GRADIENT-WISE OPTIMIZATION FOR DISTRIBUTED MACHINE LEARNING

FEIJIE WU

MPhil

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University

Department of Computing

Gradient-wise Optimization for Distributed Machine Learning

Feijie Wu

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Philosophy

June 2022

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

______________________ (Signed)

_______Feijie Wu_______ (Name of student)

Abstract

Distributed machine learning has intrigued a booming interest and achieved rapid development over the past decades. It allows multiple nodes with different data sources to collaboratively train a model using their local computational resources, which achieves linear speedup with respect to the number of nodes. However, the distributed manner mainly has threefold challenges. First, full-precision synchronizations occupy significant communication bandwidth. In particular, traditional algorithms require global synchronization at every iteration, which consumes considerable communication overhead and leads to a critical slowdown in terms of training time. Second, the computational capabilities vary among nodes, resulting in resource underutilization because all nodes should wait for the slowest one. Third, a conventional assumption on the data distribution is independent and identical among nodes. However, in reality, the data are heterogeneous because there is no intersection between any two clients when data sharing is not permitted.

To avoid the overwhelming communication consumption, a common practice is to adopt a gradient compression approach, e.g., one-bit compressed stochastic gradient descent (signSGD). Traditional signSGD has made a great success in a star topology. However, due to cascading compression, it can not be directly employed in multi-hop all-reduce (MAR), a synchronization paradigm that has been widely adopted in network-intensive high-performance computing systems like public clouds. To support signSGD implementation under MAR, we propose a learning synchronization system, Marsit. It prevents cascading compression by employing a bit-wise operation for unbiased sign aggregation and a unique global compensation approach to accommodate the compression deviation.

Another solution to reducing the communication overhead is to allow nodes to perform multiple but inconsistent local updates, which simultaneously settle computational heterogeneity. However, the strategy possibly leads to object inconsistency when data heterogeneity exists, which undermines the model performance. Consequently, we design a gradient calibration approach, FedaGrac, which calibrates the local direction to a predictive global orientation. It is guaranteed that the aggregated model does not vary substantially from the global optimum while fully utilizing the local updates of faster nodes by using the estimated orientation.

In a nutshell, we utilize the gradient-wise approaches to optimize the training efficiency in distributed machine learning. Theoretical results reveal our gradient compression framework retains the same convergence rate as non-compression mechanisms, while the gradient calibration algorithm holds an improved order of convergence rate than the state-of-the-art approaches. Extensive experiments have demonstrated the superiority of our proposed methods.

Publications Arising from the Thesis

[1] **Feijie Wu***, Shiqi He*, Song Guo, Zhihao Qu, Haozhao Wang, Weihua Zhuang, Jie Zhang, “Sign Bit is Enough: A Learning Synchronization Framework for Multi-hop All-reduce with Ultimate Compression,” in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC’22),* pp. 193–198, 2022.

[2] **Feijie Wu**, Song Guo, Haozhao Wang, Haobo Zhang, Zhihao Qu, Jie Zhang, Ziming Liu, “From Deterioration to Acceleration: A Calibration Approach to Rehabilitating Step Asynchronism in Federated Optimization,” *IEEE Transactions on Parallel and Distributed Systems,* vol. 34, no. 05, pp. 1548–1559, 2023.

List of Publications other than above

[3] Tao Guo, Song Guo, **Feijie Wu**, Wenchao Xu, Jiewei Zhang, Qihua Zhou, Quan Chen, Weihua Zhuang, “Tree Learning: Towards Promoting Coordination in Scalable Multi-Client Training Acceleration,” *IEEE Transactions on Mobile Computing,* 2023.

* Equal contribution

[4] Jie Zhang, Song Guo, Xiaosong Ma, Haozhao Wang, Wenchao Xu, **Feijie Wu**, "Parameterized Knowledge Transfer for Personalized Federated Learning," in *Advances in neural information processing systems,* vol. 34, pp. 10092–10104, 2021.

[5] **Feijie Wu**, Shiqi He, Yutong Yang, Haozhao Wang, Zhihao Qu, Song Guo, Weihua Zhuang, "On the Convergence of Quantized Parallel Restarted SGD for Central Server Free Distributed Training," arXiv preprint arXiv:2004.09125, 2020.

Acknowledgements

Throughout the accomplishment of my M.Phil. study, I have received extensive support and assistance. In this thesis, I would like to express my deepest appreciation to those who have given me a hand during my research career.

First and foremost, I would like to thank my supervisor, Prof. Song Guo, for his invaluable guidance during my M.Phil. studies at the Hong Kong Polytechnic University. His expertise has been a source of inspiration, guiding me in formulating research problems and methodologies. His insightful suggestions have kept me informed about the latest techniques, and his encouragement has motivated me to strive for excellence in my academic endeavors.

Second, I am grateful to my previous supervisors before my M.Phil. study. They are Prof. Victor C.M. Leung from the University of British Columbia, Dr. Henry C.B. Chan from the Hong Kong Polytechnic University, and Dr. Wei Cai from the Chinese University of Hong Kong, Shenzhen. They give me a lot of support and light up my passion for chasing higher degrees. Their trust offers me endless confidence such that I can smoothly finish my M.Phil. study and step forward to my future Ph.D. study.

Third, in addition to the supervisors mentioned above, it is my great honor to collaborate with different people throughout my research career. In particular, thanks to Prof. Qiao Yan, Prof. Weihua Zhuang, Dr. Quan Chen, Dr. Jingcai Guo, Dr. Zhihao Qu, Dr. Haozhao Wang, Dr. Wenchao Xu, Ms. Tao Guo, Mr. Shiqi He, Mr. Ziming Liu, Mr. Xiaosong Ma, Mr. Ho Yin Yuen, Mr. Yutong Yang, Mr. Haobo Zhang, Ms. Jie Zhang, Mr. Jiewei Zhang and Mr. Qihua Zhou. Their efforts essentially make every project I work on come a success.

Fourth, I wish to extend my special thank to all former and current PolyU Edge Intelligence Laboratory members for their implicit or explicit kind assistance. They provide me with unforgettable experiences, and I am deeply inspired when I have meetings with these colleagues. I have no excuse but to list their names here: Dr. Quan Chen, Dr. Jingcai Guo, Dr. Jing Li, Dr. Jinwen Liang, Dr. Hao Ren, Dr. Haozhao Wang, Dr. Junxiao Wang, Dr. Xin Xie, Dr. Wenchao Xu, Mr. Peiran Dong, Ms. Tao Guo, Mr. Zijian He, Mr. Zicong Hong, Ms. Xun Liu, Mr. Yi Liu, Mr. Ziming Liu, Mr. Xiaosong Ma, Mr. Jun Pan, Mr. Junbao Pan, Mr. Xueyang Tang, Ms. Yingchun Wang, Mr. Han Wu, Mr. Leijie Wu, Mr. Zhenda Xu, Ms. Jie Zhang, Mr. Jiewei Zhang, Mr. Rui Zhang, Mr. Enyuan Zhou, Mr. Qihua Zhou.

Lastly, I would be remiss in not mentioning my friends and my family, especially my parents and my little sister. Their belief in me has kept my spirits and motivation high enough to move forward to a more incredible stage.

Names list in this part following the sorting rules on people's (1) title, i.e., Prof. > Dr. > Ms. = Mr.; (2) surname, and (3) given name based on alphabetical order.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

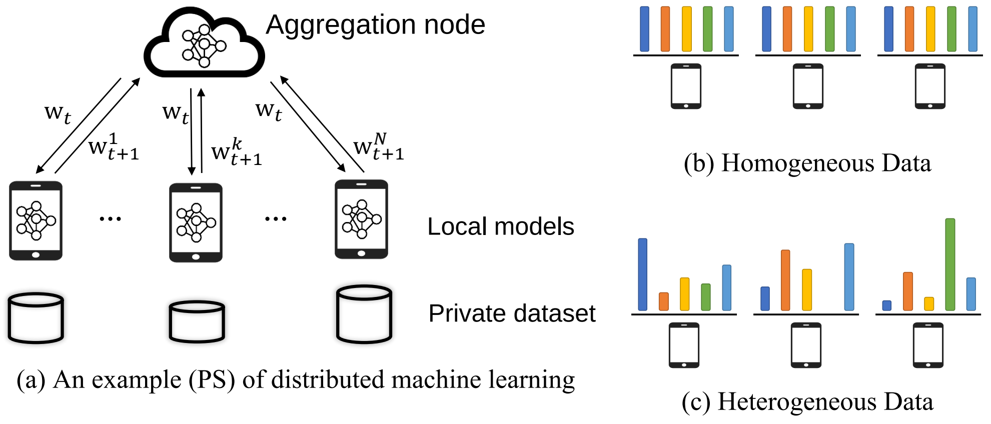
Over the past few years, IoT devices such as smartphones have been powerful enough to train complex models like deep neural networks. However, training on a single node can not be comparable with the traditional centralized training on the cloud. With the prevalence of edge devices around the world, collaborative training has now become a mainstream to expand the computational resources, inducing a golden period for the development of distributed machine learning (Alqahtani and Demirbas, 2019; Chen et al., 2018b; Li et al., 2015, 2014a,b; Sergeev and Del Balso, 2018; Verbraeken et al., 2020; Wan et al., 2020; Wu et al., 2018; Yu et al., 2019a,b), also known as large-scale machine learning (Bottou, 2010; Bottou et al., 2018; Tang et al., 2021). Recent years have witnessed its success in both computer vision (Agrawal et al., 2015; Campos et al., 2017; Liu et al., 2020c; Tron and Vidal, 2011; Yu and Liu, 2019) and natural language processing (Chen et al., 2019; Lin et al., 2021a; Liu et al., 2020a; Wu et al., 2020c). From the theoretical perspective, existing optimization studies (Bottou et al., 2018; Karimireddy et al., 2020b; Yu et al., 2019c,d) demonstrate that distributed machine learning achieves linear speedup with respect to the number of clients, while the traditional centralized training does not own such a characteristic.

Although distributed machine learning intrinsically achieves a non-trivial improvement, the research on its optimization still receives increasing attention because of its non-neglected disadvantages. Below discuss the drawbacks from three dimensions:

- **Communication:** Traditional distributed machine learning algorithms necessitate global synchronization at each iteration[1] with the full-precision transmission. It is obvious that the process consumes a significant amount of communication overhead and causes a considerable slowdown in training time. In a star topology (i.e., parameter server, abbr. PS) where all nodes communicate with the centralized server, the approaches possibly lead to network congestion and require extra cost during the training.

- **Computation:** An implicit assumption in distributed machine learning is that all devices have the same computational capability. However, a device's performance depends on various factors, including hardware configuration, battery level, etc. Thus, all nodes should wait for the slowest one at every communication round, negatively affecting training efficiency.

[1] In this thesis, an iteration is equivalent to a single local update; an epoch means the updates going through the entire local training dataset, consisting of multiple iterations. A communication round represents a global synchronization, before which each client can perform one or multiple iterations locally.



(a) An example (PS) of distributed machine learning

(b) Homogeneous Data

(c) Heterogeneous Data

Figure 1.1: Distributed machine learning and different data distributions

- **Data:** Traditionally, when a model trains on the cloud, the centralized entity partitions the training dataset into multiple nodes (or GPUs). Hence, the data are independent and identically distributed (i.i.d.) among nodes.

  When it comes to the crowdsourcing era, the data are generally held by individuals. Due to privacy concerns, the raw data do not share across the system. Therefore, the centralized server no longer controls the data distribution. Considering that the data collected by each client are less likely identical, we treat the data are heterogeneous among participants.

To avoid the overwhelming communication cost, the existing solutions are roughly classified into two categories, namely, (1) gradient compression and (2) communication frequency reduction. signSGD (Bernstein et al., 2018b; Liu et al., 2018; Safaryan and Richtárik, 2021; Tang et al., 2021), a family of gradient compression approaches, critically reduces the communication overhead because it utilizes one bit to represent each element in a gradient. The state-of-the-art methods such as SSDM (Safaryan and Richtárik, 2021) and majority vote (Bernstein et al., 2018b) have remarkable performance under PS. However, their implementation becomes limited under multi-hop all-reduce (MAR), an underlying synchronization paradigm that prevails in high-performance computing (HPC) systems. In the absence of a centralized coordinator to whom each node submits its data independently, cascading compression inevitably occurs. In this process, every participant performs decompression and compression to ensure every transmission is compressed. As a result, the cumulative error is excessively enormous when MAR uses a sign matrix that incorporates all clients' gradients through cascading compression.

FedAvg (Li et al., 2019b; McMahan et al., 2017), also known as local SGD (Stich, 2018) and parallel restarted SGD (Yu et al., 2019c,d), is another solution that saves the communication cost by reducing the synchronization frequency. To alleviate computational heterogeneity, we allow each client to perform inconsistent local updates to the best of resource utilization. Under some circumstances, this strategy can accelerate the training with the linear speedup of the averaged local updates (Yu et al., 2019d). However, with non-i.i.d. data, recent studies (Mitra et al., 2021; Wang et al., 2020b, 2021d) discover objective inconsistency under quadratic functions. In other words, the convergence property is destroyed, even though the degree of data heterogeneity is mild.

As a solution, Wang et al. (2020b, 2021d) introduces an algorithm named FedNova, a gradient normalization method that averages the normalized local gradients and accordingly updates the global model at the server. However, the approach cannot converge to the optimal solution due to the severe client-drift (Karimireddy et al., 2020b), where the global model inclines to the local model with fewer local updates. To mitigate the client-drift effect, SCAFFOLD (Karimireddy et al., 2020b) leverages variance reduced techniques (Johnson and Zhang, 2013; Liang et al., 2019; Schmidt et al., 2017) and achieves a remarkable convergence performance when participants perform a fixed number of local updates. However, its performance struggles under computational heterogeneity. By utilizing the intermittent results of the most recent local training, SCAFFOLD suffers from the client-drift effect because the bias of a client's orientation depends on the number of local updates such that the deviation varies. To sum up, existing approaches suffer from a heterogeneity dilemma: The solutions to computational heterogeneity cannot mitigate data heterogeneity, and the performance of data heterogeneity solvers is seriously affected by computational heterogeneity.

In this thesis, we separately introduce two gradient-wise approaches to breakthrough the aforementioned limitations:

- To avoid cascading compression and support signSGD implementation under MAR, we propose a gradient compression framework, Marsit. The main idea is to use a bit-wise process to accomplish unbiased sign aggregation. The sign of an element remains unchanged if and only if it has the same sign in both vectors; if it has different binary values, it follows a preset probability distribution. As a result of this action, the reception and compression processes can run parallel. We also offer a global compensation mechanism to minimize the compression error gap. Since data on the cloud can be shuffled and generated an identical distribution among workers, the design aims to equalize the clients' contributions to the final gradient. We do a full-precision transmission on a regular basis to get rid of surplus error accumulation.

- To settle objective inconsistency and jointly overcome data and computational heterogeneity, we devise a gradient calibration approach, FedaGrac. The key idea is to calibrate each local update according to the global update's predictive orientation, such that the detrimental effect of deviation on convergence is considerably reduced.

1.1 Contributions

The contribution of this thesis are highlighted as follows.

1. **Marsit: A Gradient Compression Approach**

   Traditional one-bit compressed stochastic gradient descent can not be directly employed in multi-hop all-reduce, a widely adopted distributed training paradigm in network-intensive high-performance computing systems such as public clouds. According to our theoretical findings, due to the cascading compression, the training process has considerable deterioration on the convergence performance. To overcome this limitation, we implement a sign-bit compression-based learning synchronization framework, Marsit. It prevents cascading compression via an elaborate bit-wise operation for unbiased sign aggregation and its specific global compensation mechanism for mitigating compression deviation. The proposed framework retains the same theoretical convergence rate as non-compression mechanisms. Experimental results demonstrate that Marsit reduces up to 35% training time while preserving the same accuracy as training without compression.

2. **FedaGrac: A Gradient Calibration Approach**

   In the setting of federated optimization, where a global model is aggregated periodically, step asynchronism occurs when participants conduct model training by efficiently utilizing their computational resources. It is well acknowledged that step asynchronism leads to objective inconsistency under non-i.i.d. data, which degrades the model accuracy. To address this issue, we propose a new algorithm FedaGrac, which calibrates the local direction to a predictive global orientation. Taking advantage of the estimated orientation, we guarantee that the aggregated model does not excessively deviate from the global optimum while fully utilizing the local updates of faster nodes. We theoretically prove that FedaGrac holds an improved order of convergence rate than the state-of-the-art approaches and eliminates the negative effect of step asynchronism. Empirical results show that our algorithm accelerates the training and enhances the final accuracy.

1.2 Organization

The rest of the thesis is organized as follows. In Chapter 2, we introduce the background knowledge related to this thesis, including distributed machine learning and the state-of-the-art techniques related to gradient-wise optimization. Subsequently, Chapter 3 presents a gradient compression approach named Marsit. Then, our study on a gradient calibration technique FedaGrac is proposed in Chapter 4. Chapter 5 summarizes the thesis and provides the concluding remarks and potential future research directions.

Chapter 2

Literature Review

This chapter briefly reviews the background knowledge related to the thesis. Section 2.1 provides an overview of distributed machine learning from a traditional perspective. Next, in Section 2.2, we introduce federated learning (FL) and discuss its underlying challenges and existing solutions. Later, the state-of-the-art techniques lying in gradient compression and gradient calibration are summarized in Section 2.3 and Section 2.4, respectively.

2.1 Distributed Machine Learning

Distributed machine learning targets to train large-scale deep learning systems with multiple clients. This section discusses the basic components of distributed machine learning, namely, the methodologies and the underlying synchronization paradigms.

2.1.1 Methodologies

In this field, traditional approaches can be roughly categorized as mini-batch SGD and local SGD. In mini-batch SGD, each client in parallel utilizes a mini-batch to calculate the corresponding gradient to improve the training efficiency such that mini-batch training arouses serious thinking (Dekel et al., 2012; Takác et al., 2013; Zinkevich et al., 2010). However, mini-batch SGD also has the problem of low computational efficiency. Some distributed deep learning frameworks training with large-batch only (Goyal et al., 2017; Shallue et al., 2019; You et al., 2018) often meets generalization issues, which increases training errors (Lin et al., 2019). Therefore, local SGD (also known as FedAvg) (Dieuleveut and Patel, 2019; Haddadpour and Mahdavi, 2019; Haddadpour et al., 2019; Stich, 2018) has become a more practical way to perform multiple local updates on each device before exchanging between devices. Bijral et al. (2016) analyzed the spectral norm of different datasets and constructed a graph of different clients to study local SGD. While Yun et al. (2021) focused on shuffling-based variants, that is, the practical gradient can be obtained without replacing sampling.

2.1.2 Underlying Synchronization Paradigms

Parameter Server (PS) Parameter Server (PS) is one of the most common centralized paradigms for large-scale distributed training (Dean et al., 2012; Li et al., 2014b; Smola and Narayanamurthy, 2010). It typically consists of one or more server nodes and multiple worker nodes, each carrying a subset of training data.

The worker nodes compute the gradients based on the local parameter information in parallel, and the server node processes the gradients sent by workers. The worker nodes then update their parameters using the averaged gradients. As a worker node only needs to exchange gradients with the server node, this paradigm is easy to implement and maintain (Li et al., 2014b). However, since server nodes handle all communications, the performance of PS is largely determined by the bandwidth of server nodes.

All-Reduce (AR) The workers are able to preserve a consistent model using All-Reduced (AR) paradigm without introducing central nodes (Patarasuk and Yuan, 2009). Frequently, the compute nodes are arranged in a ring-like topology, e.g., 2D-Torus or ring (Verbraeken et al., 2020). AR paradigm successfully releases the burden of the only central node to multiple transit nodes and reveals the same training efficiency as server-based architecture. One of its successful practices, Ring AR (Baidu-Research, 2017; Sergeev and Del Balso, 2018), keeps to a minimum communication overhead as well as outperforms PS because it makes good use of overlapping computation and communication (Alqahtani and Demirbas, 2019). However, its long handshaking processes sometimes slow down the training in high latency network (Lian et al., 2018).

Gossip As a fully decentralized model, Gossip is attracting growing attention because it does not require model aggregation among all workers (Lian et al., 2017b, 2018; Tang et al., 2018). Instead, each worker solely communicates with their neighbors. Such a structure is promising not just because it leverages a minimal communication overhead but because devices are increasingly connected to others using rapid communication links in modern communication networks. For example, in 5G and beyond mobile networks, mobile devices can connect directly with neighboring devices via high-speed device-to-device links. Edge devices inside the same local-area network (LAN) domain can also interact quickly without having to go via a sluggish wide-area network (WAN) (Guo et al., 2021).

2.2 Federated Learning

As one of the subcategories of distributed machine learning, FL was proposed to ensure data privacy and security with the avoidance of raw data sharing (Kairouz et al., 2019), and now it has become a hot field in the distributed system (Avdiukhin and Kasiviswanathan, 2021; Blum et al., 2021; Diao et al., 2020; Shamsian et al., 2021; Yuan and Ma, 2020; Yuan et al., 2021; Zhang et al., 2021). Frequently, edge devices such as smartphones possess abundant data, which are highly sensitive but valuable to the model training (Guo and Qu, 2022; Han et al., 2020; Lim et al., 2021; Wang et al., 2021c). The data are heterogeneous among clients because there is no predefined rule for the data distribution for each client. Besides, due to the hardware differences among devices, the computational capabilities are various as well. In this section, we briefly investigate the flaws raised by data heterogeneity and computation heterogeneity and review the existing work to tackle these two issues.

2.2.1 Data Heterogeneity

Generally, in FL settings, the data distributed among clients are agnostic, and therefore, each data portfolio has its exclusive optimal parameters. As a classical algorithm that works smoothly under data heterogeneity, FedAvg inherits the training features from local SGD (Stich, 2018; Yu et al., 2019d; Zhou and Cong, 2018), a framework that runs for multiple local updates prior to a global synchronization. This strategy significantly reduces the total communication overhead compared to parallel SGD that synchronizes the gradient at every local update. Recent studies (Gu et al., 2021; Khaled et al., 2020; Li et al., 2019b) show that FedAvg can have a great performance from theoretical and empirical perspectives. Also, FedAvg can seamlessly adopt the communication-efficient approaches such as quantization (Alistarh et al., 2017a; Basu et al., 2019) and sparsification (Stich et al., 2018; Wangni et al., 2018) to further reduce the cost of transmission (Wang et al., 2021a; Wu et al., 2020a,d; Zhou et al., 2021).

Nevertheless, numerous studies (Cheng et al., 2021; Gorbunov et al., 2020; Karimireddy et al., 2020b; Liu et al., 2020b; Zhao et al., 2018) theoretically quantify how data heterogeneity affects FedAvg and degrades the convergence property. As a result, some variants of FedAvg are designed to mitigate the negative impact. These modifications include adding a proximal term to local objective functions (Li et al., 2020), using a decreasing learning rate (Li et al., 2019b), adaptive server side updates (Hsu et al., 2019; Reddi et al., 2020), client clustering sampling (Fraboni et al., 2021; Ghosh et al., 2020; Murata and Suzuki, 2021), reinforcement learning driven incentive mechanism (Wang et al., 2020a), and etc.

2.2.2 Computational Heterogeneity

The computation capabilities vary among clients because they use different devices. As a result, all clients have to wait for the slowest node to start the next round. To minimize the computation differences, some existing works adopt a client sampling strategy (Deng et al., 2021; Huang et al., 2020; Wu et al., 2020b; Yang et al., 2020; Zhou et al., 2020), where only a small portion of clients transmit the gradients to the server. Compared to the case that requires full-worker participation, this scheme possibly reduces the total training time. However, resource underutilization still exists as the fastest client should wait for others' completion. Therefore, the existing solutions to overcoming computational heterogeneity can be categorized into two types: designing an asynchronous aggregation scheme and adopting step asynchronism.

Asynchronous Aggregation Scheme Hogwild (Recht et al., 2011) is one of the main examples of asynchronous stochastic algorithms. It does not use the memory locking protocol, so each node can modify the parameters at the same time. (Noel and Osindero, 2014) proposed Dogwild!, which is an improvement to Hogwild. It is distributed Hogwild for CPU and GPU. Similarly, for Hogwild, (De Sa et al., 2015) analyzed its non-convexity by using relaxed assumptions. With the emergence of deep neural networks, asynchronous parallel SGD algorithms have begun to adapt to environmental changes (Aytekin et al., 2016; Dai et al., 2018; Lian et al., 2015; Zhang and Kwok, 2014; Zheng et al., 2017). These methods efficiently break the barrier of heterogeneity, achieving high system throughput.

However, asynchronous algorithms raise the staleness issue, usually deteriorating the convergence rate.

Step Asynchronism Step asynchronism is another practical solution, where each client performs an inconsistent number of local updates. Although FedAvg with step asynchronism can converge to a stable point under non-convex objectives (Yu et al., 2019d), Wang et al. (2020b) point out that objective inconsistency takes place under quadratic function, leading to a suboptimal convergence. To tackle this issue, effective approaches is constituted with normalization-based approach FedNova (Wang et al., 2020b) and FedLin (Mitra et al., 2021), regularization-based approach FedProx (Li et al., 2020) and architecture-based approach HeteroFL (Diao et al., 2020). Gradient normalization is the most ubiquitous framework that overcomes step asynchronism under non-i.i.d. data settings. However, this method cannot prevent the negative impact of statistical heterogeneity on the convergence rate because the update deviation still exists after averaging. Figure 4.1 compares FedNova (Wang et al., 2020b) and FedLin (Mitra et al., 2021) with our proposed method, and we notice that the global model deviates to the one with less updates in FedNova (Wang et al., 2020b). The reason is obvious: clients updating their local models are biased to their local datasets such that the normalized gradients collected by the server are sparse. Besides, when the local models approach the local minimizers, those clients with more local updates greatly influence the global model update.

2.3 Gradient Compression Techniques

Sparsification Existing methods for sparsification of distributed learning usually sparsify the gradient sent from workers. The works in early stages for sparsifying graident used to employ a filter to select several important dimensions, e.g., with the largest changes, and discard the rest (Hsieh et al., 2017; Li et al., 2014a). To improve the sparsification efficiency, recently, a large number of works that sparsify the gradient with error compensation have been proposed (Aji and Heafield, 2017; Alistarh et al., 2018; Chen et al., 2018a; Sattler et al., 2019; Shi et al., 2019; Stich et al., 2018). Error compensated gradient sparsification is to accumulate the unselect dimensions and compensate the gradient with the accumulated error. In such a way, the convergence gap with non-sparsification distributed SGD could be bridged. Beyond SGD, there are also some work moving towards the Momentum-SGD (Lin et al., 2017; Zhao et al., 2019). Different from the methods with error compensation, Wangni et al. (2018) propose a novel sparsification operator that the gradient could be sparsifed in an unbiased manner. Though these gradient sparsification based methods have made great achievements in the field of communication compression through sparsification under PS synchronization framework, they are not compatible with multi-hop all-reduce (e.g., ring all-reduce).

