LEARN LOCALLY, CORRECT GLOBALLY: A DISTRIBUTED ALGORITHM FOR TRAINING GRAPH NEURAL NETWORKS

Morteza Ramezani*, **Weilin Cong***, **Mehrdad Mahdavi**
Mahmut T. Kandemir, **Anand Sivasubramaniam**
Pennsylvania State University, University Park, PA 16802, USA
morteza@cse.psu.edu, wxc272@psu.edu, mzm616@psu.edu
anand@cse.psu.edu, kandemir@cse.psu.edu

ABSTRACT

Despite the recent success of Graph Neural Networks (GNNs), training GNNs on large graphs remains challenging. The limited resource capacities of the existing servers, the dependency between nodes in a graph, and the privacy concern due to the centralized storage and model learning have spurred the need to design an effective distributed algorithm for GNN training. However, existing distributed GNN training methods impose either excessive communication costs or large memory overheads that hinders their scalability. To overcome these issues, we propose a communication-efficient distributed GNN training technique named *Learn Locally, Correct Globally* (LLCG). To reduce the communication and memory overhead, each local machine in LLCG first trains a GNN on its local data by ignoring the dependency between nodes among different machines, then sends the locally trained model to the server for periodic model averaging. However, ignoring node dependency could result in significant performance degradation. To solve the performance degradation, we propose to apply *Global Server Corrections* on the server to refine the locally learned models. We rigorously analyze the convergence of distributed methods with periodic model averaging for training GNNs and show that naively applying periodic model averaging but ignoring the dependency between nodes will suffer from an irreducible residual error. However, this residual error can be eliminated by utilizing the proposed global corrections to entail fast convergence rate. Extensive experiments on real-world datasets show that LLCG can significantly improve the efficiency without hurting the performance.

1 INTRODUCTION

In recent years, Graph Neural Networks (GNNs) have achieved impressive results across numerous graph-based applications, including social networks (Hamilton et al., 2017; Deng et al., 2019), recommendation systems (Ying et al., 2018; Wang et al., 2018), and drug discovery (Fout et al., 2017; Do et al., 2019). Despite their recent success, effective training of GNNs on large-scale real-world graphs, such as Facebook social network (Boldi & Vigna, 2004), remains challenging. Although several attempts have been made to scale GNN training by sampling techniques (Hamilton et al., 2017; Zou et al., 2019; Zeng et al., 2020; Chiang et al., 2019; Chen et al., 2018; Zhang et al., 2021), they are still inefficient for training on extremely large graphs, due to the unique structure of GNNs and the limited memory capacity/bandwidth of current servers. One potential solution to tackle these limitations is employing distributed training with data parallelism, which have become almost a *de facto* standard for fast and accurate training for natural language processing (Lin et al., 2021; Hard et al., 2018) and computer vision (Bonawitz et al., 2019; Konečný et al., 2018). For

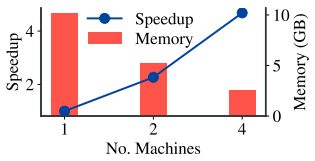


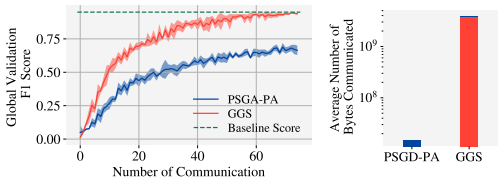
Figure 1: Comparison of the speedup and the memory consumption of distributed multi-machine training and centralized single machine training on the `Reddit` dataset.

*Equal Contribution.

example, as shown in Figure 1, moving from single machine to multiple machines reduces the training time and alleviates the memory burden on each machine. Besides, scaling the training of GNNs with sampling techniques can result in privacy concerns: existing sampling-based methods require centralized data storage and model learning, which could result in privacy concerns in real-world scenarios (Shin et al., 2018; Wu et al., 2021). Fortunately, the privacy in distributed learning can be preserved by avoiding mutual access to data between different local machines, and using only a trusted third party server to access the entire data.

Nonetheless, generalizing the existing data parallelism techniques of classical distributed training to the graph domain is non-trivial, which is mainly due to the dependency between nodes in a graph. For example, unlike solving image classification problems where images are mutually independent, such that we can divide the image dataset into several partitions without worrying about the dependency between images; GNNs are heavily relying on the information inherent to a node and its neighboring nodes. As a result, partitioning graph leads to subgraphs with edges spanning subgraphs (*cut-edges*), which will cause information loss and hinder the performance of the model (Angerd et al., 2020). To cope with this problem, (Md et al., 2021; Jiang & Rumi, 2021; Angerd et al., 2020) propose to transfer node features and (Zheng et al., 2020; Tripathy et al., 2020; Scardapane et al., 2020) propose to transfer both the node feature and its hidden embeddings between local machines, both of which can cause significant storage/communication overhead and privacy concerns (Shin et al., 2018; Wu et al., 2021).

To better understand the challenge of distributed GNN training, we compare the validation F1-score in Figure 2 (a) and the average data communicated per round in Figure 2 (b) for two different distributed GNN training methods on the `Reddit` dataset. On the one hand, we can observe that when ignoring the cut-edges, *Parallel SGD with Periodic Averaging* (PSGD-PA (Dean et al., 2012; Li et al., 2020b)) suffers from significant accuracy drop and cannot achieve the same accuracy as the single machine training, even by increasing the number of communication. However, *Global Graph Sampling* (GGS) can successfully reach the baseline by considering the cut-edges and allowing feature transfer, at the cost of significant communication overhead, and potential violation of privacy.



(a) (b)

Figure 2: Comparison of (a) the validation F1-score and (b) the average data communicated per round (in bytes and log-scale) for two different distributed GNN training settings, including *Parallel SGD with Periodic Averaging* (PSGD-PA) where the cut-edges are ignored and only the model parameters are transferred and *Global Graph Sampling* (GGS), where the cut-edges are considered and the node features of the cut-edges are transferred to the corresponding local machine, on the `Reddit` dataset using 8 machines.

In this paper, we propose a communication-efficient distributed GNN training method, called *Learn Locally, Correct Globally* (LLCG). To reduce the communication overhead, inspired by the recent success of the distributed optimization with periodic averaging (Stich, 2019; Yu et al., 2019), we propose *Local Training with Periodic Averaging*: where each local machine first locally trains a GNN model by ignoring the cut-edges, then sends the trained model to the server for periodic model averaging, and receive the averaged model from server to continue the training. By doing so we eliminate the features exchange phase between server and local machines, but it can result in a significant performance degradation due to the lack of the global graph structure and the dependency between nodes among different machines. To compensate for this error, we propose a *Global Server Correction* scheme to take advantage of the available global graph structure on the server and refine the averaged locally learned models before sending it back to each local machine. Notice that without *Global Server Correction*, LLCG is similar to PSGD-PA as introduced in Figure 2.

To get a deeper understanding on the necessity of *Global Server Correction*, we provide the first theoretical analysis on the convergence of distributed training for GNNs with periodic averaging. In particular, we show that solely averaging the local machine models and ignoring the global graph structure will suffer from an irreducible residual error, which provides sufficient explanation on why *Parallel SGD with Periodic Averaging* can never achieve the same performance as the model trained on a single machine in Figure 2 (a). Then, we theoretically analyze the convergence of our proposal LLCG . We show that by carefully choosing the number of global correction steps, LLCG can overcome the aforementioned residual error and enjoys $\mathcal{O}(1/\sqrt{PT})$ convergence rate with P local machines and T iterations of gradient updates, which matches the rate of (Yu et al.,

2019) on a general (not specific for GNN training) non-convex optimization setting. Finally, we conduct comprehensive evaluations on real-world graph datasets with ablation study to validate the effectiveness of LLCG and its improvements over the existing distributed methods.

Related works. Recently, several attempts have been made on distributed GNN training. According to how they deal with the input/hidden feature of nodes that are associated with the cut-edges (i.e., the edges spanning subgraphs of each local machine), existing methods can be classified into two main categories : (1) *Input feature only communication-based methods*: In these methods, each local machine receives the input features of all nodes required for the gradient computation from other machines, and trains individually. However, since the number of required nodes grows exponentially with the number of layers, these methods suffer from a significant communication and storage overhead. To alleviate these issues, (Md et al., 2021) proposes to split the original graph using a min-cut graph partition algorithm that can minimize the number of cut-edges. (Jiang & Rumi, 2021) proposes to use importance sampling to assign nodes on the local machine with a higher probability. (Angerd et al., 2020) proposes to sample and save a small subgraph from other local machines as an approximation of the original graph structure. Nonetheless, these methods are limited to a very shallow GNN structure and suffer from significant performance degradation when the original graph is dense. (2) *Input and hidden feature communication-based methods*: These methods propose to communicate hidden features in addition to the input node features. Although these methods reduce the number of transferred bytes during each communication round (due to the smaller size of hidden embedding and less required nodes features), the number of communication rounds grows linearly as the number of layers, and are prone to more communication delay. To address these issues, in addition to optimal partitioning of the graph, (Zheng et al., 2020) proposes to use sparse embedding to reduce the number of bytes to communicate and (Tripathy et al., 2020) proposes several graph partitioning techniques to diminish the communication overhead.

2 Background and Problem Formulation

In this section, we start by describing Graph Convolutional Network (GCN) and its training algorithm on a single machine, then formulate the problem of distributed GCN training. Note that we use GCN with mean aggregation for simplicity, however, our discussion is also applicable to other GNN architectures, such as SAGE (Hamilton et al., 2017), GAT (Velickovic et al., 2018), ResGCN (Li et al., 2019) and APPNP (Klicpera et al., 2019).

Training GCN on a single machine. Here, we consider the semi-supervised node classification in an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $N = |\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges. Each node $v_i \in \mathcal{V}$ is associated with a pair $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the input feature vector, $\mathbf{y}_i \in \mathbb{R}^{|\mathcal{C}|}$ is the ground truth label, and $\mathcal{C}$ is the candidate labels in the multi-class classifications. Besides, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{N \times d}$ denote the input node feature matrix. Our goal is to find a set of parameters $\boldsymbol{\theta} = \{\mathbf{W}^{(\ell)}\}_{\ell=1}^{L}$ by minimizing the empirical loss $\mathcal{L}(\boldsymbol{\theta})$ over all nodes in the training set, i.e.,

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum\nolimits_{i \in \mathcal{V}} \phi(\mathbf{h}_i^{(L)}, \mathbf{y}_i), \qquad \mathbf{h}_i^{(\ell)} = \sigma\Big(\frac{1}{|\mathcal{N}(v_i)|} \sum\nolimits_{j \in \mathcal{N}(v_i)} \mathbf{h}_j^{(\ell-1)} \mathbf{W}^{(\ell)}\Big), \tag{1}$$

where $\phi(\cdot, \cdot)$ is the loss function (e.g., cross entropy loss), $\sigma(\cdot)$ is the activation function (e.g., ReLU), and $\mathcal{N}(v_i)$ is the neighborhood of node v_i. In practice, we can update the model parameters by the stochastic gradient computed on a sampled mini-batch (using full-neighbors) by

$$\tilde{\nabla}\mathcal{L}(\boldsymbol{\theta}, \xi) = \frac{1}{B} \sum\nolimits_{i \in \xi} \nabla \phi(\mathbf{h}_i^{(L)}, \mathbf{y}_i), \tag{2}$$

where ξ denotes an i.i.d. sampled mini-batch of size B and we have $\mathbb{E}[\tilde{\nabla}\mathcal{L}(\boldsymbol{\theta}, \xi)] = \nabla \mathcal{L}(\boldsymbol{\theta})$.

Distributed GCN training with periodic averaging. In this paper, we consider the distributed learning setting with P local machines and a single parameter server. The original input graph $\mathcal{G}$ is partitioned into P subgraphs, where $\mathcal{G}_p(\mathcal{V}_p, \mathcal{E}_p)$ denotes the subgraph on the p-th local machine with $N_p = |\mathcal{V}_p|$ nodes, and $\mathbf{X}_p \in \mathbb{R}^{N_p \times d}$ as the input feature of all nodes in $\mathcal{V}_p$ located on the p-th machine. Then, the full-batch local gradient $\nabla \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p)$ is computed as

$$\nabla \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p) = \frac{1}{N_p} \sum\nolimits_{i \in \mathcal{V}_p} \nabla \phi(\mathbf{h}_i^{(L)}, \mathbf{y}_i), \;\; \mathbf{h}_i^{(\ell)} = \sigma\Big(\frac{1}{|\mathcal{N}_p(v_i)|} \sum\nolimits_{j \in \mathcal{N}_p(v_i)} \mathbf{h}_j^{(\ell-1)} \mathbf{W}_p^{(\ell)}\Big), \tag{3}$$

where $\boldsymbol{\theta}_p = \{\mathbf{W}_p^{(\ell)}\}_{\ell=1}^{L}$ is the model parameters on the p-th local machine, $\mathcal{N}_p(v_i) = \{v_j | (v_i, v_j) \in \mathcal{E}_p\}$ is the local neighbors of node v_i on the p-th local machine. When the graph is large, the

Algorithm 1 Distributed GCN training with "*Parallel SGD with Periodic Averaging*"

Input: Global parameters $\bar{\boldsymbol{\theta}}^0$, local parameters $\boldsymbol{\theta}_p^0 = \bar{\boldsymbol{\theta}}^0$, time-step $t = 0$, learning rate η.
1: **for** $r \leftarrow 1$ **to** R **do**
2: **for** $p \leftarrow 1$ **to** P **do in parallel** ▷ Parallel training on local machines
3: Local machine p receives the global parameters $\boldsymbol{\theta}_p^t \leftarrow \bar{\boldsymbol{\theta}}^t$. ▷ Communication
4: **for** $k \leftarrow 1$ **to** K **do**
5: $t \leftarrow t + 1$.
6: Local machine p constructs the mini-batch ξ_p^t with neighbor sampling.
7: Local machine p computes the stochastic gradients $\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t, \xi_p^t)$.
8: Local machine p updates the local parameter by $\boldsymbol{\theta}_p^{t+1} = \boldsymbol{\theta}_p^t - \eta\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t, \xi_p^t)$.
9: **end for**
10: Local machine p sends the local parameters $\boldsymbol{\theta}_p^{t+1}$ to the server. ▷ Communication
11: **end for**
12: Server updates the global parameters by parameter averaging $\bar{\boldsymbol{\theta}}^{t+1} = \frac{1}{P}\sum_{p=1}^{P}\boldsymbol{\theta}_p^{t+1}$.
13: **end for**
Output: Server returns trained GCN model with $\min_t \mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2]$.

computational complexity of forward and backward propagation could be very high. One practical solution is to compute the stochastic gradient on a sampled mini-batch with neighbor sampling, i.e.,

$$\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p, \xi_p) = \frac{1}{B_p}\sum\nolimits_{i\in\xi_p}\nabla\phi(\tilde{\mathbf{h}}_i^{(L)}, \mathbf{y}_i), \quad \tilde{\mathbf{h}}_i^{(\ell)} = \sigma\Big(\frac{1}{|\tilde{\mathcal{N}}_p(v_i)|}\sum\nolimits_{j\in\tilde{\mathcal{N}}_p(v_i)}\tilde{\mathbf{h}}_j^{(\ell-1)}\mathbf{W}_p^{(\ell)}\Big), \quad (4)$$

where ξ_p is an i.i.d. sampled mini-batch of B_p nodes, $\tilde{\mathcal{N}}_p(v_i) \subset \mathcal{N}(v_i)$ is the sampled neighbors.

