Neurocomputing 483 (2022) 474–487

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

EventGraD: Event-triggered communication in parallel machine learning

Soumyadip Ghosh *, Bernardo Aquino, Vijay Gupta

Department of Electrical Engineering, University of Notre Dame, USA

ARTICLE INFO

Article history:
Received 12 March 2021
Revised 28 July 2021
Accepted 8 August 2021
Available online 2 November 2021

Keywords:
Machine learning
Event-triggered communication
Parallel computing

ABSTRACT

Communication in parallel systems imposes significant overhead which often turns out to be a bottleneck in parallel machine learning. To relieve some of this overhead, in this paper, we present EventGraD - an algorithm with event-triggered communication for stochastic gradient descent in parallel machine learning. The main idea of this algorithm is to modify the requirement of communication at every iteration in standard implementations of stochastic gradient descent in parallel machine learning to communicating only when necessary at certain iterations. We provide theoretical analysis of convergence of our proposed algorithm. We also implement the proposed algorithm for data-parallel training of a popular residual neural network used for training the CIFAR-10 dataset and show that EventGraD can reduce the communication load by up to 60% while retaining the same level of accuracy. In addition, EventGraD can be combined with other approaches such as Top-K sparsification to decrease communication further while maintaining accuracy.

1. Introduction

Artificial intelligence in general, and machine learning in particular, is revolutionizing many aspects of our life [1]. Machine learning (ML) algorithms in various applications have achieved significant benefits through training of a large number of parameters using huge data sets. Focus has now shifted to ensure that these algorithms can be executed for complex problems in a reasonable amount of time. Initial speed-ups in ML algorithms were due to better algorithm design (e.g. using mini-batches) or hardware (e.g. introduction of graphics processing units(GPUs)). However, to stay relevant, machine learning must continue to scale up in the size and complexity of the application problem. The challenge is both in the large number of parameters that need to be trained and the consequent large amount of data that needs to be processed.

An obvious answer is to go from one processing element – that may have neither the memory nor the computational capability needed for machine learning implementations to solve complex problems – to multiple processing elements (sometimes referred to as parallel or distributed implementations) [2,3]. For instance, there has been a lot of recent interest in machine learning using artificial neural networks on large-scale clusters such as supercomputers [4,5]. Both data-parallel (in which the dataset is divided into multiple processors, with each processor having a copy of the entire neural network model) and model-parallel (in which the neural network model is divided among multiple processors, with each processor having access to the entire dataset) architectures have been considered [6]. For some applications such as federated learning which involves edge devices such as smartphones or smart speakers, distributed training is often the only choice due to data privacy concerns [7].

One of the biggest challenges of training in any parallel or distributed environment is the overhead associated with communication between different processors or devices. In high performance computing clusters, communication of messages over networks often takes a lot of time, consumes significant power and can lead to network congestion [8–10]. Specifically for parallel machine learning, during training, the processors need to exchange the weights and biases with each other before moving to the next training iteration. For example, in a data parallel architecture, the weights and biases among the different processors are averaged with each other (either directly or through a central parameter server) before executing the next training iteration. Such an exchange usually happens by message passing at the end of every iteration. As the number of processing elements increases, the issue of such communication being a major bottleneck in these implementations is known widely [11–14].

Consequently there has been a lot of research aimed at reducing communication in parallel machine learning [11–15]. This rich stream of work has suggested various ways of reducing the size or number of messages as means of alleviating the communication

* Corresponding author.
E-mail addresses: sghosh2@nd.edu (S. Ghosh), bcruz2@nd.edu (B. Aquino), vgupta2@nd.edu (V. Gupta).

https://doi.org/10.1016/j.neucom.2021.08.143
0925-2312/

overhead. In this paper, we propose a novel algorithm to reduce communication in parallel machine learning involving artificial neural networks. Specifically, we utilize the idea of event-triggered communication from control theory to design a class of communication-avoiding machine learning algorithms. In this class of algorithms, communication among the processing elements occurs intermittently and only on an as-needed basis. This leads to a significant reduction in the number of messages communicated among the processing elements. Note that algorithms to reduce communication have been proposed in other applications of parallel computing as well, such as parallel numerical simulation of partial differential equations [16–18].

The core idea of our algorithm is to exchange the neural network parameters (the weights and biases) only when a certain criterion is satisfied, i.e., in an event-triggered fashion. We present both theoretical analysis and experimental demonstration of this algorithm. Experimentally, we show that the algorithm can yield the same accuracy as standard implementations with 60% lesser number of messages communicated among processors using a popular residual neural network on the CIFAR-10 dataset. Our implementation is open-source and available at [19]. A reduction in the number of messages implies a reduction in both the time and energy overhead of communication and can also prevent congestion in the network. Theoretically, we show that our algorithm has a bound on the convergence rate (in terms of number of iterations) of the order $\mathcal{O}\left(\frac{1}{\sqrt{Kn}}+\frac{G(K-1)}{\sqrt{K}}+\frac{G_{1/2}^2(K-1)}{\sqrt{K}}\right)$ where K is number of iterations, n is number of processors, and $G(K)$ and $G_{1/2}(K)$ are terms related to a bound on the threshold of event-triggered communication. In particular, if the bound on the threshold is chosen to be a sequence that decreases geometrically as a function of the iteration number, the bound on the convergence rate becomes of the order $\mathcal{O}\left(\frac{1}{\sqrt{Kn}}+\frac{1}{\sqrt{K}}\right)$ which is similar to the asymptotic convergence rate of parallel stochastic gradient descent in general. An earlier version of this algorithm without theoretical results was experimentally demonstrated as a proof of concept on the MNIST dataset in [20]. In contrast, this paper contains a comprehensive theoretical treatment of the algorithm with additional experiments on the CIFAR-10 dataset. In our previous work [18], we have considered event-triggered communication for a different domain of parallel numerical partial differential equation solvers and highlighted the implementation challenges similar to this paper. However we considered a fixed threshold without any mathematical treatment in that work unlike the adaptive threshold along with theoretical convergence results studied in this paper.

While we focus on data-parallel stochastic gradient descent in parallel machine learning for theoretical analysis and experimental verification of the algorithm in this paper, the idea of event-triggered communication can be applied to model-parallel and hybrid configurations and can be extended to other training algorithms as well such as Adam, RMSProp, etc. Similarly, event-triggered communication can also be used in federated learning where communication can have a more severe overhead due to the geographical separation between the devices involved in training such as smartphones.

The paper is organised as follows. Section 2 surveys related work and Section 3 introduces the necessary background. The proposed algorithm is introduced in Section 4 with theoretical analysis in Section 5 and implementation details in Section 6. Section 7 contains the experimental results followed by conclusion in Section 8. For notational convenience, we denote the abbreviation PE to be a processing element that signifies one core of a processor.

2. Related work

In this section, we review some communication-efficient strategies of distributed training of neural networks from literature. We primarily focus on parallel stochastic gradient descent [21,22]. We also review some works on event-triggered communication and then highlight our specific contributions on using event-triggered communication for parallel training of neural networks.

Parameter Server – A popular approach for parallelization of stochastic gradient descent is the centralized parameter server approach where multiple workers compute the gradients in their assigned sub-dataset and send them to a central parameter server. The parameter server updates the neural network model parameters using the individual gradients and sends the updated parameters back to the workers, who then move on to the next iteration. The original approach results in a synchronized algorithm. This requirement of synchronization was relaxed by the Hogwild algorithm [23] where the worker PEs can send gradients to the parameter server asynchronously without any lock step. Elastic Averaging SGD proposed by [24] reduces communication by introducing the notion of an elastic period of communication between the workers and the parameter server. Other approaches have also been proposed in the literature [25]. There have been studies to reduce communication with the parameter server in the context of federated learning as well [26–28]. However, the parameter server approach often suffers from poor scalability due to the dependence on a central node, which can become a bottleneck.

AllReduce – Another popular approach for parallelization of stochastic gradient descent does not consider a centralized parameter server. Rather, every PE maintains a copy of the model and the PEs average the parameters of the model by communicating in an all-to-all fashion among themselves using a reduction mechanism commonly known as AllReduce [29]. Since such all-to-all communication incurs a lot of overhead, a lot of research has focused on reducing this overhead by using optimized variants. The authors in [14] have proposed one-bit quantization where each gradient update is quantized to 1-bit, resulting in a reduction of the data volume that needs to be communicated. Threshold quantization was developed in [30] where only those gradient updates that are greater than a specified threshold are transmitted after encoding them with a fixed value. A hybrid approach combining both 1-bit and threshold quantization was given in the adaptive quantization proposed in [31]. Deep Gradient Compression in [13] compresses the size of gradients and accumulates the quantization error and momentum to maintain accuracy. Several approaches have been proposed to minimize communication by reducing the precision of gradients, e.g., using half precision (16-bit) for training [32] and mixed precision [33]. Sparsified methods that communicate only the top-k most significant values have been proposed by [34,35]. Combining the two methods of quantization and sparsification is presented in [36].

Reduction with neighbors – Instead of averaging the parameters using all-to-all communication among all the PEs via AllReduce, another approach has been proposed where the averaging is done only with the neighboring PEs in the topology in which the PEs are connected [37]. This approach uses ideas from consensus algorithms which is now widely studied in many different communities [38]. We would like to point out that there is confusion in literature about the term "decentralized" - some works call the AllReduce approach involving averaging among all PEs as decentralized because of the absence of a central parameter server [39], while others call the approach that involves averaging with just the neighboring PEs as decentralized because it does not

require any central operation on all the PEs in the topology [37,40]. We adopt the latter usage and call the algorithms that require averaging with just the neighboring PEs as decentralized. While it might seem that such a scheme will converge slower than a centralized approach due to the delayed dissemination of information to all the nodes, the authors in [40] showed that the convergence rate is similar in order to the centralized approach after the same number of iterations provided the number of iterations is sufficiently large. While [40] considers that the neighbors of a particular PE remain fixed across iterations, there are interesting gossip algorithms [41–43] that choose neighbors randomly and exchange information. More recently, the authors in [15] propose an error-compensated communication compression mechanism in this context using bit-clipping that reduces communication costs.

Event-Triggered Communication – Event-Triggered Communication has been proposed as a mechanism for reducing communication in networked control systems [44,45]. Methods employing event-triggered communication in consensus algorithms have been variously proposed [46–48]. Closely related to consensus is the problem of distributed optimization where there have been various event-triggered approaches proposed [49–51]. Of particular relevance is [52] which suggests an event-triggered communication scheme with an adaptive threshold of communication that is dependent on the state of the last trigger instant in a continuous time control system.

Our Contribution – The decentralized parallel stochastic gradient descent in [40] still considers that communication of parameters with neighbor PEs happens at every iteration. Since the values of the parameters may not change significantly in every iteration, communication at every iteration may not be necessary. Thus the main idea behind the algorithms presented in this paper is to communicate these parameters in events only when their values change by a certain threshold. We consider the scenario of data-parallel training of a neural network in a high performance computing (HPC) cluster where there are fixed neighboring PEs for every PE and show that communicating in events with neighboring PEs reduces the number of messages passed in the network. Decreasing the message count decreases the overall data to be communicated and as pointed out in literature [11–15], reducing the data to be communicated reduces the overhead associated with communication. More concretely, the contributions of our work are:

- We propose an event-triggered communication algorithm where the neural network parameters, i.e., the weights and biases, are communicated only when their norm changes by some threshold. The threshold is chosen in an adaptive manner based on the rate of change of the parameters.
- We derive an expression for a bound on the convergence rate of event-triggered communication based on a generic bound on the adaptive threshold.
- We provide an open-source high performance computing (HPC) implementation of our algorithm using PyTorch and Message Passing Interface (MPI) in C++. We also highlight implementation challenges of this algorithm, particularly the need for advanced features such as one-sided communication, also called remote memory access and the requirement of the newer PyTorch C++ frontend over its traditional Python frontend. We believe that it is not possible to implement event-triggered communication without remote memory access in any computer network as elaborated later in Section 6. Our implementation is open-source and available at [19].

The paper closest to ours seems to be [53] where the authors considered a federated learning scenario and proposed an event-triggered communication scheme for the model parameters based on thresholds that are dependent on the learning rate and showed reduction in communication for distributed training. As compared to that work, we consider an adaptive threshold rather than selecting the same threshold across all parameters. In particular, the threshold is adaptive to the local slope of a parameter and thus it can adjust according to the parameter's evolution which will depend on factors such as the type of the parameter, the neural network model and the dataset. Hence the adaptive threshold makes our algorithm robust to different neural network models and different datasets. Our theoretical results are based on a generic bound on the threshold unlike [53] which provides a bound considering a certain form of threshold dependent on the learning rate. Further, we highlight the implementation challenges of event-triggered communication in an HPC environment which is different than the federated learning setting considered in [53] that usually involves wireless communication.

