EFFICIENT AND SECURE MESSAGE PASSING FOR MACHINE LEARNING

By

Xiaorui Liu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2022

ABSTRACT

EFFICIENT AND SECURE MESSAGE PASSING FOR MACHINE LEARNING

By

Xiaorui Liu

Machine learning (ML) techniques have brought revolutionary impact to human society, and they will continue to act as technological innovators in the future. To broaden its impact, it is urgent to solve the emerging and critical challenges in machine learning, such as efficiency and security issues. On the one hand, ML models have become increasingly powerful due to big data and models, but it also brings tremendous challenges in designing efficient optimization algorithms to train the big ML models from big data. The most effective way for large-scale ML is to parallelize the computation tasks on distributed systems composed of many computational devices. However, in practice, the scalability and efficiency of the systems are greatly limited by information synchronization since the message passing between the devices dominates the total running time. In other words, the major bottleneck lies in the high communication cost between devices, especially when the scale of the system and the models becomes larger while the communication bandwidth is relatively limited. This communication bottleneck often limits the practical speedup of distributed ML systems.

On the other hand, recent research has generally revealed that many ML models suffer from security vulnerabilities. In particular, deep learning models can be easily deceived by the unnoticeable perturbations in data. Meanwhile, graph is a kind of prevalent data structure for many real-world data that encodes pairwise relations between entities such as social networks, transportation networks, and chemical molecules. Graph neural networks (GNNs) generalize and extend the representation learning power of traditional deep neural networks (DNNs) from regular grids, such as image, video, and text, to irregular graph-structured data through message passing frameworks. Therefore, many important applications on these data can be treated as computational tasks on graphs, such as recommender systems, social network analysis, traffic prediction, etc. Unfortunately, the vulnerability of deep learning models also translates to GNNs, which raises significant concerns about their

applications, especially in safety-critical areas. Therefore, it is critical to design intrinsically secure ML models for graph-structured data.

The primary objective of this dissertation is to figure out the solutions to solve these challenges via innovative research and principled methods. In particular, we propose multiple distributed optimization algorithms with efficient message passing to mitigate the communication bottleneck and speed up ML model training in distributed ML systems. We also propose multiple secure message passing schemes as the building blocks of graph neural networks aiming to significantly improve the security and robustness of ML models.

To my parents and family for their love and support.

ACKNOWLEDGEMENTS

This dissertation is impossible without the help and support from my advisor Dr. Jiliang Tang. I would like to express my deepest appreciation to him for his invaluable advice, inspiration, encouragement, and support during my Ph.D. research. I have learned many skills from him that benefit my life: how to identify important and challenging research problems, how to write papers and give presentations, how to collaborate, how to mentor students, how to write proposals, how to volunteer and serve the community, how to establish my career and build a big vision. Dr. Tang is not only an advisor in research but also a sincere friend who guided me with his experience in many aspects of my life. Dr. Tang, I cannot thank you enough.

I would like to thank my committee members, Dr. Ming Yan, Dr. Anil K. Jain, and Dr. Charu Aggarwal, for their helpful suggestions and insightful comments. I enrolled in the Optimization course from Dr. Ming Yan in my first year at MSU, which prepared me with a solid technical background and benefited my Ph.D. research a lot. I always consider Dr. Ming Yan as my secondary advisor because many of my research ideas were initialized with his insightful discussions and a large portion of this dissertation was done with his collaboration. Dr. Anil K. Jain provided numerous suggestions for my research and career, and his pursuit of broader impact greatly reshapes my research vision. I was fortunate to collaborate with Dr. Charu Aggarwal on multiple research papers, and I am impressed by his knowledge and insights. In addition, his dedication to book writing and online education is very inspiring to me.

I was fortunate to work as an intern at JD.com, Kwai AI Lab, and TAL AI Lab with amazing colleagues and mentors: Dr. Dawei Yin, Dr. Hongshen Chen, Dr. Ziheng Jiang, and Dr. Zhaochun Ren from JD.com; Dr. Ji Liu and Dr. Xiangru Lian from Kwai AI Lab; and Dr. Zitao Liu from TAL AI Lab. I enjoyed the productive and wonderful summers with you.

During my Ph.D. study, many friends and colleagues provided me with consistent support, encouragement, and happiness. I am thankful to my friends and colleagues from the Data Science and Engineering Lab: Dr. Tyler Derr, Dr. Zhiwei Wang, Dr. Wenqi Fan, Dr. Xiangyu Zhao, Dr.

Hamid Karimi, Dr. Yao Ma, Chenxing Wang, Daniel K.O-Dankwa, Hansheng Zhao, Haochen Liu, Han Xu, Jamell Dacon, Wentao Wang, Wei Jin, Yaxin Li, Yiqi Wang, Juanhui Li, Harry Shomer, Jie Ren, Jiayuan Ding, Haoyu Han, Hongzhi Wen, Yuxuan Wan, Pengfei He, Hua Liu, Dr. Xin Wang, Dr. Jiangtao Huang, Dr. Xiaoyang Wang, Dr. Meznah Almutairy, Namratha Shah, Norah Alfadhli. In particular, thanks to my DSE collaborators: Dr. Tyler Derr, Dr. Zhiwei Wang, Dr. Wenqi Fan, Dr. Hamid Karimi, Dr. Xiangyu Zhao, Dr. Yao Ma, Haochen Liu, Han Xu, Wentao Wang, Wei Jin, Yaxin Li, Yiqi Wang, Jiayuan Ding, Haoyu Han, Hua Liu, and Yuxuan Wan. I would like to extend my sincere thanks to all my collaborators from outside the Data Science and Engineering Lab: Dr. Ming Yan, Dr. Yao Li, Dr. Rongrong Wang, Dr. Charu Aggarwal, Dr. Anil K. Jain, Dr. Dawei Yin, Dr. Hongshen Chen, Dr. Qing Li, Dr. Suhang Wang, Dr. Xianfeng Tang, Dr. Hui Liu, Dr. Zitao Liu, Dr. Kuan Yuan, and Dr. Neil Shah.

Finally, I would like to thank my family for their love and support. I also dedicate this dissertation to Xinyi Lu for supporting me all the way!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

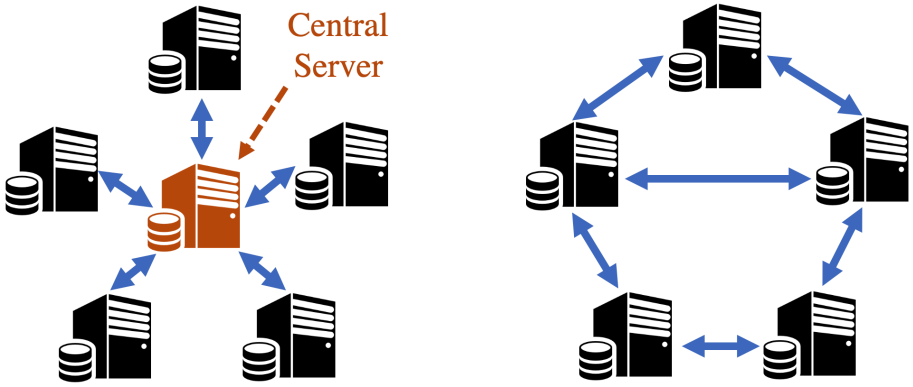
CHAPTER 1

INTRODUCTION

Machine learning (ML) techniques have brought revolutionary impact to human society, and they will continue to act as technological innovators in the future. In recent years, critical challenges in machine learning such as efficiency and security issues broadly emerge. These issues greatly limit the applications of ML techniques in many scientific and application domains. To broaden the impact of ML techniques, it is urgent to solve these emerging and critical research challenges.

On the one hand, ML models have become increasingly powerful due to big data and models, but it also brings tremendous challenges in designing efficient optimization algorithms to train the big ML models from big data. The most effective way for large-scale ML is to parallelize the computation tasks on distributed systems composed of many computational devices. There are two mainstream trends of distributed and parallel ML systems: (1) large-scale ML models trained by powerful distributed computing systems in data centers; and (2) on-device distributed training by resourced-limited edge devices (e.g., smartphones, AR/VR headsets, drones, billions of Internet of Things) as well as the massive amount of data they generate on a daily basis. In both cases, the scalability and efficiency of the systems are greatly limited since the slow information synchronization between the devices dominates the total running time. In other words, the major bottleneck lies in the high communication cost between devices, especially when the scale of the system and the models becomes larger while the communication bandwidth is relatively limited. For instance, in centralized distributed ML systems as shown in Figure 1.1a, every computing node needs to frequently synchronize information with other nodes through the central server by passing the message (e.g., model or gradient information) [4, 92, 114, 68]. But this message passing process becomes dominating when the communication network bandwidth is limited and the number of computing nodes is massive. In decentralized distributed ML systems as shown in Figure 1.1b, every computing node only needs to synchronize information with the one-hope neighbors by passing the message (e.g., the model information). Although it is more scalable in

terms of the number of computing nodes, the communication bottleneck still exists under limited network bandwidth [101, 51, 50, 75]. The communication bottleneck often limits the theoretical speedup of distributed ML systems. Therefore, how to design distributed learning algorithms and systems with efficient message passing becomes a promising and key research direction for solving the efficiency challenge in ML.



(a) Centralized Learning (b) Decentralized Learning

Figure 1.1: Distributed ML systems.

On the other hand, recent research has generally revealed that many ML models suffer from security vulnerabilities. In particular, deep learning models can be easily deceived by the unnoticeable perturbations in data [30, 99, 21]. Meanwhile, graph is a kind of prevalent data structure for many real-world data that encodes pairwise relations between entities such as social networks, transportation networks, and chemical molecules. Graph neural networks (GNNs) generalize and extend the representation learning power of traditional deep neural networks (DNNs) from regular grids, such as image, video, and text, to irregular graph-structured data as shown in Figure 1.2. Therefore, many important applications on these data can be treated as computational tasks on graphs [74, 32]. For instance, product recommendation in e-commerce and friend recommendation in social network analysis can be formulated as link prediction tasks on graphs; Traffic prediction in transportation systems can be formulated as node classification or regression on graphs. The key building block for such generalization is the message passing framework that propagates features from neighboring nodes in the graph. The message passing layer offers the node permutation invariance

and the support for arbitrary neighboring sizes in graphs. Despite the promising performance of GNNs in clean data settings, unfortunately, the vulnerability of deep learning models also translates to GNNs [134, 135, 37, 41] when the data contains adversarial perturbations. For instance, the performance greatly degrades when the node features or graph structure are modified by adversaries [41]. These raise significant concerns about the applications of GNNs, especially in safety-critical areas. Therefore, it is critical to design intrinsically secure ML models for graph-structured data.

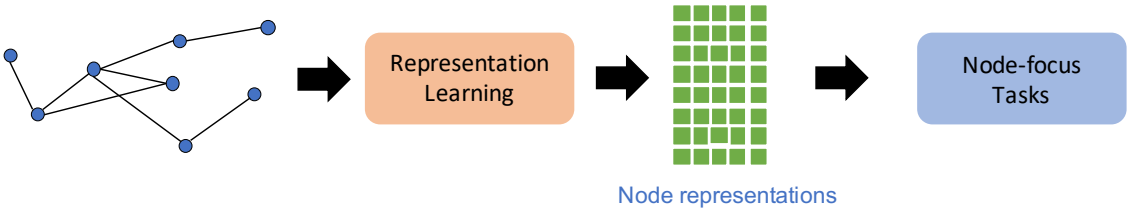


Figure 1.2: Graph representation learning for node-focus tasks.

1.1 Research Challenges

From the above research background, we can summarize the research challenges as follows:

- How to design scalable distributed ML systems and algorithms with efficient message passing between computing devices such that the communication bottleneck can be largely mitigated?
- How to maintain the convergence behaviors of the optimization algorithm when communication efficiency is improved both theoretically and empirically?
- How to design intrinsically secure ML models that are more robust to potential threats such as feature or graph structure attacks by adversaries?
- How to bypass the tradeoff between the performance under clean and adversarial settings? In other words, can we maintain good performance when the data are clean while providing strong security when the data are adversarially perturbed?

1.2 Contributions

The primary objective of this dissertation is to figure out the solutions to solve these challenges via innovative research and principled methods. In particular, we propose multiple distributed optimization algorithms with efficient message passing to mitigate the communication bottleneck and speed up ML model training in distributed ML systems. We also propose multiple secure message passing schemes as the building blocks of graph neural networks aiming to significantly improve the security and robustness of ML models. The contributions of this dissertation are summarized as:

- To fundamentally improve the efficiency of distributed ML systems, I proposed a series of innovative algorithms to break through the communication bottleneck. In particular, when the communication network is a start network as shown in Figure 1.1a, I proposed DORE [68], a double residual compression algorithm, to compress the bi-directional communication between client devices and the server such that over 95% of the communication bits can be reduced. This is the first algorithm that reduces that much communication cost while maintaining the superior convergence complexities (e.g., linear convergence) as the uncompressed counterpart, both theoretically and numerically.

- When the communication network is of any general topology (as long as it is connected) as shown in Figure 1.1b, I proposed LEAD [69], the first linear convergent decentralized optimization algorithm with communication compression, which only requires point-to-point compressed communication between neighboring devices over communication networks. Theoretically, we prove that under certain compression ratios, the convergence complexity of the proposed algorithm does not depend on the compression operator. In other words, it achieves better communication efficiency for free.

- To design intrinsically secure ML models against feature attacks, I investigate to denoise the hidden features in neural network layers caused by the adversarial perturbation using the graph structural information. This is achieved by the proposed AirGNN [66] in which the adaptive

message passing denoises perturbed features by feature aggregations and maintains feature separability by adaptive residuals. The proposed algorithm has a clear design principle and interpretation as well strong as performance both in the clean and adversarial data settings.

- To design intrinsically secure ML models against graph structure attacks, I investigate a new prior knowledge of smoothness in the design of graph neural networks. In particular, we derive an elastic message passing scheme to model the piecewise constant signal in graph data. We demonstrate its stronger resilience to adversarial structure attacks and superior performance when the data is clean through a comprehensive empirical study on the proposed model ElasticGNN [67].

1.3 Organization

The remainder of this dissertation is organized as follows. In Chapter 2, we introduce DORE, a centralized distributed optimization algorithm with communication compression. In Chapter 3, we introduce LEAD, the first linear convergent decentralized distributed optimization algorithm with communication compression. In these two chapters, we demonstrate how to significantly improve the efficiency and scalability of distributed ML systems by the co-design of efficient message passing and optimization algorithms. In Chapter 4, we investigate the possibility to utilize the graph structural information in defending against abnormal features with noise or adversarial perturbations. We derive a novel adaptive message passing scheme from a principled graph signal denoising perspective. In Chapter 5, we study a new smoothness prior knowledge, i.e., piecewise constant signal, for graph representation learning. We derive the elastic message passing to model the adaptive local smoothness in graph data. In these two chapters, we demonstrate how these secure message passing algorithms can be used as fundamental building blocks in the design of graph neural networks to defend against feature and graph structure attacks through the examples of AirGNN and ElasticGNN, respectively. We conclude the dissertation and discuss the broader impact and promising research directions in Chapter 6.

CHAPTER 2

A DOUBLE RESIDUAL COMPRESSION ALGORITHM FOR DISTRIBUTED LEARNING

Large-scale machine learning models are often trained by parallel stochastic gradient descent algorithms. However, the message passing cost of gradient aggregation and model synchronization between the master and worker nodes becomes the major obstacle for efficient learning as the number of workers and the dimension of the model increase. In this chapter, we propose DORE, a DOuble REsidual compression stochastic gradient descent algorithm, to reduce over 95% of the overall communication in message passing such that the obstacle can be immensely mitigated. Our theoretical analyses demonstrate that the proposed strategy has superior convergence properties for both strongly convex and nonconvex objective functions. The experimental results validate that DORE achieves the best communication efficiency while maintaining similar model accuracy and convergence speed in comparison with start-of-the-art baselines.

2.1 Introduction

Stochastic gradient algorithms [8] are efficient at minimizing the objective function $f : \mathbb{R}^d \to \mathbb{R}$ which is usually defined as $f(\mathbf{x}) := \mathbb{E}_{\xi\sim\mathcal{D}}[\ell(\mathbf{x}, \xi)]$, where $\ell(\mathbf{x}, \xi)$ is the objective function defined on data sample ξ and model parameter $\mathbf{x}$. A basic stochastic gradient descent (SGD) repeats the gradient "descent" step $\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma\mathbf{g}(\mathbf{x}^k)$ where $\mathbf{x}_k$ is the current iteration and γ is the step size. The stochastic gradient $\mathbf{g}(\mathbf{x}^k)$ is computed based on an i.i.d. sampled mini-batch from the distribution of the training data $\mathcal{D}$ and serves as the estimator of the full gradient $\nabla f(\mathbf{x}^k)$. In the context of large-scale machine learning, the number of data samples and the model size are usually very large. Distributed learning utilizes a large number of computers/cores to perform the stochastic algorithms aiming at reducing the training time. It has attracted extensive attention due to the demand for highly efficient model training [1, 17, 54, 122].

In this work, we focus on the data-parallel SGD [22, 61, 133], which provides a scalable solution to speed up the training process by distributing the whole data to multiple computing nodes. The

objective can be written as:

$$\underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}}\, f(\mathbf{x}) + R(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi\sim\mathcal{D}_i}[\ell(\mathbf{x},\xi)]}_{:=f_i(\mathbf{x})} + R(\mathbf{x}),$$

where each $f_i(\mathbf{x})$ is a local objective function of the worker node i defined based on the allocated data under distribution $\mathcal{D}_i$ and $R : \mathbb{R}^d \to \mathbb{R}$ is usually a closed convex regularizer.

In the well-known parameter server framework [54, 133], during each iteration, each worker node evaluates its own stochastic gradient $\{\widetilde{\nabla} f_i(\mathbf{x}^k)\}_{i=1}^n$ and send it to the master node, which collects all gradients and calculates their average $(1/n)\sum_{i=1}^n \widetilde{\nabla} f_i(\mathbf{x}^k)$. Then the master node further takes the gradient descent step with the averaged gradient and broadcasts the new model parameter $\mathbf{x}^{k+1}$ to all worker nodes. It makes use of the computational resources from all nodes. In reality, the network bandwidth is often limited. Thus, the communication cost for the gradient transmission and model synchronization becomes the dominating bottlenecks as the number of nodes and the model size increase, which hinders the scalability and efficiency of SGD.

One common way to reduce the communication cost is to compress the gradient information by either gradient sparsification or quantization [4, 92, 97, 98, 110, 112, 114, 116] such that many fewer bits of information are needed to be transmitted. However, little attention has been paid on how to reduce the communication cost for model synchronization and the corresponding theoretical guarantees. Obviously, the model shares the same size as the gradient, so does the communication cost. Thus, merely compressing the gradient can reduce at most 50% of the communication cost, which suggests the importance of model compression. Notably, the compression of model parameters is much more challenging than gradient compression. One key obstacle is that its compression error cannot be well controlled by the step size γ and thus it cannot diminish like that in the gradient compression [101]. In this work, we aim to bridge this gap by investigating algorithms to compress the full communication in the optimization process and understanding their theoretical properties. Our contributions can be summarized as:

- We proposed DORE, which can compress both the gradient and the model information such that more than 95% of the communication cost can be reduced.

- We provided theoretical analyses to guarantee the convergence of DORE under strongly convex and nonconvex assumptions without the bounded gradient assumption.
- Our experiments demonstrate the superior efficiency of DORE comparing with the state-of-art baselines without degrading the convergence speed and the model accuracy.

2.2 Related Work

Recently, many works try to reduce the communication cost to speed up the distributed learning, especially for deep learning applications, where the size of the model is typically very large (so is the size of the gradient) while the network bandwidth is relatively limited. Below we briefly review relevant papers.

Gradient quantization and sparsification. Recent works [4, 92, 114, 77, 7] have shown that the information of the gradient can be quantized into a lower-precision vector such that fewer bits are needed in communication without loss of accuracy. [92] proposed 1Bit SGD that keeps the sign of each element in the gradient only. It empirically works well, and [7] provided theoretical analysis systematically. QSGD [4] utilizes an unbiased multi-level random quantization to compress the gradient while Terngrad [114] quantizes the gradient into ternary numbers $\{0, \pm 1\}$. In DIANA [77], the gradient difference is compressed and communicated contributing to the estimator of the gradient in the master node.

Another effective strategy to reduce the communication cost is sparsification. [112] proposed a convex optimization formulation to minimize the coding length of stochastic gradients. A more aggressive sparsification method is to keep the elements with relatively larger magnitude in gradients, such as top-k sparsification [97, 98, 3].

Model synchronization. The typical way for model synchronization is to broadcast model parameters to all worker nodes. Some works [110, 42] have been proposed to reduce model size by enforcing sparsity, but it cannot be applied to general optimization problems. Some alternatives including QSGD [4] and ECQ-SGD [116] choose to broadcast all quantized gradients to all other workers such that every worker can perform model update independently. However, all-to-all

communication is not efficient since the number of transmitted bits increases dramatically in large-scale networks. DoubleSqueeze [104] applies compression on the averaged gradient with error compensation to speed up model synchronization.

Error compensation. [92] applied error compensation on 1Bit-SGD and achieved negligible loss of accuracy empirically. Recently, error compensation was further studied [116, 97, 45] to mitigate the error caused by compression. The general idea is to add the compressed error to the next compression step:

$$\hat{\mathbf{g}} = Q(\mathbf{g} + \mathbf{e}), \quad \mathbf{e} = (\mathbf{g} + \mathbf{e}) - \hat{\mathbf{g}}.$$

However, to the best of our knowledge, most of the algorithms with error compensation [116, 97, 45, 104] need to assume bounded gradient, i.e., $\mathbb{E}\|\mathbf{g}\|^2 \leq B$, and the convergence rate depends on this bound.

Contributions of DORE. The most related papers to DORE are DIANA [77] and DoubleSqueeze [104]. Similarly, DIANA compresses gradient difference on the worker side and achieves good convergence rate. However, it doesn't consider the compression in model synchronization, so at most 50% of the communication cost can be saved. DoubleSqueeze applies compression with error compensation on both worker and server sides, but it only considers non-convex objective functions. Moreover, its analysis relies on a bounded gradient assumption, i.e., $\mathbb{E}\|\mathbf{g}\|^2 \leq B$, and the convergence error has a dependency on the gradient bound like most existed error compensation works.