An illustration of distributed GCN training with *Parallel SGD with Periodic Averaging* (PSGD-PA) is summarized in Algorithm 1. Before training, the server maintains a global model $\bar{\boldsymbol{\theta}}^0$ and each local machine keeps a local copy of the same model $\boldsymbol{\theta}_p^0$. During training, the local machine first updates the local model $\boldsymbol{\theta}_p^t$ using the stochastic gradient $\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t, \xi_p^t)$ computed by Eq. 4 for K iterations (line 8), then sends the local model $\boldsymbol{\theta}_p^t$ to the server (line 10). At each communication step, the server collects and averages the model parameters from the local machines (line 12) and send the averaged model $\boldsymbol{\theta}_p^{t+1}$ back to each local machine.

Limitations. Although PSGD-PA can significantly reduce the communication overhead by transferring the locally trained models instead of node feature/embeddings (refer to Figure 2 (b)), it suffers from performance degeneration due to ignorance of the cut-edges (refer to Figure 2 (a)). In the next section, we introduce a communication-efficient algorithm LLCG that does not suffer from this issue, and can achieve almost the same performance as training the model on a single machine.

3 Proposed Algorithm: Learn Locally Correct Globally

In this section, we describe *Learn Locally, Correct Globally* (LLCG) for distributed GNN training. LLCG includes two main phases, *local training with periodic model averaging* and *global server correction*, to help reduce both the number of required communications and size of transferred data, without compromising the predictive accuracy. We summarize the details of LLCG in Algorithm 2.

3.1 Local Training with Periodic Model Averaging

At the beginning of a local epoch, each local machine receives the latest global model parameters from the server (line 3). Next, each local machine runs $K\rho^r$ iterations to update the local model (line 4 to 9), where K and ρ are the hyper-parameters that control the local epoch size. Note that instead of using a fixed local epoch size as Algorithm 1, we choose to use exponentially increasing local epoch size in LLCG with $\rho > 1$. The reasons are as follows.

At the beginning of the training phase, all local models $\boldsymbol{\theta}_p^t$ are far from the optimal solution and will receive a gradient $\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t, \xi_p^t)$ computed by Eq. 4. Using a smaller local update step at the early stage guarantees each local model does not diverge too much from each other before the model averaging step at the server side (line 12). However, towards the end of the training, all local models $\boldsymbol{\theta}_p^t$ will receive relatively smaller gradient $\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t, \xi_p^t)$, such that we can chose a larger local epoch size to reduce the number of communications, without worrying about the divergence of

Algorithm 2 Distributed GCN training by "*Learn Locally, Correct Globally*"

Input: Global parameters $\bar{\boldsymbol{\theta}}^0$, local parameters $\boldsymbol{\theta}_p^0$, time-step $t = 0$, local step size hyper-parameters K, ρ, and learning rate γ, η

1: **for** $r \leftarrow 1$ **to** R **do**
2: **for** $p \leftarrow 1$ **to** P **do in parallel** ▷ Parallel training on local machine
3: Local machine p receives the global parameters $\boldsymbol{\theta}_p^t \leftarrow \bar{\boldsymbol{\theta}}^t$ ▷ Communication
4: **for** $k \leftarrow 1$ **to** $K\rho^r$ **do**
5: $t \leftarrow t + 1$
6: Local machine p constructs the mini-batch ξ_p^t with neighbor sampling
7: Local machine p computes stochastic gradients $\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t, \xi_p^t)$
8: Local machine p updates model parameter by $\boldsymbol{\theta}_p^{t+1} = \boldsymbol{\theta}_p^t - \eta\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t, \xi_p^t)$
9: **end for**
10: Local machine p sends the local parameters $\boldsymbol{\theta}_p^{t+1}$ to the server ▷ Communication
11: **end for**
12: Server updates the global parameters using parameter averaging $\bar{\boldsymbol{\theta}}^{t+1} = \frac{1}{P}\sum_{p=1}^{P}\boldsymbol{\theta}_p^{t+1}$
13: **for** $s \leftarrow 1$ **to** S **do** ▷ Server Correction
14: $t \leftarrow t + 1$
15: Server constructs a mini-batch ξ^t with full-neighbors
16: Server computes the stochastic gradient $\tilde{\nabla}\mathcal{L}(\bar{\boldsymbol{\theta}}^t, \xi^t)$
17: Server updates the global parameters by $\bar{\boldsymbol{\theta}}^{t+1} = \bar{\boldsymbol{\theta}}^t - \gamma\tilde{\nabla}\mathcal{L}(\bar{\boldsymbol{\theta}}^t, \xi^t)$
18: **end for**
19: **end for**

Output: Server return GCN model with trained $\min_t \mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2]$

local models. By doing so, after total number of $T = \sum_{r=1}^{R} K\rho^r$ iterations, LLCG only requires $R = \log_\rho \frac{T}{K}$ rounds of communications. Therefore, compared to the fully-synchronous method, we can significantly reduce the total number of communications from $\mathcal{O}(T)$ to $\mathcal{O}(\log_\rho \frac{T}{K})$.

3.2 Global Server Correction

The design of the global server correction is to ensure that the trained model not only learns from the data on each local machine, but also learns the global structure of the graph, thus reducing the information loss caused by graph partitioning and avoiding cut-edges. Before the correction, the server receives the locally trained models from all local machines (line 10) and applies model parameter averaging (line 12). Next, S server correction steps are applied on top of the averaged model (line 13 to 18). During the correction, the server first constructs a mini-batch ξ^t using full-neighbors[1](line 15), compute the stochastic gradient $\tilde{\nabla}\mathcal{L}(\bar{\boldsymbol{\theta}}^t, \xi^t)$ on the constructed mini-batch by Eq. 2 (line 16) and update the averaged model $\bar{\boldsymbol{\theta}}^t$ for S iterations (line 17). The number of correction steps S [2] depends on the heterogeneity among the subgraphs on each local machine: the more heterogeneous the subgraphs are, the more correction steps are required to better refine the averaged model and reduce the divergence across the local models. Note that, the heterogeneity is minimized when employing GGS (Figure 2) with the local machines having access to the full graph, as a result. However, GGS requires sampling from the global graph and communication at every iteration, which results in additional overhead and lower efficiency. Instead, in LLCG we are trading computation on the server for the costly feature communication, and only requires periodic communication.

4 Theoretical Analysis

In this section, we provide the convergence analysis on the distributed training of GCN under two different settings, i.e., with and without server correction. In the following, we first introduce the notations and assumptions for the analysis (Section 4.1). Then, we show that periodic averaging of local machine models alone and ignoring the global graph structure will suffer from an irreducible residual error (Section 4.2). Finally, we show that this residual error can be eliminated by running server correction steps after each periodic averaging step on the server (Section 4.3).

[1]Note that using full neighbors is required for the server correction but not the local machines

[2]In practice, we found $S = 1$ or $S = 2$ works well on most datasets.

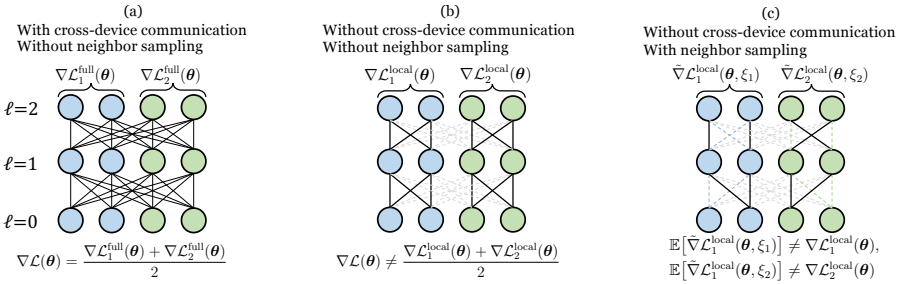


Figure 3: Comparison of notations $\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta})$, $\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}, \xi_p)$, and $\nabla\mathcal{L}_p^{\text{full}}(\boldsymbol{\theta})$ on two local machines, where the blue node and green circles represent nodes on different local machines.

4.1 Notations and Assumptions

Let us first recall the notations defined in Section 2, where $\mathcal{L}(\boldsymbol{\theta})$ denotes the global objective function computed using the all node features $\mathbf{X}$ and the original graph $\mathcal{G}$, $\mathcal{L}_p(\boldsymbol{\theta})$ denotes the local objective function computed using the local node features $\mathbf{X}_p$ and local graph $\mathcal{G}_p$, $\boldsymbol{\theta}_p^t$ denotes the model parameters on the p-th local machine at the t-th step, and $\bar{\boldsymbol{\theta}}^t = \frac{1}{P}\sum_{p=1}^{P}\boldsymbol{\theta}_p^t$ denotes the virtual averaged model at the t-th step. In the non-convex optimization, our goal is to show the expected gradient of the global objective on the virtual averaged model parameters $\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2]$ decreases as the number of local machines P and the number of training steps T increase. Besides, we introduce $\nabla\mathcal{L}_p^{\text{full}}(\boldsymbol{\theta})$ as the gradient computed on the p-th local machine but have access the full node features $\mathbf{X}$ and the original graph structure $\mathcal{G}$ as

$$\nabla\mathcal{L}_p^{\text{full}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{V}_p|}\sum\nolimits_{i\in\mathcal{V}_p}\nabla\phi(\mathbf{h}_i^{(L)}, y_i), \qquad \mathbf{h}_i^{(\ell)} = \sigma\Big(\frac{1}{|\mathcal{N}(v_i)|}\sum\nolimits_{j\in\mathcal{N}(v_i)}\mathbf{h}_j^{(\ell-1)}\mathbf{W}_p^{(\ell)}\Big). \quad (5)$$

Please refer to Figure 3 for an illustration of different gradient computations. Besides, we introduce *local-global gradient discrepancy* as $\kappa^2 = \kappa_{\mathbf{A}}^2 + \kappa_{\mathbf{X}}^2$, where $\kappa_{\mathbf{A}}^2 = \max_{p\in[P]}\{\|\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}) - \nabla\mathcal{L}_p^{\text{full}}(\boldsymbol{\theta})\|^2\}$ is the maximum difference between the gradient computed on the local machine with and without having access to the global graph structure, which is mainly due to fact that the local machines are oblivious to the full graph information; and $\kappa_{\mathbf{X}}^2 = \max_{p\in[P]}\{\|\nabla\mathcal{L}_p^{\text{full}}(\boldsymbol{\theta}) - \nabla\mathcal{L}(\boldsymbol{\theta})\|^2\}$ is the maximum difference between the gradient computed using the local node and all nodes, which is mainly due to the heterogeneity of the node features on each local machine, and we have $\kappa_{\mathbf{X}}^2 = 0$ if the nodes are i.i.d. sampled to each local machine. Notice that *local-global gradient discrepancy* κ^2 plays an important role in our theoretical results.

For the convergence analysis, we make the following standard assumptions.

Assumption 1 *The stochastic gradient on the p-th local machine (with neighbor sampling) has stochastic gradient variance bounded by σ_{var}^2 and stochastic gradient bias bounded by σ_{bias}^2, i.e.,* $\mathbb{E}[\|\tilde{\nabla}\mathcal{L}_p^{local}(\boldsymbol{\theta};\xi) - \mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{local}(\boldsymbol{\theta};\xi)]\|^2] \leq \sigma_{var}^2$, $\mathbb{E}[\|\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{local}(\boldsymbol{\theta};\xi)] - \nabla\mathcal{L}_p^{local}(\boldsymbol{\theta})\|^2] \leq \sigma_{bias}^2$.

Assumption 2 *The stochastic gradient for global server correction (with full neighbors) has stochastic gradient variance bounded by σ_{global}^2, i.e.,* $\mathbb{E}[\|\tilde{\nabla}\mathcal{L}_p^{full}(\boldsymbol{\theta};\xi) - \nabla\mathcal{L}_p^{full}(\boldsymbol{\theta})]\|^2] \leq \sigma_{global}^2$.

The existence of stochastic gradient bias and variance in sampling-based GNN training have been studied in (Cong et al., 2020; 2021), where (Cong et al., 2021) further quantify the stochastic gradient bias and variance as a function of the number of GCN layers. In particular, they show that the existence of σ_{bias}^2 is due to neighbor sampling and non-linear activation, and we have $\sigma_{\text{bias}}^2 = 0$ if all neighbors are used or the non-linear activation is removed. The existence of σ_{var}^2 is because we are sampling mini-batches to compute the stochastic gradient on each local machine during training. As the mini-batch size increases, σ_{var}^2 will be decreasing, and we have $\sigma_{\text{var}}^2 = 0$ when using full-batch.

4.2 Distributed GNN via Parameter Averaging

In the following, we provide the first convergence analysis on distributed training of GCN. We show that solely periodic averaging of the local machine models and ignoring the global graph structure suffers from an upper bound that is irreducible with the number of training steps. Comparing to the traditional distributed training (e.g., distributed training Convolutional Neural Network for image classification (Dean et al., 2012; Li et al., 2020b)), the key challenges in the distributed GCN training is the two different types of gradient bias: (1) The expectation of the local full-batch gradient is

a biased estimation of the global full-batch gradient, i.e., $\frac{1}{P}\sum_{p=1}^{P}\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}) \neq \nabla\mathcal{L}(\boldsymbol{\theta})$. This is because each local machine does not have access to the original input graph and full node feature matrix. Note that the aforementioned equivalence is important for the classifcal distributed training analysis Dean et al. (2012); Yu et al. (2019). (2) The expectation of the local stochastic gradient is a biased estimation of the local full-batch gradient i.e., $\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta},\xi_p)] \neq \nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta})$. This is because the stochastic gradient on each local machine is computed by using neighbor sampling, which has been studied in (Cong et al., 2021).