3. Problem formulation

This section lays the mathematical preliminaries for the main algorithm introduced in the next section. We consider a decentralized communication graph (V, W) where V denotes the set of n PEs and $W \in \mathbb{R}^{n\times n}$ is the symmetric doubly stochastic adjacency matrix. W_{ii} corresponds to the weight of the state of the i-th PE while W_{ij} corresponds to the weight of the state of the j-th PE on the state of the i-th PE. We assume that W represents a ring topology that remains fixed throughout iterations. Now the objective of data-parallel training of any neural network can be expressed as

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi\sim\mathbb{D}} F_i(x;\xi)}_{:=f_i(x)} \tag{1}$$

where $\mathbb{D}$ is the sampling distribution, considered to be the same in every PE. Further, the neural network in every PE is considered to have N parameters. These parameters are the weights and biases in the neural network model.

For mathematical formulation, let us define the concatenation of all local parameters X_k, random samples ξ_k, stochastic gradients $\partial F(X_k;\xi_k)$ and expected gradients $\partial f(X_k)$ at iteration k as:

$$\begin{aligned}
&X_k := \begin{bmatrix} x_{k,1} & \cdots & x_{k,n}\end{bmatrix} \in \mathbb{R}^{N\times n}, \quad \xi_k := \begin{bmatrix}\xi_{k,1} & \cdots & \xi_{k,n}\end{bmatrix}^\top \in \mathbb{R}^n,\\
&\partial F(X_k;\xi_k) := \begin{bmatrix}\nabla F_1(x_{k,1};\xi_{k,1}) & \nabla F_2(x_{k,2};\xi_{k,2}) & \cdots & \nabla F_n(x_{k,n};\xi_{k,n})\end{bmatrix} \in \mathbb{R}^{N\times n},\\
&\partial f(X_k) := \begin{bmatrix}\nabla f_1(x_{k,1}) & \nabla f_2(x_{k,2}) & \cdots & \nabla f_n(x_{k,n})\end{bmatrix} \in \mathbb{R}^{N\times n}.
\end{aligned}$$

The algorithm is said to converge to a e-approximate solution if

$$K^{-1}\left(\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2\right) \leqslant e,$$

where $\mathbf{1}_n$ represents a column vector of $\mathbf{1}$'s.

Now the training algorithm for the decentralized stochastic gradient descent mentioned in [40] can be expressed as:

$$X_{k+1} = X_k W - \gamma\partial F(X_k;\xi_k), \tag{2}$$

where γ is the step size or learning rate. From (2), it is clear that values from neighbor PEs are needed to calculate the values in a particular PE. Thus the parameters of the neural network, i.e., the weights and biases, are communicated between the neighbor PEs after every iteration. For details on how to choose W optimally, the reader is referred to [54]. Usually the values of W are taken to be $\frac{1}{\mathcal{N}_i+1}$ where $\mathcal{N}_i$ are the number of neighbors of the i-th PE. This means that the parameters at the i-th PE are averaged with that of its neighbors after every iteration. For the ring topology that we assume, $\mathcal{N}_i = 2$ for all i. After training concludes, the models in all the PEs are usually averaged to produce one model which is then evaluated on the test dataset. This algorithm from [40], named D-PSGD in that paper, is stated in pseudo code in Algorithm A. Since

communication between neighboring PEs happen *regularly* after every iteration, we refer to this algorithm as the one with regular communication. We modify this algorithm to include event-triggered communication as proposed in the next section.

Algorithm A: Regular Communication in Data Parallel SGD

for $k = 0, 1, 2, \ldots K-1$ **do**
  Randomly sample from dataset in i-th PE
  Compute the local stochastic gradient
  Communicate parameters to neighbors
  Update parameters using (2)
end for
Obtain averaged model from all PEs

4. Proposed Algorithm: EventGraD

In the decentralized algorithm in (2), the parameters in a PE are exchanged with neighbors in every iteration of the training. This might be a waste of resources since the parameters might not differ a lot in every iteration. Therefore it is possible to relax this requirement of communication with neighbors at every iteration of training. This is the main idea of our algorithm where communication happens *only when necessary* in *events*.

Our algorithm works as follows - Every PE tracks the changes in the parameters of its model. When the norm of a particular parameter in a PE has changed by some threshold, it is sent to the neighboring PEs. At other iterations, that particular parameter is not sent to the neighbors and the neighbors continue updating their own model using the last received version of that parameter.

Fig. 1 illustrates this phenomenon. As an example, the left plot shows the evolution of the norm of a parameter over training iterations. When this norm changes by more than a threshold (0.1 in Fig. 1) from the norm of the previously communicated values, an event for communication is triggered as marked by an asterisk. The first event of communication is forced to take place at iteration $k = 0$ for convenience. The right plot shows the corresponding values that the receiving PE uses when averaging its parameter with the parameter from this corresponding sending PE.

For mathematically describing the algorithm, let us first define the vector of previously communicated values $\widehat{X}_k$ as

$$\widehat{X}_k := \begin{bmatrix} \hat{x}_{k,1} & \cdots & \hat{x}_{k,n} \end{bmatrix} \in \mathbb{R}^{N \times n}. \tag{3}$$

Note that each $\hat{x}_{k,i}$ is a vector of the norm of N parameters, i.e.,

$$\hat{x}_{k,i} = \begin{bmatrix} \hat{x}_{k,i,1} & \ldots & \hat{x}_{k,i,N} \end{bmatrix}^\top \in \mathbb{R}^N. \tag{4}$$

Now the event-triggered condition can be expressed as

$$\hat{x}_{k+1,i,I} = \begin{cases} x_{k+1,i,I} & \text{if } \|\hat{x}_{k,i,I} - x_{k+1,i,I}\| \geqslant \delta_{k,i,I} \\ \hat{x}_{k,i,I} & \text{if } \|\hat{x}_{k,i,I} - x_{k+1,i,I}\| < \delta_{k,i,I}, \end{cases}$$

where $\delta_{k,i,I}$ is the threshold for the I-th parameter in the i-th PE at k-th iteration. Consequently, the training algorithm gets modified from (2) to

$$X_{k+1} = \widehat{X}_k W - \gamma \partial F\left(\widehat{X}_k; \xi_k\right), \tag{5}$$

which represents our algorithm with event-triggered communication. The pseudo code is specified in Algorithm B.

Algorithm B: EventGraD - Event-Triggered Communication in Data Parallel SGD

for $k = 0, 1, 2, \ldots K-1$ **do**
  Randomly sample from dataset in i-th PE
  Compute the local stochastic gradient
  for $I = 1, 2, \ldots, N$ **do**
    if $\|\hat{x}_{k,i,I} - x_{k+1,i,I}\| \geqslant \delta_{k,i,I}$ **then**
      Communicate parameter to neighbors
    end if
  end for
  Update parameters using (5)
end for
Obtain averaged model from all PEs

Choosing the threshold $\delta_{k,i,I}$ is a design problem. The efficiency of this algorithm depends on selecting appropriate thresholds. The simplest option would be to choose the same value of threshold for all the parameters in all the PEs as was done in [53]. However, selecting the appropriate value would involve a lot of trial and error. Further, when the neural network model changes, the process would have to be repeated all over again. More importantly, the parameters in a model and across different PEs would vary differently and selecting the same threshold for all of them is not desired. Instead, it is better to choose a dynamic threshold that is adaptive to the rate of change of the parameters. A metric that is indicative of the rate of change of a parameter is the local

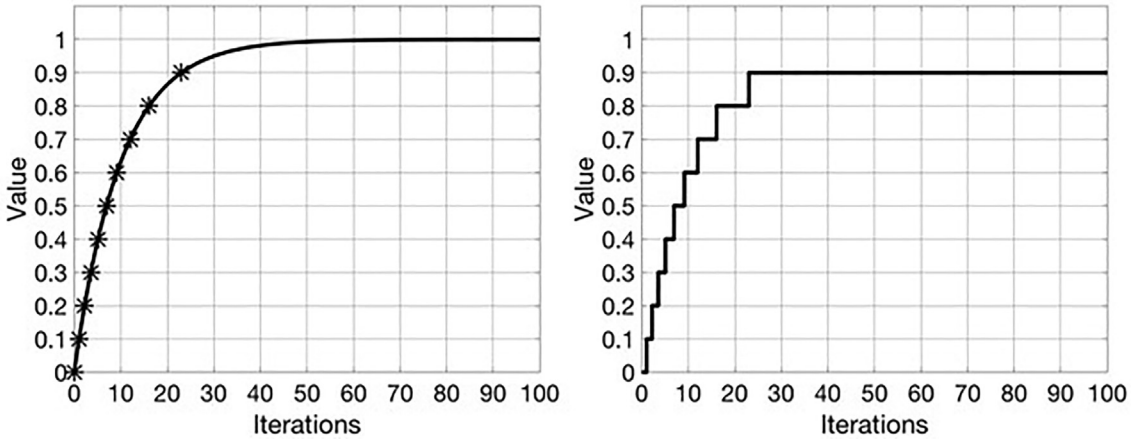

Fig. 1. Illustration of change in norm of parameters over iterations (taken from [18]). The left plot shows the norm of the parameter over iterations at the sender. The right plot shows the norm of that corresponding parameter used at the receiver.

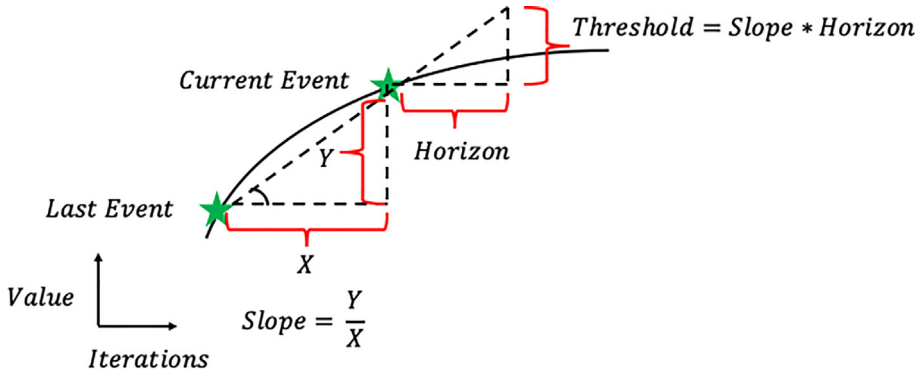

Fig. 2. Illustration of slope-based adaptive threshold. The right green star denotes the event of current communication while the left green star denotes the event of last communication. The slope is calculated between these two points which is then multiplied by the horizon to obtain the new adaptive threshold.

slope of the norm of the parameter. Thus we choose the threshold of a parameter based on the local slope of its norm. Whenever an event of communication is triggered, the slope is calculated between the current norm and the norm at the last event. This slope is multiplied by a horizon h to calculate the threshold as illustrated in Fig. 2. This threshold will be kept fixed until the next event is triggered, resulting in calculation of a new threshold. Thus we obtain:

$$\delta_{k,i,I} = \underbrace{\frac{\|\hat{x}_{k,i,I} - x_{k+1,i,I}\|}{k - \hat{k}}}_{\text{Slope}} \times h, \tag{6}$$

where $\hat{k}$ is the iteration corresponding to $\hat{x}_{k,i,I}$, i.e., when the last value was communicated.

The intuition behind making the threshold dependent on the slope is to ensure it is chosen according to the trend of evolution of the parameter. This helps on saving communication as much as possible while ensuring that communication does not stop, i.e., happens once in a while. If a parameter is changing fast, that means that it will satisfy the criterion for communication soon - thus a high threshold (due to the high slope) can be suitable. However, if the parameter is changing slowly, there might a long period before the next communication happens which might slow down convergence of the overall algorithm. Hence the threshold is decreased (due to the low slope) to incentivize communication.

The horizon h is a hyperparameter that is chosen by the user. Its purpose is to serve as a *look-ahead* to calculate the next threshold. It might seem that h requires tuning as well, thereby nullifying its advantages over the static threshold. However, the same value of h can be chosen for the different parameters because the threshold is already modulated by the slope. If the neural network model is changed due to change in the depth, width or type of layers, the threshold will adjust accordingly. Choosing a different dataset where the data follows a different distribution is also likely to change the evolution of the neural network parameters which the adaptive threshold can capture. Thus the adaptive threshold selection mechanism plays a huge role in keeping our algorithm EventGraD portable as much as possible across multiple models and multiple datasets.