In general, the uniform bound on the norm of the stochastic gradient is a strong assumption which might not hold in some cases. For example, it is violated in the strongly convex case [82, 31]. In this work, we design DORE, the first algorithm which utilizes gradient and model compression with error compensation without assuming bounded gradients. Unlike existing error compensation works, we provide a linear convergence rate to the $O(\sigma)$ neighborhood of the optimal solution for strongly convex functions and a sublinear rate to the stationary point for nonconvex functions with linear speedup. In Table 2.1, we compare the asymptotic convergence rates of different quantized SGDs with DORE.

2.3 Algorithm

In this section, we introduce the proposed DOuble REsidual compression SGD (DORE) algorithm. Before that, we introduce a common assumption for the compression operator.

In this work, we adopt an assumption from [4, 114, 77] that the compression variance is linearly proportional to the magnitude.

Assumption 1. *The stochastic compression operator* $Q : \mathbb{R}^d \to \mathbb{R}^d$ *is unbiased, i.e.,* $\mathbb{E}Q(\mathbf{x}) = \mathbf{x}$ *and satisfies*

$$\mathbb{E}\|Q(\mathbf{x}) - \mathbf{x}\|^2 \leq C\|\mathbf{x}\|^2, \tag{2.1}$$

for a nonnegative constant C *that is independent of* $\mathbf{x}$*. We use* $\hat{\mathbf{x}}$ *to denote the compressed* $\mathbf{x}$*, i.e.,* $\hat{\mathbf{x}} \sim Q(\mathbf{x})$.

Many feasible compression operators can be applied to our algorithm since our theoretical analyses are built on this common assumption. Some examples of feasible stochastic compression operators include:

- *No Compression:* $C = 0$ when there is no compression.
- *Stochastic Quantization:* A real number $x \in [a, b]$, $(a < b)$ is set to be a with probability $\frac{b-x}{b-a}$ and b with probability $\frac{x-a}{b-a}$, where a and b are predefined quantization levels [4]. It satisfies Assumption 1 when $ab > 0$ and $a < b$.
- *Stochastic Sparsification:* A real number x is set to be 0 with probability $1 - p$ and $\frac{x}{p}$ with probability p [114]. It satisfies Assumption 1 with $C = (1/p) - 1$.
- *p-norm Quantization:* A vector $\mathbf{x}$ is quantized element-wisely by $Q_p(\mathbf{x}) = \|\mathbf{x}\|_p \operatorname{sign}(\mathbf{x}) \circ \xi$, where $\circ$ is the Hadamard product and ξ is a Bernoulli random vector satisfying $\xi_i \sim \text{Bernoulli}(\frac{|x_i|}{\|\mathbf{x}\|_p})$. It satisfies Assumption 1 with $C = \max_{\mathbf{x}\in\mathbb{R}^d} \frac{\|\mathbf{x}\|_1 \|\mathbf{x}\|_p}{\|\mathbf{x}\|_2^2} - 1$ [77]. To decrease the constant C for a higher accuracy, a vector $\mathbf{x} \in \mathbb{R}^d$ can be further decomposed into blocks, i.e., $\mathbf{x} = (\mathbf{x}(1)^\top, \mathbf{x}(2)^\top, \cdots, \mathbf{x}(m)^\top)^\top$ with $\mathbf{x}(l) \in \mathbb{R}^{d_l}$ and $\sum_{l=1}^m d_l = d$, and the blocks can be compressed independently.

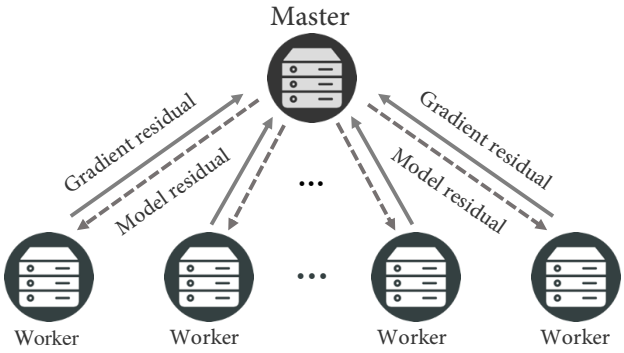


Figure 2.1: An Illustration of DORE.

2.3.1 Proposed Algorithm

Many previous works [4, 92, 114] reduce the communication cost of P-SGD by quantizing the stochastic gradient before sending it to the master node, but there are several intrinsic issues.

First, these algorithms will incur extra optimization error intrinsically. Let's consider the case when the algorithm converges to the optimal point $\mathbf{x}^*$ where we have $(1/n)\sum_{i=1}^n \nabla f_i(\mathbf{x}^*) = \mathbf{0}$. However, the data distributions may be different for different worker nodes in general, and thus we may have $\nabla f_i(\mathbf{x}^*) \neq \nabla f_j(\mathbf{x}^*), \forall i, j \in \{1, \ldots, n\}$ and $i \neq j$. In other words, each individual $\nabla f_i(\mathbf{x}^*)$ may be far away from zero. This will cause large compression variance according to Assumption 1, which indicates that the upper bound of compression variance $\mathbb{E}\|Q(\mathbf{x}) - \mathbf{x}\|^2$ is linearly proportional to the magnitude of $\mathbf{x}$.

Second, most existing algorithms [92, 4, 114, 7, 116, 77] need to broadcast the model or gradient to all worker nodes in each iteration. It is a considerable bottleneck for efficient optimization since the amount of bits to transmit is the same as the uncompressed gradient. DoubleSqueeze [104] is able to apply compression on both worker and server sides. However, its analysis depends on a strong assumption on bounded gradient. Meanwhile, no theoretical guarantees are provided for the convex problems.

We proposed DORE to address all aforementioned issues. Our motivation is that the gradient should change smoothly for smooth functions so that each worker node can keep a state variable $\mathbf{h}_i^k$ to track its previous gradient information. As a result, the residual between new gradient and the

Algorithm 1 DORE

1: **Input:** Stepsize $\alpha, \beta, \gamma, \eta$, initialize $\mathbf{h}^0 = \mathbf{h}_i^0 = \mathbf{0}^d$, $\hat{\mathbf{x}}_i^0 = \hat{\mathbf{x}}^0,\ \forall i \in \{1, \ldots, n\}$.
2: **for** $k = 1, 2, \cdots, K-1$ **do**

3: **For each worker** $i \in \{1, 2, \cdots, n\}$:
4: Sample $\mathbf{g}_i^k$ such that $\mathbb{E}[\mathbf{g}_i^k | \hat{\mathbf{x}}_i^k] = \nabla f_i(\hat{\mathbf{x}}_i^k)$
5: Gradient residual: $\Delta_i^k = \mathbf{g}_i^k - \mathbf{h}_i^k$
6: Compression: $\hat{\Delta}_i^k = Q(\Delta_i^k)$
7: $\mathbf{h}_i^{k+1} = \mathbf{h}_i^k + \alpha \hat{\Delta}_i^k$
8: $\{\ \hat{\mathbf{g}}_i^k = \mathbf{h}_i^k + \hat{\Delta}_i^k\ \}$
9: Send $\hat{\Delta}_i^k$ to the master
10: Receive $\hat{\mathbf{q}}^k$ from the master
11: $\hat{\mathbf{x}}_i^{k+1} = \hat{\mathbf{x}}_i^k + \beta \hat{\mathbf{q}}^k$

23: **end for**
24: **Output:** $\hat{\mathbf{x}}^K$ or any $\hat{\mathbf{x}}_i^K$

12: **For the master**:
13: Receive $\{\hat{\Delta}_i^k\}$ from workers
14: $\hat{\Delta}^k = 1/n \sum_i^n \hat{\Delta}_i^k$
15: $\hat{\mathbf{g}}^k = \mathbf{h}^k + \hat{\Delta}^k\ \ \{= 1/n \sum_i^n \hat{\mathbf{g}}_i^k\ \}$
16: $\mathbf{x}^{k+1} = \mathbf{prox}_{\gamma R}(\hat{\mathbf{x}}^k - \gamma \hat{\mathbf{g}}^k)$
17: $\mathbf{h}^{k+1} = \mathbf{h}^k + \alpha \hat{\Delta}^k$
18: Model residual: $\mathbf{q}^k = \mathbf{x}^{k+1} - \hat{\mathbf{x}}^k + \eta \mathbf{e}^k$
19: Compression: $\hat{\mathbf{q}}^k = Q(\mathbf{q}^k)$
20: $\mathbf{e}^{k+1} = \mathbf{q}^k - \hat{\mathbf{q}}^k$
21: $\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + \beta \hat{\mathbf{q}}^k$
22: Broadcast $\hat{\mathbf{q}}^k$ to workers

state $\mathbf{h}_i^k$ should decrease, and the compression variance of the residual can be well bounded. On the other hand, as the algorithm converges, the model would only change slightly. Therefore, we propose to compress the model residual such that the compression variance can be minimized and also well bounded. We also compensate the model residual compression error into next iteration to achieve a better convergence. Due to the advantages of the proposed double residual compression scheme, we can derive the fastest convergence rate through analyses without the bounded gradient assumption. Note that in Algorithm 1, equations in the curly bracket are just notations for the proof but does not need to computed actually. Below are some key steps of our algorithm as showed in Algorithm 1 and Figure 2.1:

[lines 4-9]: each worker node sends the compressed gradient residual ($\hat{\Delta}_i^k$) to the master node and updates its state $\mathbf{h}_i^k$ with $\hat{\Delta}_i^k$;

[lines 13-15]: the master node gathers the compressed gradient residual ($\{\hat{\Delta}_i^k\}$) from all worker nodes and recovers the averaged gradient $\hat{\mathbf{g}}^k$ based on its state $\mathbf{h}^k$;

[lines 16]: the master node applies gradient descent algorithms (possibly with the proximal operator);

[lines 18-22]: the master node broadcasts the compressed model residual with error compensation ($\hat{\mathbf{q}}^k$) to all worker nodes and updates the model;

[lines 10-11]: each worker node receives the compressed model residual ($\hat{\mathbf{q}}^k$) and updates its model $\mathbf{x}_i^k$.

In the algorithm, the state $\mathbf{h}_i^k$ serves as an exponential moving average of the local gradient in expectation, i.e., $\mathbb{E}_Q \mathbf{h}_i^{k+1} = (1-\alpha)\mathbf{h}_i^k + \alpha \mathbf{g}_i^k$, as proved in Lemma 7. Therefore, as the iteration approaches the optimum, $\mathbf{h}_i^k$ will also approach the local gradient $\nabla f_i(\mathbf{x}^*)$ rapidly which contributes to small gradient residual and consequently small compression variance. Similar difference compression techniques are also proposed in DIANA and its variance-reduced variant [77, 36].

2.3.2 Discussion

In this subsection, we provide more detailed discussions about DORE including model initialization, model update, the special smooth case as well as the compression rate of communication.

Initialization. It is important to take the identical initialization $\hat{\mathbf{x}}^0$ for all worker and master nodes. It is easy to be ensured by either setting the same random seed or broadcasting the model once at the beginning. In this way, although we don't need to broadcast the model parameters directly, every worker node updates the model $\hat{\mathbf{x}}^k$ in the same way. Thus we can keep their model parameters identical. Otherwise, the model inconsistency needs to be considered.

Model update. It is worth noting that although we can choose an accurate model $\mathbf{x}^{k+1}$ as the next iteration in the master node, we use $\hat{\mathbf{x}}^{k+1}$ instead. In this way, we can ensure that the gradient descent algorithm is applied based on the exact stochastic gradient which is evaluated on $\hat{\mathbf{x}}_i^k$ at each worker node. This dispels the intricacy to deal with inexact gradient evaluated on $\mathbf{x}^k$ and thus it simplifies the convergence analysis.

Smooth case. In the smooth case, i.e., $R = 0$, Algorithm 1 can be simplified. The master node quantizes the recovered averaged gradient with error compensation and broadcasts it to all worker nodes. This simplified algorithm is shown in Algorithm 2.

Algorithm 2 DORE with $R(\mathbf{x}) = 0$

Input: Stepsize $\alpha, \beta, \gamma, \eta$, initialize $\mathbf{h}^0 = \mathbf{h}_i^0 = \mathbf{0}^d$, $\hat{\mathbf{x}}_i^0 = \hat{\mathbf{x}}^0,\ \forall i \in \{1, \ldots, n\}$.
for $k = 1, 2, \cdots, K-1$ **do**

For each worker $\{i = 1, 2, \cdots, n\}$:
Sample $\mathbf{g}_i^k$ such that $\mathbb{E}[\mathbf{g}_i^k | \hat{\mathbf{x}}_i^k] = \nabla f_i(\hat{\mathbf{x}}_i^k)$
Gradient residual: $\Delta_i^k = \mathbf{g}_i^k - \mathbf{h}_i^k$
Compression: $\hat{\Delta}_i^k = Q(\Delta_i^k)$
$\mathbf{h}_i^{k+1} = \mathbf{h}_i^k + \alpha \hat{\Delta}_i^k$
$\{\ \hat{\mathbf{g}}_i^k = \mathbf{h}_i^k + \hat{\Delta}_i^k\ \}$
Sent $\hat{\Delta}_i^k$ to the master
Receive $\hat{\mathbf{q}}^k$ from the master
$\hat{\mathbf{x}}_i^{k+1} = \hat{\mathbf{x}}_i^k + \beta \hat{\mathbf{q}}^k$

For the master:
Receive $\hat{\Delta}_i^k$s from workers
$\hat{\Delta}^k = 1/n \sum_i^n \hat{\Delta}_i^k$
$\hat{\mathbf{g}}^k = \mathbf{h}^k + \hat{\Delta}^k\ \ \{= 1/n \sum_i^n \hat{\mathbf{g}}_i^k\ \}$
$\mathbf{h}^{k+1} = \mathbf{h}^k + \alpha \hat{\Delta}^k$
$\mathbf{q}^k = -\gamma \hat{\mathbf{g}}^k + \eta \mathbf{e}^k$
Compression: $\hat{\mathbf{q}}^k = Q(\mathbf{q}^k)$
$\mathbf{e}^{k+1} = \mathbf{q}^k - \hat{\mathbf{q}}^k$
Broadcast $\hat{\mathbf{q}}^k$ to workers

end for
Output: any $\hat{\mathbf{x}}_i^K$

Compression rate. The compression of the gradient information can reduce at most 50% of the communication cost since it only considers compression during gradient aggregation while ignoring the model synchronization. However, DORE can further cut down the remaining 50% communication.

Taking the blockwise p-norm quantization as an example, every element of $\mathbf{x}$ can be represented by $\frac{3}{2}$ bits using the simple ternary coding $\{0, \pm 1\}$, along with one magnitude for each block. For example, if we consider the uniform block size b, the number of bits to represent a d-dimension vector of 32 bit float-point numbers can be reduced from $32d$ bits to $32\frac{d}{b} + \frac{3}{2}d$ bits. As long as the block size b is relatively large with respect to the constant 32, the cost $32\frac{d}{b}$ for storing the float-point number is relatively small such that the compression rate is close to $32d/(\frac{3}{2}d) \approx 21.3$ times (for example, 19.7 times when $b = 256$).

Applying this quantization, QSGD, Terngrad, MEM-SGD, and DIANA need to transmit $(32d + 32\frac{d}{b} + \frac{3}{2}d)$ bits per iteration and thus they are able to cut down 47% of the overall $2 \times 32d$ bits per iteration through gradient compression when $b = 256$. But with DORE, we only need to transmit $2(32\frac{d}{b} + \frac{3}{2}d)$ bits per iteration. Thus DORE can reduce over 95% of the total communication by compressing both the gradient and model transmission. More efficient coding techniques such as

Elias coding [26] can be applied to further reduce the number of bits per iteration.

2.4 Convergence Analysis

To show the convergence of DORE, we will make the following commonly used assumptions when needed.

Assumption 2. *Each worker node samples an unbiased estimator of the gradient stochastically with bounded variance, i.e., for* $i = 1, 2, \cdots, n$ *and* $\forall \mathbf{x} \in \mathbb{R}^d$,

$$\mathbb{E}[\mathbf{g}_i|\mathbf{x}] = \nabla f_i(\mathbf{x}), \quad \mathbb{E}\|\mathbf{g}_i - \nabla f_i(\mathbf{x})\|^2 \leq \sigma_i^2, \tag{2.2}$$

where $\mathbf{g}_i$ *is the estimator of* ∇f_i *at* $\mathbf{x}$*. In addition, we define* $\sigma^2 = \frac{1}{n}\sum_{i=1}^n \sigma_i^2$.

Assumption 3. *Each* f_i *is L-Lipschitz differentiable, i.e., for* $i = 1, 2, \cdots, n$ *and* $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$f_i(\mathbf{x}) \leq f_i(\mathbf{y}) + \langle \nabla f_i(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2. \tag{2.3}$$

Assumption 4. *Each* f_i *is* μ*-strongly convex (*$\mu \geq 0$*), i.e., for* $i = 1, 2, \cdots, n$ *and* $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$f_i(\mathbf{x}) \geq f_i(\mathbf{y}) + \langle \nabla f_i(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|^2. \tag{2.4}$$

For simplicity, we use the same compression operator for all worker nodes, and the master node can apply a different compression operator. We denote the constants in Assumption 1 as C_q and C_q^m for the worker and master nodes, respectively. Then we set α and β in both algorithms to satisfy

$$\begin{aligned} \frac{1-\sqrt{1-\frac{4C_q(C_q+1)}{nc}}}{2(C_q+1)} \leq \alpha \leq \frac{1+\sqrt{1-\frac{4C_q(C_q+1)}{nc}}}{2(C_q+1)}, \\ 0 < \beta \leq \frac{1}{C_q^m+1}, \end{aligned} \tag{2.5}$$

with $c \geq \frac{4C_q(C_q+1)}{n}$. We consider two scenarios in the following two subsections: f is strongly convex with a convex regularizer R and f is non-convex with $R = 0$.

2.4.1 Strongly Convex Case

Theorem 1. *Under Assumptions 1-4, if* α *and* β *in Algorithm 1 satisfy* (2.5)*,* η *and* γ *satisfy*

$$\eta < \min\left(\frac{-C_q^m+\sqrt{(C_q^m)^2+4(1-(C_q^m+1)\beta)}}{2C_q^m},\right.$$

$$\frac{4\mu L}{(\mu+L)^2(1+c\alpha)-4\mu L}\Big), \tag{2.6}$$

$$\frac{\eta(\mu+L)}{2(1+\eta)\mu L} \leq \gamma \leq \frac{2}{(1+c\alpha)(\mu+L)}, \tag{2.7}$$

then we have

$$\mathbf{V}^{k+1} \leq \rho^k \mathbf{V}^1 + \frac{(1+\eta)(1+nc\alpha)}{n(1-\rho)}\beta\gamma^2\sigma^2, \tag{2.8}$$

with

$$\begin{aligned}
\mathbf{V}^k =& \beta(1-(C_q^m+1)\beta)\mathbb{E}\|\mathbf{q}^{k-1}\|^2 + \mathbb{E}\|\hat{\mathbf{x}}^k - \mathbf{x}^*\|^2 \\
&+ \frac{(1+\eta)c\beta\gamma^2}{n}\sum_{i=1}^n \mathbb{E}\|\mathbf{h}_i^k - \nabla f_i(\mathbf{x}^*)\|^2, \\
\rho =& \max\left(\frac{(\eta^2+\eta)C_q^m}{1-(C_q^m+1)\beta}, 1+\eta\beta-\frac{2(1+\eta)\beta\gamma\mu L}{\mu+L}, 1-\alpha\right) < 1.
\end{aligned}$$

Corollary 1. *When there is no error compensation and we set $\eta = 0$, then $\rho = \max(1-\frac{2\beta\gamma\mu L}{\mu+L}, 1-\alpha)$. If we further set*

$$\alpha = \frac{1}{2(C_q+1)}, \quad \beta = \frac{1}{C_q^m+1}, \quad c = \frac{4C_q(C_q+1)}{n}, \tag{2.9}$$

and choose the largest step-size $\gamma = \frac{2}{(\mu+L)(1+2C_q/n)}$, the convergent factor is

$$(1-\rho)^{-1} = \max\left(2(C_q+1), (C_q^m+1)\frac{(\mu+L)^2}{2\mu L}\left(\frac{1}{2}+\frac{C_q}{n}\right)\right). \tag{2.10}$$

Remark 1. *In particular, suppose $\{\Delta_i\}_{i=1}^n$ are compressed using the Bernoulli p-norm quantization with the largest block size $d_{\max}$, then $C_q = \frac{1}{\alpha^w} - 1$, with $\alpha^w = \min_{\mathbf{0}\neq\mathbf{x}\in\mathbb{R}^{d_{\max}}} \frac{\|\mathbf{x}\|_2^2}{\|\mathbf{x}\|_1\|\mathbf{x}\|_p} \leq 1$. Similarly, $\mathbf{q}$ is compressed using the Bernoulli p-norm quantization with $C_q^m = \frac{1}{\alpha^m} - 1$. Then the linear convergent factor is*

$$(1-\rho)^{-1} = \max\left\{\frac{2}{\alpha^w}, \frac{1}{\alpha^m}\frac{(\mu+L)^2}{\mu L}\left(\frac{1}{2}-\frac{2}{n}+\frac{2}{n\alpha^w}\right)\right\}. \tag{2.11}$$

While the result of DIANA in [77] is $\max\left\{\frac{2}{\alpha^w}, \frac{\mu+L}{\mu}\left(\frac{1}{2}-\frac{1}{n}+\frac{1}{n\alpha^w}\right)\right\}$, *which is better than* (2.11) *with $\alpha^m = 1$ (no compression for the model). When there is no compression for Δ_i, i.e., $\alpha^w = 1$, the algorithm reduces to the gradient descent, and the linear convergent factor is the same as that of the gradient descent for strongly convex functions.*

Remark 2. *Although error compensation often improves the convergence empirically, in theory, no compensation, i.e., $\eta = 0$, provides the best convergence rate. This is because we don't have much information of the error being compensated. Filling this gap will be an interesting future direction.*

2.4.2 Nonconvex Case

Theorem 2. *Under Assumptions 1-3 and the additional assumption that each worker samples the gradient from the full dataset, we set α and β according to* (2.5)*. By choosing*

$$\gamma \leq \min\left\{\frac{-1+\sqrt{1+\frac{48L^2\beta^2(C_q^m+1)^2}{C_q^m}}}{12L\beta(C_q^m+1)}, \frac{1}{6L\beta(1+c\alpha)(C_q^m+1)}\right\},$$

we have

$$\begin{aligned}&\frac{\frac{\beta}{2} - 3(1+c\alpha)(C_q^m+1)L\beta^2\gamma}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2\\ \leq&\frac{\Lambda^1 - \Lambda^{K+1}}{\gamma K} + \frac{3(C_q^m+1)(1+nc\alpha)L\beta^2\sigma^2\gamma}{n},\end{aligned} \tag{2.12}$$

where

$$\begin{aligned}\Lambda^k =&(C_q^m+1)L\beta^2\|\mathbf{q}^{k-1}\|^2 + f(\hat{\mathbf{x}}^k) - f^*\\ &+ 3c(C_q^m+1)L\beta^2\gamma^2\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\|\mathbf{h}_i^k\|^2.\end{aligned} \tag{2.13}$$

Corollary 2. *Let $\alpha = \frac{1}{2(C_q+1)}, \beta = \frac{1}{C_q^m+1}$, and $c = \frac{4C_q(C_q+1)}{n}$, then $1+nc\alpha$ is a fixed constant. If $\gamma = \frac{1}{12L(1+c\alpha)(1+\sqrt{K/n})}$, when K is relatively large, we have*

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2 \lesssim \frac{1}{K} + \frac{1}{\sqrt{Kn}}. \tag{2.14}$$

Remark 3. *The dominant term in* (2.14) *is $O(1/\sqrt{Kn})$, which implies that the sample complexity of each worker node is $O(1/(n\epsilon^2))$ in average to achieve an ϵ-accurate solution. It shows that, same as DoubleSqueeze in [104], DORE is able to perform linear speedup. Furthermore, this convergence result is the same as the P-SGD without compression. Note that DoubleSqueeze has an extra term $(1/K)^{\frac{2}{3}}$, and its convergence requires the bounded variance of the compression operator.*

Algorithm	Compression	Compression Assumed	Linear rate	Nonconvex Rate
QSGD	Grad	2-norm Quantization	N/A	$\frac{1}{K} + B$
DIANA	Grad	p-norm Quantization	✓	$\frac{1}{\sqrt{Kn}} + \frac{1}{K}$
DoubleSqueeze	Grad+Model	Bounded Variance	N/A	$\frac{1}{\sqrt{Kn}} + \frac{1}{K^{2/3}} + \frac{1}{K}$
DORE	Grad+Model	Assumption 1	✓	$\frac{1}{\sqrt{Kn}} + \frac{1}{K}$

Table 2.1: A comparison between related algorithms. DORE is able to converges linearly to the $\mathcal{O}(\sigma)$ neighborhood of optimal point like full-precision SGD and DIANA in the strongly convex case while achieving much better communication efficiency. DORE also admits linear speedup in the nonconvex case like DoubleSqueeze but DORE doesn't require the assumptions of bounded compression error or bounded gradient.

