Weighted Aggregating Stochastic Gradient Descent for Parallel Deep Learning

Pengzhan Guo, Zeyang Ye, Keli Xiao, Senior Member, IEEE, and Wei Zhu

Abstract—This paper investigates the stochastic optimization problem focusing on developing scalable parallel algorithms for deep learning tasks. Our solution involves a reformation of the objective function for stochastic optimization in neural network models, along with a novel parallel computing strategy, coined the weighted aggregating stochastic gradient descent (WASGD). Following a theoretical analysis on the characteristics of the new objective function, WASGD introduces a decentralized weighted aggregating scheme based on the performance of local workers. Without any center variable, the new method automatically gauges the importance of local workers and accepts them by their contributions. Furthermore, we have developed an enhanced version of the method, WASGD+, by (1) implementing a designed sample order and (2) upgrading the weight evaluation function. To validate the new method, we benchmark our pipeline against several popular algorithms including the state-of-the-art deep neural network classifier training techniques (e.g., elastic averaging SGD). Comprehensive validation studies have been conducted on four classic datasets: CIFAR-100, CIFAR-10, Fashion-MNIST, and MNIST. Subsequent results have firmly validated the superiority of the WASGD scheme in accelerating the training of deep architecture. Better still, the enhanced version, WASGD+, is shown to be a significant improvement over its prototype.

Index Terms—Stochastic optimization, stochastic gradient descent, parallel computing, deep learning, neural network

1 INTRODUCTION

Deep learning, largely based on multi-layer neural networks resembling the way human brain functions, has proven to be superior at pattern recognition tasks [1] based on big and unstructured data such as those in voice and face recognitions. With the rapid growth in data size and problem complexity, deep learning is increasingly touted as the mathematics engine of artificial intelligence. The key to its optimal real time deployment lies in the design of efficient parallel computing algorithms. Stochastic gradient descent (SGD) proposed by Robbins and Monro [2], by replacing the gradient obtained from using the whole dataset to a mere random subset, is an efficient optimization technique for big data problems. Parallel SGD aims to achieve further speedup of the SGD optimization process while maintaining its good learning performance.

There are three main challenges in parallel SGD. First, for application purpose, people need more efficient parallel algorithm for faster real time deployment in big data. Second, the communication burden is still very heavy (e.g., wasting time waiting for straggling workers). Lastly, the proposed parallel method should be stable in most cases to ensure convergence. Zinkevich et al. [3] proposed the first parallel SGD method called SimuParallel SGD, a synchronous algorithm that averages all workers’ weights in the training process. To reduce the waiting time of local workers, Chen et al. [4] added backup workers in their method, and partially solved the idle time problem in the synchronous algorithm. Moreover, Recht et al. [5] opened up the access to global parameters by allowing workers to update the global parameters as soon as possible. Collectively, these recent methods feature substantial progress but still need further improvement. In terms of solution quality, the methods proposed in [4] directly ignore some workers’ results and thus the solution quality depreciates. Regarding communication time, the method proposed in [3] still falls short due to its computation time growing linearly with the data size. In terms of applicable fields, methods from [3] and [5] both need special requirements (e.g., convex functions, and uniform memory access) that are not always tenable.

To address these issues, we have proposed a novel distributed parallel method coined the Weighted Aggregating SGD (WASGD) in a preliminary work to this paper [6]. Our method enables local workers to accept the aggregated parameters of all workers with evaluation weights. Our main idea is as follows. In the parallel system, we first let each local worker update its local parameters. Then, the communications among local workers are based on the local performance on loss. Finally, the communicated update of local parameters is related to a weighted combination of the parameters of all local workers taken in time and space.

To sum up, the main contributions of [6] are four-fold. First, by assigning weights to different workers based on their performances, WASGD, a new parallel SGD method, is proposed. A series of loss estimation techniques to balance the performance evaluation accuracy and computation efficiency have also been developed. Second, the convergence
of WASGD is theoretically proven. Third, the estimation method in WASGD incurs no extra computing time, which contributes to the better speedup of our method. Lastly, WASGD is applied to the convolutional neural networks (CNN). The results have consistently confirmed the superiority of WASGD, demonstrating its effective performance in different applications.

However, there are still some outstanding issues with the basic WASGD method that need further considerations. As suggested by [7], different sample orders may lead to different impacts on the final result. The basic WASGD in [6] goes through the samples in a completely random order and thus fails to consider the effect of sample order. Also, its original weight function is not equally effective in handling different weighting strategies. Moreover, our previous work lacks theoretical analysis on the model variance of the weighted cases plus sufficient discussions on some important parameters, such as the parameter (β) that controls the changing percentage of local workers. Here, we focus on addressing these critical issues. This paper differs substantially from [6] and contributes to the literature further in four ways:

- We develop an enhanced version of WASGD, namely WASGD+, by considering the sample order. The enhanced method also introduces a more flexible weight evaluation method, that is capable of computing weights effectively under different strategies.

- By formulating a general form of the parallel optimization problem for deep learning, we have enabled theoretically the stability of our method. We have also discovered the essential connection between parallel SGD and mini-batch by changing the communication period, which in turn guides us in regulating the performance of parallel SGD more efficiently.

- We perform comprehensive parameter analysis based on the new method. We analyze the impact of the parameter (β) numerically and discover that the full acceptance of the communication result is not always the optimal choice. We also compare the performances under different weighting strategies to identify conditions when weighted case is better than the equal-weight case.

- To demonstrate the consistency and practical value of the new method, besides the original experiments, we have also tested on a more complex dataset, CIFAR-100. Many additional numerical studies have also been conducted for the selection of the weighting strategy and the value of β. All results support the superiority of WASGD+ and its stability in handling different deep learning applications.

The rest of this paper is organized as follows. In Section 2, we introduce the overall backgrounds, including the preliminary definitions and the problem formulation. Section 3 presents the update rule for WASGD+ based on the modified objective function, along with our weight estimation, weight evaluating function and sample order generation methods. Subsequently, we provide the convergence and variance analysis for WASGD+ under various experiment settings in Sections 4.1 and 4.2. In Section 5, we show numerous experiment results on multiple datasets and detailed analysis on CIFAR-10 and CIFAR-100. Henceforth, we summarize additional related work in Section 6 and finally, we conclude in Section 7.

## 2 PRELIMINARIES AND PROBLEM STATEMENT

We consider minimizing the function $F(x)$ in a parallel environment [8] with $p$ workers. The stochastic optimization problem can be formulated as follows:

$$\min_x F(x) := \min_x E[f(x, \xi)],$$  \hspace{1cm} (1)

where $x$ represents model parameters and $\xi$ is a random variable that follows a probability distribution $Q$ over $\Omega$ such that $F(x) = \int_{\Omega} f(x, \xi)Q(\xi)d\xi$. In the parallel environment, as discussed in [9], one needs a nuanced formulation of the objective function. We found a good starting point in the Elastical Averaging SGD (EASGD) method [10] that reformulates the objective function with $p$ workers as:

$$\min_{x_1, x_2, \ldots, x_p} \sum_{i=1}^{p} \left( E[f(x_i', \xi')] + \frac{\lambda}{2} \|x_i' - \bar{x}\|^2 \right).$$  \hspace{1cm} (2)

where $x'$ is the variable (parameters) of local worker $i$; $\bar{x}$ represents the parameters for the center variable, which is stored by the master processor managing the communication. Related update rules are as follows:

$$x_{i+1}^t = x_i^t - \eta g_i^t(x_i^t) - \alpha(x_i^t - \bar{x}),$$  \hspace{1cm} (3)

and

$$\bar{x}_{i+1} = (1 - p\alpha)\bar{x} + \alpha \sum_{i=1}^{p} x_i^t,$$  \hspace{1cm} (4)

where $\eta$ is the learning rate; $\alpha = \eta \alpha$ represents the moving rate. During the communication, EASGD lets local workers communicate with the center variable only, and the moving rate $\alpha$ is preferred to be small leading to more exploration efforts of local workers. This setting can result in a serious issue in the searching process.

