# Weight Difference Propagation for Stochastic Gradient Descent Learning

Shahrzad Mahboubi, Hiroshi Ninomiya
*Graduate School of Electrical and Information Engineering,*
*Shonan Institute of Technology*
Fujisawa, Kanagawa, Japan
Email : mahboubi.shaa@gmail.com, ninomiya@info.shonan-it.ac.jp

***Abstract*—This paper proposes a new stochastic (mini-batch) training algorithm to reduce the computational and hardware implementation costs of the error Back Propagation (BP) method. In recent years, with the rapid development of IoT, there has been an increasing necessity to use microcomputers, especially edge computers that implement neural networks capable of training large-scale data. Since the neural network training is based on the BP method, the error propagation architecture from the output to the input layers is required for each training sample to calculate the gradient. As a result, the hardware and computational costs are increased. This paper attempts to improve the BP method by using the inner product of the weights and their updated amounts (differences) for training reducing the hardware and computational costs. This method eliminates the requirement of the BP architecture for each training sample in mini-batch and calculates the amounts of the weight update with only one weight difference propagation. This means that the proposed method can reduce the computational complexity in the backward calculations of the training to $1/b$ ($b$ is the mini-batch size) compared to BP. Computer simulations demonstrate the effectiveness of the proposed method.**

***Index Terms*—neural network, gradient-based training algorithm, stochastic gradient descent method, error back propagation, weight difference propagation**

## I. Introduction

With the development of information and communication technology, various devices have been converted to IoT, making it possible to acquire and store diverse and vast amounts of data. The complexity (nonlinearity) of data and its volume increase day by day. Developing technologies can process large-scale nonlinear data with high accuracy and speed. In recent years, Artificial Intelligence (AI) has attracted attention as one of the technologies that make this possible and has been applied in various fields, including mechanical engineering, statistics, physics, economics, cognitive science, and brain science. The core technology in the rapid development of AI is neural networks (NNs) [1] [2]. NNs are generally trained using gradient algorithms based on the error Back Propagation (BP) method. BP training is an efficient and practical algorithm, and various improvement methods have been proposed. One of the improved methods is the Stochastic Gradient Descent (SGD) method for big data learning [1].

Recently, several training algorithms have been proposed considering biological plausibility [3]–[6]. These researches began with questions such as, "Does the biological brain perform complex and accurate backward calculations like BP?" [3] [6] or "Is it necessary to calculate the exact gradient used in training?" [4] [5]. Furthermore, a gradient-based training algorithm that does not require the backpropagation of errors has been proposed [7]–[9]. This algorithm can potentially reduce computational and hardware burdens even when processed by microcontrollers installed in IoT devices. However, these algorithms are unsuitable for asynchronous parallel learning or training large datasets and still need improvement. In addition, there are preliminary simulation experiments of the proposed algorithms, and the learning accuracy for recent nonlinear data is not guaranteed.

In this paper, focusing on the algorithm of [7]–[9], a novel algorithm is proposed to simplify the calculation of the gradient. The proposed method is referred to as Stochastic Weight Difference Propagation (SWDP) to improve the disadvantages of SGD. SWDP differs from the conventional SGD in that the weights can be updated by only the inner product of the post-synaptic weights and their differences. This enables parallel training of each neuron in each layer. Therefore, the backward process for each training sample in mini-batch (stochastic) training is unnecessary, and the hardware and computational cost can be reduced. This means that the proposed method can reduce the computational complexity in the backward calculations of the training to $1/b$ ($b$ is the mini-batch size) compared to BP. The proposed method, compared with the conventional SGD method on two benchmark problems and simulations, demonstrates its effectiveness.

The paper is organized as follows: Section 2 describes the training of SGD. In Section 3, the proposed SWDP is derived and explained. In Section 4, the computational cost of SGD and SWDP is discussed. Sections 5 and 6 present computational simulations and conclusions, respectively.

## II. Stochastic Gradient Descent method

This paper considers the multi-layer feed-forward NN training, which is an unconstrained optimization problem to minimize the error function $E(\mathbf{w})$ concerning the weight vector $\mathbf{w} \in \mathbb{R}^n$.

$$\min_{\mathbf{w}\in\mathbb{R}^n} \quad E(\mathbf{w}). \qquad (1)$$

There are two training methods for error Back Propagation (BP): batch and stochastic (mini-batch) strategies. All training samples in a dataset $T_r$ are used in an epoch to calculate