2.5 Experiment

In this section, we validate the theoretical results and demonstrate the superior performance of DORE. Our experimental results demonstrate that (1) DORE achieves similar convergence speed as full-precision SGD and state-of-art quantized SGD baselines and (2) its iteration time is much smaller than most existing algorithms, supporting the superior communication efficiency of DORE.

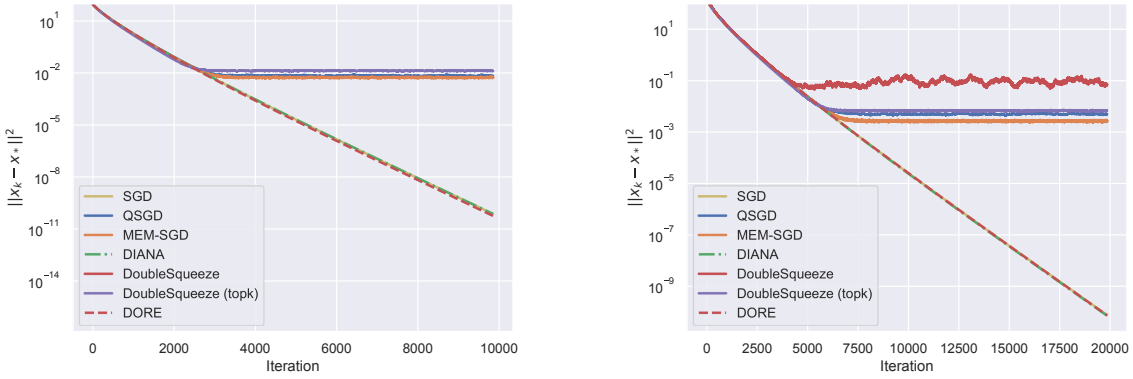
To make a fair comparison, we choose the same Bernoulli ∞-norm quantization as described in Section 2.3 and the quantization block size is 256 for all experiments if not being explicitly stated because ∞-norm quantization is unbiased and commonly used. The parameters α, β, η for DORE are chosen to be 0.1, 1 and 1, respectively.

The baselines we choose to compare include SGD, QSGD [4], MEM-SGD [97], DIANA [77], DoubleSqueeze and DoubleSqueeze (topk) [104]. SGD is the vanilla SGD without any compression and QSGD quantizes the gradient directly. MEM-SGD is the QSGD with error compensation. DIANA, which only compresses and transmits the gradient difference, is a special case of the proposed DORE. DoubleSqueeze quantizes both the gradient on the workers and the averaged gradient on the server with error compensation. Although DoubleSqueeze is claimed to work well with both biased and unbiased compression, in our experiment it converges much slower and suffers the loss of accuracy with unbiased compression. Thus, we also compare with DoubleSqueeze using the Top-k compression as presented in [104].

2.5.1 Strongly Convex

To verify the convergence for strongly convex and smooth objective functions, we conduct the experiment on a linear regression problem: $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2+\lambda\|\mathbf{x}\|^2$. The data matrix $\mathbf{A} \in \mathbb{R}^{1200\times500}$ and optimal solution $\mathbf{x}_* \in \mathbb{R}^{500}$ are randomly synthesized. Then we generate the prediction $\mathbf{b}$ by sampling from a Gaussian distribution whose mean is $\mathbf{A}\mathbf{x}_*$. The rows of the data matrix $\mathbf{A}$ are allocated evenly to 20 worker nodes. To better verify the linear convergence to the $\mathcal{O}(\sigma)$ neighborhood around the optimal solution, we take the full gradient in each node for all algorithms to exclude the effect of the gradient variance ($\sigma = 0$).

As showed in Figure 2.2, with full gradient and a constant learning rate, DORE converges linearly, same as SGD and DIANA, but QSGD, MEM-SGD, DoubleSqueeze, as well as DoubleSqueeze (topk) converge to a neighborhood of the optimal point. This is because these algorithms assume the bounded gradient and their convergence errors depend on that bound. Although they converge to the optimal solution using a diminishing step size, their converge rates will be much slower.

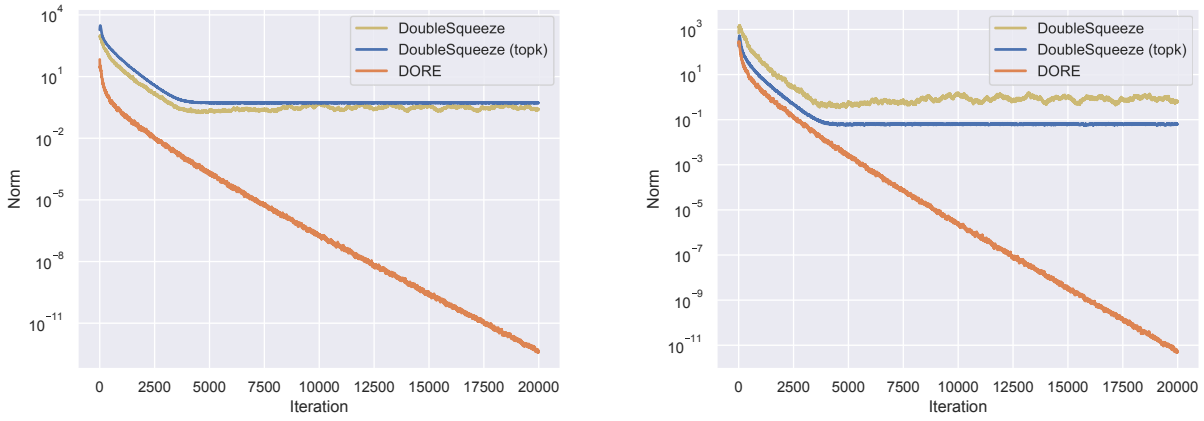


(a) Learning rate=0.05 (b) Learning rate=0.025

Figure 2.2: Linear regression on synthetic data. When the learning rate is 0.05, DoubleSqueeze diverges. In both cases, DORE, SGD, and DIANA converge linearly to the optimal point, while QSGD, MEM-SGD, DoubleSqueeze, and DoubleSqueeze (topk) only converge to the neighborhood even when full gradient is available.

Compression error The property of the compression operator indicates that the compression error is linearly proportional to the norm of the variable being compressed: $\mathbb{E}\|Q(\mathbf{x}) - \mathbf{x}\|^2 \leq C\|\mathbf{x}\|^2$.

We visualize the norm of the variables being compressed, i.e., the gradient residual (the worker side) and model residual (the master side) for DORE as well as error compensated gradient (the worker side) and averaged gradient (the master side) for DoubleSqueeze. As showed in Figure 2.3, the gradient and model residual of DORE decrease exponentially and the compression errors vanish. However, for DoubleSqueeze, their norms only decrease to some certain value and the compression error doesn't vanish. It explains why algorithms without residual compression cannot converge linearly to the $\mathcal{O}(\sigma)$ neighborhood of the optimal solution in the strongly convex case.



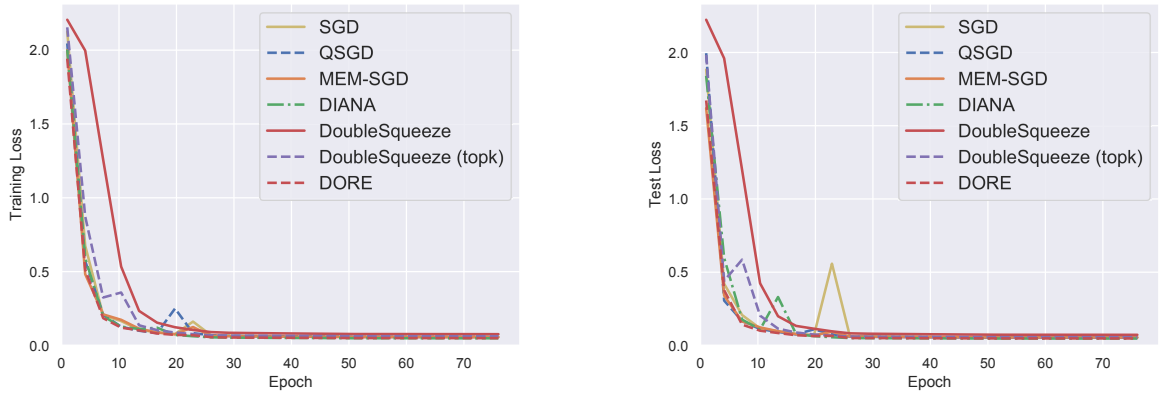
(a) Worker side

(b) Master side

Figure 2.3: The norm of variable being compressed in the linear regression experiment.

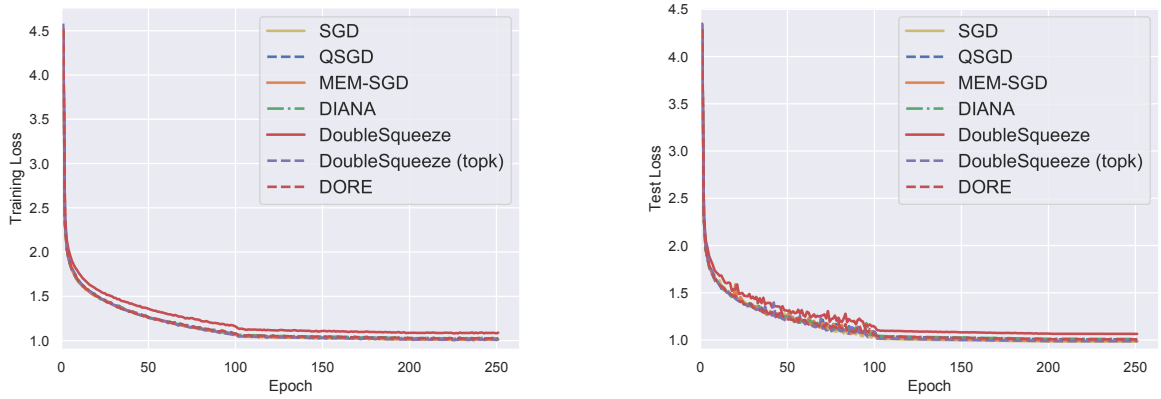
2.5.2 Nonconvex

To verify the convergence in the nonconvex case, we test the proposed DORE with two classical deep neural networks on two representative datasets, respectively, i.e., LeNet [52] on MNIST and Resnet18 [35] on CIFAR10. In the experiment, we use 1 parameter server and 10 workers, each of which is equipped with an NVIDIA Tesla K80 GPU. The batch size for each worker node is 256. We use 0.1 and 0.01 as the initial learning rates for LeNet and Resnet18, and decrease them by a factor of 0.1 after every 25 and 100 epochs, respectively. All parameter settings are the same for all algorithms.



(a) Training loss (b) Test Loss

Figure 2.4: LeNet trained on MNIST. DORE converges similarly as most baselines. It outperforms DoubleSqueeze using the same compression method while has similar performance as DoubleSqueeze (topk).



(a) Training Loss (b) Test Loss

Figure 2.5: Resnet18 trained on CIFAR10. DORE achieves similar convergence and accuracy as most baselines. DoubeSuqeeze converges slower and suffers from the higher loss but it works well with topk compression.

Figures 2.4 and 2.5 show the training loss and test loss for each epoch during the training of LeNet on the MNIST dataset and Resnet18 on CIFAR10 dataset. The results indicate that in the nonconvex case, even with both compressed gradient and model information, DORE can still achieve similar convergence speed as full-precision SGD and other quantized SGD variants. DORE achieves much better convergence speed than DoubleSqueeze using the same compression method

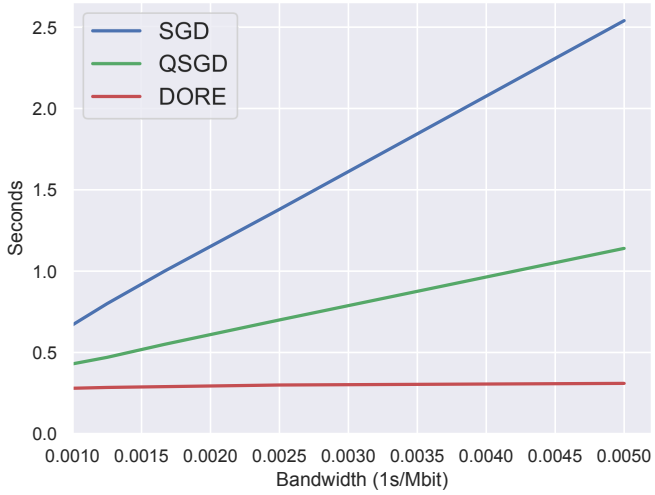


Figure 2.6: Per iteration time cost on Resnet18 for SGD, QSGD, and DORE. It is tested in a shared cluster environment connected by Gigabit Ethernet interface. DORE speeds up the training process significantly by mitigating the communication bottleneck.

and converges similarly with DoubleSqueeze with Topk compression as presented in [104]. We also validate via parameter sensitivity in Appendix A.1.2 that DORE performs consistently well under different parameter settings such as compression block size, α, β and η.

2.5.3 Communication Efficiency

In terms of communication cost, DORE enjoys the benefit of extremely efficient communication. As one example, under the same setting as the Resnet18 experiment described in the previous section, we test the time cost per iteration for SGD, QSGD, and DORE under varied network bandwidth. We didn't test MEM-SGD, DIANA, and DoubleSqueeze because MEM-SGD, DIANA have similar time cost as QSGD while DoubleSqueeze has similar time cost as DORE. The result showed in Figure 2.6 indicates that as the bandwidth becomes worse, with both gradient and model compression, the advantage of DORE becomes more remarkable compared to the baselines that don't apply compression for model synchronization. In Appendix A.1.1, we also demonstrate the communication efficiency in terms of communication bits and running time, which clearly suggests the benefit of the proposed algorithm.

2.6 Conclusion

Message passing is the dominating bottleneck for distributed training of modern large-scale machine learning models. Extensive works have compressed the gradient information to be transferred during the training process, but model compression is rather limited due to its intrinsic difficulty. In this work, we proposed the Double Residual Compression SGD named DORE to compress both gradient and model communication that can mitigate this bottleneck prominently. The theoretical analyses suggest good convergence rate of DORE under weak assumptions. Furthermore, DORE is able to reduce 95% of the communication cost in message passing while maintaining similar convergence rate and model accuracy compared with the full-precision SGD.

CHAPTER 3

LINEAR CONVERGENT DECENTRALIZED OPTIMIZATION WITH COMPRESSION

Communication compression has become a key strategy to speed up the message passing in distributed optimization. However, existing decentralized algorithms with compression mainly focus on compressing DGD-type algorithms. They are unsatisfactory in terms of convergence rate, stability, and the capability to handle heterogeneous data. Motivated by primal-dual algorithms, in this chapter, we propose the first LinEAr convergent Decentralized algorithm with compression, LEAD. Our theory describes the coupled dynamics of the inexact primal and dual update as well as compression error, and we provide the first consensus error bound in such settings without assuming bounded gradients. This is also the first work that proves in certain compression regime, the message compression in message passing do not hurt the convergence, which means it achieves better communication efficiency for free. Experiments on convex problems validate our theoretical analysis, and empirical study on deep neural nets shows that LEAD is applicable to non-convex problems as well.

3.1 Introduction

Distributed optimization solves the following optimization problem

$$\mathbf{x}^* := \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left[f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right] \tag{3.1}$$

with n computing agents and a communication network. Each $f_i(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ is a local objective function of agent i and typically defined on the data $\mathcal{D}_i$ settled at that agent. The data distributions $\{\mathcal{D}_i\}$ can be heterogeneous depending on the applications such as in federated learning. The variable $\mathbf{x} \in \mathbb{R}^d$ often represents model parameters in machine learning. A distributed optimization algorithm seeks an optimal solution that minimizes the overall objective function $f(\mathbf{x})$ collectively. According to the communication topology, existing algorithms can be conceptually categorized into centralized and decentralized ones. Specifically, centralized algorithms require global communication between

agents (through central agents or parameter servers). While decentralized algorithms only require local communication between connected agents and are more widely applicable than centralized ones. In both paradigms, the computation can be relatively fast with powerful computing devices; efficient communication is the key to improve algorithm efficiency and system scalability, especially when the network bandwidth is limited.

In recent years, various communication compression techniques, such as quantization and sparsification, have been developed to reduce communication costs. Notably, extensive studies [92, 4, 7, 97, 45, 77, 104, 68] have utilized gradient compression to significantly boost communication efficiency for centralized optimization. They enable efficient large-scale optimization while maintaining comparable convergence rates and practical performance with their non-compressed counterparts. This great success has suggested the potential and significance of communication compression in decentralized algorithms.

While extensive attention has been paid to centralized optimization, communication compression is relatively less studied in decentralized algorithms because the algorithm design and analysis are more challenging in order to cover general communication topologies. There are recent efforts trying to push this research direction. For instance, DCD-SGD and ECD-SGD [101] introduce difference compression and extrapolation compression to reduce model compression error. [88, 89] introduce QDGD and QuanTimed-DSGD to achieve exact convergence with small stepsize. DeepSqueeze [102] directly compresses the local model and compensates the compression error in the next iteration. CHOCO-SGD [51, 50] presents a novel quantized gossip algorithm that reduces compression error by difference compression and preserves the model average. Nevertheless, most existing works focus on the compression of primal-only algorithms, i.e., reduce to DGD [80, 123] or P-DSGD [62]. They are unsatisfying in terms of convergence rate, stability, and the capability to handle heterogeneous data. Part of the reason is that they inherit the drawback of DGD-type algorithms, whose convergence rate is slow in heterogeneous data scenarios where the data distributions are significantly different from agent to agent.

In the literature of decentralized optimization, it has been proved that primal-dual algorithms

can achieve faster converge rates and better support heterogeneous data [63, 96, 59, 124]. However, it is unknown whether communication compression is feasible for primal-dual algorithms and how fast the convergence can be with compression. In this work, we attempt to bridge this gap by investigating the communication compression for primal-dual decentralized algorithms. Our major contributions can be summarized as:

- We delineate two key challenges in the algorithm design for communication compression in decentralized optimization, i.e., data heterogeneity and compression error, and motivated by primal-dual algorithms, we propose a novel decentralized algorithm with compression, LEAD.
- We prove that for LEAD, a constant stepsize in the range $(0, 2/(\mu+L)]$ is sufficient to ensure linear convergence for strongly convex and smooth objective functions. To the best of our knowledge, LEAD is the first linear convergent decentralized algorithm with compression. Moreover, LEAD provably works with unbiased compression of arbitrary precision.
- We further prove that if the stochastic gradient is used, LEAD converges linearly to the $O(\sigma^2)$ neighborhood of the optimum with constant stepsize. LEAD is also able to achieve exact convergence to the optimum with diminishing stepsize.
- Extensive experiments on convex problems validate our theoretical analyses, and the empirical study on training deep neural nets shows that LEAD is applicable for nonconvex problems. LEAD achieves state-of-art computation and communication efficiency in all experiments and significantly outperforms the baselines on heterogeneous data. Moreover, LEAD is robust to parameter settings and needs minor effort for parameter tuning.

3.2 Related Work

Decentralized optimization can be traced back to the work by [107]. DGD [80] is the most classical decentralized algorithm. It is intuitive and simple but converges slowly due to the diminishing stepsize that is needed to obtain the optimal solution [123]. Its stochastic version D-PSGD [62] has been shown effective for training nonconvex deep learning models. Algorithms based on primal-dual

formulations or gradient tracking are proposed to eliminate the convergence bias in DGD-type algorithms and improve the convergence rate, such as D-ADMM [78], DLM [63], EXTRA [96], NIDS [59], D^2 [103], Exact Diffusion [125], OPTRA [120], DIGing [79], GSGT [87], etc.