Quantization At the cost of the gradient precision, quantization approaches (Alistarh et al., 2017b; Basu et al., 2019; Seide et al., 2014; Suresh et al., 2017; Wen et al., 2017; Zhang et al., 2017) reduce the number of encoding bits for each real number. Notwithstanding that it requires more rounds to reach the stationary point under both PS and gossip paradigms, the method is potent from the perspective of training efficiency and bandwidth consumption. However, it is not suitable for multi-hop all-reduce synchronization framework to shorten the training time. Each client inevitably performs decoding then encoding operation for transmission. Such a recursive execution results in the error accumulation and degrades the convergence property in theoretical analysis. Without the constraint on the number of workers M, the deterioration can be up to $\sqrt{M}$ in comparison with the compression-free algorithms (Wu et al., 2020a). Although GradiVeQ (Yu et al., 2018) utilizes singular value decomposition to achieve linear quantization under multi-hop all-reduce synchronization framework, the process requires considerable computation consumption such that the receiving period cannot cover the time length of compression.

signSGD As an extreme case of quantization, signSGD represents the elements of a gradient using their signs, which reduces the communication overhead by $32\times$ at every iteration (Bernstein et al., 2018a). It has remarkable performance under PS, including 1-bit Adam (Tang et al., 2021), SSDM (Safaryan and Richtárik, 2021) and majority vote (Bernstein et al., 2018b). However, they are not suitable for MAR since their aggregation process cannot guarantee within one bit at each transmission.

Other communication compression approaches There are various approaches to reduce communication overhead, such as low rank (Vogels et al., 2019). However, these approaches may not have well performance for MAR under some network topologies. For instance, PowerSGD (Vogels et al., 2019) requires to transmit multiple sequential vectors at a synchronization, which undermines the training efficiency under RAR.

2.4 Gradient Calibration Techniques

Gradient calibration techniques, also known as variance reduction (Allen-Zhu and Hazan, 2016; Defazio et al., 2014; Fang et al., 2018; Horváth and Richtárik, 2019; Horváth et al., 2020; Johnson and Zhang, 2013; Lan and Zhou, 2018a,b; Lei et al., 2017; Li, 2019; Li et al., 2021b; Lian et al., 2017a; Nguyen et al., 2017; Reddi et al., 2016; Roux et al., 2012; Wang et al., 2018; Zhang et al., 2016; Zhou et al., 2018), are once proposed for traditional centralized machine learning to optimize finite-sum problems (Bietti and Mairal, 2017; Bottou and Cun, 2003; Robbins and Monro, 1951) by mitigating the estimation gap between small-batch (Bottou, 2012; Ghadimi et al., 2016; Khaled and Richtárik, 2020) and large-batch (Mason et al., 1999; Nesterov, 2003; Ruder, 2016). SGD randomly samples a small batch and computes the gradient in order to approach the optimal solution. Insufficiently large batch results in convergence rate degradation since the data are generally noisy. GD can remove the noise affecting the training process by utilizing all data in every update. However, it is time-consuming because the period for a single

GD step can implement multiple SGD updates. Based on the trade-off, variance-reduced methods periodically perform GD steps while correcting SGD updates with reference to the most recent GD steps.

Variance Reduction in FL. The variance-reduced techniques have critically driven the advent of FL algorithms (Gorbunov et al., 2021a,b; Karimireddy et al., 2020a,b; Li et al., 2019a, 2021c; Liang et al., 2019; Murata and Suzuki, 2021; Shamir et al., 2014; Wu et al., 2021) by correcting each local computed gradient for the global orientation. However, a concern is addressed on how to attain a proper global orientation to mitigate the update drift from the global model, especially under the communication-efficient settings where clients perform numerous local updates. SCAFFOLD (Karimireddy et al., 2020b) adjusts every local update with the help of the global and a client's local reference orientation such that every local update keeps close to the global direction. However, as shown in Figure 4.1, SCAFFOLD cannot completely remove the drift when computational heterogeneity exists. A physical explanation for the result is that the local reference directions of the faster nodes with more local updates lead to a significant deviation from the orientation towards the local optimizer. Since the global reference direction is derived from clients' local directions, the faster nodes dominate the entire training process (see Figure 4.1), which betrays its origin intention. Although we use a similar design philosophy where every local update follows the global orientation, the global orientation consists of the gradient that depends on the number of local updates, either the normalized gradient or the initial gradient.

Chapter 3

Marsit: A Gradient Compression Approach

3.1 Introduction

In an era of data explosion, there is an increasing demand for various fields to launch AI-driven applications in image classification (Pérez-Hernández et al., 2020), natural language processing (NLP) (Roy et al., 2021), and so forth. Behind these applications are numerous models that have been fit in huge-size datasets such as ImageNet (Russakovsky et al., 2015). To minimize the development cost, cloud providers, e.g., Amazon AWS, offer various training paradigms to enable fast AI/ML solution deployment.

Nowadays, multi-hop all-reduce (MAR) training paradigm, including ring all-reduce (RAR) (Baidu-Research, 2017; Sergeev and Del Balso, 2018) and 2D-torus

	Rounds	Accuracy (%)	Time (min)
cascading compression			
$M = 3$	▰▰▱▱ 187	87.2 ± 2.31	11.2
$M = 8$	▰▰▰▰ 1K+	divergence	NA
no compression			
$M = 3$	▰▱▱▱ 129	99.1 ± 0.13	20.7
$M = 8$	▰▱▱▱ 76	99.2 ± 0.07	10.6

Table 3.1: Training MNIST over AlexNet. The results show the best test accuracy by setting the stepsize in $\{0.03, 0.01, 0.005\}$.

all-reduce (TAR) (Mikami et al., 2018), substitutes classical single-hop approaches such as parameter server (PS) and gossip, and becomes the most pervasive synchronization paradigm in high-performance computing (HPC) systems. For parallel stochastic gradient descent (PSGD) (Li et al., 2014a), MAR achieves a better resource utilization under multi-GPU circumstance than PS. Firstly, all GPUs involve in both the training and synchronization in MAR, while GPUs in PS architecture are categorized into two groups separately performing these two functionalities. Secondly, MAR prevents the network congestion at a single node because each client is not required to simultaneously process tremendous transmission requests. As a paradigm that workers are solely permitted to communicate with their neighbors, gossip has made great success in recent years (Lin et al., 2021b; Lu and De Sa, 2021). However, the performance of gossip in terms of convergence rate is much slower than MAR, especially under sparse connections such as ring topology (Chen et al., 2021).

In network-intensive HPC systems such as public clouds, it is challenging to transfer a non-compressed gradient among nodes due to overwhelming bandwidth consumption. With the increasing size of a deep learning model, e.g., 60.2M weights on ResNet-152 (He et al., 2016) and 100T on GPT-4 (Brown et al., 2020), the problem becomes severe because data transmission takes a significant amount of time. As a promising communication compression approach, signSGD (Bernstein et al., 2018b; Liu et al., 2018; Safaryan and Richtárik, 2021; Tang et al., 2021) solely uses an element's sign to represent itself, where the number of encoding bits for each real number is dramatically deducted, i.e., from single float precision (32 bits) to 1 bit.

Existing signSGD algorithms, albeit well-performed under PS, have limited performance under MAR, especially when the model is sufficiently large. Without a centralized coordinator to which each node independently sends its data, information asymmetry occurs when MAR leverages a sign matrix that includes all clients' gradients through cascading compression. Each client inevitably performs decompression and then compression operation for transmission, accumulating errors. Although cascading compression can converge at the end for a small-scale environment, empirical studies in Table 3.1 manifest its poor performance in comparison with the non-compressed algorithms. Also, more workers achieves better performance in non-compressed PSGD, whereas leading to divergence in the cascading compression scheme.

To alleviate information asymmetry, we propose a framework for multi-hop all-reduce using sign-bit, named as Marsit. The core idea is to achieve unbiased sign aggregation by means of an elaborate bit-wise operation: The sign of an element remains unchanged if and only if it has the same sign in both vectors, while it follows a predefined probability distribution if it has different binary values. Such an operation supports that the reception and compression processes can take place in parallel. Furthermore, we introduce a global compensation mechanism to bridge the gap of compression error. The design is to equalize the clients' contribution towards final gradient because data on the cloud can be shuffled and formed an identical distribution among workers. To get rid of excess error accumulation, we periodically operate a full-precision transmission.

In this chapter, our contributions are summarized as follows:

- Based on the designed one-bit operator and the global compensation scheme, we implement Marsit to support one-bit transmission without cascading compression under MAR.

- We prove that the convergence rate for non-convex objectives is $O(1/\sqrt{TM})$ under RAR framework, where T and M represent the numbers of synchronizations and workers, respectively. The theoretical result indicates that our algorithm achieves a linear speedup simultaneously with respect to the number of workers. To the best of our knowledge, this is the first work that addresses information asymmetry under MAR;

- We conduct an empirical study to illustrate the effect of our proposed algorithms on RAR and TAR. It is conducted with ResNet-50 on ImageNet for image recognition. It reduces the communication cost by around 90% as compared with non-compressed methods while preserving the same convergence performance.

The rest of the chapter is organized as follows: Section 3.2 formulates the problem and provides the motivations via analyzing cascading compression. We comprehensively elaborate the design details behind Marsit and analyze the convergence rate of Marsit in Section 3.3. An empirical setting is presented in Section 3.4, while the numerical results are presented in Section 3.5, to validate our theoretical analysis. Section 3.6 concludes this study.

3.2 Motivation

Objectives. Under an M-worker MAR, the objective is to minimize the cumulative expected loss, which can be formulated as

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \quad F(\boldsymbol{x}) = \frac{1}{M}\sum_{m=1}^{M} \underbrace{\mathbb{E}_{\xi_m\sim\mathcal{D}_m}\left[f_m(\boldsymbol{x},\xi_m)\right]}_{:=F_m(\boldsymbol{x})}, \tag{3.1}$$

where $\mathcal{D}_m$ is the local data distribution on worker m, $f_m(\boldsymbol{x},\xi_m)$ is the empirical loss given parameter $\boldsymbol{x}$ and stochastic sample ξ_m from $\mathcal{D}_m$, and $F_m(\cdot)$ is an objective function. Given that the entire training locates in the cloud, we assume that all the local datasets have an equal size. Since the objective function is randomly extracted over a given data distribution, it is a common practice that the bias does not exist between the expected loss and the empirical one.

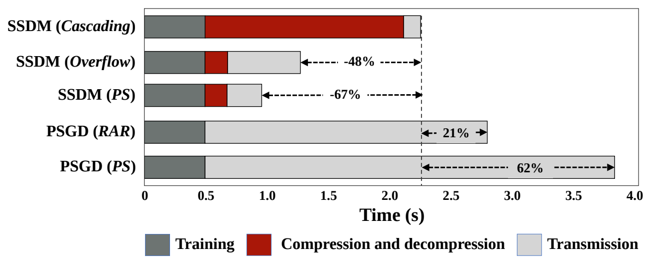
3.2.1 Why Bit Length Expansion Occurs?

In a non-compressed algorithm, MAR naturally requires less communication overhead than PS when synchronizing a model among all nodes. For example, given a D-dimension neural network, RAR requires the consumption of $2 \times (M-1) \times D$ weights, while PS needs that of $2 \times M \times D$. In Figure 3.1a, non-compressed approach under RAR costs less time than the one under PS.

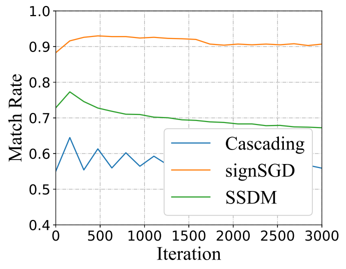
An operation compatible to MAR should be linear, which allows workers directly aggregate without extra decompression-compression process (Vogels et al., 2019). SSDM (Safaryan and Richtárik, 2021) is one of rare signSGD approaches that satisfy the requirement of linearity, where its aggregation is to sum up all the sign bit. With the operation, workers do not fit the transmission elements into one bit under MAR synchronization, but with an upper bound of $\lceil \log_2 M \rceil$. As shown in Figure 3.1a, such way spends longer time than its PS solution in transmission period due to the growing size of transmission packages. Therefore, the approach is not efficient under MAR settings and we are dedicated to implementing a compression framework that restricts the transmission size by only one bit.

3.2.2 Why Not Cascading Compression?

For a D-dimension vector $\boldsymbol{g}$, SSDM (Safaryan and Richtárik, 2021) (denote by $\mathcal{Q}$) compresses an element g_i ($i \in \{0, ..., D-1\}$) consistent with its sign following the probability of $\frac{1}{2} + \frac{|g_i|}{\|\boldsymbol{g}\|_2}$, where $\|\cdot\|$ means ℓ_2-norm. Apparently, it is an unbiased compression method. To ensure each transmission limited in one bit, a client



(a) Time length per iteration



(b) Matching rate

Figure 3.1: Training MNIST over AlexNet with 3 workers. The comparison of existing approaches on an iteration's training time length and matching rate.

performs the step-by-step sequence:

- **Receive** aggregated gradient segment(s), including corresponding ℓ_2-norm(s) and sign vector(s), from the last worker(s);
- **Recover** the gradient segment(s) as $\boldsymbol{w}$ for full precision;
- **Aggregate** local gradient $\boldsymbol{v}$ with decompressed segment(s);
- **Compress** the assembled segment into a precision-loss one, i.e., $\mathcal{Q}(\boldsymbol{w}+\boldsymbol{v})$;
- **Send** the compressed segment to the next worker(s).

The workflow, named as cascading compression, is able to broadcast and unify the updates among clients. Obviously, the expected result of cascading compression is equivalent to the sum of all gradients. However, cascading compression has two major shortcomings.

3.2.2.1 Performance Deterioration

Notably, the second step cannot actually represent the real aggregation results. In this case, the error accumulates and spreads over the network, which deteriorates the training performance. Besides, it is not suitable to use the ℓ_2-norm to achieve unbiased compression because its value is so large that the new compressed sign is more likely biased to the received one, even if the actual aggregation sign should be the opposite one. As demonstrated in Figure 3.1b, among the applicable settings, cascading compression has the lowest matching rate (i.e., around 56%) measured by the sign of non-compression aggregation value. Following remark compares the performance between cascading compression under RAR and centralized training under PS.

Remark. We assume that the ℓ_2-norm of any gradients are bounded by a non-negative scalar G. Suppose SSDM (Safaryan and Richtárik, 2021) is achieved as unbiased estimator under centralized training and cascading compression, where the expected update value is equivalent to the update of non-compression algorithm. For training a deep neural network where the value of D is quite large, the upper bound of gradient deviation, i.e., the Euclidean distance between the expected result and the actual update, for cascading compression explodes rapidly with M, while centralized training does not exist.[1]

[1] The detailed proofs for the remark present in Appendix A.

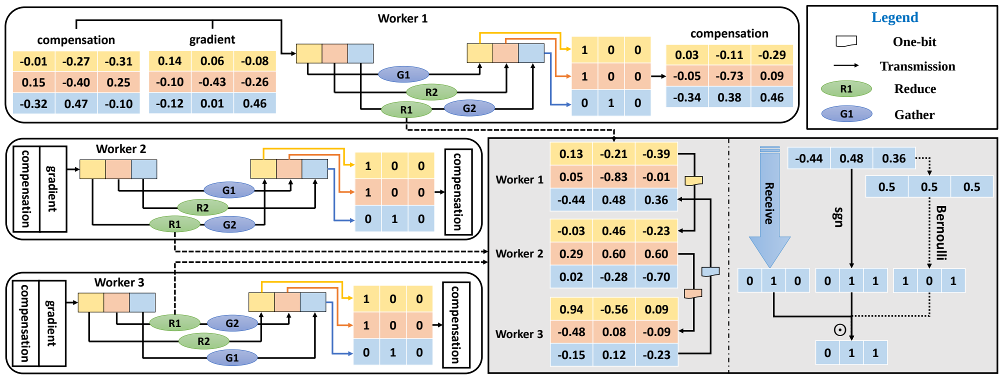


Figure 3.2: The workflow of Marsit under ring network topology with a total of three workers. Each worker holds a compensation vector and a gradient, combining them into a standalone vector. This vector is then segmented and exchanged during synchronization, involving a green-highlighted reduce period (R) and a blue-highlighted gather period (G). During R1 and R2, a worker processes received messages with its local segment, exemplified in the gray box. The left side of the box illustrates message transfer among workers, while the right side demonstrates the transfer from worker 3 to worker 1 (highlighted in azure), showcasing aggregation. In the subsequent G1 and G2 gather periods, a worker replaces its local segment with received information before transmitting it to the next worker.

3.3 Marsit

In this section, we first provide a holistic insight for Marsit. Then, in Section 3.3.1 and Section 3.3.2, we present the technical details and the theoretical analysis, respectively.

Due to the lack of centralized server under MAR, all workers should maintain a global model locally, the parameters of which are always consistent with others. Figure 3.2 illustrates the pipeline of Marsit under RAR, a common paradigm for MAR using ring network topology. Each worker possesses a compensation vector and a gradient, and aggregates them into a standalone vector. Then, they partition

```
Algorithm 1: Marsit (worker $m$)
   Require : Synchronization index $t$, number of communication rounds for
             full-precision synchronization $K$, gradient $g_t^{(m)}$, compensation vector
             $c_t^{(m)}$, global stepsize $\eta_s$
 1 Calculate the update by $g_t^{(m)} \leftarrow g_t^{(m)} + c_t^{(m)}$;
 2 Split $g_n^{(m)}$ into $M$ parts, and denote by $g_{t,i}^{(m)}, \forall i \in \{0, ..., M-1\}$;
 3 if mod(t, K) $\neq 0$ then
 4     for $i \leftarrow 0$ to $M-1$ do
 5         Receive the sign vector $v_i$ in parallel with
             • Calculate the sign vector by $v_i^* \leftarrow \text{sgn}\left(g_{t,i}^{(m)}\right)$;
 6         Update the transmission sign vector via $v_i \leftarrow v_i \odot v_i^*$;
 7         Send $v_i$ to the next worker;
 8     end
 9     Aggregate the global update via $g_t \leftarrow \eta_s \cdot \left(\bigcup_{i=0}^{M-1} v_i\right)$;
10     Update compensation vector via $c_{t+1}^{(m)} \leftarrow g_t^{(m)} - g_t$;
11 else
12     Aggregate the global update via $g_t \leftarrow \frac{1}{M}\sum_{m=1}^{M} g_t^{(m)}$;
13     Update compensation vector via $c_{t+1}^{(m)} \leftarrow 0$;
14 end
   Return : The global update $g_t$, compensation vector $c_{t+1}^{(m)}$
```

the vector into several segments and exchanges them at the synchronization phase which consists of a reduce period (highlighted in green and marked as R) and a gather period (highlighted in blue and marked as G). In the reduce period, i.e., R1 and R2, a worker processes the received message with corresponding local segment and sends it to the next worker. Here we exemplify with R1 and depict the procedures in the gray box. The left part of the gray box presents how the message transfers among workers, while the right takes the message transferring from worker 3 to worker 1 (highlighted in azure) as an example and exhibits how to aggregate the received vector and the local vector. In the gather period, i.e., G1 and G2, a worker substitutes corresponding local segment with the received information and transmits it to the next worker. The relevant processes have been widely adopted in (Baidu-Research, 2017; Sergeev and Del Balso, 2018). After the synchronization phase, all clients reach to a consensus and holds the same gradient which is used to update the global model and the local compensation vector.

3.3.1 Implementation Details

Here we discuss the key operations with in-depth justifications. Generally, the workflow lies in two phases: one-bit synchronization in each round to reduce the communication cost, and full-precision synchronization executed every K rounds to periodically eliminate the error accumulation. The full implementation is given in Algorithm 1 to demonstrate the workflow behind Marsit, and Algorithm 2 is to illustrate how we can apply Marsit to existing optimizers like stochastic gradient descent (SGD).

3.3.1.1 Global Model Synchronization (Line 4–8 in Algorithm 1)

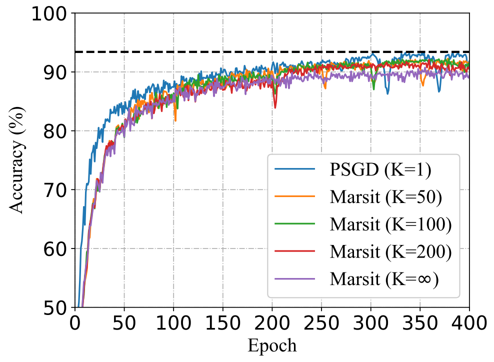
No matter which phase it is, Marsit synchronizes the gradients through MAR. Full-precision synchronization has been widely discussed in the previous studies (Baidu-Research, 2017; Jia et al., 2018; Mikami et al., 2018; Sergeev and Del Balso, 2018), which is equivalent to the aggregation result under PS, we mainly focus on the synchronization using sign bit only in this part.

As illustrated in Figure 3.2, both receiving vector v_i and local compression

v_i^* (Line 5 in Algorithm 1) run in parallel, which reduces a great amount of time in comparison with the cascading compression. Since both v_i and v_i^* are a sign-bit vector, a problem raises on how to aggregate both vectors without additional compression-decompression processes. Therefore, we define a novel bit-wise operator $\odot$ to ensure these two vectors compatible with each other. In this update process, if an index on both vectors is the same, then the transmission vector at this points remains unchanged. However, considering element inconsistency between v_i and v_i^*, we use a transient vector, v, which predetermines the transmitted binary value when confronted with inconsistent elements. It follows a Bernoulli distribution: Let b_j be the probability for the element j of vector v_i^* (denote by $v_{i,j}^*$) at worker m that marks as 1 in vector v:

$$b_j = \begin{cases} (m-1)/m & v_{i,j}^* = 0 \\ 1/m & v_{i,j}^* = 1 \end{cases} \overset{\text{Bernoulli}}{\Longrightarrow} v_j = \begin{cases} 1 & pr = b_j \\ 0 & \text{Otherwise} \end{cases} \tag{3.2}$$

Note that the process can take place in parallel with the receiving stage but after the calculation of v_i^*. With the transient vector v, the updated operator $\odot$ between v_i and v_i^* should be expressed as: $v_i \odot v_i^* = (v_i \text{ AND } v_i^*) \text{ OR } (v_i \text{ XOR } v_i^* \text{ AND } v)$. By mathematical analysis, the expected value of the sign bit is equivalent to the average of the sign bits among all clients.



(a) Epoch and Accuracy

K	Time (min)	Acc. (%)	Bits
1	40.18	93.42	32
50	22.05	92.28	1.62
100	21.34	91.73	1.31
200	22.38	92.00	1.16
∞	18.78	90.75	1

(b) Convergence results

Figure 3.3: Training CIFAR-10 over AlexNet by evaluating various values of K. (a) indicates the relation between epoch and accuracy; and (b) depicts the convergence result. $K = \infty$ means K is greater than the maximum communication rounds, i.e., 400 in this case.

3.3.1.2 Global Model Update (Line 9 and Line 12 in Algorithm 1)

The value g_t depends on whether the synchronization is under full precision. If the transmission is sign-bit only, Line 9 returns g_t that comes from a vector of signs multiplying a global learning rate. As for full-precision synchronization in Line 11, extra learning rate is not necessary since $g_t^{(m)}$ has included the local stepsize. The purpose for this update is to eliminate the accumulated error and accelerate the training process. In Figure 3.3, we demonstrate there exists a trade-off between the final accuracy and the additional communication costs due to full precision synchronizations, by choosing different system parameter K.

3.3.1.3 Global Compensation Mechanism (Line 10 and Line 13 in Algorithm 1)

At the beginning of the model training, we initialize the local compensation gradient with a zero vector (Line 1 in Algorithm 2) by default. All clients have the consensus on how to update the global model, i.e., g_t at Line 9 in Algorithm 1, which is a vector containing binary value only to indicate the sign of each element. Unlike traditional compensation approaches under single-hop synchronization, a client in Marsit cannot obtain how much it contributes to the aggregation under multi-hop synchronization. Based on the independent and identical data distribution on cloud training, every client compresses and obtains the same gradient in expectation. Thus, we apply an identical local compensation amount for each client, which then combines into the global compensation. Considering the accumulated error could be quite large, we periodically reset the error by means of full-precision synchronization, where the compensation vector can be set to 0. As we can see in Figure 3.3, although greater K costs less time to reach the stable point, they have smaller convergence accuracy. Also, greater K may not always speed up the convergence progress, for instance, when K changes from 100 to 200, more time is required to realize the convergence feature.

3.3.2 Theoretical Guarantees

To theoretically analyze the convergence results for Marsit, we have the following assumption for Problem (3.1), which are ubiquitously applied to (Bernstein et al.,

Algorithm 2: Marsit-driven SGD (worker m)

```
Input  : Initial Point $\tilde{\boldsymbol{x}}_0$, local stepsize $\eta_l$, global stepsize $\eta_s$, number of
         communication rounds for full-precision synchronization $K$, number
         of global synchronizations $T$
1 Initialize local compensation gradient $c_0^{(m)} \leftarrow \mathbf{0}$;
2 for $t \leftarrow 0$ to $T-1$ do
3     Randomly sample $\xi_k^{(m)}$ from local data $\mathcal{D}_m$;
4     Compute local stochastic gradient $g_t^{(m)} \leftarrow \nabla f_m\left(\tilde{\boldsymbol{x}}_t; \xi_k^{(m)}\right)$;
5     $g_t, c_{t+1}^{(m)} \leftarrow$ Marsit($t$, $K$, $\eta_l g_t^{(m)}$, $c_t^{(m)}$, $\eta_s$);
6     Update the parameters through $\tilde{\boldsymbol{x}}_{t+1} \leftarrow \tilde{\boldsymbol{x}}_t - g_t$;
7 end
Output : The final model $\tilde{\boldsymbol{x}}_T$
```

2018a; Guo et al., 2020; Safaryan and Richtárik, 2021).