Theorem 1 (Distributed GCN via Parameter Averaging) *Consider applying model averaging for GNN training under Assumption 1 and 2. If we choose learning rate $\eta = \frac{\sqrt{P}}{\sqrt{T}}$ and the local step size $K \leq \frac{\sqrt{2}T^{1/4}}{8LP^{3/4}}$, then for any $T \geq L^2P$ steps of gradient updates we have*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] = \mathcal{O}\left(\frac{1}{\sqrt{PT}}\right) + \mathcal{O}(\kappa^2 + \sigma_{bias}^2).$$

Theorem 1 implies that, by carefully choosing the learning rate η and the local step size K, the gradient norm computed on the virtual averaged model is bounded by $\mathcal{O}(1/\sqrt{PT})$ after $R = T/K = \mathcal{O}(\frac{P^{3/4}}{T^{3/4}})$ communication rounds, but suffers from an irreducible residual error upper bound $\mathcal{O}(\kappa^2 + \sigma_{\text{bias}}^2)$. In the next section, we show that this residual error can be eliminated by applying server correction.

4.3 Distributed GCN via Server Correction

Before proceeding to our result, in order to simplify the presentation, let us first define the notation $G_{\text{global}}^r = \min_{t\in\mathcal{T}_{\text{global}}(r)}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2]$ and $G_{\text{local}}^r = \min_{t\in\mathcal{T}_{\text{local}}(r)}\mathbb{E}\left[\left\|\frac{1}{P}\sum_{p=1}^{P}\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t)\right\|^2\right]$ as the minimum gradient computed at the r-th round global and local step, where $\mathcal{T}_{\text{global}}(r)$ and $\mathcal{T}_{\text{local}}(r)$ are the number of iteration run after the r-th communication round on server and local machine, respectively. Please refer to Eq. 42 in Appendix C.2 for a formal definition.

Theorem 2 *Consider applying model averaging for GCN training under Assumption 1 and 2. If we choose learning rate $\gamma = \eta = \frac{\sqrt{P}}{\sqrt{T}}$, the local step size K, ρ such that $\sum_{r=1}^{R}K^2\rho^{2r} \leq \frac{RT^{1/2}}{32L^2P^{3/2}}$, and server correction step size $S = \max_{r\in[R]}\left(\frac{\kappa^2+2\sigma_{bias}^2}{1-L(\sqrt{P/T})} - G_{local}^r\right)\frac{K\rho^r}{G_{local}^r}$, then for any $T \geq L^2P$ steps of gradient updates we have: $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] = \mathcal{O}\left(\frac{1}{\sqrt{PT}}\right)$.*

Theorem 2 implies that, by carefully choosing the learning rates γ and η, the local step size hyperparameters K, ρ, and the number of global correction steps S, after T steps (R rounds of communication), employing parameter averaging with *Global Server Correction*, we have the norm of gradient bounded by $\mathcal{O}(1/\sqrt{PT})$, without suffering the residual error that exists in the naive parameter averaging (in Theorem 1). Besides, the server correction step size is proportional to the scale of κ^2 and local stochastic gradient bias σ_{bias}^2. The larger κ^2 and σ_{bias}^2, the more corrections are required to eliminate the residual error. However, in practice, we observe that a very small number of correction steps (e.g., $S = 1$) performs well, which minimizes the computation overhead on the server.

5 Experiments

Real-world simulation. In a real-world distributed setting, the server and local machines are located on different machines, connected through the network (Li et al., 2020a). However, for our experiments, we only have access to a single machine with multiple GPUs. As a result, we simulate a real-world distributed learning scenario, such that each GPU is responsible for the computation of two local machines (8 in total) and the CPU acts as the server. For these reasons, in our evaluations, we opted to report the communication size and number of communication rounds, instead of the wall-clock time, which can show the benefit of distributed training. We argue that these are acceptable measures in real-world scenarios as well since the two main factors in distributed training are initializing connection overhead and bandwidth (Tripathy et al., 2020).

Baselines. To illustrate the effectiveness of LLCG, we setup two general synchronized distributed training techniques as the our baseline methods, namely "*Parallel SGD with Parameter Averaging*" (PSGD-PA) and "*Global Graph Sampling*" (GGS), as introduced in Figure 2, where the cut-edges in PSGD-PA are ignored and only the model parameters are transferred, but the cut-edges in GGS are considered and the node features of the cut-edges are transferred to the corresponding machine.

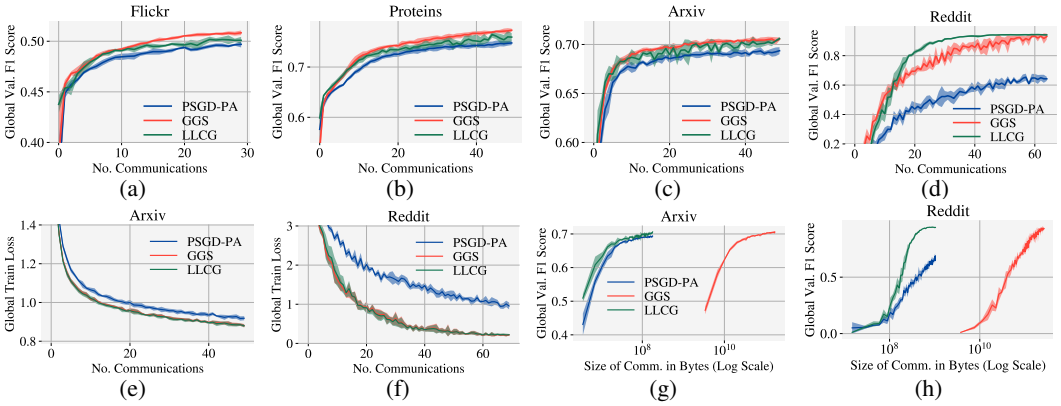


Figure 4: Comparing LLCG against PSGD-PA and GGS on real-world datasets. We show the global validation score in terms of the number of communications in (*a,b,c,d*), the training loss per round of communications in (*e,f*), and the global validation score per bytes of exchanged data in (*g,h*).

Table 1: Comparison of performance and the average Megabytes of node representation/feature communicated per round on various datasets.

	Method	No. Comm.	GCN / SAGE Performance	GCN / SAGE Avg. MB	GAT Performance	GAT Avg. MB	APPNP Performance	APPNP Avg. MB
`Flickr` (F1-score)	PSGD-PA	50	49.08±0.27	12.57	51.56±0.28	4.24	50.81±0.48	8.40
	GGS		51.22±0.13	1849.32	52.41±0.29	1895.61	51.33±0.33	1897.82
	LLCG		50.38±0.20	12.57	52.01±0.33	4.24	51.15±0.25	8.40
`OGB-Proteins` (ROC-AUC)	PSGD-PA	100	72.85±0.70	6.20	64.95±1.01	3.14	71.10±0.79	7.31
	GGS		74.78±0.36	922.42	68.11±0.60	912.79	71.29±0.31	917.20
	LLCG		73.92±0.45	6.20	67.62±0.58	3.14	71.18±0.43	7.31
`OGB-Arxiv` (F1-score)	PSGD-PA	100	69.43±0.21	3.55	69.88±0.18	3.59	68.48±0.17	7.71
	GGS		70.51±0.26	3391.03	70.82±0.23	3396.79	69.01±0.10	3394.33
	LLCG		70.21±0.13	3.55	70.58±0.37	3.59	68.73±0.29	7.71
`Reddit` (F1-score)	PSGD-PA	75	71.17±1.06	14.83	70.57±1.24	7.48	83.48±0.81	11.63
	GGS		94.77±0.20	3798.81	95.03±0.48	3805.28	95.23±0.22	3770.46
	LLCG		94.67±0.15	14.83	94.73±0.23	7.48	94.64±0.17	11.63

Note that we choose GGS as a reasonable representation for most existing proposals (Md et al., 2021; Zheng et al., 2020; Tripathy et al., 2020) for distributed GNN training, since these methods have very close communication cost and also require a large cluster of machines to truly show their performance improvement. We also use PSGD-PA as a lower bound for communication size, which is widely used in traditional distributed training and similar to the one used in (Angerd et al., 2020; Jiang & Rumi, 2021). However, we did not specifically include these methods in our results since we could not reproduce their results in our settings. Please refer to Appendix A for a detailed description of implementation, hardware specification and link to our source code.

Datasets and evaluation metric. We compare LLCG and other baselines on real-world semi-supervised node classification datasets, detail of which are summarized in Table 2 in the Appendix. The input graphs are splitted into multiple subgraphs using `METIS` before training, then the same set of subgraphs are used for all baselines. For training, we use neighborhood sampling (Hamilton et al., 2017) with 10 neighbors sampled per node and $\rho = 1.1$ for LLCG. For a fair comparison, we chose the base local update step K such that LLCG has the same number of local update steps as PSGD-PA. During evaluation, we use full-batch without sampling, and report the performance on the full graph using AUC ROC and F1 Micro as the evaluation metric. Unless otherwise stated, we conduct each experiment five times and report the mean and standard deviation.

5.1 Primary Results

In this section, we compare our proposed LLCG algorithm with baselines on four datasets. Due to space limitations we defer the detailed discussion on additional datasets to the Appendix A.4.

LLCG requires same number of communications. Figure 4 (a) through 4 (d) illustrate the validation accuracy per communication rounds on four different datasets. We run a fixed number of communication rounds and plot the global validation score (the validation score computed using the full-graph on the server) at the end of each communication step. For PSGD-PA and GGS, the score is calculated on the averaged model, whereas for LLCG the validation is calculated after the correction

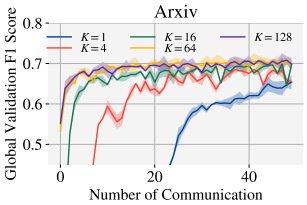


Figure 5: Effect of local epoch size (K)

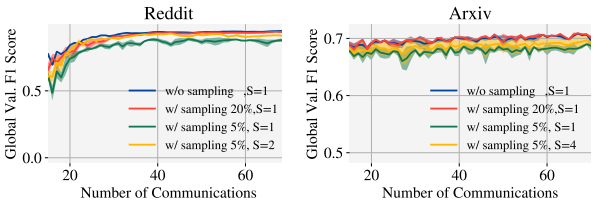


Figure 6: Effect of sampling on local machine and number of correction steps on the server

step. It can be seen that PSGD-PA suffers from performance drop compared to other two methods, due to the residual error we discussed in Section 4, while both GGS and LLCG perform well and can achieve the expected accuracy. Note that the performance drop of PSGD-PA can vary across different datasets; in some cases such as `Reddit`, PSGD-PA can significantly hurt the accuracy, while on other datasets the gap is smaller. Nevertheless, LLCG can always close the gap between PSGD-PA and GGS with minimal overhead.

LLCG convergences as fast as GGS. To represent the effect of communication on the real-time convergence, in Figure 4 (e) and 4 (f), we plot the global training loss (training loss computed on the full-graph on the server) after each communication round. Similar to the accuracy score, the training loss is also computed on the server averaged (and corrected, in case of LLCG) global model. These results clearly indicate that LLCG can improve the convergence over PSGD-PA, while it shows a similar convergence speed to GGS.

LLCG exchanges data as little as PSGD-PA. Figure 4 (g) and 4 (h) show the relation between global validation accuracy with the average size (volume) of communication in bytes. As expected, this figure clearly shows the effectiveness of LLCG . On the one hand, LLCG has a similar amount of communication volume as PSGD-PA but can achieve a higher accuracy. On the other hand, LLCG requires significantly less amount of communication volume than GGS to achieve the same accuracy, which leads to slower training time in real world settings.

LLCG works with various GNN models and aggregations. We evaluate four popular GNN models, used in recent graph learning literature: GCN Kipf & Welling (2017), SAGE Hamilton et al. (2017), GAT Velickovic et al. (2018) and APPNP Klicpera et al. (2019). In Table 1, we summarize the test score and average communication size (in MB) on different datasets for a fixed number of communication rounds. Note that we only include the results for the aggregation methods (GCN or SAGE) that have higher accuracy for the specific datasets, details of which can be found in Appendix A.2. As shown here, LLCG can consistently improve the test accuracy for all different models compared to PSGD-PA, while the communication size is significantly lower than GGS, since LLCG only needs to exchange the model parameters.

Effect of local epoch size. Figure 5 compares the effect of various values of local epoch size $K \in \{1, 4, 16, 64, 128\}$ for fixed ρ and S on the `OGB-Arxiv` dataset. When using *fully synchronous* with $K = 1$, the model suffers from very slow convergence and needs more communications. Further increasing the K to larger values can speed up the training; however, we found a diminishing return point for $K > 128$ in this dataset and extremely large K in general.

Effect of sampling in local machines. In Figure 6, we report the validation scores per round of communication to compare the effect of neighborhood sampling at local machines. We can observe that when the neighborhood sampling size is reasonably large (i.e., 20%), the performance is very similar to full neighborhood training. However, reducing the neighbor sampling ratio to 5% could result in a larger local stochastic gradient bias σ^2_{bias}, which requires using more correction steps (S).

6 Concluding Remarks

In this paper, we propose a novel distributed algorithm for training Graph Neural Networks (GNNs). We theoretically analyze various GNN models and discover that, unlike the traditional deep neural networks, due to inherent data samples dependency in GNNs, naively applying periodic parameter averaging leads to a residual error and current solutions to this issue impose huge communication overheads. Instead, our proposal tackles these problems by applying correction on top of locally learned models, to infuse the global structure of the graph back into the network and avoid any costly communication. In addition, through extensive empirical analysis, we support our theoretical findings and demonstrate that LLCG can achieve high accuracy without additional communication costs.

ACKNOWLEDGEMENTS

This work was supported in part by CRISP, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA and NSF grants 1909004, 1714389, 1912495, 1629915, 1629129, 1763681, 2008398.

REPRODUCIBILITY STATEMENT

We provide a GitHub repository in Appendix A including all code and scripts used in our experimental studies. This repository includes a `README.md` file, explaining how to install and prepare the code and required packages. Detailed instruction on how to use the partitioning scripts is provided for various datasets. In addition, we provide several configuration files (under `scripts/configs`) folder for different hyper-parameters on each individual dataset, and a general script (`scripts/run-config.py`) to run and reproduce the results with these configurations. Details of various models and parameters used in our evaluation studies can also be found in Appendix A.

REFERENCES

Alexandra Angerd, Keshav Balasubramanian, and Murali Annavaram. Distributed training of graph convolutional networks using subgraph approximation. *arXiv preprint arXiv:2012.04930*, 2020.

Paolo Boldi and Sebastiano Vigna. The webgraph framework I: compression techniques. In *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*, pp. 595–602. ACM, 2004. doi: 10.1145/988672.988752.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*. mlsys.org, 2019.

Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 941–949. PMLR, 2018.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pp. 257–266. ACM, 2019. doi: 10.1145/3292500.3330925.

Weilin Cong, Rana Forsati, Mahmut T. Kandemir, and Mehrdad Mahdavi. Minimal variance sampling with provable guarantees for fast training of graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pp. 1393–1403. ACM, 2020.

