# DaSGD: Squeezing SGD Parallelization Performance in Distributed Training Using Delayed Averaging

Qinggang Zhou[1*] qinggangz@gmail.com
Yawen Zhang[2*] zhywenwen@pku.edu.cn
Pengcheng Li[1] pengcheng.li@alibaba-inc.com
Xiaoyong Liu[1] xiaoyong.liu@alibaba-inc.com
Jun Yang[1] muzhuo.yj@alibaba-inc.com
Runsheng Wang[2] r.wang@pku.edu.cn
Ru Huang[2] ruhuang@pku.edu.cn

[1]*Alibaba Group, Sunnyvale, USA*
[2]*Peking University, Beijing, P.R. China*
[*] Both authors contributed equally to this work



***Abstract*—The state-of-the-art deep learning algorithms rely on distributed training systems to tackle the increasing sizes of models and training data sets. Minibatch stochastic gradient descent (SGD) algorithm requires workers to halt forward/back propagations, to wait for gradients aggregated from all workers, and to receive weight updates before the next batch of tasks. This synchronous execution model exposes the overheads of gradient/weight communication among the large number of workers a distributed training system. We propose a new SGD algorithm, DaSGD (Local SGD with Delayed Averaging), which parallelizes SGD and forward/back propagations to hide $100\%$ of the communication overhead. By adjusting the gradient update scheme, this algorithm uses hardware resources more efficiently and reduces the reliance on the low-latency and high-throughput inter-connects. The theoretical analysis and the experimental results show its convergence rate $O(1/\sqrt{K})$, the same as SGD. The performance evaluation demonstrates it enables a linear performance scale-up with the cluster size.**

***Index Terms*—stochastic gradient descent, local SGD, distributed training, parallelization**

## I. Introduction

Training deep learning models using data parallelism on a large-scale distributed cluster has become an effective method for deep learning model training. The enormous training data set allows a huge batch of training tasks on different data samples running in parallel. As a result, the training task can be scaled out to a massive number of servers (workers). The pinnacle of this method reduces the training time of the benchmark ResNet-50 from days to a couple of minutes. [1]–[5] However, during the Mini-batch stochastic gradient descent (SGD) at the end of a batch, these workers have to halt, wait for the computed gradients aggregated from all of the workers and receive a weight update before starting the next batch. The wait time tends to worsen when the number of workers increases. Additionally, as the workloads are spread over a larger cluser, the computation time are greatly shorten and the communication overheads take a larger portion of the overall cost.

System designers address this concern by improving inter-chip connects with higher throughput and lower latency and refining network topology [6], such as NVIDIA DGX-1 [7] and NVIDIA DGX-2 [8]. Additional care has been given to reduce the intermediate steps that would increase communication latency. These methods effectively reduce the wait time during Mini-batch SGD on a large-scale distributed system [9].

A modern data center design prefers selecting cost-efficient hardware blocks and choosing a balanced configuration for the typical workloads [10]. Under these workloads, various hardware resources would be utilized in a balanced fashion. A distributed training system works in the opposite manner. During the forward propagation and back propagation phases, the computing resources are throttled at the peak throughputs while the system inter-connects and switches are completely idle. During the SGD phase, the forward propagation and back propagation tasks of the next batch are blocked from starting. So, the computing resources are mostly idle while the system inter-connects and switches are throttled at the peak throughputs. Improving system efficency over the communication cost may be archieved from an orthogonal direction of improving system inter-connects. That is, the workloads may be restructured or re-designed for a balanced utilization of the system hardware resources.

Inspired by the modern system design practices, we propose a new SGD method called DaSGD, enabling SGD running parallelly with forward/back propagation. It replaces a Mini-batch SGD with Local SGD iterations to serialize forward/back propagations of different samples and to allow inter-worker weight averages may merge with local weights between Local SGD iterations. Model averaging may be scheduled to be delayed for a limited number of Local SGD iterations, which hides communication time on a large distributed cluster. Based on the network throughput and the data amount that training a model needs to tranfer, this algorithm may adjust the delay

amount. This algorithm makes better use of distributed training systems and reduces the reliance on low latency and high peak throughput communication hardware. The theoretical analysis clarifies its convergence rate is $O(1/\sqrt{K})$, the same as the traditional SGD. The auxiliary parameters are added to realize quantitative control, and their proper ranges and design guidelines are also provided in exprimental results. Finally, the system evaluation results show that this algorithm enables performance scale-up linearly with cluster size and is not restricted by communication.

The main contributions of this proposal are the followings.

- We present a new gradient aggregation algorithm for a large-scale deep learning training system, called DaSGD. This algorithm enables a more balanced and better utilized distributed training system.
- We provide the theoretical analysis of the algorithm's convergence rate. It shows the proposed algorithm converges at $O(1/\sqrt{K})$, the same as regular SGD.
- Our experiments show within the reasonable parameter ranges, this algorithm allows the training converges at the same rate of SGD. The experiments also explore the proper ranges of these parameters.
- A performance evaluation of real-life systems reflects the impacts from many specific design issues in the system hardware and software stacks. These include but not limited to the communication scheduling in software framework, the reduction algorithm, GPU interconnect topology and interfaces, server interconnect topology and interfaces. They introduce unneccessary complexity and are out of the scope of our discussion. Instead, we abstract an analytical model using the key performance parameters based on the system configuration and the training setup. We show the system evaluation demonstrates the method produces a linear scale of efficiency with the cluster size.
- A framework and further discussions are provided that guides how to use the method based on the system configuration and the training setup for best results.

The context of this paper is structured as follows: Session II describes the background of distributed training and the related work about SGD, Session III presents the design framework, the theoretical analysis of convergence rate and the discussion about the guidance scheme of DaSGD, Session IV shows the exprimental results, Session V provides the system evaluation results, Session VI discusses the training system design strategies, Session VII gives a conclusion.

## II. Background and Related Work

### A. Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is the backbone of numerous deep learning algorithms [11]. Supervised deep learning demands massive training datasets and super dense neural network architectures. Training a deep learning model needs many epochs for training to converge. A variant of classic SGD, synchronous mini-batch SGD [12], has become the mainstream, supported by prevalent machine learning frameworks, such as Tensorflow [13], Pytorch [14], MxNet [15].

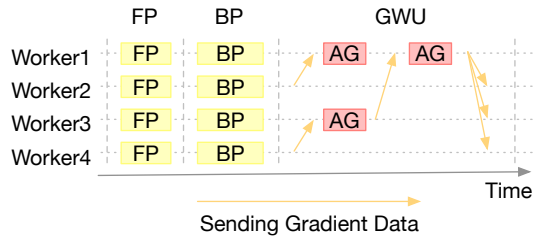


Fig. 1. The SGD timeline of multi-worker distributed training based on data parallelism. FP, BP, GWU, and AG represent forward propagation, backward propagation, global weight update, and gradient averaging.

It computes gradients from a batch of training samples, as shown in Eq. 1.

$$x_{k+1} = x_k - \frac{\eta}{B}\sum_{j=1}^{B}\nabla F(x_k, s_k^{(j)}) \tag{1}$$

where $x \in \mathbb{R}^d$ is the weight of model, $\eta$ is the learning rate, $B$ is the batch size, $\mathcal{S}$ is the training dataset, $s_k^{(j)} \subset \mathcal{S}$ is a random sample, $\nabla F(x_k, s_k^{(j)})$ is the stochastic gradient of the loss function of the sample $s_k^{(j)}$.

From a system perspective, a distributed training system may compute a batch of gradients on all workers. At the end of a batch, a reduction operation is performed on the gradients on a worker first and a worker sends out only a copy of local averaged gradients. Further reductions are performed on gradients from different workers until a final copy of the averaging gradients is obtained. The above equation may be rewritten as $x_{k+1} = x_k - \frac{\eta}{M}\sum_{j=1}^{B} g(x_k, s_k^{(j)})$. where $M$ is the number of workers, $g(x_k, s_k^{(j)})$ is the stochastic gradient that worker $j$ aggregates locally for that batch.$g(x_k, s_k^{(j)}) = \frac{M}{B}\sum_{i=1}^{\frac{B}{M}}\nabla(x_k, s_k^{(i)})$.

### B. Distributed Training Process based on SGD

The training of a neural network is an iterative process, and the *weight*s of a neural network layer need to be computed frequently. Each computation does the following phases sequentially: forward propagation, back propagation, gradient aggregation and (global) weight updating. First, the forward propagation performs a series of linear or nonlinear operations given the input data for every layer from the first to last. A layer's output is the input of the next layer. Then the observed output is compared with the expected and a loss value is calculated from the difference. Second, the backward propagation runs through from the last layer to the first by feeding the difference to the network and computes the gradient of the parameters. Last, we update the weights with the gradients based on SGD. These three stages are repeated many times during a training.

It would be extremely expensive for the computation to update the weights with a large-scale training set at one time. Bottou developed a *mini-batch SGD* [12] approach to solve the slow weight update process. A training data set contains a number of data samples. The mini-batch SGD shuffles all samples and groups them into mini-batches. It employs a

number of workers to work on these mini-batches in parallel. For a single mini-batch of samples, a worker performs forward propagation, backward propagation, and then computes the average gradient locally. Then global averaging is done and therefore the weights are updated per worker.

Distributed training is parallelized across a great number of workers. Fig. 1 shows a typical process of distributed training of a neural network. Each worker owns a copy of the network model and hence a copy of the weights. The initial weights for each worker are usually randomly generated. Afterwards, a mini-batch is sent to each worker in parallel (not shown in this figure). All workers execute forward propagation to compute loss and backward propagation to compute gradients, and aggregate the gradients of a mini-batch locally. Then, due to the gradients of each worker are different, the gradients are averaged across different workers, which is in the form of Tree All-Reduce [16] or Butterfly All-Reduce [17]. In Fig. 1, all gradients of different workers are averaged on worker 1. This process is divided into two steps: 1) the gradients of worker 3 and worker 4 are averaged to worker 3 and the gradients of worker 1 and worker 2 are averaged to worker 1; 2) the gradients of worker 1 and worker 3 are averaged to worker 1. Worker 1 updates the weights of model with these average gradients, and then broadcasts the updated model to all four workers again. Here, one iteration is over.

### C. Communication Efficient SGD Algorithms

*1) Gradient Compression and Sparsification:* Gradient sparsification [18]–[20] and gradient quantization [21] focus on compressing gradients with efficient data representation and redundant communication elimination. The default data format of gradients is single-precision floating-point 32. Gradient quantization maps gradients from a format with the regular precision format to a format with lower precision or fewer bits [22], sometimes to ternary [23] or binary [24], [25]. While quantizing gradients causes information loss, these works show that model converges with little accuracy loss. Deep Gradient Compression proposes momentum correction by accumulating quantization errors and using them at a later time. Gradient sparsification [18], [26], [27] explores that models are often over-parameterized and do not change all at once. Static or adaptive thresholds are used to determine significant gradients and are transferred for less communication bandwidth. These two groups of methods are orthogonal to our proposal.

*2) Asynchronous SGD (ASGD):* There are a few asynchronous training methods, such as Downpour SGD [28], Hogwild [29], Elastic Averaging SGD [30]. In these models, every worker has its own copy of weights. A worker performs forward propagation and back propagation on its own partition of samples, and then sends the calculated gradients asynchronously to a pool of parameter servers that manage a central copy of weights. The parameter servers update the central copy and then send the new weights asynchronously to each worker. While each worker communicates gradients at a different time and avoids congestions at worker inter-connects,

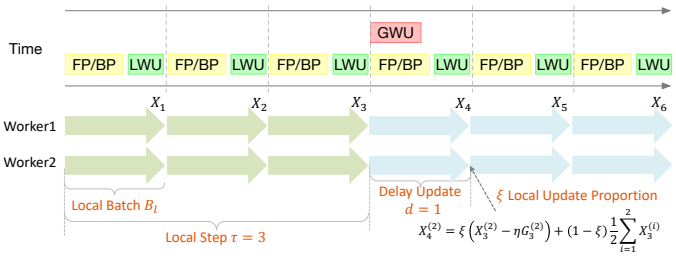


Fig. 2. Timing Diagram of DaSGD. Six iterations of two workers are shown. Each arrow represents a local update with a local batch $B_l$, which includes a forward/backward propagation (FP/BP) and a local weight update (LWU). The number of local step $\tau$ is 3, that is, after 3 local updates, the global weight update (GWU) will occur. The updated global coped model is not update directly in the current iteration, but is updated on each local coped model proportionally after $d$ local updates, where the update proportion of local model is set as the auxiliary update parameter $\xi$.

the parameter servers might be a performance bottleneck. For non-convex problems, ASGD requires that the staleness of gradients is bounded [31] to match the convergence rate $O(1/\sqrt{K})$ of synchronous SGD, where $K$ denotes the total Iteration steps.

*3) Local SGD:* Another set of methods targets at reducing the frequency of inter-worker communication and is called periodic averaging or Local SGD [32]–[34]. A worker performs SGD on its local copy of weights for $\tau$ times, where $\tau$ denotes the local iteration steps. After $\tau$ local updates, local copies are averaged across all workers globally in a synchronous manner. Several works suggested that Local SGD incurs the same convergence rate $O(1/\sqrt{K})$ as SGD [32], [33]. The total number of steps to train a model remains similar but the total amount of inter-worker communication is reduced by $\tau$ times. This has a similar effect as training with a large batch size, where the number of synchronizations decreases with an increase of batch size. However, a larger $\tau$ means more samples are processed on a single worker. With Local SGD, SGD and forward/back propagations are still blocking while system resources are unbalanced.

## III. DaSGD

In this paper, we propose a new algorithm, called *Local SGD with Delayed Averaging*, *DaSGD* for short. It aggregates gradients and updates weights in a relaxed manner, which helps parallelize the computation of forward/backward propagation with two other execution components: the execution of global weight averaging and inter-worker data communication.

Our algorithm was initially inspired by the Local SGD algorithm [32]–[34] (discussed in Section II). Although Local SGD was designed to reduce communication and synchronization overhead [12], [35], it still involves a significant amount of communication overhead. To further decrease communication overhead, even to zero, the proposed algorithm exploits a delayed averaging approach that makes two novel improvements based on Local SGD. First, in order to merge remote weights by other workers with local in a deterministic way, DaSGD serializes forward propagations and back propagations for different samples. Second, workers start with local computations for the next samples while waiting for the aggregation and

synchronization of global weights. In this way, the global communication and synchronization overhead is hidden or overlapped by local computations at the cost of a delayed update of local weights. However, theoretically we will prove that the convergence rate is the same as Mini-batch SGD. Furthermore, DaSGD parameterizes the overlapping degree so that when a large training cluster requires a longer time to synchronize, a worker may perform more iterations of local computations.