Assumption 3.1. *Problem (3.1) satisfies the following constraints:*

1. ***Smoothness:*** *All function $F_m(\cdot)$'s are continuous differentiable and their gradient functions are L-Lipschitz continuous with $L > 0$;*

2. ***Bounded variance:*** *For any worker m and vector $\boldsymbol{x} \in \mathbb{R}^d$, there exists a scalar $\sigma \geq 0$ such that $\mathbb{E}_{\xi\sim\mathcal{D}_m}\|\nabla f_m(\boldsymbol{x}, \xi) - \nabla F_m(\boldsymbol{x})\|_2^2 \leq \sigma^2$.*

Based on the preceding assumptions, the following theorem holds:

Theorem 3.1. *Under Assumption 3.1, by setting local learning rate for $\eta_l = \sqrt{M/T}$ and the global learning rate $\eta_s = \sqrt{1/TD}$, the upper bound for Algorithm 2 using RAR-based should be:*

$$\min_{t\in\{0,\dots,T-1\}} \mathbb{E}\left\|\nabla F(\tilde{\boldsymbol{x}}_t)\right\|_2^2 \leq \mathcal{O}\left(\frac{1}{\sqrt{MT}}\right) + \mathcal{O}\left(\frac{K(K+1)}{T}\right)$$

where we treat $F_* - F(\tilde{\boldsymbol{x}}_1)$, L and σ as constants.

Proof. See Appendix B for details. $\square$

Remark. Given that the value of K is much smaller than the value of T, our approach can achieve a convergence rate of $O(1/\sqrt{MT})$, which achieves linear speedup with the number of the workers. In other words, the more GPUs participate in the model training, the faster Marsit reaches a stable point.

3.4 Experimental Setup

We evaluate our proposed framework on scenarios that meet the requirement of current industrial needs and cover the most representative model training instances on the public clouds. In this section, the problem we explore mainly lies in these two categories: (i) whether there exists a significant accuracy drop in comparison with non-compression methods; (ii) how fast a model achieves convergence in comparison with existing compression approaches under MAR.

Datasets, models and tasks Our experiments consist of three datasets: CIFAR-10 (Krizhevsky et al., 2009a), ImageNet (Russakovsky et al., 2015) and IMDb reviews (Maas et al., 2011). The first two datasets are frequently used for image classification and consist of 60K 32$\times$32 and 14M 224$\times$224 colored images, respectively. The last one is for sentiment analysis with 50K movie reviews. The models vary among the datasets: AlexNet (Krizhevsky et al., 2012) and ResNet-20 (He et al., 2016) for CIFAR-10, ResNet-18 and ResNet-50 for ImageNet, and

DistilBERT (Sanh et al., 2019) for IMDb reviews.

Implementation The experiments are conducted on Huawei Cloud, where we deploy a cluster with 32 nodes and each node carries 2 Nvidia T4 GPUs. The underlying training framework is supported by Pytorch distributed computing package[2]. We implement Marsit on RAR (Baidu-Research, 2017; Sergeev and Del Balso, 2018), a classical MAR implementation over ring network topology, and 2D-torus all-reduce (TAR), a state-of-the-art MAR scheme over 2D-torus network topology. Marsit can be easily extended to other all-reduce paradigms including segmented-ring all-reduce (Jia et al., 2018) and tree all-reduce (Vogels et al., 2019).

Baselines We implement multiple baselines to evaluate the performance of Marsit. PSGD (Li et al., 2014a) is implemented under MAR with full precision, i.e., 32 bits. For EF-signSGD (Karimireddy et al., 2019), signSGD with majority vote (Bernstein et al., 2018a) and SSDM (Safaryan and Richtárik, 2021), we extend them to MAR by dynamically changing the bit length. We also utilize Elias coding (Elias, 1975) to compact the transmission message among nodes.

Optimizers and hyper-parameters To reduce the frequency of the communications among nodes, clients perform multiple local updates between two successive synchronizations. The optimizer for image classification task is Momentum, and Adam for sentiment analysis. Marsit-100 refers to the setting where local gradients operate full-precision synchronization every 100 communication rounds (i.e.,

[2]`https://pytorch.org/tutorials/intermediate/dist_tuto.html`

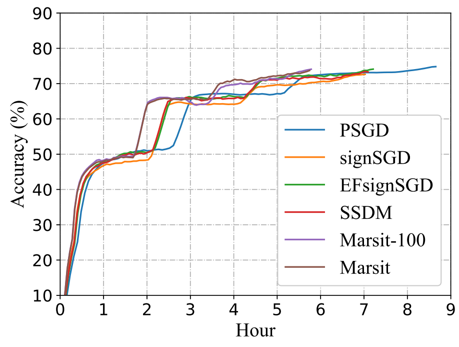
Model	Dataset	# parameters	Batch size	Top-1 Accuracy (%)					
				PSGD	signSGD	EF-signSGD	SSDM	Marsit-100	Marsit
AlexNet	CIFAR-10	23M	8192	82.38	80.74	*82.25*	81.89	**82.30**	81.58
ResNet-20	CIFAR-10	0.27M	8192	93.42	88.92	*91.85*	89.18	**92.18**	90.15
ResNet-18	ImageNet	11M	6144	69.18	67.17	68.14	68.10	**68.96**	*68.40*
ResNet-50	ImageNet	25M	6144	74.87	72.74	73.89	73.35	**74.35**	*74.10*
DistilBERT	IMDb review	67M	512	92.16	89.12	*90.57*	**91.41**	90.13	90.26

Table 3.2: Accuracy of existing works on different models training for different datasets.

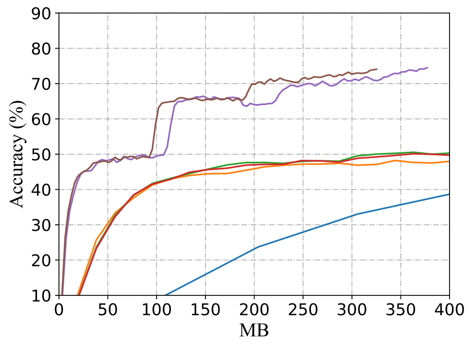
$K = 100$), while Marsit does not have full-precision synchronization. For ImageNet and CIFAR-10, the initial learning rate is set to 0.1 and 0.03, respectively, and decays by a factor of 10 every full-precision synchronization. For DistilBERT, we use a constant learning rate of 5e-5.

3.5 Numerical Results and Analysis

Performance Analysis Table 3.2 summarizes Top-1 accuracy of all test datasets. Compared to PSGD, the state-of-the-art compression approaches suffer from a noticeable accuracy drop in both image classification and sentiment analysis tasks. For instance, signSGD has up to a 5% decreasing. Moreover, in most cases, Marsit-100 and/or Marsit outperforms the existing approaches and achieves nearly the same final accuracy as PSGD. In CIFAR-10 training, Marsit with periodical full-precision synchronization (e.g., Marsit-100) has better performance than the one without full-precision synchronization, while they do not have distinct differences in both ImageNet and IMDb review datasets. As for the encoder-based transformer DistilBERT, it is noticed that our proposed method falls behind some other base-



(a) Accuracy w.r.t Time



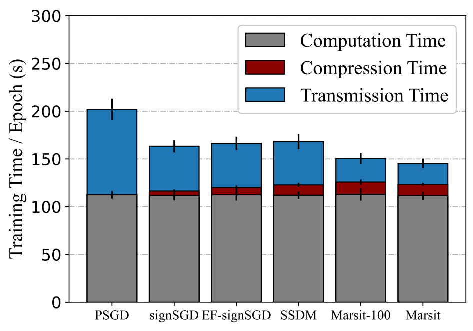
(b) Accuracy w.r.t Overhead

Figure 3.4: Experiments for training ResNet50 on ImageNet

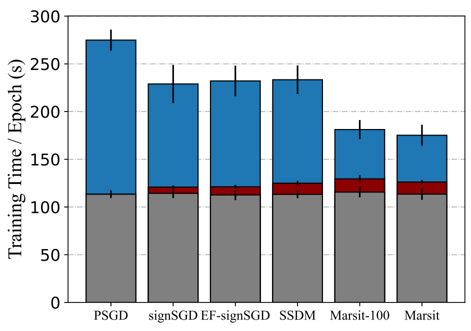
lines. A possible reason is that our proposed method limits the transmission to one bit causing a significant loss for a pretrained large language model, in comparison with those baselines that allow expandable transmission.

Figure 3.4a shows time-to-accuracy performance for ResNet-50 on ImageNet. Among these six approaches, non-compression approach, i.e., PSGD, takes a large amount of time, while Marsit achieves large speedups (1.5x) to reach a similar accuracy.

Communication Efficiency Marsit has a significant reduction in communication cost compared to the other five baselines. From Figure 3.4b, our algorithm requires 90% less communication budget, when compared to PSGD, and reduces communication cost by 70%, when compared to the existing signSGD approaches. In the mean time, with a smaller communication budget, our algorithm still preserves the same convergence rate as other baselines.



(a) TAR



(b) RAR

Figure 3.5: Experiments on training AlexNet for CIFAR-10 under TAR and RAR

In Figure 3.4b, given the same amount of communication overhead, Marsit and Marsit-100 always have higher accuracy than other baselines. Specifically, when Marsit and Marsit-100 reach convergence, other signSGD methods only attain accuracy around 50%.

Performance under Various MAR settings Figure 3.5 presents the results of Marsit and its baselines under RAR and TAR. For each method, we measure its average training time in each communication round and split the time into three phases, namely, computation (grey), compression (red) and communication (blue). We notice that Marsit introduces minor compression overheads to prepare for the real-time aggregation. Among these six approaches, it is clear that Marsit and/or Marsit-100 spends the least time in communication compared with other baselines. For TAR paradigm, each baseline takes less time to communicate. For RAR paradigm, the communication time dominates the computation time and Marsit requires less training time between two successive synchronizations.

3.6 Summary

This chapter proposes a synchronization framework, Marsit, that achieves one-bit transmission under multi-hop all-reduce. In this framework, we design a bit-wise operation to support the receiving and the compression undertake simultaneously. Besides, we introduce a global compensation mechanism to mitigate the compression deviation. Based on the structure, we offer a theoretical guarantee that it achieves the same convergence rate as the non-compression approach using the optimizer of SGD. Empirical studies present that our proposed approach can achieve a similar test accuracy to the non-compression version while using less training time by 35%.

Chapter 4

FedaGrac: A Gradient Calibration Approach

4.1 Introduction

Federated learning (FL) is thriving as a promising paradigm that refrains the leakage of users' data, including raw information and label distribution. With the rapid development of FL techniques over the past few years, a wide range of applications for computer vision (Liu et al., 2020c; Yu and Liu, 2019) and natural language processing (Chen et al., 2019; Liu et al., 2020a; Wu et al., 2020c) have deployed over a large set of edge devices (e.g., smartphones and tablets). Conventionally, clients perform a fixed number of local stochastic gradient descent (SGD) steps in each round; then, the server aggregates the updated models and finally acquires and distributes the global one to all clients (Li et al., 2019b; McMahan et al., 2017).

Table 4.1: The number of communication rounds to reach the test accuracy of 80% under Logistic Regression (LR) and 2-layer CNN on Fashion-MNIST with various settings when 10 devices participate in FedAvg. The number of local updates is 100 without step asynchronism, while under step asynchronism, clients perform at least 100 local updates. The learning rates set for LR and 2-layer CNN are 0.001 and 0.03, respectively, which are also applied to Table 4.2, Figure 4.2, and Figure 4.3.

FedAvg with	LR	2-layer CNN
neither	2	20
step async	1	8
non-i.i.d.	91	265
both	1K+	339

FedAvg follows the preceding procedure and has been proven to be a promising solution to data heterogeneity.

With an increasing number of nodes participating in the training, the traditional framework becomes infeasible because the computation capacities are substantially diverse among devices (Chai et al., 2019). A practical framework allows clients to update the local model via a flexible number of local SGD steps in each round according to its available resource capacity. And we define such a procedure as *step asynchronism* (see Figure 4.1 for visualized demonstration). To comprehensively understand the training performance of the traditional algorithm FedAvg, Table 4.1 compares the results in terms of test accuracy in two situations – step asynchronism and data heterogeneity. This experiment is under convex (i.e., logistic regression) and non-convex (i.e., 2-layer CNN) objectives using a public dataset Fashion-MNIST (Xiao et al., 2017). Performance deterioration is noticeable, especially in the logistic regression model the desired test accuracy cannot

Device	Method	Utilization[1] (Rounds)	Accuracy(%)[2]
Raspberry Pi 4	FedNova	25% (49)	66.18
	`FedaGrac`	**100% (25)**	**72.07**
Nvidia Jetson Nano	FedNova	50% (50)	64.39
	`FedaGrac`	**100% (21)**	**72.93**
Nvidia GTX 1080 Ti	FedNova	100% (40)	69.77
	`FedaGrac`	**100% (30)**	**72.00**
Nvidia GTX 2080 Ti	FedNova	**100% (29)**	72.11
	`FedaGrac`	100% (36)	**72.13**

[1]Utilization means the maximum computation capacity of Nvidia GTX 3080Ti to achieve the test accuracy of 60% in the first 50 rounds. We obtain the value by tuning the resource usage from 100% and looping a deduction of 5%. Rounds quantify when the approach achieves 60% test accuracy.
[2]Given the utilization, we measure the test accuracy after 100 rounds.

Table 4.2: Utilization and test accuracy under the setting that one Nvidia GTX 3080Ti and nine other devices are shown in the first column. Here we evaluate Fashion-MNIST with non-convex objectives (i.e., 2-layer CNN), and the data distribution among clients is heterogeneous.

be reached.

A previous study (Wang et al., 2020b) owes the performance deterioration to *objective inconsistency*, where the FL training converges to a stationary point that mismatches the optimal solution. In order to alleviate the issue, Wang et al. (Wang et al., 2020b) introduce FedNova, a normalization approach that averages the normalized local gradients and accordingly updates the global model at the server. However, in Table 4.2, we empirically disclose that FedNova cannot fully utilize the computational resources of the powerful node under heterogeneous environments, which explicitly limits the number of local updates for the faster node.

In this chapter, we propose a method named `FedaGrac` to conquer the objective inconsistency challenge under a highly imbalanced computational setting in FL. The core idea of our proposed algorithm is to calibrate each local update according to the global update orientation. Although the correct global direction is not known, it can be estimated based on the clients' local updates: If a client performs the local updates very fast, then the client will transmit the first gradient; otherwise, the averaged gradient. By this means, the negative effect of deviation on the convergence can be significantly mitigated. We conduct preliminary experiments and depict the comparison between our proposed algorithm and FedNova (Wang et al., 2020b) in terms of resource utilization and test accuracy in Table 4.2. In all cases, `FedaGrac` not only fully utilizes the computational resources, but also achieves a better accuracy than FedNova (see Table 4.2).

Our key contributions to this work are listed as follows:

1. To explore the factors that lead to performance deterioration, we analyze the convergence property under strongly-convex objectives. The theoretical result indicates that the expected loss never reaches the optimal one when both data heterogeneity and step asynchronism exist. In other words, a constant number of local updates eliminates the negative effect of data distribution differentiation, while step asynchronism magnifies the drawback of data heterogeneity.

2. We design a novel method named `FedaGrac` to address the problem of objective inconsistency via predictive gradient calibration, which makes the direction of each local update close to the direction towards the global opti-

mum. For the first time, our algorithm can jointly address statistical heterogeneity and computation heterogeneity at a time.

3. We establish the convergence rate of `FedaGrac`. Under non-convex objectives, the algorithm achieves a convergence rate of $O\left(1/\sqrt{MT\bar{K}}\right)$, where M and T represent the number of clients and communication rounds, respectively, and $\bar{K}$ indicates the weighted averaged number of local updates. This convergence rate is also achieved by FedNova only under the condition that $K_{\text{max}}/K_{\text{min}} = O(M)$, where K_{max} and K_{min} separately refer to the maximum and minimum number of local updates (Wang et al., 2020b). Otherwise, the actual convergence rate of FedNova should be $O\left(\sqrt{\bar{K}/MT}\right)$. Apparently, our algorithm can achieve a faster convergence rate by a factor up to $O(\bar{K})$.

4. We conduct extensive experiments to compare the proposed `FedaGrac` with typical and latest works such as SCAFFOLD (Karimireddy et al., 2020b) and FedNova (Wang et al., 2020b). In terms of convergence rate, `FedaGrac` achieves higher convergence efficiency compared to FedAvg and FedNova, especially in scenarios with high heterogeneity. For example, in terms of test accuracy, our algorithm can always preserve convergence while SCAFFOLD and FedNova cannot work in some cases.

The rest of this chapter is organized as follows. First, Section 4.2 provides related work and background knowledge of distributed SGD and existing solutions to heterogeneous training. Next, we state preliminaries and problem formulation for the heterogeneous Federated Learning in Section 4.3. Then, in Section 4.4, we design a novel algorithm `FedaGrac` to solve the problem. In Section 4.5, we analyze its convergence property. After that, we present our experimental results to evaluate our method in Section 4.6. Finally, Section 4.7 concludes the chapter.

4.2 Related Work

Federated learning. Frequently, edge devices such as smartphones possess abundant data, which are highly sensitive but useful to the model training (Guo and Qu, 2022; Han et al., 2020; Lim et al., 2021; Wang et al., 2021c). To utilize these data, FL is conceived to search for a generalized model (Qu et al., 2021; Wang et al., 2021b) or personalized models (T Dinh et al., 2020; Zhang et al., 2021) while safeguarding the data privacy (Konečnỳ et al., 2015; McMahan et al., 2017). Apparently, the data are heterogeneous among clients because there are no predefined rules for the data distribution for each client. Besides, due to the hardware differences among devices, the computational capabilities are various. In this section, we briefly investigate the flaws raised by data heterogeneity and computation heterogeneity and review the existing work to tackle these two issues.

Data Heterogeneity. Generally, in FL settings, the data distributed among clients are agnostic and therefore, each data portfolio has its exclusive optimal parameters. As a classical algorithm to combat data heterogeneity, FedAvg inherits the training features from local SGD (Stich, 2018; Yu et al., 2019d; Zhou and Cong, 2018), a framework that runs for multiple local updates prior to a global synchronization.

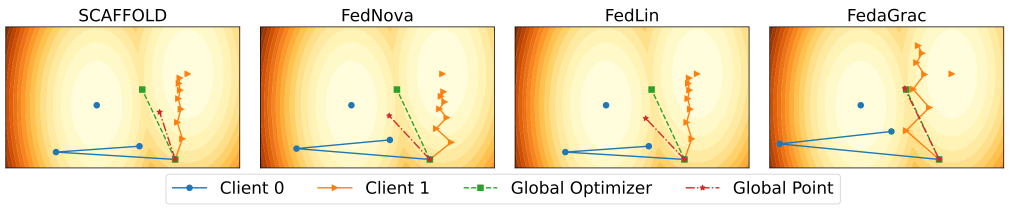


Figure 4.1: Illustration of model updating in the parameter space. For client i, we generate a set of (x, y)s fluctuated around a linear function $y = a_i x + b_i$, where a_i and b_i are real numbers. Our target is to find an optimal straight line $y = ax + b$, which is averagely close to all clients' data. Starting at the same point, each client applies mean squared error (MSE) loss and follows a predefined algorithm to update its local model so as to optimize the global one. Regarding the existence of data and computation heterogeneity, our proposed method does not deviate from the direction towards a global minimizer.

Obviously, this strategy significantly reduces the total communication overhead when compared to parallel SGD that synchronizes the gradient at every local update. Recent studies (Gu et al., 2021; Khaled et al., 2020; Li et al., 2019b) show that FedAvg can have a great performance from theoretical and empirical perspectives. Also, FedAvg can seamlessly adopt communication-efficient approaches such as quantization (Alistarh et al., 2017a; Basu et al., 2019) and sparsification (Stich et al., 2018; Wangni et al., 2018) to further reduce the cost of transmission (Wang et al., 2021a; Wu et al., 2020a,d; Zhou et al., 2021).

Nevertheless, numerous studies (Cheng et al., 2021; Gorbunov et al., 2020; Karimireddy et al., 2020b; Liu et al., 2020b; Zhao et al., 2018) theoretically prove that the issue raises the client-drift effect and degrades the convergence property. To mitigate the negative impact, existing solutions include cross-client variance reduction (Karimireddy et al., 2020b; Liang et al., 2019), client clustering sampling

(Fraboni et al., 2021; Ghosh et al., 2020; Murata and Suzuki, 2021) and reinforcement learning driven incentive mechanism (Wang et al., 2020a). Among these approaches, SCAFFOLD (Karimireddy et al., 2020b) is a superior option that adjusts every local update with the help of the global and a client's local reference orientation, such that every local update keeps close to the global direction. However, as shown in Figure 4.1, SCAFFOLD cannot completely remove the drift. A physical explanation for the result is that the local reference directions of the faster nodes with more number of local updates lead to a significant deviation from the orientation towards the local optimizer. Since the global reference direction is aggregated by clients' local ones, it is intuitively dominated by the faster nodes (see Figure 4.1), which betrays its origin intention. Although we use a similar design philosophy that ensures every local update along with the global orientation, the global orientation consists of the gradient that depends on the number of local updates, either the normalized gradient or the initial gradient.

Computational Heterogeneity. The computation capabilities vary among clients because they use different devices. To minimize the computation differences, some existing works adopt a client sampling strategy (Deng et al., 2021; Huang et al., 2020; Wu et al., 2020b; Zhou et al., 2020), where only a small portion of clients transmit the gradients to the server. Compared to the case that requires full-worker participation, this scheme reduces the total training time. However, there still exists resource underutilization as the fastest client should wait for others' completion.

A practical solution is to adopt step asynchronism, where each client performs an inconsistent number of local updates. Although FedAvg with step asynchronism can converge to a stable point under non-convex objectives (Yu et al., 2019d), Wang et al. (Wang et al., 2020b) point out that objective inconsistency takes place under quadratic function. To alleviate the challenge of computational heterogeneity, effective approaches are constituted with normalization-based approach FedNova (Wang et al., 2020b) and FedLin (Mitra et al., 2021), regularization-based approach FedProx (Li et al., 2020) and architecture-based approach HeteroFL (Diao et al., 2020). Gradient normalization is the most ubiquitous framework that overcomes step asynchronism under non-i.i.d. data setting. However, this method cannot prevent the negative impact of statistical heterogeneity on the convergence rate because the update deviation still exists after averaging. Figure 4.1 compares FedNova (Wang et al., 2020b) and FedLin (Mitra et al., 2021) with our proposed method, and we notice that the global model deviates to the one with less updates in FedNova (Wang et al., 2020b). The reason is obvious: clients update the models bias to their local datasets such that the normalized gradients collected by the server are sparse. Besides, with the local models approaching the local minimizers, the update becomes so trivial that those clients with more local updates have a dispensable influence on the global model update.

4.3 Preliminary and Problem Formulation

Formally, the learning problem can be represented as the following distributed optimization problem across M FL clients:

$$\min_{\mathbf{x} \in \mathbb{R}^d} F(\mathbf{x}) = \sum_{i=1}^{M} \omega_i F_i(\mathbf{x}), \tag{4.1}$$

where the weight $\omega_i = |\mathcal{D}_i|/|\mathcal{D}|$ is the ratio between the size of local dataset $\mathcal{D}_i$ and overall dataset $\mathcal{D} \triangleq \cup_{i=1}^{M} \mathcal{D}_i$, and $F_i(\mathbf{x}) \triangleq \mathbb{E}_{\varepsilon_i \sim \mathcal{D}_i}[f_i(\mathbf{x}; \varepsilon_i)]$ is the the local objective, i.e., the expected loss value of model x with respect to random sampling ε_i for client i.

FedAvg with step asynchronism. Naive weighted aggregation (Li et al., 2019b; McMahan et al., 2017; Stich, 2018; Yu et al., 2019d) is an effective and communication-efficient way to solve Problem (4.1) for both convex and non-convex objectives. With the increasing number of edge devices participating in model training, the framework is neither economic nor fair to require all clients to run a certain number of local updates. Instead, a practical approach is that client $i \in \{1, ..., M\}$ runs for a flexible number of SGD steps (i.e., K_i) according to its resource capability before the model aggregation at the server:

- **(Pull):** Pulls the current parameter $\mathbf{x}_0$ from the server.
- **(Compute):** Samples a realization ε randomly from the local dataset $\mathcal{D}_i$ and compute the gradient $\nabla f_i(\mathbf{x}_k, \varepsilon)$.
- **(Update):** Performs k-th local update of the form η by $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \mathbf{g}_k$,

where $k \in \{0, ..., K_i\}$ and η is the stepsize.

- **(Push):** Pushes the local parameter $\mathbf{x}_{K_i}$ to the server.

Under this framework, we let K_{max} and K_{min} separately be the maximum and the minimum number of local updates among all clients, i.e., $K_{\text{max}} = \max_{i \in \{1,...,M\}} K_i$ and $K_{\text{min}} = \min_{i \in \{1,...,M\}} K_i$. In addition, $\bar{K} = \sum_{i=1}^{M} \omega_i K_i$ is defined as the weighted averaged number of local updates. Formally, step asynchronism is defined as the following mathematical expression:

$$\exists i, j \in \{1, ..., M\}, \quad K_i \neq K_j. \tag{4.2}$$

Therefore, $K_{\text{max}} \neq K_{\text{min}}$ when step asynchronism exists. Without extra explanations, these notations are adopted throughout the chapter.

Assumptions. To establish the convergence theory of the FL optimization, we make the following assumptions that are adapted in previous works (Li et al., 2020, 2019b; Reddi et al., 2020; Wang et al., 2019, 2020b):

Assumption 4.1 (L-smooth). *The local objective functions are Lipschitz smooth: For all $v, \bar{v} \in \mathbb{R}^d$,*

$$\|\nabla F_i(v) - \nabla F_i(\bar{v})\|_2 \leq L\|v - \bar{v}\|_2, \quad \forall i \in \{1, ..., M\}.$$

Assumption 4.2 (μ-strongly convex). *The local objective functions are μ-strongly* *convex with the value of $\mu > 0$: For all $v, \bar{v} \in \mathbb{R}^d$,*

$$F_i(v) - F_i(\bar{v}) \geq \langle \nabla F(\bar{v}), (v - \bar{v}) \rangle + \frac{\mu}{2} \|v - \bar{v}\|_2^2, \quad \forall i \in \{1, ..., M\}$$

where $\langle \cdot, \cdot \rangle$ refers to the inner product of two gradients.

Assumption 4.3 (Bounded Variance)**.** *For all $v \in \mathbb{R}^d$, there exists a scalar $\sigma \geq 0$ such that*

$$\mathbb{E}\|\nabla f_i(v, \varepsilon) - \nabla F_i(v)\|_2^2 \leq \sigma^2, \quad \forall i \in \{1, ..., M\}$$

Assumption 4.4 (Bounded Dissimilarity)**.** *For some $v \in \mathbb{R}^d$ that $\|\nabla F(v)\|_2^2 > 0$ holds, there exists a scalar $B \geq 1$ such that*

$$\mathbb{E}\|\nabla F_i(v)\|_2^2 \leq B^2 \|\nabla F(v)\|_2^2, \quad \forall i \in \{1, ..., M\}.$$

Obviously, when the data are independent and identically distributed, the value of B should be 1.