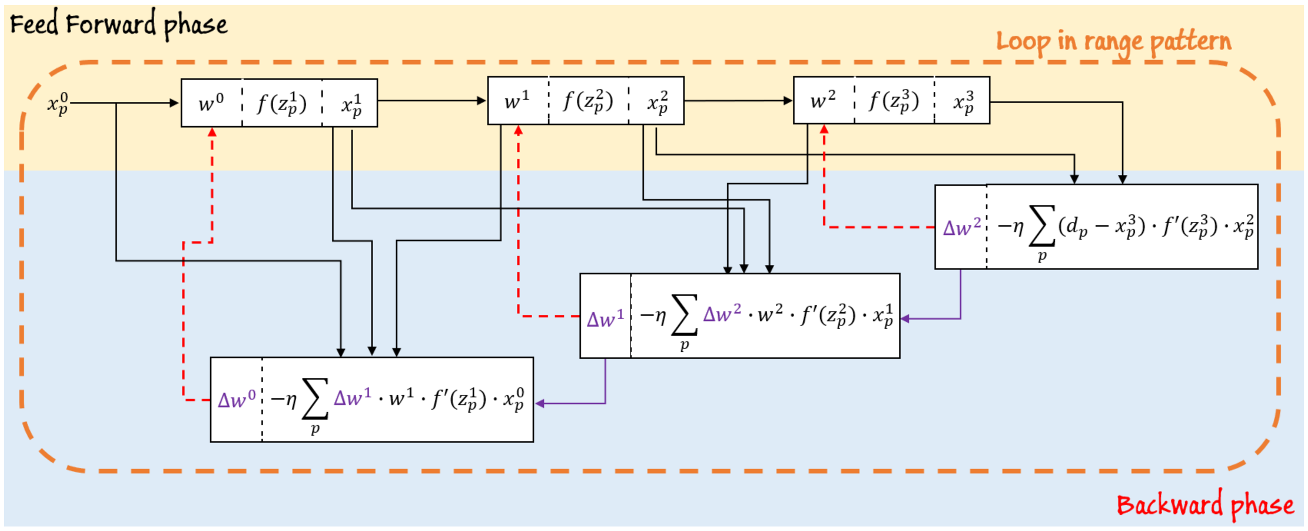
Feed Forward phase
Loop in range pattern
$x_p^0$
$w^0$ | $f(z_p^1)$ | $x_p^1$
$w^1$ | $f(z_p^2)$ | $x_p^2$
$w^2$ | $f(z_p^3)$ | $x_p^3$
$\Delta w^2$ | $-\eta \sum_p (d_p - x_p^3) \cdot f'(z_p^3) \cdot x_p^2$
$\Delta w^1$ | $-\eta \sum_p \Delta w^2 \cdot w^2 \cdot f'(z_p^2) \cdot x_p^1$
$\Delta w^0$ | $-\eta \sum_p \Delta w^1 \cdot w^1 \cdot f'(z_p^1) \cdot x_p^0$
Backward phase

Fig. 1. Structural diagram of SGD.

gradients in the batch strategy. The error function $E(\mathbf{w})$ is defined as

$$E(\mathbf{w}) = \frac{1}{|T_r|} \sum_{p \in T_r} E_p(\mathbf{w}), \tag{2}$$

where, $|T_r|$ denotes the number of samples, and $E_p(\mathbf{w})$ is the error of the $p^{\text{th}}$ sample. On the other hand, the stochastic training strategy in BP, so-call Stochastic Gradient Descent (SGD), uses $X \subseteq T_r$ dataset randomly selected from $T_r$ in an iteration to calculate the gradient. The error function $E_b(\mathbf{w})$ of SGD is defined as

$$E_b(\mathbf{w}) = \frac{1}{b} \sum_{p \in X} E_p(\mathbf{w}), \tag{3}$$

where $b = |X|$ is the mini-batch size and if $X = T_r$ and $b = |T_r|$ then mini-batch training shifts to the batch mode.

Let $\mathbf{x}_p$ and $\mathbf{o}_p$ be the $p^{\text{th}}$ input and output vectors, respectively. The relation between the inputs and outputs of the NN is defined as

$$\mathbf{o}_p = \mathbf{x}_p^{\text{out}} = f_{NN}(\mathbf{w}, \mathbf{x}_p). \tag{4}$$

Moreover, let $x_{i,p}^s (1 \leq s \leq out)$ be the output of the $i^{\text{th}}$ neuron in the $s$ layer for the $p^{\text{th}}$ sample, and $w_{ij}^s$ be the weight from the $j^{\text{th}}$ neuron of the $s-1$ layer to the $i^{\text{th}}$ neuron of the $s$ layer, the input-output relationship of the neuron is given by (5) and (6). Note that $s = out$ denotes the output layer.

$$x_{i,p}^s = f(z_{i,p}^s), \tag{5}$$

$$z_{i,p}^s = \sum_j w_{ij}^s \cdot x_{j,p}^{s-1}, \tag{6}$$

where $f(z_{i,p}^s)$ and $w_{ij}^s$ denote the activation function and a component of the weight vector $\mathbf{w}$, respectively. The update formula of SGD is defined as (7) using learning rate $\eta$.

$$w_{ij}^s(t+1) = w_{ij}^s(t) - \eta \frac{\partial E_b(\mathbf{w})}{\partial w_{ij}^s}. \tag{7}$$

where $t$ denotes the iteration number and

$$\frac{\partial E_b(\mathbf{w})}{\partial w_{ij}^s} = \frac{1}{b} \sum_{p \in X} \frac{\partial E_p(\mathbf{w})}{\partial w_{ij}^s}, \tag{8}$$

is the stochastic gradient, which is expansion by chain rule as (9).

$$\begin{aligned} \frac{\partial E_b(\mathbf{w})}{\partial w_{ij}^s} &= \frac{1}{b} \sum_{p \in X} \frac{\partial E_p(\mathbf{w})}{\partial x_{i,p}^s} \cdot \frac{\partial x_{i,p}^s}{\partial z_{i,p}^s} \cdot \frac{\partial z_{i,p}^s}{\partial w_{ij}^s} \\ &= \frac{1}{b} \sum_{p \in X} \frac{\partial E_p(\mathbf{w})}{\partial x_{i,p}^s} \cdot f'(z_{i,p}^s) \cdot x_{j,p}^{s-1}, \end{aligned} \tag{9}$$

where $f'(z_{i,p}^s)$ is the derivative of the activation function and the partial differentiation of $\partial E_b(\mathbf{w})/\partial w_{ij}^s$ varies depending on whether the $s$ is the output ($s = out$) or the hidden layers.