Weilin Cong, Morteza Ramezani, and Mehrdad Mahdavi. On the importance of sampling in learning graph convolutional networks. *arXiv preprint arXiv:2103.02696*, 2021.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew W. Senior, Paul A. Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pp. 1232–1240, 2012.

Songgaojun Deng, Huzefa Rangwala, and Yue Ning. Learning dynamic context graphs for predicting social events. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pp. 1007–1016. ACM, 2019. doi: 10.1145/3292500.3330919.

Kien Do, Truyen Tran, and Svetha Venkatesh. Graph transformation policy network for chemical reaction prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pp. 750–760. ACM, 2019. doi: 10.1145/3292500.3330958.

Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6530–6539, 2017.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015.

Peng Jiang and Masuma Akter Rumi. Communication-efficient sampling for distributed training of graph convolutional networks. *arXiv preprint arXiv:2101.07706*, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Jakub Konečnỳ, H Brendan McMahan, X Yu Felix, Ananda Theertha Suresh, Dave Bacon, and Peter Richtárik. Federated learning: Strategies for improving communication efficiency. 2018.

Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 9266–9275. IEEE, 2019. doi: 10.1109/ICCV.2019.00936.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020a.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b.

Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: A research platform for federated learning in natural language processing. *arXiv preprint arXiv:2104.08815*, 2021.

Vasimuddin Md, Sanchit Misra, Guixiang Ma, Ramanarayan Mohanty, Evangelos Georganas, Alexander Heinecke, Dhiraj Kalamkar, Nesreen K Ahmed, and Sasikanth Avancha. Distgnn: Scalable distributed training for large-scale graph neural networks. *arXiv preprint arXiv:2104.06700*, 2021.

Simone Scardapane, Indro Spinelli, and Paolo Di Lorenzo. Distributed graph convolutional networks. *arXiv preprint arXiv:2007.06281*, 2020.

Hyejin Shin, Sungwook Kim, Junbum Shin, and Xiaokui Xiao. Privacy enhanced matrix factorization for recommendation with local differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1770–1782, 2018.

Sebastian U. Stich. Local SGD converges fast and communicates little. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Chuxiong Sun and Guoshi Wu. Adaptive graph diffusion networks with hop-wise attention. *arXiv preprint arXiv:2012.15024*, 2020.

Chuxiong Sun and Guoshi Wu. Scalable and adaptive graph neural networks with self-label-enhanced training. *arXiv preprint arXiv:2104.09376*, 2021.

Alok Tripathy, Katherine Yelick, and Aydin Buluc. Reducing communication in graph neural network training. *arXiv preprint arXiv:2005.03300*, 2020.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 839–848. ACM, 2018. doi: 10.1145/3219819.3219869.

Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925*, 2021.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 974–983. ACM, 2018. doi: 10.1145/3219819.3219890.

Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5693–5700, 2019.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. Graphsaint: Graph sampling based inductive learning method. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Qingru Zhang, David Wipf, Quan Gan, and Le Song. A biased graph neural network sampler with near-optimal regret. *arXiv preprint arXiv:2103.01089*, 2021.

Da Zheng, Chao Ma, Minjie Wang, Jinjing Zhou, Qidong Su, Xiang Song, Quan Gan, Zheng Zhang, and George Karypis. Distdgl: Distributed graph neural network training for billion-scale graphs. *arXiv preprint arXiv:2010.05337*, 2020.

Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11247–11256, 2019.

SUPPLEMENTARY MATERIAL
LEARN LOCALLY, CORRECT GLOBALLY: A DISTRIBUTED ALGORITHM FOR TRAINING GRAPH NEURAL NETWORKS

Appendix

Table of Contents

A Additional experimental setup and studies **14**
- A.1 Hardware specification and environment . . . 14
- A.2 Dataset and models details . . . 14
- A.3 Minibatch selection for correction step . . . 15
- A.4 Effectiveness of correction . . . 16
- A.5 Effectiveness in large-scale settings . . . 16

B Proof of Theorem 1 **17**
- B.1 Useful lemmas . . . 17
- B.2 Main proof of Theorem 1 . . . 18
- B.3 Discussion on the irreducible error . . . 21
- B.4 Proof of Lemma 1 . . . 23
- B.5 Proof of Lemma 2 . . . 23
- B.6 Proof of Lemma 3 . . . 23

C Proof of Theorem 2 **26**
- C.1 Useful lemma . . . 26
- C.2 Main proof of Theorem 2 . . . 26
- C.3 Proof of Lemma 4 . . . 29

A ADDITIONAL EXPERIMENTAL SETUP AND STUDIES

We provide detailed experimental setup and additional evaluations in this section. The link to the GitHub repository is available at https://github.com/MortezaRamezani/llcg/.

A.1 HARDWARE SPECIFICATION AND ENVIRONMENT

For all our experiments we use a single server equipped with 4 NVIDIA QUADRO RTX 8000 GPUs with driver version 460.80, two Intel Xeon 6230 CPU and 768GB of main memory using Ubuntu 18.04 running kernel 5.4.0. For fair comparison between all methods, we developed a unified framework for performing the experiments, using Pytorch 1.7 compiled for CUDA 11.2. We also used Pytorch Geometric 1.7.0 and Pytorch Sparse 0.6.8 for various GNN and sparse operations. Please refer to README.md for detailed instruction on how to install and run the code, and also dataset specific configurations.

Table 2: Summary of datasets statistics. ‡ indicates multi-labels dataset.

Dataset	Nodes	Edges	Feature	Classes	Train / Validation / Test	Base Arch.
Flickr Zeng et al. (2020)	89,250	899,756	500	7	50% / 25% / 25%	BSBSBL
OGB-Proteins Hu et al. (2020)	132,534	39,561,252	8	112‡	65% / 16% / 19%	SSS
OGB-Arxiv Hu et al. (2020)	169,343	1,166,243	12	40	54% / 17% / 28%	GBGBG
Reddit Hamilton et al. (2017)	232,965	11,606,919	602	41	66% / 10% / 24%	SBSBS
Yelp Zeng et al. (2020)	716,847	13,954,819	300	100‡	75% / 15% / 10%	BSBSBL
OGB Products Hu et al. (2020)	2,449,029	61,859,140	100	47	8% / 2% / 90%	GGG

A.2 DATASET AND MODELS DETAILS

In all experiments, we used ADAM optimizer, and the learning rate, alongside the local machine mini-batch size (B_L) and server mini-batch size (B_S) are included in config files under scripts/configs folder. We used three datasets Flickr, Reddit, Yelp from Zeng et al. (2020) and OGB-Proteins, OGB-Arxiv, OGB-Products from Hu et al. (2020). For each dataset, we choose the aggregation method that has been shown to achieve the highest accuracy, and reported the results in Table 1. In addition, we also evaluate two frequently used architectures, GAT and APPNP to show the flexibility of our proposed method to other models. Note that some of OGB datasets can use more complicated models Sun & Wu (2021; 2020), however for the sake of simplicity, we only used popular GNN operators which show competitive results in most cases. Next we formally introduce various layers and operators used in our experiments, followed by the base architecture for each dataset in Table 2.

- **GCN (G)**: Originally introduced by Kipf et. al Kipf & Welling (2017), the representation for node v_i is calculated using:
$$\mathbf{h}_i^{(\ell)} = \sigma\Big(\sum_{j\in\mathcal{N}(v_i)} \alpha_{i,j}\mathbf{h}_j^{(\ell-1)}\mathbf{W}^{(\ell)}\Big), \tag{6}$$
where $\alpha_{i,j} = \frac{1}{\sqrt{\deg(v_i)\deg(v_j)}}$ for symmetric normalized Laplacian and $\alpha_{i,j} = \frac{1}{\deg(v_i)}$ for row normalized Laplacian.

- **SAGE (S)**: First introduce in Hamilton et al. (2017), SAGE learn different weights for the node and its neighbors and the representation is calculated by:
$$\mathbf{h}_i^{(\ell)} = \sigma\Big(\mathbf{h}_i^{(\ell-1)}\mathbf{W}_1^{(\ell)} + \frac{1}{|\mathcal{N}(v_i)|}\sum_{j\in\mathcal{N}(v_i)} \mathbf{h}_j^{(\ell-1)}\mathbf{W}_2^{(\ell)}\Big) \tag{7}$$
Note that different operator such as concatenation can be used instead of addition in the above formula. However in our experiments we used addition for all SAGE layers.

- **Linear (L)**: Applies a linear transformation to the node features. In other word, the graph structured is completely ignored in this layer and the representation is computed as follows.
$$\mathbf{h}_i^{(\ell)} = \mathbf{h}_i^{(\ell-1)}\mathbf{W}^{(\ell)} \tag{8}$$

- **BatchNorm (B)**: Applies batch normalization according to Ioffe & Szegedy (2015) using the following where γ and β are learnable parameters and ϵ is added for numerical stability.

$$\mathbf{h}_i^{(\ell)} = \frac{\mathbf{h}_i^{(\ell-1)} - \mathbb{E}(\mathbf{h}_i^{(\ell-1)})}{\sqrt{\mathbf{Var}(\mathbf{h}_i^{(\ell-1)}) + \epsilon}} * \gamma + \beta \tag{9}$$

- **GAT**: The Graph Attention layer proposed by (Velickovic et al., 2018) as follows.

$$\mathbf{h}_i^{(\ell)} = \sigma\Big(\sum_{j \in \mathcal{N}(v_i)} \alpha_{i,j} \mathbf{h}_j^{(\ell-1)} \mathbf{W}^{(\ell)}\Big) \tag{10}$$

where $\alpha_{i,j}$ is the attention between node i and j and calculated by:

$$\alpha_{i,j} = \frac{exp(\text{LeakyRelu}(\boldsymbol{a}[\mathbf{h}_i^{(\ell-1)}\mathbf{W}^{(\ell)} || \mathbf{h}_j^{(\ell-1)}\mathbf{W}^{(\ell)}]))}{\sum_{k \in \mathcal{N}(v_i)} exp(\text{LeakyRelu}(\boldsymbol{a}[\mathbf{h}_i^{(\ell-1)}\mathbf{W}^{(\ell)} || \mathbf{h}_k^{(\ell-1)}\mathbf{W}^{(\ell)}]))} \tag{11}$$

where $\boldsymbol{a}$ is the learnable weight vector and $||$ indicates concatenation.

- **APPNP**: This network is proposed by (Klicpera et al., 2019), which is a combination of graph-agnostic prediction and label propagation:

$$\mathbf{h}_i^{(\ell)} = \beta \mathbf{h}_i^{(0)} + (1-\beta) \sum_{j \in \mathcal{N}(v_i)} \alpha_{i,j} \mathbf{h}_j^{(\ell-1)} \tag{12}$$

where $\alpha_{i,j}$ is same as defined in GCN, β is the teleport probability and $\mathbf{h}^{(0)}$ is the output of first linear layer.

Effect of sampling at correction. Recall that LLCG requires full-neighbors for the server correction step for the convergence analysis, however, we find that server correction with neighbor sampling also works well in practice. As shown in Figure 7 and Figure 8, although server correction with neighbor sampling can introduce some randomness at the beginning of the training phase, the final accuracy of training is very close to the server correction with full-neighbors.

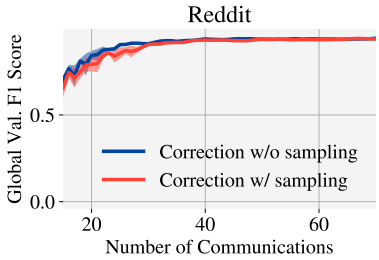


Figure 7: Impact of sampling in correction steps on the `Reddit` dataset.

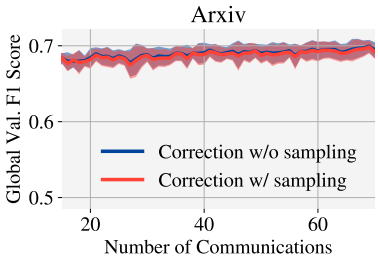


Figure 8: Impact of sampling in correction steps on the `Arxiv` dataset.

A.3 Minibatch Selection for Correction Step

Recall that in a server correction step, the mini-batch is selected by sampling from the entire graph, uniformly at random to estimate an unbiased stochastic gradient of the global loss function. However, one might suggest to include more cut-edges, which are missing from the local machines, in the correction minibatch instead of uniform random sampling, to improve the performance of the global model. To this end, we conduct an experiments on two datasets (`Reddit` and `Arxiv`), where we compare the random minibatch (default setting in LLCG) and minibatch with higher number of cut-edges, by selecting the nodes on the ends of cut-edges and building the minibatch from there. As shown in Figure 9, including more cut-edges nodes in the mini-batch does not make significant improvement when comparing to selecting mini-batchs using uniform sampling. This potentially happens due to the fact that sampling more cut-edges make the gradient of server correction step biased, while we need an unbiased gradient of full graph for correction steps at server.

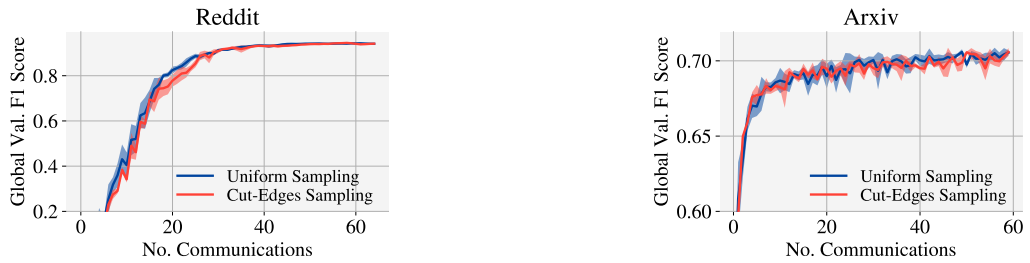


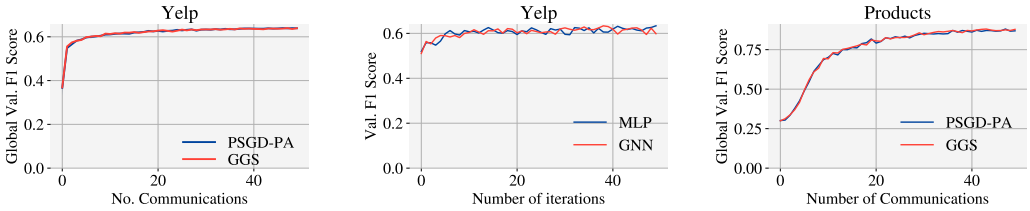
Figure 9: Comparing the validation F1-score of uniform sampling and sampling more cut-edges (i.e., *max. cut edges mini-batch*) on the `Reddit` and `Arxiv` dataset.