For the main method in [10], assuming that the communication order of local workers denotes their indices and the communication period is $r$. After finishing communications with all the workers, the modified parameters of the master can be written as:

$$\bar{x}_{t+p} = (1 - p\alpha)^p \bar{x}_t + \sum_{i=1}^{p} \alpha(1 - p\alpha)^{p-i} x_i^t.$$  \hspace{1cm} (5)

Since $1 - p\alpha < 1$, Equation (5) indicates that the maximum weight can only be assigned to $\bar{x}_t$ or the slowest worker $x_i^t$.

By applying Bernoulli’s inequality [11], we obtain the following condition:

**Condition 1.** If $\alpha < \frac{1}{1+p}$, then the maximum weight is given to the master’s previous parameters $\bar{x}_t.$

The proof of Condition 1 can be found in the appendix, which can be found on the Computer Society Digital Library at [http://doi.ieeecomputersociety.org/10.1109/TKDE.2020.3059458](http://doi.ieeecomputersociety.org/10.1109/TKDE.2020.3059458).
0.3047894. The condition depicts when the biggest weight is given to master’s previous parameters. Due to the preference of small \(\alpha\), the center master \(\bar{x}\) has higher possibility to be allocated the maximum weight. Given that \(\bar{x}\) is initialized at the beginning of the algorithm and \(\alpha\) is small, the center variable has a high chance to keep bad parameters for local workers in each iteration, which slows down the convergence speed.

To address the unreasonable assignment issue, we propose to remove \(\bar{x}\) and introduce a new term coined the \textit{weighted aggregation} in the quadratic penalty term [12] to generate a more healthy allocation, which shares the similar communication strategy in [13], [14], [15]. The combination term aggregates the parameters of all local workers and assigns larger weights to better performing local workers. Such settings can ensure all workers search around the current optimization, and enable them to do many explorations at the same time. Thus, we propose a new format of the stochastic optimization as follows.

Problem 1 (The Parallel Stochastic Optimization Problem).

\[
\min_{x^1, \ldots, x^p} \sum_{i=1}^{p} \left( E[f(x^i, \xi^i)] + \lambda \left\| x^i - \sum_{j=1}^{p} w^j x^j \right\|^2 \right),
\]

where \(w^j\) denotes the weight for the \(j\)th worker; \(x^i\) represents the parameters of local worker \(i\); and \(\xi^i\) follows the same distribution \(Q\).

Note that we consider all processors can sample the entire dataset. The problem of the equivalence of our objective and the original one is studied in the literature and is known as the \textit{augmentability} or the \textit{global variable consensus} problem [16], [10]. The penalty term \(\lambda\) is expected to control all local workers searching near the current optimal parameters.

### 3 METHODOLOGY

This section introduces the update rule of WASGD and then discusses its enhanced version WASGD+.

#### 3.1 Update Rule for WASGD

Given \(w^i\) the weight of the local variable \(x^i\), let \(\theta^i = \frac{w^i}{\sum_{j=1}^{p} w^j}\), Equation (6) can be rewritten as:

\[
\min_{x^1, \ldots, x^p} \sum_{i=1}^{p} \left( E[f(x^i, \xi^i)] + \lambda \left\| x^i - \sum_{j=1}^{p} \theta^i w^j x^j \right\|^2 \right).
\]

If we take the stochastic gradient descent with respect to \(x^i\), then we obtain the update rule for \(x^i\) on \(i\)th worker as:

\[
x_{i+1}^j = x_{i}^j - \eta \frac{\partial \left( E[f(x, \xi)] + \frac{1}{2} \left\| x - \sum_{j=1}^{p} \theta^j x^j \right\|^2 \right)}{\partial x^j},
\]

where \( \eta \) is the learning rate and \( g_i^j(x_i^j) \) is the partial derivative of \( E[f(x, \xi)] \) in terms of \( x^i \). Since \( x_i^j \) denotes the latest parameter from all workers, the sum term is a constant. Thus we have the following equation:

\[
\lambda \left\| x_i^j - \sum_{j=1}^{p} \theta^j x^j \right\|^2 = \lambda \left( x_i^j - \sum_{j=1}^{p} \theta^j x^j \right)
\]

Combining Equations (8) and (9), we can obtain the update rule for \(x_{i+1}^j\) as:

\[
x_{i+1}^j = x_i^j - \eta g_i^j(x_i^j) - \eta \lambda \left( x_i^j - \sum_{j=1}^{p} \theta^j x^j \right).
\]

If we denote \( \beta = \lambda \eta \), then Equation (10) can be rewritten as:

\[
x_{i+1}^j = x_i^j (1 - \beta) + \beta \sum_{j=1}^{p} \theta^j x^j - \eta g_i^j(x_i^j),
\]

where \( \beta \in [0, 1] \). Here \( \beta = 0 \) indicates a full rejection of the aggregation results, so there is no communication between local workers, whereas \( \beta = 1 \) indicates the system fully accepts the aggregation results. If \( \beta \in (0, 1) \), each local worker negotiates itself with the combination result. The bigger the \( \beta \), the more compromise from the local worker. Since deep neural networks usually have numerous local minima [17], [18], which have similar values of loss, a suitable \( \beta \) can help us improve the convergence rate. We will show the selection of \( \beta \) with more details in Section 5.

As shown in Fig. 1, suppose we have three local workers and only need to reach one of the local minima, as any of them would lead to a good enough loss function value. First, two local workers are attracted by the local minimum on the right, and one local worker is attracted by the left local minimum. After \( r \) iterations, the three local workers stop and communicate with each other. Then, since the blue worker and the yellow worker both move toward the local minimum on the right and have good performance, the orange worker leaves the left local minimum after the communication. They will restart at new positions closer to the local minimum on the right.

Based on Equation (11), the basic WASGD was proposed in the preliminary work, and detailed analysis can be found in [6]. The main algorithm of WASGD can be found in the Appendix, available in the online supplemental material. Here, we focus on discussing the enhanced version of the method, WASGD+, which will introduce improvements on the weight evaluation function and the sample order strategy.
3.2 Weight Evaluation Function

Based on [19] and [20], we learn that SGD shares many properties with the Simulated Annealing (SA). The core of the SA is controlled by the Boltzmann distribution that determines the possibility of accepting updates and guiding the result. Inspired by the enormous success of the Boltzmann distribution in SA, we have applied it to our communication process as well.