Fig. 2 illustrates the proposed algorithm by showing a wall-clock time diagram of 2 training epochs. There are 2 workers, dividing a global batch into 6 local batches. Each worker computes 3 local batches. Each local batch contains $d$ samples. Each worker maintains a local copy of model. According to Local SGD, for a local batch, each worker operates $d$ forward/backward propagations and then updates the weights of its local model. After 2 local updates, a worker synchronizes local weights with the other workers, resulting in an all-reduce operation being generated to average the model weights. For example in Fig. 2, all workers wait for, at local step 3, the global synchronization to be finished and then start to operate on the next local batch each, in the scenario of Local SGD.

DaSGD implements a key feature by imposing delay update on Local SGD. As shown in Fig. 2, a worker, at local step 3, broadcasts its local weights to the wild and then immediately starts to compute on next local batch, without waiting for the global synchronization to be finished. Later, at local step 4, the worker receives all the other workers' weights and then updates its local weights. This design very efficiently overlaps the communication of weights and forward/backward propagations of next local batch.

In DaSGD, we use $\tau$ to denote the number of local batches between two consecutive global synchronizations. Therefore, $\tau$ is a controlling parameter that quantifies the number of propagations between weight averaging globally. During the delay update, both local computation and the global communication of weights are executed in parallel. As long as communication time is no more than the computation time of $d$ local iterations, the communication time can be hidden in the overall model training time. Careful tuning of $d$ and $\tau$ can realize full parallelism of global averaging and local computations. Unlike Local SGD, $\tau$ does not have to be large, as it is not only used to reduce inter-worker communication overhead [18].

In the following part of this section, in order to compare the proposed algorithm and traditional SGDs, we start with the update framework of each algorithm, and then qualitatively analyze execution time. Finally, we discussed the updated rules and the convergence rate in detail.

### A. Update Flow of Different SGD

Fig. 3 explains the mechanisms of weight update flows of *Mini-batch SGD* [12], [35], *Local SGD* [32]–[34], and DaSGD by taking an example of a 2-worker parallel training process that sets the batch size as 2 samples. The 2 workers are distinguished by yellow and green arrows. In the Mini-batch

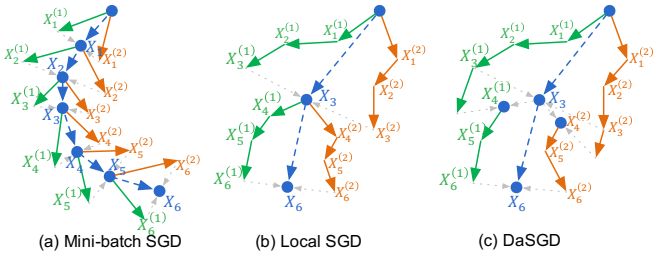


Fig. 3. Loss landscape of (a) Mini-batch SGD, (b) Local SGD, and (c) DaSGD. 12 samples are updated on two workers. The orange and green arrows represent the updated loss function of each sample, and the blue arrows describe the location of the updated loss function on the global model.

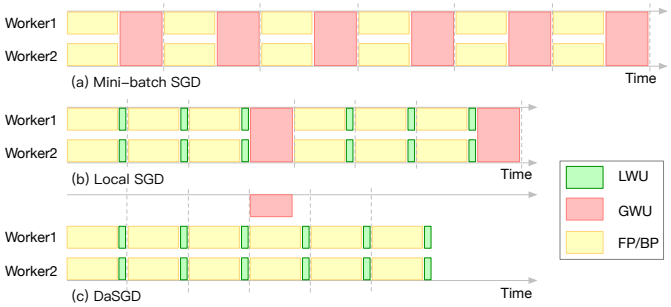


Fig. 4. Execution time diagrams of different SGDs.

SGD (as shown in Fig. 3(a)), every worker updates its local weights once every *mini-batch*, which is computed as the batch size divided by the number of workers. When both workers finish local updates for a mini-batch, local weights are merged to compute their average (shown by blue arrows). Next, both workers update their local weights with the average. Local SGD (shown in Fig. 3(b)) reduces the weight aggregation times by letting every worker first update weights locally for continuous $\tau$ *local batch*es in a row before a global merge is made. Local batch in the context of Local SGD is just a synonyms of the mini-batch in the context of Mini-batch SGD.

Same as the regular periodic averaging method (i.e., Local SGD), in the proposed algorithm, each worker updates local weights for $\tau$ local batches before a global aggregation. A novel change made by the proposed algorithm is to delay weight update from global to local after the global averaging. A worker may delay the update for $d$ *steps* (i.e., samples) of local weight updates ($d = 1$ in this example, as shown in Fig. 3(c)). With this novel algorithmic design, the time of global weight averaging can be hidden by parallelizing it with local computation by a worker, i.e., forward propagation, backward propagation, and local weight update. Large $d$ can be set if the time of global weight aggregation is very long in a large-scale distributed training system to shorten the overall training time.

### B. Execution Time

Before discussing the convergence rate, we first qualitatively analyze and compare the execution time between *Mini-batch SGD*, *Local SGD* and DaSGD. Figure 4 presents schematic diagrams of the three SGD algorithms for 6 iterations. For Mini-batch SGD, the weights are aggregated after every iteration, so the total execution time is measured by 6 communications and

6 local computations. By setting $\tau$ as 3, Local SGD reduces to 2 communications, with the total execution time measured by 2 communications and 6 computations. Expectedly, DaSGD performs the best by hiding communication time cost in the delayed weight update. As a result, the total execution time of DaSGD is measured by just 6 computations.

### C. Convergence Analysis

*1) Update rule:* The update rule of our algorithm is given by

$$x_{k+1}^{(m)} = \begin{cases} x_k^{(m)} - \eta g\left(x_k^{(m)}\right), & \text{otherwise} \\ \xi x_k^{(m)} - \eta\xi g\left(x_k^{(m)}\right) + \frac{(1-\xi)\sum_{j=1}^{M}\left[x_{k-d}^{(j)} - \eta g\left(x_{k-d}^{(j)}\right)\right]}{M}, & (k+1) \mod \tau = d \end{cases} \tag{2}$$

where $x_k^{(m)}$ is the weights of worker $m$ at $k$-th iteration, $\eta$ the learning rate, $M$ the number of workers, and $g(x_k^{(m)})$ the stochastic gradient of worker $m$. For every $k$ that satisfies $(k+1) \mod \tau = d$, a global average is updated to local weights. Besides, $\xi$ is an auxiliary parameter to adjust the weight of local weights in contrast to the global average when fusing them together.

We define the average weight and the average gradient

$$\mu_k = \frac{1}{M}\sum_{i=1}^{M} x_k^{(i)},\ \bar{g}_k = \frac{1}{M}\sum_{i=1}^{M} g\left(x_k^{(i)}\right).$$

After rearranging, the update rule for the average weight is obtained by

$$\mu_{\tau(k+1)+d} = \mu_{\tau k+d} - \eta\left[\xi\sum_{i=\tau-d}^{\tau-1}\bar{g}_{\tau k+d+i} + \sum_{i=0}^{\tau-1-d}\bar{g}_{\tau k+d+i}\right]$$

It is observed that the averaged weight $\mu_{\tau(k+1)+d}$ is performing a perturbed stochastic gradient descent. Thus, we will focus on the convergence of the averaged weight $\mu_{\tau(k+1)+d}$, which is common approach in the literature of distributed optimization [32], [33]. SGD can converge to a local minimum or saddle point due to the non-convex objective function $F(x)$. Therefore, the expected gradient norm is used as an index of convergence.

*2) Assumptions:* The common assumptions of the SGD analysis are defined as the following constraints [33]:

- Lipschitzian gradient: $||\bigtriangledown F(x) - \bigtriangledown F(y)|| \leq L||x-y||$
- Unbiased gradients: $E_{\mathcal{S}_k|x}\left[g(x)\right] = \bigtriangledown F(x)$
- Lower bounder: $F(x) \geq F_{inf}$
- Bounded variance: $E_{\mathcal{S}_k|x}||g(x) - \bigtriangledown F(x)||^2 \leq \beta||\bigtriangledown F(x)||^2 + \sigma^2$
- Independence: All random variables are independent to each other
- Bounded age: The delay is bounded, $d \leq \tau$

where $\mathcal{S}$ is the training dataset, $\mathcal{S}_k$ is set $\left\{s_k^{(1)}, ..., s_k^{(M)}\right\}$ of randomly sampled local batches, $L$ is the Lipschitz constant.

*3) Convergence Rate:* The learning rate is usually set as a constant and is decayed only whenthe training process is saturated. Therefore, we analyze the case of fixed learning rate and study the lower limit of error at convergence.

**Theorem (Convergence of DaSGD).** Under assumptions, if the learning rate satisfies $\eta \leq \min\left\{\sqrt{a}, \sqrt{b}\right\}$, where $a$ and $b$ and shown in Appendix. Then the average-squared gradient norm after $K$ iterations is bounded as follows

$$\begin{aligned} &\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\|\bigtriangledown F(\mu_k)\|^2\right] \\ \leq &\frac{2M\left[F(\mu_1) - F_{inf}\right] + 2MKL\eta^2\sigma^2\left[\xi^2 d + \tau - d\right]}{\eta MK(\xi d + \tau - d)} \\ &+ \frac{3\eta^4\xi L^2(\tau - d + d\xi)}{MK(\xi d + \tau - d)}\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1} g(\mathbf{X}_{d-1})\right\|_F^2 \\ &+ \frac{6\eta^4 L^2\sigma^2}{\xi d + \tau - d}\left(\frac{\tau\xi^2(\tau - d + \xi d)}{1-\xi^2} + (\tau - d)^2 + \xi d(\tau - 1)\right) \end{aligned}$$

where $\boldsymbol{X}_k = \left[x_k^1, ..., x_k^m\right]$, $\|\ \|_F^2$ is the Frobenius norm.All proofs are provided in the Appendix.

**Corollary.** Under sssumptions, if the learning rate is $\eta = A/\sqrt{K}$ the average-squared gradient norm after $K$ iterations is bounded by

$$\begin{aligned} &\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\|\bigtriangledown F(\mu_k)\|^2\right] \\ \leq &\frac{2M\left[F(\mu_1) - F_{inf}\right] + 2MLA^2\sigma^2\left[\xi^2 d + \tau - d\right]}{AM\sqrt{K}(\xi d + \tau - d)} \\ &+ \frac{3A^4\xi L^2(\tau - d + d\xi)}{MK^3(\xi d + \tau - d)}\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1} g(\mathbf{X}_{d-1})\right\|_F^2 \\ &+ \frac{6A^4 L^2\sigma^2}{K^2(\xi d + \tau - d)}\left[\frac{\tau\xi^2(\tau - d + \xi d)}{1-\xi^2} + (\tau - d)^2 + \xi d(\tau - 1)\right] \end{aligned}$$

If the total iterations $K$ is sufficiently large, then the average-squared gradient norm will be bounded by

$$\begin{aligned} &\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\|\bigtriangledown F(\mu_k)\|^2\right] \\ \leq &\frac{2M\left[F(\mu_1) - F_{inf}\right] + 2MLA^2\sigma^2\left[\xi^2 d + \tau - d\right]}{AM\sqrt{K}(\xi d + \tau - d)} \end{aligned}$$

Therefore, on non-convex objectives, the convergence rate of the proposed algorithm is consistent with the Mini-batch SGD and the Local SGD as $O(1/\sqrt{K})$.

### D. Guidelines for Using DaSGD

DaSGD is similar to Local SGD, the only difference is that the global model is updated to every local workers after $d$ local steps. The adjustment of other parameters is the same as that of Local SGD. Here we mainly discuss the setting of delay, which is the key of DaSGD. In order to realize the parallel communication and computation in DaSGD, the weight/gradient transfer time $t_c$ across workers is required to

TABLE I
ACCURACY OF DASGD, MINI-BATCH SGD AND LOCAL SGD ON CIFAR-10.

| Model | Accuracy after 50 epochs | | |
|---|---|---|---|
| | Mini-batch SGD | Local SGD | DaSDG |
| GoogleNet | 0.9409 | 0.9468 | 0.9444 |
| VGG-16 | 0.9264 | 0.9330 | 0.9343 |
| ResNet-50 | 0.9037 | 0.9062 | 0.9088 |
| ResNet-101 | 0.9019 | 0.9061 | 0.9045 |
| DenseNet-121 | 0.9332 | 0.9369 | 0.9357 |
| MobileNetV2 | 0.9304 | 0.9241 | 0.9304 |
| ResNeXt29 | 0.9403 | 0.9424 | 0.9415 |
| DPN-92 | 0.9354 | 0.9513 | 0.9502 |

be less than $d$ local iteration time, that is, $t_c < dt_p$, where $t_p$ is the computation time in one local update. For deep learning systems, the weight/gradient transfer time $t_c$ across multiple workers among multiple is approximately calculated as the number of parameters $n_p$ of neural network models multiplied by the number of workers $m$ divided by bandwidth BW of the device, $t_c = mn_p/\text{BW}$. The computation time $t_p$ in one local update is approximately calculated as the FLOP (floating-point operation) counts of the operation multiplied by local batch size divided by the computation speed FLOPS (floating-point operation per second) of the device, $t_p = B_l\text{FLOP}/\text{FLOPS}$. Therefore, the delay is given by

$$d > \frac{t_c}{t_p} = \frac{m \cdot n_p \cdot \text{FLOPS}}{B_l \cdot \text{BW} \cdot \text{FLOP}}. \tag{3}$$

It is worth noting that the delay is related to the structure of neural network models (the number of parameters and FLOP) and the configurations of deep learning systems (the local batch, the worker number, the bandwidth of the device and the computation speed). The current deep learning system has significantly improved the bandwidth and performance, and the discovery of residual network makes the growth of network parameters not obvious. So in most cases, when the delay is 1, the weight/gradient transfer can be processed completely in parallel with local updates. In addition, as the worker number increases, the increase of the worker number will lead to the increase of the number of the transferred weight/gradient increases, and the delay needs to be increased moderately. The cooperative design of various parameters in DaSGD and hardware is discussed in detail in the following sessions.

## IV. EXPERIMENTAL RESULTS

In this session, we will introduce our experimental settings and the convergence rate of DaSGD, Local SGD and Mini-batch SGD for different models. Then the influence of DaSGD parameters on the convergence rate in given.

### A. Parameter Setup

The training process is implemented under the Fast.Ai [36] platform based on CIFAR-10 dataset. The learning rate is adopted *One Cycle Policy* [37], which makes it linearly increase first (from $0.0001$ to $0.01$ in $30\%$ epochs) and then linearly decrease (from $0.01$ to $0.0001$ in $70\%$ epochs) within a reasonable range. A higher learning rate helps to prevent the model from falling in the steep area of the loss function, hoping to find a flatter minimum; A lower learning rate prevents training from diverging and converging to a local minimum. This learning schedule improves the accuracy in fewer iterations, allowing us to get more accurate results in only $50$ epochs. The weight decay is $0.01$ and the moment is $0.9$. Since we only want to analyze the convergence rate and accuracy, the comparison with the Local SGD and Mini-batch SGD is performed in $50$ epochs.