Recently, communication compression is applied to decentralized settings by [101]. It proposes two algorithms, i.e., DCD-SGD and ECD-SGD, which require compression of high accuracy and are not stable with aggressive compression. [88, 89] introduce QDGD and QuanTimed-DSGD to achieve exact convergence with small stepsize and the convergence is slow. DeepSqueeze [102] compensates the compression error to the compression in the next iteration. Motivated by the quantized average consensus algorithms, such as [13], the quantized gossip algorithm CHOCO-Gossip [51] converges linearly to the consensual solution. Combining CHOCO-Gossip and D-PSGD leads to a decentralized algorithm with compression, CHOCO-SGD, which converges sublinearly under the strong convexity and gradient boundedness assumptions. Its nonconvex variant is further analyzed in [50]. A new compression scheme using the modulo operation is introduced in [71] for decentralized optimization. A general algorithmic framework aiming to maintain the linear convergence of distributed optimization under compressed communication is considered in [75]. It requires a contractive property that is not satisfied by many decentralized algorithms including the algorithm in this work.

3.3 Algorithm

We first introduce notations and definitions used in this work. We use bold upper-case letters such as $\mathbf{X}$ to define matrices and bold lower-case letters such as $\mathbf{x}$ to define vectors. Let $\mathbf{1}$ and $\mathbf{0}$ be vectors with all ones and zeros, respectively. Their dimensions will be provided when necessary. Given two matrices $\mathbf{X},\ \mathbf{Y} \in \mathbb{R}^{n\times d}$, we define their inner product as $\langle \mathbf{X}, \mathbf{Y}\rangle = \mathrm{tr}(\mathbf{X}^\top \mathbf{Y})$ and the norm as $\|\mathbf{X}\| = \sqrt{\langle \mathbf{X}, \mathbf{X}\rangle}$. We further define $\langle \mathbf{X}, \mathbf{Y}\rangle_{\mathbf{P}} = \mathrm{tr}(\mathbf{X}^\top \mathbf{P}\mathbf{Y})$ and $\|\mathbf{X}\|_{\mathbf{P}} = \sqrt{\langle \mathbf{X}, \mathbf{X}\rangle_{\mathbf{P}}}$ for any given symmetric positive semidefinite matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$. For simplicity, we will majorly use the matrix notation in this work. For instance, each agent i holds an individual estimate $\mathbf{x}_i \in \mathbb{R}^d$ of the global variable $\mathbf{x} \in \mathbb{R}^d$. Let $\mathbf{X}^k$ and $\nabla \mathbf{F}(\mathbf{X}^k)$ be the collections of $\{\mathbf{x}_i^k\}_{i=1}^n$ and $\{\nabla f_i(\mathbf{x}_i^k)\}_{i=1}^n$ which are

defined below:

$$\mathbf{X}^k = \left[\mathbf{x}_1^k, \ldots, \mathbf{x}_n^k\right]^\top \in \mathbb{R}^{n\times d}, \quad \nabla\mathbf{F}(\mathbf{X}^k) = \left[\nabla f_1(\mathbf{x}_1^k), \ldots, \nabla f_n(\mathbf{x}_n^k)\right]^\top \in \mathbb{R}^{n\times d}. \tag{3.2}$$

We use $\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)$ to denote the stochastic approximation of $\nabla\mathbf{F}(\mathbf{X}^k)$. With these notations, the update $\mathbf{X}^{k+1} = \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)$ means that $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k - \eta\nabla f_i(\mathbf{x}_i^k;\xi_i^k)$ for all i. In this work, we need the average of all rows in $\mathbf{X}^k$ and $\nabla\mathbf{F}(\mathbf{X}^k)$, so we define $\overline{\mathbf{X}}^k = (\mathbf{1}^\top\mathbf{X}^k)/n$ and $\overline{\nabla}\mathbf{F}(\mathbf{X}^k) = (\mathbf{1}^\top\nabla\mathbf{F}(\mathbf{X}^k))/n$. They are row vectors, and we will take a transpose if we need a column vector. The pseudoinverse of a matrix $\mathbf{M}$ is denoted as $\mathbf{M}^\dagger$. The largest, ith-largest, and smallest nonzero eigenvalues of a symmetric matrix $\mathbf{M}$ are $\lambda_{\max}(\mathbf{M})$, $\lambda_i(\mathbf{M})$, and $\lambda_{\min}(\mathbf{M})$.

Assumption 5 (Mixing matrix)**.** *The connected network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consists of a node set $\mathcal{V} = \{1, 2, \ldots, n\}$ and an undirected edge set $\mathcal{E}$. The primitive symmetric doubly-stochastic matrix $\mathbf{W} = [w_{ij}] \in \mathbb{R}^{n\times n}$ encodes the network structure such that $w_{ij} = 0$ if nodes i and j are not connected and cannot exchange information.*

Assumption 5 implies that $-1 < \lambda_n(\mathbf{W}) \le \lambda_{n-1}(\mathbf{W}) \le \cdots \lambda_2(\mathbf{W}) < \lambda_1(\mathbf{W}) = 1$ and $\mathbf{W}\mathbf{1} = \mathbf{1}$ [118, 96]. The matrix multiplication $\mathbf{X}^{k+1} = \mathbf{W}\mathbf{X}^k$ describes that agent i takes a weighted sum from its neighbors and itself, i.e., $\mathbf{x}_i^{k+1} = \sum_{j\in\mathcal{N}_i\cup\{i\}} w_{ij}\mathbf{x}_j^k$, where $\mathcal{N}_i$ denotes the neighbors of agent i.

The proposed algorithm LEAD to solve problem (3.1) is showed in Alg. 3 with matrix notations for conciseness. We will refer to the line number in the analysis. A complete algorithm description from the agent's perspective can be found in Algorithm 4. The motivation behind Alg. 3 is to achieve two goals: (a) consensus ($\mathbf{x}_i^k - (\overline{\mathbf{X}}^k)^\top \to \mathbf{0}$) and (b) convergence ($(\overline{\mathbf{X}}^k)^\top \to \mathbf{x}^*$). We first discuss how goal (a) leads to goal (b) and then explain how LEAD fulfills goal (a).

In essence, LEAD runs the approximate SGD globally and reduces to the exact SGD under consensus. One key property for LEAD is $\mathbf{1}_{n\times 1}^\top\mathbf{D}^k = \mathbf{0}$, regardless of the compression error in $\hat{\mathbf{Y}}^k$. It holds because that for the initialization, we require $\mathbf{D}^1 = (\mathbf{I} - \mathbf{W})\mathbf{Z}$ for some $\mathbf{Z} \in \mathbb{R}^{n\times d}$, e.g., $\mathbf{D}^1 = \mathbf{0}^{n\times d}$, and that the update of $\mathbf{D}^k$ ensures $\mathbf{D}^k \in \mathbf{Range}(\mathbf{I} - \mathbf{W})$ for all k and $\mathbf{1}_{n\times 1}^\top(\mathbf{I} - \mathbf{W}) = \mathbf{0}$ as we will explain later. Therefore, multiplying $(1/n)\mathbf{1}_{n\times 1}^\top$ on both sides of Line 7 leads to a global

Algorithm 3 LEAD

Input: Stepsize η, parameter $(\alpha,\ \gamma)$, $\mathbf{X}^0,\ \mathbf{H}^1,\ \mathbf{D}^1 = (\mathbf{I}-\mathbf{W})\mathbf{Z}$ for any $\mathbf{Z}$
Output: $\mathbf{X}^K$ or $1/n\sum_{i=1}^n \mathbf{X}_i^K$

$\mathbf{H}_w^1 = \mathbf{W}\mathbf{H}^1$
$\mathbf{X}^1 = \mathbf{X}^0 - \eta\nabla\mathbf{F}(\mathbf{X}^0;\xi^0)$
for $k = 1, 2, \cdots, K-1$ **do**
  $\mathbf{Y}^k = \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \eta\mathbf{D}^k$
  $\hat{\mathbf{Y}}^k, \hat{\mathbf{Y}}_w^k, \mathbf{H}^{k+1}, \mathbf{H}_w^{k+1} = \mathrm{COMM}(\mathbf{Y}^k, \mathbf{H}^k, \mathbf{H}_w^k)$
  $\mathbf{D}^{k+1} = \mathbf{D}^k + \frac{\gamma}{2\eta}(\hat{\mathbf{Y}}^k - \hat{\mathbf{Y}}_w^k)$
  $\mathbf{X}^{k+1} = \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \eta\mathbf{D}^{k+1}$
end for

procedure $\mathrm{COMM}(\mathbf{Y}, \mathbf{H}, \mathbf{H}_w)$:
  $\mathbf{Q} = \mathrm{Compress}(\mathbf{Y} - \mathbf{H})$
  $\hat{\mathbf{Y}} = \mathbf{H} + \mathbf{Q}$
  $\hat{\mathbf{Y}}_w = \mathbf{H}_w + \mathbf{W}\mathbf{Q}$
  $\mathbf{H} = (1-\alpha)\mathbf{H} + \alpha\hat{\mathbf{Y}}$
  $\mathbf{H}_w = (1-\alpha)\mathbf{H}_w + \alpha\hat{\mathbf{Y}}_w$
  return: $\hat{\mathbf{Y}}, \hat{\mathbf{Y}}_w, \mathbf{H}, \mathbf{H}_w$
end procedure

average view of Alg. 3:

$$\overline{\mathbf{X}}^{k+1} = \overline{\mathbf{X}}^k - \eta\overline{\nabla}\mathbf{F}(\mathbf{X}^k;\xi^k), \tag{3.3}$$

which doesn't contain the compression error. Note that this is an approximate SGD step because, as shown in (3.2), the gradient $\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)$ is not evaluated on a global synchronized model $\overline{\mathbf{X}}^k$. However, if the solution converges to the consensus solution, i.e., $\mathbf{x}_i^k - (\overline{\mathbf{X}}^k)^\top \to \mathbf{0}$, then $\mathbb{E}_{\xi^k}[\overline{\nabla}\mathbf{F}(\mathbf{X}^k;\xi^k) - \nabla f(\overline{\mathbf{X}}^k;\xi^k)] \to \mathbf{0}$ and (3.3) gradually reduces to exact SGD.

With the establishment of how consensus leads to convergence, the obstacle becomes how to achieve consensus under local communication and compression challenges. It requires addressing two issues, i.e., data heterogeneity and compression error. To deal with these issues, existing algorithms, such as DCD-SGD, ECD-SGD, QDGD, DeepSqueeze, Moniqua, and CHOCO-SGD, need a diminishing or constant but small stepsize depending on the total number of iterations. However, these choices unavoidably cause slower convergence and bring in the difficulty of parameter tuning. In contrast, LEAD takes a different way to solve these issues, as explained below.

Data heterogeneity. It is common in distributed settings that there exists data heterogeneity among agents, especially in real-world applications where different agents collect data from different scenarios. In other words, we generally have $f_i(\mathbf{x}) \neq f_j(\mathbf{x})$ for $i \neq j$. The optimality condition of problem (3.1) gives $\mathbf{1}_{n\times 1}^\top \nabla\mathbf{F}(\mathbf{X}^*) = \mathbf{0}$, where $\mathbf{X}^* = [\mathbf{x}^*, \cdots, \mathbf{x}^*]$ is a consensual and optimal solution. The data heterogeneity and optimality condition imply that there exist at least two agents i and j

such that $\nabla f_i(\mathbf{x}^*) \neq \mathbf{0}$ and $\nabla f_j(\mathbf{x}^*) \neq \mathbf{0}$. As a result, a simple D-PSGD algorithm cannot converge to the consensual and optimal solution as $\mathbf{X}^* \neq \mathbf{W}\mathbf{X}^* - \eta\mathbb{E}_\xi \nabla \mathbf{F}(\mathbf{X}^*; \xi)$ even when the stochastic gradient variance is zero.

Gradient correction. Primal-dual algorithms or gradient tracking algorithms are able to convergence much faster than DGD-type algorithms by handling the data heterogeneity issue, as introduced in Section 3.2. Specifically, LEAD is motivated by the design of primal-dual algorithm NIDS [59] and the relation becomes clear if we consider the two-step reformulation of NIDS adopted in [57]:

$$\mathbf{D}^{k+1} = \mathbf{D}^k + \frac{\mathbf{I} - \mathbf{W}}{2\eta}(\mathbf{X}^k - \eta\nabla \mathbf{F}(\mathbf{X}^k) - \eta\mathbf{D}^k), \tag{3.4}$$

$$\mathbf{X}^{k+1} = \mathbf{X}^k - \eta\nabla \mathbf{F}(\mathbf{X}^k) - \eta\mathbf{D}^{k+1}, \tag{3.5}$$

where $\mathbf{X}^k$ and $\mathbf{D}^k$ represent the primal and dual variables respectively. The dual variable $\mathbf{D}^k$ plays the role of gradient correction. As $k \to \infty$, we expect $\mathbf{D}^k \to -\nabla \mathbf{F}(\mathbf{X}^*)$ and $\mathbf{X}^k$ will converge to $\mathbf{X}^*$ via the update in (3.5) since $\mathbf{D}^{k+1}$ corrects the nonzero gradient $\nabla \mathbf{F}(\mathbf{X}^k)$ asymptotically. The key design of Alg. 3 is to provide compression for the auxiliary variable defined as $\mathbf{Y}^k = \mathbf{X}^k - \eta\nabla \mathbf{F}(\mathbf{X}^k) - \eta\mathbf{D}^k$. Such design ensures that the dual variable $\mathbf{D}^k$ lies in $\mathbf{Range}(\mathbf{I}-\mathbf{W})$, which is essential for convergence. Moreover, it achieves the implicit error compression as we will explain later. To stabilize the algorithm with inexact dual update, we introduce a parameter γ to control the stepsize in the dual update. Therefore, if we ignore the details of the compression, Alg. 3 can be concisely written as

$$\mathbf{Y}^k = \mathbf{X}^k - \eta\nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \eta\mathbf{D}^k \tag{3.6}$$

$$\mathbf{D}^{k+1} = \mathbf{D}^k + \frac{\gamma}{2\eta}(\mathbf{I} - \mathbf{W})\hat{\mathbf{Y}}^k \tag{3.7}$$

$$\mathbf{X}^{k+1} = \mathbf{X}^k - \eta\nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \eta\mathbf{D}^{k+1} \tag{3.8}$$

where $\hat{\mathbf{Y}}^k$ represents the compression of $\mathbf{Y}^k$ and $\mathbf{F}(\mathbf{X}^k; \xi^k)$ denote the stochastic gradients.

Nevertheless, how to compress the communication and how fast the convergence we can attain with compression error are unknown. In the following, we propose to carefully control the compression error by difference compression and error compensation such that the inexact

dual update (Line 6) and primal update (Line 7) can still guarantee the convergence as proved in Section 3.4.

Compression error. Different from existing works, which typically compress the primal variable $\mathbf{X}^k$ or its difference, LEAD first construct an intermediate variable $\mathbf{Y}^k$ and apply compression to obtain its coarse representation $\hat{\mathbf{Y}}^k$ as shown in the procedure $Comm\mathbf{Y}, \mathbf{H}, \mathbf{H}_w$:

- Compress the difference between $\mathbf{Y}$ and the state variable $\mathbf{H}$ as $\mathbf{Q}$;
- $\mathbf{Q}$ is encoded into the low-bit representation, which enables the efficient local communication step $\hat{\mathbf{Y}}_w = \mathbf{H}_w + \mathbf{W}\mathbf{Q}$. It is *the only communication step* in each iteration.
- Each agent recovers its estimate $\hat{\mathbf{Y}}$ by $\hat{\mathbf{Y}} = \mathbf{H} + \mathbf{Q}$ and we have $\hat{\mathbf{Y}}_w = \mathbf{W}\hat{\mathbf{Y}}$.
- States $\mathbf{H}$ and $\mathbf{H}_w$ are updated based on $\hat{\mathbf{Y}}$ and $\hat{\mathbf{Y}}_w$, respectively. We have $\mathbf{H}_w = \mathbf{W}\mathbf{H}$.

By this procedure, we expect when both $\mathbf{Y}^k$ and $\mathbf{H}^k$ converge to $\mathbf{X}^*$, the compression error vanishes asymptotically due to the assumption we make for the compression operator in Assumption 6.

Remark 4. *Note that difference compression is also applied in DCD-PSGD [101] and CHOCO-SGD [51], but their state update is the simple integration of the compressed difference. We find this update is usually too aggressive and cause instability as showed in our experiments. Therefore, we adopt a momentum update* $\mathbf{H} = (1-\alpha)\mathbf{H} + \alpha\hat{\mathbf{Y}}$ *motivated from DIANA [77], which reduces the compression error for gradient compression in centralized optimization.*

Implicit error compensation. On the other hand, even if the compression error exists, LEAD essentially compensates for the error in the inexact dual update (Line 6), making the algorithm more stable and robust. To illustrate how it works, let $\mathbf{E}^k = \hat{\mathbf{Y}}^k - \mathbf{Y}^k$ denote the compression error and $\mathbf{e}_i^k$ be its i-th row. The update of $\mathbf{D}^k$ gives

$$\mathbf{D}^{k+1} = \mathbf{D}^k + \frac{\gamma}{2\eta}(\hat{\mathbf{Y}}^k - \hat{\mathbf{Y}}_w^k) = \mathbf{D}^k + \frac{\gamma}{2\eta}(\mathbf{I} - \mathbf{W})\mathbf{Y}^k + \frac{\gamma}{2\eta}(\mathbf{E}^k - \mathbf{W}\mathbf{E}^k)$$

where $-\mathbf{W}\mathbf{E}^k$ indicates that agent i spreads total compression error $-\sum_{j\in\mathcal{N}_i\cup\{i\}} w_{ji}\mathbf{e}_i^k = -\mathbf{e}_i^k$ to all agents and $\mathbf{E}^k$ indicates that each agent compensates this error locally by adding $\mathbf{e}_i^k$ back. This error compensation also explains why the global view in (3.3) doesn't involve compression error.

Remark 5. *Note that in LEAD, the compression error is compensated into the model* $\mathbf{X}^{k+1}$ *through Line 6 and Line 7 such that the gradient computation in the next iteration is aware of the compression error. This has some subtle but important difference from the error compensation or error feedback in [92, 116, 97, 45, 104, 68, 102], where the error is stored in the memory and only compensated after gradient computation and before the compression.*

LEAD in agent's perspective In Algorithm 3, we described the algorithm with matrix notations for concision. Here we further provide a complete algorithm description from the agents' perspective.

Algorithm 4 LEAD in Agent's Perspective

input: stepsize η, compression parameters $(\alpha,\ \gamma)$, initial values $\mathbf{x}_i^0, \mathbf{h}_i^1, \mathbf{z}_i,\ \forall i \in \{1, 2, \ldots, n\}$

output: $\mathbf{x}_i^K,\ \forall i \in \{1, 2, \ldots, n\}$ or $\frac{\sum_{i=1}^n \mathbf{x}_i^K}{n}$

1: **for** each agent $i \in \{1, 2, \ldots, n\}$ **do**
2: $\quad \mathbf{d}_i^1 = \mathbf{z}_i - \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} \mathbf{z}_j$
3: $\quad (\mathbf{h}_w)_i^1 = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} (\mathbf{h}_w)_j^1$
4: $\quad \mathbf{x}_i^1 = \mathbf{x}_i^0 - \eta \nabla f_i(\mathbf{x}_i^0; \xi_i^0)$
5: **end for**
6: **for** $k = 1, 2, \ldots, K-1$ (in parallel for all agents $i \in \{1, 2, \ldots, n\}$) **do**
7: $\quad$ compute $\nabla f_i(\mathbf{x}_i^k; \xi_i^k)$ ▷ Gradient computation
8: $\quad \mathbf{y}_i^k = \mathbf{x}_i^k - \eta \nabla f_i(\mathbf{x}_i^k; \xi_i^k) - \eta \mathbf{d}_i^k$
9: $\quad \mathbf{q}_i^k = \text{Compress}(\mathbf{y}_i^k - \mathbf{h}_i^k)$ ▷ Compression
10: $\quad \hat{\mathbf{y}}_i^k = \mathbf{h}_i^k + \mathbf{q}_i^k$
11: $\quad$ **for** neighbors $j \in \mathcal{N}_i$ **do**
12: $\quad\quad$ Send $\mathbf{q}_i^k$ and receive $\mathbf{q}_j^k$ ▷ Communication
13: $\quad$ **end for**
14: $\quad (\hat{\mathbf{y}}_w)_i^k = (\mathbf{h}_w)_i^k + \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} \mathbf{q}_j^k$
15: $\quad \mathbf{h}_i^{k+1} = (1-\alpha)\mathbf{h}_i^k + \alpha \hat{\mathbf{y}}_i^k$
16: $\quad (\mathbf{h}_w)_i^{k+1} = (1-\alpha)(\mathbf{h}_w)_i^k + \alpha (\hat{\mathbf{y}}_w)_i^k$
17: $\quad \mathbf{d}_i^{k+1} = \mathbf{d}_i^k + \frac{\gamma}{2\eta}\left(\hat{\mathbf{y}}_i^k - (\hat{\mathbf{y}}_w)_i^k\right)$
18: $\quad \mathbf{x}_i^{k+1} = \mathbf{x}_i^k - \eta \nabla f_i(\mathbf{x}_i^k; \xi_i^k) - \eta \mathbf{d}_i^{k+1}$ ▷ Model update
19: **end for**

Connections with exiting algorithms The non-compressed variant of LEAD in Alg. 3 recovers NIDS [59], D^2 [103] and Exact Diffusion [125] as shown in Proposition 1. In Corollary 3, we show that the convergence rate of LEAD exactly recovers the rate of NIDS when $C = 0, \gamma = 1$ and $\sigma = 0$.