During the communication, the most important part is how to weigh the results; naturally, we assume that the weight should be inversely proportional to the loss energy, which in turn means that if the loss of the ith worker is small, then we should strengthen the weight of this worker, and vice versa. Since different tasks have different requirements for weight [21], the optimal weight function, subsequently, cannot always be the same [22]. The weight evaluation function for WASGD is \( \frac{1}{z} \) which can hardly meet such optimality requirement. We propose a new weight evaluation function based on the Boltzmann distribution [23]:

\[
\omega = e^{-Th(I \in \mathbb{R})},
\]

where \( h \) is the loss energy. If \( h \to 0 \), then \( e^{-Th} \to 1 \), the weight of each sample is equivalent to \( \frac{1}{p} \). In order to avoid this pitfall, we normalize the energy first. The weight function should be defined as:

\[
\omega^i = e^{-Th^i},
\]

where \( h^i = \frac{h}{\sum_{j=1}^{p} h} \). Then the normalized weight of the ith worker should be:

\[
\theta^i = \frac{e^{-Th^i}}{\sum_{j=1}^{p} e^{-Th^j}},
\]

where \( p \) is the number of local workers. Equation (14) shares the same form as the Boltzmann distribution and we can have the following property:

**Property 1.** If \( \Gamma \to \infty \), the weighting strategy is equivalent to broadcasting the best performance case while if \( \Gamma \to 0 \), the weighting strategy is equivalent to the equally weighted case.

The proof of Property 1 can be found in the Appendix, available in the online supplemental material. Property 1 shows that the new weight evaluating function is more flexible and can satisfy most cases as required.

**Property 2.** If we set equally weighted case as the baseline and try different values of \( \Gamma \). As \( \Gamma \to \infty \), the performance of the weighted case must be worse than that of the baseline, while as \( \Gamma \to 0 \), the weighted case will approach the performance of the baseline.

The proof of Property 2 is provided in the Appendix, available in the online supplemental material. Property 2 provides the asymptotic properties of the extreme case in terms of \( \Gamma \) where we can find the proper value of \( \Gamma \) more efficiently based on these properties. As [24] had successfully applied the Boltzmann distribution in the mixing of states of parallel SA, we also apply which as a weight evaluation function in the parallel SGD case.

3.3 Weight Accuracy Estimation

As stated above, let \( F \) denote the objective function, the normalized value of the loss energy during communication about the ith worker should be calculated as:

\[
h^i = \frac{\sum_{j=1}^{N} F(x^j_t - 1, \xi^j)}{\sum_{k=1}^{p} \sum_{j=1}^{N} F(x^k_t - 1, \xi^j)},
\]

where \( \tau \) denotes the communication period, \( p \) is the number of workers, and \( N \) is the size of the training samples. Based on Equation (13), the weight of the ith worker is defined as:

\[
\theta^i = \frac{\sum_{j=1}^{N} F(x^j_t - 1, \xi^j)}{\sum_{j=1}^{N} \sum_{k=1}^{p} F(x^k_t - 1, \xi^j)}.
\]

This step means we need to go through the whole training dataset in order to obtain the weight. Time needed for computing this process depends on the size of the dataset. If the sample size is very small, then it will not take too much time. For example, in CIFAR-10, the sample size is equal to half of a hundred thousand, computing these samples is very expensive.

We can use \( m \) samples to estimate the weight, then Equation (16) becomes:

\[
\theta^i_{true} \approx \theta^i = \frac{\sum_{j=1}^{m} F(x^j_t - 1, \xi^j)}{\sum_{j=1}^{m} \sum_{k=1}^{p} F(x^k_t - 1, \xi^j)} (m \leq N),
\]

where \( d_j \) denotes the index of the samples. Equation (17) will spend nearly \( \frac{m}{N} \) time of Equation (16). However, it still needs extra computing time. The loss function for classification has been defined as:

\[
F^i(x) = -\log \frac{e^{y_i}}{\sum_{k} e^{y_k}},
\]

where \( s_k \in z(x^i, \xi^i) \), \( z \) is the given function, \( j \) denotes the index of label, \( k \in [0, n_1] \), and \( n_1 \) is the number of label type. During the first step of the back propagation, we can obtain

\[
s_j = z_j = \sum_{k=1}^{N'} (W_{jk} s'_{k}(x^i, \xi^i) + b_{jk}),
\]

where \( N' \) is the size of the hidden layers, \( s'_{k} \) denotes function of the previous result. If we want to update the value of \( W_{jk} \), based on the chain rule, we will have

\[
\frac{\partial F^i}{\partial W_{jk}} = \frac{\partial F^i}{\partial z_j} \frac{\partial z_j}{\partial W_{jk}}.
\]

From Equation (20), we know that calculating \( \frac{\partial F^i}{\partial z_j} \) needs the value of \( \sum_{k} e^{y_k} \), which means the current loss energy of the ith sample does not need extra computing time. If we can use the loss energy during the back propagation, then the weight evaluation process does not need extra
computing time; therefore, we reformulate Equation (17) as:

\[ \hat{\theta}^t \approx \frac{-\frac{1}{m} \sum_{i=1}^{m} P(x_{i-1}, y_{i-1})}{-\frac{1}{m} \sum_{i=1}^{m} P(x_{i-1}, y_{i-1})} \cdot \sum_{i=1}^{m} \frac{\sum_{j=1}^{m} P(x_{i-1}, y_{i-1})}{\sum_{j=1}^{m} P(x_{i-1}, y_{i-1})}. \] (21)

Based on Equation (21), the loss should be recorded at the same time and space for all workers, which will restrict the exploration for local workers. Allowing more explorations will go through different training steps. So we only record the loss at the same space due to the fact that different workers will use different samples and will be closer to each other. The range for the error can be shown as follows:

\[ \text{error} = \sum_{i=1}^{p} |\hat{\theta} - \hat{\theta}^{(i)}| \] (23)

The range for the error can be shown as follows:

\[ \begin{cases} 
\text{error} \geq \sum_{i=1}^{p} (\hat{\theta}^{(i)} - \hat{\theta}^{\text{true}}) = 0 \\
\text{error} \leq \sum_{i=1}^{p} (\hat{\theta}^{(i)} + \hat{\theta}^{\text{true}}) = 2. 
\end{cases} \] (24)

We can also apply this estimation method to the Multiplicative Weight Update Method [26] which also needs the weight to update the possibility of selection.

### 3.4 Sample Order

Correct weighting of samples can speed up the convergence rate [27]. However, this process needs to find the right distribution of the samples causing extra analysis time. The right weight of the samples can be reflected by the best order of the samples that an SGD goes through.

In the SGD, each sample will guide the solution to a direction (gradient). The effect of samples in the same direction of the current sample will be reinforced and the effect of samples in the opposite directions of the current sample will be diminished. Therefore, the order of the samples for the SGD affects the solution quality in one epoch, which in turn will influence the overall convergence rate. This is illustrated by the example below.

As shown in Fig. 2, we are given 12 samples where half of which have the value 1 and the other half, the value 0. We want to determine the value of \( d \) in the function \( y = d \) such that it has the smallest least square error among these samples. To emphasize the importance of the order of the samples, we assign each sample an index. This index denotes the order of going through these 12 samples when we run the SGD. The solid red points indicate used samples and the blue empty points refer to the unused samples. The dashed blue line represents the previous solution while the solid blue line corresponds to the current solution obtained by the SGD steps based on the solid samples. Figs. 2a, 2b, 2c, and 2d show the fitting procedure with one sample order. The initial solution starts at \( y = c, \) in (a) and (b) the solutions are modified by six SGD steps based on the six samples with the value \( b, \) The solution will arrive at a position which will definitely approach \( y = b \) from the above. Then the SGD improves the current solution by going through the rest six samples with the value \( a, \) Finally, the solution will be close to \( y = a. \) This is an inappropriate solution as we know the optimal solution is \( y = (a + b)/2. \) Figs. 2e, 2d, 2e, and 2f show the procedure with another sample order where the samples with value \( a \) are separated by the samples with value \( b \). The current solution will be optimized for both \( a \) and \( b \) for every two SGD steps and will be closer to \( y = (a + b)/2. \)

When training with a large dataset, it has a high possibility to select some samples with the same label without grouping the samples in advance. The time of sample grouping is proportional to its size. Even if we group the samples, for a dataset with \( N \) samples, the number of sample orders is \((N! - c), \) where \( c \) denotes a few known bad orders. Such process is equivalent to selecting some data in a dataset which can maximally differentiate these data from others, which is proven NP-hard [28]. It is nearly impossible to try all these orders and find the best sample order even in a parallel environment. However, in parallelization, we can obtain a relatively better sample order by comparisons.