- *$s$ is the output layer ($s = out$)*:

If $s$ is the output layer, the partial differentiation of $\partial E_b(\mathbf{w})/\partial w_{ij}^s$ is directly derived from the error function. Here, the two types of error functions, that is, the mean squared error (MSE) and the cross entropy (CE) are considered.

<MSE >

The error function is defined as

$$E_b(\mathbf{w}) = \frac{1}{b} \sum_{p \in X} \sum_{i=1}^{L} \frac{1}{2} \left( d_{i,p} - x_{i,p}^{out} \right)^2, \tag{10}$$

where $L$ is the number of the output units and $d_{i,p}$ denote $i^{\text{th}}$ unit of the output layer for the $p^{\text{th}}$ desired vector. Therefore, the partial differentiation of $\partial E_b(\mathbf{w})/\partial w_{ij}^s$ is given by

$$\begin{aligned} \frac{\partial E_b(\mathbf{w})}{\partial w_{ij}^s} &= \frac{1}{b} \sum_{p \in X} \frac{\partial E_p(\mathbf{w})}{\partial x_{i,p}^s} \cdot f'(z_{i,p}^s) \cdot x_{j,p}^{s-1} \\ &= -\frac{1}{b} \sum_{p \in X} (d_{i,p} - x_{i,p}^{out}) \cdot f'(z_{i,p}^s) \cdot x_{j,p}^{s-1}. \end{aligned} \tag{11}$$

<CE >

The error function and the activation function are defined as (12) and the softmax function of (13), respectively.

$$E_b(\mathbf{w}) = -\frac{1}{b}\sum_{p\in X}\sum_{i=1}^{L} d_{i,p}\log(x^{out}_{i,p}), \tag{12}$$

$$x^{out}_{i,p} = f(z^{s}_{i,p}) = \frac{exp^{(z^s_{i,p})}}{\sum_{i=1}^{L} exp(z^s_{i,p})}. \tag{13}$$

where $L$ is the classification class. The gradient is given by

$$\begin{aligned}\frac{\partial E_b(\mathbf{w})}{\partial w^s_{ij}} &= \frac{1}{b}\sum_{p\in X}\frac{\partial E_p(\mathbf{w})}{\partial x^s_{i,p}}\cdot f'(z^s_{i,p})\cdot x^{s-1}_{j,p} \\ &= -\frac{1}{b}\sum_{p\in X}(d_{i,p} - x^{out}_{i,p})\cdot x^{s-1}_{j,p}.\end{aligned} \tag{14}$$

- *s is a hidden layer*:

If $s$ is the hidden layer, the partial differentiation of $\partial E_p(\mathbf{w})/\partial x^s_{i,p}$ in (9) is given by

$$\frac{\partial E_p(\mathbf{w})}{\partial x^{s}_{i,p}} = \sum_k \frac{\partial E_p(\mathbf{w})}{\partial x^{s+1}_{k,p}}\cdot\frac{\partial x^{s+1}_{k,p}}{\partial z^{s+1}_{k,p}}\cdot\frac{\partial z^{s+1}_{k,p}}{\partial x^s_{i,p}} \tag{15}$$

$$= \sum_k \frac{\partial E_p(\mathbf{w})}{\partial x^{s+1}_{k,p}}\cdot f'(z^{s+1}_{k,p})\cdot w^{s+1}_{ki}. \tag{16}$$

Substituting (16) into (9), the gradient of $\partial E_b(\mathbf{w})/\partial w^s_{ij}$ can be obtained as (17).

$$\frac{\partial E_b(\mathbf{w})}{\partial w^s_{ij}} = \frac{1}{b}\sum_{p\in X}\sum_k \frac{\partial E_p(\mathbf{w})}{\partial x^{s+1}_{k,p}}\cdot f'(z^{s+1}_{k,p})\cdot w^{s+1}_{ki}\cdot f'(z^s_{i,p})\cdot x^{s-1}_{j,p}. \tag{17}$$

(17) shows that the inner product of the back propagation components $\partial E_p(\mathbf{w})/\partial x^{s+1}_{k,p}\cdot f'(z^{s+1}_{k,p})$ and the weights $w^{s+1}_{ki}$ is necessary for the all samples in a mini-batch $b$. This means that the derivation of the gradient requires $b$ times backward processes. The structural diagram of SGD is shown in Fig. 1. This figure also shows that SGD requires an error propagation architecture from the output layer to the input layer for each training sample in order to calculate the gradient.