Proposition 1 (Connection to NIDS, D^2 and Exact Diffusion)**.** *When there is no communication compression (i.e.,* $\hat{\mathbf{Y}}^k = \mathbf{Y}^k$*) and* $\gamma = 1$*, Alg. 3 recovers* D^2*:*

$$\mathbf{X}^{k+1} = \frac{\mathbf{I}+\mathbf{W}}{2}\Big(2\mathbf{X}^k - \mathbf{X}^{k-1} - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) + \eta\nabla\mathbf{F}(\mathbf{X}^{k-1};\xi^{k-1})\Big). \tag{3.9}$$

Furthermore, if the stochastic estimator of the gradient $\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)$ *is replaced by the full gradient, it recovers NIDS and Exact Diffusion with specific settings.*

Corollary 3 (Consistency with NIDS)**.** *When* $C = 0$ *(no communication compression),* $\gamma = 1$ *and* $\sigma = 0$ *(full gradient), LEAD has the convergence consistent with NIDS with* $\eta \in (0, 2/(\mu + L)]$*:*

$$\mathcal{L}^{k+1} \le \max\left\{1 - \mu(2\eta - \mu\eta^2), 1 - \frac{1}{2\lambda_{\max}((\mathbf{I}-\mathbf{W})^\dagger)}\right\}\mathcal{L}^k. \tag{3.10}$$

See the proof in B.3.5.

Proof of Proposition 1. Let $\gamma = 1$ and $\hat{\mathbf{Y}}^k = \mathbf{Y}^k$. Combing Lines 4 and 6 of Alg. 3 gives

$$\mathbf{D}^{k+1} = \mathbf{D}^k + \frac{\mathbf{I}-\mathbf{W}}{2\eta}(\mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \eta\mathbf{D}^k). \tag{3.11}$$

Based on Line 7, we can represent $\eta\mathbf{D}^k$ from the previous iteration as

$$\eta\mathbf{D}^k = \mathbf{X}^{k-1} - \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^{k-1};\xi^{k-1}). \tag{3.12}$$

Eliminating both $\mathbf{D}^k$ and $\mathbf{D}^{k+1}$ by substituting (3.11)-(3.12) into Line 7, we obtain

$$\begin{aligned}
\mathbf{X}^{k+1} &= \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \Big(\eta\mathbf{D}^k + \frac{\mathbf{I}-\mathbf{W}}{2}(\mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \eta\mathbf{D}^k)\Big) \quad \text{(from (3.11))}\\
&= \frac{\mathbf{I}+\mathbf{W}}{2}(\mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)) - \frac{\mathbf{I}+\mathbf{W}}{2}\eta\mathbf{D}^k\\
&= \frac{\mathbf{I}+\mathbf{W}}{2}(\mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)) - \frac{\mathbf{I}+\mathbf{W}}{2}(\mathbf{X}^{k-1} - \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^{k-1};\xi^{k-1})) \quad \text{(from (3.12))}\\
&= \frac{\mathbf{I}+\mathbf{W}}{2}(2\mathbf{X}^k - \mathbf{X}^{k-1} - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) + \eta\nabla\mathbf{F}(\mathbf{X}^{k-1};\xi^{k-1})),
\end{aligned} \tag{3.13}$$

which is exactly D^2. It also recovers Exact Diffusion with $\mathbf{A} = \frac{\mathbf{I}+\mathbf{W}}{2}$ and $\mathbf{M} = \eta\mathbf{I}$ in Eq. (97) of [125]. □

3.4 Theoretical Analysis

In this section, we show the convergence rate for the proposed algorithm LEAD. Before showing the main theorem, we make some assumptions, which are commonly used for the analysis of decentralized optimization algorithms. All proofs are provided in Appendix B.3.

Assumption 6 (Unbiased and C-contracted operator)**.** *The compression operator $Q : \mathbb{R}^d \to \mathbb{R}^d$ is unbiased, i.e., $\mathbb{E}Q(\mathbf{x}) = \mathbf{x}$, and there exists $C \geq 0$ such that $\mathbb{E}\|\mathbf{x} - Q(\mathbf{x})\|_2^2 \leq C\|\mathbf{x}\|_2^2$ for all $\mathbf{x} \in \mathbb{R}^d$.*

Assumption 7 (Stochastic gradient)**.** *The stochastic gradient $\nabla f_i(\mathbf{x}; \xi)$ is unbiased, i.e., $\mathbb{E}_\xi \nabla f_i(\mathbf{x}; \xi) = \nabla f_i(\mathbf{x})$, and the stochastic gradient variance is bounded: $\mathbb{E}_\xi \|\nabla f_i(\mathbf{x}; \xi) - \nabla f_i(\mathbf{x})\|_2^2 \leq \sigma_i^2$ for all $i \in [n]$. Denote $\sigma^2 = \frac{1}{n}\sum_{i=1}^n \sigma_i^2$.*

Assumption 8. *Each f_i is L-smooth and μ-strongly convex with $L \geq \mu > 0$, i.e., for $i = 1, 2, \ldots, n$ and $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we have*

$$f_i(\mathbf{y}) + \langle \nabla f_i(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|^2 \leq f_i(\mathbf{x}) \leq f_i(\mathbf{y}) + \langle \nabla f_i(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2.$$

Theorem 3 (Constant stepsize)**.** *Let $\{\mathbf{X}^k, \mathbf{H}^k, \mathbf{D}^k\}$ be the sequence generated from Alg. 3 and $\mathbf{X}^*$ is the optimal solution with $\mathbf{D}^* = -\nabla \mathbf{F}(\mathbf{X}^*)$. Under Assumptions 5-8, for any constant stepsize $\eta \in (0, 2/(\mu + L)]$, if the compression parameters α and γ satisfy*

$$\gamma \in \left(0, \min\left\{\frac{2}{(3C+1)\beta}, \frac{2\mu\eta(2-\mu\eta)}{[2-\mu\eta(2-\mu\eta)]C\beta}\right\}\right), \tag{3.14}$$

$$\alpha \in \left[\frac{C\beta\gamma}{2(1+C)}, \frac{1}{a_1}\min\left\{\frac{2-\beta\gamma}{4-\beta\gamma}, \mu\eta(2-\mu\eta)\right\}\right], \tag{3.15}$$

with $\beta \coloneqq \lambda_{\max}(\mathbf{I} - \mathbf{W})$. Then, in total expectation we have

$$\frac{1}{n}\mathbb{E}\mathcal{L}^{k+1} \leq \rho\frac{1}{n}\mathbb{E}\mathcal{L}^k + \eta^2\sigma^2, \tag{3.16}$$

where

$$\mathcal{L}^k \coloneqq (1 - a_1\alpha)\|\mathbf{X}^k - \mathbf{X}^*\|^2 + (2\eta^2/\gamma)\mathbb{E}\|\mathbf{D}^k - \mathbf{D}^*\|^2_{(\mathbf{I}-\mathbf{W})^\dagger} + a_1\|\mathbf{H}^k - \mathbf{X}^*\|^2, \tag{3.17}$$

$$\rho \coloneqq \max\left\{\frac{1-\mu\eta(2-\mu\eta)}{1-a_1\alpha}, 1 - \frac{\gamma}{2\lambda_{\max}((\mathbf{I}-\mathbf{W})^\dagger)}, \ 1-\alpha\right\} < 1, a_1 \coloneqq \frac{4(1+C)}{C\beta\gamma + 2} \tag{3.18}$$

The result holds for $C \to 0$.

Corollary 4 (Complexity bounds)**.** *Define the condition numbers of the objective function and communication graph as* $\kappa_f = \frac{L}{\mu}$ *and* $\kappa_g = \frac{\lambda_{\max}(\mathbf{I}-\mathbf{W})}{\lambda^+_{\min}(\mathbf{I}-\mathbf{W})}$*, respectively. Under the same setting in Theorem 3, we can choose* $\eta = \frac{1}{L}$, $\gamma = \min\{\frac{1}{C\beta\kappa_f}, \frac{1}{(1+3C)\beta}\}$, *and* $\alpha = O(\frac{1}{(1+C)\kappa_f})$ *such that*

$$\rho = \max\left\{1 - O\Big(\frac{1}{(1+C)\kappa_f}\Big), 1 - O\Big(\frac{1}{(1+C)\kappa_g}\Big), 1 - O\Big(\frac{1}{C\kappa_f\kappa_g}\Big)\right\}.$$

With full-gradient (i.e., $\sigma = 0$*), we obtain the following complexity bounds:*

- *LEAD converges to the* ϵ*-accurate solution with the iteration complexity*

$$O\Big(\big((1+C)(\kappa_f+\kappa_g) + C\kappa_f\kappa_g\big)\log\frac{1}{\epsilon}\Big).$$

- *When* $C = 0$ *(i.e., there is no compression), we obtain* $\rho = \max\{1 - O(\frac{1}{\kappa_f}), 1 - O(\frac{1}{\kappa_g})\}$, *and the iteration complexity* $O\Big((\kappa_f+\kappa_g)\log\frac{1}{\epsilon}\Big)$*. This exactly recovers the convergence rate of NIDS [59].*

- *When* $C \le \frac{\kappa_f+\kappa_g}{\kappa_f\kappa_g+\kappa_f+\kappa_g}$*, the asymptotical complexity is* $O\Big((\kappa_f+\kappa_g)\log\frac{1}{\epsilon}\Big)$*, which also recovers that of NIDS [59] and indicates that the compression doesn't harm the convergence in this case.*

- *With* $C = 0$ *(or* $C \le \frac{\kappa_f+\kappa_g}{\kappa_f\kappa_g+\kappa_f+\kappa_g}$*) and fully connected communication graph (i.e.,* $\mathbf{W} = \frac{\mathbf{1}\mathbf{1}^\top}{n}$*), we have* $\beta = 1$ *and* $\kappa_g = 1$*. Therefore, we obtain* $\rho = 1 - O(\frac{1}{\kappa_f})$ *and the complexity bound* $O(\kappa_f log\frac{1}{\epsilon})$*. This recovers the convergence rate of gradient descent [81].*

Remark 6. *Under the setting in Theorem 3, LEAD converges linearly to the* $O(\sigma^2)$ *neighborhood of the optimum and converges linearly exactly to the optimum if full gradient is used, e.g.,* $\sigma = 0$*. The linear convergence of LEAD holds when* $\eta < 2/L$*, but we omit the proof.*

Remark 7 (Arbitrary compression precision)**.** *Pick any* $\eta \in (0, 2/(\mu+L)]$*, based on the compression-related constant* C *and the network-related constant* β*, we can select* γ *and* α *in certain ranges to achieve the convergence. It suggests that LEAD supports unbiased compression with arbitrary precision, i.e., any* $C > 0$*.*

Corollary 5 (Consensus error)**.** *Under the same setting in Theorem 3 , let* $\bar{\mathbf{x}}^k = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i^k$ *be the averaged model and* $\mathbf{H}^0 = \mathbf{H}^1$*, then all agents achieve consensus at the rate*

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left\|\mathbf{x}_i^k - \bar{\mathbf{x}}^k\right\|^2 \leq \frac{2\mathcal{L}^0}{n}\rho^k + \frac{2\sigma^2}{1-\rho}\eta^2. \tag{3.19}$$

where ρ *is defined as in Corollary 4 with appropriate parameter settings.*

Theorem 4 (Diminishing stepsize)**.** *Let* $\{\mathbf{X}^k, \mathbf{H}^k, \mathbf{D}^k\}$ *be the sequence generated from Alg. 3 and* $\mathbf{X}^*$ *is the optimal solution with* $\mathbf{D}^* = -\nabla\mathbf{F}(\mathbf{X}^*)$*. Under Assumptions 5-8, if* $\eta_k = \frac{2\theta_5}{\theta_3\theta_4\theta_5 k+2}$ *and* $\gamma_k = \theta_4\eta_k$*, by taking* $\alpha_k = \frac{C\beta\gamma_k}{2(1+C)}$*, in total expectation we have*

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left\|\mathbf{x}_i^k - \mathbf{x}^*\right\|^2 \lesssim \mathcal{O}\left(\frac{1}{k}\right) \tag{3.20}$$

where $\theta_1, \theta_2, \theta_3, \theta_4$ *and* θ_5 *are constants defined in the proof. The complexity bound for arriving at the* ϵ*-accurate solution is* $\mathcal{O}(\frac{1}{\epsilon})$*.*

Remark 8. *Compared with CHOCO-SGD, LEAD requires unbiased compression and the convergence under biased compression is not investigated yet. The analysis of CHOCO-SGD relies on the bounded gradient assumptions, i.e.,* $\|\nabla f_i(\mathbf{x})\|^2 \leq G$*, which is restrictive because it conflicts with the strong convexity while LEAD doesn't need this assumption. Moreover, in the theorem of CHOCO-SGD, it requires a specific point set of* γ *while LEAD only requires* γ *to be within a rather large range. This may explain the advantages of LEAD over CHOCO-SGD in terms of robustness to parameter setting.*

3.5 Numerical Experiment

We consider three machine learning problems – ℓ_2-regularized linear regression, logistic regression, and deep neural network. The proposed LEAD is compared with QDGD [88], DeepSqueeze [102], CHOCO-SGD [51], and two non-compressed algorithms DGD [123] and NIDS [59].

Setup. We consider eight machines connected in a ring topology network. Each agent can only exchange information with its two 1-hop neighbors. The mixing weight is simply set as $1/3$. For

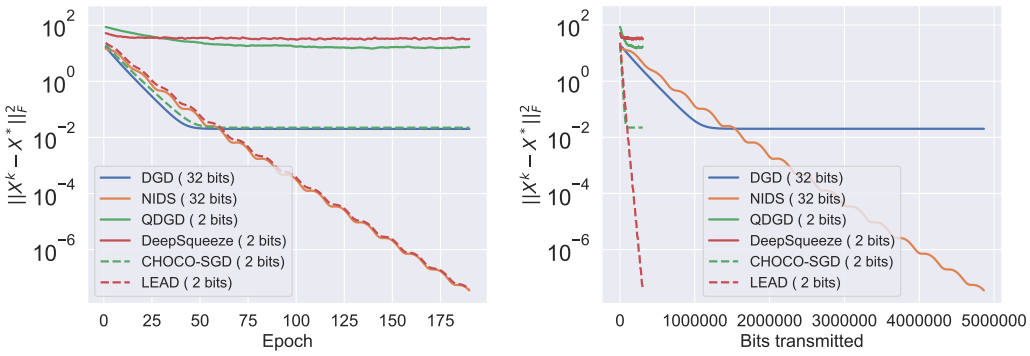
compression, we use the unbiased b-bits quantization method with ∞-norm

$$Q_\infty(\mathbf{x}) := \left(\|\mathbf{x}\|_\infty 2^{-(b-1)}\text{sign}(\mathbf{x})\right) \cdot \left\lfloor \frac{2^{(b-1)}|\mathbf{x}|}{\|\mathbf{x}\|_\infty} + \mathbf{u} \right\rfloor, \tag{3.21}$$

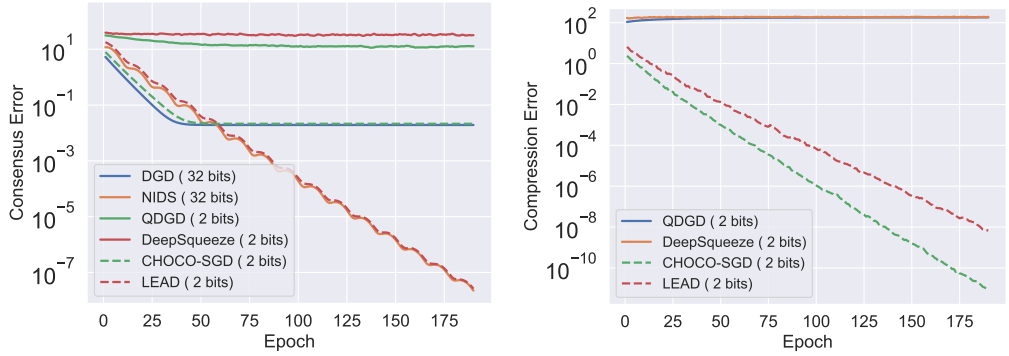
where $\cdot$ is the Hadamard product, $|\mathbf{x}|$ is the elementwise absolute value of $\mathbf{x}$, and $\mathbf{u}$ is a random vector uniformly distributed in $[0,1]^d$. Only $\text{sign}(\mathbf{x})$, norm $\|\mathbf{x}\|_\infty$, and integers in the bracket need to be transmitted. Note that this quantization method is similar to the quantization used in QSGD [4] and CHOCO-SGD [51], but we use the ∞-norm scaling instead of the 2-norm. This small change brings significant improvement on compression precision as justified both theoretically and empirically in Appendix B.1. In this section, we choose 2-bit quantization and quantize the data blockwise (block size = 512).

For all experiments, we tune the stepsize η from $\{0.01, 0.05, 0.1, 0.5\}$. For QDGD, CHOCO-SGD and Deepsqueeze, γ is tuned from $\{0.01, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$. Note that different notations are used in their original papers. Here we uniformly denote the stepsize as η and the additional parameter in these algorithms as γ for simplicity. For LEAD, we simply fix $\alpha = 0.5$ and $\gamma = 1.0$ for all experiments since we find LEAD is robust to parameter settings as we validate in the parameter sensitivity analysis in the below. This indicates the minor effort needed for tuning LEAD. Detailed parameter settings for all experiments are summarized in Appendix B.2.2.

Linear regression. We consider the problem: $f(\mathbf{x}) = \sum_{i=1}^n (\|\mathbf{A}_i\mathbf{x} - \mathbf{b}_i\|^2 + \lambda\|\mathbf{x}\|^2)$. Data matrices $\mathbf{A}_i \in \mathbb{R}^{200\times 200}$ and the true solution $\mathbf{x}'$ is randomly synthesized. The values $\mathbf{b}_i$ are generated by adding Gaussian noise to $\mathbf{A}_i\mathbf{x}'$. We let $\lambda = 0.1$ and the optimal solution of the linear regression problem be $\mathbf{x}^*$. We use full-batch gradient to exclude the impact of gradient variance. The performance is showed in Fig. 3.1. The distance to $\mathbf{x}^*$ in Fig. 3.1a and the consensus error in Fig. 3.1c verify that LEAD converges exponentially to the optimal consensual solution. It significantly outperforms most baselines and matches NIDS well under the same number of iterations. Fig. 3.1b demonstrates the benefit of compression when considering the communication bits. Fig. 3.1d shows that the compression error vanishes for both LEAD and CHOCO-SGD while the compression error is pretty large for QDGD and DeepSqueeze because they directly compress the local models.



(a) $\|\mathbf{X}^k - \mathbf{X}^*\|_F$ (b) $\|\mathbf{X}^k - \mathbf{X}^*\|_F$

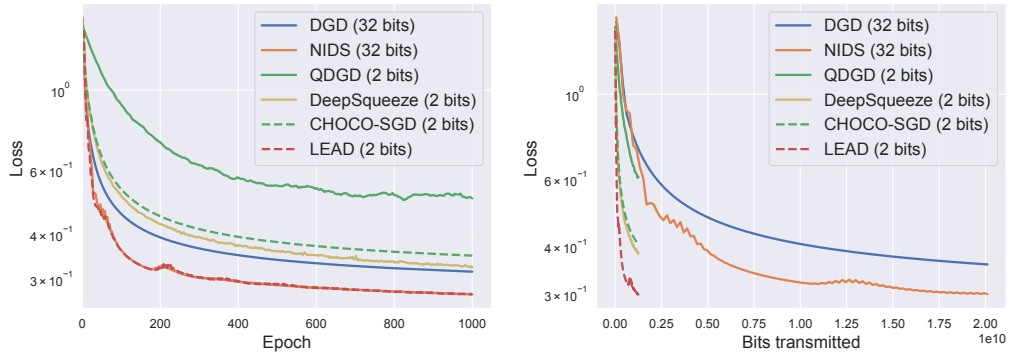


(c) $\|\mathbf{X}^k - \mathbf{1}_{n\times 1}\bar{\mathbf{X}}^k\|_F$ (d) Compression error

Figure 3.1: Linear regression problem.

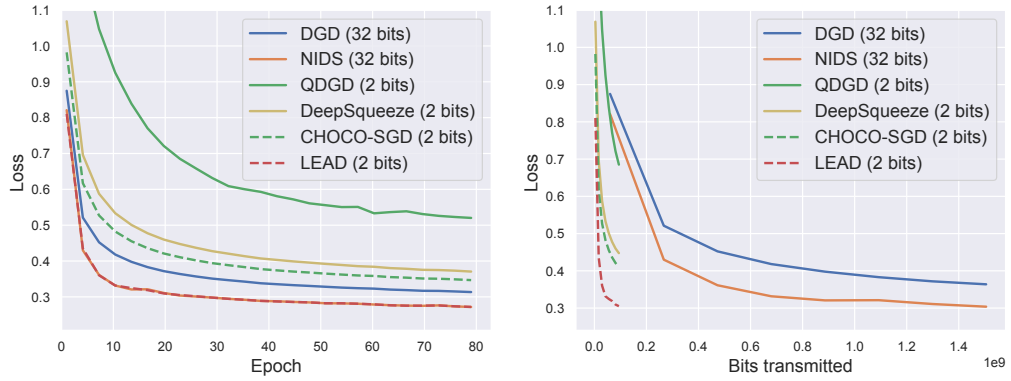
Logistic regression. We further consider a logistic regression problem on the MNIST dataset. The regularization parameter is 10^{-4}. We consider both *homogeneous* and *heterogeneous* data settings. In the homogeneous setting, the data samples are randomly shuffled before being uniformly partitioned among all agents such that the data distribution from each agent is very similar. In the heterogeneous setting, the samples are first sorted by their labels and then partitioned among agents. Due to the space limit, we mainly present the results in heterogeneous setting here and defer the homogeneous setting to Appendix B.2.1. The results using full-batch gradient and mini-batch gradient (the mini-batch size is 512 for each agent) are showed in Fig. 3.2 and Fig. 3.3 respectively and both settings shows the faster convergence and higher precision of LEAD.