### B. Convergence Rate and Accuracy

Compared with the Mini-batch SGD and Local SGD in distributed training, we analyze the convergence rate and accuracy. TABLE I shows the accuracy of Mini-batch SGD, Local SGD and DaSGD after $50$ epochs based on CIFAR-10 dataset. It includes the existing common neural network models, such as GoogleNet [38], VGG-16 [39], ResNet-50 [40], ResNet-101, DenseNet-121 [41], MobileNetV2 [42], ResNeXt29 [43], and DPN-92 [44]. All models are trained under $32$ workers. The total batch size of Mini-batch SGD is $1024$. According to the data parallelism, the batch size distributed to each worker is $32$. The local batch size $B_l$ of Local SGD and DaSGD is $32$, the number of local steps $\tau$ is $4$, and the delayed iteration steps $d$ of DaSGD is $1$. For the three algorithms, the total number of iterations is the same under different models, which is $2450$.

As shown in TABLE I, we can find that for different models, with 1K batch size, the network model with higher accuracy can be obtained in a short iteration steps without adjusting the hyper-parameters. Due to the large batch size for each iteration of Mini-batch SGD, the hyper-parameters needs to be adjusted carefully. The optimization difficulty leads to the accuracy loss for large-batch training. Only the linear scale rule for adjusting the learning rate as a function of the total mini-batch size and the warm-up scheme are not enough. It is necessary to change the network structure, like adding batch normalization, for the high-accuracy training. These additional optimization methods for large-batch training are complex and tedious, and the algorithm based on local update overcomes this problem since the batch size of local updates is small. Thus, without any hyper-parameter adjustment for large-batch training, in addition to MobileNetV2, the accuracy of Local SGD and DaSGD is higher than that of the Mini-batch SGD. Fig. 5 shows this more clearly. At the beginning of distributed training, since the batch size is large, the algorithm based on Mini-batch SGD is usually very unstable, and the accuracy fluctuates greatly. The convergence rate is slower than that of the Local SGD and DaSGD. At the end of training, although the training loss of Mini-batch SGD is smaller, Local SGD and DaSGD has small test loss and higher accuracy.

### C. Parameter Influence of DaSGD

We evaluate the influence of different parameters on the convergence rate and accuracy in ResNet-50 model. Five adjustable parameters in DaSGD algorithm, which are the number of workers, the local batch size, the number of local step, the local update proportion and the delay, are discussed

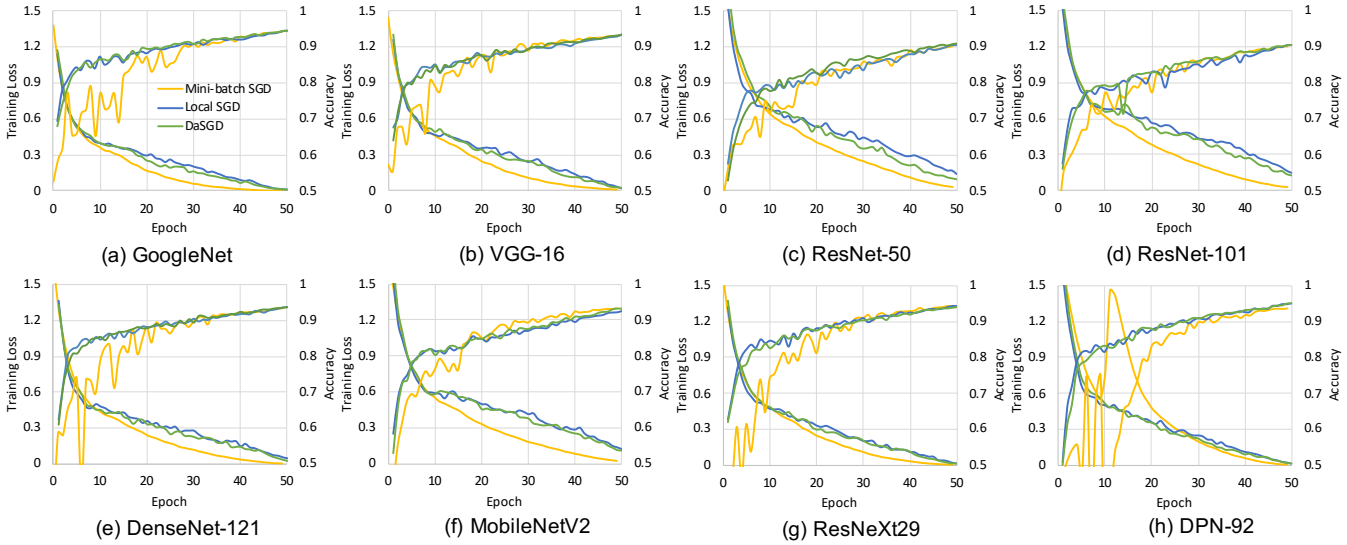


Fig. 5. Training loss and accuracy in different model based on CIFAR-10 dataset.

and analyzed respectively in Fig. 6. The baseline is set, where the number of workers $m$ is 32, the local step $\tau$ is 4, the delay is 2, the local batch $B_l$ is 32, the local update proportion $\xi$ is 0.25.

*1) Worker number:* Fig. 6(a) shows the accuracy of different worker numbers based on ResNet-50, illuminating that DaSGD has a fast convergence rate and high accuracy in general. As the number of workers increases from 2 to 256, the convergence rate slows down and the accuracy decreases. Since the local batch size remains unchanged as 32, when the worker number is 256, the total batch size has reached 8192, resulting in a decrease of accuracy of about 2% and a high training loss. In addition, DaSGD only communicates across workers every four local updates, and the samples of four local iterations has reached 32k, which is a huge batch for CIFAR-10 dataset with only 50000 training samples. The effect of the worker number on the distributed training is mainly reflected in that increasing the worker number can accelerate the training process, but the increase of the worker number leads to the linear increase of weight/gradient transmission, which increases the communication time and weaks the acceleration. Since forward/backward propagation and weight/gradient transfer are parallel, the increase in communication time caused by the increase of worker number is not reflected in the total execution time. However, in order to eliminate the increase of communication time driven by the increase of worker number in parallel, it is necessary to increase the delay update steps appropriately when the number of workers increases to a certain extent. This part is discussed in detail in Session VI. Through the analysis system model, we can evaluate the communication time under multiple workers and computation time of one local update, and determine the number of delay update steps to make the communication process completely parallel.

*2) Local batch size:* Fig. 6(b) illustrates that the DaSGD algorithm has a poor convergence rate for too large or too small local batch size. When the local batch is too large as 256, the accuracy is significantly reduced, and when the local batch is too small as 8, the convergence rate is slowed down. This phenomenon also exists in the Mini-batch SGD. Too large batch size leads to poor generalization ability, but it can reduce the total number of iterations; while too small batch size reduces the generalization error due to noise, but it requires a large number of iterations. Therefore, the selection of the local batch size is very important for DaSGD. Fig. 6(b) demonstrates that the local batch size of 32 or 64 has high accuracy and low training loss. It is worth noting that the total batch size is described as $B = mB_l$. When the worker number is 32 and the local batch size is 256, the total batch size rises to 32k, which is faced with the problem of adjusting hyper-parameter of large-batch training discussed above. The convergence rate of training needs more cooperation with the adjustment of hyper-parameters at such a high batch size.

*3) Local step:* When the number of local steps increases from 4 to 32, the accuracy of DaSGD decreases slightly and the training loss increases, as shown in Fig. 6(c). For DaSGD algorithm, the number of local steps should be reduced as much as possible under the condition of ensuring parallel communication, which is very different from Local SGD. By increasing the number of local steps, the Local SGD allocates time to several local iterations, resulting in a reduction in total execution time. In other words, increasing the number of local steps increases local iterations, which reduces the frequency of weight/gradient transfer across different workers. In order to reduce communication time, a large local step is required in Local SGD to share the communication time at the cost of accuracy loss. In addition, communication time is not essentially eliminated. Local SGD realizes the trade-off between communication time and accuracy by using local steps. While, DaSGD only uses the local step as a quantitative method to describe parallel communication. As long as the local step computation time is larger than the

weight/gradient communication time, the communication time can be eliminated in the total execution time. Therefore, the DaSGD algorithm requires a small number of local steps, which is conducive to convergence rate and high accuracy.

*4) Update proportion:* Fig. 6(d) shows that the different proportions of local weights in the delay update of global weights have little effect on accuracy. From the update rule (2), the local update proportion has the same meaning as the momentum in hyper-parameters. One cycle policy in Fast.Ai has shown that different momentum has little effect on accuracy.

*5) Delay:* The difference between DaSGD and Local SGD is that DaSGD delays the average model of every $\tau$ local steps by $d$ local update steps. The number of delay is closely related to the number of local steps. Fig. 6(e) and (c) shows the two relationships between the number of delay update and the number of local steps, in which one is to keep the number of local steps and change the delay and the other is to keep the number of delay and change the local steps. In addition, the delay update is also limited by the local step. It is assumed that it is smaller than the local step, that is, the global model update of the current iteration must be completed before the next global update. Delay has little effect on the convergence rate in general. When the delay increases from 0 to 7, the convergence rate slows down and the accuracy decreases, as shown in Fig. 6(e). The Local SGD is shown as the delay is 0, so the accuracy of DaSGD is slightly lower than that of Local SGD in the same local steps. Besides, a large delay update is usually not implemented. Since the weight/gradient transfer time is relatively small compared to the forward/backward propagation time of the local iteration, 1 delay update can eliminate the weight/gradient communication time in the total execution time in most cases. Due to the increase in the worker number, the time of the weight/gradient transfer across workers may be longer than the forward/backward propagation time of local iterations. In this case, the number of delay update can be appropriately increased to eliminate communication time, which is also discussed in the influence of the worker number part.

## V. System Performance Evaluation

### A. Analytical Model of Distributed Training Performance

We analyze the performance of the distributed system under different SGD algorithms and show the analytical model. The performance of real-life systems are affected by many issues in the system hardware and software stacks, such as whether the software framework overlaps communication and computation, what reduction algorithm is used, how GPUs interconnects, how much network throughputs the servers have. The differentiation of an algorithm may be obscured by these issues. We abstract an analytical model with the following key performance parameters based on the system configuration and the training setup. The total execution time $t_{total}$ of distributed training is decomposed into forward propagation time for a single sample $t_f$, backward propagation time for a single sample $t_b$, the time for gradient aggregation and weight update on the same worker $t_l$, the time for gradient aggregation and weight update among multiple workers that are not hidden behind computation time (communication time) $t_c$. The total amount of training data in a dataset is defined as $n_s$, the number of samples a worker computes parallel is defined as $p$, and the number of workers is defined as $m$.

*1) Mini-batch SGD:* We formulate the training process using Mini-batch SGD into three steps. (a). Forward/backward propagation. Forward propagation is performed layer by layer in each worker, and the gradient is generated using the chain rule to realize the backward propagation. The samples in a mini-batch are divided to $m$ worker and are processed in $p$-parallel. Each worker performs $B/pm$ times forward/backward propagations in a mini-batch. (b). Local gradient accumulation on each local worker. We assume the framework is optimized and gradients from samples are accumualted locally at the worker first before being synchronized among workers. Each worker perform local gradient accumulation for $t_l$ in a mini-batch. (c). Gradient aggregation and weight update among all workers. Each worker needs $t_c$ for this in a mini-batch. Therefore, the total time of training a model $t_{total}$ is described as

$$t_{total} = \left[\frac{B}{pm}\left(t_f + t_b\right) + t_l + t_c\right]\frac{n_s}{B}. \tag{4}$$

*2) Local SGD:* Local SGD needs to complete $\tau$ local updates before global model averaging. We formulate it into the the following steps: $\tau$ (a). local updates: each worker completes $\tau B/pm$ forward and back propagation. (b). $\tau$ local SGD aggregation and local weight update. (c). Update model every $\tau$ local updates using the average local model between all workers. The total execution time is represented as

$$t_{total} = \left[\frac{B}{pm}\left(t_f + t_b\right) + t_l + \frac{t_c}{\tau}\right]\frac{n_s}{B}. \tag{5}$$

The above formula also proves that the difference between Local SGD and Mini-batch SGD is that the communication time $t_c$ in Mini-batch SGD is reduced $\tau$ times to $t_c/\tau$. In order to reduce the weight/gradient transfer time effectively, a large $\tau$ value is usually required.

*3) DaSGD:* By delaying the local update of the global model, DaSGD algorithm realizes the local update computation and weight/gradient communication in parallel. The process is similar to Local SGD involving three steps: (a). $\tau$ local updates: each worker completes $\tau B/pm$ layer-by-layer feed forward calculation and gradient back propagation. (b). $\tau$ local SGD calculations include weight aggregation and weight apply. (c). The averaging model is updated after $d$ local iterations every $\tau$ local updates . The third step will not be reflected in the total execution time, since it can be performed in parallel with the previous two processes. When $t_c < d\left[B\left(t_f + t_b\right)/pm + t_l\right]$, it means that the weight/gradient transfer time is shorter than that of $d$ local iterations, and the total execution time is showed as

$$t_{total} = \left[\frac{B}{pm}\left(t_f + t_b\right) + t_l\right]\frac{n_s}{B}. \tag{6}$$

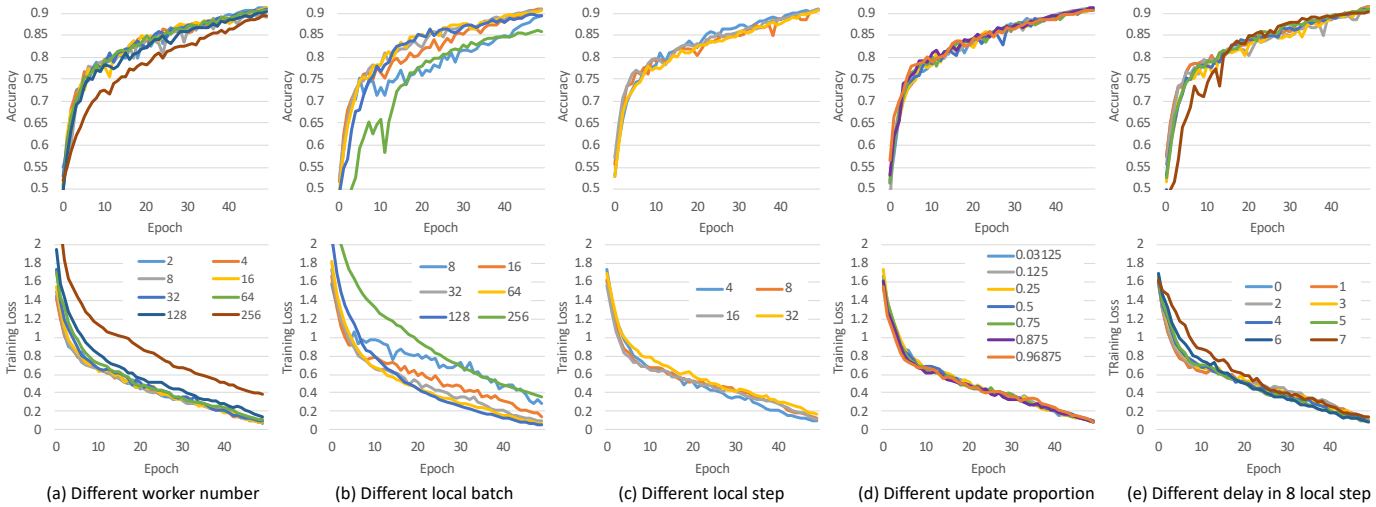


Fig. 6. Effect of different parameters of DaSGD based on ResNet-50.