## III. Stochastic Weight Difference Propagation Learning

In this paper, to reduce the computational cost of training in SGD, Stochastic Weight Difference Propagation (SWDP) is proposed. The proposed SWDP focuses on the backpropagation stream of each training sample in SGD training and eliminates the BP architecture requirement for each training sample in a mini-batch. As a result, SWDP can calculate the amounts of weight updates with only one weight difference propagation. This means that the proposed method can reduce the computational complexity in the backward calculations of the training to $1/b$ compared to SGD.

Since the training of SWDP is based on SGD, and the computation process differs depending on whether $s$ is the output or hidden layers.

- *s is the output layer (s = out)*:

If $s$ is the output layer, $\partial E_b(\mathbf{w})/\partial w^s_{ij}$ is same as SGD given by (11) or (14).

- *s is the hidden layer*:

If $s$ is the hidden layer, a variant of (17) is considered. In the proposed SWDP, the gradient of $\partial E_b(\mathbf{w})/\partial w^s_{ij}$ given by (17) is transformed as follows:

$$\frac{\partial E_b(\mathbf{w})}{\partial w^s_{ij}} = \frac{1}{b}\sum_{p\in X}\sum_k \frac{\partial E_p(\mathbf{w})}{\partial x^{s+1}_{k,p}}\cdot f'(z^{s+1}_{k,p})\cdot w^{s+1}_{ki}\cdot f'(z^s_{i,p})\cdot x^{s-1}_{j,p} \tag{17}$$

$$= \sum_k w^{s+1}_{ki}\cdot\frac{1}{b}\sum_{p\in X}\frac{\partial E_p(\mathbf{w})}{\partial x^{s+1}_{k,p}}\cdot f'(z^{s+1}_{k,p})\cdot f'(z^s_{i,p})\cdot x^{s-1}_{j,p}. \tag{18}$$

Here, (18) is transformed to (19) by assuming that the output of $i$th neuron in $s$ layer $x^s_{i,p}\neq 0$.

$$\begin{aligned}\frac{\partial E_b(\mathbf{w})}{\partial w^s_{ij}} = \sum_k w^{s+1}_{ki}\cdot\frac{1}{b}\sum_{p\in X}\frac{\partial E_p(\mathbf{w})}{\partial x^{s+1}_{k,p}}\cdot f'(z^{s+1}_{k,p})\cdot x^s_{i,p} \\ \cdot\frac{f'(z^s_{i,p})}{x^s_{i,p}}\cdot x^{s-1}_{j,p}.\end{aligned} \tag{19}$$

From (9),

$$\frac{\partial E_p(\mathbf{w})}{\partial w^{s+1}_{ki}} = \frac{\partial E_p(\mathbf{w})}{\partial x^{s+1}_{k,p}}\cdot f'(z^{s+1}_{k,p})\cdot x^s_{i,p}. \tag{20}$$

Therefore, (21) can obtained by substituting (20) into (19).

$$\frac{\partial E_b(\mathbf{w})}{\partial w^s_{ij}} = \sum_k w^{s+1}_{ki}\cdot\frac{1}{b}\sum_{p\in X}\frac{\partial E_p(\mathbf{w})}{\partial w^{s+1}_{ki}}\cdot\frac{f'(z^s_{i,p})}{x^s_{i,p}}\cdot x^{s-1}_{j,p}. \tag{21}$$

Here, (21) is transformed as follows: The second sum in (21) for the sample $p$ is divided into two parts, that is, $\partial E_p(\mathbf{w})/\partial w^s_{ij}$ and $f'(z^s_{i,p})/x^s_{i,p}\cdot x^{s-1}_{j,p}$. Then (21) is rewritten as (22),

$$\begin{aligned}\frac{\partial E_b(\mathbf{w})}{\partial w^s_{ij}} = \sum_k w^{s+1}_{ki}\Bigg\{&\frac{1}{b}\sum_{p\in X}\frac{\partial E_p(\mathbf{w})}{\partial w^{s+1}_{ki}}\cdot\frac{1}{b}\sum_{p\in X}\frac{f'(z^s_{i,p})}{x^s_{i,p}}\cdot x^{s-1}_{j,p} \\ &+\frac{1}{b}\sum_{p\in X}\Bigg(\left(\frac{\partial E_p(\mathbf{w})}{\partial w^{s+1}_{ki}} - \frac{1}{b}\sum_{p\in X}\frac{\partial E_p(\mathbf{w})}{\partial w^{s+1}_{ki}}\right) \\ &\cdot\left(\frac{f'(z^s_{i,p})}{x^s_{i,p}}\cdot x^{s-1}_{j,p} - \frac{1}{b}\sum_{p\in X}\frac{f'(z^s_{i,p})}{x^s_{i,p}}\cdot x^{s-1}_{j,p}\right)\Bigg)\Bigg\}.\end{aligned} \tag{22}$$

That is, the first term denotes the product of the averages of $\partial E_p(\mathbf{w})/\partial w^{s+1}_{ki}$ and $f'(z^s_{i,p})/x^s_{i,p}\cdot x^{s-1}_{j,p}$. The second term is the covariance of these parts. In (22), if $\partial E_p(\mathbf{w})/\partial w^{s+1}_{ki}$ and $f'(z^s_{i,p})/x^s_{i,p}\cdot x^{s-1}_{j,p}$ are independent variables, (23) holds.