Assumption 4.4 seems to be a little bit strong as $\|\nabla F(v)\|_2^2$ cannot be 0. However, considering ϵ-accuracy as the learning criterion, i.e., $\|\nabla F(v)\|_2^2 \leq \epsilon$ under non-convex objectives such as deep neural networks which possess multiple local minimizers, the value of ϵ cannot strictly be 0. In other words, there exists $\epsilon_1 \leq \epsilon$ such that $\|\nabla F(v)\|_2^2 \geq \epsilon_1$ for all v always holds.

Key factor that raises objective inconsistency. Although (Wang et al., 2020b) indicates that objective inconsistency occurs when using FedAvg with step asynchronism under quadratic functions, the factor that makes it happen remains a mystery. To explore in depth, the following theorem analyzes FedAvg with step asynchronism under a strongly-convex objective.

Theorem 4.1. *Suppose the local objective functions are non-negative. Denote the parameter at t-th communication round by $\mathbf{x}_t$. Let T be the total number of communication rounds. Under Assumption 4.1, 4.2, 4.3 and 4.4, by setting the learning rate $\eta = \mathcal{O}(1/\mu LT\bar{K}) \leq 1/L\bar{K}$, the output of FedAvg with step asynchronism satisfies*

$$\lim_{T\to\infty} \mathbb{E}[F(\mathbf{x}_T)] - F(\mathbf{x}_*) \leq O\left(\sum_{i=1}^{M} \omega_i \left(\frac{K_i}{K_{\min}} - 1\right) F_i(\mathbf{x}_*)\right) \tag{4.3}$$

where $\mathbf{x}_1$ and $\mathbf{x}_*$ indicates the initial and optimal model parameters, respectively.

Proof. See Appendix C for details. □

Remark The theoretical result in Equation 4.3 is consistent with the result of FedAvg analysis in (Karimireddy et al., 2020b) as the number of local updates is identical, i.e., $K_i = \bar{K}, \forall i \in \{1, ..., M\}$. Besides, when the data are identical and independent distributed among clients, where the global optimizer is not equivalent to the clients' local minimizer, we can easily induce that $\mathbf{x}_T$ is close to $\mathbf{x}_*$ when $T \to \infty$. The conclusion holds regardless of the number of local updates. However, when data heterogeneity and step asynchronism coexist, the right-hand side of Equation (4.3) is non-zero. As a result, when T tends to be infinite, the model cannot converge to the optimal parameters, which can explain the result manifested in Table 4.1 under LR. Based on the theoretical discovery, we can draw a conclusion that step asynchronism leads to a significant accuracy drop in the non-i.i.d. cases, which impedes normal training.

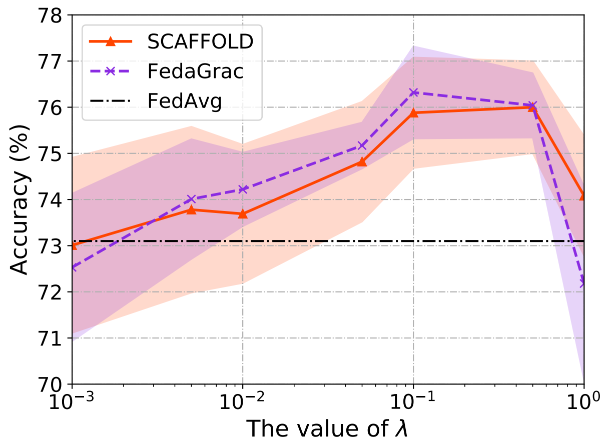
4.4 `FedaGrac` algorithm

To ensure that $\mathbb{E}[F(\mathbf{x}_T)] - F(\mathbf{x}_*)$ is close to 0 when $T \to \infty$, we target to remove the constant term in the right-hand side of Equation (4.3). Based on the remark in Section 3.2, a practical approach is to minimize the effectiveness of data heterogeneity. In this section, we elaborate our proposed algorithm, Federated Accelerating Gradient Calibration (`FedaGrac`), to avoid the objective inconsistency as well as enhance the convergence performance when step asynchronism is adopted to improve the resource utilization. The implementation details are presented as Algorithm 3.

At first, apart from the hyperparameters such as learning rate η and calibration rate λ, we initialize a d-dimension model with arbitrary parameters $\mathbf{x}_1$. Besides, to ease the theoretical analysis in Section 4.5, we set $\nu^{(i)}$ as $\nabla f_i(\mathbf{x}_1, \mathcal{D}_i)$ for all $i \in \{1, ..., M\}$. Then, we define ν as:

$$\nu = \sum_{i=1}^{M} \omega_i \nu^{(i)} = \sum_{i=1}^{M} \omega_i \nabla f_i(\mathbf{x}_1, \mathcal{D}_i).$$

In this algorithm, client i performs the local updates for K_i times in parallel. During each local update, clients calibrate the local client deviation with reference to the global reference orientation, which is estimated at every global synchroniza-



(a) w/ step async

λ	Acc. (%)
0	77.37±0.8
0.001	77.23±1.2
0.005	77.18±0.9
0.01	77.49±0.6
0.05	78.23±0.5
0.1	78.26±0.3
0.5	77.03±0.7
1	77.29±0.9
Increase	**79.21±0.4**

(b) w/o step async

Figure 4.2: Preliminary test for a 2-layer CNN with the recognition of Fashion-MNIST. In the line graph, the experiments are conducted when the clients run inconsistent updates. "Increase" in the table shows the value of λ changes over time, i.e., 0.1 for the first 50 rounds, 0.5 for the next 100 rounds, and 1 for the rest.

tion. In the following two subsections, we separately discuss the effectiveness of two main components, namely,

- **Calibrating the local client deviation** (Line 9 in Algorithm 3) migrates the data heterogeneity;
- **Estimating the global reference orientation** (Line 14 in Algorithm 3) accelerates the training process.

4.4.1 Calibrating the local client deviation

As a classical approach, FedAvg updates the parameters using stochastic gradient descent (SGD), where the gradient is computed in accordance with Line 8.

Algorithm 3: Federated Accelerating Gradient Calibration (`FedaGrac`)

```
Require: Initialize $M$ clients, set the initial model to be $\tilde{\mathbf{x}}_1 \in \mathbb{R}^d$. Set $\nu^{(i)}$ and $\nu$
    for all clients $i \in \{1, \cdots, M\}$. Set learning rate $\eta > 0$, calibration rate
    $\lambda > 0$, the number of global synchronizations $T$ and the number of local
    iterations of each client $K_i$ for all clients $i \in \{1, \cdots, M\}$.
1:  On client $i \in \{1, \cdots, M\}$:
2:  for $t = 1$ to $T$ do
3:      Pull $\tilde{\mathbf{x}}_t, \nu$ from server
4:      Set $\mathbf{x}_{t;0}^{(i)} = \tilde{\mathbf{x}}_t$
5:      Set $\mathbf{c} = \nu - \nu^{(i)}$
6:      for $k = 0$ to $K_i - 1$ do
7:          Randomly sample a realization $\varepsilon_k^{(i)}$ from $\mathcal{D}_i$
8:          $\mathbf{g}_{t;k}^{(i)} = \nabla f_i(\mathbf{x}_{t;k}^{(i)}, \varepsilon_k^{(i)})$
9:          $\mathbf{x}_{t;k+1}^{(i)} = \mathbf{x}_{t;k}^{(i)} - \eta(\mathbf{g}_{t;k}^{(i)} + \lambda\mathbf{c})$
10:     end for
11:     Set $\nu^{(i)} = \frac{1}{K_i}\sum_{k=0}^{K_i-1} g_{t;k}^{(i)}$
12:     Push $\mathbf{x}_{t;K_i}^{(i)}, K_i$ to the server
13:     Receive $\bar{K}$ from the server
14:     if $K_i \le \bar{K}$, then send $\nu^{(i)}$; else send $g_{t;0}^{(i)}$
15: end for
16: On server:
17: for $t = 1$ to $T$ do
18:     Push $\tilde{\mathbf{x}}_t, \nu$ to clients
19:     Pull $\mathbf{x}_{t;K_i}^{(i)}, K_i$ from client $i \in [1, \ldots, M]$
20:     $\tilde{\mathbf{x}}_{t+1} = \sum_{i=1}^{M} \omega_i \mathbf{x}_{t;K_i}^{(i)}$
21:     $\bar{K} = \sum_{i=1}^{M} \omega_i K_i$
22:     Push $\bar{K}$ to clients and receive $v_{\text{transit}}^{(i)}$ from clients
23:     $\nu = \sum_{i=1}^{M} \omega_i v_{\text{transit}}^{(i)}$
24: end for
```

Suppose the gradient is equivalent to the first order derivative of the true local objective, i.e., $\nabla f_i(v, \varepsilon) = \nabla F_i(v)$, where ε is randomly sampled from the local dataset $\mathcal{D}_i$. Then, given the stepsize η, the model update follows $\mathbf{x}_{t,k+1}^{(i)} =$

$\mathbf{x}_{t,k}^{(i)} - \eta \nabla F_i \left(\mathbf{x}_{t,k}^{(i)}\right)$. Previous works (Bottou, 2010; Zinkevich et al., 2010) show that this scheme can converge to a stable point when a client performs sufficient local updates. Under a heterogeneous data setting, the points vary among clients because each of them is determined by the local data distribution. Therefore, clients are biased from the global orientation, and the phenomenon is named as client deviation[1].

Existing works to overcome client deviation mainly focus on the variance reduction approach, i.e., SCAFFOLD (Karimireddy et al., 2020b). It is somehow similar to our proposed algorithm when λ is set to 1. In this case, client i's local reference direction $v^{(i)}$ is assumed to be equivalent to the vector from the current point to its local optimizer. As for the global reference orientation, v overlaps with the gradient from the current point to the global minimizer. However, it is nearly impossible to coincide with the case, especially when applied with the gradient calibration technique. Generally speaking, using an obsolete gradient to predict the coming gradient is not reasonable because the aggregated direction presumably deviates from the expected one.

Therefore, we introduce a calibration rate λ for the correction term. With this hyperparameter, a gradient can be adjusted and approximated to the global update. Empirical results in Figure 4.2 intuitively present the effectiveness of λ. Generally speaking, a smaller λ has a similar performance as FedAvg because the calibrated gradient is still biased to the local computed one. For a greater λ, the test accuracy goes down dramatically since the gradient is over-calibrated. As a result, a

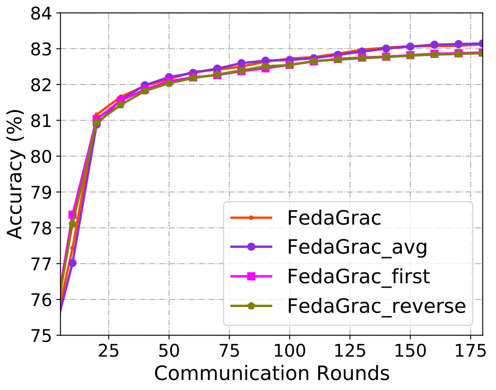
[1]Client deviation is also known as client drift (Karimireddy et al., 2020b; Mitra et al., 2021).

constant λ cannot be too large or too small such that the calibration term is effective. Furthermore, in Figure 4.2b, we evaluate a case where λ increases over time. Apparently, the strategy is impressive because it outperforms all other constant settings. The reason for the improvement is clear: at the beginning stage, the difference between two successive updates is significant because the model is far away from convergence. When the training comes to a stable point, the value of λ should be 1 such that the gradient eliminates the deviation towards the local minimizer.

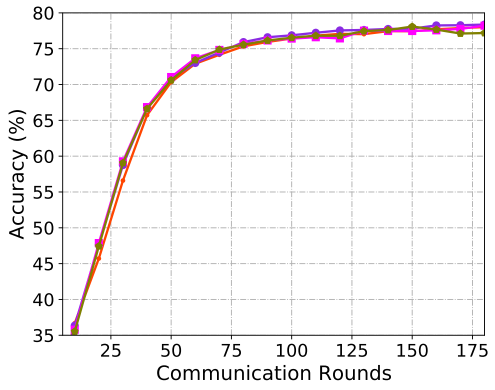
4.4.2 Estimating the global reference orientation

While applying SCAFFOLD (Karimireddy et al., 2020b) to train a model, we notice that the model update is biased to the fastest node under step-asynchronous settings. Given a model $\mathbf{x}$, some clients, e.g., client i, are close to a stable point such that the computed local reference orientation significantly deviates from the expected one, i.e., $\nabla F_i(\mathbf{x})$. Regarding that the clients (client i) with fewer local updates can better estimate the local orientation $\nabla F_i(\mathbf{x})$, the model prefers those with more local updates, which undermines the convergence property.

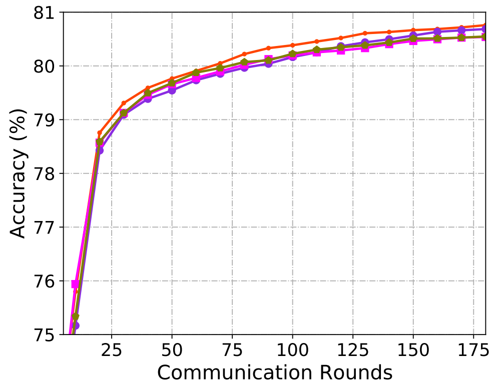
At the beginning of round $t \in \{1, ..., T\}$, the centralized server broadcasts the model $\tilde{\mathbf{x}}_t$ to all clients. To obtain an exact result of $\nabla F(\tilde{\mathbf{x}}_t)$, each client $i \in \{1, ..., M\}$ should provide an accurate estimation for $\nabla F_i(\tilde{\mathbf{x}}_t)$, or the bias of the estimation $\nu^{(i)}$ can be eliminated by the sum, i.e., $\sum_{i=1}^{M} \omega_i \nu^{(i)}$. Therefore, there are two practical ways to estimate $\nabla F_i(\tilde{\mathbf{x}}_t)$ for client i, namely, (i) the first stochastic gradient, i.e., $\nabla f_i(\tilde{\mathbf{x}}_t, \varepsilon)$, and (ii) the averaged stochastic gradient, i.e.,



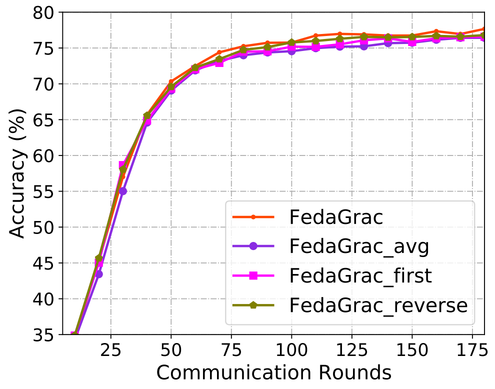
(a) LR w/o step async



(b) CNN w/o step async



(c) LR w/ step async



(d) CNN w/ step async

Figure 4.3: Empirical evaluation for how to estimate the global reference orientation using Fashion-MNIST with convex (i.e., LR) and non-convex objectives (i.e., 2-layer CNN). The horizontal axis indicates the communication rounds, and the vertical axis shows the test accuracy in percentage. (a)(b) indicate the results when the clients run for the constant number of updates, and (c)(d) is when they perform various numbers of SGD steps. (Zoom in for the best view)

$\frac{1}{K_i}\sum_{k=0}^{K_i-1}\nabla f_i\left(\mathbf{x}_{t,k}^{(i)}, \varepsilon_k^{(i)}\right)$ in Line 11 of Algorithm 3. Based on these two strategies, we design and empirically evaluate four different schemes to find a proper estimation for the global reference orientation: (Note: faster or slower nodes are classified by whether the number of local updates is greater than the average updates)

- **FedaGrac** requires faster nodes to transmit the first stochastic gradient while the rest push the average one;
- **FedaGrac_avg** (a.k.a. SCAFFOLD) requires all nodes to transmit the average stochastic gradient;
- **FedaGrac_first** requires all nodes to transmit the first stochastic gradient;
- **FedaGrac_reverse** requires faster nodes to transmit the average stochastic gradient while the rest push the first one.

Figure 4.3 presents the results of different strategies. As we can see, without step asynchronism, these four schemes do not have considerable differences. However, with step asynchronism, `FedaGrac` outperforms another three potential approaches under both convex and non-convex objectives. This is why Line 14 of Algorithm 3 is introduced. To further reduce the communication overhead, the algorithm solely requests the faster nodes to upload the first stochastic gradient, while the rest can be computed via $\frac{1}{\eta K_i}\left(\tilde{\mathbf{x}}_t - \mathbf{x}_{t,K_i}^{(i)}\right) - \lambda\left(\nu - \nu^{(i)}\right)$ if $\nu^{(i)}$ is preserved on the server.

4.5 Theoretical Convergence Analysis

In this section, we analyze the convergence property of `FedaGrac` under both non-convex objectives and strongly-convex objectives for solving Problem (4.1). The details of the mathematical proof are provided in the supplementary materials with step-by-step explanations.

4.5.1 Mathematical expression for Algorithm 3

In Section 4.4, we describe the details in Algorithm 3. Below represents how to derive the recursive function step by step.

Local reference orientation. To ensure every local update can calibrate to the expected one, we should use the averaged local update such that after multiple local updates, the acquired model does not deviate from the expected orientation. Therefore, the local reference orientation is defined as:

$$\nu^{(i)} = \begin{cases} \frac{1}{K_i} \sum_{k=0}^{K_i-1} g_{t-1;k}^{(i)}, & K_i \leq \bar{K} \\ g_{t-1;0}^{(i)}, & \text{Otherwise} \end{cases} \tag{4.4}$$

Global reference orientation. SCAFFOLD (Karimireddy et al., 2020b) presents a remarkable performance with the aggregation of $\nu^{(i)}$ for all $i \in \{1, ..., M\}$. However, the approach presumably does not work due to step asynchronism, where local reference orientations deviated from the expected direction are dramatically various among clients. To avoid this issue, we let the faster node with more number of local updates transfer the initial gradient while others send the local reference orientation to the server, which can be formally written as:

$$\nu = \sum_{i, K_i \leq \bar{K}} \frac{\omega_i}{K_i} \sum_{k=0}^{K_i-1} g_{t-1;k}^{(i)} + \sum_{i, K_i > \bar{K}} \omega_i g_{t-1;0}^{(i)}$$

Recursion function. According to Line 9 in Algorithm 3, for client i, the recursion between two successive local updates can be presented as:

$$\mathbf{x}_{t,k+1}^{(i)} = \mathbf{x}_{t,k}^{(i)} - \eta \left[g_{t;k}^{(i)} + \lambda \left(\nu - \nu^{(i)} \right) \right] \tag{4.5}$$

Then, based on the equation above, i.e., Equation (4.5), for client i with the local updates of K_i, $\mathbf{x}_{t,K_i}^{(i)} - \tilde{\mathbf{x}}_t$ can be formulated in mathematical expression as:

$$\mathbf{x}_{t,K_i}^{(i)} - \tilde{\mathbf{x}}_t = \sum_{k=0}^{K_i-1} \left(\mathbf{x}_{t,k+1}^{(i)} - \mathbf{x}_{t,k}^{(i)} \right) = -\eta \sum_{k=0}^{K_i-1} g_{t;k}^{(i)} - \eta \lambda K_i \left(\nu - \nu^{(i)} \right)$$

Finally, according to the definition in Problem (4.1), the recursion function between two successive global updates is the weighted average of all clients' models, which is written as:

$$\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t = \sum_{i=1}^{M} \omega_i \mathbf{x}_{t,K_i}^{(i)} - \tilde{\mathbf{x}}_t = -\eta \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i g_{t;k}^{(i)} - \eta \lambda \bar{K} \nu + \eta \lambda \sum_{i=1}^{M} \omega_i K_i \nu^{(i)}$$

4.5.2 Non-convex objectives

Theorem 4.2 (Non-convex objectives)**.** *Considering the same* $\mathbf{x}_1$ *and* $\mathbf{x}_*$ *as Theorem 4.1, under Assumption 4.1, 4.3 and 4.4, by setting* $\eta = \mathcal{O}\left(\sqrt{\frac{M}{T\bar{K}}}\right)$*, the convergence rate of Algorithm 1 with step asynchronism for non-convex objectives is*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E} \| \nabla F(\tilde{\mathbf{x}}_t) \|_2^2 \leq \mathcal{O}\left(\frac{(F(\mathbf{x}_1) - F(\mathbf{x}_*))}{\lambda \sqrt{\bar{K} M T}} \right) + \mathcal{O}\left(\frac{\sigma^2 L \sqrt{M}}{\lambda \sqrt{\bar{K}^3 T}} \sum_{i=1}^{M} \omega_i^2 K_i \right)$$

$$+\mathcal{O}\left(\frac{\sigma^2 L\lambda\sqrt{M}}{\sqrt{\bar{K}T}}\sum_{i=1}^{M}\omega_i^2\left(\frac{(\bar{K}-K_i)^2}{\bar{K}K_i}+1\right)\right)$$
$$+\mathcal{O}\left(\frac{L^2\sigma^2 M}{\lambda\bar{K}^2T}\sum_{i=1}^{M}\omega_i K_i^4\right)$$
$$+\mathcal{O}\left(\frac{L^2\sigma^2\lambda M}{\bar{K}^2T}\sum_{i=1}^{M}\omega_i K_i^3\left(K_i\sum_{j=1}^{M}\frac{\omega_j^2}{K_j}+1\right)\right). \quad (4.6)$$

Proof. See Appendix D for details. □

Corollary 4.2.1. *By setting $\omega_1 = ... = \omega_M = 1/M$ and $\lambda = \mathcal{O}(1)$, the following inequality holds under Theorem 4.2:*

$$\min_{t\in\{1,...,T\}} \mathbb{E}\|\nabla F(\tilde{\mathbf{x}}_t)\|_2^2 \leq \mathcal{O}\left(\frac{1}{\sqrt{MT\bar{K}}}\right). \quad (4.7)$$

Remark (Wang et al., 2020b) states that FedaNova can achieve the convergence rate same as Equation 4.7, but there exists an explicit condition that $\sum_{i=1}^{M}(\bar{K}/MK_i)$ is a constant when $\omega_1 = ... = \omega_M = 1/M$. Let us consider an extreme case that the slow nodes locally update once, i.e., $K_i = 1$ for all $i \in \{1, ..., M-1\}$ while Client M can run for a very large number of times. This case is possible, for instance, a system consists of multiple Raspberry Pi and a single Nvidia GTX 3080Ti GPU, the computational difference between which can be up to a thousandfold. Under such situation, the aforementioned term should be bounded by $\mathcal{O}(\bar{K})$ instead of $\mathcal{O}(1)$ and therefore, the convergence rate for FedNova should be $\mathcal{O}(\sqrt{\bar{K}/MT})$. In comparison with Equation 4.7, `FedaGrac` achieves an increment up to $\mathcal{O}(\bar{K})$.

Furthermore, the algorithms such as FedAvg (Yu et al., 2019d) and SCAF-

FOLD (Karimireddy et al., 2020b) that use the homogeneous setting achieve a convergence rate of $\mathcal{O}(1/\sqrt{MTK_{\min}})$. Obviously, `FedaGrac` admits better convergence rate as $K_{\min} \leq \bar{K}$ always holds under heterogeneous computational resources. This is because our algorithm can fully utilize the computational resources from all participants such that it outperforms those algorithms that solely supports the homogeneous environment.

4.5.3 Strongly-convex objectives

Theorem 4.3 (Strongly-convex objectives)**.** *Considering the same* $\mathbf{x}_1$ *and* $\mathbf{x}_*$ *as Theorem 4.1, under Assumption 4.1, 4.2 and 4.3, by setting* $\lambda = 1$, $\eta = \mathcal{O}(1/\mu LT\bar{K}) \leq 1/L\bar{K}$*, the convergence rate of Algorithm 1 with step asynchronism for strongly-convex objectives is*

$$\mathbb{E}[F(\tilde{\mathbf{x}}_T)] - F(\mathbf{x}_*) \leq \tilde{\mathcal{O}}\left(\mu\|\mathbf{x}_1 - \mathbf{x}_*\|_2^2 \exp\left(-\frac{\mu T}{L}\right) + \frac{\mathcal{H}}{\mu T} + \frac{\mathcal{P}}{\mu^2 T^2}\right), \quad (4.8)$$

where

$$\mathcal{H} = \frac{\sigma^2}{\bar{K}^2}\sum_{i=1}^{M}\omega_i^2\left(K_i + \bar{K} + \frac{(\bar{K} - K_i)^2}{K_i}\right), \mathcal{P} = \frac{L^2\sigma^2}{\mu}\left(\sum_{i=1}^{M}\omega_i K_i^3\right)\sum_{j=1}^{M}\frac{\omega_j^2}{K_j}.$$

Proof. See Appendix E for details. □

Corollary 4.3.1. *By setting* $\omega_1 = \ldots = \omega_M = 1/M$*, the following inequality holds* *under Theorem 4.3:*

$$\mathbb{E}[F(\tilde{\mathbf{x}}_T)] - F(\mathbf{x}_*) \leq \tilde{\mathcal{O}}\left(\frac{\sigma^2}{\mu M T \bar{K}}\right). \tag{4.9}$$

Remark Compared to FedNova (Wang et al., 2020b) that has convergence theory only for non-convex objectives, we have established the rigorous convergence theory for our method `FedaGrac` on strongly-convex objectives. Compared with Theorem 4.1, `FedaGrac` not only converges to the optimal parameters, but also obtains a better convergence rate as $\tilde{\mathcal{O}}(1/\bar{K}) \leq \tilde{\mathcal{O}}(1/K_{\text{min}})$.

4.6 Empirical Evaluation

In this section, we conduct extensive experiments to evaluate the performance of `FedaGrac` in the real cases that are widely accepted by the existing studies. To further obtain an intuitive understanding of the numerical results, `FedaGrac` competes against other up-to-date benchmarks that are comparable under various settings. The code is implemented with PyTorch and available at `https://github.com/HarliWu/FedaGrac`.

4.6.1 Setup

Datasets. We leverage Fashion-MNIST (Xiao et al., 2017) to run the preliminary experiments in the previous sections. This dataset comprises 60000 28×28 greyscale training images and 10000 test images, which can be categorized into ten

Layer	Output Shape	Trainable Parameters	Activation	Hyperparameters
Input	(1,28,28)	0		
Conv2d	(10, 24, 24)	260	ReLU	kernel size=5
MaxPool2d	(10, 12, 12)	0		kernel size=2
Conv2d	(20, 8, 8)	5020	ReLU	kernel size=5
Dropout2d	(20, 8, 8)	0		p=0.5
MaxPool2d	(20, 4, 4)	0		kernel size=2
Flatten	320	0		
Dense	50	16050	ReLU	
Dropout	50	0		p=0.5
Dense	10	510	softmax	

Table 4.3: Details for 2-layer CNN on Fashion-MNIST. Typically, Fashion-MNIST consists of grey-scale images possessing a single channel.

classes related to the clothes type. In this section, we utilize two more datasets: a9a[2] and CIFAR-10 (Krizhevsky et al., 2009b). As a binary classification task, a9a consists of 32561 training samples and 16281 test samples, and each sample possesses 123 features. CIFAR-10 is a 10-category image classification task, constituting 60000 32×32 color images divided into the training and test set with the size of 50000 and 10000, respectively.