---

Fig. 2. Results of different sample sample orders.
In each communication procedure, we assign a score to each local worker based on its performance. To determine the score, we first find the average performance, which is the mean of the losses from all the workers at the current step. Furthermore, we calculate the fluctuation, that is, the standard deviation of these losses. Then, we normalize each local worker’s current loss by subtracting the average performance and dividing by the fluctuation and defining it as the score. Assume that the losses for all the workers at the same step follow a normal distribution. Then, the score follows a standard normal distribution with mean zero and standard deviation one. By the empirical rule, if this score is less than minus one, its performance is better than 84 percent of the local workers. We consider the sample order of this local worker relatively better and save it for the next epoch.

For some bad orders, they have some wayward parts during the searching process. The bad performance is caused by unsuitable orders from the other parts. In order to generate relatively better sample orders, we can retain these special parts and shuffle the other parts. For example, we divide the total sample into n parts in which bad orders exist. Suppose that there are m parts with scores equal to or greater than -1, indicating the existence of bad orders. We change the m parts while retaining the seeds for the rest n - m parts.

As suggested by many previous studies (e.g., [29]), learning from different views in the training process should enhance the model performance. Thus, we expect that having different orders of workers will improve the aggregation result given different positions of the local workers. This method has a higher possibility to generate a better order than simply randomly shuffling all the samples.

### 3.5 WASGD+
Following [30], we introduce a distributed deep learning system without a parameter server. We use $e^{-\gamma t}$ to evaluate the weight of the $i$th worker. The detailed procedure is demonstrated in the Appendix, available in the online supplemental material. Each local worker is only responsible for updating its own parameters and going through the samples in different orders. Note that the iteration will increase by one after each update, and the local workers will wait when the iterations are the multiples of communication period $\tau$. When iterations are divisible by $\tau$, the worker will hold the search until it receives an aggregation result based on all workers. During communication, the estimated loss will be used to generate score and weight for the performance of the current parameters. After fixed iterations, each local worker will compare its score with the judgment. If the score satisfies the judgment, the current order will be preserved; otherwise, it will be replaced by a new one.

The choice of asynchronous or synchronous algorithm of our method is based on the communication period and the time difference for computing each sample. If the time difference for computing each sample is large and the communication period is small, the total computing time for the local worker has higher possibility to vary a lot from each other. Then we choose asynchronous algorithm. If the time difference in computing each sample is large and communication period is large, the possibility that computing time varies substantially from one another has been decreased, then we use the synchronous algorithm. If the time difference is small which means each local worker completes the work almost at the same time, then we use the synchronous algorithm.

Since the time difference for computing samples in MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100 is small, we use the synchronous version. The algorithm will be terminated when the required loss is reached. We also provide the asynchronous version of our method in the Appendix, available in the online supplemental material.

### 4 THEORETICAL ANALYSIS
In this section, we shall discuss the convergence and stability of our method.

#### 4.1 Convergence Analysis
Recall that Equation (7) is the update rule for the parameter vector $x$ and we set $\beta$ to be one. Then the solutions from all the workers of WASGD or WASGD+ are the same in every $t$ steps. We reformulate the update rule as:

$$ x_{t+\tau} = \sum_{i=1}^{p} \left[ \theta_i \sum_{j=1}^{\tau} \left[ x_j - \eta g_j(x_j) \right] \right]. $$

As shown above, the convergence of WASGD+ is equivalent to the convergence of the sequence of the solutions $\{x_{\tau t}\}_{t=1}^{\infty}$. Let $S_{t;k}^{i} = (s_{t;k}^{i})_{t=1}^{\infty}$ be a sequence of sample index at time step $t$. We can define $h_{t;k}^{i}$ and $H_{t;k}^{i}$ for the $i$th worker as:

$$ h_{t;k}^{i}(x|s_{t;k}^{i}) = x - \eta g_{s_{t;k}^{i}}(x), \tag{26} $$

and

$$ H_{t;k}^{i}(x|S_{t;k}^{i}) = \begin{cases} h_{t;k}^{i}(x|s_{t}^{1}), & k = 1 \\ h_{t;k}^{i}(x|s_{t;k}^{i}) \circ H_{t;k-1}^{i}(x|S_{t;k-1}^{i}), & k > 1. \end{cases} \tag{27} $$

Then, Equation (25) can be rewritten as:

$$ x_{t+\tau} = \phi(x_t) = \sum_{i=1}^{p} \left[ \phi_i \cdot H_{t,\tau}^{i}(x_t|S_{t,\tau}^{i}) \right]. \tag{28} $$

Assume that $F(x)$ is convex and Lipschitz continuous. We first introduce the lemma below:

**Lemma 1.** $\forall i, t, k$, when the learning rate in $h_{t;k}^{i}$ is low, let $(P(R,m), W_2)$ be the space of distributions, where $(R, m)$ is a Rondon space, $P(R,m)$ is the set of all distributions over $(R, m)$, and $W_2$ is the Wasserstein distance between two distributions. If we define $\varphi_{t;k}^{i}$ by applying $h_{t;k}^{i}$ pointwise to $R$, $\varphi_{t;k}^{i}$ is a contraction mapping.

**Proof.** This lemma can be obtained in [3]. It first shows that $h_{t;k}^{i}$ is a contraction mapping and then proves that the induced mapping $\varphi_{t;k}^{i}$ is a contraction mapping over $(P(R,m), W_2)$.\hfill $\square$

By the connection between $h_{t;k}^{i}$ and $\varphi_{t;k}^{i}$, we can study $\Phi$. As shown in [3], they discussed the situation that $\Phi$ is the linear combination of $\varphi_{t;k}^{i}$, which in our case is when $\tau = 1$. 

We extend its proof to $\tau > 1$ where the composite functions exist.

**Theorem 1.** Let $\Phi$ be the mapping in $(P(R, m), W_p)$ induced by $\phi$ in Equation (28) by applying $\phi$ pointwise to $R$. Then $\Phi$ is a contraction mapping.

**Proof.** $\forall R_1, R_2 \in R$, by the definition of $\Phi$, we have

$$W_p(\Phi(R_1), \Phi(R_2)) = \sum_{i=1}^{p} \left[ \theta_i \cdot \phi_i^t \circ \phi_i^t \circ \ldots \circ \phi_i^t (W_p(R_1, R_2)) \right].$$

(29)

Since $\phi_i^t$ is a contraction mapping when $1 \leq k \leq \tau$, there exists a Lipschitz constant $c_{i,k} < 1$ such that

$$\text{RHS of Equation (29)} \leq \sum_{i=1}^{p} \left[ \theta_i \cdot c_{i,1} \cdot \phi_i^t \circ \ldots \circ \phi_i^t (W_p(R_1, R_2)) \right].$$

(30)

By induction, we have:

$$\text{RHS of Equation (30)} \leq \sum_{i=1}^{p} \left[ \theta_i \cdot \prod_{k=1}^{\tau} c_{i,k} (W_p(R_1, R_2)) \right]$$

(31)

Since $c_{i,k} < 1$ and $\sum_{i=1}^{p} \theta_i = 1$, $\Phi$ is a contraction mapping in $(P(R, m), W_p)$. $\square$

By Theorem 1, per the properties of contraction mapping, the sequence $\{x_i\}_{i=1}^{\infty}$ will converge at an exponential rate.