$$\begin{aligned}&\left(\frac{\partial E_p(\mathbf{w})}{\partial w^{s+1}_{ki}} - \frac{1}{b}\sum_{p\in X}\frac{\partial E_p(\mathbf{w})}{\partial w^{s+1}_{ki}}\right) \\ &\cdot\left(\frac{f'(z^s_{i,p})}{x^s_{i,p}}\cdot x^{s-1}_{j,p} - \frac{1}{b}\sum_{p\in X}\frac{f'(z^s_{i,p})}{x^s_{i,p}}\cdot x^{s-1}_{j,p}\right) = 0\end{aligned} \tag{23}$$

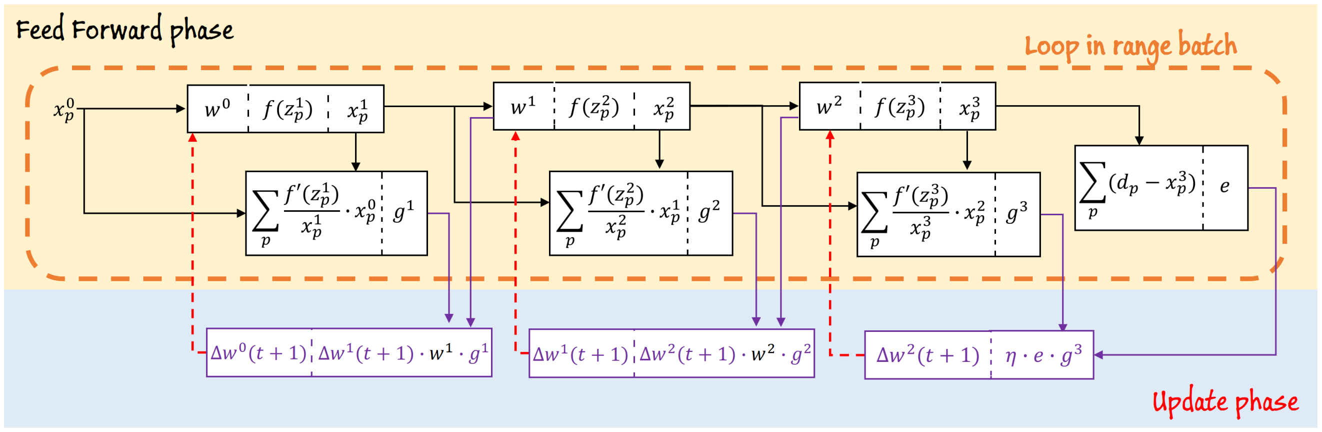
Feed Forward phase
Loop in range batch
$x_p^0$
$w^0$ $f(z_p^1)$ $x_p^1$
$w^1$ $f(z_p^2)$ $x_p^2$
$w^2$ $f(z_p^3)$ $x_p^3$
$\sum_p \frac{f'(z_p^1)}{x_p^1} \cdot x_p^0$ $g^1$
$\sum_p \frac{f'(z_p^2)}{x_p^2} \cdot x_p^1$ $g^2$
$\sum_p \frac{f'(z_p^3)}{x_p^3} \cdot x_p^2$ $g^3$
$\sum_p (d_p - x_p^3)$ $e$
$\Delta w^0(t+1)$ $\Delta w^1(t+1) \cdot w^1 \cdot g^1$
$\Delta w^1(t+1)$ $\Delta w^2(t+1) \cdot w^2 \cdot g^2$
$\Delta w^2(t+1)$ $\eta \cdot e \cdot g^3$
Update phase

Fig. 2. Structural diagram of SWDP.

However, these terms are clearly not independent because these parts are derived from the sample $p$. It is also estimated that as the number of samples in the mini-batch increases, the value of (23) becomes larger. On the other hand, it is clear that learning of NN can be performed even if there is some error in gradients [3]–[6]. Therefore, the proposed SWDP assumes that (23) holds and then obtains (24).

$$\frac{\partial E_b(\mathbf{w})}{\partial w_{ij}^s} \simeq \sum_k w_{ki}^{s+1} \left( \frac{1}{b} \sum_{p \in X} \frac{\partial E_p(\mathbf{w})}{\partial w_{ki}^{s+1}} \cdot \frac{1}{b} \sum_{p \in X} \frac{f'(z_{i,p}^s)}{x_{i,p}^s} \cdot x_{j,p}^{s-1} \right)$$
$$= \sum_k \left( \frac{\partial E_b(\mathbf{w})}{\partial w_{ki}^{s+1}} \cdot w_{ki}^{s+1} \right) \cdot \left( \frac{1}{b} \sum_{p \in X} \frac{f'(z_{i,p}^s)}{x_{i,p}^s} \cdot x_{j,p}^{s-1} \right). \tag{24}$$

In this study, the gradient $\partial E_b(\mathbf{w}) / \partial w_{ij}^s$ of (24) is used for learning. Consider the update formula for $w_{ki}^{s+1}$ in SGD of (25).

$$w_{ki}^s(t+1) = w_{ki}^s(t) - \eta \frac{\partial E_b(\mathbf{w})}{\partial w_{ki}^s}. \tag{25}$$