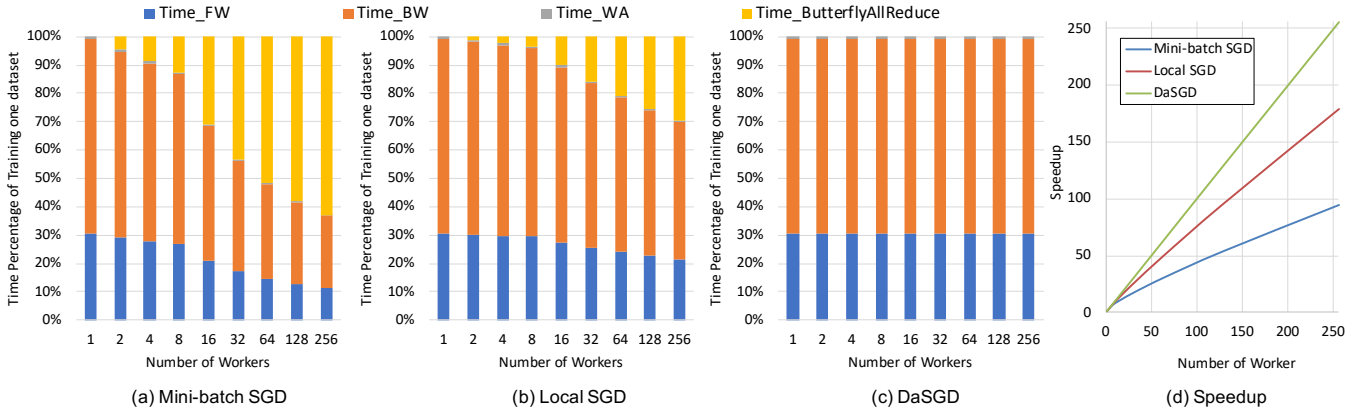


Fig. 7. PALEO analytical results of data parallel distributed training of ResNet-50 with up to 256 servers on the NVIDIA TITAN X GPUs. A comparison of (a) Mini-batch SGD, (b) Local SGD, and (c) DaSGD is shown. (d) Weak scaling speedup results based on the Butterfly AllReduce communication scheme.

It is worth noting that compared with Mini-batch SGD and Local SGD, DaSGD completely eliminates communication time by parallelizing processing mode. The training process can be accelerated only by changing the algorithm without any special requirements for the deep learning system

### *B. Performance Simulation*

In order to effectively evaluate the performance improvement of DaSGD to the distributed deep learning system for a given problem instance, we use the PALEO, a DNN performance model, which provides performance estimations within $10\%$–$30\%$ prediction errors [45]. We analyze the distributed training of ResNet-50, which has $25.5$ million parameters and occupies $102$ MB of memory. The experiments simulate a distributed training cluster with a less optimal configuration. It consists of up to $256$ the NVIDIA TITAN X GPUs with PCIe3.0. An enhancement is made on the Paleo to simulate the case each server uses PCIe3.0 ($16$ GBps) connecting $8$ GPUs with $20$ Gbps Ethernet between servers. The Butterfly AllReduce scheme is adopted for gradient aggregation.

PALEO decomposes the total execution time $t_{total}$ into computation time and communication time with all layers included, which includes forward propagation time $t_f$, backward propagation time $t_b$, the time for gradient aggregation and weight update within a single worker $t_l$, and the time for gradient aggregation and weight update between workers (communication time) $t_c$. Fig. 7 shows a comparison of three algorithms for training the ResNet-50 with a mini-batch size $64$ on up to $256$ workers under weak scaling, where weak scaling means that the global batch size is increasing as increasing the number of workers. The execution time breakdown for various workloads of processing the whole dataset (one epoch) is shown in Fig. 7.

*1) Gradient aggregation and weight update within a single worker:* The time for gradient aggregation and weight update within a single worker can be ignored, since it comprises a small percentage of the total training time, as shown in Fig. 7(a) and (b) (grey block).

*2) Forward/backward propagation:* The forward/backward propagations time of the three algorithms are the same (Fig. 7(a), (b) and (c)), since the batch size of each worker in each iteration is $64$. In addition, forward/backward propagations are compute-bound computations. Based on the PALEO performance model, the forward/backward propagation time

of each layer of neural network is calculated in detail.

*3) Gradient aggregation and weight update between workers:* In the Mini-batch SGD algorithm (Fig. 7(a)), when the number of workers is $256$, the gradient aggregation and weight update between workers contributes approximately $45.9\%$ of the total execution time. It means that the larger the number of workers is, the more vulnerable the Mini-batch SGD algorithm is to be affected by the communication bottleneck. When evaluating this overhead in Local SGD, it is reduced to $17.5\%$. The communication time of Local SGD is four times shorter than that of the Mini-batch SGD when the number of local update steps is $4$. The speedup of different algorithms with respect ot the number of workers is shown in Fig. 7(d). A large number of workers does not scale at the linear rate of $1$, and the scaling rate is less for Mini-batch SGD, due to a larger proportion of time spent on gradient transfer. In Mini-batch SGD, although increasing the number of workers can shorten the amount of computation time, it increases the total amount of data communication. The communication overhead increases linearly with the larger number of workers. Considering the high communication overhead for a large distributed training cluster, DaSGD parallels communication tasks with computation tasks and removes the weight/gradient transfer time from the total execution time. As shown in Fig. 7(c), when DaSGD is applied, the time for gradient aggregation and weight update is mainly communication overhead and is completely hidden behind forward/back propagations.

## VI. Training System Design Strategies and Discussion

### A. Use the Hyper-Parameter Receipt for a Large Global Batch

When the local batch size is $32$ and the worker number is $256$, the global batch size attains $8192$. With regular Mini-batch SGD, training with a large batch size from $8$k to $64$k requires a specific set of hyper-parameter receipts. DaSGD needs no additional hyper-parameter adjustment to achieve high accuracy, but if it wants to achieve higher accuracy, it needs to further optimize the hyper-parameter receipt for large-batch training. This receipt is not used in our experiments.

### B. Select the Local Batch Size, Such as 32, 64

Local batch size $32$ is a common practice for reasonable batch normalization results. Local batch size affects accelerator performance. The sweet spot for a single GPU is at $128$ and $256$. But for better parallelism on a large cluster, $32$ or $64$ is recommended.

### C. Using System Analysis Model to Determine Delay

DaSGD algorithm parallels the communication process with $d$-local updating, and requires that the time of weight/gradient transfer between workers is less than $d$-local update time, that is, $t_c < dt_p$. Weight/gradient transfer time can be calculated through the neural network model structure and the hardware parameters of the deep learning system, such as the network interconnection bandwidth, the number of workers, and the communication mode (Tree All-Reduce and Butterfly All-Reduce). While, the local update time included forward/backward propagation and weight aggregation/apply is determined by the number of parameters of neural network model and the peak FLOPS of deep learning system. TABLE II analyzes the parameter number, the computation time of one local update $t_p$ and the weight/gradient communication time $t_c$ under the NVIDIA TITAN X GPUs system connected to $20$ Gbps Ethernet network and the GPU K80 system connected to a $10$ Gbps Ethernet network. For the deep learning system with high network interconnection bandwidth, the communication time is small. Even when the worker number is up to $256$ using the Tree AllReduce, the weight/gradient transfer can be completely parallel if the delay is $1$ or $2$. On the contrary, it is high in the Ethernet network with $10$ Gbps bandwidth. In addition, Butterfly AllReduce optimizes that half of the nodes in Tree AllReduce do not send at the halving stage, and its communication time is about twice that of the Tree AllReduce. However, for large data block transferring, Butterfly AllReduce is prone to the communication time fluctuation caused by the insufficient utilization of bandwidth.

### D. Set Local Steps to Delay Plus One for High Accuracy

The DaSGD algorithm requires that the global averaging model is updated after $d$ local steps. When it is updated to the local worker, the global averaging model is a stale calculation result. Updating the global averaging model to the local workers can effectively reduce the randomness between different local models, but the global model returned in the older version causes a slower convergence rate . Therefore, we optimize the trade-off between the randomness in the local model and the staleness in the global model. In other words, we improve the trade-off between the number of local updates and the delayed update. The delay is obtained according to the model structure and the distributed system. Since the increase of the local steps reduces the accuracy, the number of local steps is set as the number of delay steps plus $1$, as shown in TABLE II, that is, $\tau = d + 1$ to obtain higher accuracy and fast convergence rate. This means that the local step cannot be too long so that both the staleness of the delayed update and the randomness of the local models can be reduced.

## VII. Conclusion

In this work, we propose a new SGD algorithm called DaSGD, which parallelizes SGD and forward/back propagation to hide communication time. Just adjusting the update schedule at the software level, DaSGD algorithm makes better use of distributed training systems and reduces the reliance on low latency and high peak throughput communication hardware. Theoretical analysis and experimental results clarify that its convergence rate is $O\left(1/\sqrt{K}\right)$, which is the same as the mini-batch SGD. The auxiliary parameters are added to realize quantitative control, and their proper ranges and guidelines for using DaSGD are also provided. The system evaluation demonstrates that DaSGD can speed up the deep learning system linearly without being weakened by high communication time.

TABLE II
PARAMETERS AND TIME @256 WORKERS, 64 LOCAL BATCH

| Model | Parameters | TITAN X with 20 Gbps Ethernet | | | | | K80 with 10 Gbps Ethernet | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $t_p$ | $t_c$ AllReduce | | delay | $\tau$ | $t_p$ | $t_c$ AllReduce | | delay | $\tau$ |
| | | | Tree | Butterfly | | | | Tree | Butterfly | | |
| Network-in-Network | 7,595,176 | 119.08 | 132.91 | 66.45 | 2 | 3 | 129.80 | 254.43 | 127.21 | 2 | 3 |
| VGG-16 | 138,357,544 | 2164.32 | 2421.25 | 1210.62 | 2 | 3 | 2361.61 | 4634.97 | 2317.48 | 2 | 3 |
| VGG-19 | 143,667,240 | 2684.73 | 2514.17 | 1257.08 | 1 | 2 | 2932.49 | 4812.85 | 2406.42 | 2 | 3 |
| ResNet-50 | 25,530,472 | 526.05 | 446.78 | 223.39 | 1 | 2 | 575.29 | 855.27 | 427.63 | 2 | 3 |
| ResNeXt-50 | 167,153,128 | 1640.05 | 2925.17 | 1462.58 | 2 | 3 | 1795.83 | 5599.62 | 2799.81 | 4 | 5 |
| DenseNet-121 | 7,905,448 | 358.23 | 138.34 | 69.17 | 1 | 2 | 390.73 | 264.83 | 132.41 | 1 | 2 |
| DenseNet-201 | 17,900,106 | 538.06 | 313.25 | 156.62 | 1 | 2 | 587.64 | 599.65 | 299.82 | 2 | 3 |

## REFERENCES


[1] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[2] Y. You, Z. Zhang, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Imagenet training in 24 minutes," *arXiv preprint arXiv:1709.05011*, 2017.

[3] T. Akiba, S. Suzuki, and K. Fukuda, "Extremely large minibatch sgd: Training resnet-50 on imagenet in 15 minutes," *arXiv preprint arXiv:1711.04325*, 2017.

[4] Y. You, I. Gitman, and B. Ginsburg, "Scaling sgd batch size to 32k for imagenet training," *arXiv preprint arXiv:1708.03888*, vol. 6, 2017.

[5] C. Ying, S. Kumar, D. Chen, T. Wang, and Y. Cheng, "Image classification at supercomputer scale," *arXiv preprint arXiv:1811.06992*, 2018.

[6] A. Li, S. L. Song, J. Chen, J. Li, X. Liu, N. R. Tallent, and K. J. Barker, "Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 1, pp. 94–110, 2019.

[7] "Nvidia dgx-1 with tesla v100 system architecture," http://images.nvidia.com/content/pdf/dgx1-v100-system-architecture-whitepaper.pdf, 2017.

[8] "Nvidia nvswitch," https://images.nvidia.com/content/pdf/nvswitch-technical-overview.pdf, April 2018.

[9] "Gaudi™ training platform white paper," https://habana.ai/wp-content/uploads/2019/06/Habana-Gaudi-Training-Platform-whitepaper.pdf, 2019.

[10] L. A. Barroso, U. Hölzle, and P. Ranganathan, "The datacenter as a computer: Designing warehouse-scale machines," *Synthesis Lectures on Computer Architecture*, vol. 13, no. 3, pp. i–189, 2018.

[11] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization 23(4):2341–2368*, 2013.

[12] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems 32:8024-8035*, 2019.

[15] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:11512.01274*, 2015.

[16] A. Agarwal, O. Chapelle, M. Dudík, and J. Langford, "A reliable effective terascale linear learning system," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1111–1133, 2014.

[17] P. Patarasuk and X. Yuan, "Bandwidth efficient all-reduce operation on tree topologies," in *2007 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2007, pp. 1–8.

[18] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[19] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 1299–1309.

[20] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Advances in Neural Information Processing Systems*, 2018, pp. 5973–5983.

[21] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1709–1720.

[22] X. Jia, S. Song, W. He, Y. Wang, H. Rong, F. Zhou, L. Xie, Z. Guo, Y. Yang, L. Yu *et al.*, "Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes," *arXiv preprint arXiv:1807.11205*, 2018.

[23] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in neural information processing systems*, 2017, pp. 1509–1519.

[24] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[25] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[26] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017.

[27] C. Renggli, S. Ashkboos, M. Aghagolzadeh, D. Alistarh, and T. Hoefler, "Sparcml: High-performance sparse communication for machine learning," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–15.

[28] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[29] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in neural information processing systems*, 2011, pp. 693–701.

[30] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging sgd," in *Advances in neural information processing systems*, 2015, pp. 685–693.

[31] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Advances in Neural Information Processing Systems*, 2015, pp. 2737–2745.

[32] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd," *arXiv preprint arXiv:1810.08313*, 2018.

[33] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms," *arXiv preprint arXiv:1808.07576*, 2018.

[34] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local sgd," *arXiv preprint arXiv:1808.07217*, 2018.

[35] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal

distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 165–202, 2012.

[36] J. Howard, "Now anyone can train imagenet in 18 minutes," https://www.fast.ai/2018/08/10/fastai-diu-imagenet/, August 2018.

[37] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 464–472.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: http://arxiv.org/abs/1409.4842

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[41] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[43] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5987–5995.

[44] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4467–4475. [Online]. Available: http://papers.nips.cc/paper/7033-dual-path-networks.pdf

[45] H. Qi, E. R. Sparks, and A. Talwalkar, "Paleo: A performance model for deep neural networks," in *Proceedings of the International Conference on Learning Representations*, 2017.