### 4.2 Variance Analysis

Based on [31], we propose a general version of variance under different weighted strategy about the following function: $F(x) = \frac{1}{2} c x^2$. By using the gradient samples of the form $g(x) = cx - \hat{b}x - \hat{h}$ where $\hat{b}, \hat{h}$ are random variables with mean zero and variance $\sigma^2_b, \sigma^2_h$ p local workers will choose to communicate with each other with the probability $\xi$ after each step. We can define the variance of our method as shown in Lemma 2 below.

**Lemma 2.** Given $\omega, \delta$, where $\omega = \sum_{i=1}^{p} (\theta_i)^2, \delta = \frac{\xi}{(1-\xi)(2-\eta^2)}$, $\eta$ is the learning rate and $\xi$ is the communication probability, the asymptotic variance of the weighted aggregating SGD is

$$\lim_{t \to \infty} \text{Var} \left( \sum_{i=1}^{p} \theta_i^t x_i^t \right) = \eta \sigma^2_b \omega \left( 2c - \eta c^2 - \eta \sigma^2_h \frac{1 + \delta \omega}{1 + \delta} \right)^{-1}.$$  

(32)

The proof of Lemma 2 can be found in the Appendix, available in the online supplemental material. The above Lemma shows that the variance of the weighted aggregating SGD can be determined and controlled based on Equation (32). A predetermined variance will result in the stability of our method. To better understand the performance of our method, we have the following Lemma:

**Lemma 3.** With a large data, given $p$ workers, if communication probability $\xi = 1$, equally weighted case is equivalent to mini-batch gradient descent with the same learning rate $\eta$.

The proof of Lemma 3 is provided in the Appendix, available in the online supplemental material. As suggested by [32], mini-batching can effectively reduce the variance of the stochastic gradient estimates. Our Lemma 3 further serves as a boundary condition regarding the performance of the parallel stochastic gradient descent.

### 5 EXPERIMENTS AND RESULTS

We evaluate our method by applying it to the convolutional neural network (CNN) for specific classification tasks.

#### 5.1 Order Effect

Sample order is a key factor in influencing the aggregation performance. A proper sample order can help improve the performance of local workers and modify the communication result. We numerically prove the effect of sample order by performing certain tests Fashion-MNIST and CIFAR-10 datasets with $\delta = 1, 10, 100, 1000$ where $\delta$ denotes the numbers of continuously gone-through samples with the same label.

As shown in Fig. 3, if $\delta = 1000$ which means that the local worker was given only one type of label during each communication period, the improvement in accuracy and loss can be largely ignored. For $\delta = 100$, although it sometimes achieves better performance compared with $\delta = 1000$, the convergence rate is still very slow. For $\delta = 1, 10$, their performance is better than $\delta = 100, 1000$. As CIFAR-10 is more complex than Fashion-MNIST, the difference between $\delta = 1$ and $\delta = 10$ on the training error is becoming quite enlarged. Based on such phenomenon, we can learn that the more complex the dataset, the more important the sample order.

Although $\delta = 1$ achieves the best result among the other three situations, one cannot conclude that $\delta = 1$ is the best sample order among all the orders. Even if we can find the best sample order for one dataset, it might not be suitable for the other datasets. Given that it is hard to try all the possible orders, our parallel method can help approach the optimal sample order sooner.
5.2 Experiment Settings

In this section, we shall discuss the general experiment settings for the datasets and the detailed settings for different benchmarks.

5.2.1 General Settings

Now, we show the experiments on four datasets (i.e., MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100\(^1\)), given the main focus of CNN on image classification [33].

For the CIFAR-10 and CIFAR-100 datasets, we implement CNN with eight convolution layers and four fully connected layers. Following [10], given C the fully-connected convolutional operator, M the max pooling operator, D the linear operator with dropout rate being one, and F the linear operator with softmax output, our CNN structure can be described as (3,32)C(64,32) M(64,16) C(128,16) M(128,8) C(256,8) M(256,4) C(512,4) M(512,2) D(128,1) D(256,1) D(512,1) D(1024,1) F(10,1). For the MNIST and Fashion-MNIST datasets, we implement the following 6-layer CNN: (1,28) C(16,24) M(16,12) C(32,8) M(32,4).

We set the constant learning rate for all experiments, \(\eta = 0.001\) for CIFAR-10, CIFAR-100 and \(\eta = 0.01\) for Fashion-MNIST, MNIST. We check the cross entropy loss of the whole training examples every 10,000 iterations. The running time includes two parts: computation and communication. For CIFAR-10 and CIFAR-100, we test the algorithm performance under different numbers of Tesla K80 \((p = 2, 4, 8)\). For Fashion-MNIST and MNIST, we implement the experiments on different numbers of CPUs \((p = 4, 8, 16)\). In order to better present the tendency of performance on error, we take the logarithm of the results. Also, the termination condition is based on the convergence situation of all the methods. As long as one of the benchmark methods has converged, we shall terminate all the methods. Due to page limitation, we show detailed results on CIFAR-10 and CIFAR-100, while only the main accuracy results for Fashion-MNIST and MNIST. All experiments are conducted under TensorFlow [34].

5.2.2 Benchmark Settings

We compared our WASGD+ with five parallel methods and one sequential method:

- **SGD** [35]. The standard sequential SGD with constant learning rate \(\eta\).
- **SimuParallelSGD (SPSGD)** [3]. The algorithm divides the data into \(p\) parts, and each local worker only gets one part. After several iterations they average the sum of all parameters.
- **Elastical Averaging SGD (EASGD)** [10]. It is an efficient asynchronous optimization method. Local workers only communicate with the center variable. During the communication, the center variable changes itself based on the parameters of local workers and also updates the parameters of the local workers. We empirically set the communication period \(\tau = 50\). Based on [10], we set \(\alpha = 0.9^p\) for CIFAR-10, CIFAR-100 and \(\alpha = 0.009\) for Fashion-MNIST, MNIST.
- **Original Multiplicative Weight Update Method (OMWU)** [26]. This is a classical method published in 1957. The weights of workers’ performance influence their chosen possibilities in the next \(\tau\) steps. As long as the iterations are large enough, it can find the best performance of local workers. We compute the total loss at the last step before communication to evaluate the weight and the communication period \(\tau = 1000\).
- **Modified Multiplicative Weight Update Method (MMWU)**. We modify the process of calculating the weight. The weight in the original work was based on the performance of the entire training dataset. We replaced it with the proposed estimation approach and \(\tau = 1000\).
- **Weighted Aggregating SGD (WASGD)** [6]. This is a synchronous version of the parallel method that we have published in 2019. The weights of the workers are determined by the inverse of their loss energies. The communication result is aggregating based on the weights of local workers and is totally accepted, so \(\beta = 1\). To achieve the same estimation accuracy as WASGD+, we set \(m = 150\), and the communication period \(\tau = 1000\).

5.3 Parameter Analysis in WASGD+

Now, we discuss the impact of parameter selections on the performance of the proposed method.