5. Analysis

The theoretical convergence properties of the proposed algorithm are studied in this section. Let us consider the error or difference between the last communicated state and the current state as:

$$\epsilon_{k,i,I} = \hat{x}_{k,i,I} - x_{k,i,I}, \tag{7}$$

$$\Rightarrow \mathscr{E}_k = \widehat{X}_k - X_k, \tag{8}$$

where $\mathscr{E}_k = \begin{bmatrix}\epsilon_{k,1} & \dots & \epsilon_{k,n}\end{bmatrix} \in \mathbb{R}^{N\times n}$. According to our algorithm, the error is bounded by the corresponding threshold as

$$\|\epsilon_{k,i,I}\| \leqslant \delta_{k,i,I}. \tag{9}$$

Since $\delta_{k,i,I}$ is different for different i and different I, considering the different values for any theoretical analysis seems intractable. Rather we consider the following assumption:

Assumption 1. The thresholds $\delta_{k,i,I}$ can be bounded by a function dependent on only k as

$$\|\delta_{k,i,I}\|^2 \leqslant g(k). \tag{10}$$

Assumption 1 makes analysis of the convergence properties of the algorithm feasible by considering a bound on thresholds for all parameters in all PEs. Further, we consider the following assumptions that are usually used for analysis of SGD algorithms.

Assumption 2. The following assumptions hold:

1. Lipschitz Gradient: All functions $f_i(.)$'s have L-Lipschitz Gradients.
2. Spectral Gap: Given the symmetric doubly stochastic matrix W, the value $\rho := (\max\{|\lambda_2(W)|, |\lambda_n(W)|\})$ satisfies $\rho < 1$ where λ represents eigenvalues.
3. Bounded Variance: The variance of the stochastic gradient
$$\mathbb{E}_{i\sim\mathscr{U}_{i([n])}}\mathbb{E}_{\xi\sim\mathbb{D}_i}\|\nabla F_i(x;\xi) - \nabla f(x)\|^2$$
is bounded for any x with i sampled uniformly from $\{1,\dots,n\}$ and ξ from the distribution $\mathbb{D}_i$. That is, there are constants σ and ς, such that:
$$\mathbb{E}_{\xi\sim\mathbb{D}_i}\|\nabla F_i(x;\xi) - \nabla f_i(x)\|^2 \leqslant \sigma^2, \forall i, \forall x,$$
$$\mathbb{E}_{i\sim\mathscr{U}_{i([n])}}\|\nabla f_i(x) - \nabla f(x)\|^2 \leqslant \varsigma^2, \forall x.$$
4. We start from $X_0 = 0$ without loss of generality.

Let

$$C_1 = \left(\frac{1-\gamma}{2} - \frac{72\gamma^3}{C_2\left(1-\sqrt{\rho}\right)^2}L^2\right),\ C_2 = \left(1 - \frac{36\gamma^2}{\left(1-\sqrt{\rho}\right)^2}nL^2\right)$$
$$G(K) = \sum_{k=0}^{K} g(k),\ G_{1/2}(K) = \sum_{k=0}^{K}\sqrt{g(k)} \tag{11}$$

Theorem 1. *Considering the assumptions, we obtain the following convergence rate for the algorithm*

$$\frac{C_1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2+\frac{\gamma-\gamma^2L}{2K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\frac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\|^2\leqslant\frac{f(0)-f^*}{K}+\frac{\gamma^2L\sigma^2}{2n}$$
$$+\left(12C_2^{-1}\gamma^3nL^2\left(2L^2+1\right)+\frac{3\gamma L^2+L+1}{2K}+\frac{72\gamma^3L^4}{KC_2\left(1-\sqrt{\rho}\right)^2}\right)G(K-1)$$
$$+C_2^{-1}\gamma\rho L^2G_{1/2}^2(K-1)+\frac{2n\gamma^3\sigma^2L^2}{C_2(1-\rho)}+\frac{18n\gamma^3\varsigma^2L^2}{C_2\left(1-\sqrt{\rho}\right)^2} \tag{12}$$

Proof. Provided in appendix. □

Theorem 1 represents the convergence of the average of the models in all PEs. In order to obtain a closer result, we consider an appropriate learning rate and then state the following corollary:

Let

$$C_3=\frac{\left(1-\sqrt{\rho}\right)^2\left(2L^2+1\right)}{6\rho L^2},\ C_4=\frac{7L^2+L+1}{2} \tag{13}$$

Corollary 1. *Under the same assumptions as in Theorem 1, if we set* $\gamma=\frac{1}{2\rho L^2\sqrt{K}+\sigma\sqrt{K/n}}$*, we have the following convergence rate:*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2\leqslant(2(f(0)-f^*)+L)\left(\frac{1}{K}+\frac{1}{\sqrt{Kn}}\right)$$
$$+\left(\frac{2C_3}{\sqrt{K}}+\frac{2C_4}{K}\right)G(K-1)+\frac{2}{\sqrt{K}}G_{1/2}^2(K-1)$$

if the total number of iterations K is large enough, in particular,

$$K\geqslant\frac{4n^3L^2}{\sigma^3\left(f(0)-f^*+L/2\right)}\left(\frac{\sigma^2}{(1-\rho)}+\frac{9\varsigma^2}{\left(1-\sqrt{\rho}\right)^2}\right),\text{ and}$$
$$K\geqslant\frac{72L^2n^2}{\sigma^2\left(1-\sqrt{\rho}\right)^2},\text{ and}$$
$$K\geqslant\left(\frac{\sqrt{n}(L+1)}{2\rho L^2\sqrt{n}+\sigma}\right)^2$$

Proof. Provided in appendix. □

Corollary 1 shows that the bound on the convergence rate is dependent on the threshold related terms $G(K)$ and $G_{1/2}(K)$. When K is large enough, the $\frac{1}{K}$ terms will decay faster than the $\frac{1}{\sqrt{K}}$ terms and therefore the convergence rate is of the order $\mathcal{O}\left(\frac{1}{\sqrt{Kn}}+\frac{G(K-1)}{\sqrt{K}}+\frac{G_{1/2}^2(K-1)}{\sqrt{K}}\right)$.

Note that a threshold of 0 reduces to the regular algorithm in [40]. Thus with $g(k)=0$, we obtain $G(k)=0$ and $G_{1/2}(k)=0$ and hence the rate of convergence reduces to $\mathcal{O}\left(\frac{1}{K}+\frac{1}{\sqrt{Kn}}\right)$ which is consistent with [40]. Now we provide a more concrete bound by choosing $g(k)$ according to a popular event-triggered threshold specified in [55].

Corollary 2. *If* $g(k)$ *is chosen of the form* $g(k)=\alpha\beta^k$ *where* α,β *are appropriate constants and* $0<\beta<1$*, then the rate of convergence is of the order* $\mathcal{O}\left(\frac{1}{\sqrt{Kn}}+\frac{1}{\sqrt{K}}\right)$.

Proof. We obtain $G(K-1)=\alpha\left(\frac{1-\beta^{K-1}}{1-\beta}\right)$ and $G_{1/2}^2(K)=\alpha\left(\frac{1-\sqrt{\beta}^{K-1}}{1-\sqrt{\beta}}\right)^2$. The corollary then follows by noting that $\left(\frac{2C_3}{\sqrt{K}}+\frac{2C_4}{K}\right)\alpha\left(\frac{1-\beta^{K-1}}{1-\beta}\right)\sim\mathcal{O}\left(\frac{1}{\sqrt{K}}\right)$ and $\frac{2}{\sqrt{K}}\alpha\left(\frac{1-\sqrt{\beta}^{K-1}}{1-\sqrt{\beta}}\right)^2\sim\mathcal{O}\left(\frac{1}{\sqrt{K}}\right)$ when K is sufficiently large. □

6. Implementation

There are a lot of popular frameworks for machine learning like PyTorch, TensorFlow, CNTK, etc. Almost all of these frameworks support parallel or distributed training. TensorFlow follows the parameter server approach for parallelization. PyTorch provides a module called DistributedDataParallel that implements AllReduce based training. Horovod is another framework developed by Uber that implements an optimized AllReduce algorithm. However, none of these frameworks provide native support for the training involving averaging with just neighbors. Hence we decided to implement the proposed algorithm without using any of the distributed modules in these frameworks.

We use PyTorch and MPI for our implementation. First, we point out why one-sided communication or remote memory access is necessary. Usually communication in high performance computing networks is two-sided. In other words, the sending PE starts the communication of a message by invoking a MPI_Send operation and then the receiving PE completes the communication and receives the message by invoking a MPI_Recv operation [29]. In our event-triggered communication algorithm, the events for communication are dependent on the change in values of the parameters of the sender which is a local phenomenon. Thus, when an event is triggered in the sending PE, it can issue a MPI_Send operation. However, since the intended receiving PE is not aware of when the event is triggered at the sender, it does not know when to issue a MPI_Recv operation. So two-sided communication using MPI_Send and MPI_Recv cannot be used for our algorithm.

Hence we select one-sided communication for our purpose. In one-sided communication, only the sending PE has to know all the parameters of the message for both the sending and receiving side and can remotely write to a portion of the memory of the receiver without the receiver's involvement - hence the alternate name of Remote Memory Access [56]. That region of memory in the receiver is called *window* and can be publicly accessed. In our case, it is used to store the model parameters from the neighbors. So when an event for communication is triggered in the sending PE, it uses MPI_Put to write its model parameters directly into the window of the corresponding neighbor PE. An illustration of one-sided vs two-sided communication is provided in Fig. 3.

It is worth noting that PyTorch does not support one-sided communication at this point. Recently, PyTorch released a C++ frontend called Libtorch which can be integrated with traditional C++ MPI implementations. Further, the C++ frontend is more suitable for HPC environments unlike the Python frontend. Hence we combine the neural network training functionalities of Libtorch with communication routines in MPI to implement our algorithm. For further details on our implementation, the reader is referred to [20].

7. Results

We perform experiments to evaluate the performance of our algorithm. All our simulations are done on CPUs. We use an HPC cluster of nodes with each node having 2 CPU Sockets of AMD's EPYC 24-core 2.3 GHz processor and 128 GB RAM per node. The cluster uses Mellanox EDR interconnect. The MPI library chosen is Open MPI 4.0.1 compiled with gcc 8.3.0. The version of Libtorch used is 1.5.0. We conduct our experiments on the CIFAR-10 dataset. We choose the residual neural network commonly used for training on CIFAR-10 [57]. Our simulations use the ResNet-18 configuration. For training this network, a learning rate of 0.01 is used with cross-entropy as the loss function and a mini-batch size of 256. The value of horizon h used to calculate the threshold in (6) is taken to be 1. Note that we performed experiments on the MNIST dataset in our previous work [20].

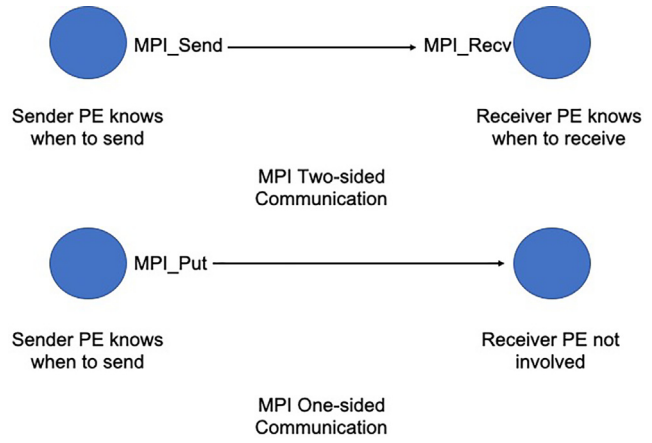

Fig. 3. Illustration of the difference between two-sided and one-sided communication.

To illustrate our adaptive event-triggered threshold selection scheme, we look at how the norm of the parameters, i.e., the weights and biases, in the ResNet-18 model change with iterations. Note that the ResNet-18 model has a lot of parameters, all of which cannot be shown in this paper due to space constraints. Therefore we show a few parameters which vary in their style of evolution in Fig. 4. After few initial oscillations, the change of the values is gradual which suggests that not all parameters need to be communicated at every iteration. This paves the way for saving on communication of messages by event-triggered communication. The corresponding threshold evolution as calculated by the equation in (6) is shown in Fig. 5. Since the threshold is proportional to the local slope, we see in parameter 1 and 3 that higher slopes during the early iterations of training lead to higher thresholds followed by a decrease in threshold due to decrease in slope. For parameters 2 and 5, which stay relatively flat, the threshold also follows a flat trend. It is important to note that since every parameter changes differently, their thresholds also vary accordingly.

The thresholds in Fig. 5 have an oscillatory behavior. This is due to the fact that often the parameters in a neural network have local

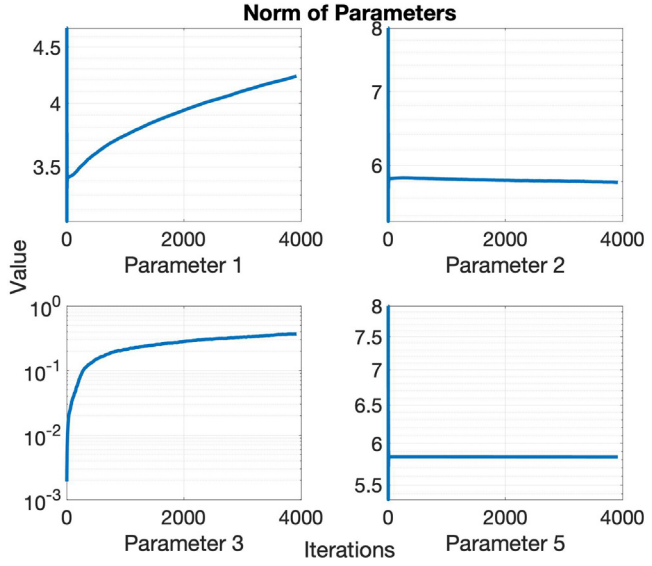

Fig. 4. Plot showing the evolution of the norm of the parameters of the neural network in a certain PE. Note that the parameters may vary in their trend of evolution.