## APPENDIX

## CONVERGENCE ANALYSIS OF DASGD

### A. Assumptions

We define some notations. $\mathcal{S}$ is the training dataset, $\mathcal{S}_k$ is set $\left\{s_k^{(1)}, ..., s_k^{(M)}\right\}$ of randomly sampled local batches at $M$ workers in $k$ iteration, $L$ is the Lipschitz constant, $d$ is the number of local iteration that global weight updates are delayed, $\tau$ is the number of local steps, $x$ is the weight of devices. The convergence analysis is conducted under the following assumptions:

- Lipschitzian gradient: $|| \bigtriangledown F(x) - \bigtriangledown F(y)|| \leq L||x - y||$
- Unbiased gradients: $E_{\mathcal{S}_k|x}\left[g(x)\right] = \bigtriangledown F(x)$
- Lower bounder: $F(x) \geq F_{inf}$
- Bounded variance in local SGD: $E_{\mathcal{S}_k|x}||g(x) - \bigtriangledown F(x)||^2 \leq \beta|| \bigtriangledown F(x)||^2 + \sigma^2$
- Independence: All random variables are independent to each other
- Bounded age: The delay is bounded, $d \leq \tau$

### B. Update Rule

The update rule of DaSGD is given by

$$x_{k+1}^{(m)} = \begin{cases} x_k^{(m)} - \eta g\left(x_k^{(m)}\right), & \text{otherwise} \\ \xi x_k^{(m)} - \eta\xi g\left(x_k^{(m)}\right) + \frac{1-\xi}{M}\sum\limits_{j=1}^{M}\left[x_{k-d}^{(j)} - \eta g\left(x_{k-d}^{(j)}\right)\right], & (k+1-d) \mod \tau = 0 \end{cases}$$

where $x_k^{(m)}$ is the weights at $m$ worker in $k$ iteration, $\eta$ is the learning rate, $M$ is the number of workers, $g(x_k^{(m)})$ is the stochastic gradient of worker $m$, $\xi$ is the local update proportion, delayed update is the case $(k+1-d) \mod \tau = 0$.

**Matrix Representation.** Define matrices $\boldsymbol{X}_k$, $\boldsymbol{G}_k \in \mathbb{R}^{d\times M}$ that concatenate all local models and gradients in $k$ iteration:

$$\boldsymbol{X}_k = \left[x_k^1, ..., x_k^m\right],\ \boldsymbol{G}_k = \left[g\left(x_k^{(1)}\right), ..., g\left(x_k^{(m)}\right)\right]$$

Then, the update rule is

$$\boldsymbol{X}_{k+1} = \begin{cases} \xi\left(\boldsymbol{X}_k - \eta\boldsymbol{G}_k\right) + (1-\xi)\left(\boldsymbol{X}_{k-d} - \eta\boldsymbol{G}_{k-d}\right)\boldsymbol{J}, & (k+1-d) \mod \tau = 0 \\ \boldsymbol{X}_k - \eta\boldsymbol{G}_k, & \text{otherwise} \end{cases} \tag{7}$$

**Update Rule for the Averaged Model.** The update rule of DaSGD is given by

$$x_{k+1}^{(m)} = \begin{cases} x_k^{(m)} - \eta g\left(x_k^{(m)}\right), & \text{otherwise} \\ \xi x_k^{(m)} - \eta\xi g\left(x_k^{(m)}\right) + \frac{1-\xi}{M}\sum\limits_{j=1}^{M}\left[x_{k-d}^{(j)} - \eta g\left(x_{k-d}^{(j)}\right)\right], & (k+1-d) \mod \tau = 0 \end{cases}$$

Here, we set

$$\bar{x}_k = \frac{1}{M}\sum_{i=1}^{M} x_k^{(i)},\ \bar{g}_k = \frac{1}{M}\sum_{i=1}^{M} g\left(x_k^{(i)}\right)$$

The average weight on different workers is obtained by

$$\bar{x}_{k+1} = \begin{cases} \bar{x}_k - \eta\bar{g}_k, & \text{otherwise} \\ \xi\bar{x}_k + (1-\xi)\bar{x}_{k-d} - \eta\xi\bar{g}_k - \eta(1-\xi)\bar{g}(x_{k-d}), & (k+1-d) \mod \tau = 0 \end{cases}$$

When $z = \tau(k+1)$ for $z \mod \tau = 0$, we have

$$\begin{aligned} \bar{x}_{\tau(k+1)+d} =& \ \xi\bar{x}_{\tau(k+1)+d-1} + (1-\xi)\bar{x}_{\tau(k+1)-1} - \xi\eta\bar{g}_{\tau(k+1)+d-1} - (1-\xi)\eta\bar{g}_{\tau(k+1)-1} \\ =& \ \xi\bar{x}_{\tau k+d} + (1-\xi)\bar{x}_{\tau k+d} - \xi\eta\sum_{i=0}^{\tau-1}\bar{g}_{\tau k+d+i} - (1-\xi)\eta\sum_{i=0}^{\tau-1-d}\bar{g}_{\tau k+d+i} \\ =& \ \bar{x}_{\tau k+d} - \eta\left[\xi\left(\sum_{i=0}^{\tau-1}\bar{g}_{\tau k+d+i} - \sum_{i=0}^{\tau-1-d}\bar{g}_{\tau k+d+i}\right) + \sum_{i=0}^{\tau-1-d}\bar{g}_{\tau k+d+i}\right] \\ =& \ \bar{x}_{\tau k+d} - \eta\left[\xi\sum_{i=\tau-d}^{\tau-1}\bar{g}_{\tau k+d+i} + \sum_{i=0}^{\tau-1-d}\bar{g}_{\tau k+d+i}\right] \end{aligned}$$

If we set $K(k) = \tau k + d$

$$\bar{x}_{K(k+1)} = \bar{x}_{K(k)} - \eta \left[ \xi \sum_{i=\tau-d}^{\tau-1} \bar{g}_{K(k)+i} + \sum_{i=0}^{\tau-1-d} \bar{g}_{K(k)+i} \right]$$

For the ease of writing, we first define some notations. Let $\mathcal{S}_k$ denote the set $\left\{ s_k^{(1)}, ..., s_k^{(m)} \right\}$ of mini-batches at $m$ workers in iteration $k$. Besides, define averaged stochastic gradient and averaged full batch gradient as follows:

$$\mathcal{G}_{K(k)} = \frac{1}{M} \sum_{m=1}^{M} \left[ \sum_{i=\tau-d}^{\tau-1} \xi g\left( x_{\tau k+d+i}^{(m)} \right) + \sum_{i=0}^{\tau-1-d} g\left( x_{\tau k+d+i}^{(m)} \right) \right] \tag{8}$$

$$\mathcal{H}_{K(k)} = \frac{1}{M} \sum_{m=1}^{M} \left[ \sum_{i=\tau-d}^{\tau-1} \xi \bigtriangledown F\left( x_{\tau k+d+i}^{(m)} \right) + \sum_{i=0}^{\tau-1-d} \bigtriangledown F\left( x_{\tau k+d+i}^{(m)} \right) \right] \tag{9}$$

$$\mu_{K(k)} = \frac{1}{M} \sum_{i=1}^{M} x_{\tau k+d}^{(i)} \tag{10}$$

Then we have

$$\mu_{K(k+1)} = \mu_{K(k)} - \eta \mathcal{G}_{K(k)}$$

### C. Convergence Rate

**Theorem (Convergence of DaSGD).** Under assumptions, if the learning rate satisfies

$$\eta \leq \min\left\{ \sqrt{a}, \sqrt{b} \right\}$$

where $a = 1/\left\{ 2L\xi^2(\beta+1)(1-\xi) + 6L^2(d\xi+\tau-d)[(\beta+k\tau)+(\beta+1)(1-\xi)] \right\}$, $b = \xi M(1-\xi)/\left\{ 2L\xi^2(\beta+1)(1-\xi) + 3L^2M(\tau-d)(2\beta+2k\tau) + 6dM\xi L^2[(\beta+k\tau)+(\beta+1)(1-\xi)] \right\}$. Then the average-squared gradient norm after $K$ iterations is bounded as

$$\begin{aligned}
&\mathbb{E}\left[ \frac{1}{K} \sum_{k=1}^{K} \left\| \bigtriangledown F(\mu_k) \right\|^2 \right] \\
\leq &\frac{2M\left[ F(\mu_1) - F_{inf} \right] + 2MKL\eta^2\sigma^2\left[ \xi^2 d + \tau - d \right]}{\eta MK(\xi d + \tau - d)} + \frac{3\eta^4 \xi L^2(\tau - d + d\xi)}{MK(\xi d + \tau - d)} \frac{\xi^2}{1-\xi^2} \left\| \sum_{i=1}^{d-1} g(\mathbf{X}_{d-1}) \right\|_F^2 \\
&+ \frac{6\eta^4 L^2 \sigma^2}{\xi d + \tau - d} \left( \tau \frac{\xi^2}{1-\xi^2}(\tau - d + \xi d) + (\tau - d)^2 + \xi d(\tau - 1) \right)
\end{aligned}$$

where $\mu_k = \frac{1}{M} \sum_{i=1}^{M} x_{\tau k+d}^{(i)}$, $\| \ \|_F^2$ is the Frobenius norm.

**Corollary.** Under sssumptions, if the learning rate is $\eta = A/\sqrt{K}$ the average-squared gradient norm after $K$ iterations is bounded by

$$\begin{aligned}
&\mathbb{E}\left[ \frac{1}{K} \sum_{k=1}^{K} \left\| \bigtriangledown F(\mu_k) \right\|^2 \right] \\
\leq &\frac{2M\left[ F(\mu_1) - F_{inf} \right] + 2MLA^2\sigma^2\left[ \xi^2 d + \tau - d \right]}{AM\sqrt{K}(\xi d + \tau - d)} + \frac{3A^4 \xi L^2(\tau - d + d\xi)}{MK^3(\xi d + \tau - d)} \frac{\xi^2}{1-\xi^2} \left\| \sum_{i=1}^{d-1} g(\mathbf{X}_{d-1}) \right\|_F^2 \\
&+ \frac{6A^4 L^2 \sigma^2}{K^2(\xi d + \tau - d)} \left( \tau \frac{\xi^2}{1-\xi^2}(\tau - d + \xi d) + (\tau - d)^2 + \xi d(\tau - 1) \right).
\end{aligned}$$

If the total iterations $K$ is sufficiently large, then the average-squared gradient norm is bounded by

$$\mathbb{E}\left[ \frac{1}{K} \sum_{k=1}^{K} \left\| \bigtriangledown F(\mu_k) \right\|^2 \right] \leq \frac{2M\left[ F(\mu_1) - F_{inf} \right] + 2MLA^2\sigma^2\left[ \xi^2 d + \tau - d \right]}{AM\sqrt{K}(\xi d + \tau - d)}.$$

*D. Proof of Convergence Rate*

**Lemma 1.** If the learning rate satisfies $\eta \leq M/[2L\xi^2(\beta+1)]$ and all local model parameters are initialized at the same point, then the average-squared gradient after $K$ iterations is bounded as follows

$$\begin{aligned}
&\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right] \\
\leq &\frac{2\left[F(\mu_1)-F_{inf}\right]}{\eta K(\xi d+\tau-d)}+\frac{2L\eta\sigma^2\left[\xi^2 d+\tau-d\right]}{M(\xi d+\tau-d)} \\
&+\frac{\eta^2L^2}{KM(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{m=1}^{M}\left[\sum_{i=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\mu_{K(k)}-x_{\tau k+d+i}^{(m)}\right\|^2+\xi\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\mu_{K(k)}-x_{\tau k+d+i}^{(m)}\right\|^2\right]
\end{aligned}$$

Proof.

From the Lipschitzisan gradient assumption $||\bigtriangledown F(x)-\bigtriangledown F(y)||\leq L||x-y||$, we have

$$\begin{aligned}
F(X_{K(k+1)})-F(X_{K(k)}) \leq &\left\langle\bigtriangledown F(X_{K(k)}),X_{K(k+1)}-X_{K(k)}\right\rangle+\frac{L}{2}\left\|X_{K(k+1)}-X_{K(k)}\right\|^2 \\
= &-\eta\left\langle\bigtriangledown F(X_{K(k)}),\mathcal{G}_{K(k)}\right\rangle+\frac{L\eta^2}{2}\left\|\mathcal{G}_{K(k)}\right\|^2
\end{aligned} \tag{11}$$

Taking expectation respect to $\mathcal{S}_{K(k)}$ on both sides of (11), we have

$$\mathbb{E}_{K(k)}\left[F(X_{K(k+1)})\right]-F(X_{K(k)})\leq-\eta\mathbb{E}_{K(k)}\left[\left\langle\bigtriangledown F(X_{K(k)}),\mathcal{G}_{K(k)}\right\rangle\right]+\frac{L\eta^2}{2}\mathbb{E}_{K(k)}\left[\left\|\mathcal{G}_{K(k)}\right\|^2\right]$$

From the fact

$$\langle a,b\rangle=\frac{1}{2}\left(||a||^2+||b||^2-||a-b||^2\right)$$

we have

$$\mathbb{E}_{K(k)}\left[F(X_{K(k+1)})\right]-F(X_{K(k)})\leq-\eta\mathbb{E}_{K(k)}\left[\left\langle\bigtriangledown F(X_{K(k)}),\mathcal{G}_{K(k)}\right\rangle\right]+\frac{L\eta^2}{2}\mathbb{E}_{K(k)}\left[\left\|\mathcal{G}_{K(k)}\right\|^2\right]$$