Here, the update formula of (25) can be transformed into an expression for the amount of update (difference) $\Delta w_{ki}^{s+1}$ as

$$\Delta w_{ki}^{s+1} = w_{ki}^{s+1}(t+1) - w_{ki}^{s+1}(t) = -\eta \frac{\partial E_b(\mathbf{w})}{\partial w_{ki}^{s+1}}, \tag{26}$$

and then

$$\frac{\partial E_b(\mathbf{w})}{\partial w_{ki}^{s+1}} = -\frac{1}{\eta} \Delta w_{ki}^{s+1}. \tag{27}$$

By substituting (27) into (24),

$$\frac{\partial E_b(\mathbf{w})}{\partial w_{ij}^s} = \sum_k \left( -\frac{1}{\eta} \Delta w_{ki}^{s+1} \cdot w_{ki}^{s+1} \right) \cdot \left( \frac{1}{b} \sum_{p \in X} \frac{f'(z_{i,p}^s)}{x_{i,p}^s} \cdot x_{j,p}^{s-1} \right). \tag{28}$$

Furthermore, by substituting (28) into the SGD update formula (7), the proposed SWDP update formula can be obtained as (29).

$$w_{ij}^s(t+1) = w_{ij}^s(t) - \eta \frac{\partial E_b(\mathbf{w})}{\partial w_{ij}^s}$$
$$= w_{ij}^s(t) + \sum_k \left( \Delta w_{ki}^{s+1} \cdot w_{ki}^{s+1} \right) \cdot \left( \frac{1}{b} \sum_{p \in X} \frac{f'(z_{i,p}^s)}{x_{i,p}^s} \cdot x_{j,p}^{s-1} \right). \tag{29}$$

In (29), $\left(1/b \sum_p f'(z_{i,p}^s)/x_{i,p}^s \cdot x_{j,p}^{s-1}\right)$ can be calculated in the forward operation in the mini-batch. Therefore, the backward operation for the learning of NN is one calculation of $\sum_k \left(\Delta w_{ki}^{s+1} \cdot w_{ki}^{s+1}\right)$ in the stochastic learning. This means that the learning architecture is simple, and the computational cost decreases compared to SGD. However, it is estimated that the error of the gradients affects the learning from the assumption of (23) holds. In Section V, the effect on learning is investigated through computer experiments. The structural diagram of SWDP is shown in Fig. 2. This figure shows that SWDP eliminates the requirement of the BP architecture for each training sample in mini-batch and calculates the amounts of the weight update with only one weight difference propagation.

## IV. Computational Cost

This section discusses the computational costs of the proposed SWDP and SGD for updating a weight of $w_{ij}^s$. Since the forward calculations of SWDP and SGD are the same, the calculation costs of the backward processes for both SWDP and SGD are compared. The summary of the backward computational cost is illustrated in TABLE I. To estimate the costs of backward processes, (17) and (28) are considered for SGD and SWDP, respectively. In SGD, the computational cost is $b(2k + 2)$, which denotes the inner product of the propagated error components and weights in the $s + 1$ layer for all samples in the mini-batch $b$. In SWDP, the cost $k$ indicates the inner product of weights and their differences in the $s + 1$ layer. Therefore, it can be seen from TABLE

I that the proposed SWDP can reduce the computational complexity in the backward calculations of the training to $1/b$ compared to SGD. Here, the computational cost of $\left(1/b \sum_{p\in X} f'(z^s_{i,p})/x^s_{i,p} \cdot x^{s-1}_{j,p}\right)$ in (28) is $2b+1$, but this term can be calculated in forward process. Therefore, this cost is ignored in the backward process of SWDP.

TABLE I
SUMMARY OF THE COMPUTATIONAL COST.

| Algorithm | Computational Cost of Backward |
|---|---|
| SGD | b(2k + 2) + 1 |
| SWDP | k + 1 |

## V. Simulation Results

To investigate the performance of the proposed method, SWDP is compared with SGD on two classification benchmark problems. For all problems, the learning rate is set to $\eta = 0.1$ for both algorithms. The mini-batch sizes are set to $b = 1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20$, and 32 for the problems. The termination condition were set to $E(\mathbf{w}) < 10^{-3}$ and the maximum iteration counts $t_{max} = 500,000$.

### A. $8 \times 8$ MNIST

The first problem is the classification problem of MNIST handwritten digits dataset [10] with $8 \times 8$ pixels as shown in Fig. 3. The $8{\times}8$ MNIST problem classifies a handwritten digit image of $|T| = 1,797$ training data samples into ten classes from 0 to 9. In this experiment, the dataset was randomly divided to 75% (1,347) as the training data $T_r$ and 25% (450) as the test data $T_s$. The network structure is 64-10-10, which denotes inputs, the number of neurons in a hidden layer, and the outputs. In this problem, the activation functions for the hidden and the output layers were set to the sigmoid and the softmax of (13) functions, respectively. The error function is cross-entropy of (12). The experimental results are shown in Fig. 4 and TABLE II. In the first, Fig. 4 shows the training and test accuracies for each mini-batch size, where the $x$-axis is the mini-batch size, and the $y$-axis is the accuracy. This figure shows that the training accuracies of SGD and SWDP are almost the same. In terms of test accuracy, both algorithms maintain almost the same accuracy too. That is, SWDP was

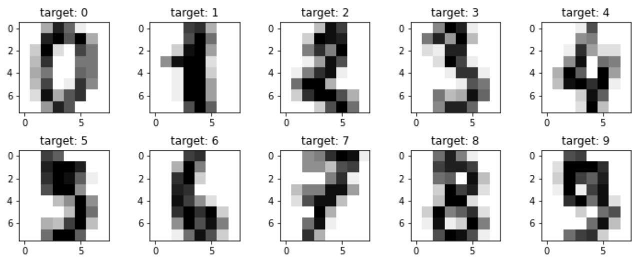

Fig. 3. Examples of $8 \times 8$ MNIST handwritten digits dataset.