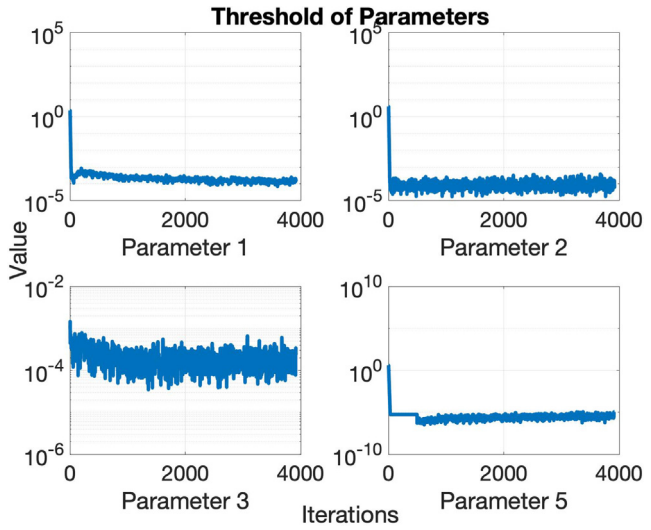

Fig. 5. Plot showing the adaptive threshold of the parameters shown in Fig. 4. The trend of the threshold is adaptive to the trend of evolution of the corresponding parameter.

minor oscillations because of the nature of the stochastic gradient descent algorithm. The parameters in Fig. 4 have these local oscillations, however they are not prominently visible due to the higher scale of the plot. Further, the stochastic nature of the MPI one-sided implementation of the algorithm amplifies the oscillations. It is desired that the threshold reflect the aggregate trend of evolution in the parameter and not the local oscillations. In order to solve this issue, the sender can keep a history of multiple previously communicated events instead of just one previous event. Then the average slope is calculated which is the mean of the slopes between two consecutive events in that history. This average slope is then multiplied by the horizon to obtain the threshold. The length of the history is a hyperparameter which is similar in notion to the length of a moving average filter. The higher the length, the smoother the trend but at the cost of increased computational complexity. For our experiments, we choose the length of this history to be 2.

Having described details of selecting the threshold, we now look at the experimental convergence properties of the algorithm. We compare our event-triggered communication algorithm proposed in Algorithm B with respect to the regular communication algorithm in Algorithm A from [40]. Fig. 6 shows the loss function over epochs for both these algorithms, each repeated for 10 different runs shown by the errorbars. Note that an epoch refers to processing the entire dataset allotted to a PE once while an iteration refers to processing a mini-batch once. Thus one epoch has multiple iterations which depends on the size of the mini-batch. From Fig. 6, we see that the decay in loss function seems similar for both the algorithms, indicating that they have similar speed of convergence. It is important to observe that the theoretical results in Section 5 deal with a bound on convergence of the average of the parameters in all the PEs whereas the plot in Fig. 6 is concerned with experimental convergence of the loss function.

After demonstrating similar rate of convergence, we focus on the main advantage of the event-triggered algorithm over the regular algorithm - reduction in the number of messages communicated while attaining similar accuracy. The reduction in messages is quantified by the percentage of messages of the regular algorithm that is sent in the event-triggered algorithm. Table 1 states the accuracy of regular and event-triggered communication as well

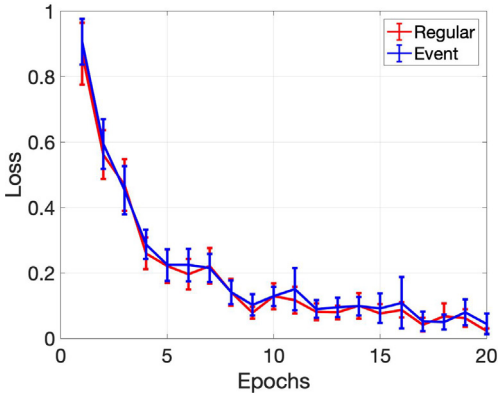

Fig. 6. Plot showing the loss function over epochs. The experiments have been repeated 10 times to account for variations which have been considered in the errorbars. It is seen that the event-triggered communication algorithm has an experimental rate of convergence similar to that of the regular communication algorithm.

as the percentage of messages in event-triggered communication after training for 20 epochs.

We see that the event-triggered communication algorithm exchanges approximately 40% of the messages of the regular (baseline) communication algorithm while achieving similar accuracy. As a reminder, the regular algorithm is the D-PSGD algorithm specified in [40] that is also suitable for the asynchronous decentralized environment that we consider. In other words, our event-triggered algorithm saves around 60% of the messages as compared to the baseline algorithm while maintaining similar accuracy, thus alleviating the communication overhead. Note that the accuracy of both the regular and event-triggered algorithms decrease as the number of PEs increase. This is because as the ring of PEs get larger, messages comprising of neural network parameters require more hops to propagate through the entire ring. Hence, given same number of epochs, the larger ring comprising of more PEs will have lesser accuracy. If the algorithm is run for more epochs on more PEs, the accuracy will not degrade. Additionally, if the number of neighbors of each PE is increased, information can flow sooner, resulting in more accuracy.

A noteworthy feature of our algorithm is that it is complementary to other algorithms for reduced communication that have been proposed in the literature. In other words, our algorithm can be combined with these algorithms. For instance, we can apply the techniques of quantization and sparsification on top of event-triggered communication to get even more savings in communication. It is important to clarify that most of the existing works in literature apply quantization and sparsification in the parameter server or AllReduce architecture [14,34] which is different from the decentralized reduction with just neighbors scenario that we deal with. However, to demonstrate how these approaches can be extended to our decentralized scenario and combined with

Table 1
Comparison of Regular Communication vs Event-Triggered Communication after 20 epochs. The Event-Triggered Communication algorithm drastically reduces the number of messages to be communicated by around 60% while maintaining similar accuracy.

Number of PEs	Regular Accuracy	Event-Triggered Accuracy	Percentage of Messages
4	86.5	87	43.24
8	86.3	86.2	42.98
16	84.9	84.2	45.91
32	82.5	81.9	44.89

Table 2
Combining Top-K% Sparsification with Event-Triggered (ET) Communication for K= 10. The percent of communication is 2 K= 20% of the percent of messages. It is seen that the communication required here is around $\frac{1}{6}$-th of that in Event-Triggered (ET) communication without sparsification while maintaining similar accuracy as in Table 1.

Number of PEs	Sparse ET Accuracy	Percent of Messages	Percent of communication
4	85.4	36.6	7.3
8	85.26	37.7	7.5
16	83.09	37.7	7.5

the event-triggered approach, we focus on the sparsification method of Top-K. Specifically, when an event is triggered, we send just the Top-K percentage of the elements in a parameter, i.e., the weight matrix or the bias vector. Note that for a Top-K percent value of K, 2K percent of messages is being sent because the indices of the Top-K percent elements have to be sent in addition to their values.

Table 2 shows the results of combining Top-K percent sparsification with event-triggered communication. Before we compare the results in Table 2 with Table 1, we note some important points. Firstly, even though all the simulation details of the event-triggered communication are kept the same between Table 1 and Table 2, the percentage of messages sent are different between them. This is because sending just the Top-K elements of a parameter changes the overall evolution of the neural network which in turn leads to different adaptive thresholds and hence different sequence of events. Secondly, we have to consider the percentage of overall communication for Top-K sparsification in contrast to percentage of messages considered in Table 1. This is due to the fact that the objective of Top-K sparsification is to reduce the size of each message sent. Note that in Table 1, the percentage of overall communication is equivalent to the percentage of messages mentioned since entire parameters are sent during events. However, in the case of Top-K in Table 2, the percentage of overall communication is 2 K % of the percentage of messages sent. Now we see that the accuracy in Table 2 remains almost similar to that of Table 1 but the overall communication is approximately 7% of that of the regular (baseline) communication algorithm. This is in contrast to the overall communication of around 40% in the event-triggered algorithm in Table 1 with respect to the baseline. Thus Top-K sparsification combined with event-triggered communication requires around $\frac{1}{6}$-th of the communication required in just event-triggered communication while maintaining similar accuracy.

8. Conclusion

This paper introduces a novel algorithm that reduces communication in parallel training of neural networks. The proposed EventGraD algorithm communicates the model parameters in events only when the value of the parameter changes by a threshold. The choice of the threshold for triggering events is chosen adaptively based on the slope of the parameter values. The algorithm can be applied to different neural network configurations and different datasets. An asymptotic bound on the rate of convergence is provided. The challenges of implementing this algorithm in a high performance computing cluster, such as the requirement of advanced communication protocols and libraries, are discussed. Experiments on the CIFAR-10 dataset show the superior communication performance of the algorithm while maintaining the same level of accuracy.

CRediT authorship contribution statement

Soumyadip Ghosh: Conceptualization, Methodology, Software, Formal analysis, Writing - original draft, Writing - review & editing. **Bernardo Aquino:** Methodology, Formal analysis. **Vijay Gupta:** Supervision, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was supported in part by the University of Notre Dame Center for Research Computing through its computing resources. The work of the authors was supported in part by NSF CBET-1953090.

Appendix A

The proof for the theoretical results in this paper are provided here. First we state some necessary lemmas. Lemma 1 and Lemma 2 are reproduced from [40]. The vector e_i is the one-hot encoded vector.

Lemma 1. *Using Assumption 2, we obtain*

$$\left\|\frac{\mathbf{1}_n}{n} - W^k e_i\right\|^2 \leqslant \rho^k, \forall i \in 1, 2, \ldots, n, k \in \mathbb{N}$$

Proof. Let $W^\infty := lim_{k\to\infty} W^k$. Because of the assumptions, we get $\frac{\mathbf{1}_n}{n} = W^\infty e_i \; \forall \; i$ since W is doubly stochastic and $\rho < 1$. Thus

$$\begin{aligned}\left\|\tfrac{\mathbf{1}_n}{n} - W^k e_i\right\|^2 = & \left\|\left(W^\infty - W^k\right)e_i\right\|^2 \\ & \leqslant \left\|W^\infty - W^k\right\|^2 \|e_i\|^2 \\ & = \left\|W^\infty - W^k\right\|^2 \\ & \leqslant \rho^k.\end{aligned}$$

□

Lemma 2. *Under Assumption 2, the following holds:*

$$\mathbb{E}\left\|\partial f(X_j)\right\|^2 \leqslant \sum_{h=1}^{n} 3\mathbb{E}L^2 \left\|\frac{\sum_{i'=1}^{n} x_{j,i'}}{n} - x_{j,h}\right\|^2 + 3n\varsigma^2 + 3\mathbb{E}\left\|\nabla f\left(\frac{X_j\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2, \forall j$$

Proof. The term $\mathbb{E}\left\|\partial f(X_j)\right\|^2$ is bounded as follows:

$$\begin{aligned}&\mathbb{E}\left\|\partial f(X_j)\right\|^2 \\ &\leqslant 3\mathbb{E}\left\|\partial f(X_j) - \partial f\left(\tfrac{X_j\mathbf{1}_n}{n}\mathbf{1}_n^\top\right)\right\|^2 + 3\mathbb{E}\left\|\partial f\left(\tfrac{X_j\mathbf{1}_n}{n}\mathbf{1}_n^\top\right) - \nabla f\left(\tfrac{X_j\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2 + 3\mathbb{E}\left\|\nabla f\left(\tfrac{X_j\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2 \\ &\leqslant 3\mathbb{E}\left\|\partial f(X_j) - \partial f\left(\tfrac{X_j\mathbf{1}_n}{n}\mathbf{1}_n^\top\right)\right\|_F^2 + 3n\varsigma^2 + 3\mathbb{E}\left\|\nabla f\left(\tfrac{X_j\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2 \\ &\leqslant \sum_{h=1}^{n} 3\mathbb{E}L^2 \left\|\frac{\sum_{i'=1}^{n} x_{j,i'}}{n} - x_{j,h}\right\|^2 + 3n\varsigma^2 + 3\mathbb{E}\left\|\nabla f\left(\tfrac{X_j\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2.\end{aligned}$$

