recent excitement about neural networks. *Nature*, 337: 129–132.

Dinh, L.; Krueger, D.; and Bengio, Y. 2014. NICE: Nonlinear Independent Components Estimation. In *ICLR*.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2016. Density Estimation using Real NVP.

Ernoult, M.; Normandin, F.; Moudgil, A.; Spinney, S.; Belilovsky, E.; Rish, I.; Richards, B.; and Bengio, Y. 2022. Towards Scaling Difference Target Propagation by Learning Backprop Targets. In *ICML*.

He, K.; Ren, S.; Sun, J.; and Zhang, X. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative Flow with Invertible 1x1 Convolutions. In *NeurIPS*.

Launay, J.; Iacopo, P.; Francois, B.; and Krzakala, F. 2020. Direct Feedback Alignment Scales to Modern Deep Learning Tasks and Architectures. *arXiv preprint arXiv:2006.12878*.

Le Cun, Y.; Galland, C.; and Hinton, G. E. 1988. GEMINI: Gradient Estimation Through Matrix Inversion After Noise Injection. In *NeurIPS*.

LeCun, Y. 1986. Learning processes in an asymmetric threshold network. *Disordered systems and biological organization*.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4): 541–551.

Lee, D.-H.; Zhang, S.; Fischer, A.; and Bengio, Y. 2015. Difference Target Propagation. In *ECML/PKDD*.

Lillicrap, T.; Cownden, D.; Tweed, D.; and Akerman, C. 2016. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7.

Lillicrap, T.; Santoro, A.; Marris, L.; Akerman, C.; and Hinton, G. 2020. Backpropagation and the brain. *Nature*, 21: 335–346.

Manchev, N.; and Spratling, M. 2020. Target Propagation in Recurrent Neural Networks. *Journal of Machine Learning Research*, 21: 1–33.

Meulemans, A.; Carzaniga, F. S.; Suykens, J. A.; Sacramento, J.; and Grewe, B. F. 2020. A theoretical framework for target propagation. In *NeurIPS*.

Nøkland, A. 2016. Direct Feedback Alignment Provides Learning in Deep Neural Networks. In *NeurIPS*.

Ororbia, A. G.; and Mali, A. 2019. Biologically Motivated Algorithms for Propagating Local Target Representations. In *AAAI*.

Ororbia, A. G.; Mali, A.; Giles, C. L.; and Kifer, D. 2020. Continual Learning of Recurrent Neural Networks by Locally Aligning Distributed Representations. *IEEE Transactions on Neural Networks and Learning Systems*.

Ororbia, A. G.; Mali, A.; Kifer, D.; and Giles, C. L. 2018. Conducting Credit Assignment by Aligning Local Representations. *arXiv preprint arXiv:1803.01834*.

Roelfsema, P. R.; and Ooyen, A. v. 2005. Attention-Gated Reinforcement Learning of Internal Representations for Classification. *Neural Computation*, 17(10): 2176–2214.

Rohwer, R. 1989. The "moving targets" training algorithm. In *NeurIPS*.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature*, 323: 533–536.

Shibuya, T.; Inoue, N.; Kawakami, R.; and Sato, I. 2023. Fixed-Weight Difference Target Propagation. In *AAAI*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Tolstikhin, I.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Keysers, D.; Uszkoreit, J.; Lucic, M.; et al. 2021. Mlp-mixer: An all-mlp architecture for vision. In *NeurIPS*.

Xu, B.; Wang, N.; Chen, T.; and Li, M. 2015. Empirical Evaluation of Rectified Activations in Convolutional Network. In *ICML Deep Learning Workshop*.