Models. For the assessment of convex objectives, we train a logistic regression (LR) model using a9a. In addition, we investigate the performance under non-convex objectives through an image classification task CIFAR-10 (Krizhevsky et al., 2009b) with AlexNet (Krizhevsky et al., 2012) and VGG-19 (Simonyan and Zisserman, 2014), deep neural networks with total parameters of 7.21M and 20.55M, respectively. As for Fashion-MNIST, 2-layer CNN and LR are utilized to evaluate the performance under non-convex and convex objectives, respectively. Based on the dataset used, the details for 2-layer CNN, AlexNet and VGG-19 are

[2]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Layer	Output Shape	Trainable Parameters	Activation	Hyperparameters
Input	(3,32,32)	0		
Conv2d	(64, 8, 8)	23296	ReLU	kernel size=11, stride=4, padding=5
MaxPool2d	(64, 4, 4)	0		kernel size=2, stride=2
Conv2d	(192, 4, 4)	307392	ReLU	kernel size=5, padding=2
MaxPool2d	(192, 2, 2)	0		kernel size=2, stride=2
Conv2d	(384, 2, 2)	663936	ReLU	kernel size=3, padding=1
Conv2d	(256, 2, 2)	884992	ReLU	kernel size=3, padding=1
Conv2d	(256, 2, 2)	590080	ReLU	kernel size=3, padding=1
MaxPool2d	(256, 1, 1)	0		kernel size=2, stride=2
Flatten	256	0		
Dropout	256	0		p = 0.5
Dense	2048	526336	ReLU	
Dropout	2048	0		p = 0.5
Dense	2048	4196352	ReLU	
Dense	10	20490	softmax	

Table 4.4: Details for AlexNet on CIFAR-10. Output shape follows the format of (channel, height, width). Generally, color images like CIFAR-10 dataset are with three channels.

separately described in Table 4.3, Table 4.4 and Table 4.5.

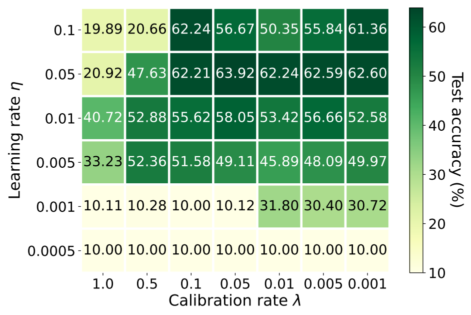
Data Heterogeneity. As for the non-i.i.d. settings, we adopt two different partitioned ways. The first one that we split the dataset across the clients follows the Dirichlet distribution with parameter 0.3, denoted as DP1. This approach is suitable for both datasets. The other method disjoints the dataset via sharding, and thus each client holds 5 classes. We let such a method be DP2 and ensure clients carry the same volume of data. It is worth noting that this partition is only compatible with CIFAR-10 because a9a is a binary classification challenge.

Computational Heterogeneity. To simulate a heterogeneous computing environment, we suppose the computation differences among workers follow the Gaussian distribution. Then, the number of local updates varies among clients and follows the normal distribution with predefined mean and variance. And the number of local updates may change over time for each client.

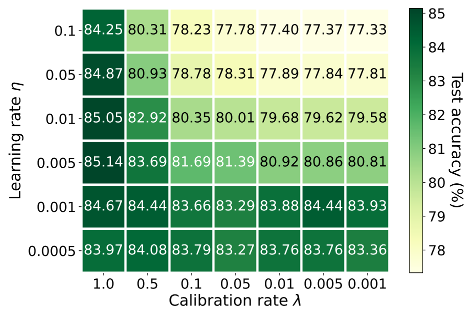
Layer	Output Shape	Trainable Parameters	Activation	Hyperparameters
Input	(3,32,32)	0		
2 $\times$ Conv2d	(64, 32, 32)	38720	ReLU	kernel size=3; padding=1
MaxPool2d	(64, 16, 16)	0		kernel size=2, stride=2
2 $\times$ Conv2d	(128, 16, 16)	221440	ReLU	kernel size=3; padding=1
MaxPool2d	(128, 8, 8)	0		kernel size=2, stride=2
4 $\times$ Conv2d	(256, 8, 8)	2065408	ReLU	kernel size=3; padding=1
MaxPool2d	(256, 4, 4)	0		kernel size=2, stride=2
4 $\times$ Conv2d	(512, 4, 4)	8259584	ReLU	kernel size=3; padding=1
MaxPool2d	(512, 2, 2)	0		kernel size=2, stride=2
4 $\times$ Conv2d	(512, 2, 2)	9439232	ReLU	kernel size=3; padding=1
MaxPool2d	(512, 1, 1)	0		kernel size=2, stride=2
Flatten	512	0		
Dropout	512	0		p = 0.5
Dense	512	262656	ReLU	
Dropout	512	0		p = 0.5
Dense	512	262656	ReLU	
Dense	10	5130	softmax	

Table 4.5: Network architecture for VGG-19 on CIFAR-10.

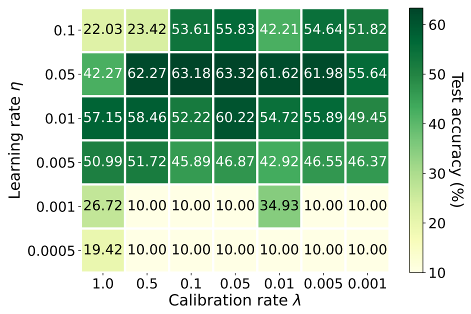
Implementation and Hyperparameter Settings. The experiments are conducted with an MPI-supported cluster with the configurations of 100GB RAM, 25 CPU cores, and 1 Nvidia P100 GPU. Based on the resource, we utilize 20 cores to act as clients and a single core as the federated server. Besides, the batch sizes throughout our experiments are set as 25 and 20 for CIFAR-10 and a9a, respectively. We choose FedAvg (McMahan et al., 2017), FedNova (Wang et al., 2020b), SCAFFOLD (Karimireddy et al., 2020b) and FedProx (Li et al., 2020) as benchmarks and present the effectiveness of our proposed approach `FedaGrac`. For a fair comparison, we compare these algorithms with the results when they achieve the best performance under the constant learning rates $\{0.01, 0.008, 0.005\}$ and $\{0.005, 0.001, 0.0005\}$ for AlexNet/VGG-19 and LR, respectively. And other required hyperparameters are also carefully picked from a set, such as the coefficient of the regularization term for FedProx in $\{1, 0.1, 0.01\}$. We specified other unmen-



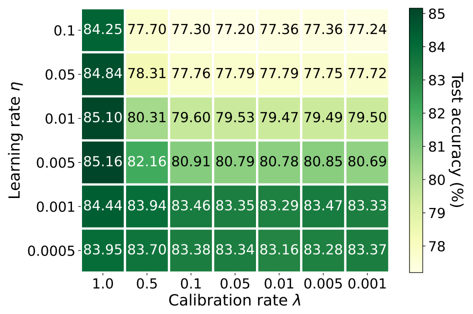
(a) AlexNet w/ step async



(b) LR w/ step async



(c) AlexNet w/o step async



(d) LR w/o step async

Figure 4.4: Comparison of various setting combinations for learning rate η and calibration rate λ using DP1 data distribution under AlexNet and LR after 100 communication rounds. The horizontal index indicates the value of λ while the vertical index shows the value of η. The numeric in the box presents the averaged test accuracy of the last 10 rounds under the specific hyperparameter settings. The mean number of local updates is 500, and the variance with step asynchronism is 10000.

tioned but necessary settings in the captions of the figures and the tables.

4.6.2 Numerical Results

Performance under Various combinations for learning rate and calibration rate. As learning rate η and calibration rate λ need tuning in `FedaGrac`, we first explore how to set both hyperparameters scientifically. Figure 4.4 depicts the test accuracy under various relations between η and λ. As we observe, the differences regarding the convexity are quite significant, e.g., AlexNet in Figure 4.4a and LR in Figure 4.4b, while the computation heterogeneity has minor influence on the selection of hyperparameters under the same model, e.g., AlexNet in Figure 4.4a and Figure 4.4c. Based on the acquired results, we discuss how to set the hyperparameters for `FedaGrac` under convex or non-convex objectives.

Both Figure 4.4a and Figure 4.4c illustrate the performance under AlexNet with and without computational heterogeneity. In both cases, most λs achieve the highest accuracy at $\eta = 0.05$, while some have the best performance at $\eta = 0.01$. When the learning rate initializes with a value smaller or equal to 0.001, most AlexNets seem untrained after 100 rounds because they are less likely to escape a saddle point. Although some portfolios successfully get out of the minima, they still cannot outperform the aforementioned settings because they may (i) trap into a non-optimal stable point or (ii) need a longer period to reach the optimal solution. A constant λ that performs well in all learning rates does not exist. However, when we shrink the choice of learning rate between 0.01 and 0.05, $\lambda = 0.05$ has a remarkable performance. In our experiments, the calibration rate is chosen from $\{0.01, ..., 0.05\}$ depending on the algorithm's performance.

Figure 4.4b and Figure 4.4d present the results under the convex objectives. Regardless of the step asynchronism, $\lambda = 1$ always has remarkable performance for any learning rate. And it is noticeable that `FedaGrac` can obtain the best performance when $\lambda = 1$ and $\eta = 0.005$. As for a $\lambda \neq 1$, `FedaGrac` can achieve bet-

Model	Data Distribution	Target Accuracy	Variance	Mode	Number of communication rounds (↓)				
					`FedaGrac`	FedAvg	FedNova	SCAFFOLD	FedProx
AlexNet	DP1	68%	V = 0	-	**113**	123	127	**113**	145
			V = 100	fixed	**106**	130	147	114	144
				random	**116**	140	154	133	140
			V = 10000	fixed	**126**	156	172	141	142
				random	**121**	177	170	136	152
AlexNet	DP2	70%	V = 0	-	152	183	186	160	**147**
			V = 100	fixed	**111**	179	119	124	143
				random	**112**	200+	195	137	141
			V = 10000	fixed	**111**	200+	113	131	145
				random	**118**	200+	200+	123	152
VGG-19	DP2	80%	V = 0	-	73	83	79	**72**	90
			V = 100	fixed	73	75	72	**66**	72
				random	**73**	85	74	78	102
			V = 10000	fixed	77	73	**70**	72	77
				random	**71**	85	76	72	99

Table 4.6: The number of communication rounds when first achieving the target test accuracy under AlexNet and VGG-19. The computational capabilities among workers follow the Gaussian distribution with a mean of 500 and different variances (i.e., V = 0, V = 100, and V = 10000) using two different data distributions (i.e., DP1 and DP2). Random mode indicates the number of local updates on a client varies among communication rounds, while fixed mode does not possess the feature. Each experiment runs for a maximum of 200 rounds.

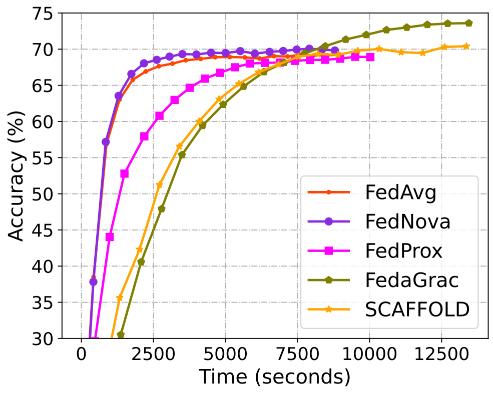
ter performance as the learning rate becomes smaller. With such a phenomenon, we hypothesize that `FedaGrac` cannot exactly reach the identical minimizer when $\lambda \neq 1$ and approaches the expected point as the learning rate reduces.

Performance under various data distributions. Table 4.6 validates our algorithm under different data heterogeneities, i.e., DP1 and DP2 under AlexNet. The target accuracy is determined by the best performance that these five algorithms can achieve when they run a constant number of updates. By comparing each algorithm under these two data distributions, DP2 is more challenging for FedAvg and FedNova because the algorithms generally require more communication rounds to achieve the target. Even worse, these two algorithms cannot achieve the goal within 200 rounds in some DP2 settings. As for the regularization-based approach (i.e., FedProx) and the variance reduction approaches (i.e., `FedaGrac` and SCAFFOLD), the task shifting does not cause a distinct influence[3] in terms of the required communication rounds. As we can see in both cases with computational differences, `FedaGrac` demonstrates the superiority over other benchmarks.

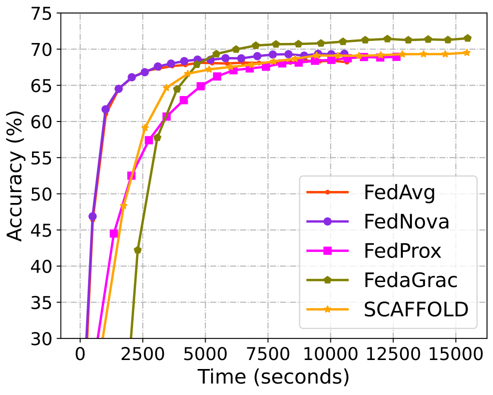
Performance under various neural networks. In addition to exploring various data distributions based on Table 4.6, we investigate the performance of `FedaGrac` under different neural networks. As we notice, the approach in VGG-19 does not outperform all benchmarks in some computation heterogeneity cases. Specifically, it requires several more rounds than the best algorithm. An explanation for this phenomenon is that obtaining an 80%-accuracy VGG-19 on CIFAR-10 is not a difficult task. In contrast to getting an AlexNet with a test accuracy of 70%, the algorithms can adopt a greater learning rate to improve training efficiency. Since there are some restricted terms in `FedaGrac`, it is reasonable that our proposed algorithm cannot outperform the benchmarks. Meanwhile, it is common that some benchmarks cannot outperform FedAvg (Li et al., 2021a). However, it is worth noting that, as presented in Figure 4.5, the faster algorithm may not surpass the slower ones in terms of the final test accuracy.

Performance under various computational capabilities. While adopting Gaussian distribution to tune the computation heterogeneity, we should manually set both mean and variance. To explore whether these two hyperparameters influence the algorithms' performances, we conduct extensive experiments, and the relevant results are presented in Table 4.6 and Figure 4.5. Table 4.6 evaluates the

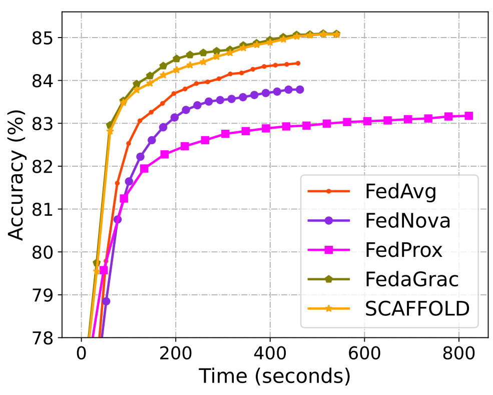
[3]The difference between the numbers of the communication rounds is less than 15%.



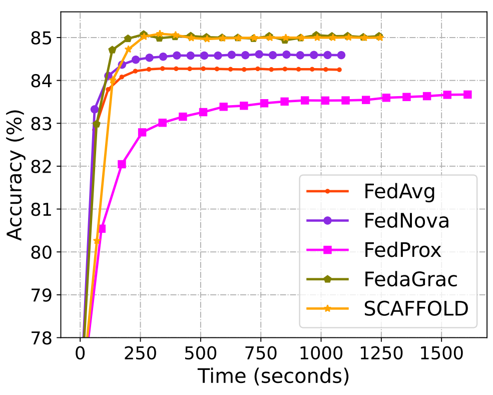
(a) AlexNet with 500



(b) AlexNet with 1000



(c) LR with 500



(d) LR with 1000

Figure 4.5: Test accuracy v.s. Training time under different gaussian mean w.r.t. the fixed variance of 10000. Leftmost two figures: AlexNet using DP2; Rightmost two figures: LR using DP1. The number of local updates on a client is fixed and initialized at the beginning of the model training. Each algorithm runs for a total of 200 rounds, and the gap between two markers represents an interval of 10 communication rounds.

performance under the computational capabilities with a constant mean of 500 and different variances, while Figure 4.5 assesses the convergence tendency under a fixed variance of 10000 and diverse means.

Table 4.6 presents the results given different variances with/without time-varying local updates. Our analysis is mainly based on AlexNet because it gives noticeable differences when the variances or the modes switch. Admittedly, these algorithms are not sensitive to whether the number of local updates is time-varying. However, it is worth noting that FedNova is vulnerable to time-varying settings in DP2. This is because a large learning rate may lead FedNova to a surrogate solution (Mitra et al., 2021) such that FedNova has to adopt a smaller learning rate to achieve the target accuracy. In contrast to time-varying local updates, the variance plays an important role in training efficiency. When the variance becomes larger, it is likely that the algorithms require more communication rounds. Nevertheless, a greater variance sometimes improves the training efficiency of those algorithms which mitigate the client-drift effect, i.e., `FedaGrac`, SCAFFOLD, and FedProx.

Figure 4.5 illustrates the entire training progress, i.e., the test accuracy with respect to the training time and the communication rounds under both convex and non-convex objectives. Our proposed algorithm achieves competitive accuracy compared to other baselines, despite a slow start likely taking place because the calibration is yet to settle the client-drift effects properly in the beginning. Although FedAvg and FedNova require half communication overhead as our proposed algorithm does, they cannot keep dominant alongside the training. Use AlexNet as an example (Figure 4.5a and 4.5b), and `FedaGrac` is capable of achieving the same performance with fewer rounds. In addition, it is interesting to see FedProx consuming more time to implement 200 rounds than FedAvg. A reasonable explanation for this phenomenon is that extra computation is required by the regularization terms. As the model gets larger, this effect becomes minor since the communication consumption asymptotically occupies most training time (compare between LR (Figure 4.5c) and AlexNet (Figure 4.5a) for this heuristic conclusion). As a convex objective, LR depicts the issue of objective inconsistency (the latter two plots in Figure 4.5). The performances of FedAvg, FedNova, and FedProx are much worse than `FedaGrac` and SCAFFOLD. With the increasing mean and the unchanged variance, the deterioration gets mitigation but cannot eliminate. As for the comparison between SCAFFOLD and our proposed method, the latter possesses dominance nearly all the time.

4.7 Summary

This chapter introduces a new algorithm named `FedaGrac` to tackle the challenges of both statistical heterogeneity and computation heterogeneity in FL. By calibrating the local client deviations according to an estimated global orientation in each communication round, the negative effect of step asynchronism on model accuracy can be greatly mitigated, and the training process is remarkably accelerated. We establish the theoretical convergence rate of `FedaGrac`. The results imply that `FedaGrac` admits a faster convergence rate and has a better tolerance to computation heterogeneity than the state-of-the-art approachs. Extensive experiments are also conducted to validate the advantages of `FedaGrac`.

Chapter 5

Conclusion and Future Research

5.1 Conclusion

As an active field of distributed machine learning, optimization problems have been studied for decades, but they are still looking for more efficient algorithms. This thesis studies the optimization problems in distributed machine learning from the gradient-wise perspectives.

Firstly, for the sake of gradient compression to reduce communication overhead, we propose Marsit, a synchronization system that achieves one-bit transmission under multi-hop all-reduce. We implement a bit-wise operation in this framework to facilitate simultaneous reception and compression and prevent cascading compression. In addition, we build a global compensation mechanism to reduce compression deviations. Theoretically, the proposed framework retains the same theoretical convergence rate as non-compression mechanisms. According to empirical investigations, our proposed approach can achieve equal test accuracy to the non-compression version while requiring 35% less training time.

Furthermore, by means of gradient calibration to accelerate the model training, we introduce FedaGrac, a new algorithm designed to jointly address the issues of statistical and computational heterogeneity in FL. With the proposed algorithm, clients correct the gradients based on an estimated global orientation in every local update. As a result, the detrimental effect of step asynchronism on model accuracy can be considerably minimized, and the training efficiency can be dramatically improved. According to the theoretical findings, our proposed algorithm admits a faster convergence rate and is more tolerant of computational heterogeneity than the current state-of-the-art technique. Extensive empirical studies are also carried out to prove the benefits of FedaGrac.

5.2 Future Work

In the future, we plan to conduct the research in the following three directions:

- **Arbitrary Device Unavailability.** The methods in this thesis mainly focus on the setting of full client participation. With the proliferation of edge devices in the FL system, the number of inactive nodes or stragglers inflates, resulting in arbitrary device unavailability. Although they can directly apply to partial worker scenarios where the server actively samples a constant number of clients, it is unknown whether they are able to retain the convergence property under arbitrary device unavailability. In future work, we will design a new approach for this case and avoid the past knowledge preservation because the obsoleted information is likely harmful to the model updates.

- **Second-Order Guarantee.** In this thesis, we provide proof of the first-order guarantee under non-convex objectives. While the first-order guarantee is the most common practice in distributed machine learning, its stationary point could be a saddle point under non-convex objectives. This has been identified as a challenging problem in deep learning optimization. To escape the saddle points, we should devise an algorithm that enjoys second-order optimality without hurting the communication efficiency.

- **Gradient Decomposition.** In addition to the aforementioned gradient-wise approaches, gradient decomposition is a meaningful way to optimize distributed machine learning. It not only reduces the communication overhead but distinguishes the significant elements in model updates. In the future, we will explore how to robustly decompose a gradient such that the training efficiency could be greatly improved.

Bibliography

Harsh Agrawal, Clint Solomon Mathialagan, Yash Goyal, Neelima Chavali, Prakriti Banik, Akrit Mohapatra, Ahmed Osman, and Dhruv Batra. Cloudcv: Large-scale distributed computer vision as a cloud service. In *Mobile cloud visual media computing*, pages 265–290. Springer, 2015.

Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in Neural Information Processing Systems*, 30, 2017a.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017b.

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. *Advances in Neural Information Processing Systems*, 31, 2018.

Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *International conference on machine learning*, pages 699–707. PMLR, 2016.

Salem Alqahtani and Murat Demirbas. Performance analysis and comparison of distributed machine learning systems. *arXiv preprint arXiv:1909.02061*, 2019.

Dmitrii Avdiukhin and Shiva Kasiviswanathan. Federated learning under arbitrary communication patterns. In *Proceedings of the 38th International Conference on Machine Learning*, pages 425–435. PMLR, 2021.

Arda Aytekin, Hamid Reza Feyzmahdavian, and Mikael Johansson. Analysis and implementation of an asynchronous optimization algorithm for the parameter server. *arXiv preprint arXiv:1610.05507*, 2016.

Baidu-Research. tensorflow-allreduce. [Source Code]. `https://github.com/baidu-research/tensorflow-allreduce`, 2017.

Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, pages 14695–14706, 2019.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR, 2018a.

Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd with majority vote is communication efficient and fault tolerant. In *International Conference on Learning Representations*, 2018b.

Alberto Bietti and Julien Mairal. Stochastic optimization with variance reduction for infinite datasets with finite sum structure. *Advances in Neural Information Processing Systems*, 30, 2017.

Avleen S Bijral, Anand D Sarwate, and Nathan Srebro. On data dependence in distributed stochastic optimization. *arXiv preprint arXiv:1603.04379*, 2016.

Avrim Blum, Nika Haghtalab, Richard Lanas Phillips, and Han Shao. One for one, or all for all: Equilibria and optimality of collaboration in federated learning. *arXiv preprint arXiv:2103.03228*, 2021.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

Léon Bottou and Yann Cun. Large scale online learning. *Advances in neural information processing systems*, 16, 2003.

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

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Víctor Campos, Francesc Sastre, Maurici Yagües, Míriam Bellver, Xavier Giró-i Nieto, and Jordi Torres. Distributed training strategies for a computer vision deep learning algorithm on a distributed gpu cluster. *Procedia Computer Science*, 108:315–324, 2017.

Zheng Chai, Hannan Fayyaz, Zeshan Fayyaz, Ali Anwar, Yi Zhou, Nathalie Baracaldo, Heiko Ludwig, and Yue Cheng. Towards taming the resource and data heterogeneity in federated learning. In *2019 {USENIX} Conference on Operational Machine Learning (OpML 19)*, pages 19–21, 2019.

Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. Adacomp: Adaptive residual gradient compression for data-parallel distributed training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018a.

Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. Adacomp: Adaptive residual gradient compression for data-parallel distributed training. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018b.

Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril

Allauzen, Françoise Beaufays, and Michael Riley. Federated learning of n-gram language models. *arXiv preprint arXiv:1910.03432*, 2019.

Yiming Chen, Kun Yuan, Yingya Zhang, Pan Pan, Yinghui Xu, and Wotao Yin. Accelerating gossip sgd with periodic global averaging. In *International Conference on Machine Learning*, 2021.

Daning Cheng, Shigang Li, Hanping Zhang, Fen Xia, and Yunquan Zhang. Why dataset properties bound the scalability of parallel machine learning training algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1702–1712, 2021.

Wei Dai, Yi Zhou, Nanqing Dong, Hao Zhang, and Eric P Xing. Toward understanding the impact of staleness in distributed machine learning. *arXiv preprint arXiv:1810.03264*, 2018.

Christopher M De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the wild: A unified analysis of hogwild-style algorithms. *Advances in neural information processing systems*, 28, 2015.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research*, 13:165–202, 2012.

Yongheng Deng, Feng Lyu, Ju Ren, Huaqing Wu, Yuezhi Zhou, Yaoxue Zhang, and Xuemin Shen. Auction: Automated and quality-aware client selection framework for efficient federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(8):1996–2009, 2021.

Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

Aymeric Dieuleveut and Kumar Kshitij Patel. Communication trade-offs for local-sgd with large step size. *Advances in Neural Information Processing Systems*, 32, 2019.

Peter Elias. Universal codeword sets and representations of the integers. *IEEE transactions on information theory*, 21(2):194–203, 1975.

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in Neural Information Processing Systems*, 31, 2018.

Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. *arXiv preprint arXiv:2105.05883*, 2021.

Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1):267–305, 2016.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of sgd: Variance reduction, sampling, quantization and coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, pages 680–690. PMLR, 2020.

Eduard Gorbunov, Konstantin P Burlachenko, Zhize Li, and Peter Richtárik. Marina: Faster non-convex distributed learning with compression. In *International Conference on Machine Learning*, pages 3788–3798. PMLR, 2021a.

Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. Local sgd: Unified theory and new efficient methods. In *International Conference on Artificial Intelligence and Statistics*, pages 3556–3564. PMLR, 2021b.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Xinran Gu, Kaixuan Huang, Jingzhao Zhang, and Longbo Huang. Fast federated learning in the presence of arbitrary device unavailability. *arXiv preprint arXiv:2106.04159*, 2021.

Jinrong Guo, Songlin Hu, Wang Wang, Chunrong Yao, Jizhong Han, Ruixuan Li, and Yijun Lu. Tail: an automated and lightweight gradient compression framework for distributed deep learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

Song Guo and Zhihao Qu. *Edge Learning for Distributed Big Data Analytics: Theory, Algorithms, and System Design*. Cambridge University Press, 2022.

Yuanxiong Guo, Ying Sun, Rui Hu, and Yanmin Gong. Hybrid local sgd for federated learning with heterogeneous communications. In *International Conference on Learning Representations*, 2021.

Farzin Haddadpour and Mehrdad Mahdavi. On the convergence of local descent methods in federated learning. *arXiv preprint arXiv:1910.14425*, 2019.

Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck Cadambe. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. *Advances in Neural Information Processing Systems*, 32, 2019.

Rui Han, Shilin Li, Xiangwei Wang, Chi Harold Liu, Gaofeng Xin, and Lydia Y

Chen. Accelerating gossip-based deep learning in heterogeneous edge computing platforms. *IEEE Transactions on Parallel and Distributed Systems*, 32(7): 1591–1602, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

Samuel Horváth and Peter Richtárik. Nonconvex variance reduced optimization with arbitrary sampling. In *International Conference on Machine Learning*, pages 2781–2789. PMLR, 2019.

Samuel Horváth, Lihua Lei, Peter Richtárik, and Michael I Jordan. Adaptivity of stochastic gradient methods for nonconvex optimization. *arXiv preprint arXiv:2002.05359*, 2020.

Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. Gaia:{Geo-Distributed} machine learning approaching {LAN} speeds. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 629–647, 2017.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Tiansheng Huang, Weiwei Lin, Wentai Wu, Ligang He, Keqin Li, and Albert Y Zomaya. An efficiency-boosting client selection scheme for federated learning with fairness guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1552–1564, 2020.

Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *arXiv preprint arXiv:1807.11205*, 2018.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, and et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020a.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020b.

Ahmed Khaled and Peter Richtárik. Better theory for sgd in the nonconvex world. *arXiv preprint arXiv:2002.03329*, 2020.

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 4519–4529. PMLR, 2020.

Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009a.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009b.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *Mathematical programming*, 171(1):167–215, 2018a.

Guanghui Lan and Yi Zhou. Random gradient extrapolation for distributed and stochastic optimization. *SIAM Journal on Optimization*, 28(4):2753–2782, 2018b.

Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. Non-convex finite-sum optimization via scsg methods. *Advances in Neural Information Processing Systems*, 30, 2017.

Hao Li, Asim Kadav, Erik Kruus, and Cristian Ungureanu. Malt: distributed data-parallelism for existing ml applications. In *European Conference on Computer Systems*, pages 1–16, 2015.

Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, pages 583–598, 2014a.

Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, pages 583–598, 2014b.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smithy. Feddane: A federated newton-type method. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1227–1231. IEEE, 2019a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019b.

Zhize Li. Ssrgd: Simple stochastic recursive gradient descent for escaping saddle points. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhize Li, Hongyan Bao, Xiangliang Zhang, and Peter Richtárik. Page: A simple and optimal probabilistic gradient estimator for nonconvex optimization. In *International Conference on Machine Learning*, pages 6286–6295. PMLR, 2021b.

Zhize Li, Slavomír Hanzely, and Peter Richtárik. Zerosarah: Efficient nonconvex finite-sum optimization with zero full gradient computation. *arXiv preprint arXiv:2103.01447*, 2021c.

Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. *Advances in Neural Information Processing Systems*, 28, 2015.

Xiangru Lian, Mengdi Wang, and Ji Liu. Finite-sum composition optimization via variance reduced gradient descent. In *Artificial Intelligence and Statistics*, pages 1159–1167. PMLR, 2017a.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017b.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1710.06952*, 2018.

Xianfeng Liang, Shuheng Shen, Jingchang Liu, Zhen Pan, Enhong Chen, and Yifei Cheng. Variance reduced local sgd with lower communication complexity. *arXiv preprint arXiv:1912.12844*, 2019.

Wei Yang Bryan Lim, Jer Shyuan Ng, Zehui Xiong, Jiangming Jin, Yang Zhang, Dusit Niyato, Cyril Leung, and Chunyan Miao. Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(3):536–550, 2021.

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*, 2021a.

Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local sgd. In *International Conference on Learning Representations*, 2019.

Tao Lin, Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi. Quasi-global momentum: Accelerating decentralized deep learning on heterogeneous data. In *International Conference on Machine Learning*, 2021b.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

Fenglin Liu, Xian Wu, Shen Ge, Wei Fan, and Yuexian Zou. Federated learning for vision-and-language grounding problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11572–11579, 2020a.

Sijia Liu, Pin-Yu Chen, Xiangyi Chen, and Mingyi Hong. signsgd via zeroth-order oracle. In *International Conference on Learning Representations*, 2018.

Wei Liu, Li Chen, Yunfei Chen, and Wenyi Zhang. Accelerating federated learning via momentum gradient descent. *IEEE Transactions on Parallel and Distributed Systems*, 31(8):1754–1766, 2020b.

Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. Fedvision: An online visual object detection platform powered by federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13172–13179, 2020c.

Yucheng Lu and Christopher De Sa. Optimal complexity in decentralized training. In *International Conference on Machine Learning*, pages 7111–7123. PMLR, 2021.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. *Advances in neural information processing systems*, 12, 1999.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

Hiroaki Mikami, Hisahiro Suganuma, Pongsakorn U-chupala, Yoshiki Tanaka, and Yuichi Kageyama. Massively distributed sgd: Imagenet/resnet-50 training in a flash. *arXiv preprint arXiv:1811.05233*, 2018.

Aritra Mitra, Rayana Jaafar, George Pappas, and Hamed Hassani. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. *Advances in Neural Information Processing Systems*, 34, 2021.

Tomoya Murata and Taiji Suzuki. Bias-variance reduced local sgd for less heterogeneous federated learning. *arXiv preprint arXiv:2102.03198*, 2021.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pages 2613–2621. PMLR, 2017.

Cyprien Noel and Simon Osindero. Dogwild!-distributed hogwild for cpu & gpu. In *NIPS Workshop on Distributed Machine Learning and Matrix Computations*, pages 693–701, 2014.

Pitch Patarasuk and Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2): 117–124, 2009.

Francisco Pérez-Hernández, Siham Tabik, Alberto Lamas, Roberto Olmos, Hamido Fujita, and Francisco Herrera. Object detection binary classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance. *Knowledge-Based Systems*, 194:105590, 2020.

Zhihao Qu, Song Guo, Haozhao Wang, Baoliu Ye, Yi Wang, Albert Zomaya, and Bin Tang. Partial synchronization to accelerate federated learning over relay-assisted edge networks. *IEEE Transactions on Mobile Computing*, pages 1–1, 2021. doi: 10.1109/TMC.2021.3083154.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems*, 24, 2011.

Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323. PMLR, 2016.

Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2020.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Nicolas Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. *Advances in neural information processing systems*, 25, 2012.

Khushi Roy, Subhra Debdas, Sayantan Kundu, Shalini Chouhan, Shivangi Mohanty, and Biswarup Biswas. Application of natural language processing in healthcare. *Computational Intelligence and Healthcare Informatics*, pages 393–407, 2021.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Mher Safaryan and Peter Richtárik. Stochastic sign descent methods: New algorithms and better theory. In *International Conference on Machine Learning*, pages 9224–9234. PMLR, 2021.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

Christopher J Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20:1–49, 2019.

Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008. PMLR, 2014.

Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *Proceedings of the 38th International Conference on Machine Learning*, pages 9489–9502. PMLR, 2021.

Shaohuai Shi, Kaiyong Zhao, Qiang Wang, Zhenheng Tang, and Xiaowen Chu. A convergence analysis of distributed sgd with communication-efficient gradient sparsification. In *IJCAI*, pages 3411–3417, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Alexander Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *VLDB Endowment*, 3(1-2):703–710, 2010.

Sebastian U Stich. Local sgd converges fast and communicates little. In *International Conference on Learning Representations*, 2018.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *arXiv preprint arXiv:1809.07599*, 2018.

Ananda Theertha Suresh, Felix X Yu, Sanjiv Kumar, and H Brendan McMahan. Distributed mean estimation with limited communication. In *International Conference on Machine Learning*. JMLR. org, 2017.

Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

Martin Takác, Avleen Bijral, Peter Richtárik, and Nati Srebro. Mini-batch primal and dual methods for svms. In *International Conference on Machine Learning*, pages 1022–1030. PMLR, 2013.

Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. Communication compression for decentralized training. In *Advances in Neural Information Processing Systems*, pages 7652–7662, 2018.

Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In

International Conference on Machine Learning, pages 10118–10129. PMLR, 2021.

Roberto Tron and René Vidal. Distributed computer vision algorithms through distributed averaging. In *CVPR 2011*, pages 57–63. IEEE, 2011.

Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. A survey on distributed machine learning. *ACM Comput. Surv.*, 53(2), March 2020. ISSN 0360-0300. doi: 10.1145/3377454. URL `https://doi.org/10.1145/3377454`.

Thijs Vogels, Sai Praneeth Karinireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32(CONF), 2019.

Xinchen Wan, Hong Zhang, Hao Wang, Shuihai Hu, Junxue Zhang, and Kai Chen. Rat-resilient allreduce tree for distributed machine learning. In *4th Asia-Pacific Workshop on Networking*, pages 52–57, 2020.

Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1698–1707. IEEE, 2020a.

Haozhao Wang, Song Guo, and Ruixuan Li. Osp: Overlapping computation and communication in parameter server for fast machine learning. In *Proceedings of the 48th International Conference on Parallel Processing*, pages 1–10, 2019.

Haozhao Wang, Song Guo, Zhihao Qu, Ruixuan Li, and Ziming Liu. Error-compensated sparsification for communication-efficient decentralized training in edge environment. *IEEE Transactions on Parallel and Distributed Systems*, 2021a.

Haozhao Wang, Zhihao Qu, Song Guo, Ningqi Wang, Ruixuan Li, and Weihua Zhuang. Losp: Overlap synchronization parallel with local compensation for fast distributed training. *IEEE Journal on Selected Areas in Communications*, 39(8):2541–2557, 2021b. doi: 10.1109/JSAC.2021.3087272.

Haozhao Wang, Zhihao Qu, Qihua Zhou, Haobo Zhang, Boyuan Luo, Wenchao Xu, Song Guo, and Ruixuan Li. A comprehensive survey on training acceleration for large machine learning models in iots. *IEEE Internet of Things Journal*, 2021c.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems*, 33, 2020b.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. A novel framework for the analysis and design of heterogeneous federated learning. *IEEE Transactions on Signal Processing*, 69:5234–5249, 2021d.

Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. Spiderboost: A class of faster variance-reduced algorithms for nonconvex optimization. *arXiv*, 2018, 2018.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pages 1299–1309, 2018.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems*, pages 1509–1519, 2017.

Feijie Wu, Shiqi He, Yutong Yang, Haozhao Wang, Zhihao Qu, Song Guo, and Weihua Zhuang. On the convergence of quantized parallel restarted sgd for central server free distributed training. *arXiv e-prints*, pages arXiv–2004, 2020a.

Feijie Wu, Song Guo, Haozhao Wang, Zhihao Qu, Haobo Zhang, Jie Zhang, and Ziming Liu. From deterioration to acceleration: A calibration approach to rehabilitating step asynchronism in federated optimization, 2021. URL `https://arxiv.org/abs/2112.09355`.

Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *International Conference on Machine Learning*, pages 5325–5333, 2018.

Wentai Wu, Ligang He, Weiwei Lin, and Rui Mao. Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1539–1551, 2020b.

Xing Wu, Zhaowang Liang, and Jianjia Wang. Fedmed: A federated learning framework for language modeling. *Sensors*, 20(14):4048, 2020c.

Xueyu Wu, Xin Yao, and Cho-Li Wang. Fedscr: Structure-based communication reduction for federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1565–1577, 2020d.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Haibo Yang, Minghong Fang, and Jia Liu. Achieving linear speedup with partial worker participation in non-iid federated learning. In *International Conference on Learning Representations*, 2020.

Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. Imagenet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*, pages 1–10, 2018.

Chen Yu, Hanlin Tang, Cedric Renggli, Simon Kassing, Ankit Singla, Dan Alistarh, Ce Zhang, and Ji Liu. Distributed learning over unreliable networks. In *International Conference on Machine Learning*, pages 7202–7212. PMLR, 2019a.

Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Conference on Machine Learning*, pages 7184–7193, 2019b.

Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Conference on Machine Learning*, pages 7184–7193. PMLR, 2019c.

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, pages 5693–5700, 2019d.

Mingchao Yu, Zhifeng Lin, Krishna Narra, Songze Li, Youjie Li, Nam Sung Kim, Alexander Schwing, Murali Annavaram, and Salman Avestimehr. Gradiveq: Vector quantization for bandwidth-efficient gradient aggregation in distributed cnn training. In *Advances in Neural Information Processing Systems*, pages 5123–5133, 2018.

Peihua Yu and Yunfeng Liu. Federated object detection: Optimizing object detection model with federated learning. In *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, pages 1–6, 2019.

Honglin Yuan and Tengyu Ma. Federated accelerated stochastic gradient descent. *arXiv preprint arXiv:2006.08950*, 2020.

Zhuoning Yuan, Zhishuai Guo, Yi Xu, Yiming Ying, and Tianbao Yang. Federated deep auc maximization for hetergeneous data with a constant communication complexity. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12219–12229. PMLR, 2021.

Chulhee Yun, Shashank Rajput, and Suvrit Sra. Minibatch vs local sgd with shuffling: Tight convergence bounds and beyond. *arXiv preprint arXiv:2110.10342*, 2021.

Hantian Zhang, Jerry Li, Kaan Kara, Dan Alistarh, Ji Liu, and Ce Zhang. Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning. In *International Conference on Machine Learning*, pages 4035–4043. JMLR. org, 2017.

Hongyi Zhang, Sashank J Reddi, and Suvrit Sra. Riemannian svrg: Fast stochastic optimization on riemannian manifolds. *Advances in Neural Information Processing Systems*, 29, 2016.

Jie Zhang, Song Guo, Xiaosong Ma, Haozhao Wang, Wenchao Xu, and Feijie Wu. Parameterized knowledge transfer for personalized federated learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Ruiliang Zhang and James Kwok. Asynchronous distributed admm for consensus optimization. In *International conference on machine learning*, pages 1701–1709. PMLR, 2014.

Shen-Yi Zhao, Hao Gao, and Wu-Jun Li. On the convergence of memory-based distributed sgd. *arXiv preprint arXiv:1905.12960*, 2019.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation. In *International Conference on Machine Learning*, pages 4120–4129. PMLR, 2017.

Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduction for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Fan Zhou and Guojing Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3219–3227, 2018.

Qihua Zhou, Song Guo, Zhihao Qu, Peng Li, Li Li, Minyi Guo, and Kun Wang. Petrel: Heterogeneity-aware distributed deep learning via hybrid synchronization. *IEEE Transactions on Parallel and Distributed Systems*, 32(5):1030–1043, 2020.

Yuhao Zhou, Qing Ye, and Jian Cheng Lv. Communication-efficient federated learning with compensated overlap-fedavg. *IEEE Transactions on Parallel and Distributed Systems*, 2021.

Martin Zinkevich, Markus Weimer, Lihong Li, and Alex Smola. Parallelized stochastic gradient descent. *Advances in neural information processing systems*, 23, 2010.

Appendices

A Proof for Cascading Compression

SSDM. An element v_j in vector $\boldsymbol{v}$ is compressed for $\{+1,-1\}$ following the probability that:

$$\tilde{\operatorname{sign}}(v_j) = \begin{cases} +1, & pr = \frac{1}{2} + \frac{v_j}{2\|\boldsymbol{v}\|} \\ -1, & pr = \frac{1}{2} - \frac{v_j}{2\|\boldsymbol{v}\|} \end{cases}$$

where $\tilde{\operatorname{sign}}(\cdot)$ refers to the compression operator. In such an operation, the expected value for $\tilde{\operatorname{sign}}(v_j)$ is $v_j/\|\boldsymbol{v}\|$. Therefore, $\mathbb{E}\tilde{\operatorname{sign}}(\boldsymbol{v}) = \boldsymbol{v}/\|\boldsymbol{v}\|$. Since the ℓ_2-norm $\|\boldsymbol{v}\|$ is a constant, SSDM can achieve unbiased update with the gradient $\|\boldsymbol{v}\| \cdot \tilde{\operatorname{sign}}(\boldsymbol{v})$, which we define as $\mathcal{Q}(\boldsymbol{v})$.

Suppose the gradients calculated by all clients are $s^{(1)}$, ..., $s^{(M)} \in \mathbb{R}^D$. Following lists various model updates, including

- **Non-compression approach:** $s_1 \triangleq \frac{1}{M}\sum_{m=1}^{M} s^{(m)}$
- **SSDM under PS:** $s_2 \triangleq \frac{1}{M}\sum_{m=1}^{M} \mathcal{Q}\left(s^{(m)}\right)$

- **SSDM using cascading compression:** $s_3 \triangleq \frac{1}{M} \underbrace{\mathcal{Q}\left(...\mathcal{Q}\left(s^{(1)}\right) + ... + s^{(M)}\right)}_{M \text{ recursive compressions}}$

Since the compressor $\mathcal{Q}$ is unbiased, the equality that $\mathbb{E}(s_2) = \mathbb{E}(s_3) = s_1$ holds. It is universally acknowledged that the update under MAR is equivalent to that under PS. In this part, we aim to evaluate the deviation between the compression and the non-compression results, i.e., $\|s_2 - s_1\|_2^2$ for SSDM under PS and $\|s_3 - s_1\|_2^2$ for SSDM using cascading compression. Prior to analyzing these two bounds, we introduce a assumption that widely adopts in (Bernstein et al., 2018a; Safaryan and Richtárik, 2021):

Assumption 1 (Bounded gradient)**.** *For any worker $m \in \{1, ..., M\}$ and vector $\boldsymbol{x} \in \mathbb{R}^D$, a scalar $G \geq 0$ satisfies*

$$\mathbb{E}\left\|s^{(m)}\right\|_2^2 \leq G^2.$$

Next, we first analyze the upper bound for the deviation under PS paradigm:

Theorem 1. *Under Assumption 1, the upper bound for $\|s_2 - s_1\|_2^2$ is $\mathcal{O}(DG^2)$.*

Proof. Based on the expression of the variance,

$$\begin{aligned}
&\mathbb{E}\left\|\frac{1}{M}\sum_{m=1}^{M}\mathcal{Q}\left(s^{(m)}\right) - \frac{1}{M}\sum_{m=1}^{M}s^{(m)}\right\|_2^2 \\
&= \mathbb{E}\left\|\frac{1}{M}\sum_{m=1}^{M}\mathcal{Q}\left(s^{(m)}\right)\right\|_2^2 - \mathbb{E}\left\|\frac{1}{M}\sum_{m=1}^{M}s^{(m)}\right\|_2^2 \leq \mathbb{E}\left\|\frac{1}{M}\sum_{m=1}^{M}\mathcal{Q}\left(s^{(m)}\right)\right\|_2^2 \\
&\leq \frac{1}{M}\sum_{m=1}^{M}\left\|s^{(m)}\right\|_2^2 \cdot \left\|\tilde{\operatorname{sign}}\left(s^{(m)}\right)\right\|_2^2 \leq DG^2
\end{aligned}$$

where the second last inequality is based on Cauchy–Schwarz inequality, and the last inequality follows Assumption 1 and the sign matrix containing D $(+1)$s or (-1)s, i.e., $\left\|\tilde{\text{sign}}\left(s^{(m)}\right)\right\|_2^2 = D, \forall m \in \{1, ..., M\}$. □

Then, the following theorem analyzes the boundary of cascading compression:

Theorem 2. *Under Assumption 1, the upper bound for the deviation of cascading compression is*

$$\|s_3 - s_1\|_2^2 \leq \frac{(2D)^M G^2}{M} \tag{1}$$

Proof. Similar to Theorem 1, we have:

$$\begin{aligned}
&\mathbb{E}\left\|\frac{1}{M}\mathcal{Q}\left(...\mathcal{Q}\left(s^{(1)}\right) + ... + s^{(M)}\right) - \frac{1}{M}\sum_{m=1}^{M} s^{(m)}\right\|_2^2 \\
&\leq \mathbb{E}\left\|\frac{1}{M}\mathcal{Q}\left(...\mathcal{Q}\left(s^{(1)}\right) + ... + s^{(M)}\right)\right\|_2^2 \\
&\leq \frac{D}{M^2}\mathbb{E}\left\|\mathcal{Q}\left(...\mathcal{Q}\left(s^{(1)}\right) + ... + s^{(M-1)}\right) + s^{(M)}\right\|_2^2 \\
&\leq \frac{2D}{M^2}\mathbb{E}\left\|\mathcal{Q}\left(...\mathcal{Q}\left(s^{(1)}\right) + ... + s^{(M-1)}\right)\right\|_2^2 + \frac{2DG^2}{M^2} \leq \frac{G^2}{M^2}\sum_{m=1}^{M}(2D)^m \leq \frac{G^2}{M}(2D)^M
\end{aligned}$$

where the third last inequality is based on $\|a+b\|_2^2 \leq 2\|a\|_2^2 + 2\|b\|_2^2$ and follows Assumption 1, while the last inequality is based on a common sense that $D \geq 1$. Generally speaking, the dimension of a neural network is far larger than the number of workers, i.e., $D >> M$, and therefore, current bound is tighter than the result $G^2(2D)^{M+1}/M^2$. □

Obviously, when $M = 1$ such that the training under PS paradigms is equivalent to that under cascading compression, they have consistent upper bound. Although both theorems show the upper bound, PS paradigm is unlike cascading compression approach that explodes rapidly with respect to the number of workers M.

B Proof for Marsit (Theorem 3.1)

By constructing an auxiliary array $\{\tilde{y}\}$ such that $\tilde{y}_t = \tilde{\mathbf{x}}_t - c_t$, where $c_t = \sum_{m=1}^{M} c_t^{(m)}/M$, we analyze its recursive function from the following two aspects:

- $c_{t+1} = 0$:

$$\tilde{y}_{t+1} = \tilde{\mathbf{x}}_{t+1} = \tilde{\mathbf{x}}_t - \frac{1}{M}\sum_{m=1}^{M}\left(\eta_l g_t^{(m)} + c_t^{(m)}\right) = \tilde{y}_t - \frac{\eta_l}{M}\sum_{m=1}^{M} g_t^{(m)} \tag{2}$$

- $c_{t+1} \neq 0$:

$$\tilde{y}_{t+1} = \tilde{\mathbf{x}}_{t+1} - c_{t+1} = \tilde{\mathbf{x}}_t - g_t - \frac{1}{M}\sum_{m=1}^{M}\left(\eta_l g_t^{(m)} + c_t^{(m)} - g_t\right) = \tilde{y}_t - \frac{\eta_l}{M}\sum_{m=1}^{M} g_t^{(m)} \tag{3}$$

Let $\tilde{g}_t = \sum_{m=1}^{M} g_t^{(m)}/M$ and $g_t^{(m)}$ here only means $\nabla f_m\left(\tilde{\mathbf{x}}_t; \xi_k^{(m)}\right)$ in this proof. Obviously, regardless the value of c_{t+1}, the recursive function is $\tilde{y}_{t+1} = \tilde{y}_t - \eta_l \tilde{g}_t$. According to L-smooth assumption for the non-convex objectives, we have:

$$\begin{aligned}\mathbb{E}F(\tilde{y}_{t+1}) - F(\tilde{y}_t) &\leq \mathbb{E}\left\langle \nabla F\left(\tilde{y}_t\right), \tilde{y}_{t+1} - \tilde{y}_t\right\rangle + \frac{L}{2}\mathbb{E}\left\|\tilde{y}_{t+1} - \tilde{y}_t\right\|_2^2 \\ &= -\eta_l \mathbb{E}\left\langle \nabla F\left(\tilde{y}_t\right), \nabla F\left(\tilde{\mathbf{x}}_t\right)\right\rangle + \frac{L\eta_l^2}{2}\mathbb{E}\left\|\frac{1}{M}\sum_{m=1}^{M} g_t^{(m)}\right\|_2^2\end{aligned}$$

$$\leq -\frac{\eta_l}{2}\left(1-L\eta_l\right)\left\|\nabla F\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2+\frac{\eta_l L^2}{2}\mathbb{E}\left\|\tilde{y}_t-\tilde{\mathbf{x}}_t\right\|_2^2+\frac{L\eta_l^2\sigma^2}{2M} \tag{4}$$

where the last inequality is based on the unbiased estimator for the calculated gradient, i.e., $\nabla f_m\left(\tilde{\mathbf{x}}_t;\xi_k^{(m)}\right)$. Next, we will find the bound for $\mathbb{E}\left\|\tilde{y}_t-\tilde{\mathbf{x}}_t\right\|_2^2$, which is equivalent to $\mathbb{E}\left\|c_t\right\|_2^2$. Algorithm 1 performs the full precision synchronization every K rounds and therefore, there exists a $t_0 > t-K$ such that $c_{t_0}=0$. Following analyzes the case that c_t is a non-zero vector:

$$\begin{aligned}
\mathbb{E}\left\|c_t\right\|_2^2 &= \mathbb{E}\left\|c_{t-1}+\eta_l\tilde{g}_{t-1}-\eta_s g_{t-1}\right\|_2^2\\
&\leq\left(1+\frac{1}{K}\right)\mathbb{E}\left\|c_{t-1}\right\|_2^2+(1+K)\mathbb{E}\left\|\eta_l\tilde{g}_{t-1}-\eta_s g_{t-1}\right\|_2^2\\
&\leq\sum_{\tau=t_0}^{t-1}\left(1+\frac{1}{K}\right)^{t-1-\tau}\cdot(1+K)\mathbb{E}\left\|\eta_l\tilde{g}_\tau-\eta_s g_\tau\right\|_2^2\\
&\leq 3(1+K)\cdot\sum_{\tau=t_0}^{t-1}\mathbb{E}\left\|\eta_l\tilde{g}_\tau-\eta_s g_\tau\right\|_2^2\\
&\leq 6\eta_l^2(1+K)\sum_{\tau=t_0}^{t-1}\mathbb{E}\left\|\tilde{g}_\tau\right\|_2^2+6\eta_s^2(1+K)\sum_{\tau=t_0}^{t-1}\mathbb{E}\left\|g_\tau\right\|_2^2\\
&= 6\eta_l^2(1+K)\sum_{\tau=t_0}^{t-1}\mathbb{E}\left\|\nabla F(\tilde{\mathbf{x}}_\tau)\right\|_2^2+6\eta_l^2(1+K)K\cdot\frac{\sigma^2}{M}+6\eta_s^2(1+K)KD
\end{aligned} \tag{5}$$

where the first inequality is based on $(a+b)^2\leq(1+\frac{1}{K})a^2+(1+K)b^2$, and the last equality is according to $\left\|g_\tau\right\|_2^2=D$ because it is only constituted with $\{+1,-1\}$ for all D dimensions. Suppose the optimal solution for the non-convex objective

$F(\cdot)$ is F_*. Therefore, plugging the result from Equation 5 into Equation 4, and summing Equation 4 for all ts from 0 to T, we have:

$$\begin{aligned} F_* - F(\tilde{\mathbf{x}}_0) &\leq \sum_{t=0}^{T-1} \left(\mathbb{E}F(\tilde{y}_{t+1}) - F(\tilde{y}_t)\right) \\ &\leq -\frac{\eta_l}{2}\left(1 - L\eta_l - 3L^2\eta_l^2 K(K+1)\right) \sum_{t=0}^{T-1} \|\nabla F\left(\tilde{\mathbf{x}}_t\right)\|_2^2 \\ &\quad + \frac{\eta_l L^2 T}{2}\left(6\eta_l^2(1+K)K \cdot \frac{\sigma^2}{M} + 6\eta_s^2(1+K)KD\right) + \frac{L\eta_l^2\sigma^2 T}{2M} \end{aligned}$$

By setting $\eta_l = \sqrt{M/T}$ and $\eta_s = 1/\sqrt{TD}$, and assuming that T is sufficiently large, i.e., $T \geq 9L^2K^2(K+1)^2$, we can obtain the desired conclusion.