A.4 Effectiveness of correction

To illustrate the effectiveness of correction steps and relate it to the benefits of global graph structure, we conduct the same experiments on `Yelp` and `OGB-Products` with 700K and 2.4M nodes, respectively and report the results in Figure 10.

Yelp dataset. In case of `Yelp` in Figure 10 (a), as we can see the PSGD-PA and GGS are performing quite similarly. To further investigate this, we compared the validation accuracy of Yelp using GCN against MLP (i.e., without utilizing the graph structure in training), where the GCN layers replaced by Linear layer (i.e LLL instead of GGG architecture) and plot the validation accuracy per iterations in Figure 10 (b). As it can be deducted from this figure, for this dataset MLP is performing as good as GCN, which means this dataset does not depend on the global structure of the graph, further explaining why there is no performance gap between PSGD-PA and GGS. In other words, no server correction is necessary and we can use $S = 0$ in this case for LLCG.

OGB-Products. Similarly, in case of `OGB-Products` in Figure 10 (c), we cannot see any noticeable accuracy drop due to distributed training. However, unlike `Yelp`, this is mainly due to very small κ which is caused by two main factors: (1) very small ratio of training nodes for this dataset, only 8% of the nodes are used for training, and (2) very small number of cut-edges (less than 7%) after applying `METIS` on this dataset. It is also worth mentioning that due to shallow structure of the model required for this dataset, we barely need multi-hop neighbors and consequently we cannot see any noticeable performance drop.



(a) (b) (c)

Figure 10: Comparing (a) PSGD-PA vs GGS and (b) MLP vs GNN on the `Yelp` dataset. (c) Comparing PSGD-PA and GGS for `OGB-Products`. We exclude the error bar from the figure for better visibility.

A.5 Effectiveness in large-scale settings

In this section, we further investigate the effect of large number of local machines and large graphs on the performance of proposed algorithm, under the simulated environment with up to 16 local machines on `OGB-Products` and `OGB-MAG240M` Hu et al. (2021) datasets for node classification. Besides,

we also compare against subgraph approximation distributed GNN training algorithm Angerd et al. (2020) using 10% extra storage overhead and fully synchronous distributed GNN training.[3]

Node classification tasks. For the `OGB-Products` dataset, we train a 3-layer GraphSAGE model with learning rate 0.003 with 50 rounds of communications , which has the same hyper-parameter configuration and model architecture as `OGB-Products`'s leaderboard implementation . For the `OGB-MAG240M` dataset, we train a 2-layer skip-connected GraphSAGE model with learning rate 0.001 with 400 rounds of communications, which has the same hyper-parameter selection and model architecture as `OGB-MAG240M`'s leaderboard implementation. As shown in Figure 11, we can observe that

- PSGD-PA suffers from performance degeneration and has a large gap to the full sync training accuracy. Subgraph approximation might alleviate the issue to some extent, but requires significant storage overhead. Our proposal LLCG can bridge the gap between PSGD-PA and full sync training.
- The pure computation time[4] is negligible when ignoring the communication time but just consider the computation time. Besides, the server correction step does not introduce significant computation overhead.

It is worth mentioning that we do not observe the performance gap between PSGD-PA and fully sync distributed training in Figure 10 (c) because of different number of layers (2-layers) and limited number of local machine (8 local machines) are used in the previous experiments. However, as the number of layers and number of local machines increase, the effect of ignoring cut-edges and data heterogeneity is becoming more serious, which is an interesting observation and worth exploring as a potential future direction.

B Proof of Theorem 1

In this section, we first introduce the useful lemmas in Section B.1, then process our proof of theorem in Section B.2. In particular, we show that solely averaging the local machine models and ignoring the global graph structure will suffer from an irreducible residual error.

B.1 Useful lemmas

The following lemma gives the upper bound for the norm of the stochastic gradient on each local machine as the norm of the expectation of local gradient and the stochastic gradient variance scaled by the number of local machines P.

Lemma 1 *Let $\tilde{\nabla}\mathcal{L}_p^{local}(\boldsymbol{\theta}_p;\xi_p)$ be stochastic gradients such that*

$$\mathbb{E}\left[\left\|\tilde{\nabla}\mathcal{L}_p^{local}(\boldsymbol{\theta}_p;\xi_p)-\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{local}(\boldsymbol{\theta}_p;\xi_p)]\right\|^2\right]\leq\sigma_{var}^2, \tag{13}$$

Then, we have

$$\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\tilde{\nabla}\mathcal{L}_p^{local}(\boldsymbol{\theta}_p;\xi_p)\Big\|^2\Big]\leq\frac{\sigma_{var}^2}{P}+\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{local}(\boldsymbol{\theta}_p,\xi_p^t)\right]\Big\|^2\Big] \tag{14}$$

The following lemma provides an upper bound on the difference of each local gradient $\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p)$ compared to $\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q)$, which is a function of:

- The deviation of each local model to the virtual averaged model, i.e., $\frac{1}{P}\sum_{i=1}^{P}\|\boldsymbol{\theta}_p-\bar{\boldsymbol{\theta}}\|^2$.
- The difference of gradient computed with and without having access to the full graph information, i.e., κ^2.

[3] Suppose there are N_p nodes on the p-th local machine, then subgraph of size $10\%\times N_p$ is sampled and stored on the pth local machine. Notice that 10% storage overhead is the maximum overhead recommended by Angerd et al. (2020), which is expected to bring the best accuracy performance.

[4] We record the time using Python's `time.time()` function.

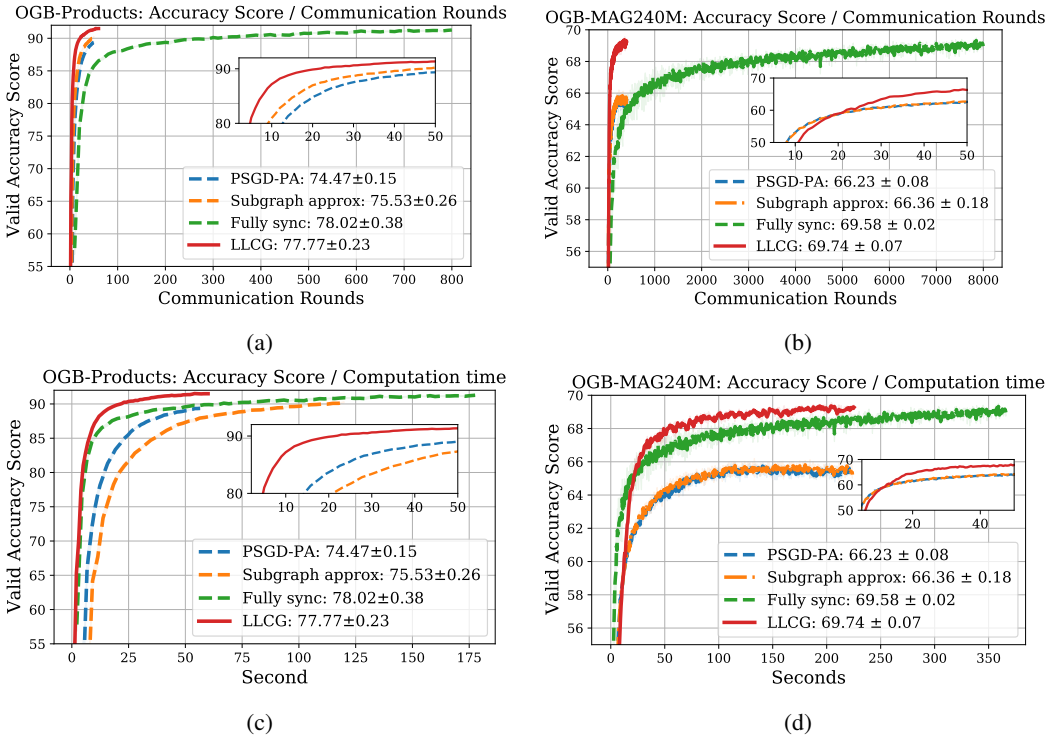


Figure 11: Comparing PSGD-PA, periodic model averaging with subgraph approximation, full sync distributed GNN training, and LLCG on `OGB-Products` and `OGB-MAG240M` datasets on a simulated enviroment with 16 local machines.

Lemma 2 *For any P machines $\boldsymbol{\theta}_p,\ p \in \{1,\dots,P\}$, if we define $\bar{\boldsymbol{\theta}} = \frac{1}{P}\sum_{i=1}^{P}\boldsymbol{\theta}_p$ and let $\kappa > 0$ such that $\|\nabla\mathcal{L}_p^{local}(\boldsymbol{\theta}) - \nabla\mathcal{L}(\boldsymbol{\theta})\|^2 \leq \kappa^2$, we have*

$$\frac{1}{P}\sum_{p=1}^{P}\left\|\nabla\mathcal{L}_p^{local}(\boldsymbol{\theta}_p) - \frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}_q^{local}(\boldsymbol{\theta}_q)\right\|^2 \leq \frac{8L^2}{P}\sum_{i=1}^{P}\|\boldsymbol{\theta}_p - \bar{\boldsymbol{\theta}}\|^2 + 8\kappa^2. \tag{15}$$

The following lemma provides an upper bound on the deviation of each local model to virtual averaged model, which plays an important role in the previous lemma.

Lemma 3 *For all P machines with parameters $\boldsymbol{\theta}_p,\ p \in \{1,\dots,P\}$, if we define $\bar{\boldsymbol{\theta}} = \frac{1}{P}\sum_{i=1}^{P}\boldsymbol{\theta}_p$ and let $\kappa > 0$ such that $\|\mathcal{L}_p^{local}(\boldsymbol{\theta}) - \nabla\mathcal{L}(\boldsymbol{\theta})\|^2 \leq \kappa^2$, we have*

$$\sum_{t=0}^{T-1}\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2] \leq \frac{4\eta^2 K\sigma_{var}^2}{1-16L^2\eta^2K^2}T + \frac{4\eta^2K^2\sigma_{bias}^2 + 16\eta^2K^2\kappa^2}{1-16L^2\eta^2K^2}T. \tag{16}$$

B.2 Main proof of Theorem 1

Equipped with the above lemmas, we are now ready to present the proof of Theorem 1. From the smoothness assumption, we have

$$\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})] \leq \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)] + \mathbb{E}[\langle\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\rangle] + \frac{L}{2}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\|^2]. \tag{17}$$

Recall that $\bar{\boldsymbol{\theta}}^t$ is defined as $\bar{\boldsymbol{\theta}}^t = \frac{1}{P}\sum_{i=1}^{P}\boldsymbol{\theta}_p^t$ for any t. Then, according to the update rule

$$\bar{\boldsymbol{\theta}}^{t+1} = \bar{\boldsymbol{\theta}}^t - \frac{\eta}{P}\sum_{p=1}^{P}\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t;\xi_p^t), \tag{18}$$

we have

$$\begin{aligned}\mathbb{E}[\langle \nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \rangle] &= -\eta \mathbb{E}\Big[\Big\langle \nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t), \frac{1}{P}\sum_{p=1}^{P} \tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t; \xi_p^t) \Big\rangle\Big] \\ &= -\eta \Big\langle \nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t), \frac{1}{P}\sum_{p=1}^{P} \mathbb{E}\left[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t; \xi_p^t)\right] \Big\rangle.\end{aligned} \tag{19}$$

We can upper bound the right hand side of Eq. 19 by

$$\begin{aligned}&-\eta \Big\langle \nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t), \frac{1}{P}\sum_{p=1}^{P} \mathbb{E}[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t; \xi_p^t)] \Big\rangle \\ &\underset{(a)}{=} -\frac{\eta}{2}\Big(\|\nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2 + \Big\|\frac{1}{P}\sum_{p=1}^{P} \mathbb{E}[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t; \xi_p^t)]\Big\|^2 - \Big\|\nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t) - \frac{1}{P}\sum_{p=1}^{P} \mathbb{E}[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t; \xi_p^t)]\Big\|^2\Big) \\ &\underset{(b)}{\leq} -\frac{\eta}{2}\Big(\|\nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2 + \Big\|\frac{1}{P}\sum_{p=1}^{P} \mathbb{E}[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t; \xi_p^t)]\Big\|^2 - 2\kappa^2 - 4\sigma_{\text{bias}}^2 - \frac{4L^2}{P}\sum_{p=1}^{P} \|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2\Big),\end{aligned} \tag{20}$$

where (a) is due to $2\langle \mathbf{x}, \mathbf{y} \rangle = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \|\mathbf{x} - \mathbf{y}\|^2$ and (b) is due to

$$\begin{aligned}&\Big\|\nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t) - \frac{1}{P}\sum_{p=1}^{P} \mathbb{E}[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t; \xi_p^t)]\Big\|^2 \\ &= \Big\|\frac{1}{P}\sum_{p=1}^{P}\Big(\nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t) - \nabla \mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}}^t) + \nabla \mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}}^t) - \mathbb{E}[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t; \xi_p^t)]\Big)\Big\|^2 \\ &\leq \frac{2}{P}\sum_{p=1}^{P} \|\nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t) - \nabla \mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}}^t)\|^2 + \frac{4}{P}\sum_{p=1}^{P} \|\nabla \mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}}^t) - \nabla \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t)\|^2 \\ &\quad + \frac{4}{P}\sum_{p=1}^{P} \Big\|\mathbb{E}[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}}^t; \xi_p^t)] - \nabla \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t)\Big\|^2 \\ &\underset{(c)}{\leq} 2\kappa^2 + 4\sigma_{\text{bias}}^2 + \frac{4L^2}{P}\sum_{p=1}^{P} \|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2,\end{aligned} \tag{21}$$

where (c) follows from smoothness assumption and the definition of $\kappa^2, \sigma_{\text{bias}}^2$ in Theorem 1.