Combining with Lemmas 4 and 5, we obtain

$$\mathbb{E}_{K(k)}\left[F(X_{K(k+1)})\right]-F(X_{K(k)}) \tag{12}$$

$$\leq-\eta\mathbb{E}_{K(k)}\left[\left\langle\bigtriangledown F(X_{K(k)}),\mathcal{G}_{K(k)}\right\rangle\right]+\frac{L\eta^2}{2}\mathbb{E}_{K(k)}\left[\left\|\mathcal{G}_{K(k)}\right\|^2\right] \tag{13}$$

$$\leq-\eta\frac{\xi d+\tau-d}{2}\left\|\bigtriangledown F(X_{K(k)})\right\|^2 \tag{14}$$

$$+\left[\frac{L\xi^2\eta^2(\beta+1)}{M^2}-\frac{\eta\xi}{2M}\right]\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2+\left[\frac{L\xi^2\eta^2(\beta+1)}{M^2}-\frac{\eta}{2M}\right]\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 \tag{15}$$

$$+\eta\frac{1}{2M}\sum_{m=1}^{M}\left[\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F(X_{K(k)})-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2+\xi\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F(X_{K(k)})-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2\right] \tag{16}$$

$$+\frac{L\eta^2\sigma^2\left[\xi^2 d+\tau-d\right]}{M} \tag{17}$$

$$\leq-\eta\frac{\xi d+\tau-d}{2}\left\|\bigtriangledown F(X_{K(k)})\right\|^2 \tag{18}$$

$$+\left[\frac{L\xi^2\eta^2(\beta+1)}{M^2}-\frac{\eta\xi}{2M}\right]\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2+\left[\frac{L\xi^2\eta^2(\beta+1)}{M^2}-\frac{\eta}{2M}\right]\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 \tag{19}$$

$$+\frac{\eta L^2}{2M}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\left\|\mu_{K(k)}-x_{\tau k+d+i}^{(m)}\right\|^2+\frac{\eta\xi L^2}{2M}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\left\|\mu_{K(k)}-x_{\tau k+d+i}^{(m)}\right\|^2 \tag{20}$$

$$+\frac{L\eta^2\sigma^2\left[\xi^2 d+\tau-d\right]}{M} \tag{21}$$

where (20) is due to the Lipschitzisan gradient assumption $|| \bigtriangledown F(x) - \bigtriangledown F(y)|| \leq L||x-y||$. After minor rearranging and according to the definition of Frobenius norm, it is easy to show

$$\begin{aligned}
&\eta\frac{\xi d+\tau-d}{2}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2 \\
\leq& F(\mu_{K(k)}) - \mathbb{E}_{K(k)}\left[F(\mu_{K(k+1)})\right] + \frac{L\eta^2\sigma^2\left[\xi^2 d+\tau-d\right]}{M} && (22)\\
&+\left[\frac{L\xi^2\eta^2(\beta+1)}{M^2}-\frac{\eta\xi}{2M}\right]\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2+\left[\frac{L\xi^2\eta^2(\beta+1)}{M^2}-\frac{\eta}{2M}\right]\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 && (23)\\
&+\frac{\eta L^2}{2M}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\left\|\mu_{K(k)}-\ x_{\tau k+d+i}^{(m)}\right\|^2+\frac{\eta\xi L^2}{2M}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\left\|\mu_{K(k)}-\ x_{\tau k+d+i}^{(m)}\right\|^2 && (24)
\end{aligned}$$

Taking the total expectation and averaging over all iterates, we have

$$\begin{aligned}
&\eta\frac{\xi d+\tau-d}{2}\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right]\\
\leq&\frac{F(\mu_1)-F_{inf}}{K}+\frac{L\eta^2\sigma^2\left[\xi^2 d+\tau-d\right]}{M}\\
&+\left[\frac{L\xi^2\eta^2(\beta+1)}{KM^2}-\frac{\eta\xi}{2KM}\right]\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2+\left[\frac{L\xi^2\eta^2(\beta+1)}{KM^2}-\frac{\eta}{2KM}\right]\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2\\
&+\frac{\eta L^2}{2KM}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\mu_{K(k)}-\ x_{\tau k+d+i}^{(m)}\right\|^2+\frac{\eta\xi L^2}{2KM}\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\mu_{K(k)}-\ x_{\tau k+d+i}^{(m)}\right\|^2
\end{aligned}$$

Then, we have

$$\begin{aligned}
\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right]\leq&\frac{2\left[F(\mu_1)-F_{inf}\right]}{\eta K(\xi d+\tau-d)}+\frac{2L\eta\sigma^2\left[\xi^2 d+\tau-d\right]}{M(\xi d+\tau-d)}\\
&+\frac{2L\xi^2\eta^4(\beta+1)-\eta^2\xi M}{KM^2(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2\\
&+\frac{2L\xi^2\eta^4(\beta+1)-\eta^2 M}{KM^2(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 && (25)\\
&+\frac{\eta^2 L^2}{KM(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\mu_{K(k)}-\ x_{\tau k+d+i}^{(m)}\right\|^2\\
&+\frac{\eta^2\xi L^2}{KM(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\mu_{K(k)}-\ x_{\tau k+d+i}^{(m)}\right\|^2
\end{aligned}$$

If the learning rate satisfies $\eta\leq\sqrt{\frac{M}{2L\xi^2(\beta+1)}}$, then

$$\begin{aligned}
\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right]\leq&\frac{2\left[F(\mu_1)-F_{inf}\right]}{\eta K(\xi d+\tau-d)}+\frac{2L\eta\sigma^2\left[\xi^2 d+\tau-d\right]}{M(\xi d+\tau-d)}\\
&+\frac{\eta^2 L^2}{KM(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\mu_{K(k)}-\ x_{\tau k+d+i}^{(m)}\right\|^2\\
&+\frac{\eta^2\xi L^2}{KM(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\mu_{K(k)}-\ x_{\tau k+d+i}^{(m)}\right\|^2
\end{aligned}$$

Recalling the definition $\mu_{K(k)}=\frac{1}{M}\sum_{i=1}^{M}x_{\tau k+d}^{(i)}=\mathbf{X}_{K(k)}\mathbf{1}_M/M$ and adding a positive term to the RHS, one can get

$$\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\left\|\mu_{K(k)}-\ x_{\tau k+d+i}^{(m)}\right\|^2=\sum_{i=\tau-d}^{\tau-1}\left\|\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+i}\right\|_F^2$$

We have

$$\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right] \leq \frac{2\left[F(\mu_1)-F_{inf}\right]}{\eta K(\xi d+\tau-d)} + \frac{2L\eta\sigma^2\left[\xi^2 d+\tau-d\right]}{M(\xi d+\tau-d)} + \frac{\eta^2L^2}{KM(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+i}\right\|_F^2 + \frac{\eta^2\xi L^2}{KM(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+i}\right\|_F^2$$

---

**Lemma 2**.

$$\left\|\mathcal{H}_{K(k)}\right\|^2 \leq \frac{2d\xi^2}{M}\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 + \frac{2(\tau-d)}{M}\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 \tag{26}$$

Proof.

$$\begin{aligned}
\left\|\mathcal{H}_{K(k)}\right\|^2 =& \left\|\xi\frac{1}{M}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)+\frac{1}{M}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2 && (27)\\
\leq& \frac{2d\xi^2}{M^2}\sum_{i=\tau-d}^{\tau-1}\left\|\sum_{m=1}^{M}\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2+\frac{2(\tau-d)}{M^2}\sum_{i=0}^{\tau-1-d}\left\|\sum_{m=1}^{M}\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2 && (28)\\
\leq& \frac{2d\xi^2}{M}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\left\|\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2+\frac{2(\tau-d)}{M}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\left\|\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2 && (29)\\
=& \frac{2d\xi^2}{M}\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2+\frac{2(\tau-d)}{M}\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 && (30)
\end{aligned}$$

where (28) is due to $\|a+b\|^2 \leq 2\|a\|^2+2\|b\|^2$, (29) comes from the convexity of vector norm and Jensen's inequality.

---

**Lemma 3**. Under assumptions $\mathbb{E}_{\mathcal{S}_k|x}\left[g(x)\right]=\bigtriangledown F(x)$ and $\mathbb{E}_{\mathcal{S}_k|x}||g(x)-\bigtriangledown F(x)||^2 \leq \beta||\bigtriangledown F(x)||^2+\sigma^2$, we have the following variance bound for the averaged stochastic gradient:

$$\mathbb{E}_{K(k)}\left[\left\|\mathcal{G}_{K(k)}-\mathcal{H}_{K(k)}\right\|^2\right] \leq \frac{2\sigma^2\left[\xi^2 d+\tau-d\right]}{M}+\frac{2\beta\xi^2}{M^2}\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2+\frac{2\beta}{M^2}\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 \tag{31}$$

Proof. According to the definition of (8), (9), and (10), we have

$$\begin{aligned}
&\mathbb{E}_{K(k)}\left[\left\|\mathcal{G}_{K(k)}-\mathcal{H}_{K(k)}\right\|^2\right] && (32)\\
=& \frac{1}{M^2}\mathbb{E}_{K(k)}\left[\left\|\xi\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\left[g\left(x_{\tau k+d+i}^{(m)}\right)-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right]+\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\left[g\left(x_{\tau k+d+i}^{(m)}\right)-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right]\right\|^2\right] && (33)\\
\leq& \frac{2}{M^2}\mathbb{E}_{K(k)}\left[\left\|\xi\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\left[g\left(x_{\tau k+d+i}^{(m)}\right)-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right]\right\|^2+\left\|\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\left[g\left(x_{\tau k+d+i}^{(m)}\right)-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right]\right\|^2\right] && (34)\\
=& \frac{2}{M^2}\mathbb{E}_{K(k)}\left[\xi^2\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\left\|g\left(x_{\tau k+d+i}^{(m)}\right)-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2+\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\left\|g\left(x_{\tau k+d+i}^{(m)}\right)-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2\right. && (35)\\
&\left.+\xi^2\sum_{j\neq i}^{\tau-1}\sum_{l\neq m}^{M}\left\langle g\left(x_{\tau k+d+i}^{(m)}\right)-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right),g\left(x_{\tau k+d+j}^{(l)}\right)-\bigtriangledown F\left(x_{\tau k+d+j}^{(l)}\right)\right\rangle\right] && (36)
\end{aligned}$$

$$+\sum_{j\neq i}^{\tau-1-d}\sum_{l\neq m}^{M}\left\langle g\left(x_{\tau k+d+i}^{(m)}\right)-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right),g\left(x_{\tau k+d+j}^{(l)}\right)-\bigtriangledown F\left(x_{\tau k+d+j}^{(l)}\right)\right\rangle\Bigg] \tag{37}$$

$$=\frac{2\xi^2}{M^2}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|g\left(x_{\tau k+d+i}^{(m)}\right)-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2+\frac{2}{M^2}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|g\left(x_{\tau k+d+i}^{(m)}\right)-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2 \tag{38}$$

where (34) is due to $\|a+b\|^2\leq 2\|a\|^2+2\|b\|^2$, (38) is due to $s_k^i$ are independent random variables and the assumption $\mathbb{E}_{\mathcal{S}_k|x}\left[g(x)\right]=\bigtriangledown F(x)$. Now, directly applying assumption $\mathbb{E}_{\mathcal{S}_k|x}||g(x)-\bigtriangledown F(x)||^2\leq\beta||\bigtriangledown F(x)||^2+\sigma^2$ to (38). Then, we have

$$\mathbb{E}_{K(k)}\left[\left\|\mathcal{G}_{K(k)}-\mathcal{H}_{K(k)}\right\|^2\right]\leq\frac{2\xi^2}{M^2}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\left[\beta\left\|\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right\|^2+\sigma^2\right]+\frac{2}{M^2}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\left[\beta\left\|\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right\|^2+\sigma^2\right]$$

$$=\frac{2\sigma^2\left[\xi^2d+\tau-d\right]}{M}+\frac{2\xi^2}{M^2}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\beta\left\|\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right\|^2+\frac{2}{M^2}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\beta\left\|\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right\|^2 \tag{39}$$

$$=\frac{2\sigma^2\left[\xi^2d+\tau-d\right]}{M}+\frac{2\beta\xi^2}{M^2}\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2+\frac{2\beta}{M^2}\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 \tag{40}$$

---

**Lemma 4**. Under assumption $\mathbb{E}_{\mathcal{S}_k|x}\left[g(x)\right]=\bigtriangledown F(x)$, the expected inner product between stochastic gradient and full batch gradient can be expanded as

$$\mathbb{E}_{K(k)}\left[\left\langle\bigtriangledown F(X_{K(k)}),\mathcal{G}_{K(k)}\right\rangle\right]$$
$$=\frac{\xi d+\tau-d}{2}\left\|\bigtriangledown F(X_{K(k)})\right\|^2+\frac{1}{2M}\left[\xi\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2+\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2\right]$$
$$-\frac{1}{2M}\sum_{m=1}^{M}\left[\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F(X_{K(k)})-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2+\xi\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F(X_{K(k)})-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2\right]$$

Proof.

$$\mathbb{E}_{K(k)}\left[\left\langle\bigtriangledown F(X_{K(k)}),\mathcal{G}_{K(k)}\right\rangle\right] \tag{41}$$

$$=\mathbb{E}_{K(k)}\left[\left\langle\bigtriangledown F(X_{K(k)}),\xi\frac{1}{M}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}g\left(x_{\tau k+d+i}^{(m)}\right)+\frac{1}{M}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}g\left(x_{\tau k+d+i}^{(m)}\right)\right\rangle\right] \tag{42}$$

$$=\xi\frac{1}{M}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\left\langle\bigtriangledown F(X_{K(k)}),\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\rangle+\frac{1}{M}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\left\langle\bigtriangledown F(X_{K(k)}),\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\rangle \tag{43}$$

$$=\frac{\xi}{2M}\sum_{i=\tau-d}^{\tau-1}\sum_{m=1}^{M}\left[\left\|\bigtriangledown F(X_{K(k)})\right\|^2+\left\|\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2-\left\|\bigtriangledown F(X_{K(k)})-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2\right] \tag{44}$$

$$+\frac{1}{2M}\sum_{i=0}^{\tau-1-d}\sum_{m=1}^{M}\left[\left\|\bigtriangledown F(X_{K(k)})\right\|^2+\left\|\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2-\left\|\bigtriangledown F(X_{K(k)})-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2\right] \tag{45}$$

$$=\frac{\xi d+\tau-d}{2}\left\|\bigtriangledown F(X_{K(k)})\right\|^2+\frac{1}{2M}\left[\xi\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2+\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2\right] \tag{46}$$

$$-\frac{1}{2M}\sum_{m=1}^{M}\left[\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F(X_{K(k)})-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2+\xi\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F(X_{K(k)})-\bigtriangledown F\left(x_{\tau k+d+i}^{(m)}\right)\right\|^2\right] \tag{47}$$

where (44) and (45) come from $\langle a,b\rangle=\frac{1}{2}\left(||a||^2+||b||^2-||a-b||^2\right)$.

---

**Lemma 5.** Under assumptions $E_{\xi|x}\left[g(x)\right] = \bigtriangledown F(x)$ and $E_{\xi|x}||g(x) - \bigtriangledown F(x)||^2 \leq \beta||\bigtriangledown F(x)||^2 + \sigma^2$, the squared norm of stochastic gradient can be bounded as

$$\mathbb{E}_{K(k)}\left[\left\|\mathcal{G}_{K(k)}\right\|^2\right] \leq \frac{2\sigma^2\left[\xi^2 d + \tau - d\right]}{M} + \frac{2(\beta+1)}{M^2}\left[\xi^2\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 + \sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2\right]$$

Proof.

$$\begin{aligned}
\mathbb{E}_{K(k)}\left[\left\|\mathcal{G}_{K(k)}\right\|^2\right] &= \mathbb{E}_{K(k)}\left[\left\|\mathcal{G}_{K(k)} - \mathbb{E}_{K(k)}[\mathcal{G}_{K(k)}]\right\|^2\right] + \left\|\mathbb{E}_{K(k)}[\mathcal{G}_{K(k)}]\right\|^2 && (48)\\
&= \mathbb{E}_{K(k)}\left[\left\|\mathcal{G}_{K(k)} - \mathcal{H}_{K(k)}\right\|^2\right] + \left\|\mathcal{H}_{K(k)}\right\|^2 && (49)\\
&\leq \frac{2\sigma^2\left[\xi^2 d + \tau - d\right]}{M} + \frac{2d\xi^2 M + 2\beta\xi^2}{M^2}\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 + \frac{2(\tau-d)M + 2\beta}{M^2}\sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 && (50)\\
&= \frac{2\sigma^2\left[\xi^2 d + \tau - d\right]}{M} + \frac{2(\beta+1)}{M^2}\left[\xi^2\sum_{i=\tau-d}^{\tau-1}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 + \sum_{i=0}^{\tau-1-d}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2\right] && (51)
\end{aligned}$$

where (50) follows (26) and (31).