5.3.1 Γ analysis

Different values of \(\Gamma\) denote different weighting strategies. We try different values of \(T\) where \(T = \frac{1}{\Gamma}\) compared with the equally weighted case. When \(T \rightarrow 0\), it is equivalent to the sequential case, the result must be worse than the equally weighted one [3]. When \(T \rightarrow \infty\), it means the weight value of each worker is almost equal to each other, the result will be the same as the equally weighted one.

In order to avoid outliers, we conduct five experiments with one epoch. The point in Fig. 4 is calculated by:

\[
\frac{1}{5} \sum_{i=1}^{5} \frac{\sum_{j=1}^{N} (v_{f}(jud) - v_{f}(cur))}{N},
\]

where \(N\) denotes the number of records in one experiment, \(v_{f}(jud)\) means the average baseline value of the \(j\)th record and \(v_{f}(cur)\) represents the value of the \(j\)th record for experiment \(i\). We also plot the error bar of each point which is equal to the variance of \(\sum_{j=1}^{N} \frac{(v_{f}(jud) - v_{f}(cur))}{N}, \sum_{j=1}^{N} \frac{(v_{f}(jud) - v_{f}(cur))}{N}\). As the error bar is very small, this confirms the stability of our experiments.

As shown in Fig. 4, for MNIST and CIFAR-10 datasets, the optimal \(T\) is equal to one. For Fashion-MNIST dataset, the optimal \(T\) is 10. Regarding CIFAR-100 dataset, the proper choice of \(T\) is \(10^{-1}\).

Due to the decreasing property of the curve compared to the baseline when \(T \rightarrow 0\), we can also estimate the scope of the change point that distinguishes the performance between the weighted case and the baseline from Fig. 4. For CIFAR-10, Fashion-MNIST and MNIST, when \(T \leq 0.1\), the

---

1. MNIST (http://yann.lecun.com/exdb/mnist/), Fashion-MNIST (https://arxiv.org/abs/1708.07747), CIFAR-10 and CIFAR-100 (https://www.cs.toronto.edu/~kriz/cifar.html).
results begin to be worse than the baseline. So the range of change point for these three datasets is around $10^{-1}$. As to CIFAR-100 dataset, the scope of change point is around $10^{-2}$. We learn that as the complexity of the dataset increases, the range of change point approaches to zero.

### 5.3.2 $\beta$ Exploration

$\beta$ determines how much we accept from the communication result. When $\beta = 0$, it means rejecting the communication result which is equivalent to the sequential one and must be the worst case, and $\beta = 1$ denotes totally accepting the communication result. In order to find the optimal $\beta$, we set $\beta = 1$ as the baseline case and compare it to other values of $\beta$. To be fair, the experiments have been conducted five times with one epoch. We use the same formula as Equation (33) to calculate the point in Fig. 5. We also plot the error bar of each point in the same way above.

As shown in Fig. 5, for MNIST and CIFAR-10 datasets, the optimal $\beta$ is equal to 0.9. For CIFAR-100, the proper choice is 0.8 and for Fashion-MNIST, the optimal $\beta$ is 0.7. As $\beta$ decreases, the results on all four datasets become worse than the baseline case. Since the error bar is very small, the stability of our experiments is guaranteed.

### 5.3.3 Choice of Estimation Sample Size $m$

Estimation sample size $m$ is also an important parameter that needs to be adjusted. If $m$ is small, local workers will get more freedom to explore, but the accuracy of the estimated weight will be affected. If $m$ is large, the accuracy of the estimated weight can be guaranteed, but the efficiency of parallelization will be weakened. So we need to balance the trade off between the accuracy of estimation and the efficiency of parallel searching.

We have tried different values of $m$ and found the best choice. As shown in Fig. 6, for $m = 1, 10$, although they allow more explorations than $m = 100, 1000$, their accuracy of estimation is low and unstable.

Comparing $m = 100$ and $m = 1000$, they both perform well in terms of estimating the accuracy, we choose $m = 100$ for maintaining the efficiency of parallel computing.

### 5.4 Performances on Different $\tau$

We have multiple choices for the communication period $\tau$, smaller $\tau$ might improve the convergence rate but spends too much time on communicating. Larger $\tau$, on the other hand, will weaken the effect of parallel computing. So we want to explore the performance on different $\tau$ and find an optimal one to balance between time and convergence rate.

The author of EASGD has tried $\tau = \{1, 4, 16, 64\}$ and find the optimal $\tau = \{16, 64\}$. The tendency to achieve better test performance with larger $\tau$ is a strength of the EASGD algorithm. According to this characteristic, we tested the communication periods from the following set $\tau = \{10^1, 50, 10^2, 10^3, 10^4\}$ for EASGD, WASGD and WASGD+. In order to guarantee the fairness of the results, we compared the performance after two epochs of training. Fig. 7 depicts the performance of two methods under different processors. As shown in the picture, when $\tau$ and $p$ are the same, WASGD can achieve better performance than EASGD. WASGD+ is outperforming all other benchmarks. We also find that $\tau = 1000$ in WASGD+ achieves almost the same performance as $\tau = 50$ in EASGD under the same $p$.

### 5.5 Performance Comparison

We first test the effect of our two proposed techniques (sample order -SO- and adaptive weight -AW-). To do so, we benchmark them against the original version of WASGD. Table 1 reports the training and testing performance of the original version of WASGD versus the additive versions (SO and AW) on CIFAR10 and CIFAR100. As can be seen, both the reported losses and errors suggest that the additive versions of WASGD outperform the original version, indicating a positive impact of the sample order and adaptive weighting method on the overall performance.

Then, we benchmark our WASGD and WASGD+ against the baseline methods. Figs. 8 and 9 illustrate the training error, training loss, test error, and test loss of our method and the baselines with different parallel sizes on CIFAR-10 and CIFAR-100. As expected, our WASGD+ method consistently outperforms all the baselines. Regarding the
benchmark methods, with the increase of computing processors (workers), the performance of SPSGD becomes unstable, due to the fact that averaging the parameters in non-convex cases leads to divergence. MMWU shares the same performance with the sequential SGD, indicating that the sequential SGD reaches the best performance it may achieve. As MMWU automatically allocates the weights based on the current performance, given a large number of iterations, it will converge to the best performance of local workers. In our experiments, the iteration numbers are large, so that the sequential SGD achieves its optimal performance, which is found comparable with MMWU. Also, OMWU does not show a strong performance in our tests, mainly due to its heavy burden of the weight computing based on the whole training samples.

Fig. 10 demonstrates the results on Fashion-MNIST and Fig. 11 depicts the results on MNIST. Same conclusions are reached. We can see that our WASGD+ method consistently outperforms all the benchmarks. Although EASGD may have a better performance with a careful parameter selection, the algorithm does not offer a standard way of yielding good performance.

Overall, our experiments on four datasets with different sizes and scales demonstrate the stability and scalability of our methods in handling simple and complex learning tasks alike.

### 6 RELATED WORK

Multicore and distributed optimization algorithms have aroused much attention in recent years [4]. Based on the communication method, parallel SGD can be divided into two major types: (i) centralized algorithms, and (ii) decentralized algorithms.

#### TABLE 1

| Method        | Metric | Training error | Training loss | Test error | Test loss |
|---------------|--------|----------------|---------------|------------|----------|
| WASGD (CIFAR10) | 50%    | 0.96           | 39%           | 1.12       |
| WASGD + SO    | 30%    | 0.85           | 36%           | 1.00       |
| WASGD + AW    | 71%    | 0.86           | 36%           | 1.01       |
| WASGD (CIFAR100) | 78%     | 3.16           | 79%           | 3.28       |
| WASGD + SO    | 73%    | 2.91           | 75%           | 3.04       |
| WASGD + AW    | 77%    | 3.07           | 79%           | 3.20       |
Fig. 8. Experiments on CIFAR-10.