Combining Eq. 19 and Eq. 20 gives us

$$\begin{aligned}&\mathbb{E}[\langle \nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \rangle] \\ &\leq -\frac{\eta}{2}\mathbb{E}[\|\nabla \mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] - \frac{\eta}{2}\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P} \mathbb{E}\left[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t, \xi_p^t)\right]\Big\|^2\Big] \\ &\quad + \eta(\kappa^2 + 2\sigma_{\text{bias}}^2) + \frac{2\eta L^2}{P}\sum_{p=1}^{P} \mathbb{E}[\|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2].\end{aligned} \tag{22}$$

According to the update rule

$$\bar{\boldsymbol{\theta}}^{t+1} = \bar{\boldsymbol{\theta}}^t - \frac{\eta}{P}\sum_{p=1}^{P} \tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t; \xi_p^t), \tag{23}$$

we have

$$\mathbb{E}[\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\|^2] = \eta^2 \mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P} \tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t; \xi_p^t)\Big\|^2\Big]. \tag{24}$$

Substituting Eq. 22 and Eq. 24 to Eq. 17, we have

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})] \leq \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t})] - \frac{\eta}{2}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^{t})\|^2] - \frac{\eta}{2}\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t,\xi_p^t)\right]\Big\|^2\Big] \\ + \eta(\kappa^2 + 2\sigma_{\text{bias}}^2) + \frac{2\eta L^2}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2] \\ + \frac{\eta^2 L}{2}\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t;\xi_p^t)\Big\|^2\Big].\end{aligned} \tag{25}$$

Dividing both sides by $\frac{\eta}{2}$ and rearranging the terms yields

$$\begin{aligned}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^{t})\|^2] &\leq \frac{2}{\eta}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t})] - \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big) - \mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t,\xi_p^t)\right]\Big\|^2\Big] \\ &\quad + 2(\kappa^2 + 2\sigma_{\text{bias}}^2) + \frac{4L^2}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2] + \eta L\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t;\xi_p^t)\Big\|^2\Big] \\ &\underset{(a)}{\leq} \frac{2}{\eta}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t})] - \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big) - \mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t,\xi_t^p)\right]\Big\|^2\Big] \\ &\quad + 2(\kappa^2 + 2\sigma_{\text{bias}}^2) + \frac{4L^2}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2] \\ &\quad + \eta L\Big(\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t,\xi_p^t)\right]\Big\|^2\Big] + \frac{\sigma_{\text{var}}^2}{P}\Big) \\ &= \frac{2}{\eta}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t})] - \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big) + (\eta L - 1)\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t,\xi_p^t)\right]\Big\|^2\Big] \\ &\quad + 2(\kappa^2 + 2\sigma_{\text{bias}}^2) + \frac{4L^2}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2] + \frac{\eta L\sigma_{\text{var}}^2}{P},\end{aligned} \tag{26}$$

where (a) is due to Lemma 1.

Summing over $t \in \{0, \dots, T-1\}$ and dividing both side by T, we get

$$
\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] \\
&\leq \frac{2}{\eta T}\Big(\mathcal{L}(\bar{\boldsymbol{\theta}}^0)-\mathcal{L}(\boldsymbol{\theta}^\star)\Big)+\frac{\eta L-1}{T}\sum_{t=0}^{T-1}\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t,\xi_p^t)\right]\Big\|^2\Big] \\
&\quad+2(\kappa^2+2\sigma_{\text{bias}}^2)+\frac{4L^2}{T}\sum_{t=0}^{T-1}\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2]+\frac{\eta L\sigma_{\text{var}}^2}{P} \\
&\underset{(a)}{\leq} \frac{2}{\eta T}\Big(\mathcal{L}(\bar{\boldsymbol{\theta}}^0)-\mathcal{L}(\boldsymbol{\theta}^\star)\Big)+\frac{\eta L-1}{T}\sum_{t=0}^{T-1}\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t,\xi_p^t)\right]\Big\|^2\Big]+\frac{\eta L\sigma_{\text{var}}^2}{P} \\
&\quad+2(\kappa^2+2\sigma_{\text{bias}}^2)+L^2\frac{16\eta^2K\sigma_{\text{var}}^2}{(1-16\eta^2L^2K^2)}+L^2\frac{16\eta^2K^2\sigma_{\text{bias}}^2+64\eta^2K^2\kappa^2}{(1-16\eta^2L^2K^2)} \\
&\underset{(b)}{\leq} \frac{2}{\eta T}\Big(\mathcal{L}(\bar{\boldsymbol{\theta}}^0)-\mathcal{L}(\boldsymbol{\theta}^\star)\Big)+2(\kappa^2+2\sigma_{\text{bias}}^2)+\frac{\eta L\sigma_{\text{var}}^2}{P} \\
&\quad+L^2\frac{16\eta^2K\sigma_{\text{var}}^2}{(1-16\eta^2L^2K^2)}+L^2\frac{16\eta^2K^2\sigma_{\text{bias}}^2+64\eta^2K^2\kappa^2}{(1-16\eta^2L^2K^2)} \\
&\underset{(c)}{\leq} \frac{2}{\eta T}\Big(\mathcal{L}(\bar{\boldsymbol{\theta}}^0)-\mathcal{L}(\boldsymbol{\theta}^\star)\Big)+2(\kappa^2+2\sigma_{\text{bias}}^2)+\frac{\eta L\sigma_{\text{var}}^2}{P} \\
&\quad+32L^2\eta^2K\sigma_{\text{var}}^2+128L^2\eta^2K^2\kappa^2+32L^2\eta^2K^2\sigma_{\text{bias}}^2,
\end{aligned}
\tag{27}
$$

where (a) is due to Lemma 3, (b) is due to $0 < \eta < \frac{1}{L}$, and (c) is due to $K \leq \frac{\sqrt{2}}{8L\eta}$ is selected to satisfy $1 - 16\eta^2L^2K^2 \geq \frac{1}{2}$.

If we choose $\eta = \frac{\sqrt{P}}{\sqrt{T}}$ and $K \leq \frac{\sqrt{2}T^{1/4}}{8LP^{3/4}} = \mathcal{O}(\frac{T^{1/4}}{P^{3/4}})$, then for any $T \geq L^2P$ we have

$$
\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] = \mathcal{O}\left(\frac{1}{\sqrt{PT}}\right)+\mathcal{O}(\kappa^2+\sigma_{\text{bias}}^2),
\tag{28}
$$

with $R = \mathcal{O}(P^{3/4}T^{3/4})$ rounds of communication.

B.3 Discussion on the irreducible error

To better understand why the error in the upper bound might be irreducible (i.e., the second term in RHS of Theorem 1 is independent of T), let's first recall the key sources that cause gradient diversity $\left\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t) - \frac{1}{P}\sum_{p=1}^{P}\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t;\xi_p^t)]\right\|^2$ (in Eq. 21), in our setting:

- The κ^2 term, which is the upper bound of $\frac{1}{P}\sum_{p=1}^{P}\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)-\nabla\mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}}^t)\|^2$ and is caused by ignoring the cut-edges for local gradient computation;
- The σ_{bias}^2 term, which is the upper bound of $\frac{1}{P}\sum_{p=1}^{P}\left\|\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}}^t;\xi_p^t)]-\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t)\right\|^2$ and is caused by using neighbor sampling;
- The model divergence term $\frac{1}{P}\sum_{p=1}^{P}\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2$ which is caused by the difference between the model parameters and the virtual average model due to periodic averaging. Notice that the model divergence term also exists in distributed learning regardless of whether the dataset is graph or not and caused by infrequent synchronization.

Now, let's have a closer look at the model divergence term. As shown in Lemma 3 the model divergence term $\frac{1}{P}\sum_{p=1}^{P}\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2$ is further caused by three factors:

- The σ_{var}^2 term (in Eq. 64), which is the upper bound of mini-batch sampling variance $\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\Big[\Big\|\sum_{\tau=t_0}^{t-1}\Big(\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau)]-\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)\Big)\Big\|^2\Big]$

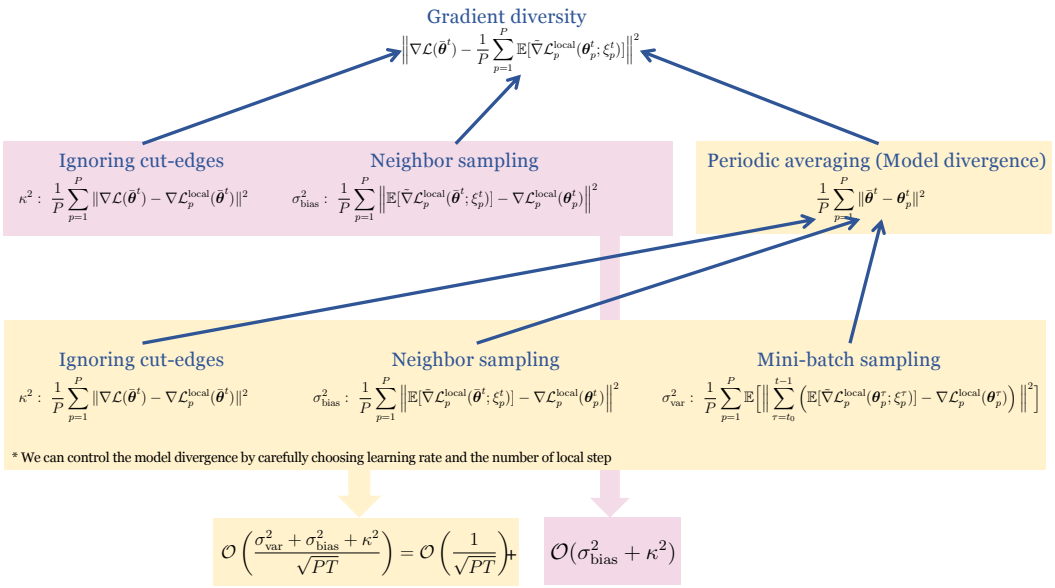


Figure 12: An overview on the existence of irreducible error from the theoretical point of view. We use A $\rightarrow$ B to denote A is causing B.

- The κ^2 term, which is the upper bound of $\frac{1}{P}\sum_{p=1}^{P}\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t) - \nabla\mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}}^t)\|^2$ and is caused by ignoring the cut-edges for local gradient computation;
- The σ_{bias}^2 term, which is the upper bound of $\frac{1}{P}\sum_{p=1}^{P}\left\|\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}}^t;\xi_p^t)] - \nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t)\right\|^2$ and is caused by using neighbor sampling.

Fortunately, the model divergence term can be controlled by the number of local gradient update steps and learning rate, which is reducing with respect to the number of total training steps T and the number of local machine P, and it leads to the first term in our upper bound in Theorem 1, i.e., $\mathcal{O}(\frac{\sigma_{\text{var}}^2+\sigma_{\text{bias}}^2+\kappa^2}{\sqrt{PT}}) = \mathcal{O}(\frac{1}{\sqrt{PT}})$. However, unfortunately, κ^2 and σ_{bias}^2 in gradient diversity are isolated from the model diversity part, therefore are irreducible and results in $\mathcal{O}(\sigma_{\text{bias}}^2 + \kappa^2)$ in the second term in our upper bound in Theorem 1. Please refer to Figure 12 for an illustration.

Intuitively, these theoretical results make sense. During training, we are minimizing the loss without cut-edges $\frac{1}{P}\sum_{p=1}^{P}\mathcal{L}^{\text{local}}(\boldsymbol{\theta})$, which is a different objective to the loss defined on the full-graph by taking the cut-edges into consideration $\mathcal{L}(\boldsymbol{\theta})$. Therefore, solely by adding the number of training iterations, we cannot guarantee a small gradient of $\mathcal{L}(\boldsymbol{\theta})$ by minimizing $\frac{1}{P}\sum_{p=1}^{P}\mathcal{L}^{\text{local}}(\boldsymbol{\theta})$.

B.4 PROOF OF LEMMA 1

By the definition of $\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p;\xi_p)$, we have

$$
\begin{aligned}
&\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p;\xi_p)\Big\|^2\Big] \\
&= \mathbb{E}\left[\Big\|\frac{1}{P}\sum_{p=1}^{P}\Big(\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p;\xi_p)-\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p,\xi_p^t)\right]\Big)\Big\|^2\right]+\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p,\xi_p^t)\right]\Big\|^2\Big] \\
&\underset{(a)}{\leq} \frac{1}{P^2}\sum_{p=1}^{P}\mathbb{E}\left[\Big\|\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p;\xi_p)-\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p;\xi_p)\right]\Big\|^2\right]+\mathbb{E}\left[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p,\xi_p^t)\right]\Big\|^2\right] \\
&\underset{(b)}{\leq} \frac{\sigma_{\text{var}}^2}{P}+\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\left[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p,\xi_p^t)\right]\Big\|^2\Big]
\end{aligned}
\tag{29}
$$

where (a) is due to fact that each $\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p;\xi_p)-\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p;\xi_p)]$ is independent random vectors with zero mean and (b) is from Assumption 1.

B.5 PROOF OF LEMMA 2

By the definition of local gradient $\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta})$, we have

$$
\begin{aligned}
&\frac{1}{P}\sum_{p=1}^{P}\Big\|\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p)-\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q)\Big\|^2 \\
&= \frac{1}{P}\sum_{p=1}^{P}\Big\|\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p)-\nabla\mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}})+\nabla\mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}})-\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}) \\
&\quad+\nabla\mathcal{L}(\bar{\boldsymbol{\theta}})-\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}(\boldsymbol{\theta}_q)+\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}(\boldsymbol{\theta}_q)-\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q)\Big\|^2 \\
&\underset{(a)}{\leq} \frac{1}{P}\sum_{p=1}^{P}4\|\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p)-\nabla\mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}})\|^2+\frac{1}{P}\sum_{p=1}^{P}4\|\nabla\mathcal{L}_p^{\text{local}}(\bar{\boldsymbol{\theta}})-\nabla\mathcal{L}(\bar{\boldsymbol{\theta}})\|^2 \\
&\quad+\frac{1}{P}\sum_{p=1}^{P}4\Big\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}})-\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}(\boldsymbol{\theta}_q)\Big\|^2+\frac{1}{P}\sum_{p=1}^{P}4\Big\|\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}(\boldsymbol{\theta}_q)-\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q)\Big\|^2 \\
&\underset{(b)}{\leq} \frac{8L}{P}\sum_{p=1}^{P}\|\boldsymbol{\theta}_p-\bar{\boldsymbol{\theta}}\|^2+8\kappa^2,
\end{aligned}
\tag{30}
$$

where (a) is due to $\|\sum_{i=1}^{n}\mathbf{x}_i\|^2\leq\sum_{i=1}^{n}n\|\mathbf{x}_i\|$ and (b) is due to the definition of κ.

B.6 PROOF OF LEMMA 3

When $(t \bmod K)=0$ we have $\bar{\boldsymbol{\theta}}^t=\boldsymbol{\theta}_p^t$. When $(t \bmod K)\neq 0$ and $t\geq 1$, let $t_0<t$ be the largest iteration index that $t_0 \bmod K=0$. For any $\tau\in\{t_0+1,\ldots,t\}$, we have

$$
\boldsymbol{\theta}_p^{\tau}-\boldsymbol{\theta}_p^{\tau-1}=-\eta\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^{\tau-1};\xi_p^{\tau-1}). \tag{31}
$$

Summing over $\tau \in \{t_0+1, \dots, t\}$, we have

$$\begin{aligned}\boldsymbol{\theta}_p^t &= \boldsymbol{\theta}_p^{t_0} - \eta \sum_{\tau=t_0}^{t-1} \tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau; \xi_p^\tau) \\ &\underset{(a)}{=} \bar{\boldsymbol{\theta}}^{t_0} - \eta \sum_{\tau=t_0}^{t-1} \tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau; \xi_p^\tau),\end{aligned} \tag{32}$$

where (a) is due to $\bar{\boldsymbol{\theta}}^t = \boldsymbol{\theta}_p^t$ when $(t \bmod K) = 0$.