Neural network. We empirically study the performance of LEAD in optimizing deep neural network by training AlexNet (240 MB) on CIFAR10 dataset. The mini-batch size is 64 for each agents. Both the homogeneous and heterogeneous case are showed in Fig. 3.4. In the homogeneous



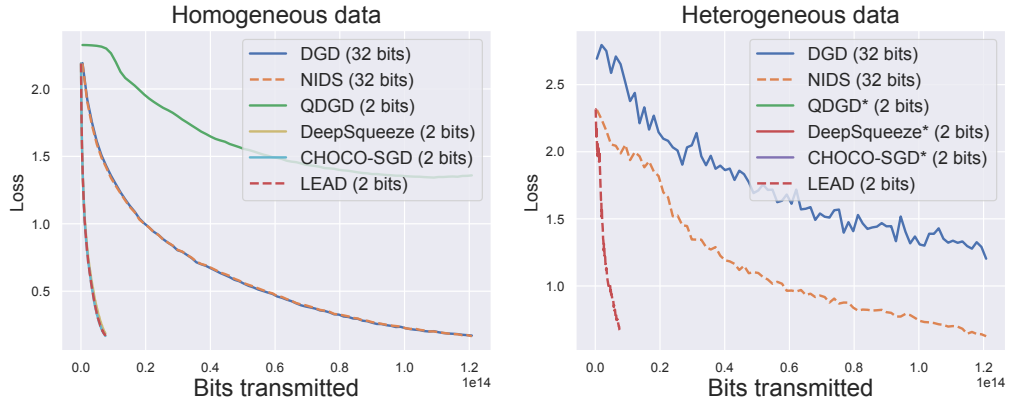
(a) Loss $f(\overline{\mathbf{X}}^k)$ (b) Loss $f(\overline{\mathbf{X}}^k)$

Figure 3.2: Logistic regression problem in the heterogeneous case (full-batch gradient).



(a) Loss $f(\overline{\mathbf{X}}^k)$ (b) Loss $f(\overline{\mathbf{X}}^k)$

Figure 3.3: Logistic regression in the heterogeneous case (mini-batch gradient).



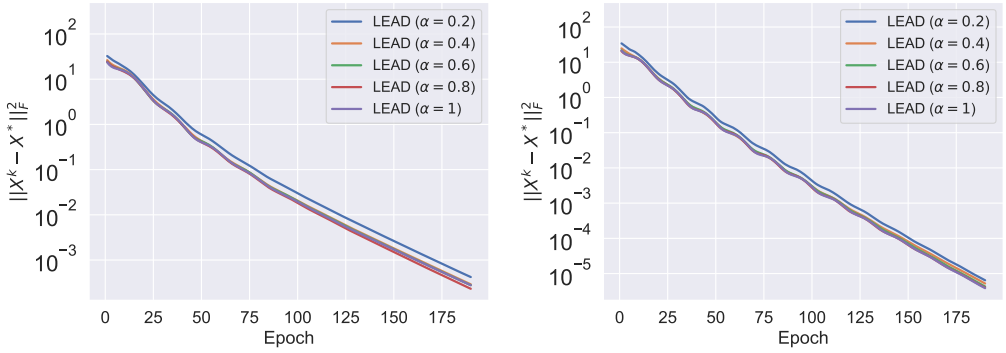
(a) Loss $f(\overline{\mathbf{X}}^k)$ (b) Loss $f(\overline{\mathbf{X}}^k)$

Figure 3.4: Stochastic optimization on deep neural network ($*$ means divergence).

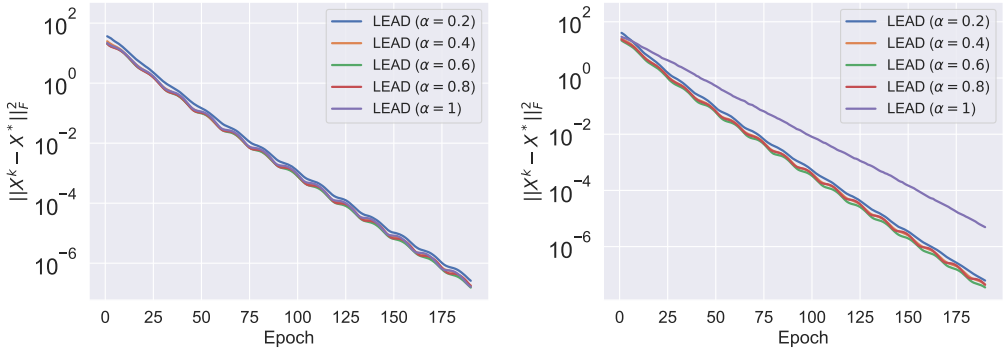
case, CHOCO-SGD, DeepSqueeze and LEAD perform similarly and outperform the non-compressed variants in terms of communication efficiency, but CHOCO-SGD and DeepSqueeze need more efforts for parameter tuning because their convergence is sensitive to the setting of γ. In the

heterogeneous cases, LEAD achieves the fastest and most stable convergence. Note that in this setting, sufficient information exchange is more important for convergence because models from different agents are moving to significantly diverse directions. In such case, DGD only converges with smaller stepsize and its communication compressed variants, including QDGD, DeepSqueeze and CHOCO-SGD, diverge in all parameter settings we try.

Parameter sensitivity. In the linear regression problem, the convergence of LEAD under different parameter settings of α and γ are tested. The result showed in Figure 3.5 indicates that LEAD performs well in most settings and is robust to the parameter setting. Therefore, in this work, we simply set $\alpha = 0.5$ and $\gamma = 1.0$ for LEAD in all experiment, which indicates the minor effort needed for parameter tuning.



(a) $\gamma = 0.4$ (b) $\gamma = 0.6$



(c) $\gamma = 0.8$ (d) $\gamma = 1.0$

Figure 3.5: Parameter analysis on linear regression problem.

In summary, our experiments verify our theoretical analysis and show that LEAD is able to handle data heterogeneity very well. Furthermore, the performance of LEAD is robust to parameter

settings and needs less effort for parameter tuning, which is critical in real-world applications.

3.6 Conclusion

In this work, we investigate the communication compression in message passing for decentralized optimization. Motivated by primal-dual algorithms, a novel decentralized algorithm with compression, LEAD, is proposed to achieve faster convergence rate and to better handle heterogeneous data while enjoying the benefit of efficient communication. The nontrivial analyses on the coupled dynamics of inexact primal and dual updates as well as compression error establish the linear convergence of LEAD when full gradient is used and the linear convergence to the $O(\sigma^2)$ neighborhood of the optimum when stochastic gradient is used. Extensive experiments validate the theoretical analysis and demonstrate the state-of-the-art efficiency and robustness of LEAD. LEAD is also applicable to non-convex problems as empirically verified in the neural network experiments. In addition, we also proposed a linear convergent decentralized algorithm with compression (ProxLEAD) for composite optimization problems [56].

CHAPTER 4

GRAPH NEURAL NETWORKS WITH ADAPTIVE RESIDUAL

Graph neural networks (GNNs) have shown the power in graph representation learning for numerous tasks. In this chapter, we discover an interesting phenomenon that although residual connections in the message passing of GNNs help improve the performance, they immensely amplify GNNs' vulnerability against abnormal node features. This is undesirable because in real-world applications, node features in graphs could often be abnormal such as being naturally noisy or adversarially manipulated. We analyze possible reasons to understand this phenomenon and aim to design GNNs with stronger resilience to abnormal features. Our understandings motivate us to propose and derive a simple, efficient, interpretable, and adaptive message passing scheme, leading to a novel GNN with Adaptive residual, AirGNN. Extensive experiments under various abnormal feature scenarios demonstrate the effectiveness of the proposed algorithm. The implementation is available at https://github.com/lxiaorui/AirGNN.

4.1 Introduction

Recent years have witnessed the great success of graph neural networks (GNNs) in representation learning for graph structure data [74]. Essentially, GNNs generalize deep neural networks (DNNs) from regular grids, such as image, video and text, to irregular data such as social, energy, transportation, citation, and biological networks. Such data can be naturally represented as graphs with nodes and edges. The key building block for such generalization is the neural message passing framework [29]:

$$\mathbf{x}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{x}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right) \tag{4.1}$$

where $\mathbf{x}_u^{(k)} \in \mathbb{R}^d$ denotes the feature vector of node u in the k-th iteration of message passing, and $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ is the message aggregated from u's neighborhood $\mathcal{N}(u)$. The specific design of message passing scheme can be motivated from spectral domain [48, 23] or spatial domain [33, 109, 91, 29].

It usually linearly smooths the features in a local neighborhood on the graph.

GNNs have achieved superior performance in a large number of benchmark datasets [117] where the node features are assumed to be complete and informative. However, in real-world applications, some node features could be abnormal from various aspects. For instance, in social networks, new users might not have complete profile before they make connections with others, leading to missing user features. In transportation networks, node features can be noisy since there exist certain uncertainty and dynamics in the observation of the traffic information. What is worse, node features can be adversarially chosen by the attacker to maliciously manipulate the prediction made by GNNs. Therefore, it is greatly desired to design GNN models with stronger resilience to abnormal node features.

In this work, we first perform empirical investigations on how representative GNN models behave on graphs with abnormal features. Specifically, based upon standard benchmark datasets, we simulate the abnormal features by replacing the features of randomly selected nodes with random Gaussian noise. Then the performance of node classification on abnormal features and normal features are examined separately. From our preliminary study in Section 4.2, we reveal two interesting observations: (1) Feature aggregation can boost the resilience to abnormal features, but too many aggregations could hurt the performance on both normal and abnormal features; and (2) Residual connection helps GNNs benefit from more layers for normal features, while making GNNs more fragile to abnormal features. We then provide possible explanations to understand these observed phenomena from the perspective of graph Laplacian smoothing. Our analyses imply that there might exist an intrinsic tension between feature aggregation and residual connection, which results in a performance tradeoff between normal features and abnormal features.

Motivated by these findings and understandings, we aim to design new GNNs with stronger resilience to abnormal features while largely maintaining the performance on normal features. Our contributions can be summarized as follows:

- We discover an intrinsic tension between feature aggregation and residual connection in GNNs, and the corresponding performance tradeoff between abnormal and normal features.

We also analyze possible reasons to explain and understand these findings.

- We propose a simple, efficient, principled and adaptive message passing scheme, which leads to a novel GNN model with adaptive residual, named as AirGNN.
- Extensive experiments under various abnormal feature scenarios demonstrate the superiority of the proposed algorithm. The ablation study demonstrates how the adaptive residuals mitigate the impact of abnormal features.

4.2 Preliminary

Before introducing the preliminary study, we first define the notations used throughout the paper.

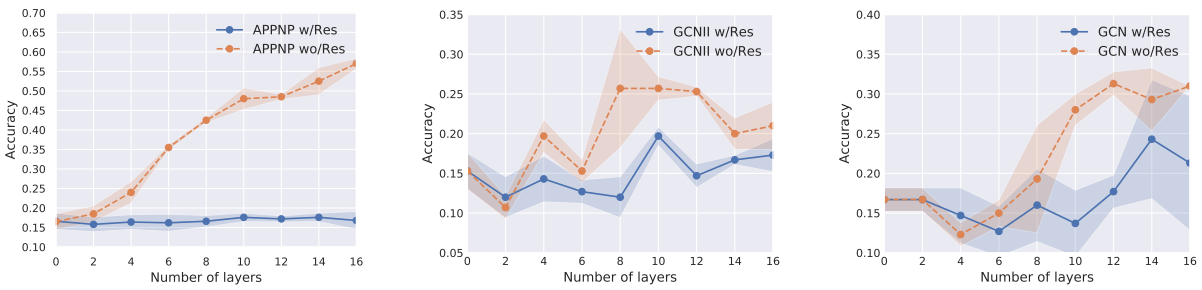
Notations. We use bold upper-case letters such as $\mathbf{X}$ to denote matrices. Given a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, we use $\mathbf{X}_i$ to denote its i-th row and $\mathbf{X}_{ij}$ to denote its element in i-th row and j-th column. The Frobenius norm and ℓ_{21} norm of a matrix $\mathbf{X}$ are defined as $\|\mathbf{X}\|_F = \sqrt{\sum_{ij} \mathbf{X}_{ij}^2}$ and $\|\mathbf{X}\|_{21} = \sum_i \|\mathbf{X}_i\|_2 = \sum_i \sqrt{\sum_j \mathbf{X}_{ij}^2}$, respectively. We define $\|\mathbf{X}\|_2 = \sigma_{\max}(\mathbf{X})$ where $\sigma_{\max}(\mathbf{X})$ is the largest singular value of $\mathbf{X}$.

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a graph with the node set $\mathcal{V} = \{v_1, \ldots, v_n\}$ and the undirected edge set $\mathcal{E} = \{e_1, \ldots, e_m\}$. We use $\mathcal{N}(v_i)$ to denote the neighboring nodes of node v_i, including v_i itself. Suppose that each node is associated with a d-dimensional feature vector, and the features for all nodes are denoted as $\mathbf{X}_{\text{fea}} \in \mathbb{R}^{n\times d}$. The graph structure $\mathcal{G}$ can be represented as an adjacent matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, where $\mathbf{A}_{ij} = 1$ when there exists an edge between nodes v_i and v_j, and $\mathbf{A}_{ij} = 0$ otherwise. The graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal degree matrix. Let us denote the commonly used feature aggregation matrix in GNNs [48] as $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$ where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacent matrix with self-loop and its degree matrix is $\hat{\mathbf{D}}$. The corresponding Laplacian matrix is defined as $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$.

In this work, we focus on the setting where a subset of nodes in the graph contain abnormal features, while the remaining nodes have normal features. In the remaining of this chapter, we use *abnormal/normal features* to denote *nodes with abnormal/normal features*, for simplicity.

4.2.1 Preliminary Study

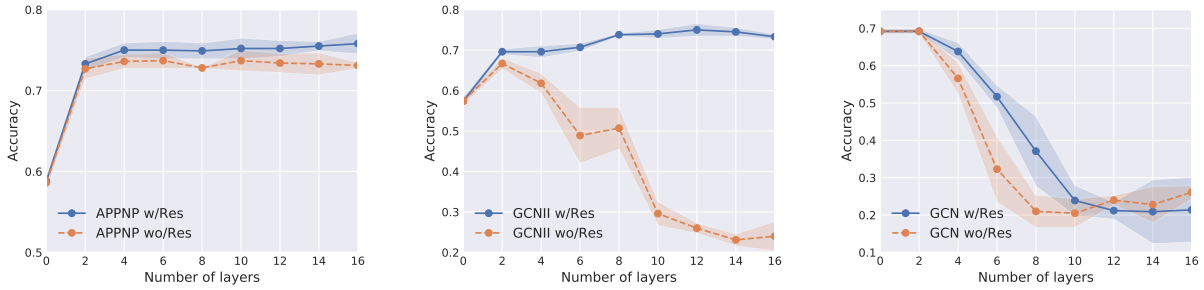
Experimental setup. To investigate how GNNs behave on abnormal and normal node features, we design semi-supervised node classification experiments on three common datasets (i.e., Cora, CiteSeer and PubMed), following the data splits in the work [48]. Moreover, we simulate the abnormal features by assigning 10% of the nodes with random features sampled from a standard Gaussian distribution. The experiments are performed on representative GNN models covering coupled and decoupled architectures, including GCN [48], GCNII [14], APPNP [49], and their variants with or without residual connections in feature aggregations, denoted as w/Res and wo/Res. All methods follow the hyperparameter settings in their original papers. We examine how these models perform when the number of layers increases. Note that for the decoupled architectures such as APPNP, we fix the 2-layer MLP and increase the number of propagation layers. While for the coupled architectures such as GCN and GCNII, we increase the number of feature transformation and propagation layers simultaneously. We report the average performance over 10 times of random selection of the noise node sets. The node classification accuracy (mean and standard variance) on nodes with abnormal and normal features is illustrated in Figure 4.1 and Figure. 4.2, separately.



(a) APPNP (b) GCNII (c) GCN

Figure 4.1: Node classification accuracy on abnormal nodes (Cora).

Observations. From Figure 4.1 and Figure 4.2, we can make the following observations: (1) Without residual connection, more layers (e.g., > 2 for GCN and GCNII, > 10 for APPNP) hurt the accuracy on nodes with normal features. However, more layers boost the accuracy on nodes with abnormal features significantly, before finally starting to decrease; (2) With residual connection, the



(a) APPNP (b) GCNII (c) GCN

Figure 4.2: Node classification accuracy on normal nodes (Cora).

accuracy on nodes with normal features keeps increasing with more layers[1]. However, the accuracy on nodes with abnormal features only increases marginally when stacking more layers, and then starts to decrease. While we only present the experiments on Cora, we defer the results on other datasets to Appendix C.1, which provide similar observations. To conclude, we can summarize these observations into two major findings:

- *Finding I:* Feature aggregation can boost the resilience to abnormal features, but too many aggregations could hurt the performance on both normal and abnormal nodes;
- *Finding II:* Residual connection helps GNNs benefit from more layers for nodes with normal features, while making GNNs more fragile to abnormal features.

4.2.2 Understandings

In this subsection, we provide the understanding and explanation for aforementioned findings, from the perspective of graph Laplacian smoothing.

Understanding Finding I: Feature aggregation as Laplacian smoothing

The message passing in GCN [48], GCNII wo/ residual and APPNP wo/ residual (as well as many popular GNN models), follows the feature aggregation

$$\mathbf{X}_{\text{out}} = \tilde{\mathbf{A}}\mathbf{X}_{\text{in}}, \tag{4.2}$$

[1]GCN w/Res is an exception because its residual is not appropriate, which is consistent with the experiments in the work [48].

where $\mathbf{X}_{\text{in}}$ and $\mathbf{X}_{\text{out}}$ represent the features before and after message passing layer, respectively. It can be interpreted as one gradient descent step for the Laplacian smoothing problem [73]

$$\underset{\mathbf{X}\in\mathbb{R}^{n\times d}}{\arg\min}\ \mathcal{L}_1(\mathbf{X}) \coloneqq \frac{1}{2}\text{tr}\Big(\mathbf{X}^\top(\mathbf{I}-\tilde{\mathbf{A}})\mathbf{X}\Big) = \frac{1}{2}\sum_{(v_i,v_j)\in\mathcal{E}} \|\frac{\mathbf{X}_i}{\sqrt{d_i+1}} - \frac{\mathbf{X}_j}{\sqrt{d_j+1}}\|_2^2, \tag{4.3}$$

where d_i is the node degree of node v_i. Eq. (4.2) can be derived from $\mathbf{X}_{\text{out}} = \mathbf{X}_{\text{in}} - (\mathbf{I}-\tilde{\mathbf{A}})\mathbf{X}_{\text{in}} = \tilde{\mathbf{A}}\mathbf{X}_{\text{in}}$, with the initialization $\mathbf{X} = \mathbf{X}_{\text{in}}$ and stepsize $\gamma = 1$. The Laplacian smoothing problem penalizes the feature difference between neighboring nodes. To reduce this penalty, the feature aggregation in Eq. (4.2) smooths the node features by taking the average of local neighbors, and thus can be considered as low-pass filter which gradually filters out high-frequency signals [84, 126]. Therefore, it increases the resilience to abnormal features which are likely to be high-frequency signals. In other words, the local neighboring nodes help to correct the abnormal features. Unfortunately, if applied too many times, these low-pass filters could overly smooth the features (well-known as oversmoothing [55, 85]) such that nodes are not distinguishable enough, providing an explanation to the degraded performance on both abnormal and normal features when stacking too many layers.

Understanding Finding II: Residual connection maintains feature proximity

To adjust the feature smoothness for better performance, APPNP [49] utilizes residual connections in message passing as follows

$$\mathbf{X}^{k+1} = (1-\alpha)\tilde{\mathbf{A}}\mathbf{X}^k + \alpha\mathbf{X}_{\text{in}}, \tag{4.4}$$

where $\mathbf{X}^0 = \mathbf{X}_{\text{in}}$. It can be considered as an iterative solution for the regularized Laplacian smoothing problem [73]

$$\underset{\mathbf{X}\in\mathbb{R}^{n\times d}}{\arg\min}\ \mathcal{L}_2(\mathbf{X}) \coloneqq \frac{\alpha}{2(1-\alpha)}\|\mathbf{X}-\mathbf{X}_{\text{in}}\|_F^2 + \frac{1}{2}\text{tr}\Big(\mathbf{X}^\top(\mathbf{I}-\tilde{\mathbf{A}})\mathbf{X}\Big), \tag{4.5}$$

with initialization $\mathbf{X} = \mathbf{X}_{\text{in}}$ and stepsize $\gamma = 1-\alpha$ due to

$$\mathbf{X}^{k+1} = \mathbf{X}^k - (1-\alpha)\Big(\frac{\alpha}{1-\alpha}(\mathbf{X}^k - \mathbf{X}_{\text{in}}) + (\mathbf{I}-\tilde{\mathbf{A}})\mathbf{X}^k\Big) = (1-\alpha)\tilde{\mathbf{A}}\mathbf{X}^k + \alpha\mathbf{X}_{\text{in}}.$$

GCNII [14] adopts a similar message passing but further combines a feature transformation layer in each message passing step, which leads to a coupled architecture, as contrast to the decoupled

architecture of APPNP. The residual connection naturally arises when regularizing the proximity between input and output features, as showed in the first term of $\mathcal{L}_2(\mathbf{X})$. Such proximity can help avoid the trivial solution for the problem in Eq. (4.3), i.e., totally oversmoothed features only depending on node degrees, and consequently mitigates the oversmoothing issue. More intuitively, residual connections in GNNs provide direct information flows between layers that can preserve some necessary high-frequency signals for better discrimination between classes. More layers with residual provide a more accurate solution to Eq. (4.5), which explains the performance gain from deeper GNNs. Unfortunately, these residual connections also undesirably carry on abnormal features which are detrimental, leading to the inferior performance on abnormal features.

4.3 Algorithm

In this section, we first motivate the proposed adaptive message passing scheme (AMP) with further discussions on our preliminary study. We then introduce more details about AMP, its interpretations, convergence guarantee and computation complexity, as well as the model architecture of AirGNN.