---

**Theorem 1 (Convergence of SGD).** Under assumptions, if the learning rate satisfies the following two formulas at the same time

$$\begin{aligned}
\eta \leq& \sqrt{\frac{1}{2L\xi^2(\beta+1)(1-\xi) + 3L^2(\tau-d)[(1-\xi)(2\beta+2) + (2\beta+2k\tau)] + 3d\xi L^2[(2\beta+2k\tau) + (2\beta+2)(1-\xi)]}}\\
=& \sqrt{\frac{1}{2L\xi^2(\beta+1)(1-\xi) + 6L^2(d\xi+\tau-d)[(\beta+k\tau) + (\beta+1)(1-\xi)]}}
\end{aligned}$$

$$\eta \leq \sqrt{\frac{\xi M(1-\xi)}{2L\xi^2(\beta+1)(1-\xi) + 3L^2M(\tau-d)(2\beta+2k\tau) + 6dM\xi L^2[(\beta+k\tau) + (\beta+1)(1-\xi)]}}$$

Then the average-squared gradient norm after $K$ iterations is bounded as

$$\begin{aligned}
&\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right]\\
\leq& \frac{2M\left[F(\mu_1) - F_{inf}\right] + 2MKL\eta^2\sigma^2\left[\xi^2 d + \tau - d\right]}{\eta MK(\xi d + \tau - d)} + \frac{3\eta^4\xi L^2(\tau - d + d\xi)}{MK(\xi d + \tau - d)}\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1} g(\mathbf{X}_{d-1})\right\|_F^2\\
&+ \frac{6\eta^4 L^2\sigma^2}{\xi d + \tau - d}\left(\tau\frac{\xi^2}{1-\xi^2}(\tau - d + \xi d) + (\tau-d)^2 + \xi d(\tau-1)\right)
\end{aligned}$$

Proof.

Recall the intermediate result (25) in the proof of Lemma 1:

$$\begin{aligned}
\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right] \leq& \frac{2\left[F(\mu_1) - F_{inf}\right]}{\eta K(\xi d + \tau - d)} + \frac{2L\eta\sigma^2\left[\xi^2 d + \tau - d\right]}{M(\xi d + \tau - d)}\\
&+ \frac{2L\xi^2\eta^3 = 4(\beta+1) - \eta^2\xi M}{KM^2(\xi d + \tau - d)}\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2\\
&+ \frac{2L\xi^2\eta^4(\beta+1) - \eta^2 M}{KM^2(\xi d + \tau - d)}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2\\
&+ \frac{\eta^2 L^2}{KM(\xi d + \tau - d)}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J} - \mathbf{X}_{\tau k+d+i}\right\|_F^2\\
&+ \frac{\eta^2\xi L^2}{KM(\xi d + \tau - d)}\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J} - \mathbf{X}_{\tau k+d+i}\right\|_F^2
\end{aligned}$$

Our goal is to provide an upper bound for the network error term $\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+i}\right\|_F^2$. First of all, let us derive a specific expression for $\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+i}$.

According to the update rule (7), one can observe that

$$\begin{aligned}
&\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+i} && (52)\\
=\;&\mathbf{X}_{\tau k+d}(\mathbf{J}-\mathbf{I})+\eta\sum_{j=0}^{i}\mathbf{G}_{\tau k+d+j} && (53)\\
=\;&\xi\left(\mathbf{X}_{\tau k+d-1}-\eta\mathbf{G}_{\tau k+d-1}\right)(\mathbf{J}-\mathbf{I})+(1-\xi)\left(\mathbf{X}_{\tau k}-\eta\mathbf{G}_{\tau k}\right)\mathbf{J}(\mathbf{J}-\mathbf{I})+\eta\sum_{j=0}^{i}\mathbf{G}_{\tau k+d+j} && (54)\\
=\;&\xi\mathbf{X}_{\tau(k-1)+d}(\mathbf{J}-\mathbf{I})-\xi\eta\sum_{i=0}^{\tau-1}\mathbf{G}_{\tau(k-1)+d+i}(\mathbf{J}-\mathbf{I})+\eta\sum_{j=0}^{i}\mathbf{G}_{\tau k+d+j} && (55)\\
=\;&\xi^2\mathbf{X}_{\tau(k-2)+d}(\mathbf{J}-\mathbf{I})-\eta\sum_{j=1}^{2}\sum_{i=0}^{\tau-1}\xi^j\mathbf{G}_{\tau(k-j)+d+i}(\mathbf{J}-\mathbf{I})+\eta\sum_{j=0}^{i}\mathbf{G}_{\tau k+d+j} && (56)\\
=\;&\xi^k\mathbf{X}_{d}(\mathbf{J}-\mathbf{I})-\eta\sum_{j=1}^{k}\sum_{i=0}^{\tau-1}\xi^j\mathbf{G}_{\tau(k-j)+d+i}(\mathbf{J}-\mathbf{I})+\eta\sum_{j=0}^{i}\mathbf{G}_{\tau k+d+j} && (57)\\
=\;&\xi^k\left(\mathbf{X}_{d-1}-\eta\mathbf{G}_{d-1}\right)(\mathbf{J}-\mathbf{I})-\eta\sum_{j=1}^{k}\sum_{i=0}^{\tau-1}\xi^j\mathbf{G}_{\tau(k-j)+d+i}(\mathbf{J}-\mathbf{I})+\eta\sum_{j=0}^{i}\mathbf{G}_{\tau k+d+j} && (58)\\
=\;&\xi^k\mathbf{X}_{1}(\mathbf{J}-\mathbf{I})-\eta\xi^k\sum_{i=1}^{d-1}\mathbf{G}_{i}(\mathbf{J}-\mathbf{I})-\eta\sum_{j=1}^{k}\sum_{i=0}^{\tau-1}\xi^j\mathbf{G}_{\tau(k-j)+d+i}(\mathbf{J}-\mathbf{I})+\eta\sum_{j=0}^{i}\mathbf{G}_{\tau k+d+j} && (59)\\
=\;&-\eta\xi^k\sum_{i=1}^{d-1}\mathbf{G}_{i}(\mathbf{J}-\mathbf{I})-\eta\sum_{j=1}^{k}\sum_{i=0}^{\tau-1}\xi^j\mathbf{G}_{\tau(k-j)+d+i}(\mathbf{J}-\mathbf{I})+\eta\sum_{j=0}^{i}\mathbf{G}_{\tau k+d+j} && (60)
\end{aligned}$$

where (60) follows the fact that all workers start from the same point at the beginning of each local update period. Accordingly, we have

$$\begin{aligned}
&\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+i}\right\|_F^2 && (61)\\
=&\sum_{l=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|-\eta\xi^k\sum_{i=1}^{d-1}\mathbf{G}_{i}(\mathbf{J}-\mathbf{I})-\eta\sum_{j=1}^{k}\sum_{i=0}^{\tau-1}\xi^j\mathbf{G}_{\tau(k-j)+d+i}(\mathbf{J}-\mathbf{I})+\eta\sum_{i=0}^{l}\mathbf{G}_{\tau k+d+i}\right\|_F^2 && (62)\\
\le&3\eta^2\mathbb{E}_{K(k)}\left[\xi^{2k}d\left\|\sum_{i=1}^{d-1}\mathbf{G}_{i}(\mathbf{J}-\mathbf{I})\right\|_F^2+d\left\|\sum_{j=1}^{k}\sum_{i=0}^{\tau-1}\xi^j\mathbf{G}_{\tau(k-j)+d+i}(\mathbf{J}-\mathbf{I})\right\|_F^2+\sum_{l=\tau-d}^{\tau-1}\left\|\sum_{i=0}^{l}\mathbf{G}_{\tau k+d+i}\right\|_F^2\right] && (63)\\
\le&3\eta^2\mathbb{E}_{K(k)}\left[\xi^{2k}d\left\|\sum_{i=1}^{d-1}\mathbf{G}_{d-1}\right\|_F^2+d\left\|\sum_{j=1}^{k}\sum_{i=0}^{\tau-1}\xi^j\mathbf{G}_{\tau(k-j)+d+i}\right\|_F^2+\sum_{l=\tau-d}^{\tau-1}\left\|\sum_{i=0}^{l}\mathbf{G}_{\tau k+d+i}\right\|_F^2\right] && (64)\\
=&3\eta^2\sum_{m=1}^{M}\left[\xi^{2k}d\mathbb{E}_{K(k)}\left\|\sum_{i=1}^{d-1}g(x_{d-1}^{(m)})\right\|^2+d\mathbb{E}_{K(k)}\left\|\sum_{j=1}^{k}\sum_{i=0}^{\tau-1}\xi^j g(x_{\tau(k-j)+d+i}^{(m)})\right\|^2+\mathbb{E}_{K(k)}\sum_{l=\tau-d}^{\tau-1}\left\|\sum_{i=0}^{l}g(x_{\tau k+d+i}^{(m)})\right\|^2\right] && (65)\\
=&3\eta^2 d\left[\underbrace{\sum_{m=1}^{M}\xi^{2k}\left\|\sum_{i=1}^{d-1}g(x_{d-1}^{(m)})\right\|^2}_{T_1}+\underbrace{\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\sum_{j=1}^{k}\xi^j\sum_{i=0}^{\tau-1}g(x_{\tau(k-j)+d+i}^{(m)})\right\|^2}_{T_2}+\underbrace{\frac{1}{d}\sum_{m=1}^{M}\sum_{l=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\sum_{i=0}^{l}g(x_{\tau k+d+i}^{(m)})\right\|^2}_{T_3}\right] && (66)
\end{aligned}$$

where the (64) is due to the operator norm of $\mathbf{J}-\mathbf{I}$ is less than 1. For $T2$, we have

$$\begin{aligned}
&\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\sum_{j=1}^{k}\xi^{j}\sum_{i=0}^{\tau-1}g(x^{(m)}_{\tau(k-j)+d+i})\right\|^{2}\\
&=\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\sum_{j=1}^{k}\xi^{j}\sum_{i=0}^{\tau-1}\left[g(x^{(m)}_{\tau(k-j)+d+i})-\bigtriangledown F(x^{(m)}_{\tau(k-j)+d+i})\right]+\sum_{j=1}^{k}\xi^{j}\sum_{i=0}^{\tau-1}\bigtriangledown F(x^{(m)}_{\tau(k-j)+d+i})\right\|^{2} &(67)\\
&\leq\underbrace{2\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\sum_{j=1}^{k}\xi^{j}\sum_{i=0}^{\tau-1}\left[g(x^{(m)}_{\tau(k-j)+d+i})-\bigtriangledown F(x^{(m)}_{\tau(k-j)+d+i})\right]\right\|^{2}}_{T_4}+\underbrace{2\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\sum_{j=1}^{k}\xi^{j}\sum_{i=0}^{\tau-1}\bigtriangledown F(x^{(m)}_{\tau(k-j)+d+i})\right\|^{2}}_{T_5} &(68)
\end{aligned}$$

For the first term $T_4$, since the stochastic gradients are unbiased, all cross terms are zero. Thus, combining with Assumption 3, we have

$$\begin{aligned}
T_4 =&\ 2\sum_{m=1}^{M}\sum_{j=1}^{k}\xi^{2j}\sum_{i=0}^{\tau-1}\mathbb{E}_{K(k)}\left\|g(x^{(m)}_{\tau(k-j)+d+i})-\bigtriangledown F(x^{(m)}_{\tau(k-j)+d+i})\right\|^{2}\\
\leq&\ 2\sum_{m=1}^{M}\sum_{j=1}^{k}\xi^{2j}\sum_{i=0}^{\tau-1}\left[\beta\left\|\bigtriangledown F(x^{(m)}_{\tau(k-j)+d+i})\right\|^{2}+\sigma^{2}\right] &(69)\\
=&\ 2\sum_{j=1}^{k}\xi^{2j}\sum_{i=0}^{\tau-1}\left[\beta\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_{F}^{2}+M\sigma^{2}\right] &(70)
\end{aligned}$$

For the second term $T_5$, directly applying Jensen's inequality, we get

$$\begin{aligned}
T_5 =&\ 2\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\sum_{j=1}^{k}\xi^{j}\sum_{i=0}^{\tau-1}\bigtriangledown F(x^{(m)}_{\tau(k-j)+d+i})\right\|^{2}\\
\leq&\ 2k\sum_{m=1}^{M}\mathbb{E}_{K(k)}\sum_{j=1}^{k}\xi^{2j}\left\|\sum_{i=0}^{\tau-1}\bigtriangledown F(x^{(m)}_{\tau(k-j)+d+i})\right\|^{2} &(71)\\
\leq&\ 2k\tau\sum_{m=1}^{M}\mathbb{E}_{K(k)}\sum_{j=1}^{k}\xi^{2j}\sum_{i=0}^{\tau-1}\left\|\bigtriangledown F(x^{(m)}_{\tau(k-j)+d+i})\right\|^{2} &(72)\\
=&\ 2k\tau\mathbb{E}_{K(k)}\sum_{j=1}^{k}\xi^{2j}\sum_{i=0}^{\tau-1}\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_{F}^{2} &(73)
\end{aligned}$$

Substituting the bounds of $T_4$ and $T_5$ into $T_2$

$$\begin{aligned}
T_2\leq&\ 2\sum_{j=1}^{k}\xi^{2j}\sum_{i=0}^{\tau-1}\left[\beta\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_{F}^{2}+\sigma^{2}\right]+2k\tau\sum_{j=1}^{k}\xi^{j}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_{F}^{2} &(74)\\
=&\ 2\sum_{j=1}^{k}\xi^{2j}M\sigma^{2}\tau+(2\beta+2k\tau)\sum_{j=1}^{k}\xi^{j}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_{F}^{2} &(75)\\
\leq&\ 2M\sigma^{2}\tau\frac{\xi^{2}}{1-\xi^{2}}+(2\beta+2k\tau)\sum_{j=1}^{k}\xi^{j}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_{F}^{2} &(76)
\end{aligned}$$

where (76) according to the summation formula of power

$$\sum_{j=1}^{k}\xi^{2j}\leq\sum_{j=1}^{\infty}\xi^{2j}\leq\frac{\xi^{2}}{1-\xi^{2}}$$