Similarly, we have

$$\bar{\boldsymbol{\theta}}^t = \bar{\boldsymbol{\theta}}^{t_0} - \eta \sum_{\tau=t_0}^{t-1} \frac{1}{P} \sum_{p=1}^{P} \tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau; \xi_p^\tau). \tag{33}$$

Combining Eq. 32 and Eq. 33, we have

$$\begin{aligned}&\frac{1}{P} \sum_{p=1}^{P} \mathbb{E}[\|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2] \\ &= \frac{\eta^2}{P} \sum_{p=1}^{P} \mathbb{E}\Big[\Big\| \sum_{\tau=t_0}^{t-1} \Big(\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau; \xi_p^\tau) - \frac{1}{P} \sum_{q=1}^{P} \tilde{\nabla} \mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau; \xi_q^\tau) \Big) \Big\|^2 \Big] \\ &\underset{(a)}{\leq} 2\eta^2 \underbrace{\frac{1}{P} \sum_{p=1}^{P} \mathbb{E}\Big[\Big\| \sum_{\tau=t_0}^{t-1} \Big(\big(\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau; \xi_p^\tau) - \nabla \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)\big) - \frac{1}{P} \sum_{q=1}^{P} \big(\tilde{\nabla} \mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau; \xi_q^\tau) - \nabla \mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau)\big) \Big) \Big\|^2 \Big]}_{(A)} \\ &\quad + 2\eta^2 \underbrace{\frac{1}{P} \sum_{p=1}^{P} \mathbb{E}\Big[\Big\| \sum_{\tau=t_0}^{t-1} \Big(\nabla \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau) - \frac{1}{P} \sum_{q=1}^{P} \nabla \mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau) \Big) \Big\|^2 \Big]}_{(B)},\end{aligned} \tag{34}$$

where (a) is due to addition and subtraction of $\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau; \xi_p^\tau)$ and $\nabla \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)$ and using $\|\mathbf{x}+\mathbf{y}\|^2 \leq \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2$.

We can upper bound (A) in Eq. 63 by

$$\begin{aligned}(A) &\underset{(a)}{\leq} \frac{1}{P} \sum_{p=1}^{P} \mathbb{E}\Big[\Big\| \sum_{\tau=t_0}^{t-1} \big(\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau; \xi_p^\tau) - \nabla \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)\big) \Big\|^2 \Big] \\ &\leq \frac{2}{P} \sum_{p=1}^{P} \mathbb{E}\Big[\Big\| \sum_{\tau=t_0}^{t-1} \Big(\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau; \xi_p^\tau) - \mathbb{E}[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau; \xi_p^\tau)] \Big) \Big\|^2 \Big] \\ &\quad + \frac{2}{P} \sum_{p=1}^{P} \mathbb{E}\Big[\Big\| \sum_{\tau=t_0}^{t-1} \Big(\mathbb{E}[\tilde{\nabla} \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau; \xi_p^\tau)] - \nabla \mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau) \Big) \Big\|^2 \Big] \\ &\leq 2K\sigma_{\text{var}}^2 + 2K^2\sigma_{\text{bias}}^2,\end{aligned} \tag{35}$$

where (a) is due to

$$\frac{1}{n} \sum_{i=1}^{n} \Big\| \mathbf{x}_i - \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j \Big\|^2 = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i\|^2 - \Big\| \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \Big\|^2 \leq \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i\|^2. \tag{36}$$

We can further bound (B) in Eq. 63 by

$$
\begin{aligned}
(B) &\leq \frac{1}{P}\sum_{p=1}^{P}(t-t_0)\sum_{\tau=t_0}^{t-1}\mathbb{E}\Big[\Big\|\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)-\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau)\Big\|^2\Big] \\
&\leq K\sum_{\tau=t_0}^{t-1}\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\Big[\Big\|\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)-\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau)\Big\|^2\Big] \\
&\underset{(a)}{\leq} K\sum_{\tau=t_0}^{t-1}\Big(\frac{8L^2}{P}\sum_{i=1}^{P}\|\boldsymbol{\theta}_p^\tau-\bar{\boldsymbol{\theta}}^\tau\|^2+8\kappa^2\Big) \\
&\leq \frac{8L^2K}{P}\sum_{\tau=t_0}^{t-1}\sum_{i=1}^{P}\|\boldsymbol{\theta}_p^\tau-\bar{\boldsymbol{\theta}}^\tau\|^2+8K^2\kappa^2,
\end{aligned}
\tag{37}
$$

where (a) is due to Lemma 2 and the definition of κ.

By plugging Eq. 64 and Eq. 66 into Eq. 63, we have

$$
\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2]\leq 4\eta^2K\sigma_{\text{var}}^2+4\eta^2K^2\sigma_{\text{bias}}^2+16\eta^2L^2K\sum_{\tau=t_0}^{t-1}\frac{1}{P}\sum_{i=1}^{P}\|\boldsymbol{\theta}_p^\tau-\bar{\boldsymbol{\theta}}^\tau\|^2+16\eta^2K^2\kappa^2. \tag{38}
$$

By summing over $t\in\{0,\dots,T\}$, we have

$$
\begin{aligned}
\underbrace{\sum_{t=0}^{T-1}\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2]}_{(A)} &\leq 4\eta^2K\sigma_{\text{var}}^2T+4\eta^2K^2\sigma_{\text{bias}}^2T \\
&\quad+16\eta^2L^2K^2\underbrace{\sum_{t=0}^{T-1}\frac{1}{P}\sum_{i=1}^{P}\|\boldsymbol{\theta}_p^t-\bar{\boldsymbol{\theta}}^t\|^2}_{(A)}+16\eta^2K^2\kappa^2T.
\end{aligned}
\tag{39}
$$

Collecting common terms and dividing both sides by $(1-16\eta^2L^2K^2)$ gives

$$
(A)\leq\frac{4\eta^2K\sigma^2T}{(1-16\eta^2L^2K^2)}+\frac{4\eta^2K^2\sigma_{\text{bias}}^2T+16\eta^2K^2\kappa^2T}{(1-16\eta^2L^2K^2)}. \tag{40}
$$

C PROOF OF THEOREM 2

In the following, we first introduce the useful lemmas in Section C.1, then process our proof of theorem in Section C.2. In particular, we show that this residual error (which is cause by ignoring the cut-edges) can eliminated by running server correction steps after the parameter averaging on the server.

C.1 USEFUL LEMMA

The following lemma provides an upper bound on the deviation of each local model to virtual averaged model, which is important to upper bound on the difference of each local gradient $\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p)$ compared to $\frac{1}{P}\sum_{p=1}^{P}\nabla\mathcal{L}_p(\boldsymbol{\theta})$.

Lemma 4 *For all P machines with $\boldsymbol{\theta}_p,\ p\in\{1,\dots,P\}$, if we define $\bar{\boldsymbol{\theta}}=\frac{1}{P}\sum_{p=1}^{P}\boldsymbol{\theta}_p$ and let $\kappa>0$ such that $\|\mathcal{L}_p^{local}(\boldsymbol{\theta})-\nabla\mathcal{L}(\boldsymbol{\theta})\|^2\leq\kappa^2$, we have*

$$\sum_{t\in\mathcal{T}_{local}(r)}\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2]\leq\frac{4\eta^2(\sigma_{bias}^2+\sigma_{var}^2)K^2\rho^{2r}+16\eta^2\kappa^2K^2\rho^{2r}}{1-16\eta^2L^2K^2\rho^{2r}}. \tag{41}$$

C.2 MAIN PROOF OF THEOREM 2

Equipped with the above lemma and results from Appendix B.1 , we are now ready to present the proof of Theorem 2.

Let $\mathcal{T}_{\text{local}}(r)$ and $\mathcal{T}_{\text{server}}(r)$ as the iteration indicies that a local machine and server run on, which is defined as

$$\begin{aligned}\mathcal{T}_{\text{local}}(r)&=\left\{k+\left(\sum_{j=1}^{r-1}K\rho^j\right)+S(r-1):k=1,\dots,K\rho^r\right\}\\ \mathcal{T}_{\text{server}}(r)&=\left\{s+\left(\sum_{j=1}^{r}K\rho^j\right)+S(r-1):s=1,\dots,S\right\},\end{aligned} \tag{42}$$

and let define $\mathcal{T}_{\text{local}}=\mathcal{T}_{\text{local}}(1)\cup\ldots\cup\mathcal{T}_{\text{local}}(R)$ and $\mathcal{T}_{\text{global}}=\mathcal{T}_{\text{global}}(1)\cup\ldots\cup\mathcal{T}_{\text{global}}(R)$.

By the smoothness assumption, we have

$$\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\leq\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)]+\mathbb{E}[\langle\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t),\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\rangle]+\frac{L}{2}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\|^2]. \tag{43}$$

Let first consider $t\in\mathcal{T}_{\text{server}}(r)$, with the following update

$$\begin{aligned}\bar{\boldsymbol{\theta}}^{t+1}&=\bar{\boldsymbol{\theta}}^t-\gamma\tilde{\nabla}\mathcal{L}(\bar{\boldsymbol{\theta}}^t;\xi^t)\\ &=\bar{\boldsymbol{\theta}}^t-\frac{\gamma}{P}\sum_{p=1}^{P}\tilde{\nabla}\mathcal{L}_p^{\text{full}}(\bar{\boldsymbol{\theta}}^t;\xi_p^t).\end{aligned} \tag{44}$$

Therefore, we have

$$\begin{aligned}\mathbb{E}[\langle\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t),\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\rangle]&=-\gamma\mathbb{E}\left[\left\langle\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t),\frac{1}{P}\sum_{p=1}^{P}\tilde{\nabla}\mathcal{L}_p^{\text{full}}(\bar{\boldsymbol{\theta}}^t;\xi_p^t)\right\rangle\right]\\ &\underset{(a)}{=}-\gamma\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2],\end{aligned} \tag{45}$$

where the equality is due to $\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{\text{full}}(\bar{\boldsymbol{\theta}}^t;\xi_p^t)]=\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)$ since all neighbors are used for the server correction steps.

Besides, by taking the norm on the both side of Eq. 44, we have the following equality

$$\begin{aligned}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}\|^2] &= \gamma^2\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\tilde{\nabla}\mathcal{L}_p^{\text{full}}(\bar{\boldsymbol{\theta}}^t;\xi_p^t)\Big\|^2\Big] \\ &\underset{(a)}{\leq} \gamma^2\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\nabla\mathcal{L}_p^{\text{full}}(\bar{\boldsymbol{\theta}}^t)\Big\|^2\Big] + \gamma^2\frac{\sigma_{\text{global}}^2}{P} \\ &= \gamma^2\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] + \gamma^2\frac{\sigma_{\text{global}}^2}{P}.\end{aligned} \tag{46}$$

where inequality (b) hold due to Lemma 1. Notice that the result in Lemma 1 holds for both the "local" and the "global" setting since $\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{\text{full}}(\bar{\boldsymbol{\theta}}^t;\xi_p^t)] = \nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)$.

Substituting Eq. 45, 46 into Eq. 43, we know that for $t \in \mathcal{T}_{\text{server}}(r)$, we have

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})] &\leq \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)] + \Big(\frac{\gamma^2 L}{2}-\gamma\Big)\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] + \frac{\gamma^2 L}{2}\frac{\sigma_{\text{global}}^2}{P} \\ &= \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)] + \frac{\gamma}{2}\Big(\gamma L-1\Big)\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] + \frac{\gamma^2 L}{2}\frac{\sigma_{\text{global}}^2}{P} - \frac{\gamma}{2}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2].\end{aligned} \tag{47}$$

Dividing both sides by $\frac{\gamma}{2}$ and rearranging terms yields

$$\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] \leq \frac{2}{\gamma}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)] - \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big) + (\gamma L-1)\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] + \gamma L\frac{\sigma_{\text{global}}^2}{P}. \tag{48}$$

Then, let consider the local update steps where $t \in \mathcal{T}_{\text{local}}(r)$. According to Eq. 25 in proof of Theorem 1, we have

$$\begin{aligned}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] \leq \frac{2}{\eta}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)] - \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big) + (\eta L-1)\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t)\Big\|^2\Big] \\ + 2(\kappa^2+2\sigma_{\text{bias}}^2) + \frac{4L^2}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2] + \frac{\eta L\sigma_{\text{var}}^2}{P}.\end{aligned} \tag{49}$$

Let $T = \Big(\sum_{r=1}^{R}K\rho^r\Big) + SR$ and summing over $t \in \{1,\dots,T\}$, combining Eq. 48 and Eq. 49 we have

$$\begin{aligned}&\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] \\ &= \sum_{r=1}^{R}\sum_{t\in\mathcal{T}_{\text{local}}(r)}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] + \sum_{r=1}^{R}\sum_{t\in\mathcal{T}_{\text{global}}(r)}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] \\ &\leq \sum_{r=1}^{R}\sum_{t\in\mathcal{T}_{\text{global}}(r)}\Big[\frac{2}{\gamma}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)] - \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big) + (\gamma L-1)\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] + \gamma L\frac{\sigma_{\text{global}}^2}{P}\Big] \\ &\quad + \sum_{r=1}^{R}\sum_{t\in\mathcal{T}_{\text{local}}(r)}\Big[\frac{2}{\eta}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)] - \mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big) + (\eta L-1)\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t)\Big\|^2\Big] \\ &\quad + 2(\kappa^2+\sigma_{\text{bias}}^2) + \frac{4L^2}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2] + \frac{\eta L\sigma_{\text{var}}^2}{P}\Big].\end{aligned} \tag{50}$$

Rearrange the above equation, we have

$$
\begin{aligned}
&\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] \\
&\leq \sum_{r=1}^{R}\left[\sum_{t\in\mathcal{T}_{\text{global}}(r)}\frac{2}{\gamma}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)]-\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big)+\sum_{t\in\mathcal{T}_{\text{local}}(r)}\frac{2}{\eta}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)]-\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big)\right] \\
&\quad+\sum_{r=1}^{R}\left[\sum_{t\in\mathcal{T}_{\text{global}}(r)}(\gamma L-1)\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2]+\sum_{t\in\mathcal{T}_{\text{local}}(r)}(\eta L-1)\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t)\Big\|^2\right] \\
&\quad+\sum_{r=1}^{R}\left[\sum_{t\in\mathcal{T}_{\text{global}}(r)}\gamma L\frac{\sigma_{\text{global}}^2}{P}+\sum_{t\in\mathcal{T}_{\text{local}}(r)}\frac{\eta L\sigma_{\text{var}}^2}{P}\right]+\left(\sum_{r=1}^{R}K\rho^r\right)(\kappa^2+2\sigma_{\text{var}}^2) \\
&\quad+\sum_{r=1}^{R}\sum_{t\in\mathcal{T}_{\text{local}}(r)}\left(\frac{4L^2}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2]\right).
\end{aligned}
\tag{51}
$$