Fig. 9. Experiments on CIFAR-100.
6.1 Centralized Algorithm

For the implementation of centralized algorithms, the proposed strategy [10] is that the master will only be responsible for receiving the update parameters. Once the local worker sends its newest parameters to the master, the master will change a part of its current parameters and return which to the local worker. After receiving the modified result from the master, the local worker also changes its results partially based on the information received. Due to the heavy burden of computing gradients and updating parameters, the case when only the master is in charge of updating the parameters was further discussed in [4]. The main task of the local workers is to calculate the gradients based on the current parameters and sending them to the master. To avoid the staleness of local workers, they also add extra backup workers in the training process. The master should follow the “first come, first serve” principle to update the parameters. Upon receiving enough number of gradients, the master will update the parameters and send the new parameters to all the local workers. For those delayed results based on the previous parameters, the master rejects them.

Following the idea that the first few steps are highly related to the final result, a new warm up scheme [36] was adopted in the parallel method. Such strategy that applies small learning rate at the beginning of training can ensure that the algorithm will approach the local optimal smoothly at the beginning and land at a proper position while converging in the end. To devote sufficient resources to the searching process, we adopt the decentralized method in our framework.

6.2 Decentralized Algorithm

Regarding decentralized algorithms, the work in [3] is considered as the first parallel SGD method. This method is also the first attempt to introduce the idea of weight. To deal with large data, it proposes to equally split the data among local workers and let each worker update itself based on the assigned data. Once the local worker finishes its job, it will send its parameters to all the other workers. When all the local workers finish their jobs, they will average the sum of received parameters and their own parameters. Such process can be seen as a special case of equally weighted case by setting the communication period equal to the size of assigned data, which is also covered in our method. Our method can be considered as a more generalized form of [3].

In order to update the parameters as soon as possible, the strategy of [5] suggested to open the access of global parameters to all the local workers. As long as the local workers get the updated direction of the global parameters, they can directly modify it. To avoid eliminating other workers’ results, gradient sparsity is required. A more general version of [5] was proposed by [37]. It extended Hogwild to a more common case by letting each worker updates the global model vector stored in a shared memory following some topological structures (e.g., bipartite graph). By applying such topological structure, it can eliminate the effect of overlapping updated result in the dense case.

6.3 Other Work

For the implementation environments, some studies are conducted in distributed systems [4], [10], [36], [38]; while others are implemented in a shared memory architecture [5], [37], [39].

Here we also summarize several newly proposed parallel SGD methods. The work in [40] proposed a distributed version of SGD with variance reduction named DisSVRG. In order to accelerate the training process, it uses an
acceleration factor to control the adaptive learning rate along with an adaptive sampling strategy. The paradigm of [41] estimated the cost of asynchronous communication by providing closer upper and lower bounds. By discovering the balance between the maximum delay and the SGD convergence rate, it enhances the performance of the parallel SGD. Finally, the work in [42] applied the Bayesian method to mitigate the straggler effect in synchronous gradient-based optimization and achieved speedup in the parallel SGD training process.

7 Concluding Remarks

In this paper, we modified the parallel SGD method from our previous work [6], and coined this novel algorithm WAGSD+. In this new method, we connect the quality of samples with sample orders, providing a reasonable way of finding the relatively optimal sample orders based on the performance of all local workers. We also provide a more flexible weight evaluation function that can satisfy different requirements of weighing strategies (e.g., equally weighted [31] or broadcast the best performance of local workers [43]). The newly proposed weight evaluation function also builds a connection between two independent stochastic processes: SA and SGD. Since SA has already been thoroughly studied, we can thereby explore the parallel SGD more efficiently. After proving the convergence and stability of our method, we apply it to CNN, and compare to the state-of-the-art techniques in handling parallel stochastic optimization problems. Our results suggest a consistent superiority of our method over other existing methods under various datasets and with different numbers of computing cores. Taken together, these affirm the effectiveness and efficiency of our new WAGSD+ method in training deep learning models.

Acknowledgments

The authors thank Stony Brook Research Computing and Cyberinfrastructure, and the Institute for Advanced Computational Science at Stony Brook University for access to the high-performance SeaWulf computing system. Keli Xiao acknowledges the support from the National Science Foundation of China (Grant #62076112 and #91746109). Wei Zhu acknowledges the support from the Institute of Education Sciences (Grant #R305A18027).