C Proof of Theorem 4.1

The following lemma describes the relationship among three different parameters under strongly-convex function:

Lemma 1. *Under Assumption 4.1 and Assumption 4.2, given* $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^d$*, the following formula holds under the strongly-convex objectives* F*:*

$$\langle \nabla F(\mathbf{a}), \mathbf{b} - \mathbf{c}\rangle \leq F(\mathbf{b}) - F(\mathbf{c}) - \frac{\mu}{4}\|\mathbf{b} - \mathbf{c}\|_2^2 + L\|\mathbf{a} - \mathbf{c}\|_2^2 \tag{6}$$

Proof. With $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ that are within the domain of F, we can get the following inequalities that come from Assumption 4.1 and Assumption 4.2, respectively:

$$\langle \nabla F(\mathbf{a}), \mathbf{b}-\mathbf{a} \rangle \leq F(\mathbf{b}) - F(\mathbf{a}) + \frac{L}{2}\|\mathbf{a}-\mathbf{b}\|_2^2$$

$$\langle \nabla F(\mathbf{a}), \mathbf{a}-\mathbf{c} \rangle \leq F(\mathbf{a}) - F(\mathbf{c}) - \frac{\mu}{2}\|\mathbf{a}-\mathbf{c}\|_2^2$$

By a formula that

$$\|\mathbf{a}-\mathbf{b}\|_2^2 \leq 2\|\mathbf{a}-\mathbf{c}\|_2^2 + 2\|\mathbf{b}-\mathbf{c}\|_2^2$$

we have:

$$\langle \nabla F(\mathbf{a}), \mathbf{b}-\mathbf{c} \rangle \leq F(\mathbf{b}) - F(\mathbf{c}) - \frac{\mu}{4}\|\mathbf{b}-\mathbf{c}\|_2^2 + \frac{L+\mu}{2}\|\mathbf{a}-\mathbf{c}\|_2^2$$

The inequality holds when $L \geq \mu$. □

The update rule for FedAvg under heterogeneous steps:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \sum_{i=1}^{M} \omega_i \sum_{k=0}^{K_i-1} \nabla f_i\left(\mathbf{x}_{t,k}^{(i)}, \varepsilon_k^{(i)}\right) \tag{7}$$

Therefore, the bound established for $\mathbb{E}\left\|\mathbf{x}_{t+1} - \mathbf{x}_*\right\|_2^2$ should be:

$$\begin{aligned}\mathbb{E}\left\|\mathbf{x}_{t+1} - \mathbf{x}^*\right\|_2^2 &= \mathbb{E}\left\|\mathbf{x}_t - \eta \sum_{i=1}^{M} \omega_i \sum_{k=0}^{K_i-1} \nabla f_i\left(\mathbf{x}_{t,k}^{(i)}, \varepsilon_k^{(i)}\right) - \mathbf{x}_*\right\|_2^2 \\ &= \mathbb{E}\left\|\mathbf{x}_t - \mathbf{x}^*\right\|_2^2 + \mathbb{E}\left\|\eta \sum_{i=1}^{M} \omega_i \sum_{k=0}^{K_i-1} \nabla f_i\left(\mathbf{x}_{t,k}^{(i)}, \varepsilon_k^{(i)}\right)\right\|_2^2 \\ &\quad - 2\mathbb{E}\left\langle \mathbf{x}_t - \mathbf{x}_*, \eta \sum_{i=1}^{M} \omega_i \sum_{k=0}^{K_i-1} \nabla f_i\left(\mathbf{x}_{t,k}^{(i)}, \varepsilon_k^{(i)}\right)\right\rangle\end{aligned}$$

$$= \mathbb{E}\left\|\mathbf{x}_t - \mathbf{x}^*\right\|_2^2 + \eta^2 \sum_{i=1}^{M} \omega_i^2 K_i \sigma^2 \tag{8}$$

$$\underbrace{+ \eta^2 \mathbb{E}\left\| \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right) \right\|_2^2}_{\mathcal{A}_2} \tag{9}$$

$$\underbrace{-2\mathbb{E}\left\langle \mathbf{x}_t - \mathbf{x}_*, \eta \sum_{i=1}^{M} \omega_i \sum_{k=0}^{K_i-1} \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right) \right\rangle}_{\mathcal{A}_1} \tag{10}$$

We first find a upper bound for $\mathcal{A}_1$ in accordance with Lemma 1:

$$\begin{aligned}
\mathcal{A}_1 &= 2\eta \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \cdot \mathbb{E}\left\langle \mathbf{x}^* - \mathbf{x}_t, \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right) \right\rangle \\
&\leq 2\eta \sum_{i=1}^{M} \omega_i K_i \left[F_i\left(\mathbf{x}^*\right) - F_i\left(\mathbf{x}_t\right)\right] - \frac{\eta\mu\bar{K}}{2}\left\|\mathbf{x}_t - \mathbf{x}^*\right\|_2^2 && (11) \\
&\quad + 2\eta L \underbrace{\sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)} - \mathbf{x}_t\right\|_2^2}_{\mathcal{A}_3} && (12)
\end{aligned}$$

To find the maximum value for $\mathcal{A}_3$, we bound $\mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)} - \mathbf{x}_t\right\|_2^2$ for $k \in \{1, ..., K_i\}$ via the following inequality:

$$\begin{aligned}
\mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)} - \mathbf{x}_t\right\|_2^2 &= \mathbb{E}\left\|\mathbf{x}_{t,k-1}^{(i)} - \eta\nabla f_i\left(\mathbf{x}_{t,k-1}^{(i)}, \varepsilon_{k-1}^{(i)}\right) - \mathbf{x}_t\right\|_2^2 \\
&\leq \mathbb{E}\left\|\mathbf{x}_{t,k-1}^{(i)} - \mathbf{x}_t - \eta\nabla F_i\left(\mathbf{x}_{t,k-1}^{(i)}\right)\right\|_2^2 + \eta^2\sigma^2 \\
&\overset{(a)}{\leq} \left(1 + \frac{1}{K_i - 1}\right)\left\|\mathbf{x}_{t,k-1}^{(i)} - \mathbf{x}_t\right\|_2^2 + \eta^2\sigma^2 \\
&\quad + K_i\eta^2\left(2\left\|\nabla F_i\left(\mathbf{x}_{t,k-1}^{(i)}\right) - \nabla F_i\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2 + 2\left\|\nabla F_i\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2\right)
\end{aligned}$$

$$\begin{aligned}
&\leq \left(1+\frac{1}{K_i-1}+2K_i\eta^2L^2\right)\left\|\mathbf{x}_{t,k-1}^{(i)}-\mathbf{x}_t\right\|_2^2 \\
&\quad+\eta^2\sigma^2+2K_i\eta^2\left\|\nabla F_i\left(\mathbf{x}_t\right)\right\|_2^2
\end{aligned}$$

where (a) follows triangle inequality, i.e., $(x+y)^2 \leq (1+k)x^2+(1+1/k)y^2$ for all $k>0$. By setting $\eta \leq \sqrt{\frac{1}{2K_{\text{max}}(K_{\text{max}}-1)L^2}}$, we have:

$$\begin{aligned}
\mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)}-\mathbf{x}_t\right\|_2^2 &\leq \sum_{\kappa=0}^{k-1}\left(1+\frac{2}{K_i-1}\right)^{\kappa}\left(\eta^2\sigma^2+2K_i\eta^2\left\|\nabla F_i\left(\mathbf{x}_t\right)\right\|_2^2\right) \\
&= \frac{\left(1+\frac{2}{K_i-1}\right)^k-1}{\frac{2}{K_i-1}}\left(\eta^2\sigma^2+2K_i\eta^2\left\|\nabla F_i\left(\mathbf{x}_t\right)\right\|_2^2\right) \\
&\overset{(a)}{\leq} 4K_i\left(\eta^2\sigma^2+2K_i\eta^2\left\|\nabla F_i\left(\mathbf{x}_t\right)\right\|_2^2\right)
\end{aligned} \tag{13}$$

where (a) is on account for:

$$\left(1+\frac{2}{K_i-1}\right)^{K_i} \leq 9 \qquad \text{for} \qquad K_i \geq 2$$

Based on the derivative above, we can obtain the bound for $\mathcal{A}_3$ with:

$$\begin{aligned}
\mathcal{A}_3 &\leq \sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\cdot 4K_i\left(\eta^2\sigma^2+2K_i\eta^2\left\|\nabla F_i\left(\mathbf{x}_t\right)\right\|_2^2\right) \\
&= 4\eta^2\sigma^2\sum_{i=1}^{M}\omega_iK_i^2+8\eta^2\sum_{i=1}^{M}\omega_iK_i^3\left\|\nabla F_i\left(\mathbf{x}_t\right)\right\|_2^2
\end{aligned}$$

Therefore, the bound for $\mathcal{A}_1$ can be further simplified as:

$$\begin{aligned}\mathcal{A}_1 \le\ & 2\eta \sum_{i=1}^{M} \omega_i K_i \left[F_i\left(\mathbf{x}^*\right) - F_i\left(\mathbf{x}_t\right)\right] - \frac{\eta\mu\bar{K}}{2}\left\|\mathbf{x_t} - \mathbf{x}_*\right\|_2^2 \\ & + 2\eta L\left[4\eta^2\sigma^2 \sum_{i=1}^{M}\omega_i K_i^2 + 8\eta^2 \sum_{i=1}^{M}\omega_i K_i^3 \left\|\nabla F_i\left(\mathbf{x}_t\right)\right\|_2^2\right]\end{aligned}$$

Next, we consider the bound for $\mathcal{A}_2$:

$$\begin{aligned}\mathcal{A}_2 \le\ & 2\eta^2 \cdot \mathbb{E}\left\|\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\left[\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right) - \nabla F_i\left(\mathbf{x}_t\right)\right]\right\|_2^2 \\ & + 2\eta^2 \cdot \mathbb{E}\left\|\sum_{i=1}^{M}\omega_i K_i \nabla F_i\left(\mathbf{x}_t\right)\right\|_2^2 \\ \le\ & 2\eta^2 L^2 \sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i K_i \cdot \mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)} - \mathbf{x}_t\right\|_2^2 + 2\eta^2 \sum_{i=1}^{M}\omega_i K_i^2 \left\|\nabla F_i\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2 \\ \le\ & 8\eta^4 L^2 \sigma^2 \sum_{i=1}^{M}\omega_i K_i^3 + 2\eta^2 \sum_{i=1}^{M}\omega_i K_i^2\left(1 + 8\eta^2 L^2 K_i^2\right)\left\|\nabla F_i\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2\end{aligned}$$

Therefore, pluging $\mathcal{A}_1$ and $\mathcal{A}_2$ into the inequality bound for $\mathbb{E}\left\|\mathbf{x}_{t+1} - \mathbf{x}^*\right\|_2^2$, the bound can be simplified as:

$$\begin{aligned}\mathbb{E}\left\|\mathbf{x}_{t+1} - \mathbf{x}^*\right\|_2^2 \le\ & \left(1 - \frac{\eta\mu\bar{K}}{2}\right)\mathbb{E}\left\|\mathbf{x}_t - \mathbf{x}^*\right\|_2^2 + 2\eta\sum_{i=1}^{M}\omega_i K_i\left[F_i(\mathbf{x}^*) - F_i\left(\mathbf{x}_t\right)\right] \\ & + 8\eta^3\sigma^2 L \sum_{i=1}^{M}\omega_i K_i^2 + \eta^2\sigma^2 \sum_{i=1}^{M}\omega_i^2 K_i \\ & + 8\eta^4 L^2\sigma^2 \sum_{i=1}^{M}\omega_i K_i^3 + 16\eta^3 L \sum_{i=1}^{M}\omega_i K_i^3 \left\|\nabla F_i\left(\mathbf{x}_t\right)\right\|_2^2\end{aligned}$$

$$+ 2\eta^2 \sum_{i=1}^{M} \omega_i K_i^2 \left(1 + 8\eta^2 L^2 K_i\right) \left\|\nabla F_i\left(\mathbf{x}_t\right)\right\|_2^2$$

Divided $\bar{K}$ on the both side, we can obtain the following formula:

$$\begin{aligned}
&\frac{1}{\bar{K}} \sum_{i=1}^{M} \omega_i K_i \left[F_i(\mathbf{x}^*) - F_i\left(\mathbf{x}_t\right)\right] \\
&\leq \frac{1}{\bar{K}\eta}\left(1 - \frac{\eta\mu\bar{K}}{2}\right) \mathbb{E}\left\|\mathbf{x}_t - \mathbf{x}^*\right\|_2^2 - \frac{1}{\bar{K}\eta} \mathbb{E}\left\|\mathbf{x}_{t+1} - \mathbf{x}^*\right\|_2^2 \\
&\quad + \frac{8\eta^2\sigma^2 L}{\bar{K}} \sum_{i=1}^{M} \omega_i K_i^2 + \frac{\eta\sigma^2}{\bar{K}} \sum_{i=1}^{M} \omega_i^2 K_i + \frac{8\eta^3 L^2 \sigma^2}{\bar{K}} \sum_{i=1}^{M} \omega_i K_i^3
\end{aligned}$$

By applying Lemma 1 from (Karimireddy et al., 2020b), we can obtain the desirable result. It is worthwhile to mention a formula below that supports the reason why the gap exists between a stable point and the optimal solution:

$$\begin{aligned}
\sum_{i=1}^{M} \omega_i K_i (F_i(\mathbf{x}_t) - F_i(\mathbf{x}_*)) \geq K_{\min} &\sum_{i=1}^{M} \omega_i (F_i(\mathbf{x}_t) - F_i(\mathbf{x}_*)) \\
&- \sum_{i=1}^{M} \omega_i (K_i - K_{\min}) F_i(\mathbf{x}_*)
\end{aligned}$$

The inequality holds when the value of the objective function is non-negative. This formula indicates that the data heterogeneity can be eliminated under homogeneous computing environment since for all $i \in \{1, ..., M\}$, $K_i = K_{\min}$. Thus, in this case, we can obtain the same convergence order as (Karimireddy et al., 2020b).

D Proof of Theorem 4.2

According to L-smooth, we have:

$$\mathbb{E}\left[F(\tilde{\mathbf{x}}_{t+1})\right] - F(\tilde{\mathbf{x}}_t) \leq \mathbb{E}\left\langle \nabla F\left(\tilde{\mathbf{x}}_t\right), \tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t \right\rangle + \frac{L}{2}\mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t\right\|_2^2 \tag{14}$$

The first term of Equation (14). We firstly find the bound for the first term of Equation (14):

$$\begin{aligned}
&\mathbb{E}\left\langle \nabla F\left(\tilde{\mathbf{x}}_t\right), \tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t \right\rangle \\
&= \mathbb{E}\Bigg\langle \nabla F\left(\tilde{\mathbf{x}}_t\right), -\eta \sum_{i=1}^{M}\sum_{k=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right) \\
&\qquad + \eta\lambda \sum_{i=1}^{M} \omega_i \sum_{\kappa=0}^{K_i-1} \nabla F_i\left(\mathbf{x}_{t-1,\kappa}^{(i)}\right) - \eta\lambda\bar{K}\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right) \\
&\qquad - \eta\lambda\bar{K} \sum_{i, K_i \leq \bar{K}} \sum_{k=0}^{K_i-1} \frac{\omega_i}{K_i}\left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right) - \nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right)\Bigg\rangle \\
&= -\frac{\eta\lambda}{\bar{K}}\mathbb{E}\Bigg\langle \bar{K}\nabla F\left(\tilde{\mathbf{x}}_t\right), \sum_{i=1}^{M}\sum_{k=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right) \\
&\qquad\qquad - \sum_{i=1}^{M}\sum_{\kappa=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t-1,\kappa}^{(i)}\right) + \bar{K}\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right) \\
&\qquad\qquad + \bar{K} \sum_{i, K_i \leq \bar{K}} \sum_{k=0}^{K_i-1} \frac{\omega_i}{K_i}\left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right) - \nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right)\Bigg\rangle \\
&\quad - \eta(1-\lambda)K_{\text{max}}\mathbb{E}\Bigg\langle \nabla F\left(\tilde{\mathbf{x}}_t\right), \frac{1}{K_{\text{max}}} \sum_{i=1}^{M}\sum_{k=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)\Bigg\rangle \\
&= -\frac{\eta\lambda\bar{K}}{2}\left\|\nabla F\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2 \qquad\qquad (15) \\
&\quad - \frac{\eta\lambda}{2\bar{K}}\mathbb{E}\left\|\sum_{i=1}^{M}\sum_{k=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)\right.
\end{aligned}$$

$$-\sum_{i=1}^{M}\sum_{\kappa=0}^{K_i-1}\omega_i\nabla F_i\left(\mathbf{x}_{t-1,\kappa}^{(i)}\right)+\bar{K}\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right)$$

$$+\bar{K}\sum_{i,K_i\leq\bar{K}}\sum_{k=0}^{K_i-1}\frac{\omega_i}{K_i}\left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right)-\nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right)\Bigg\|_2^2 \tag{16}$$

$$+\frac{\eta\lambda}{2\bar{K}}\mathbb{E}\Bigg\|\bar{K}\nabla F\left(\tilde{\mathbf{x}}_t\right)-\bar{K}\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right)\sum_{k=0}^{K_i-1}\omega_i\left(\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)-\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right)\right)$$

$$-\bar{K}\sum_{i,K_i\leq\bar{K}}\sum_{k=0}^{K_i-1}\frac{\omega_i}{K_i}\left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right)-\nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right)\Bigg\|_2^2 \tag{17}$$

$$-\frac{\eta(1-\lambda)K_{\max}}{2}\left\|\nabla F\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2-\frac{\eta(1-\lambda)}{2K_{\max}}\mathbb{E}\left\|\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)\right\|_2^2 \tag{18}$$

$$+\frac{\eta(1-\lambda)K_{\max}}{2}\mathbb{E}\left\|\nabla F\left(\tilde{\mathbf{x}}_t\right)-\frac{1}{K_{\max}}\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)\right\|_2^2 \tag{19}$$

Next, we bound the term of Equation (19) ignoring the coefficient term, i.e., $\frac{\eta(1-\lambda)K_{\max}}{2}$:

$$\mathbb{E}\left\|\nabla F\left(\tilde{\mathbf{x}}_t\right)-\frac{1}{K_{\max}}\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)\right\|_2^2$$

$$\leq\mathbb{E}\left\|\sum_{i=1}^{M}\omega_i\left(1-\frac{K_i}{K_{\max}}\right)\nabla F\left(\tilde{\mathbf{x}}_t\right)-\frac{1}{K_{\max}}\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\left(\nabla F_i\left(\tilde{\mathbf{x}}_t\right)-\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)\right)\right\|_2^2$$

$$\overset{(a)}{\leq}\left(1+\frac{K_{\min}}{K_{\max}}\right)\mathbb{E}\left\|\sum_{i=1}^{M}\omega_i\left(1-\frac{K_i}{K_{\max}}\right)\nabla F_i\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2+\left(1+\frac{K_{\max}}{K_{\min}}\right)T_1$$

$$\leq\left(1-\frac{\bar{K}}{K_{\max}}\right)B^2\mathbb{E}\left\|\nabla F\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2+\left(1+\frac{K_{\max}}{K_{\min}}\right)T_1$$

where

$$\begin{aligned} T_1 &= \mathbb{E}\left\|\frac{1}{K_{\max}}\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\left(\nabla F_i\left(\tilde{\mathbf{x}}_t\right)-\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)\right)\right\|_2^2 \\ &\leq \frac{L^2}{K_{\max}^2}\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i K_i\mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)}-\tilde{\mathbf{x}}_t\right\|_2^2 \end{aligned}$$

and (a) follows triangle inequality, i.e., $(x+y)^2 \leq (1+c)x^2+(1+1/c)y^2$ for all $c>0$. We denote Equation (17) omitted the coefficient $\frac{\eta\lambda}{2\bar{K}}$ by T_2. Therefore, its upper bound is obtained through the following derivation:

$$\begin{aligned} T_2 &\overset{(a)}{\leq} 5\bar{K}^2\mathbb{E}\left\|\nabla F\left(\tilde{\mathbf{x}}_t\right)-\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right)\right\|_2^2+5\mathbb{E}\left\|\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\left(\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)-\nabla F\left(\tilde{\mathbf{x}}_t\right)\right)\right\|_2^2 \\ &\quad+5\mathbb{E}\left\|\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right)-\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right)\right)\right\|_2^2 \\ &\quad+5\mathbb{E}\left\|\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\left(\nabla F\left(\tilde{\mathbf{x}}_t\right)-\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right)\right)\right\|_2^2 \\ &\quad+5\bar{K}^2\mathbb{E}\left\|\sum_{i,K_i\leq\bar{K}}\sum_{k=0}^{K_i-1}\frac{\omega_i}{K_i}\left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right)-\nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right)\right\|_2^2 \\ &\leq 5L^2\left(\bar{K}^2+\sum_{i=1}^{M}\omega_i K_i^2\right)\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\tilde{\mathbf{x}}_{t-1}\right\|_2^2+5L^2\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i K_i\mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)}-\tilde{\mathbf{x}}_t\right\|_2^2 \\ &\quad+5L^2\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i K_i\mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)}-\tilde{\mathbf{x}}_{t-1}\right\|_2^2 \\ &\quad+5\bar{K}^2L^2\sum_{i,K_i\leq\bar{K}}\sum_{k=0}^{K_i-1}\frac{\omega_i}{K_i}\mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)}-\tilde{\mathbf{x}}_{t-1}\right\|_2^2 \\ &\leq 10L^2\sum_{i=1}^{M}\omega_i K_i^2\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\tilde{\mathbf{x}}_{t-1}\right\|_2^2+5L^2\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i K_i\mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)}-\tilde{\mathbf{x}}_t\right\|_2^2 \end{aligned}$$

$$+ 5L^2 \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \left(K_i + \frac{\bar{K}^2}{K_i} \right) \mathbb{E} \left\| \mathbf{x}_{t-1,k}^{(i)} - \tilde{\mathbf{x}}_{t-1} \right\|_2^2 \tag{20}$$

where (a) divides $\left(\nabla F_i \left(\mathbf{x}_{t,k}^{(i)}\right) - \nabla F_i \left(\mathbf{x}_{t-1,k}^{(i)}\right)\right)$ into three terms, i.e., (each bracket should be treated as an individual term): $\left(\nabla F_i \left(\mathbf{x}_{t,k}^{(i)}\right) - \nabla F_i \left(\tilde{\mathbf{x}}_t\right)\right) - \left(\nabla F_i \left(\mathbf{x}_{t-1,k}^{(i)}\right) - \nabla F_i \left(\tilde{\mathbf{x}}_{t-1}\right)\right) + \left(\nabla F_i \left(\tilde{\mathbf{x}}_t\right) - \nabla F_i \left(\tilde{\mathbf{x}}_{t-1}\right)\right)$. By observing Equation (20), we notice that it is indispensable to acquire the upper limit of $\mathbb{E} \left\| \mathbf{x}_{t,k}^{(i)} - \tilde{\mathbf{x}}_t \right\|_2^2$:

$$\begin{aligned}
\mathbb{E} \left\| \mathbf{x}_{t,k}^{(i)} - \tilde{\mathbf{x}}_t \right\|_2^2 &= \mathbb{E} \left\| \mathbf{x}_{t,k-1}^{(i)} - \eta \left[g_{t,k-1}^{(i)} + \lambda \left(\nu - \nu^{(i)} \right) \right] - \tilde{\mathbf{x}}_t \right\|_2^2 \\
&\leq \left(1 + \frac{1}{K_i - 1} \right) \mathbb{E} \left\| \mathbf{x}_{t,k-1}^{(i)} - \tilde{\mathbf{x}}_t \right\|_2^2 \\
&\quad + K_i \eta^2 \mathbb{E} \left\| g_{t,k-1}^{(i)} - \frac{\lambda}{K_i} \sum_{\kappa=0}^{K_i-1} g_{t-1,\kappa}^{(i)} + \lambda \sum_{j=1}^{M} \sum_{k=0}^{K_j-1} \frac{\omega_j}{K_j} g_{t-1,k}^{(j)} \right. \\
&\qquad \left. + \lambda \sum_{j, K_j > \bar{K}} \sum_{k=0}^{K_j-1} \frac{\omega_j}{K_j} \left(g_{t-1,0}^{(j)} - g_{t-1,k}^{(j)} \right) \right\|_2^2 \\
&\leq \left(1 + \frac{1}{K_i - 1} \right) \mathbb{E} \left\| \mathbf{x}_{t,k-1}^{(i)} - \tilde{\mathbf{x}}_t \right\|_2^2 && (21) \\
&\quad + K_i \eta^2 \mathbb{E} \left\| \nabla F_i \left(\mathbf{x}_{t,k-1}^{(i)} \right) - \frac{\lambda}{K_i} \sum_{\kappa=0}^{K_i-1} \nabla F_i \left(\mathbf{x}_{t-1,\kappa}^{(i)} \right) \right. \\
&\qquad + \lambda \sum_{j=1}^{M} \frac{\omega_j}{K_j} \sum_{k=0}^{K_j-1} \nabla F_j \left(\mathbf{x}_{t-1,k}^{(j)} \right) \\
&\qquad \left. + \lambda \sum_{j, K_j > \bar{K}} \frac{\omega_j}{K_j} \sum_{k=0}^{K_j-1} \left(\nabla F_j \left(\tilde{\mathbf{x}}_{t-1} \right) - \nabla F_j \left(\mathbf{x}_{t-1,k}^{(j)} \right) \right) \right\|_2^2 && (22)
\end{aligned}$$

$$+ 4K_i^2 \eta^2 \sigma^2 + 4\lambda^2 \eta^2 \sigma^2 + 12 K_i^2 \eta^2 \lambda^2 \sigma^2 \sum_{j=1}^{M} \frac{\omega_j^2}{K_j} \tag{23}$$