□

Lemma 3. *For any two vectors a, b, the following is satisfied*

$$\|a+b\|^2 \leqslant 2\|a\|^2 + 2\|b\|^2 \tag{14}$$

Proof. We start with $\|a+b\|^2 = \|a\|^2 + 2\langle a, b\rangle + \|b\|^2$. Now

$$\langle a, b\rangle < \sqrt{\|a\|^2\|b\|^2} < \frac{\|a\|^2 + \|b\|^2}{2}$$

where the first is Cauchy–Schwarz inequality and the second is the geometric mean-arithmetic mean inequality. Substituting the above, the inequality follows. □

Proof to Theorem 1. We begin with $f\left(\frac{X_{k+1}\mathbf{1}_n}{n}\right)$:

$$\begin{aligned}\mathbb{E}f\left(\tfrac{X_{k+1}\mathbf{1}_n}{n}\right) &= \mathbb{E}f\left(\tfrac{\widehat{X}_k W\mathbf{1}_n}{n} - \gamma\tfrac{\partial F\left(\widehat{X}_k;\xi_k\right)\mathbf{1}_n}{n}\right) \\ &= \mathbb{E}f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n} - \gamma\tfrac{\partial F\left(\widehat{X}_k;\xi_k\right)\mathbf{1}_n}{n}\right) \\ &\leqslant \mathbb{E}f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right) - \gamma\mathbb{E}\left\langle\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right), \tfrac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\rangle + \tfrac{\gamma^2 L}{2}\mathbb{E}\left\|\sum_{i=1}^{n}\tfrac{\nabla F_i\left(\hat{x}_{k,i};\xi_{k,i}\right)}{n}\right\|^2\end{aligned} \tag{15}$$

where the previous step comes from the general Lipschitz property $f(y) < f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$. The last term above is the second order moment of $\sum_{i=1}^{n}\frac{\nabla F_i\left(\hat{x}_{k,i};\xi_{k,i}\right)}{n}$. Now we can write

$$\mathbb{E}\left\|\sum_{i=1}^{n}\frac{\nabla F_i\left(\hat{x}_{k,i};\xi_{k,i}\right)}{n}\right\|^2 = \mathbb{E}\left\|\sum_{i=1}^{n}\frac{\nabla F_i\left(\hat{x}_{k,i};\xi_{k,i}\right) - \nabla f_i\left(\hat{x}_{k,i}\right)}{n}\right\|^2 + \mathbb{E}\left\|\sum_{i=1}^{n}\frac{\nabla f_i\left(\hat{x}_{k,i}\right)}{n}\right\|^2. \tag{16}$$

Applying (16) in (15), we obtain:

$$\begin{aligned}&\mathbb{E}f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right) - \gamma\mathbb{E}\left\langle\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right), \frac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\rangle \\ &+ \frac{\gamma^2 L}{2}\mathbb{E}\left\|\sum_{i=1}^{n}\frac{\nabla F_i\left(\hat{x}_{k,i};\xi_{k,i}\right) - \nabla f_i\left(\hat{x}_{k,i}\right)}{n}\right\|^2 \\ &+ \frac{\gamma^2 L}{2}\mathbb{E}\left\|\sum_{i=1}^{n}\frac{\nabla f_i\left(\hat{x}_{k,i}\right)}{n}\right\|^2\end{aligned} \tag{17}$$

For the second last term, we can show that

$$\begin{aligned}&\mathbb{E}\left\|\sum_{i=1}^{n}\frac{\nabla F_i\left(\hat{x}_{k,i};\xi_{k,i}\right) - \nabla f_i\left(\hat{x}_{k,i}\right)}{n}\right\|^2 \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\mathbb{E}\left\|\left(\nabla F_i\left(\hat{x}_{k,i};\xi_{k,i}\right) - \nabla f_i\left(\hat{x}_{k,i}\right)\right)\right\|^2.\end{aligned} \tag{18}$$

Applying (18) into (17),

$$\mathbb{E}f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)-\gamma\mathbb{E}\left\langle\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right),\frac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\rangle+\frac{\gamma^2L}{2n^2}\sum_{i=1}^{n}\mathbb{E}\left\|\left(\nabla F_i\left(\hat{x}_{k,i};\xi_{k,i}\right)-\nabla f_i\left(\hat{x}_{k,i}\right)\right)\right\|^2$$
$$+\frac{\gamma^2L}{2}\mathbb{E}\left\|\sum_{i=1}^{n}\frac{\nabla f_i\left(\hat{x}_{k,i}\right)}{n}\right\|^2\leqslant \tag{19}$$
$$\mathbb{E}f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)-\gamma\mathbb{E}\left\langle\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right),\frac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\rangle+\frac{\gamma^2L\sigma^2}{2n}+\frac{\gamma^2L}{2}\mathbb{E}\left\|\sum_{i=1}^{n}\frac{\nabla f_i\left(\hat{x}_{k,i}\right)}{n}\right\|^2 \tag{20}$$

Using the property $\langle a,b\rangle=\frac{1}{2}\left(\|a\|^2+\|b\|^2-\|a-b\|^2\right)$, we can rewrite the above as:

$$\mathbb{E}f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)+\frac{\gamma^2L\sigma^2}{2n}-\frac{\gamma}{2}\mathbb{E}\left\|\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)\right\|^2$$
$$+\frac{\gamma^2L-\gamma}{2}\mathbb{E}\left\|\sum_{i=1}^{n}\frac{\nabla f_i\left(\hat{x}_{k,i}\right)}{n}\right\|^2$$
$$+\underbrace{\frac{\gamma}{2}\mathbb{E}\left\|\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)-\frac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\|^2}_{:=T_1} \tag{21}$$

Using the Lipschitz property again, we bound the first term as:

$$\begin{aligned}\mathbb{E}f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)&\leqslant\mathbb{E}f\left(\frac{X_k\mathbf{1}_n}{n}\right)+\mathbb{E}\left\langle\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right),\frac{\mathscr{E}_k1^n}{n}\right\rangle+\frac{L}{2}\left\|\frac{\mathscr{E}_k1^n}{n}\right\|^2\\&\leqslant\mathbb{E}f\left(\frac{X_k\mathbf{1}_n}{n}\right)+\frac{1}{2}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2+\frac{1}{2}\left\|\frac{\mathscr{E}_k1^n}{n}\right\|^2+\frac{L}{2}\left\|\frac{\mathscr{E}_k1^n}{n}\right\|^2\\&=\mathbb{E}f\left(\frac{X_k\mathbf{1}_n}{n}\right)+\frac{1}{2}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2+\frac{L+1}{2}\left\|\frac{\mathscr{E}_k1^n}{n}\right\|^2\\&\leqslant\mathbb{E}f\left(\frac{X_k\mathbf{1}_n}{n}\right)+\frac{1}{2}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2+\frac{L+1}{2}\|\mathscr{E}_k\|_F^2\left\|\frac{1^n}{n}\right\|^2\\&\leqslant\mathbb{E}f\left(\frac{X_k\mathbf{1}_n}{n}\right)+\frac{1}{2}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2+\frac{L+1}{2}ng(k)\frac{1}{n}\\&=\mathbb{E}f\left(\frac{X_k\mathbf{1}_n}{n}\right)+\frac{1}{2}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2+\frac{L+1}{2}g(k)\end{aligned} \tag{22}$$

where we used the inequality $\|\mathscr{E}_k\|_F^2\leqslant ng(k)$. $\|\ \|_F$ is the Frobenius norm.

Now we bound T_1 as:

$$T_1=\mathbb{E}\left\|\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)-\frac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\|^2\leqslant\frac{1}{n^2}\sum_{i=1}^{n}\mathbb{E}\left\|\nabla f_i\left(\frac{\sum_{j=1}^{n}\hat{x}_{k,j}}{n}\right)-\nabla f_i\left(\hat{x}_{k,i}\right)\right\|^2 \tag{23}$$
$$\leqslant\underbrace{\frac{L^2}{n^2}\sum_{i=1}^{n}\mathbb{E}\left\|\frac{\sum_{j=1}^{n}\hat{x}_{k,j}}{n}-\hat{x}_{k,i}\right\|^2}_{:=\widehat{Q}_{k,i}},$$

where $\widehat{Q}_{k,i}$ is the square distance of the broadcasted variable i to the average of all broadcasted local variables. From (7) and Lemma 3, we can conclude that $\widehat{Q}_{k,i}\leqslant 2Q_{k,i}+2Q_{k,i}^{\epsilon}$, i.e.,

$$T_1\leqslant\frac{2L^2}{n^2}\left(\sum_{i=1}^{n}\underbrace{\mathbb{E}\left\|\frac{\sum_{j=1}^{n}x_{k,j}}{n}-x_{k,i}\right\|^2}_{Q_{k,i}}+\sum_{i=1}^{n}\underbrace{\mathbb{E}\left\|\frac{\sum_{j=1}^{n}\epsilon_{k,j}}{n}-\epsilon_{k,i}\right\|^2}_{Q_{k,i}^{\epsilon}}\right) \tag{24}$$

where $\epsilon_{k,i}$ contains $\epsilon_{k,i,I}$ for all parameters $I=\{1,\dots,N\}$. Now we can bound $Q_{k,i}^{\epsilon}$ as:

$$Q_{k,i}^{\epsilon}\leqslant ng(k) \tag{25}$$

which implies

$$\widehat{Q}_{k,i}\leqslant 2Q_{k,i}+2ng(k). \tag{26}$$

Now we need to find a bound for $Q_{k,i}$

$$\begin{aligned}Q_{k,i}&=\mathbb{E}\left\|\frac{\sum_{j=1}^{n}x_{k,j}}{n}-x_{k,i}\right\|^2\\&=\mathbb{E}\left\|\frac{X_k\mathbf{1}_n}{n}-X_ke_i\right\|^2\text{ where }\ e_i\ \text{ is the one-hot encoded vector}\\&=\mathbb{E}\left\|\frac{\widehat{X}_{k-1}W\mathbf{1}_n-\gamma\partial F\left(\widehat{X}_{k-1};\xi_{k-1}\right)\mathbf{1}_n}{n}-\widehat{X}_{k-1}We_i+\gamma\partial F\left(\widehat{X}_{k-1};\xi_{k-1}\right)e_i\right\|^2\\&=\mathbb{E}\left\|\frac{X_{k-1}\mathbf{1}_n+\mathscr{E}_{k-1}\mathbf{1}_n-\gamma\partial F\left(\widehat{X}_{k-1};\xi_{k-1}\right)\mathbf{1}_n}{n}-X_{k-1}We_i-\mathscr{E}_{k-1}We_i+\gamma\partial F\left(\widehat{X}_{k-1};\xi_{k-1}\right)e_i\right\|^2\\&=\mathbb{E}\left\|\frac{X_0\mathbf{1}_n+\sum_{k=0}^{K-1}\mathscr{E}_k\mathbf{1}_n-\gamma\sum_{k=0}^{K-1}\partial F\left(\widehat{X}_k;\xi_k\right)\mathbf{1}_n}{n}-X_0W^ke_i-\sum_{k=0}^{K-1}\mathscr{E}_kWe_i\right.\\&\quad\left.+\gamma\sum_{k=0}^{K-1}\partial F\left(\widehat{X}_k;\xi_k\right)W^{K-1+k}e_i\right\|^2\end{aligned}$$
$$\begin{aligned}X_0=0\ \mathbb{E}&\left\|\sum_{k=0}^{K-1}\frac{\mathscr{E}_k\mathbf{1}_n}{n}-\sum_{k=0}^{K-1}\mathscr{E}_kWe_i-\gamma\sum_{k=0}^{K-1}\partial F\left(\widehat{X}_k;\xi_k\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2\\&\leqslant\mathbb{E}\left\|\sum_{k=0}^{K-1}\mathscr{E}_k\left(\frac{1^n}{n}-We_i\right)\right\|^2+\gamma^2\mathbb{E}\left\|\sum_{k=0}^{K-1}\partial F\left(\widehat{X}_k;\xi_k\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2\\&\leqslant\mathbb{E}\sum_{k=0}^{K-1}\|\mathscr{E}_k\|_F^2\left\|\left(\frac{1^n}{n}-We_i\right)\right\|^2+2\mathbb{E}\sum_{k\neq k'}\|\mathscr{E}_k\|_F\left\|\left(\frac{1^n}{n}-We_i\right)\right\|\|\mathscr{E}_{k'}\|_F\left\|\left(\frac{1^n}{n}-We_i\right)\right\|\\&\quad+\gamma^2\mathbb{E}\left\|\sum_{k=0}^{K-1}\partial F\left(\widehat{X}_k;\xi_k\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2\\&\leqslant\mathbb{E}\sum_{k=0}^{K-1}ng(k)\rho+2\mathbb{E}\sum_{k\neq k'}\sqrt{ng(k)}\sqrt{\rho}\sqrt{\rho}\sqrt{ng(k')}\\&\quad+\gamma^2\mathbb{E}\left\|\sum_{k=0}^{K-1}\partial F\left(\widehat{X}_k;\xi_k\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2\\&\leqslant n\rho\sum_{k=0}^{K-1}g(k)+n\rho\sum_{k\neq k'}2\sqrt{g(k)g(k')}\\&\quad+\gamma^2\mathbb{E}\left\|\sum_{k=0}^{K-1}\partial F\left(\widehat{X}_k;\xi_k\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2\\&\leqslant n\rho\underbrace{\left(\sum_{k=0}^{K-1}\sqrt{g(k)}\right)^2}_{:=\left(G_{1/2}(K-1)\right)^2}\\&\quad+2\gamma^2\underbrace{\mathbb{E}\left\|\sum_{k=0}^{K-1}\left(\partial F\left(\widehat{X}_k;\xi_k\right)-\partial f\left(\widehat{X}_k\right)\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2}_{:=T_2}\\&\quad+2\gamma^2\underbrace{\mathbb{E}\left\|\sum_{k=0}^{K-1}\partial f\left(\widehat{X}_k\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2}_{:=T_3},\end{aligned}$$