References

[1] C. Zhang, P. Patras, and H. Haddadi, “Deep learning in mobile and wireless networking: A survey,” IEEE Commun. Surveys Tuts., vol. 21, no. 3, pp. 2224–2287, Third Quarter 2019.
[2] H. Robbins and S. Monro, “A stochastic approximation method,” Ann Math. Statist., vol. 22, pp. 400–407, 1951.
[3] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, “Parallelized stochastic gradient descent,” in Proc. Advances Neur. Inf. Process. Syst., 2010, pp. 2595–2603.
[4] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, “Revisiting distributed synchronous SGD,” 2016, arXiv:1604.00981.
[5] B. Recht, C. Re, S. Wright, and F. Niu, “Hogwild!: A lock-free approach to parallelizing stochastic gradient descent,” in Proc. Advances Neur. Inf. Process. Syst., 2011, pp. 693–701.
[6] P. Guo, Z. Ye, and K. Xiao, “A weighted aggregating SGD for scalable parallelization in deep learning,” in Proc. IEEE Int. Conf. Data Mining, 2019, pp. 1072–1077.
[7] A. T. Lopes, E. de Aguiar, A. F. De Souza, and T. Oliveira-Santos, “Facial expression recognition with convolutional neural networks: coping with few data and the training sample order,” Pattern Recognit., vol. 61, pp. 610–628, 2017.
[8] D. P. Bertsekas and J. N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, vol. 23, Englewood Cliffs, NJ, USA: Prentice Hall, 1989.
[9] Z. Ye, K. Xiao, and Y. Deng, “A unified theory of the mobile sequential recommendation problem,” in Proc. IEEE Int. Conf. Data Mining, 2018, pp. 1380–1385.
[10] S. Zhang, A. E. Charomanska, and Y. LeCun, “Deep learning with elastic averaging sgd,” in Proc. Advances Neur. Inf. Process. Syst., 2015, pp. 685–693.
[11] P. S. Bullen, Handbook of Means and Their Inequalities, vol. 560, Berlin, Germany: Springer, 2013.
[12] J. Nocedal and S. Wright, Numerical Optimization. Berlin, Germany: Springer, 2006.
[13] K. Xiao, Z. Ye, L. Zhang, W. Zhou, Y. Ge, and Y. Deng, “Multiuser mobile sequential recommendation for route optimization,” ACM Trans. Knowl. Discovery Data, vol. 14, no. 5, pp. 1–28, 2020.
[14] Z. Ye, L. Zhang, K. Xiao, W. Zhou, Y. Ge, and Y. Deng, “Multiuser mobile sequential recommendation: An efficient parallel computing paradigm,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2018, pp. 2624–2633.
[15] Z. Ye, K. Xiao, Y. Ge, and Y. Deng, “Applying simulated annealing and parallel computing to the mobile sequential recommendation,” IEEE Trans. Knowl. Data Eng., vol. 31, no. 2, pp. 243–256, Feb. 2019.
[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” Foundations Trends® Mach. Learn., vol. 3, no. 1, pp. 1–122, 2011.
[17] A. Choromanska, M. Henaff, M. Mathieiu, G. B. Arous, and Y. LeCun, “The loss surfaces of multilayer networks,” in Proc. Int. Conf. Artif. Intell. Statist., 2015, pp. 192–204.
[18] S. J. Reddi, A. Hefny, S. Star, B. Poczos, and A. Smola, “Stochastic variance reduction for nonconvex optimization,” in Proc. Int. Conf. Mach. Learn., 2016, pp. 314–323.
[19] L. Bottou, “Stochastic gradient learning in neural networks,” Proc. Neuro-Nanos, vol. 91, no. 8, 1991, Art. no. 12.
[20] L. R. Rere, M. I. Fanany, and A. M. Armymurphy, “Simulated annealing algorithm for deep learning,” Procedia Comput. Sci., vol. 74, 2015, pp. 137–144, 2015.
[21] C.-H. Yeh, R. J. Willis, H. Deng, and H. Pan, “Task oriented weighting in multi-criteria analysis,” Eur. J. Oper. Res., vol. 119, no. 1, pp. 130–146, 1999.
[22] S. Zhang, S. Li, J. Hu, H. Xing, and W. Zhu, “An iterative algorithm for optimal variable weighting in k-means clustering,” Commun. Statist.-Simul. Comput., vol. 48, no. 5, pp. 1346–1369, 2019.
[23] D. A. Wolf-Gladrow, “Lattice Boltzmann models,” in Lattice Gas Cellular Automata and Lattice Boltzmann Models. Berlin, Germany: Springer, 2000, pp. 159–246.
[24] K.-W. Chu, Y. Deng, and J. Reinitz, “Parallel simulated annealing by mixing of states,” J. Comput. Phys., vol. 148, no. 2, pp. 646–662, 1999.
[25] Y. Xie et al., “Joint deep multi-view learning for image clustering,” IEEE Trans. Knowl. Data Eng., early access, Feb. 14, 2020, doi: 10.1109/TKDE.2020.2973981.
[26] C. Dwork et al., “The algorithmic foundations of differential privacy,” Foundations Trends® Theor. Comput. Sci., vol. 9, no. 3–4, pp. 211–407, 2014.
[27] D. Needell, R. Ward, and N. Srebro, “Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm,” in Proc. Advanc. Neur. Inf. Process. Syst., 2014, pp. 1017–1025.
[28] Z. Liu and Y. He, “Differentiating search results on structured data,” ACM Trans. Database Syst., vol. 37, no. 1, pp. 1–30, 2012.
[29] X. Jin, F. Zhuang, S. Wang, Q. He, and Z. Shi, “Shared structure learning for multiple tasks with multiple views,” in Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases, 2013, pp. 353–368.
[30] P. Watcharapichat, V. L. Morales, R. C. Fernandez, and P. M. A. Zehet, “Ako: Decentralised deep learning with partial gradient exchange,” in Proc. 7th ACM Symp. Cloud Comput., 2016, pp. 84–97.
[31] J. Zhang, C. De Sa, I. Mitliagkas, and C. Ré, “Parallel SGD: When does averaging help?,” 2016, arXiv:1606.07365.
[32] F. Shang et al., “VR-SGD: A simple stochastic variance reduction method for machine learning,” IEEE Trans. Knowl. Data Eng., vol. 32, no. 1, pp. 186–202, Jan. 2018.
S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural-network approach,” IEEE Trans. Neural Netw., vol. 8, no. 1, pp. 98–113, Jan. 1997.

M. Abadi et al., “Tensorflow: A system for large-scale machine learning,” in Proc. 12th USENIX Symp. Operating Syst. Des. Implementation, 2016, pp. 265–283.

S. Ruder, “An overview of gradient descent optimization algorithms,” 2016, arXiv:1609.04747.

M. Abadi et al., “Tensorflow: A system for large-scale machine learning,” in Proc. 12th USENIX Symp. Operating Syst. Design and Implementation, 2016, pp. 265–283.

P. Goyal et al., “Accurate, large minibatch SGD: Training imagenet in 1 hour,” 2017, arXiv:1706.02677.

X. Pan et al., “Cyclades: Conflict-free asynchronous machine learning,” in Proc. Advances Neural Inf. Process. Syst., 2016, pp. 2568–2576.

J. Dean et al., “Large scale distributed deep networks,” in Proc. Advances Neural Inf. Process. Syst., 2012, pp. 1223–1231.

H. Zhang, C.-J. Hsieh, and V. Akella, “Hogwild++: A new mechanism for decentralized asynchronous stochastic gradient descent,” in Proc. IEEE 16th Int. Conf. Data Mining, 2016, pp. 629–638.

Y. Ming, Y. Zhao, C. Wu, K. Li, and J. Yin, “Distributed and asynchronous stochastic gradient descent with variance reduction,” Neurocomputing, vol. 281, pp. 27–36, 2018.

D. Alistarh, C. De Sa, and N. Konstantinov, “The convergence of stochastic gradient descent in asynchronous shared memory,” in Proc. ACM Symp. Princ. Distrib. Comput., 2018, pp. 169–178.

M. Teng and F. Wood, “Bayesian distributed stochastic gradient descent,” in Proc. Advances Neural Inf. Process. Syst., 2018, pp. 6378–6388.

L. Zhang, Z. Ye, K. Xiao, and B. Jin, “A parallel simulated annealing enhancement of the optimal-matching heuristic for ridesharing,” in Proc. IEEE Int. Conf. Data Mining, 2019, pp. 906–915.

Pengzhan Guo is currently working toward the PhD degree in the Department of Applied Mathematics and Statistics, Stony Brook University. His research interests include data mining, stochastic optimization, neural network, and parallel computing. He received the ICDM-2019 Student Travel Award.

Zeyang Ye received the PhD degree from the Department of Applied Mathematics & Statistics, Stony Brook University, in 2020. His research interests include route recommendations, data mining, and parallel computing. He has published papers in refereed journals and conference proceedings such as IEEE Transactions on Knowledge and Data Engineering, ACM SIGKDD Conference on Knowledge Discovery and Data Mining, etc. He has obtained many awards, e.g., KDD Best Paper Nominee, IACS Junior Researcher Award, etc.

Keli Xiao (Senior Member, IEEE) received the PhD degree from Rutgers University, in 2013. He is an associate professor with the College of Business, Stony Brook University. His research interests include business analytics, big data, and quantitative finance. He has published papers in refereed journals and conference proceedings, e.g., IEEE Transaction on Knowledge and Data Engineering, ACM Transactions on Knowledge Discovery from Data, ACM SIGKDD Conference on Knowledge Discovery and Data Mining, etc. He regularly serves as a SPC or PC member in prestigious conferences, e.g., IJCAI, KDD, AAAI, etc.

Wei Zhu received the PhD degree from the Department of Biostatistics, University of California at Los Angeles (UCLA), in 1996. She is a professor with the Department of Applied Mathematics and Statistics, College of Engineering and Applied Sciences, Stony Brook University. Her research interests include Bayesian statistics, cancer modeling, imaging analysis, machine learning, and quantitative finance. She has published papers in refereed journals and conference proceedings such as Nature, PNAS, Journal of the American Statistical Association (JASA), Human Brain Mapping, IEEE Transactions on Medical Imaging, International Journal of Computer Assisted Radiology and Surgery, Physica A, IEEE Information Visualization Symposium, and IEEE Nuclear Science Symposium.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.