For $T_3$, we have

$$
\begin{aligned}
T_3 = & \frac{1}{d}\sum_{m=1}^{M}\sum_{l=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\sum_{i=0}^{l} g(x_{\tau k+d+i}^{(m)})\right\|^2 \\
= & \frac{1}{d}\sum_{m=1}^{M}\sum_{l=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\sum_{i=0}^{l}\left(g(x_{\tau k+d+i}^{(m)})-\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right)+\sum_{i=0}^{l}\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right\|^2 && (77)\\
\leq & \frac{2}{d}\sum_{m=1}^{M}\sum_{l=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\sum_{i=0}^{l}\left(g(x_{\tau k+d+i}^{(m)})-\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right)\right\|^2+\frac{2}{d}\sum_{m=1}^{M}\sum_{l=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\sum_{i=0}^{l}\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right\|^2 && (78)\\
\leq & \frac{2}{d}\sum_{m=1}^{M}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\mathbb{E}_{K(k)}\left\|g(x_{\tau k+d+i}^{(m)})-\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right\|^2+\frac{2}{d}\sum_{m=1}^{M}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\mathbb{E}_{K(k)}\left\|\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right\|^2 && (79)\\
\leq & \frac{2}{d}\sum_{m=1}^{M}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\left[\beta\mathbb{E}_{K(k)}\left\|\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right\|^2+\sigma^2\right]+\frac{2}{d}\sum_{m=1}^{M}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\mathbb{E}_{K(k)}\left\|\bigtriangledown F(x_{\tau k+d+i}^{(m)})\right\|^2 && (80)\\
= & \frac{2M}{d}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\sigma^2+\frac{2\beta+2}{d}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\mathbb{E}_{K(k)}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2 && (81)
\end{aligned}
$$

We have

$$
\begin{aligned}
& \sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+i}\right\|_F^2 \\
= & 3\eta^2 d\left[\sum_{m=1}^{M}\xi^{2k}\left\|\sum_{i=1}^{d-1}g(x_{d-1}^{(m)})\right\|^2+\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\sum_{j=1}^{k}\xi^j\sum_{i=0}^{\tau-1}g(x_{\tau(k-j)+d+i}^{(m)})\right\|^2+\sum_{m=1}^{M}\sum_{l=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\sum_{i=0}^{l}g(x_{\tau k+d+i}^{(m)})\right\|^2\right] && (82)\\
\leq & 3\eta^2 d\left[\sum_{m=1}^{M}\xi^{2k}\left\|\sum_{i=1}^{d-1}g(x_{d-1}^{(m)})\right\|^2+2M\sigma^2\tau\frac{\xi^2}{1-\xi^2}+(2\beta+2k\tau)\sum_{j=1}^{k}\xi^j\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_F^2\right. && (83)\\
& \left.+\frac{2M}{d}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\sigma^2+\frac{2\beta+2}{d}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\mathbb{E}_{K(k)}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2\right] && (84)
\end{aligned}
$$

And

$$
\begin{aligned}
& \sum_{l=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+l}\right\|_F^2 \\
= & 3\eta^2\left[(\tau-d)\sum_{m=1}^{M}\xi^{2k}\left\|\sum_{i=1}^{d-1}g(x_{d-1}^{(m)})\right\|^2+(\tau-d)\sum_{m=1}^{M}\mathbb{E}_{K(k)}\left\|\sum_{j=1}^{k}\xi^j\sum_{i=0}^{\tau-1}g(x_{\tau(k-j)+d+i}^{(m)})\right\|^2+\sum_{m=1}^{M}\sum_{l=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\sum_{i=0}^{l}g(x_{\tau k+d+i}^{(m)})\right\|^2\right] && (85)\\
\leq & 3\eta^2(\tau-d)\left[\sum_{m=1}^{M}\xi^{2k}\left\|\sum_{i=1}^{d-1}g(x_{d-1}^{(m)})\right\|^2+2M\sigma^2\tau\frac{\xi^2}{1-\xi^2}+(2\beta+2k\tau)\sum_{j=1}^{k}\xi^j\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_F^2\right. && (86)\\
& \left.+\frac{2M}{\tau-d}\sum_{l=0}^{\tau-1-d}\sum_{i=0}^{l}\sigma^2+\frac{2\beta+2}{\tau-d}\sum_{l=0}^{\tau-1-d}\sum_{i=0}^{l}\mathbb{E}_{K(k)}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2\right] && (87)
\end{aligned}
$$

Then, summing over all periods from $k=0$ to $k=K$, where $K$ is the total global iterations:

$$
\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+i}\right\|_F^2 \tag{88}
$$

$$\leq\ 3\eta^2 d\sum_{k=1}^{K}\left[\sum_{m=1}^{M}\xi^{2k}\left\|\sum_{i=1}^{d-1}g(x_{d-1}^{(m)})\right\|^2+2M\sigma^2\tau\frac{\xi^2}{1-\xi^2}+(2\beta+2k\tau)\sum_{j=1}^{k}\xi^j\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_F^2\right. \tag{89}$$

$$\left.+\ \frac{2M}{d}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\sigma^2+\frac{2\beta+2}{d}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\mathbb{E}_{K(k)}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2\right] \tag{90}$$

$$\leq 3\eta^2 d\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1}g(\mathbf{X}_{d-1})\right\|_F^2+6\eta^2 dKM\sigma^2\tau\frac{\xi^2}{1-\xi^2}+\frac{6\eta^2 dMK}{d}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\sigma^2 \tag{91}$$

$$+\ 3\eta^2 d(2\beta+2k\tau)\sum_{k=1}^{K}\sum_{j=1}^{k}\xi^j\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_F^2+3\eta^2(2\beta+2)\sum_{k=1}^{K}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2 \tag{92}$$

Expanding the summation, we have

$$\sum_{k=1}^{K}\sum_{j=1}^{k}\xi^j\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau(k-j)+d+i})\right\|_F^2=\ \sum_{k=1}^{K}\sum_{r=0}^{k-1}\left[\xi^{k-r}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau r+d+i})\right\|_F^2\right] \tag{93}$$

$$\leq\ \sum_{r=1}^{K}\left[\left(\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau r+d+i})\right\|_F^2\right)\left(\sum_{k=r}^{K}\xi^{k-r}\right)\right] \tag{94}$$

$$\leq\ \sum_{r=1}^{K}\left[\left(\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau r+d+i})\right\|_F^2\right)\left(\sum_{k=r}^{+\infty}\xi^{k-r}\right)\right] \tag{95}$$

$$\leq\ \frac{1}{1-\xi}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2 \tag{96}$$

Thus, we have

$$\sum_{k=1}^{K}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2\leq d\sum_{k=1}^{K}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2 \tag{97}$$

Plugging (96) and (97) into (92),

$$\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+i}\right\|_F^2$$

$$\leq 3\eta^2 d\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1}g(\mathbf{X}_{d-1})\right\|_F^2+6\eta^2 dKM\sigma^2\tau\frac{\xi^2}{1-\xi^2}+\frac{6\eta^2 dMK}{d}\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\sigma^2 \tag{98}$$

$$+\ 3\eta^2 d\frac{2\beta+2k\tau}{1-\xi}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2+3\eta^2(2\beta+2)d\sum_{k=1}^{K}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2 \tag{99}$$

And

$$\sum_{k=1}^{K}\sum_{l=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\mathbf{X}_{\tau k+d}\mathbf{J}-\mathbf{X}_{\tau k+d+l}\right\|_F^2$$

$$\leq\ 3\eta^2(\tau-d)\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1}g(\mathbf{X}_{d-1})\right\|_F^2+6\eta^2(\tau-d)KM\sigma^2\tau\frac{\xi^2}{1-\xi^2}+6\eta^2 MK\sum_{l=0}^{\tau-1-d}\sum_{i=0}^{l}\sigma^2 \tag{100}$$

$$+\ 3\eta^2(\tau-d)\frac{2\beta+2k\tau}{1-\xi}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2+3\eta^2(2\beta+2)(\tau-d)\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2 \tag{101}$$

Recall the intermediate result (25) in the proof of Lemma 1:

$$\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right] \tag{102}$$

$$\leq\frac{2\left[F(\mu_1)-F_{inf}\right]}{\eta K(\xi d+\tau-d)}+\frac{2L\eta\sigma^2\left[\xi^2 d+\tau-d\right]}{M(\xi d+\tau-d)} \tag{103}$$

$$+\frac{\eta^2L^2}{KM(\xi d+\tau-d)}\left[3\eta^2(\tau-d)\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1}g(\mathbf{X}_{d-1})\right\|_F^2+6\eta^2MK\left((\tau-d)\sigma^2\tau\frac{\xi^2}{1-\xi^2}+\sum_{l=0}^{\tau-1-d}\sum_{i=0}^{l}\sigma^2\right)\right] \tag{104}$$

$$+\frac{\eta^2\xi L^2}{KM(\xi d+\tau-d)}\left[3\eta^2d\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1}g(\mathbf{X}_{d-1})\right\|_F^2+6\eta^2MK\left(d\sigma^2\tau\frac{\xi^2}{1-\xi^2}+\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\sigma^2\right)\right] \tag{105}$$

$$+\frac{2L\xi^2\eta^4(\beta+1)-\eta^2\xi M}{KM^2(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 \tag{106}$$

$$+\frac{2L\xi^2\eta^4(\beta+1)-\eta^2M}{KM^2(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2 \tag{107}$$

$$+\frac{\eta^2L^2}{KM(\xi d+\tau-d)}\left[3\eta^2(\tau-d)\frac{2\beta+2k\tau}{1-\xi}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2+3\eta^2(2\beta+2)(\tau-d)\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2\right] \tag{108}$$

$$+\frac{\eta^2\xi L^2}{KM(\xi d+\tau-d)}\left[3\eta^2d\frac{2\beta+2k\tau}{1-\xi}\sum_{k=1}^{K}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2+3\eta^2(2\beta+2)d\sum_{k=1}^{K}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2\right] \tag{109}$$

Rearrange (106), (107), (108), and (109)

$$\begin{aligned}
&\frac{2L\xi^2\eta^4(\beta+1)/M-\eta^2\xi}{KM(\xi d+\tau-d)}\sum_{k=1}^{K}\sum_{i=\tau-d}^{\tau-1}\mathbb{E}_{K(k)}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2\\
&+\left(\frac{2L\eta\xi^2\eta^3(\beta+1)(1-\xi)-\eta^2+3L^2\eta^4(2\beta+2)(\tau-d)(1-\xi)}{KM(\xi d+\tau-d)(1-\xi)}\right)\sum_{k=1}^{K}\sum_{i=0}^{\tau-1-d}\mathbb{E}_{K(k)}\left\|\bigtriangledown F\left(\mathbf{X}_{\tau k+d+i}\right)\right\|_F^2\\
&+\left(\frac{\eta^2L^23\eta^2(\tau-d)(2\beta+2k\tau)+3d\eta^4\xi L^2[(2\beta+2k\tau)+(2\beta+2)(1-\xi)]}{KM(\xi d+\tau-d)(1-\xi)}\right)\sum_{k=1}^{K}\sum_{i=0}^{\tau-1}\mathbb{E}\left\|\bigtriangledown F(\mathbf{X}_{\tau k+d+i})\right\|_F^2
\end{aligned}$$

When the learning rate satisfies the following two formulas at the same time

$$\begin{aligned}
\eta\leq&\sqrt{\frac{1}{2L\xi^2(\beta+1)(1-\xi)+3L^2(\tau-d)[(1-\xi)(2\beta+2)+(2\beta+2k\tau)]+3d\xi L^2[(2\beta+2k\tau)+(2\beta+2)(1-\xi)]}}\\
=&\sqrt{\frac{1}{2L\xi^2(\beta+1)(1-\xi)+6L^2(d\xi+\tau-d)[(\beta+k\tau)+(\beta+1)(1-\xi)]}}
\end{aligned} \tag{110}$$

$$\eta\leq\sqrt{\frac{\xi M(1-\xi)}{2L\xi^2(\beta+1)(1-\xi)+3L^2M(\tau-d)(2\beta+2k\tau)+6dM\xi L^2[(\beta+k\tau)+(\beta+1)(1-\xi)]}}$$

We have

$$\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right]$$

$$\leq\frac{2\left[F(\mu_1)-F_{inf}\right]}{\eta K(\xi d+\tau-d)}+\frac{2L\eta\sigma^2\left[\xi^2d+\tau-d\right]}{M(\xi d+\tau-d)} \tag{111}$$

$$+\frac{\eta^4L^2}{KM(\xi d+\tau-d)}\left[3(\tau-d)\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1}g(\mathbf{X}_{d-1})\right\|_F^2+6MK\left((\tau-d)\sigma^2\tau\frac{\xi^2}{1-\xi^2}+\sum_{l=0}^{\tau-1-d}\sum_{i=0}^{l}\sigma^2\right)\right] \tag{112}$$

$$+\frac{\eta^4\xi L^2}{KM(\xi d+\tau-d)}\left[3d\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1}g(\mathbf{X}_{d-1})\right\|_F^2+6MK\left(d\sigma^2\tau\frac{\xi^2}{1-\xi^2}+\sum_{l=\tau-d}^{\tau-1}\sum_{i=0}^{l}\sigma^2\right)\right] \tag{113}$$

$$\leq\frac{2M\left[F(\mu_1)-F_{inf}\right]+2MKL\eta^2\sigma^2\left[\xi^2d+\tau-d\right]}{\eta MK(\xi d+\tau-d)}+\frac{3\eta^4\xi L^2(\tau-d+d\xi)}{MK(\xi d+\tau-d)}\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1}g(\mathbf{X}_{d-1})\right\|_F^2 \tag{114}$$

$$+\frac{6\eta^4L^2\sigma^2}{\xi d+\tau-d}\left(\tau\frac{\xi^2}{1-\xi^2}(\tau-d+\xi d)+(\tau-d)^2+\xi d(\tau-1)\right) \tag{115}$$

---

**Corollary 1.** Under sssumptions, if the learning rate is $\eta = A/\sqrt{K}$ the average-squared gradient norm after $K$ iterations is bounded by

$$\begin{aligned}
&\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right] \\
\leq&\frac{2M\left[F(\mu_1)-F_{inf}\right]+2MLA^2\sigma^2\left[\xi^2d+\tau-d\right]}{AM\sqrt{K}(\xi d+\tau-d)}+\frac{3A^4\xi L^2(\tau-d+d\xi)}{MK^3(\xi d+\tau-d)}\frac{\xi^2}{1-\xi^2}\left\|\sum_{i=1}^{d-1}g(\mathbf{X}_{d-1})\right\|_F^2 \\
&+\frac{6A^4L^2\sigma^2}{K^2(\xi d+\tau-d)}\left(\tau\frac{\xi^2}{1-\xi^2}(\tau-d+\xi d)+(\tau-d)^2+\xi d(\tau-1)\right)
\end{aligned}$$

If the total iterations $K$ is sufficiently large, then the average-squared gradient norm will be bounded by

$$\mathbb{E}_{K(k)}\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\bigtriangledown F(\mu_{K(k)})\right\|^2\right]\leq\frac{2M\left[F(\mu_1)-F_{inf}\right]+2MLA^2\sigma^2\left[\xi^2d+\tau-d\right]}{AM\sqrt{K}(\xi d+\tau-d)}$$

.