where $G_{1/2}(k)=\sum_{k=0}^{K}\sqrt{g(k)}$ as defined before. We then bound T_2 as follows:

$$\begin{aligned}T_2&=\mathbb{E}\left\|\sum_{k=0}^{K-1}\left(\partial F\left(\widehat{X}_k;\xi_k\right)-\partial f\left(\widehat{X}_k\right)\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2\\&=\sum_{k=0}^{K-1}\mathbb{E}\left\|\left(\partial F\left(\widehat{X}_k;\xi_k\right)-\partial f\left(\widehat{X}_k\right)\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2\\&\leqslant\sum_{k=0}^{K-1}\mathbb{E}\left\|\left(\partial F\left(\widehat{X}_k;\xi_k\right)-\partial f\left(\widehat{X}_k\right)\right)\right\|_F^2\left\|\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2\\&\leqslant\sum_{k=0}^{K-1}n\sigma^2\left\|\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2\leqslant n\sigma^2\sum_{k=0}^{K-1}\rho^{K-1+k}\leqslant\frac{n\sigma^2}{1-\rho}\end{aligned} \tag{27}$$

The bound for T_3 is as follows:

$$\begin{aligned} T_3 &= \mathbb{E}\left\|\sum_{k=0}^{K-1}\partial f\left(\widehat{X}_k\right)\left(\tfrac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2 \\ &= \underbrace{\textstyle\sum_{k=0}^{K-1}\mathbb{E}\left\|\partial f\left(\widehat{X}_k\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2}_{:=T_4} \\ &+\underbrace{\textstyle\sum_{k\neq k'}^{K-1}\mathbb{E}\left\langle\partial f\left(\widehat{X}_k\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right),\partial f\left(\widehat{X}_{k'}\right)\left(\frac{\mathbf{1}_n}{n}-W^{K-1+k'}e_i\right)\right\rangle}_{:=T_5} \end{aligned} \tag{28}$$

We will bound T_4 and T_5 separately. T_4 is bound as:

$$\begin{aligned} T_4 &= \sum_{k=0}^{K-1}\mathbb{E}\left\|\partial f\left(\widehat{X}_k\right)\left(\tfrac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2 \\ &\leqslant \sum_{k=0}^{K-1}\mathbb{E}\left\|\partial f\left(\widehat{X}_k\right)\right\|^2\left\|\left(\tfrac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right)\right\|^2 \\ &\overset{\text{Lemma 2}}{\leqslant} 3\sum_{k=0}^{K-1}\sum_{i=1}^{n}\mathbb{E}L^2\widehat{Q}_{k,i}\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right\|^2+\tfrac{3n\varsigma^2}{1-\rho} \\ &+3\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1-k}\right\|^2 \end{aligned} \tag{29}$$

T_5 is bound as:

$$\begin{aligned} T_5 &= \sum_{k\neq k'}^{K-1}\mathbb{E}\left\langle\partial f\left(\widehat{X}_k\right)\left(\tfrac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right),\partial f\left(\widehat{X}_{k'}\right)\left(\tfrac{\mathbf{1}_n}{n}-W^{K-1+k'}e_i\right)\right\rangle \\ &\leqslant \sum_{k\neq k'}^{K-1}\mathbb{E}\left\|\partial f\left(\widehat{X}_k\right)\right\|\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right\|\left\|\partial f\left(\widehat{X}_{k'}\right)\right\|\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1+k'}e_i\right\| \\ &\leqslant \sum_{k\neq k'}^{K-1}\mathbb{E}\left(\frac{\left\|\partial f\left(\widehat{X}_k\right)\right\|^2}{2}+\frac{\left\|\partial f\left(\widehat{X}_{k'}\right)\right\|^2}{2}\right)\rho^{K-1-\frac{k+k'}{2}} \\ &\leqslant \sum_{k\neq k'}^{K-1}\mathbb{E}\left(\left\|\partial f\left(\widehat{X}_k\right)\right\|^2\right)\rho^{K-1-\frac{k+k'}{2}} \\ &\overset{\text{Lemma 2}}{\leqslant} 3\textstyle\sum_{k\neq k'}^{K-1}\left(\sum_{i=1}^{n}\mathbb{E}L^2\widehat{Q}_{k,i}+\underbrace{\left\|\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2}_{:=T_6}\right)\rho^{K-1-\frac{k+k'}{2}}+\underbrace{\textstyle\sum_{k\neq k'}^{K-1}3n\varsigma^2\rho^{K-1-\frac{k+k'}{2}}}_{:=T_7} \end{aligned} \tag{30}$$

Now T_6 is bounded as follows:

$$\begin{aligned} T_6 &= 3\sum_{k\neq k'}^{K-1}\left(\sum_{i=1}^{n}\mathbb{E}L^2\widehat{Q}_{k,i}+\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\right)\rho^{K-1-\frac{k+k'}{2}} \\ &\overset{(26)}{\leqslant} 6\sum_{k=0}^{K-1}\left(\sum_{i=1}^{n}2\mathbb{E}L^2Q_{k,i}+2n^2L^2g(k)+\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\right)\sum_{k'=k+1}^{K-1}\sqrt{\rho}^{2K-2-k-k'} \\ &\leqslant 6\sum_{k=0}^{K-1}\left(\sum_{i=1}^{n}2\mathbb{E}L^2Q_{k,i}+2n^2L^2g(k)+\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\right)\tfrac{\sqrt{\rho}^{K-1-k}}{1-\sqrt{\rho}} \end{aligned} \tag{31}$$

And T_7 is bounded as such:

$$T_7 = 6n\varsigma^2\sum_{k>k'}^{K-1}\rho^{K-1-\frac{k+k'}{2}} = 6n\varsigma^2\frac{\left(\rho^{\frac{k}{2}}-1\right)\left(\rho^{\frac{k}{2}}-\sqrt{\rho}\right)}{\left(\sqrt{\rho}-1\right)^2\left(\sqrt{\rho}+1\right)} \tag{32}$$

$$\leqslant 6n\varsigma^2\frac{1}{\left(1-\sqrt{\rho}\right)^2} \tag{33}$$

Now combining T_6 and T_7 into T_5, and then T_5 and T_4 into T_3, we obtain the following upper bound:

$$\begin{aligned} T_3 &\leqslant 3\sum_{k=0}^{K-1}\sum_{i=1}^{n}\mathbb{E}L^2\widehat{Q}_{k,i}\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right\|^2 \\ &+\tfrac{3n\varsigma^2}{1-\rho}+3\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1-k}\right\|^2 \\ &+6\sum_{k=0}^{K-1}\left(\sum_{i=1}^{n}2\mathbb{E}L^2Q_{k,i}+2n^2L^2g(k)+\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\right) \\ &\tfrac{\sqrt{\rho}^{K-1-k}}{1-\sqrt{\rho}}+\tfrac{6n\varsigma^2}{\left(1-\sqrt{\rho}\right)^2} \\ &\leqslant 3\sum_{k=0}^{K-1}\sum_{i=1}^{n}\mathbb{E}L^2\widehat{Q}_{k,i}\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right\|^2 \\ &+3\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1-k}\right\|^2 \\ &+6\sum_{k=0}^{K-1}\left(\sum_{i=1}^{n}2\mathbb{E}L^2Q_{k,i}+2n^2L^2g(k)+\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\right)\tfrac{\sqrt{\rho}^{K-1-k}}{1-\sqrt{\rho}}+\tfrac{9n\varsigma^2}{\left(1-\sqrt{\rho}\right)^2} \\ &\leqslant 3\sum_{k=0}^{K-1}\sum_{i=1}^{n}\mathbb{E}L^2\widehat{Q}_{k,i}\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1+k}e_i\right\|^2 \\ &+3\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1-k}\right\|^2 \\ &+6\sum_{k=0}^{K-1}\left(\sum_{i=1}^{n}2\mathbb{E}L^2Q_{k,i}+\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\right)\tfrac{\sqrt{\rho}^{K-1-k}}{1-\sqrt{\rho}} \\ &+\tfrac{9n\varsigma^2}{\left(1-\sqrt{\rho}\right)^2}+12n^2L^2G(K-1) \end{aligned} \tag{34}$$

As a reminder, we have defined the following expressions as $G(k)=\sum_{k=0}^{K}g(k)$, $G_{1/2}(k)=\sum_{k=0}^{K}\sqrt{g(k)}$. Now we plug the above bound, along with the bound on T_2 to obtain the following:

$$\begin{aligned} Q_{k,i} &\leqslant n\rho G_{1/2}^2(K-1)+24\gamma^2n^2L^2G(K-1)+\tfrac{2\gamma^2n\sigma^2}{1-\rho} \\ &+6\gamma^2\sum_{k=0}^{K-1}\sum_{i=1}^{n}\mathbb{E}L^2\widehat{Q}_{k,i}\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1-k}e_i\right\|^2 \\ &+6\gamma^2\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\left\|\tfrac{\mathbf{1}_n}{n}-W^{K-1-k}\right\|^2 \\ &+12\gamma^2\sum_{k=0}^{K-1}\left(\sum_{i=1}^{n}2\mathbb{E}L^2Q_{k,i}+\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\right)\tfrac{\sqrt{\rho}^{K-1-k}}{1-\sqrt{\rho}}+\tfrac{18\gamma^2n\varsigma^2}{\left(1-\sqrt{\rho}\right)^2} \\ &\leqslant n\rho G_{1/2}^2(K-1)+24\gamma^2n^2L^2G(K-1)+\tfrac{2\gamma^2n\sigma^2}{1-\rho}+\tfrac{18\gamma^2n\varsigma^2}{\left(1-\sqrt{\rho}\right)^2} \\ &+12\gamma^2\sum_{k=0}^{K-1}\sum_{i=1}^{n}\mathbb{E}L^2\left(Q_{k,i}+ng(k)\right)\rho^{K-1-k}+6\gamma^2\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\rho^{K-1-k} \\ &+12\gamma^2\sum_{k=0}^{K-1}\left(\sum_{i=1}^{n}2\mathbb{E}L^2Q_{k,i}+\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\right)\tfrac{\sqrt{\rho}^{K-1-k}}{1-\sqrt{\rho}} \\ &\leqslant n\rho G_{1/2}^2(K-1)+12\gamma^2n^2\left(2L^2+1\right)G(K-1)+\tfrac{2\gamma^2n\sigma^2}{1-\rho}+\tfrac{18\gamma^2n\varsigma^2}{\left(1-\sqrt{\rho}\right)^2} \\ &+12\gamma^2\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|^2\left(\rho^{K-1-k}+\tfrac{2\sqrt{\rho}^{K-1-k}}{1-\sqrt{\rho}}\right) \\ &+12\gamma^2\sum_{k=0}^{K-1}\sum_{i=1}^{n}\mathbb{E}L^2Q_{k,i}\left(\rho^{K-1-k}+\tfrac{2\sqrt{\rho}^{K-1-k}}{1-\sqrt{\rho}}\right). \end{aligned} \tag{35}$$

Let $M_k=\mathbb{E}\sum_{i=1}^{n}\frac{Q_{k,i}}{n}$,i.e., the expected average of Q_k in all nodes. We then adjust the previous equation to obtain a bound on M_k as follows:

$$M_k \leqslant n\rho G_{1/2}^2(K-1) + 12\gamma^2 n^2\left(2L^2+1\right)G(K-1) + \frac{2n\gamma^2\sigma^2}{1-\rho} + \frac{18n\gamma^2\varsigma^2}{\left(1-\sqrt{\rho}\right)^2} + 12\gamma^2\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2\left(\rho^{K-1-k} + \frac{2\sqrt{\rho}^{K-1-k}}{1-\sqrt{\rho}}\right) + 12\gamma^2L^2n\sum_{k=0}^{K-1}M_k\left(\rho^{K-1-k} + \frac{2\sqrt{\rho}^{K-1-k}}{1-\sqrt{\rho}}\right) \tag{36}$$