We denote the second norm in Equation (22) by T_3. Similar to the derivation for T_2, i.e., Equation (20), we have the following inequality under Assumption 4.4:

$$\begin{aligned} T_3 \leq & 8L^2 \mathbb{E} \left\| \mathbf{x}_{t,k-1}^{(i)} - \tilde{\mathbf{x}}_t \right\|_2^2 + \frac{8\lambda^2 L^2}{K_i} \sum_{\kappa=0}^{K_i-1} \mathbb{E} \left\| \mathbf{x}_{t-1,\kappa}^{(i)} - \tilde{\mathbf{x}}_{t-1} \right\|_2^2 \\ & + 16\lambda^2 L^2 \sum_{j=1}^{M} \sum_{k=0}^{K_j-1} \frac{\omega_j}{K_j} \mathbb{E} \left\| \mathbf{x}_{t-1,k}^{(j)} - \tilde{\mathbf{x}}_{t-1} \right\|_2^2 \\ & + 8\left((1-\lambda)^2 B^2 + \lambda^2\right) \mathbb{E} \left\| \nabla F\left(\tilde{\mathbf{x}}_t\right) \right\|_2^2 \\ & + 16\lambda^2 L^2 \mathbb{E} \left\| \tilde{\mathbf{x}}_t - \tilde{\mathbf{x}}_{t-1} \right\|_2^2 \end{aligned} \tag{24}$$

By setting $\eta \leq \frac{1}{2\sqrt{2}LK_{\max}}$ and following the steps of Equation (13), we have:

$$\mathbb{E} \left\| \mathbf{x}_{t,k}^{(i)} - \tilde{\mathbf{x}}_t \right\|_2^2 \leq \frac{K_i - 1}{2} \left(\left(1 + \frac{2}{K_i - 1} \right)^{k+1} - 1 \right) T_4 \tag{25}$$

where

$$\begin{aligned} T_4 = & 16\lambda^2 L^2 K_i \eta^2 \left\| \tilde{\mathbf{x}}_t - \tilde{\mathbf{x}}_{t-1} \right\|_2^2 + 8\lambda^2 L^2 \eta^2 \sum_{k=0}^{K_i-1} \left\| \mathbf{x}_{t-1,k}^{(i)} - \tilde{\mathbf{x}}_{t-1} \right\|_2^2 \\ & + 16\lambda^2 L^2 K_i \eta^2 \sum_{j=1}^{M} \sum_{k=0}^{K_j-1} \frac{\omega_j}{K_j} \left\| \mathbf{x}_{t-1,k}^{(j)} - \tilde{\mathbf{x}}_{t-1} \right\|_2^2 \\ & + 8K_i \eta^2 \left((1-\lambda)^2 B^2 + \lambda^2\right) \left\| \nabla F\left(\tilde{\mathbf{x}}_t\right) \right\|_2^2 + 4K_i \eta^2 \sigma^2 \\ & + 4\lambda^2 \eta^2 \sigma^2 + 12 K_i \eta^2 \lambda^2 \sigma^2 \sum_{j=1}^{M} \frac{\omega_j^2}{K_j}. \end{aligned}$$

As a result, when the learning rate η is sufficiently small, the upper bound for T_2 should be

$$\begin{aligned}
T_2 \leq & \left(20L^2 \sum_{i=1}^{M} \omega_i K_i^2\right) \mathbb{E}\left\|\tilde{\mathbf{x}}_t - \tilde{\mathbf{x}}_{t-1}\right\|_2^2 \\
& + 20\bar{K}^2 L^2 \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)} - \tilde{\mathbf{x}}_{t-1}\right\|_2^2 \\
& + 60\eta^2 L^2 K_{\text{max}}^4 \left((1-\lambda)^2 B^2 + \lambda^2\right) \mathbb{E}\left\|\nabla F\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2 \\
& + 30\eta^2 L^2 \sigma^2 \sum_{i=1}^{M} \omega_i K_i^4 + 30\eta^2 L^2 \sigma^2 \lambda^2 \sum_{i=1}^{M} \omega_i K_i^3 \\
& + 90\eta^2 L^2 \sigma^2 \lambda^2 \left(\sum_{j=1}^{M} \frac{\omega_j^2}{K_j}\right) \sum_{i=1}^{M} \omega_i K_i^4
\end{aligned} \tag{26}$$

The second term of Equation (14). We now give the upper limit for $\mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t\right\|_2^2$:

$$\begin{aligned}
& \mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t\right\|_2^2 \\
\leq & \eta^2 \mathbb{E}\left\| \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right) \right. \\
& + \lambda \sum_{i, K_i > \bar{K}} \sum_{k=0}^{K_i-1} \frac{\omega_i \bar{K}}{K_i} \left(\nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right) - \nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right)\right) \\
& \left. + \lambda \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \left(\frac{\bar{K}}{K_i} - 1\right) \nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right) \right\|_2^2 \\
\leq & 2\eta^2 \lambda^2 \mathbb{E}\left\| \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right) - \sum_{i=1}^{M} \sum_{\kappa=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t-1,\kappa}^{(i)}\right) \right. \\
& + \sum_{i, K_i \leq \bar{K}} \frac{\omega_i \bar{K}}{K_i} \sum_{k=0}^{K_i-1} \left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right) - \nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right)
\end{aligned}$$

$$+\bar{K}\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right)\|_2^2$$

$$+ 2\eta^2(1-\lambda)^2\left\|\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)\right\|_2^2$$

$$+ 3\eta^2\sigma^2\sum_{i=1}^{M}\omega_i^2K_i + 3\eta^2\lambda^2\sigma^2\sum_{i=1}^{M}\omega_i^2\left(\frac{5\bar{K}^2}{K_i}+K_i\right) \tag{27}$$

Final result. By the inequality from Equation (26) and Equation (27), we can add two extra terms on the left hand side of Equation 14, i.e., $\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\tilde{\mathbf{x}}_{t-1}\right\|_2^2$ and $\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_iK_i\mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)}-\tilde{\mathbf{x}}_{t-1}\right\|_2^2$, and obtain the following bound when the learning rate is sufficiently small:

$$\mathbb{E}[F(\tilde{\mathbf{x}}_{t+1})] + p_1\mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1}-\tilde{\mathbf{x}}_t\right\|_2^2 + p_2\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_iK_i\mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)}-\tilde{\mathbf{x}}_t\right\|_2^2$$
$$\leq F(\tilde{\mathbf{x}}_t) + p_1\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\tilde{\mathbf{x}}_{t-1}\right\|_2^2 + p_2\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_iK_i\mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)}-\tilde{\mathbf{x}}_{t-1}\right\|_2^2$$
$$-\left(\frac{\eta\lambda\bar{K}}{2}+\frac{\eta\left(1-\lambda\right)K_{\text{max}}}{2}\left(1-\left(1-\frac{\bar{K}}{K_{\text{max}}}\right)B^2\right)\right)\left\|\nabla F\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2$$
$$+3\left(\frac{L}{2}+p_1\right)\eta^2\sigma^2\left(\sum_{i=1}^{M}\omega_i^2K_i+\lambda^2\sum_{i=1}^{M}\omega_i^2\left(\frac{5\bar{K}^2}{K_i}+K_i\right)\right)$$

where $p_1 = o\left(\frac{\eta\lambda}{\bar{K}}L^2\sum_{i=1}^{M}\omega_i\left(K_i^2+\bar{K}^2\right)\right)$ and $p_2 = o\left(\frac{\eta\lambda}{\bar{K}}L^2\left(1+\frac{\bar{K}^2}{K_{\text{min}}^2}\right)\right)$. Therefore, the final result is:

$$\frac{1}{T}\sum_{t=1}^{T}\left\|\nabla F\left(\tilde{\mathbf{x}}_t\right)\right\|_2^2 = \mathcal{O}\left(\frac{F(\mathbf{x}_*)-F(\tilde{\mathbf{x}}_1)}{\eta\lambda\bar{K}T}\right)+\mathcal{O}\left(\frac{\eta\sigma^2L}{\lambda\bar{K}}\sum_{i=1}^{M}\omega_i^2K_i\right)$$
$$+\mathcal{O}\left(\frac{\eta\sigma^2L\lambda}{\bar{K}}\sum_{i=1}^{M}\omega_i^2\left(\frac{\bar{K}}{K_i}-1\right)^2\right)+\mathcal{O}\left(\eta\sigma^2L\lambda\sum_{i,K_i>\bar{K}}\frac{\omega_i^2\bar{K}}{K_i}\right)$$

E Proof of Theorem 4.3

At the very beginning, we set $\lambda = 1$ to find a valid bound. Based on the definition, we can find a recursion function for $\mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1} - \mathbf{x}_*\right\|_2^2$:

$$\begin{aligned}
&\mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1} - \mathbf{x}_*\right\|_2^2 = \mathbb{E}\left\|(\tilde{\mathbf{x}}_t - \mathbf{x}_*) + (\tilde{\mathbf{x}}_{t+1} - \mathbf{x}_*)\right\|_2^2 \\
&= \mathbb{E}\left\|\tilde{\mathbf{x}}_t - \mathbf{x}_*\right\|_2^2 + \mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t\right\|_2^2 + 2\mathbb{E}\left\langle \tilde{\mathbf{x}}_t - \mathbf{x}_*, \tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t \right\rangle \\
&= \mathbb{E}\left\|\tilde{\mathbf{x}}_t - \mathbf{x}_*\right\|_2^2 + \mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t\right\|_2^2 && (28) \\
&\quad + 2\mathbb{E}\left\langle \tilde{\mathbf{x}}_t - \mathbf{x}_*, -\eta \sum_{i=1}^{M}\sum_{k=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right) \right\rangle && (29) \\
&\quad + 2\mathbb{E}\left\langle \tilde{\mathbf{x}}_t - \mathbf{x}_*, \eta \sum_{i=1}^{M}\sum_{k=0}^{K_i-1} \omega_i \nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right) \right\rangle && (30) \\
&\quad + 2\mathbb{E}\left\langle \tilde{\mathbf{x}}_t - \mathbf{x}_*, \right. \\
&\qquad \left. -\eta \bar{K} \sum_{i, K_i \le \bar{K}} \frac{\omega_i}{K_i} \sum_{k=0}^{K_i-1} \left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right) - \nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right) \right\rangle && (31) \\
&\quad + 2\mathbb{E}\left\langle \tilde{\mathbf{x}}_t - \mathbf{x}_*, -\eta \bar{K} \nabla F\left(\tilde{\mathbf{x}}_{t-1}\right) \right\rangle && (32)
\end{aligned}$$

We denote Equation (29) to (32) by $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}_3$ and $\mathcal{Q}_4$, respec. There atívelyre two terms in Equation (29) and Equation (30), namely $\mathcal{Q}_1$ and $\mathcal{Q}_2$, between which the subscript is different (i.e., one for t-th update while the others for $t-1$-th update). The following will present how to bound $\mathcal{Q}_1$ first, and then $\mathcal{Q}_2$.

$$\mathcal{Q}_1 = \eta \sum_{i=1}^{M}\sum_{k=0}^{K_i-1} \omega_i \mathbb{E}\left\langle \nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right) - \nabla F_i\left(\tilde{\mathbf{x}}_t\right), \mathbf{x}_* - \tilde{\mathbf{x}}_t \right\rangle$$

$$
\begin{aligned}
&+ \eta \sum_{i=1}^{M} \omega_i K_i \mathbb{E} \left\langle \nabla F_i \left(\tilde{\mathbf{x}}_t \right), \mathbf{x}_* - \tilde{\mathbf{x}}_t \right\rangle \\
\leq\; & \frac{\eta}{2} \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \left(\frac{16}{\mu} \mathbb{E} \left\| \nabla F_i \left(\mathbf{x}_{t,k}^{(i)} \right) - \nabla F_i \left(\tilde{\mathbf{x}}_t \right) \right\|_2^2 + \frac{\mu}{16} \mathbb{E} \left\| \mathbf{x}_* - \tilde{\mathbf{x}}_t \right\|_2^2 \right) \\
&+ \eta \sum_{i=1}^{M} \omega_i K_i \mathbb{E} \left\langle \nabla F_i \left(\tilde{\mathbf{x}}_t \right), \mathbf{x}_* - \tilde{\mathbf{x}}_t \right\rangle \\
\leq\; & \frac{8\eta L^2}{\mu} \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \mathbb{E} \left\| \mathbf{x}_{t,k}^{(i)} - \mathbf{x}_* \right\|_2^2 + \frac{\eta \mu \bar{K}}{32} \mathbb{E} \left\| \mathbf{x}_* - \tilde{\mathbf{x}}_t \right\|_2^2 \\
&+ \eta \sum_{i=1}^{M} \omega_i K_i \mathbb{E} \left\langle \nabla F_i \left(\tilde{\mathbf{x}}_t \right), \mathbf{x}_* - \tilde{\mathbf{x}}_t \right\rangle
\end{aligned}
$$

where the first inequality refers to $\langle a, b \rangle \leq (\|a\|_2^2 + \|b\|_2^2)/2$ and the last one is according to Assumption 4.1. Likewise, we can find the bound for $\mathcal{Q}_2$:

$$
\begin{aligned}
\mathcal{Q}_2 \leq\; & \frac{8\eta L^2}{\mu} \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \mathbb{E} \left\| \mathbf{x}_{t-1,k}^{(i)} - \mathbf{x}_* \right\|_2^2 + \frac{\eta \mu \bar{K}}{32} \mathbb{E} \left\| \mathbf{x}_* - \tilde{\mathbf{x}}_t \right\|_2^2 \\
&+ \eta \sum_{i=1}^{M} \omega_i K_i \mathbb{E} \left\langle \nabla F_i \left(\tilde{\mathbf{x}}_{t-1} \right), \tilde{\mathbf{x}}_t - \mathbf{x}_* \right\rangle
\end{aligned}
$$

As a result, we have:

$$
\begin{aligned}
\mathcal{Q}_1 + \mathcal{Q}_2 \leq\; & \frac{8\eta L^2}{\mu} \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \left(\mathbb{E} \left\| \mathbf{x}_{t,k}^{(i)} - \mathbf{x}_* \right\|_2^2 + \mathbb{E} \left\| \mathbf{x}_{t-1,k}^{(i)} - \mathbf{x}_* \right\|_2^2 \right) \\
&+ \eta \sum_{i=1}^{M} \omega_i K_i \mathbb{E} \left\langle \nabla F_i \left(\tilde{\mathbf{x}}_{t-1} \right) - \nabla F_i \left(\tilde{\mathbf{x}}_t \right), \tilde{\mathbf{x}}_t - \mathbf{x}_* \right\rangle + \frac{\eta \mu \bar{K}}{16} \mathbb{E} \left\| \mathbf{x}_* - \tilde{\mathbf{x}}_t \right\|_2^2
\end{aligned}
$$

$$\begin{aligned}&\leq \frac{8\eta L^2}{\mu}\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\left(\mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)}-\mathbf{x}_*\right\|_2^2+\mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)}-\mathbf{x}_*\right\|_2^2\right)\\&+\frac{3\eta\mu\bar{K}}{32}\mathbb{E}\left\|\mathbf{x}_*-\tilde{\mathbf{x}}_t\right\|_2^2+\frac{8\eta\bar{K}L^2}{\mu}\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\tilde{\mathbf{x}}_{t-1}\right\|_2^2\end{aligned}$$

where the last inequality is based on $\langle a,b\rangle \leq (\|a\|_2^2+\|b\|_2^2)/2$ and Assumption 4.1.

For the term in Equation (31), according to the inequality that $\langle a,b\rangle \leq (\|a\|_2^2+\|b\|_2^2)/2$ and the assumption of L-smooth, we have:

$$\begin{aligned}\mathcal{Q}_3&=-\eta\bar{K}\sum_{i,K_i\leq\bar{K}}\frac{\omega_i}{K_i}\sum_{k=0}^{K_i-1}\mathbb{E}\left\langle\tilde{\mathbf{x}}_t-\mathbf{x}_*,\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right)-\nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right\rangle\\&\leq\frac{8\eta\bar{K}L^2}{\mu}\sum_{i,K_i\leq\bar{K}}\sum_{k=0}^{K_i-1}\frac{\omega_i}{K_i}\mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)}-\tilde{\mathbf{x}}_{t-1}\right\|_2^2+\frac{\eta\mu\bar{K}}{32}\mathbb{E}\left\|\mathbf{x}_*-\tilde{\mathbf{x}}_t\right\|_2^2\end{aligned}$$

The last equality holds because $\omega_i > 0$ and $K_i > 0$ such that the sum for those $K_i \leq \bar{K}$ is not greater than the one for all workers.

Based on Lemma 1 above, it is easy to derive the bound for $\mathcal{Q}_4$, which is:

$$\begin{aligned}\mathcal{Q}_4&=\eta\bar{K}\mathbb{E}\left\langle\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right),\mathbf{x}_*-\tilde{\mathbf{x}}_t\right\rangle\\&\leq\eta\bar{K}\left(F(\mathbf{x}_*)-F(\tilde{\mathbf{x}}_t)-\frac{\mu}{4}\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\mathbf{x}_*\right\|_2^2+L\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\tilde{\mathbf{x}}_{t-1}\right\|_2^2\right)\end{aligned}$$

According to the bound for $\mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1}-\tilde{\mathbf{x}}_t\right\|_2^2$ in the proof non-convex objectives, i.e.,

T_2, we have:

$$\mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1}-\tilde{\mathbf{x}}_t\right\|_2^2 \leq 2\eta^2\lambda^2\mathcal{Q}_5+3\eta^2\sigma^2\sum_{i=1}^{M}\omega_i^2K_i+3\eta^2\lambda^2\sigma^2\sum_{i=1}^{M}\omega_i^2\left(\frac{\bar{K}}{K_i}-1\right)^2K_i + 12\eta^2\lambda^2\sigma^2\sum_{i,K_i>\bar{K}}\frac{\omega_i^2\bar{K}^2}{K_i}$$

where

$$\mathcal{Q}_5=\mathbb{E}\left\|\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)-\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right) +\bar{K}\sum_{i,K_i\leq\bar{K}}\frac{\omega_i}{K_i}\sum_{k=0}^{K_i-1}\left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right)-\nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right)+\bar{K}\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right)\right\|_2^2 \tag{33}$$

Unlike the procedure in non-convex objectives, $\mathcal{Q}_5$ cannot be eliminated. Therefore, we should find a general bound for Equation (31), where we can further simplify as:

$$\begin{aligned}\mathcal{Q}_5=\mathbb{E}\Bigg\|&\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\left(\nabla F_i\left(\mathbf{x}_{t,k}^{(i)}\right)-\nabla F_i\left(\tilde{\mathbf{x}}_t\right)\right)\\&-\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_i\left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right)-\nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right)\\&+\bar{K}\sum_{i,K_i\leq\bar{K}}\frac{\omega_i}{K_i}\sum_{k=0}^{K_i-1}\left(\nabla F_i\left(\mathbf{x}_{t-1,k}^{(i)}\right)-\nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right)\\&+\bar{K}\left(\nabla F\left(\tilde{\mathbf{x}}_{t-1}\right)-\nabla F\left(\tilde{\mathbf{x}}_t\right)\right)+\bar{K}\nabla F\left(\tilde{\mathbf{x}}_t\right)\end{aligned}$$

$$
\begin{aligned}
&\quad + \sum_{i=1}^{M} \omega_i K_i \left(\nabla F_i\left(\tilde{\mathbf{x}}_t\right) - \nabla F_i\left(\tilde{\mathbf{x}}_{t-1}\right)\right)\Bigg\|_2^2 \\
&\le 6L^2 \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i K_i \mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)} - \tilde{\mathbf{x}}_t\right\|_2^2 + 6L^2 \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i K_i \mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)} - \tilde{\mathbf{x}}_{t-1}\right\|_2^2 \\
&\quad + 6\bar{K}^2 L^2 \sum_{i, K_i \le \bar{K}} \sum_{k=0}^{K_i-1} \frac{\omega_i}{K_i} \mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)} - \tilde{\mathbf{x}}_{t-1}\right\|_2^2 \\
&\quad + 12\bar{K}^2 L^2 \mathbb{E}\left\|\tilde{\mathbf{x}}_t - \tilde{\mathbf{x}}_{t-1}\right\|_2^2 + 6\bar{K}^2 L^2 \mathbb{E}\left\|\tilde{\mathbf{x}}_t - \mathbf{x}_*\right\|_2^2
\end{aligned}
$$

Plugging the results above, we can obtain the bound for $\mathbb{E}\|\tilde{\mathbf{x}}_{t+1} - \mathbf{x}_*\|_2^2$ as:

$$
\begin{aligned}
&\mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1} - \mathbf{x}_*\right\|_2^2 \\
\le &\left(1 - \frac{\eta\mu\bar{K}}{4}\right) \mathbb{E}\left\|\tilde{\mathbf{x}}_t - \mathbf{x}_*\right\|_2^2 \\
&+ 12\eta^2\lambda^2 L^2 \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i K_i \mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)} - \tilde{\mathbf{x}}_t\right\|_2^2 \\
&+ 12\eta^2\lambda^2 L^2 \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i K_i \mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)} - \tilde{\mathbf{x}}_{t-1}\right\|_2^2 \\
&+ 12\eta^2\lambda^2 \bar{K}^2 L^2 \sum_{i, K_i \le \bar{K}} \sum_{k=0}^{K_i-1} \frac{\omega_i}{K_i} \mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)} - \tilde{\mathbf{x}}_{t-1}\right\|_2^2 \\
&+ 24\eta^2\lambda^2 \bar{K}^2 L^2 \mathbb{E}\left\|\tilde{\mathbf{x}}_t - \tilde{\mathbf{x}}_{t-1}\right\|_2^2 + 12\eta^2\lambda^2 \bar{K}^2 L^2 \mathbb{E}\left\|\tilde{\mathbf{x}}_t - \mathbf{x}_*\right\|_2^2 \\
&+ 3\eta^2\sigma^2 \sum_{i=1}^{M} \omega_i^2 K_i + 3\eta^2\lambda^2\sigma^2 \sum_{i=1}^{M} \omega_i^2 \left(\frac{\bar{K}}{K_i} - 1\right)^2 K_i + 12\eta^2\lambda^2\sigma^2 \sum_{i, K_i > \bar{K}} \frac{\omega_i^2 \bar{K}^2}{K_i} \\
&+ \frac{16\eta L^2}{\mu} \sum_{i=1}^{M} \sum_{k=0}^{K_i-1} \omega_i \left(\mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)} - \mathbf{x}_*\right\|_2^2 + \mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)} - \mathbf{x}_*\right\|_2^2\right)
\end{aligned}
$$

$$+\frac{16\eta\bar{K}L^2}{\mu}\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\tilde{\mathbf{x}}_{t-1}\right\|_2^2+\frac{16\eta\bar{K}L^2}{\mu}\sum_{i,K_i\leq\bar{K}}\sum_{k=0}^{K_i-1}\frac{\omega_i}{K_i}\mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)}-\tilde{\mathbf{x}}_{t-1}\right\|_2^2$$
$$+2\eta\bar{K}\left(F(\mathbf{x}_*)-F(\tilde{\mathbf{x}}_t)+L\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\tilde{\mathbf{x}}_{t-1}\right\|_2^2\right)$$

By observation, there are two recursive formulas, i.e., $\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\tilde{\mathbf{x}}_{t-1}\right\|_2^2$ and $\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_iK_i\mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)}-\tilde{\mathbf{x}}_{t-1}\right\|_2^2$, and the coefficients of both of them contain the stepsize η. To release these formulas, we let the formula at the left hand side be the formula as follows:

$$\mathcal{Y}_{t+1}=\mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1}-\mathbf{x}_*\right\|_2^2+u_1\mathbb{E}\left\|\tilde{\mathbf{x}}_{t+1}-\tilde{\mathbf{x}}_t\right\|_2^2+u_2\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_iK_i\mathbb{E}\left\|\mathbf{x}_{t,k}^{(i)}-\tilde{\mathbf{x}}_t\right\|_2^2$$

where u_1 and u_2 are the coefficients containing the stepsize of η such that the coefficient for $\mathbb{E}\left\|\tilde{\mathbf{x}}_t-\tilde{\mathbf{x}}_{t-1}\right\|_2^2$ and $\sum_{i=1}^{M}\sum_{k=0}^{K_i-1}\omega_iK_i\mathbb{E}\left\|\mathbf{x}_{t-1,k}^{(i)}-\tilde{\mathbf{x}}_{t-1}\right\|_2^2$ on the right hand side become negative and thereby, can be omitted when finding the bound. Also, we notice that the added term includes η^2 after simplification. Therefore, the formula can be further simplified as:

$$\begin{aligned}\mathcal{Y}_{t+1}\leq&\left(1-\frac{\eta\mu\bar{K}}{4}\right)\mathcal{Y}_t-2\eta\bar{K}(F(\tilde{\mathbf{x}}_t)-F(\mathbf{x}_*))\\&+6\eta^2\sigma^2\sum_{i=1}^{M}\omega_i^2K_i+6\eta^2\lambda^2\sigma^2\sum_{i=1}^{M}\omega_i^2\left(\frac{\bar{K}}{K_i}-1\right)^2K_i\\&+24\eta^2\lambda^2\sigma^2\sum_{i,K_i>\bar{K}}\frac{\omega_i^2\bar{K}^2}{K_i}\end{aligned}$$

The rest step is similar to the proof in Theorem 4.1. Therefore, we can obtain the expected result in Theorem 4.3. Different from Theorem 4.1, this theorem release the term of data heterogeneity and therefore, our result can successfully converge to the global optimizer.