4.3.1 Design Motivation

Our preliminary study in Section 4.2 reveals an intrinsic tension between feature aggregation and residual connection: (1) feature aggregation helps smooth out abnormal features, while it could cause inappropriate smoothing for normal features; (2) residual connection is essential for adjusting the feature smoothness, but it could be detrimental for abnormal features. Although this conflict can be partially mitigated by adjusting the residual connection such as the residual weight α in GCNII [14] and APPNP [49], such global adjustment cannot be adaptive to a subset of the nodes, e.g., the nodes with abnormal features. This is crucial because in practice we often encounter the scenario where only a subset of nodes contain abnormal features. Therefore, how to reconcile this dilemma still desires dedicated efforts. We then naturally ask a question: *Can we design a better message passing scheme with node-wise adaptive feature aggregation and residual connection?*

The motivation of the proposed idea builds upon the following intuition: while it is important to

maintain the proximity between input and output features as in Eq. (4.5), it could be over aggressive to penalize their deviations by the square of Frobenius norm, i.e., $\|\mathbf{X} - \mathbf{X}_{\text{in}}\|_F^2 = \sum_{i=1}^n \|\mathbf{X}_i - (\mathbf{X}_{\text{in}})_i\|_2^2$. The fact that this penalty does not tolerate large deviations weakens the capability to remove abnormal features through Laplacian smoothing. This motivates us to consider an alternative proximity penalty

$$\|\mathbf{X} - \mathbf{X}_{\text{in}}\|_{21} \coloneqq \sum_{i=1}^{n} \|\mathbf{X}_i - (\mathbf{X}_{\text{in}})_i\|_2, \tag{4.6}$$

which instead penalizes the deviations by the ℓ_1 norm of row-wise ℓ_2 norms, namely ℓ_{21} norm. The ℓ_{21} norm promotes row sparsity in $\mathbf{X} - \mathbf{X}_{\text{in}}$, and it also allows large deviations because the penalty on large values is less aggressive, leading to the potential removal of abnormal features. Therefore, we propose the following Laplacian smoothing problem regularized by ℓ_{21} norm proximity control:

$$\arg\min_{\mathbf{X}\in\mathbb{R}^{n\times d}} \lambda\|\mathbf{X} - \mathbf{X}_{\text{in}}\|_{21} + \frac{1}{2}\text{tr}(\mathbf{X}^\top(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{X}), \tag{4.7}$$

where $\lambda \in [0, \infty)$ is a parameter to adjust the balance between proximity and Laplacian smoothing. In order to easy the tuning of λ, we made a modification of Eq. (4.7):

$$\arg\min_{\mathbf{X}\in\mathbb{R}^{n\times d}} \mathcal{L}(\mathbf{X}) \coloneqq \lambda\|\mathbf{X} - \mathbf{X}_{\text{in}}\|_{21} + (1-\lambda)\text{tr}(\mathbf{X}^\top(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{X}), \tag{4.8}$$

where $\lambda \in [0, 1]$ controls the balance.

4.3.2 Adaptive Message Passing

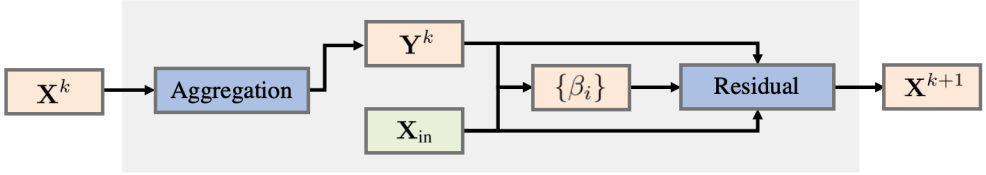


Figure 4.3: Diagram of Adaptive Message Passing.

$\mathcal{L}(\mathbf{X})$ is a composite objective with non-smooth and smooth components. We optimize it by proximal gradient descent [9] and obtain the following iterations as the adaptive message passing

(AMP):

$$\mathbf{Y}^k = \mathbf{X}^k - 2\gamma(1-\lambda)(\mathbf{I}-\tilde{\mathbf{A}})\mathbf{X}^k = \big(1-2\gamma(1-\lambda)\big)\mathbf{X}^k + 2\gamma(1-\lambda)\tilde{\mathbf{A}}\mathbf{X}^k \tag{4.9}$$

$$\mathbf{X}^{k+1} = \arg\min_{\mathbf{X}} \left\{ \lambda\|\mathbf{X}-\mathbf{X}_{\text{in}}\|_{21} + \frac{1}{2\gamma}\|\mathbf{X}-\mathbf{Y}^k\|_F^2 \right\} \tag{4.10}$$

where $\mathbf{X}^0 = \mathbf{X}_{\text{in}}$ and γ is the stepsize to be specified later. Let $\mathbf{Z} = \mathbf{X} - \mathbf{X}_{\text{in}}$, and Eq. (4.10) can be rewritten as:

$$\begin{aligned}\mathbf{Z}^{k+1} &= \arg\min_{\mathbf{Z}} \left\{ \lambda\|\mathbf{Z}\|_{21} + \frac{1}{2\gamma}\|\mathbf{Z}-(\mathbf{Y}^k-\mathbf{X}_{\text{in}})\|_F^2 \right\} \\ &= \mathbf{prox}_{\gamma\lambda\|\cdot\|_{21}}(\mathbf{Y}^k - \mathbf{X}_{\text{in}}) \qquad (4.11)\\ \mathbf{X}^{k+1} &= \mathbf{X}_{\text{in}} + \mathbf{Z}^{k+1}. \qquad (4.12)\end{aligned}$$

The i-th row of the proximal operator in Eq. (4.11) can be computed analytically

$$\big(\mathbf{prox}_{\gamma\lambda\|\cdot\|_{21}}(\mathbf{X})\big)_i = \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_2}\max(\|\mathbf{X}_i\|_2 - \gamma\lambda, 0) = \max(1-\frac{\gamma\lambda}{\|\mathbf{X}_i\|_2}, 0)\cdot\mathbf{X}_i. \tag{4.13}$$

Note that the proximal operator returns $\mathbf{0}$ if the input vector is $\mathbf{0}$. Substituting $\mathbf{X}$ in Eq. (4.13) with $\mathbf{Y}^k - \mathbf{X}_{\text{in}}$ and combining Eq. (4.11) and Eq. (4.12), then Eq. (4.12) becomes

$$\mathbf{X}_i^{k+1} = (\mathbf{X}_{\text{in}})_i + \beta_i(\mathbf{Y}_i^k - (\mathbf{X}_{\text{in}})_i) = (1-\beta_i)(\mathbf{X}_{\text{in}})_i + \beta_i\mathbf{Y}_i^k, \quad \forall i \in [n], \tag{4.14}$$

where $\beta_i := \max(1 - \frac{\gamma\lambda}{\|\mathbf{Y}_i^k - (\mathbf{X}_{\text{in}})_i\|_2}, 0)$. To summarize, the proposed adaptive message passing (AMP) scheme is showed in Figure 4.4, and a diagram is showed in Figure 4.3. In detail, AMP works as follows:

- The first step takes a *feature aggregation* within the local neighbors with a self-loop weighted by $1-2\gamma(1-\lambda)$;
- The second step computes a weight $\beta_i \in [0,1]$ for each node v_i depending on the local deviation $\|\mathbf{Y}_i^k - (\mathbf{X}_{\text{in}})_i\|_2$.
- The final step takes a *linear combination* of input features $\mathbf{X}_{\text{in}}$ and the aggregated features $\mathbf{Y}^k$, where the node-wise residual is adaptively weighted by $1-\beta_i$ for each node v_i.

$$\mathbf{Y}^k = \big(1 - 2\gamma(1-\lambda)\big)\mathbf{X}^k + 2\gamma(1-\lambda)\tilde{\mathbf{A}}\mathbf{X}^k$$
$$\beta_i = \max(1 - \frac{\gamma\lambda}{\|\mathbf{Y}_i^k - (\mathbf{X}_{\text{in}})_i\|_2}, 0) \quad \forall i \in [n]$$
$$\mathbf{X}_i^{k+1} = (1-\beta_i)(\mathbf{X}_{\text{in}})_i + \beta_i \mathbf{Y}_i^k \qquad \forall i \in [n]$$

Figure 4.4: Adaptive Message Passing (AMP).

The convergence guarantee of AMP and parameter setting for the stepsize γ are illustrated in Theorem 5. According to Theorem 5, if we set $\gamma = \frac{1}{4(1-\lambda)}$ or $\gamma = \frac{1}{2(1-\lambda)}$, then the first step of AMP can be simplified as $\mathbf{Y}^k = \frac{1}{2}\mathbf{X}^k + \frac{1}{2}\tilde{\mathbf{A}}\mathbf{X}^k$ and $\mathbf{Y}^k = \tilde{\mathbf{A}}\mathbf{X}^k$, respectively. The choice of stepsize will only impact the convergence speed but not the ultimate effect of AMP when it convergences to the fixed point solution. We also discuss the computation complexity per iteration of AMP in Remark 9.

Theorem 5 (Convergence of AMP)**.** *Under the stepsize setting $\gamma < \frac{1}{(1-\lambda)\|\tilde{\mathbf{L}}\|_2}$, the proposed adaptive message passing scheme (AMP) in Eq.* (4.9) *and Eq.* (4.10) *converges to the optimal solution of the problem defined in Eq.* (4.8)*. In practice, it is sufficient to choose any $\gamma < \frac{1}{2(1-\lambda)}$ since $\|\tilde{\mathbf{L}}\|_2 \leq 2$. Moreover, if the connected components of the graph $\mathcal{G}$ are not bipartite graphs, it is sufficient to choose $\gamma = \frac{1}{2(1-\lambda)}$ since $\|\tilde{\mathbf{L}}\|_2 < 2$.*

Proof. The objective that the iterations in AMP try to optimize is

$$\underset{\mathbf{X}\in\mathbb{R}^{n\times d}}{\arg\min} \ \mathcal{L}(\mathbf{X}) \coloneqq \underbrace{\lambda\|\mathbf{X} - \mathbf{X}_{\text{in}}\|_{21}}_{g(\mathbf{X})} + \underbrace{(1-\lambda)\text{tr}(\mathbf{X}^\top(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{X})}_{f(\mathbf{X})}, \tag{4.15}$$

where f and g are both convex functions. Moreover, g is a non-smooth function, while f is a smooth function. In particular, f is L-smoothness where $L = 2(1-\lambda)\|\tilde{\mathbf{L}}\|_2 = 2(1-\lambda)\|\mathbf{I} - \tilde{\mathbf{A}}\|_2$ due to

$$\|\nabla f(\mathbf{X}_1) - \nabla f(\mathbf{X}_2)\|_F = \|2(1-\lambda)\tilde{\mathbf{L}}(\mathbf{X}_1 - \mathbf{X}_2)\|_F \leq 2(1-\lambda)\|\tilde{\mathbf{L}}\|_2 \, \|\mathbf{X}_1 - \mathbf{X}_2\|_F. \tag{4.16}$$

AMP essentially applies a forward-backward splitting on the composite objective $g(\mathbf{X}) + f(\mathbf{X})$:

$$\mathbf{X}^{k+1} = (\mathbf{I} + \gamma\partial g)^{-1}(\mathbf{X}^k - \gamma\nabla f(\mathbf{X}^k)) \tag{4.17}$$

$$= \arg\min_{\mathbf{X}} \frac{1}{2}\|\mathbf{X} - (\mathbf{X}^k - \gamma\nabla f(\mathbf{X}^k))\|_F^2 + \gamma g(\mathbf{X}), \tag{4.18}$$

which is known as proximal gradient method. The convergence of this forward-backward splitting is ensured if the stepsize satifies $\gamma < \frac{2}{L}$ according to Lemma 4.4 in [20]. Therefore, AMP provably converges to the optimal solution under the setting $\gamma < \frac{1}{(1-\lambda)\|\tilde{\mathbf{L}}\|_2}$. For the symmetrically normalized Laplacian matrix, we have $\|\tilde{\mathbf{L}}\|_2 \leq 2$ [19] and thus $\frac{1}{2(1-\lambda)} \leq \frac{1}{(1-\lambda)\|\tilde{\mathbf{L}}\|_2}$. Therefore, any $\gamma < \frac{1}{2(1-\lambda)}$ will be sufficient. Moreover, according to [19], if the connected components of the graph $\mathcal{G}$ are not bipartite graphs, we have $\|\tilde{\mathbf{L}}\|_2 < 2$ and thus $\gamma = \frac{1}{2(1-\lambda)} < \frac{1}{(1-\lambda)\|\tilde{\mathbf{L}}\|_2}$ is sufficient.

□

Remark 9 (Computation complexity)**.** *AMP is as efficient as simple feature aggregation* $\mathbf{X}_{out} = \tilde{\mathbf{A}}\mathbf{X}_{in}$ *because the additional computation cost from the second and third steps in Figure 4.4 is in the order* $O(nd)$*, where* n *is the number of nodes and* d *is the feature dimension. This is negligible compared with the computation cost* $O(md)$ *in feature aggregation, where* m *is the number of edges, due to the fact that usually there are many more edges than nodes in real-world graphs, i.e.,* $m \gg n$*.*

4.3.3 Interpretation of AMP

Interestingly, the proposed AMP has a simple and intuitive interpretation as adaptive residual connection, which aligns well with our design motivation:

- If the feature of node v_i, i.e., $(\mathbf{X}_{\text{in}})_i$, is significantly inconsistent with its local neighbors, i.e., the aggregated feature $\mathbf{Y}_i^k$, then the local deviation $\|\mathbf{Y}_i^k - (\mathbf{X}_{\text{in}})_i\|_2$ will be large, which leads to a β_i close to 1. Therefore, the final step will assign a small weight to the residual, i.e., $(1-\beta_i)(\mathbf{X}_{\text{in}})_i$, and the aggregated feature $\mathbf{Y}_i^k$ will dominate.

- On the contrary, if $(\mathbf{X}_{\text{in}})_i$ is already consistent with its local neighbors, $\|\mathbf{Y}_i^k - (\mathbf{X}_{\text{in}})_i\|_2$ will be small, which leads to a β_i close to 0. Thus, the residual will dominate, which is reasonable since there is less need to aggregate features in this case.

- To summarize, the local deviation $\|\mathbf{Y}_i^k - (\mathbf{X}_{\text{in}})_i\|_2$ provides a natural transition from $\beta_i \to 1$ to $\beta_i \to 0$, and the transition can be modulated by λ which can be either learned or tuned as a hyperparameter through cross-validation. This transition provides an node-wise adaptive residual connection for the message passing scheme.

Adaptivity for abnormal & normal features. According to the homophily assumption on graph structure data [76, 130, 127, 48], the feature representations of normal features should be more consistent with local neighbors than abnormal features. As a result, AMP will assign more residual (i.e., smaller β) to normal features but less residual (i.e., larger β) to abnormal features, providing a customized tradeoff between feature aggregation and residual connection. Consequently, it can promote both the resilience to abnormal features and the performance on normal features. Above discussion also implies a clear physical meaning for β in AMP, and we formally define it as the adaptive score.

Definition 1 (Adaptive score)**.** *The variables $\{\beta_1, \cdots, \beta_n\}$ in the adaptive message passing scheme (AMP) are defined as the adaptive scores for nodes $\{v_1, \cdots, v_n\}$ respectively in graph $\mathcal{G}$. In particular, the larger β_i is, the more likely the feature of node v_i is abnormal.*

Remark 10 (Nonlinear smoother)**.** *Different from most existing message passing scheme which are linear smoothers, AMP is a nonlinear smoother because the weights $\{\beta_i\}$ are computed from $\mathbf{Y}^k$ and $\mathbf{X}_{in}$. This nonlinearity is the key to achieve adaptive residual connection for different nodes.*

4.3.4 Model Architecture

The proposed adaptive message passing (AMP) can be used as a building block in many GNN models to improve the resilience to abnormal node features. In this work, we choose the the decoupled architectures as APPNP [49] and DAGNN [65], and propose the Adaptive residual GNN (AirGNN):

$$\mathbf{X}_{\text{in}} = h_\theta(\mathbf{X}_{\text{fea}}), \tag{4.19}$$

$$\mathbf{Y}_{\text{pre}} = \mathbf{AMP}\ (\mathbf{X}_{\text{in}}, K, \lambda). \tag{4.20}$$

$h_\theta(\cdot)$ is any machine learning model parameterized by learnable parameters θ, such as multilayer perceptrons (MLPs). $\mathbf{X}_{\text{fea}} \in \mathbb{R}^{n\times d}$ denotes the initial node features. The model $h_\theta(\cdot)$ will first transform the initial node features as $\mathbf{X}_{\text{in}} = h_\theta(\mathbf{X}_{\text{fea}})$. AMP takes $h_\theta(\mathbf{X}_{\text{fea}})$ as input, and performs K steps of AMP with the hyperparameter λ. Similar to the majority of existing GNN models, the training objective is the cross-entropy classification loss on the labeled nodes, and the whole model is trained in an end-to-end way. Note that AirGNN is very efficient as explained in Remark 9, and it only requires two hyperparameters K and λ without introducing additional parameters to learn, which could reduce the risk of overfitting.

4.4 Experiment

In this section, we aim to verify the effective of the proposed adaptive message passing scheme (AMP) and the AirGNN model through the semi-supervised node classification tasks. Specifically, we try to answer the following questions: (1) How does AirGNN perform on abnormal and normal features? (Section 4.4.2 and 4.4.3) and (2) How does AirGNN work by adjusting the adaptive residual? (Section 4.4.4)

4.4.1 Experimental Settings

Datasets and baselines. We conduct experiments on 8 real-world datasets including three citation graphs, i.e., Cora, Citeseer, Pubmed [93], two co-authorship graphs, i.e., Coauthor CS and Coauthor Physics [95], two co-purchase graphs, i.e., Amazon Computers and Amazon Photo [95], and one OGB dataset, i.e., ogbn-arxiv [113]. Due to the space limit, we only present the results on Cora, Citeseer, and Pubmed in this section, but defer the results on other datasets to Appendix C.2.1.

In the experiments, the data statistics (full graphs) used in Section 4.4.2 are summarized in Table 4.1. The data statistics (largest connected components) used in Section 4.4.3 are summarized in Table 4.2. We use fixed data splits for Cora, CiteSeer, PubMed and ogbn-arxiv datasets, and random data split for other datasets.

The proposed AirGNN is compared with representative GNNs, including GCN [48], GAT [109],

Table 4.1: Data statistics on benchmark datasets.

Dataset	Classes	Nodes	Edges	Features	Training Nodes	Validation Nodes	Test Nodes
Cora	7	2708	5278	1433	20 per class	500	1000
CiteSeer	6	3327	4552	3703	20 per class	500	1000
PubMed	3	19717	44324	500	20 per class	500	1000
Coauthor CS	15	18333	81894	6805	20 per class	30 per class	Rest nodes
Coauthor Physics	5	34493	247962	8415	20 per class	30 per class	Rest nodes
Amazon Computers	10	13381	245778	767	20 per class	30 per class	Rest nodes
Amazon Photo	8	7487	119043	745	20 per class	30 per class	Rest nodes
obgn-arxiv	40	169343	1166243	128	54%	18%	28%

Table 4.2: Dataset statistics for adversarially attacked datasets.

Dataset	$\mathbf{N_{LCC}}$	$\mathbf{E_{LCC}}$	Classes	Features
Cora	2,485	5,069	7	1,433
CiteSeer	2,110	3,668	6	3,703
PubMed	19,717	44,338	3	500

APPNP [49] and GCNII [14]. We defer the comparison with the variants of APPNP and Robust GCN [128] to Appendix C.2.3 and C.2.4 respectively.

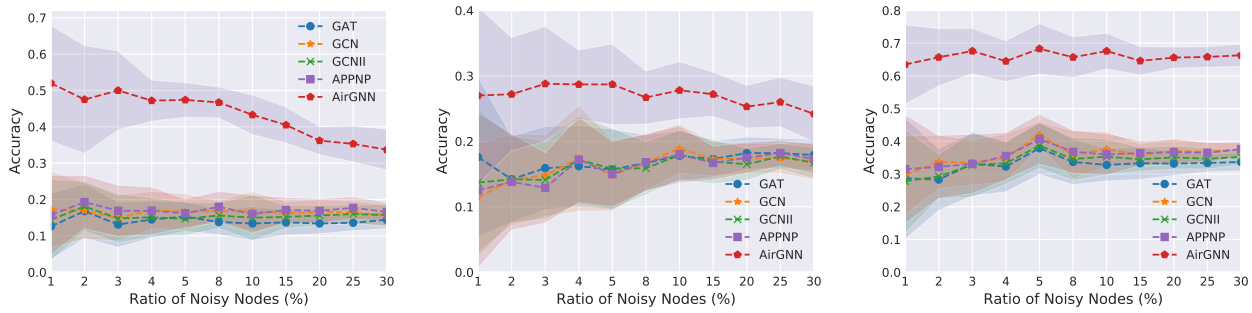
Parameter settings. For all baselines, we follow the best hyperparameter settings in their original papers. Additionally, we tune a best residual weight α for APPNP and GCNII in the range $[0, 1]$. For AirGNN, we use a two-layer MLP as the base model $h_\theta(\cdot)$, following APPNP. We fix the learning rate 0.01, dropout 0.8, and weight decay 0.0005. Moreover, we set $\gamma = \frac{1}{2(1-\lambda)}$ as suggested by Theorem 5. We choose $K = 10$ and tune λ in the range $[0, 1]$. Adam optimizer [47] is used in all experiments. We run all experiments by 10 times, and report the mean and variance.

Evaluation setting. We assess the performance of all models under two types of abnormal feature scenarios, including noisy features and adversarial features. The abnormal features are injected to randomly selected test nodes after model training. By default, all hyperparameters are tuned according to the performance on validation sets when the dataset is clean. If tuning the hyperparameter λ of AirGNN according to the validation sets after injecting abnormal features, the performance will be even better, as discussed in Appendix C.2.2. The performance on clean data are showed in Appendix 4.4.5 to demonstrate that AirGNN doesn't need to sacrifice accuracy for better robustness against abnormal features.