Now we sum it from $k=0$ to $K-1$ to obtain the following:

$$\begin{aligned}\sum_{k=0}^{K-1}M_k &\leqslant Kn\rho G_{1/2}^2(K-1) + 12K\gamma^2n^2\left(2L^2+1\right)G(K-1) + \frac{2Kn\gamma^2\sigma^2}{1-\rho} + \frac{18Kn\gamma^2\varsigma^2}{\left(1-\sqrt{\rho}\right)^2} \\ &\quad + 12\gamma^2\sum_{k=0}^{K-1}\sum_{i=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{\widehat{X}_i\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2\left(\rho^{K-1-i} + \frac{2\sqrt{\rho}^{K-1-i}}{1-\sqrt{\rho}}\right) \\ &\quad + 12n\gamma^2L^2\sum_{k=0}^{K-1}\sum_{i=0}^{K-1}M_i\left(\rho^{K-1-i} + \frac{2\sqrt{\rho}^{K-1-i}}{1-\sqrt{\rho}}\right) \\ &\leqslant Kn\rho G_{1/2}^2(K-1) + 12K\gamma^2n^2\left(2L^2+1\right)G(K-1) + \frac{2Kn\gamma^2\sigma^2}{1-\rho} + \frac{18Kn\gamma^2\varsigma^2}{\left(1-\sqrt{\rho}\right)^2} \\ &\quad + 12\gamma^2\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2\left(\sum_{i=0}^{K-1}\rho^{K-1-i} + \frac{2\sum_{i=0}^{K-1}\sqrt{\rho}^{K-1-i}}{1-\sqrt{\rho}}\right) \\ &\quad + 12n\gamma^2L^2\sum_{k=0}^{K-1}M_k\left(\sum_{i=0}^{K-1}\rho^{K-1-i} + \frac{2\sum_{i=0}^{K-1}\sqrt{\rho}^{K-1-i}}{1-\sqrt{\rho}}\right) \\ &\leqslant Kn\rho G_{1/2}^2(K-1) + 12K\gamma^2n^2\left(2L^2+1\right)G(K-1) + \frac{2Kn\gamma^2\sigma^2}{1-\rho} + \frac{18Kn\gamma^2\varsigma^2}{\left(1-\sqrt{\rho}\right)^2} \\ &\quad + \frac{36\gamma^2}{\left(1-\sqrt{\rho}\right)^2}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2 + \frac{36n\gamma^2L^2}{\left(1-\sqrt{\rho}\right)^2}\sum_{k=0}^{K-1}M_k\end{aligned} \tag{37}$$

Rearranging the terms, the expression becomes:

$$\begin{aligned}&\left(1 - \frac{36n\gamma^2L^2}{\left(1-\sqrt{\rho}\right)^2}\right)\sum_{k=0}^{K-1}M_k \leqslant Kn\rho G_{1/2}^2(K-1) \\ &+12K\gamma^2n^2\left(2L^2+1\right)G(K-1) + \frac{2Kn\gamma^2\sigma^2}{1-\rho} \\ &+\frac{18Kn\gamma^2\varsigma^2}{\left(1-\sqrt{\rho}\right)^2} + \frac{36\gamma^2}{\left(1-\sqrt{\rho}\right)^2}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2\end{aligned} \tag{38}$$

Defining $C_2 = \left(1 - \frac{36n\gamma^2L^2}{\left(1-\sqrt{\rho}\right)^2}\right)$, we can rewrite it to:

$$\begin{aligned}\sum_{k=0}^{K-1}M_k &\leqslant C_2^{-1}\left(Kn\rho G_{1/2}^2(K-1) + 12K\gamma^2n^2\left(2L^2+1\right)G(K-1)\right) \\ &\quad + \frac{2Kn\gamma^2\sigma^2}{C_2(1-\rho)} + \frac{18Kn\gamma^2\varsigma^2}{C_2\left(1-\sqrt{\rho}\right)^2} + \frac{36\gamma^2}{C_2\left(1-\sqrt{\rho}\right)^2}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^\top\right\|^2\end{aligned} \tag{39}$$

We also know that T_1 is bounded as such:

$$\begin{aligned}T_1 &\leqslant \frac{2L^2}{n^2}\left(\sum_{i=1}^{n}Q_{k,i} + \sum_{i=1}^{n}ng(k)\right) \\ &\leqslant \frac{2L^2}{n}(M_k + ng(k))\end{aligned} \tag{40}$$

We then input this bound on T_1 in (22) and obtain:

$$\begin{aligned}&\mathbb{E}f\left(\tfrac{X_{k+1}\mathbf{1}_n}{n}\right) \\ &\leqslant \mathbb{E}f\left(\tfrac{X_k\mathbf{1}_n}{n}\right) + \tfrac{1}{2}\mathbb{E}\left\|\nabla f\left(\tfrac{X_k\mathbf{1}_n}{n}\right)\right\|^2 + \tfrac{L+1}{2}g(k) + \tfrac{\gamma^2L\sigma^2}{2n} - \tfrac{\gamma}{2}\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\right\|^2 \\ &\quad + \tfrac{\gamma^2L-\gamma}{2}\mathbb{E}\left\|\tfrac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\|^2 + \tfrac{\gamma L^2}{n}M_k + \gamma L^2g(k) \\ &\leqslant \mathbb{E}f\left(\tfrac{X_k\mathbf{1}_n}{n}\right) + \tfrac{\gamma L^2}{2}\mathbb{E}\left\|\tfrac{\xi_k\mathbf{1}_n}{n}\right\|^2 + \tfrac{L+1}{2}g(k) + \tfrac{\gamma^2L\sigma^2}{2n} - \tfrac{1-\gamma}{2}\mathbb{E}\left\|\nabla f\left(\tfrac{X_k\mathbf{1}_n}{n}\right)\right\|^2 \\ &\quad + \tfrac{\gamma^2L-\gamma}{2}\mathbb{E}\left\|\tfrac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\|^2 + \tfrac{\gamma L^2}{n}M_k + \gamma L^2g(k) \\ &\leqslant \mathbb{E}f\left(\tfrac{X_k\mathbf{1}_n}{n}\right) + \tfrac{3\gamma L^2+L+1}{2}g(k) + \tfrac{\gamma^2L\sigma^2}{2n} - \tfrac{1-\gamma}{2}\mathbb{E}\left\|\nabla f\left(\tfrac{X_k\mathbf{1}_n}{n}\right)\right\|^2 \\ &\quad + \tfrac{\gamma^2L-\gamma}{2}\mathbb{E}\left\|\tfrac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\|^2 + \tfrac{\gamma L^2}{n}M_k\end{aligned} \tag{41}$$

Summing from $k=0$ to $k=K-1$ on both sides yields:

$$\begin{aligned}&\tfrac{1-\gamma}{2}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\tfrac{X_k\mathbf{1}_n}{n}\right)\right\|^2 - \tfrac{\gamma^2L-\gamma}{2}\sum_{k=0}^{K-1}\mathbb{E}\left\|\tfrac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\|^2 \\ &\leqslant f(0) - f^* + \tfrac{K\gamma^2L\sigma^2}{2n} + \tfrac{\gamma L^2}{n}\sum_{k=0}^{K-1}M_k + \tfrac{3\gamma L^2+L+1}{2}G(K-1) \\ &\overset{(39)}{\leqslant} f(0) - f^* + \tfrac{K\gamma^2L\sigma^2}{2n} + C_2^{-1}K\rho\gamma L^2G_{1/2}^2(K-1) \\ &+\left(12C_2^{-1}K\gamma^3nL^2\left(2L^2+1\right) + \tfrac{3\gamma L^2+L+1}{2}\right)G(K-1) + \tfrac{2K\gamma^3n\sigma^2L^2}{C_2(1-\rho)} \\ &+\tfrac{18K\gamma^3\varsigma^2L^2}{C_2\left(1-\sqrt{\rho}\right)^2} + \tfrac{36\gamma^3L^2}{C_2\left(1-\sqrt{\rho}\right)^2}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\tfrac{\widehat{X}_k\mathbf{1}_n}{n}\right)\right\|^2\end{aligned} \tag{42}$$

Rearranging the terms, dividing by K and using the Lipschitz inequality $\left\|\nabla f\left(\frac{\widehat{X}_k\mathbf{1}_n}{n}\right)\right\|^2 \leqslant 2\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2 + 2L^2g(k)$ (similar to what we have done in (22)), we obtain:

$$\begin{aligned}&\tfrac{1}{K}\left(\tfrac{1-\gamma}{2} - \tfrac{72\gamma^3L^2}{C_2\left(1-\sqrt{\rho}\right)^2}\right)\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\tfrac{X_k\mathbf{1}_n}{n}\right)\right\|^2 + \tfrac{\gamma-\gamma^2L}{2K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\tfrac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\|^2 \\ &\leqslant \tfrac{f(0)-f^*}{K} + \tfrac{\gamma^2L\sigma^2}{2n} \\ &+\left(12C_2^{-1}\gamma^3nL^2\left(2L^2+1\right) + \tfrac{3\gamma L^2+L+1}{2K} + \tfrac{72\gamma^3L^4}{KC_2\left(1-\sqrt{\rho}\right)^2}\right)G(K-1) \\ &+C_2^{-1}\gamma\rho L^2G_{1/2}^2(K-1) + \tfrac{2n\gamma^3\sigma^2L^2}{C_2(1-\rho)} + \tfrac{18n\gamma^3\varsigma^2L^2}{C_2\left(1-\sqrt{\rho}\right)^2}\end{aligned} \tag{43}$$

where $C_1 = \left(\frac{1-\gamma}{2} - \frac{72\gamma^3}{C_2\left(1-\sqrt{\rho}\right)^2}L^2\right)$. This completes the proof.

Proof to Corollary 1. First we want to remove the term $\sum_{k=0}^{K-1}\mathbb{E}\left\|\frac{\partial f\left(\widehat{X}_k\right)\mathbf{1}_n}{n}\right\|^2$ in the LHS and maintain the inequality. For that, the coefficient of that term has to satisfy:

$$\begin{aligned}&\tfrac{\gamma-\gamma^2L}{2K} > 0 \\ &\Rightarrow \gamma < \tfrac{1}{L}\end{aligned} \tag{44}$$

Now we have

$$\begin{aligned}&\frac{C_1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2\\&\leqslant\frac{f(0)-f^*}{K}+\frac{\gamma^2L\sigma^2}{2n}\\&+\left(12C_2^{-1}\gamma^3nL^2\left(2L^2+1\right)+\frac{3\gamma L^2+L+1}{2K}+\frac{72\gamma^3L^4}{KC_2\left(1-\sqrt{\rho}\right)^2}\right)G(K-1)\\&+C_2^{-1}\gamma\rho L^2G_{1/2}^2(K-1)+\frac{2n\gamma^3\sigma^2L^2}{C_2(1-\rho)}+\frac{18n\gamma^3\varsigma^2L^2}{C_2\left(1-\sqrt{\rho}\right)^2}\end{aligned}\tag{45}$$

We choose $\gamma=\frac{1}{2\rho L^2\sqrt{K}+\sigma\sqrt{K/n}}$. Also γ should satisfy $\gamma<1$. In order to satisfy that as well as (44), we enforce

$$\begin{aligned}&\frac{1}{2\rho L^2\sqrt{K}+\sigma\sqrt{K/n}}<\frac{1}{L+1}\\&\Rightarrow K>\left(\frac{\sqrt{n}(L+1)}{2\rho L^2\sqrt{n}+\sigma}\right)^2\end{aligned}\tag{46}$$

The γ satisfies the following as well:

$$\begin{aligned}&\gamma<\frac{1}{\sigma\sqrt{\frac{K}{n}}}\\&\Rightarrow\gamma^2<\frac{n}{\sigma^2K}\end{aligned}\tag{47}$$

Since $\gamma<1$, we also have $\gamma^3\leqslant\frac{n}{\sigma^2K}$.