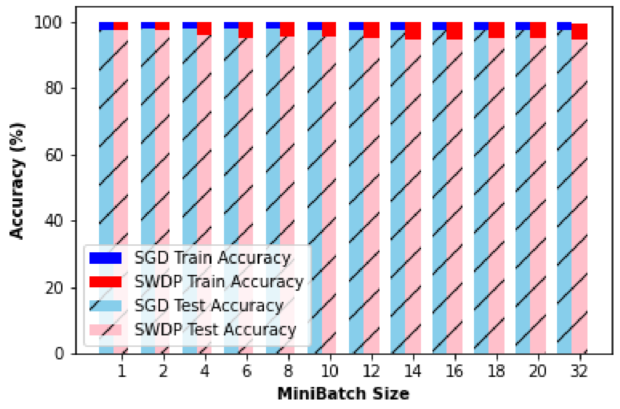

Fig. 4. Accuracy .vs. mini-batches for $8 \times 8$ MNIST.

up to 3% less accurate than SGD. Therefore, it concluded that training with almost the same accuracy is possible even if simplified backward calculation. The table summarizes the epoch, iteration, time, and time per iteration required for the convergence of SGD and SWDP. The table shows that SWDP requires more epoch, iteration, and time as the mini-batch size increases. It can be attributed to (23) becoming stronger as the mini-batch size increases. However, the computation time per iteration of the proposed SWDP is shorter than SGD because the backward computation is simplified. Therefore, it can be concluded that the proposed SWDP method is practical for this problem.

### B. 3-Spiral

The next problem is another nonlinear problem called the 3-Spiral problem [11]. 3-Spiral is a problem of classifying training data samples $T_r = 1,050$ into three classes, as shown in Fig. 5, where each class has 350 data samples. The input data is the coordinates of each point, and the NN structure is 2-10-3, which denotes inputs, hidden layer neurons, and outputs numbers, respectively. The activation and error functions were the same as the previous problem. The experimental results are shown in Fig. 6. Fig. 6 shows the training accuracy for each mini-batch size, where the $x$-axis is the mini-batch size, and the $y$-axis is the training accuracy. This figure shows that the proposed method and SGD have similar accuracy when the mini-batch size $b = 1$ and 2. However, as the mini-batch size increases, the accuracy of the proposed SWDP method decreases from 3% to a maximum of 22% ($b = 10$). Therefore, in training this problem, the training of SWDP strongly depends on the mini-batch size and the initial values.

## VI. Conclusion

In this research, a novel training algorithm was proposed. The proposed method was referred to as Stochastic Weight Difference Propagation (SWDP) to simplify the gradient calculation (backward processes) in SGD. SGD, based on the error propagation architecture from the output to the input layers, is required for each training sample to calculate the gradient and

TABLE II
THE RESULTS SUMMARY OF $8 \times 8$ MNIST.

| Algorithm | Mini-batch Size | Epoch | Iteration | Time (*sec*) | per Iteration Time(*msec*) |
|---|---|---|---|---|---|
| SGD | 1 | 28 | 39,062 | 0.364 | $0.9318 \times 10^{-2}$ |
| | 2 | 26 | 18,170 | 0.32 | $0.1761 \times 10^{-1}$ |
| | 4 | 26 | 9,071 | 0.284 | $0.3130 \times 10^{-1}$ |
| | 6 | 26 | 6,047 | 0.284 | $0.4696 \times 10^{-1}$ |
| | 8 | 26 | 4,535 | 0.268 | $0.5909 \times 10^{-1}$ |
| | 10 | 27 | 3,751 | 0.276 | $0.7358 \times 10^{-1}$ |
| | 12 | 26 | 3,023 | 0.262 | $0.8666 \times 10^{-1}$ |
| | 14 | 26 | 2,591 | 0.277 | 0.1069 |
| | 16 | 26 | 2,267 | 0.268 | 0.1182 |
| | 18 | 26 | 1,997 | 0.27 | 0.1352 |
| | 20 | 26 | 1,808 | 0.263 | 0.1454 |
| | 32 | 27 | 1,175 | 0.274 | 0.2331 |
| SWDP | 1 | 28 | 39,062 | 0.366 | $0.9369 \times 10^{-2}$ |
| | 2 | 92 | 62,588 | 0.947 | $0.1513 \times 10^{-1}$ |
| | 4 | 192 | 64,847 | 1.826 | $0.2815 \times 10^{-1}$ |
| | 6 | 625 | 140,223 | 5.614 | $0.4003 \times 10^{-1}$ |
| | 8 | 860 | 144,647 | 7.691 | $0.5317 \times 10^{-1}$ |
| | 10 | 1,233 | 165,355 | 10.943 | $0.6617 \times 10^{-1}$ |
| | 12 | 2,254 | 252,559 | 19.474 | $0.7710 \times 10^{-1}$ |
| | 14 | 3,312 | 318,047 | 28.558 | $0.8979 \times 10^{-1}$ |
| | 16 | 4,922 | 413,499 | 41.972 | 0.1015 |
| | 18 | 5,006 | 370,517 | 42.398 | 0.1144 |
| | 20 | 5,898 | 395,232 | 50.48 | 0.1277 |
| | 32 | 11,905 | 500,000 | 99.671 | 0.1993 |