Let define

$$
G_{\text{global}}^r=\min_{t\in\mathcal{T}_{\text{global}}(r)}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2],\ G_{\text{local}}^r=\min_{t\in\mathcal{T}_{\text{local}}(r)}\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t)\Big\|^2. \tag{52}
$$

Our goal is to select the size of $\mathcal{T}_{\text{local}}(r)$ and $\mathcal{T}_{\text{global}}(r)$ such that the following inequality holds

$$
\begin{aligned}
(K\rho^r)\,(\kappa^2+2\sigma_{\text{bias}}^2)&\leq(1-\gamma L)\sum_{t\in\mathcal{T}_{\text{global}}(r)}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2]+(1-\eta L)\sum_{t\in\mathcal{T}_{\text{local}}(r)}\mathbb{E}\Big[\Big\|\frac{1}{P}\sum_{p=1}^{P}\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^t)\Big\|^2 \\
&\leq(1-\gamma L)SG_{\text{global}}^r+(1-\eta L)K\rho^rG_{\text{local}}^r
\end{aligned}
\tag{53}
$$

After rearranging it, we will have

$$
S\geq\frac{K\rho^r}{G_{\text{global}}^r(1-\gamma L)}\left(\kappa^2+2\sigma_{\text{bias}}^2-(1-\eta L)G_{\text{local}}^r\right). \tag{54}
$$

Suppose Eq. 54 holds, we have

$$
\begin{aligned}
&\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] \\
&\leq\sum_{r=1}^{R}\left[\sum_{t\in\mathcal{T}_{\text{global}}(r)}\frac{2}{\gamma}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)]-\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big)+\sum_{t\in\mathcal{T}_{\text{local}}(r)}\frac{2}{\eta}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)]-\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big)\right] \\
&\quad+\sum_{r=1}^{R}\left[\sum_{t\in\mathcal{T}_{\text{global}}(r)}\gamma L\frac{\sigma_{\text{global}}^2}{P}+\sum_{t\in\mathcal{T}_{\text{local}}(r)}\frac{\eta L\sigma_{\text{var}}^2}{P}\right]+\sum_{r=1}^{R}\sum_{t\in\mathcal{T}_{\text{local}}(r)}\left(\frac{4L^2}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2]\right) \\
&\underset{(a)}{\leq}\sum_{r=1}^{R}\left[\sum_{t\in\mathcal{T}_{\text{global}}(r)}\frac{2}{\gamma}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)]-\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big)+\sum_{t\in\mathcal{T}_{\text{local}}(r)}\frac{2}{\eta}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)]-\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big)\right] \\
&\quad+\sum_{r=1}^{R}\left[\sum_{t\in\mathcal{T}_{\text{global}}(r)}\gamma L\frac{\sigma_{\text{global}}^2}{P}+\sum_{t\in\mathcal{T}_{\text{local}}(r)}\frac{\eta L\sigma_{\text{var}}^2}{P}\right] \\
&\quad+4L^2\sum_{r=1}^{R}\left(\frac{4\eta^2(\sigma_{\text{bias}}^2+\sigma_{\text{var}}^2)K^2\rho^{2r}}{1-16\eta^2L^2K^2\rho^{2r}}+\frac{16\eta^2\kappa^2K^2\rho^{2r}}{1-16\eta^2L^2K^2\rho^{2r}}\right),
\end{aligned}
\tag{55}
$$

where (a) is due to Lemma 4.

By selecting $K\rho^r$ such that $1-16\eta^2L^2K^2\rho^{2r}\geq\frac{1}{2}$, we have $K\rho^r\leq\frac{\sqrt{2}}{8L\eta}$ and

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2]\\
&=\frac{1}{T}\sum_{r=1}^{R}\left[\sum_{t\in\mathcal{T}_{\text{global}}(r)}\frac{2}{\gamma}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)]-\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big)+\sum_{t\in\mathcal{T}_{\text{global}}(r)}\gamma L\frac{\sigma_{\text{global}}^2}{P}\right]\\
&\quad+\frac{1}{T}\sum_{r=1}^{R}\left[\sum_{t\in\mathcal{T}_{\text{local}}(r)}\frac{2}{\eta}\Big(\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^t)]-\mathbb{E}[\mathcal{L}(\bar{\boldsymbol{\theta}}^{t+1})]\Big)+\sum_{t\in\mathcal{T}_{\text{local}}(r)}\frac{\eta L\sigma_{\text{var}}^2}{P}\right]\\
&\quad+\frac{4\eta^2L^2\left(8(\sigma_{\text{var}}^2+\sigma_{\text{bias}}^2)+32\kappa^2\right)}{T}\sum_{r=1}^{R}K^2\rho^{2r}.
\end{aligned}\tag{56}$$

Notice that the condition $K\rho^r\leq\frac{\sqrt{2}}{8L\eta}$ implies

$$\sum_{r=1}^{R}K^2\rho^{2r}=K^2\frac{1-\rho^{2r}}{1-\rho^2}\leq\frac{R}{32L^2\eta^2}.\tag{57}$$

To this end, by selecting $\eta=\gamma=\frac{\sqrt{P}}{\sqrt{T}}$, $\sum_{r=1}^{R}K^2\rho^{2r}\leq\frac{RT^{1/2}}{32L^2P^{3/2}}$, and $\sigma^2=\max\{\sigma_{\text{var}}^2,\sigma_{\text{global}}^2\}$, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2]\leq\frac{2}{\sqrt{PT}}\Big(\mathcal{L}(\bar{\boldsymbol{\theta}}^0)-\mathcal{L}(\bar{\boldsymbol{\theta}}^\star)\Big)+\frac{L\sigma^2}{\sqrt{PT}}+\frac{4L^2}{\sqrt{PT}}\sum_{r=1}^{R}(8(\sigma_{\text{bias}}^2+\sigma_{\text{var}}^2)+32\kappa^2),\tag{58}$$

and

$$S\geq\left(\frac{\kappa^2+2\sigma_{\text{bias}}^2}{1-L(\sqrt{P/T})}-G_{\text{local}}^r\right)\frac{K\rho^r}{G_{\text{global}}^r}.\tag{59}$$

C.3 Proof of Lemma 4

Recall that when $t\in\mathcal{T}_{\text{global}}(r)$ we have $\bar{\boldsymbol{\theta}}^t=\boldsymbol{\theta}_p^t$. For any $t\in\mathcal{T}_{\text{local}}(r)$, let t_0+1 denote as the first indices of $\mathcal{T}_{\text{local}}(r)$ and $|\mathcal{T}_{\text{local}}(r)|=K\rho^r$. Then, for any $\tau\in\{t_0+1,\ldots,t\}$, we have

$$\boldsymbol{\theta}_p^\tau-\boldsymbol{\theta}_p^{\tau-1}=-\eta\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^{\tau-1};\xi_p^{\tau-1})\tag{60}$$

Summing over $\tau\in\{t_0+1,\ldots,t\}$, we have

$$\begin{aligned}
\boldsymbol{\theta}_p^t&=\boldsymbol{\theta}_p^{t_0}-\eta\sum_{\tau=t_0}^{t-1}\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau)\\
&\underset{(a)}{=}\bar{\boldsymbol{\theta}}^{t_0}-\eta\sum_{\tau=t_0}^{t-1}\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau),
\end{aligned}\tag{61}$$

where (a) is due to $\bar{\boldsymbol{\theta}}^t=\boldsymbol{\theta}_p^t$ when $t\in\mathcal{T}_{\text{global}}(r)$.

Similarly, we have

$$\bar{\boldsymbol{\theta}}^t=\bar{\boldsymbol{\theta}}^{t_0}-\eta\sum_{\tau=t_0}^{t-1}\frac{1}{P}\sum_{p=1}^{P}\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau).\tag{62}$$

By combining Eq. 61 and Eq. 62, we have

$$
\begin{aligned}
&\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2] \\
&=\frac{\eta^2}{P}\sum_{p=1}^{P}\mathbb{E}\Big[\Big\|\sum_{\tau=t_0}^{t-1}\Big(\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau)-\frac{1}{P}\sum_{q=1}^{P}\tilde{\nabla}\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau;\xi_q^\tau)\Big)\Big\|^2\Big] \\
&\underset{(a)}{\leq} 2\eta^2\underbrace{\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\Big[\Big\|\sum_{\tau=t_0}^{t-1}\Big(\big(\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau)-\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)\big)-\frac{1}{P}\sum_{q=1}^{P}\big(\tilde{\nabla}\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau;\xi_q^\tau)-\nabla\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau)\big)\Big)\Big\|^2\Big]}_{(A)} \\
&\quad+2\eta^2\underbrace{\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\Big[\Big\|\sum_{\tau=t_0}^{t-1}\Big(\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)-\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau)\Big)\Big\|^2\Big]}_{(B)},
\end{aligned}
\tag{63}
$$

where (a) follows by adding and substracting $\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau),\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)$ and using $\|\mathbf{x}+\mathbf{y}\|^2\leq\|\mathbf{x}\|^2+\|\mathbf{y}\|^2$.

We can upper bound (A) in Eq. 63 by

$$
\begin{aligned}
(A)&\underset{(a)}{\leq}\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\Big[\Big\|\sum_{\tau=t_0}^{t-1}\big(\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau)-\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)\big)\Big\|^2\Big] \\
&\leq\frac{2}{P}\sum_{p=1}^{P}\sum_{\tau=t_0}^{t-1}\mathbb{E}\Big[\Big\|\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau)-\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau)]\Big\|^2\Big] \\
&\quad+\frac{2}{P}\sum_{p=1}^{P}\sum_{\tau=t_0}^{t-1}\mathbb{E}\Big[\Big\|\mathbb{E}[\tilde{\nabla}\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau;\xi_p^\tau)]-\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)\Big\|^2\Big] \\
&\leq 2K\rho^r(\sigma_{\text{var}}^2+\sigma_{\text{bias}}^2),
\end{aligned}
\tag{64}
$$

where (a) is due to

$$
\frac{1}{n}\sum_{i=1}^{n}\Big\|\mathbf{x}_i-\frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j\Big\|^2=\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i\|^2-\Big\|\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\Big\|^2\leq\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i\|^2. \tag{65}
$$

We can further bound (B) in Eq. 63 by

$$
\begin{aligned}
(B)&\leq\frac{1}{P}\sum_{p=1}^{P}(t-t_0)\sum_{\tau=t_0}^{t-1}\mathbb{E}\Big[\Big\|\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)-\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau)\Big\|^2\Big] \\
&\leq K\rho^r\sum_{\tau=t_0}^{t-1}\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\Big[\Big\|\nabla\mathcal{L}_p^{\text{local}}(\boldsymbol{\theta}_p^\tau)-\frac{1}{P}\sum_{q=1}^{P}\nabla\mathcal{L}_q^{\text{local}}(\boldsymbol{\theta}_q^\tau)\Big\|^2\Big] \\
&\underset{(a)}{\leq}K\rho^r\sum_{\tau=t_0}^{t-1}\Big(\frac{8L^2}{P}\sum_{p=1}^{P}\|\boldsymbol{\theta}_p^\tau-\bar{\boldsymbol{\theta}}^\tau\|^2+8\kappa^2\Big) \\
&\leq\frac{8L^2K\rho^r}{P}\sum_{\tau=t_0}^{t-1}\sum_{p=1}^{P}\|\boldsymbol{\theta}_p^\tau-\bar{\boldsymbol{\theta}}^\tau\|^2+8K^2\rho^{2r}\kappa^2,
\end{aligned}
\tag{66}
$$

where (a) is due to Lemma 2 and the definition of κ.

By plugging Eq. 64 and Eq. 66 into Eq. 63, we have

$$
\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}[\|\bar{\boldsymbol{\theta}}^t-\boldsymbol{\theta}_p^t\|^2]\leq 4\eta^2K\rho^r(\sigma_{\text{var}}^2+\sigma_{\text{bias}}^2)+16\eta^2L^2K\rho^r\sum_{\tau=t_0}^{t-1}\frac{1}{P}\sum_{p=1}^{P}\|\boldsymbol{\theta}_p^\tau-\bar{\boldsymbol{\theta}}^\tau\|^2+16\eta^2K^2\rho^{2r}\kappa^2. \tag{67}
$$

Let us define $\mathcal{T}_{\text{local}} = \mathcal{T}_{local}(1) \cup \ldots \cup \mathcal{T}_{local}(R)$. By summing over $t \in \mathcal{T}_{\text{local}}$, we have

$$\begin{aligned}
&\underbrace{\sum_{r=1}^{R} \sum_{t \in \mathcal{T}_{\text{local}}(r)} \frac{1}{P} \sum_{p=1}^{P} \mathbb{E}[\|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2]}_{(A)} \\
&\leq \sum_{r=1}^{R} \left(4\eta^2(\sigma_{\text{bias}}^2 + \sigma_{\text{var}}^2)K^2\rho^{2r} + 16\eta^2\kappa^2K^2\rho^{2r}\right) \\
&\quad + \sum_{r=1}^{R} 16\eta^2L^2K^2\rho^{2r} \underbrace{\sum_{t \in \mathcal{T}_{\text{local}}(r)} \frac{1}{P} \sum_{p=1}^{P} \|\boldsymbol{\theta}_p^t - \bar{\boldsymbol{\theta}}^t\|^2}_{(A)} .
\end{aligned} \tag{68}$$

Rearranging the terms gives

$$\sum_{r=1}^{R}(1-16\eta^2L^2K^2\rho^{2r}) \underbrace{\sum_{t \in \mathcal{T}_{\text{local}}(r)} \frac{1}{P} \sum_{p=1}^{P} \mathbb{E}[\|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2]}_{(A)} \leq \sum_{r=1}^{R} \left(4\eta^2((\sigma_{\text{bias}}^2 + \sigma_{\text{var}}^2))K^2\rho^{2r} + 16\eta^2\kappa^2K^2\rho^{2r}\right) . \tag{69}$$

Therefore, we conclude that

$$\sum_{t \in \mathcal{T}_{\text{local}}(r)} \frac{1}{P} \sum_{p=1}^{P} \mathbb{E}[\|\bar{\boldsymbol{\theta}}^t - \boldsymbol{\theta}_p^t\|^2] \leq \frac{4\eta^2((\sigma_{\text{bias}}^2 + \sigma_{\text{var}}^2))K^2\rho^{2r} + 16\eta^2\kappa^2K^2\rho^{2r}}{1 - 16\eta^2L^2K^2\rho^{2r}}. \tag{70}$$

which completes the proof.