Now if $K\geqslant\frac{72L^2n^2}{\sigma^2\left(1-\sqrt{\rho}\right)^2}$, we can bound C_2 as $C_2\geqslant\frac{1}{2}$. Further we can also bound C_1 as follows:

$$\begin{aligned}C_1&=\left(\frac{1-\gamma}{2}-\frac{72\gamma^3}{C_2\left(1-\sqrt{\rho}\right)^2}L^2\right)\\&\geqslant\left(\frac{1-\gamma}{2}-\frac{144L^2\gamma^3}{\left(1-\sqrt{\rho}\right)^2}\right)\geqslant\frac{1}{2}\end{aligned}\tag{48}$$

Finally we have:

$$\begin{aligned}&\frac{1}{2K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2\leqslant\frac{\left(f(0)-f^*+L/2\right)}{K}\\&+\left(\frac{\left(1-\sqrt{\rho}\right)^2\left(2L^2+1\right)}{3\left(2\rho L^2\sqrt{K}+\sigma\sqrt{K/n}\right)}+\frac{7L^2+L+1}{2K}\right)G(K-1)\\&+\frac{2\rho L^2G_{1/2}^2(K-1)}{2\rho L^2\sqrt{K}+\sigma\sqrt{K/n}}+\frac{4nL^2}{\left(2\rho L^2\sqrt{K}+\sigma\sqrt{K/n}\right)^3}\left(\frac{\sigma^2}{(1-\rho)}+\frac{9\varsigma^2}{\left(1-\sqrt{\rho}\right)^2}\right)\\&\leqslant\frac{\left(f(0)-f^*+L/2\right)}{K}+\frac{C_3}{\sqrt{K}}G(K-1)+\frac{C_4}{K}G(K-1)\\&+\frac{1}{\sqrt{K}}G_{1/2}^2(K-1)+\frac{4n^3L^2}{\sigma^3K\sqrt{Kn}}\left(\frac{\sigma^2}{(1-\rho)}+\frac{9\varsigma^2}{\left(1-\sqrt{\rho}\right)^2}\right)\end{aligned}\tag{49}$$

where $C_3=\frac{\left(1-\sqrt{\rho}\right)^2\left(2L^2+1\right)}{6\rho L^2}$ and $C_4=\frac{7L^2+L+1}{2}$.

If K is large enough, in particular, if $K\geqslant\frac{4n^3L^2}{\sigma^3\left(f(0)-f^*+L/2\right)}\left(\frac{\sigma^2}{(1-\rho)}+\frac{9\varsigma^2}{\left(1-\sqrt{\rho}\right)^2}\right)$, the last term is bounded by $\frac{\left(f(0)-f^*+L/2\right)}{\sqrt{Kn}}$. Thus the final expression is

$$\begin{aligned}\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2&\leqslant(2(f(0)-f^*)+L)\left(\frac{1}{K}+\frac{1}{\sqrt{Kn}}\right)\\&+\left(\frac{2C_3}{\sqrt{K}}+\frac{2C_4}{K}\right)G(K-1)\\&+\frac{2}{\sqrt{K}}G_{1/2}^2(K-1)\end{aligned}\tag{50}$$

which completes the proof.

References

[1] Juan M Górriz, Javier Ramírez, Andrés Ortíz, Francisco J Martínez-Murcia, Fermin Segovia, John Suckling, Matthew Leming, Yu-Dong Zhang, Jose Ramón Álvarez-Sánchez, Guido Bologna, et al. Artificial intelligence within the interplay between natural and artificial computation: Advances in data science, trends and applications. Neurocomputing, 410:237–270, 2020.

[2] Sujatha R Upadhyaya, Parallel approaches to machine learning–a comprehensive survey, Journal of Parallel and Distributed Computing 73 (3) (2013) 284–292.

[3] Tal Ben-Nun, Torsten Hoefler, Demystifying parallel and distributed deep learning: An in-depth concurrency analysis, ACM Computing Surveys (CSUR) 52 (4) (2019) 1–43.

[4] Steven R Young, Derek C Rose, Travis Johnston, William T Heller, Thomas P Karnowski, Thomas E Potok, Robert M Patton, Gabriel Perdue, and Jonathan Miller. Evolving deep networks using hpc. In Proceedings of the Machine Learning on HPC Environments, pages 1–7, 2017.

[5] Junqi Yin, Shubhankar Gahlot, Nouamane Laanait, Ketan Maheshwari, Jack Morrison, Sajal Dash, and Mallikarjun Shankar. Strategies to deploy and scale deep learning on the summit supercomputer. In 2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS), pages 84–94. IEEE, 2019.

[6] Ron Bekkerman, Mikhail Bilenko, John Langford, Scaling up machine learning: Parallel and distributed approaches, Cambridge University Press, 2011.

[7] Amine Boulemtafes, Abdelouahid Derhab, Yacine Challal, A review of privacy-preserving techniques for deep learning, Neurocomputing 384 (2020) 21–45.

[8] Keren Bergman et al. Exascale computing study: Technology challenges in achieving exascale systems. Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep, 15, 2008.

[9] Robert Lucas, James Ang, Keren Bergman, Shekhar Borkar, William Carlson, Laura Carrington, George Chiu, Robert Colwell, William Dally, Jack Dongarra, et al., Doe advanced scientific computing advisory subcommittee (ascac) report: top ten exascale research challenges, Technical report, USDOE Office of Science (SC)(United States), 2014.

[10] Siddhartha Jana, Oscar Hernandez, Stephen Poole, Barbara Chapman, Power consumption due to data movement in distributed programming models, in: European Conference on Parallel Processing, Springer, 2014, pp. 366–378.

[11] Yuchen Zhang, John C Duchi, Martin J Wainwright, Communication-efficient algorithms for statistical optimization, Journal of Machine Learning Research 14 (1) (2013) 3321–3363.

[12] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. arXiv preprint arXiv:1610.02132, 2016.

[13] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In International Conference on Learning Representations, 2018.

[14] Frank Seide, Fu. Hao, Jasha Droppo, Gang Li, Yu. Dong, 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.

[15] Bo Liu, Zhengtao Ding, A consensus-based decentralized training algorithm for deep neural networks with communication compression, Neurocomputing (2021).

[16] A.T. Chronopoulos, Charles William Gear, s-step iterative methods for symmetric linear systems, Journal of Computational and Applied Mathematics 25 (2) (1989) 153–168.

[17] Mark Hoemmen. Communication-avoiding Krylov subspace methods. PhD thesis, UC Berkeley, 2010.

[18] Soumyadip Ghosh, Kamal K Saha, Vijay Gupta, and Gretar Tryggvason. Event-triggered communication in parallel computing. In 2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (scalA), pages 1–8. IEEE, 2018b.

[19] URL:https://github.com/soumyadipghosh/eventgrad/tree/master/dcifar10.

[20] Soumyadip Ghosh, Vijay Gupta, Eventgrad: Event-triggered communication in parallel stochastic gradient descent, in: 2020 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC) and Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S) IEEE, 2020, pp. 1–8.

[21] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In Advances in neural information processing systems, pages 2595–2603, 2010.

[22] Léon Bottou, Frank E. Curtis, Jorge Nocedal, Optimization methods for large-scale machine learning, Siam Review 60 (2) (2018) 223–311.

[23] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In Advances in neural information processing systems, pages 693–701, 2011.

[24] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In Advances in neural information processing systems, pages 685–693, 2015.

[25] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In Advances in Neural Information Processing Systems, pages 2737–2745, 2015.

[26] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, Wojciech Samek, Robust and communication-efficient federated learning from non-iid data, IEEE Transactions on Neural Networks and Learning Systems (2019).

[27] Yang Chen, Xiaoyan Sun, Yaochu Jin, Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation, IEEE Transactions on Neural Networks and Learning Systems (2019).
[28] Jinjin Xu, Wenli Du, Ran Cheng, Wangli He, and Yaochu Jin. Ternary compression for communication-efficient federated learning. arXiv preprint arXiv:2003.03564, 2020.
[29] William D Gropp, William Gropp, Ewing Lusk, Anthony Skjellum, Using MPI: portable parallel programming with the message-passing interface, volume 1, MIT press, 1999.
[30] Nikko Strom, Scalable distributed dnn training using commodity gpu cloud computing, in: Sixteenth Annual Conference of the International Speech Communication Association, 2015.
[31] Nikoli Dryden, Tim Moon, Sam Ade Jacobs, and Brian Van Essen. Communication quantization for data-parallel training of deep neural networks. In 2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC), pages 1–8. IEEE, 2016.
[32] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, Pritish Narayanan, Deep learning with limited numerical precision, in: International Conference on Machine Learning, 2015, pp. 1737–1746.
[33] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In International Conference on Learning Representations, 2018.
[34] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In Advances in Neural Information Processing Systems, pages 5973–5983, 2018.
[35] Cèdric Renggli, Saleh Ashkboos, Mehdi Aghagolzadeh, Dan Alistarh, Torsten Hoefler, Sparcml: High-performance sparse communication for machine learning, in: In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019, pp. 1–15.
[36] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations. In Advances in Neural Information Processing Systems, pages 14695–14706, 2019.
[37] Kun Yuan, Qing Ling, Wotao Yin, On the convergence of decentralized gradient descent, SIAM Journal on Optimization 26 (3) (2016) 1835–1854.
[38] J. Reza Olfati-Saber, Alex Fax, Richard M Murray, Consensus and cooperation in networked multi-agent systems, Proceedings of the IEEE 95 (1) (2007) 215–233.
[39] Youjie Li, Mingchao Yu, Songze Li, Salman Avestimehr, Nam Sung Kim, and Alexander Schwing. Pipe-sgd: A decentralized pipelined sgd framework for distributed deep net training. arXiv preprint arXiv:1811.03619, 2018.
[40] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In Advances in Neural Information Processing Systems, pages 5330–5340, 2017.
[41] Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome. Gossip training for deep learning. arXiv preprint arXiv:1611.09726, 2016..
[42] Peter H Jin, Qiaochu Yuan, Forrest Iandola, and Kurt Keutzer. How to scale distributed deep learning? arXiv preprint arXiv:1611.04581, 2016.
[43] Jeff Daily, Abhinav Vishnu, Charles Siegel, Thomas Warfel, and Vinay Amatya. Gossipgrad: Scalable deep learning using gossip communication based asynchronous gradient descent. arXiv preprint arXiv:1803.05880, 2018.
[44] Michael Lemmon, Event-triggered feedback in control, estimation, and optimization, in: Networked control systems, Springer, 2010, pp. 293–358.
[45] Dimos V. Dimarogonas, Emilio Frazzoli, Karl H. Johansson, Distributed event-triggered control for multi-agent systems, IEEE Transactions on Automatic Control 57 (5) (2012) 1291–1297.
[46] Aijuan Wang, Tao Dong, Xiaofeng Liao, Distributed optimal consensus algorithms in multi-agent systems, Neurocomputing 339 (2019) 26–35.
[47] Xin Chen, Xiaofeng Liao, Lan Gao, Shasha Yang, Huiwei Wang, Huaqing Li, Event-triggered consensus for multi-agent networks with switching topology under quantized communication, Neurocomputing 230 (2017) 294–301.
[48] Cameron Nowzari, Eloy Garcia, Jorge Cortés, Event-triggered communication and control of networked systems for multi-agent consensus, Automatica 105 (2019) 1–27.
[49] Zhongyuan Zhao, Gang Chen, Mingxiang Dai, Distributed event-triggered scheme for a convex optimization problem in multi-agent systems, Neurocomputing 284 (2018) 90–98.
[50] Qingguo Lü, Huaqing Li, Dawen Xia, Distributed optimization of first-order discrete-time multi-agent systems with event-triggered communication, Neurocomputing 235 (2017) 255–263.
[51] Dean Richert, Jorge Cortes, Distributed linear programming with event-triggered communication, SIAM Journal on Control and Optimization 54 (3) (2016) 1769–1797.
[52] Jin Zhang, Chen Peng, Du. Dajun, Min Zheng, Adaptive event-triggered communication scheme for networked control systems with randomly occurring nonlinearities and uncertainties, Neurocomputing 174 (2016) 475–482.
[53] Jemin George and Prudhvi Gurram. Distributed deep learning with event-triggered communication. arXiv preprint arXiv:1909.05020, 2019.
[54] Stephen Boyd, Persi Diaconis, Lin Xiao, Fastest mixing markov chain on a graph, SIAM Review 46 (4) (2004) 667–689.
[55] Georg S. Seyboth, Dimos V. Dimarogonas, Karl H. Johansson, Event-based broadcasting for multi-agent average consensus, Automatica 49 (1) (2013) 245–252.
[56] William Gropp, Torsten Hoefler, Rajeev Thakur, and Ewing Lusk. Using advanced MPI: Modern features of the message-passing interface. MIT Press, 2014.
[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

Soumyadip Ghosh is a PhD candidate in Electrical Engineering at the University of Notre Dame. He obtained his Bachelors in Electrical Engineering from Jadavpur University, India. His research interests are in areas at the intersection of systems theory and high performance computing with applications in parallel machine learning, optimization and numerical solution of partial differential equations.

Bernardo Aquino received the B.S. (Hons.) degree in electronics engineering and the M.S. degree in electrical engineering from the Federal University of Rio de Janeiro, Brazil in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, University of Notre Dame. His research interests include controls, optimization, and machine learning.

Vijay Gupta is in the Department of Electrical Engineering at the University of Notre Dame, having joined the faculty in January 2008. He received his B. Tech degree at Indian Institute of Technology, Delhi, and his M.S. and Ph.D. at California Institute of Technology, all in Electrical Engineering. He received the 2018 Antonio J Rubert Award from the IEEE Control Systems Society, the 2013 Donald P. Eckman Award from the American Automatic Control Council and a 2009 National Science Foundation (NSF) CAREER Award. His research interests are broadly at the interface of communication, control, distributed computation, and human decision making.