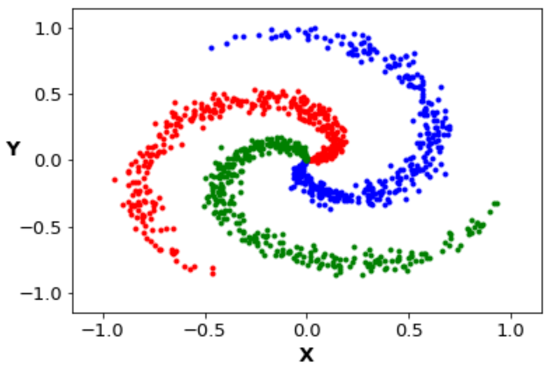

Fig. 5. Layout of the 3-Spirals dataset.

causes an increase in hardware and computational costs. The proposed SWDP was focused on the disadvantage of SGD and reduced the computational complexity in the backward calculations of the training compared to SGD. The proposed SWDP simplifies the backward process of SGD using only the inner product of the weights and their differences in training. The computational costs of the backward processes during SWDP training are reduced to $1/b$ ($b$ is the mini-batch size) compared to SGD. Experimental results showed that SWDP could learn accurately close to SGD even when the backward processes were simplified. However, the number of epochs required for learning SWDP increased as the mini-batch size increased. This is caused by the fact that SWDP ignored the covariance term required for the SGD backward processes.

In future works, the proposed method SWDP will be improved to achieve more robust learning similar to SGD regardless of the mini-batch size, initial values, and nonlinearity of the problem. In addition to investigating the effectiveness of the proposed method on edge computing, we plan to implement SWDP on hardware such as FPGA to verify its effectiveness.

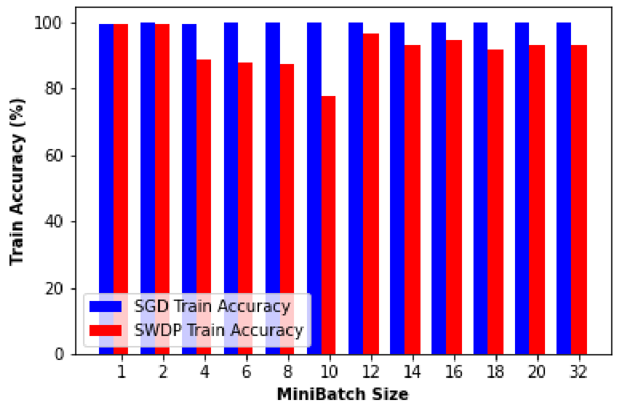

Fig. 6. Training accuracy for mini-batches for 3-Spirals.

## ACKNOWLEDGMENT

The authors gratefully acknowledge the Honjo International Scholarship Foundation for supporting this work. The Japan Society supported this work for the Promotion of Science (JSPS), KAKENHI (20K11979).

## REFERENCES

[1] I. Goodfellow, Y. Bengio and A. Courville, "*Deep Learning*", MIT Press, 2016.
[2] S. Haykin: "*Neural Networks and Learning Machines*", Pearson, 2009.
[3] D. H. Lee, S. Zhang, A. Fischer and Y. Bengio, "Difference target propagation," *Proc. ECML/PKDD*, Springer, Cham, pp. 498–515, 2015.
[4] T. P. Lillicrap, D. Cownden, D. B. Tweed and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Communications*, 2016.
[5] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," *Proc. NeurIPS*, vol. 29, 2016.
[6] T. Miyato, D. Okanohara, S. I. Maeda and M. Koyama, "Synthetic gradient methods with virtual forward-backward networks," *Proc. Workshop trac - ICLR*, 2017.
[7] R. D. Brandt and F. Lin, "Supervised Learning in neural networks without explicit error backpropagation," *Proc. Annual Allerton Conference on Communication Control and Computing*, pp. 294–303, 1994.
[8] R. D. Brandt and F. Lin, "Can supervised learning be achieved without explicit error backpropagation?", *Proc. IEEE ICNN*, pp. 300–305, 1996.
[9] H. Ninomiya and N. Kinoshita, "A new learning algorithm without explicit error backpropagation," *Proc. IJCNN'99*, vol.2, No.99CH36339, pp. 1389–1392, 1999.
[10] E. Alpaydin and C. Kaynak, "Optical recognition of handwritten digits data set", *UCI Machine Learning Repository*, 1998.
[11] J. Bassey, X. Li and L. Qian, "An Experimental Study of Multi-Layer Multi-Valued Neural Network", *Proc. IEEE ICDIS*, pp. 233–236, 2019.