4.4.2 Performance Comparison with Noisy Features

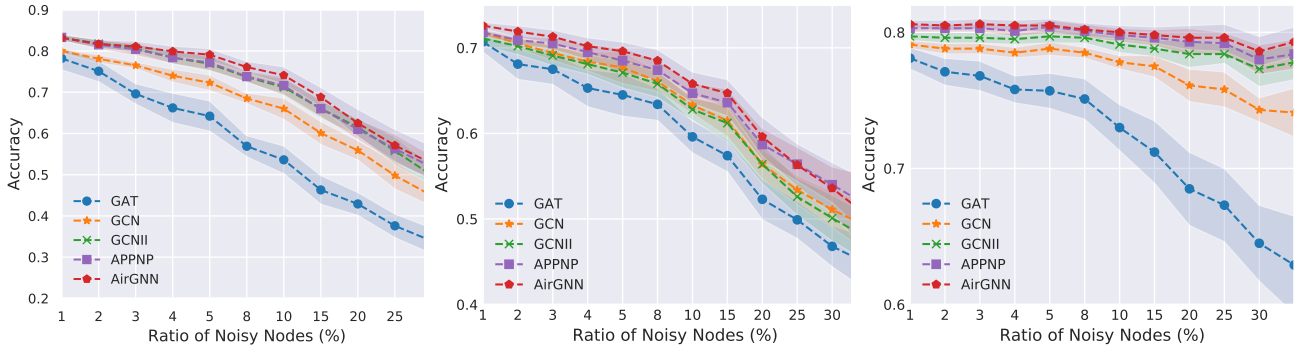
In this subsection, we consider the abnormal features in the noisy feature scenario. Specifically, we simulate the noisy features by assigning a subset of the nodes with random features sampled from a multivariate standard Gaussian distribution. Note that the selection of noise subsets has a apparent impact on the performance since some nodes are less vulnerable to abnormal features while others are more vulnerable. To reduce such variance, we report the average performance over 10 times of random selection of the noise node sets, similar to the settings in the preliminary study in Section 4.2. We report the node classification test accuracy on abnormal (noisy) features and normal features in Figure 4.5 and Figure 4.6, separately, under varying noisy ratio. From these figures, we can observe:

- Figure 4.5 shows that AirGNN significantly outperforms all baselines on all datasets in terms of the performance on noisy nodes. This verifies that AMP is able to improve the resilience to noisy features, aligning well with the design motivation.
- Figure 4.6 shows that AirGNN promotes the performance on normal nodes when abnormal nodes exist. This is because AMP can remove some abnormal features which are detrimental to normal nodes.



(a) Cora (b) CiteSeer (c) PubMed

Figure 4.5: Node classification accuracy on abnormal (noisy) nodes.



(a) Cora (b) CiteSeer (c) PubMed

Figure 4.6: Node classification accuracy on normal nodes.

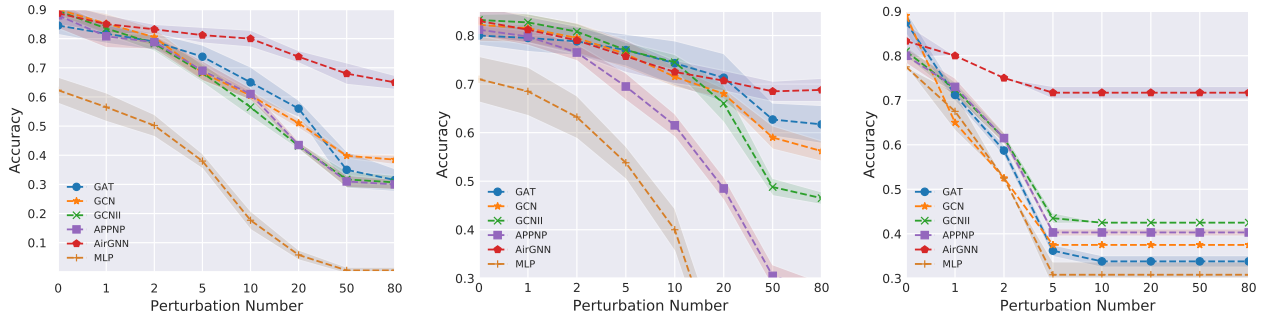
4.4.3 Performance Comparison with Adversarial Features

In this subsection, we consider the abnormal feature scenario when the node features are maliciously attacked by the attacker to manipulate the prediction of GNNs. We use the Nettack [134] implemented in DeepRobust[2] [58], a PyTorch library for adversarial attacks and defenses, to generate the adversarial features. We randomly choose 40 test nodes as the targeted nodes, and assess the performance under increasing perturbation budgets $\{0, 5, 10, 20, 50, 80\}$, where the perturbation numbers denote the number of feature dimensions that can be manipulated. The node classification accuracy on these attacked nodes are showed in Figure 4.7. From these figures, we can make the following observations:

- AirGNN is significantly more robust against adversarially attacked features than all baselines. MLP is the most vulnerable model, which demonstrates the usefulness of graph structure information in combating against abnormal node features.

- The advantages of AirGNN over the baselines become much stronger with larger perturbation budgets. This suggests that AMP can significantly improve the resilience to abnormal features.

[2]https://github.com/DSE-MSU/DeepRobust



(a) Cora (b) CiteSeer (c) PubMed

Figure 4.7: Node classification accuracy on adversarial nodes.

4.4.4 Adaptive Residual for Abnormal & Normal Nodes

To further understand and verify how AMP and AirGNN work, we investigate the adaptive score β_i for each node v_i. Specifically, the average adaptive scores for abnormal nodes and normal nodes in the last layer of AMP are computed separately. In the noisy feature scenario, we fix ratio of noisy nodes as 10%. In the adversarial feature scenario, we choose 40 target nodes and fix the perturbation number as 80. The results in noisy and adversarial feature scenarios are showed in Table 4.3 and Table 4.4, respectively. From these tables, we can observe:

- On the one hand, it can be clearly observed that in both scenarios, the average adaptive scores for abnormal nodes are significantly higher than those for normal nodes. Therefore, it verifies our intuition that large adaptive scores are strongly related to abnormal features.
- On the other hand, it also implies that the residual weights (i.e., $1 - \beta_i$) for abnormal nodes are much lower than those of normal nodes. This perfectly aligns with our motivation to remove abnormal features by reducing their residual connections.

The study on adaptive scores verifies how the adaptive residuals in AMP and AirGNN work as designed. It corroborates that AirGNN not only tremendously boosts the resilience to abnormal features but also provides interpretable information for anomaly detection that will be useful in many security-critical scenarios since the adaptive score serves as a good indicator of abnormal nodes. Morever, it is expected that APPNP without residual will perform well on abnormal nodes but it will

sacrifice the performance on normal nodes. We provide detailed comparison with APPNP w/Res and APPNP wo/Res in Appendix C.2.3 to show the advantages of adaptive residual of AirGNN.

Table 4.3: Average adaptive score (β) and residual weight ($1-\beta$) in the noisy feature scenario.

Measure	**Cora**	**CiteSeer**	**PubMed**
Average adaptive score for abnormal nodes	0.998 ± 0.000	0.988 ± 0.000	0.996 ± 0.000
Average adaptive score for normal nodes	0.924 ± 0.002	0.807 ± 0.005	0.869 ± 0.006
Average residual weight for abnormal nodes	0.002 ± 0.000	0.012 ± 0.000	0.004 ± 0.000
Average residual weight for normal nodes	0.076 ± 0.002	0.193 ± 0.005	0.131 ± 0.006

Table 4.4: Average adaptive score (β) and residual weight ($1-\beta$) in the adversarial feature scenario.

Measure	**Cora**	**CiteSeer**	**PubMed**
Average adaptive score for abnormal nodes	0.987 ± 0.000	0.930 ± 0.007	0.959 ± 0.005
Average adaptive score for normal nodes	0.922 ± 0.004	0.689 ± 0.024	0.826 ± 0.016
Average residual weight for abnormal nodes	0.013 ± 0.000	0.070 ± 0.007	0.041 ± 0.005
Average residual weight for normal nodes	0.078 ± 0.004	0.311 ± 0.024	0.174 ± 0.016

4.4.5 Performance in the Clean Setting

Table 4.5 shows the overall performance when the dataset does not contain abnormal node features. The performance of APPNP and AirGNN are comparable, which supports that AirGNN doesn't need to sacrifice clean performance for better robustness. AirGNN also outperforms Robust GCN in the clean data setting.

Table 4.5: Comparison between AirGNN, APPNP, and Robust GCN in the clean setting.

Dataset	Cora	CiteSeer	PubMed
Robust GCN	0.817 ± 0.005	0.710 ± 0.005	0.791 ± 0.003
APPNP	0.842 ± 0.004	0.719 ± 0.004	0.804 ± 0.003
AirGNN	0.839 ± 0.004	0.726 ± 0.004	0.806 ± 0.003

4.5 Related Work

GNNs generalize convolutional neural networks (CNN) to graph structure data through the message passing framework [74, 29, 91]. The design of message passing and GNN architectures are majorly

motivated in spectral domain [48, 23] and spatial domain [33, 109, 91, 29, 24]. Recent works have shown that the message passing in GNNs can be regarded as low-pass graph filters [84, 126]. More generally, it has been proven that message passing in many GNNs can be uniformly derived from graph signal denoising [73, 86, 129, 16]. Classic GNNs such as GCN [48] and GAT [109] achieve their best performance with shallow models, but their performance degrades when stacking more layers, which can be partially explained through oversmoothing analyses [55, 85]. Recent works propose to use residual connections or skip connections to mitigate the oversmoothing issues, and they demonstrate the potential benefits from more feature aggregations. Examples include but not limited to DeepGCNs [53], JKNet [121], GCNII [14], APPNP [49] and DeeperGNN [65]. These models use global residual connection that can not be adaptive for each node, which significantly differ from the proposed AirGNN. Graph-level, neighborhood-wise and pair-level smoothness are studied in the framework of graph feature gating networks [39]. Beyond oversmoothing, feature over-correlation in GNNs [38] and automated self-supervised learning for graphs [40] are studied.

Recently, there are growing interests in reducing GNNs' vulnerability to the graph structure noise, such as Robust GCN [128], GCN-SVD [27], Pro-GNN [41], IDGL [18], ElasticGNN [67], etc. Please refer to the comprehensive surveys [37, 132] for more details. However, how to design GNNs with strong resilience to abnormal node features remains to be developed. To the best of our knowledge, AirGNN is the first GNN model that is intrinsically robust to many types of abnormal node features by design. It improves the performance in various kinds of abnormal scenarios without needing to sacrifice clean accuracy in normal settings.

4.6 Conclusion

In this work, we discover an intrinsic tension between feature aggregation and residual connection in the message passing scheme of GNNs, as well as the corresponding performance tradeoff between nodes with abnormal and normal features. We analyze possible reasons to explain these findings from the perspective of graph Laplacian smoothing. Our understandings further motivate us to propose a simple, efficient, interpretable and adaptive message passing scheme as well as a new GNN

model with adaptive residual, named AirGNN. AirGNN provides a node-wise adaptive transition between feature aggregation and residual connection, and the significant advantages of AirGNN are demonstrated through extensive experiments. In the future, it is promising to study the interaction between multiple perspectives of trustworthy AI [64] such as robustness and fairness [119].

CHAPTER 5

ELASTIC GRAPH NEURAL NETWORKS

In this chapter, we discuss a severe limitation of the message passing schemes in existing graph neural networks (GNNs) – they are proven to perform ℓ_2-based graph smoothing that enforces smoothness globally and such a global smoothness property might lead to the lack of robustness under adversarial graph attacks. In this work, we propose to design a more robust message passing algorithm for GNNs by enhancing the local smoothness adaptivity of GNNs via ℓ_1-based graph smoothing. To this end, we introduce a family of GNNs (Elastic GNNs) based on ℓ_1 and ℓ_2-based graph smoothing. In particular, we propose a novel and general message passing scheme into GNNs. This message passing algorithm is not only friendly to back-propagation training but also achieves the desired smoothing properties with a theoretical convergence guarantee. Experiments on semi-supervised learning tasks demonstrate that the proposed Elastic GNNs obtain better adaptivity on benchmark datasets and are significantly robust to graph adversarial attacks. The implementation of Elastic GNNs is available at https://github.com/lxiaorui/ElasticGNN.

5.1 Introduction

Graph neural networks (GNNs) generalize traditional deep neural networks (DNNs) from regular grids, such as image, video, and text, to irregular data such as social networks, transportation networks, and biological networks, which are typically denoted as graphs [23, 48]. One popular such generalization is the neural message passing framework [29]:

$$\mathbf{x}_u^{(k+1)} = \mathrm{UPDATE}^{(k)}\big(\mathbf{x}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\big) \tag{5.1}$$

where $\mathbf{x}_u^{(k)} \in \mathbb{R}^d$ denotes the feature vector of node u in k-th iteration of message passing and $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ is the message aggregated from u's neighborhood $\mathcal{N}(u)$. The specific architecture design has been motivated from spectral domain [48, 23] and spatial domain [33, 109, 91, 29]. Recent study [73] has proven that the message passing schemes in numerous popular GNNs, such as GCN, GAT, PPNP,

and APPNP, intrinsically perform the ℓ_2-based graph smoothing to the graph signal, and they can be considered as solving the graph signal denoising problem:

$$\arg\min_{\mathbf{F}} \mathcal{L}(\mathbf{F}) := \|\mathbf{F} - \mathbf{X}_{\text{in}}\|_F^2 + \lambda \operatorname{tr}(\mathbf{F}^\top \mathbf{L}\mathbf{F}), \tag{5.2}$$

where $\mathbf{X}_{\text{in}} \in \mathbb{R}^{n\times d}$ is the input signal and $\mathbf{L} \in \mathbb{R}^{n\times n}$ is the graph Laplacian matrix encoding the graph structure. The first term guides $\mathbf{F}$ to be close to input signal $\mathbf{X}_{\text{in}}$, while the second term enforces global smoothness to the filtered signal $\mathbf{F}$. The resulted message passing schemes can be derived by different optimization solvers, and they typically entail the aggregation of node features from neighboring nodes, which intuitively coincides with the cluster or consistency assumption that neighboring nodes should be similar [130, 127]. While existing GNNs are prominently driven by ℓ_2-based graph smoothing, ℓ_2-based methods enforce smoothness globally and the level of smoothness is usually shared across the whole graph. However, the level of smoothness over different regions of the graph can be different. For instance, node features or labels can change significantly between clusters but smoothly within the cluster [131]. Therefore, it is desired to enhance the local smoothness adaptivity of GNNs.

Motivated by the idea of trend filtering [46, 106, 111], we aim to achieve the goal via ℓ_1-based graph smoothing. Intuitively, compared with ℓ_2-based methods, ℓ_1-based methods penalize large values less and thus preserve discontinuity or non-smooth signal better. Theoretically, ℓ_1-based methods tend to promote signal sparsity to trade for discontinuity [90, 105, 94]. Owning to these advantages, trend filtering [106] and graph trend filter [111, 108] demonstrate that ℓ_1-based graph smoothing can adapt to inhomogenous level of smoothness of signals and yield estimators with k-th order piecewise polynomial functions, such as piecewise constant, linear and quadratic functions, depending on the order of the graph difference operator. While ℓ_1-based methods exhibit various appealing properties and have been extensively studied in different domains such as signal processing [25], statistics and machine learning [34], it has rarely been investigated in the design of GNNs. In this work, we attempt to bridge this gap and enhance the local smoothnesss adaptivity of GNNs via ℓ_1-based graph smoothing.

Incorporating ℓ_1-based graph smoothing in the design of GNNs faces tremendous challenges. First, since the message passing schemes in GNNs can be derived from the optimization iteration of the graph signal denoising problem, a fast, efficient and scalable optimization solver is desired. Unfortunately, to solve the associated optimization problem involving ℓ_1 norm is challenging since the objective function is composed by smooth and non-smooth components and the decision variable is further coupled by the discrete graph difference operator. Second, to integrate the derived messaging passing scheme into GNNs, it has to be composed by simple operations that are friendly to the back-propagation training of the whole GNNs. Third, it requires an appropriate normalization step to deal with diverse node degrees, which is often overlooked by existing graph total variation and graph trend filtering methods. Our attempt to address these challenges leads to a family of novel GNNs, i.e., Elastic GNNs. Our key contributions can be summarized as follows:

- We introduce ℓ_1-based graph smoothing in the design of GNNs to further enhance the local smoothness adaptivity, for the first time;
- We derive a novel and general message passing scheme, i.e., Elastic Message Passing (EMP), and develop a family of GNN architectures, i.e., Elastic GNNs, by integrating the proposed message passing scheme into deep neural nets;
- Extensive experiments demonstrate that Elastic GNNs obtain better adaptivity on various real-world datasets, and they are significantly robust to graph adversarial attacks. The study on different variants of Elastic GNNs suggests that ℓ_1 and ℓ_2-based graph smoothing are complementary and Elastic GNNs are more versatile.

5.2 Preliminary

We use bold upper-case letters such as $\mathbf{X}$ to denote matrices and bold lower-case letters such as $\mathbf{x}$ to define vectors. Given a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, we use $\mathbf{X}_i$ to denote its i-th row and $\mathbf{X}_{ij}$ to denote its element in i-th row and j-th column. We define the Frobenius norm, ℓ_1 norm, and ℓ_{21} norm of matrix $\mathbf{X}$ as $\|\mathbf{X}\|_F = \sqrt{\sum_{ij} \mathbf{X}_{ij}^2}$, $\|\mathbf{X}\|_1 = \sum_{ij} |\mathbf{X}_{ij}|$, and $\|\mathbf{X}\|_{21} = \sum_i \|\mathbf{X}_i\|_2 = \sum_i \sqrt{\sum_j \mathbf{X}_{ij}^2}$, respectively. We

define $\|\mathbf{X}\|_2 = \sigma_{\max}(\mathbf{X})$ where $\sigma_{\max}(\mathbf{X})$ is the largest singular value of $\mathbf{X}$. Given two matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n\times d}$, we define the inner product as $\langle \mathbf{X}, \mathbf{Y}\rangle = \mathrm{tr}(\mathbf{X}^\top \mathbf{Y})$.

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a graph with the node set $\mathcal{V} = \{v_1, \ldots, v_n\}$ and the undirected edge set $\mathcal{E} = \{e_1, \ldots, e_m\}$. We use $\mathcal{N}(v_i)$ to denote the neighboring nodes of node v_i, including v_i itself. Suppose that each node is associated with a d-dimensional feature vector, and the features for all nodes are denoted as $\mathbf{X}_{\text{fea}} \in \mathbb{R}^{n\times d}$. The graph structure $\mathcal{G}$ can be represented as an adjacent matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, where $\mathbf{A}_{ij} = 1$ when there exists an edge between nodes v_i and v_j. The graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal degree matrix. Let $\Delta \in \{-1, 0, 1\}^{m\times n}$ be the oriented incident matrix, which contains one row for each edge. If $e_\ell = (i, j)$, then Δ has ℓ-th row as:

$$\Delta_\ell = (0, \ldots, \underbrace{-1}_{i}, \ldots, \underbrace{1}_{j}, \ldots, 0)$$

where the edge orientation can be arbitrary. Note that the incident matrix and unnormalized Laplacian matrix have the equivalence $\mathbf{L} = \Delta^\top \Delta$. Next, we briefly introduce some necessary background about the graph signal denoising perspective of GNNs and the graph trend filtering methods.

5.2.1 GNNs as Graph Signal Denoising

It is evident from recent work [73] that many popular GNNs can be uniformly understood as graph signal denoising with Laplacian smoothing regularization. Here we briefly describe several representative examples.

GCN. The message passing scheme in Graph Convolutional Networks (GCN) [48],

$$\mathbf{X}_{\text{out}} = \tilde{\mathbf{A}}\mathbf{X}_{\text{in}},$$

is equivalent to one gradient descent step to minimize $\mathrm{tr}(\mathbf{F}^\top(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{F})$ with the initial $\mathbf{F} = \mathbf{X}_{\text{in}}$ and stepsize $1/2$. Here $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$ with $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ being the adjacent matrix with self-loop, whose degree matrix is $\hat{\mathbf{D}}$.

PPNP & APPNP. The message passing scheme in PPNP and APPNP [49] follow the aggregation rules

$$\mathbf{X}_{\text{out}} = \alpha\big(\mathbf{I} - (1-\alpha)\tilde{\mathbf{A}}\big)^{-1}\mathbf{X}_{\text{in}},$$

and

$$\mathbf{X}^{(k+1)} = (1-\alpha)\tilde{\mathbf{A}}\mathbf{X}^{(k)} + \alpha\mathbf{X}_{\text{in}}.$$

They are shown to be the exact solution and one gradient descent step with stepsize $\alpha/2$ for the following problem

$$\min_{\mathbf{F}} \|\mathbf{F} - \mathbf{X}_{\text{in}}\|_F^2 + (1/\alpha - 1)\ \text{tr}(\mathbf{F}^\top(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{F}). \tag{5.3}$$

For more comprehensive illustration, please refer to [73]. We point out that all these message passing schemes adopt ℓ_2-based graph smoothing as the signal differences between neighboring nodes are penalized by the square of ℓ_2 norm, e.g., $\sum_{(v_i,v_j)\in\mathcal{E}} \|\frac{\mathbf{F}_i}{\sqrt{d_i+1}} - \frac{\mathbf{F}_j}{\sqrt{d_j+1}}\|_2^2$ with d_i being the node degree of node v_i. The resulted message passing schemes are usually linear smoothers which smooth the input signal by their linear transformation.

5.2.2 Graph Trend Filtering

In the univariate case, the k-th order graph trend filtering (GTF) estimator [111] is given by

$$\arg\min_{\mathbf{f}\in\mathbb{R}^n} = \frac{1}{2}\|\mathbf{f} - \mathbf{x}\|_2^2 + \lambda\|\Delta^{(k+1)}\mathbf{f}\|_1 \tag{5.4}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the 1-dimensional input signal of n nodes and $\Delta^{(k+1)}$ is a k-th order graph difference operator. When $k = 0$, it penalizes the absolute difference across neighboring nodes in graph $\mathcal{G}$:

$$\|\Delta^{(1)}\mathbf{f}\|_1 = \sum_{(v_i,v_j)\in\mathcal{E}} |\mathbf{f}_i - \mathbf{f}_j|$$

where $\Delta^{(1)}$ is equivalent to the incident matrix Δ. Generally, k-th order graph difference operators can be defined recursively:

$$\Delta^{(k+1)} = \begin{cases} \Delta^\top\Delta^{(k)} = \mathbf{L}^{\frac{k+1}{2}} \in \mathbb{R}^{n\times n} & \text{for odd k} \\ \Delta\Delta^{(k)} = \Delta\mathbf{L}^{\frac{k}{2}} \in \mathbb{R}^{m\times n} & \text{for even k.} \end{cases}$$

It is demonstrated that GTF can adapt to inhomogeneity in the level of smoothness of signal and tends to provide piecewise polynomials over graphs [111]. For instance, when $k = 0$, the sparsity induced by the ℓ_1-based penalty $\|\Delta^{(1)}\mathbf{f}\|_1$ implies that many of the differences $\mathbf{f}_i - \mathbf{f}_j$ are zeros across edges $(v_i, v_j) \in \mathcal{E}$ in $\mathcal{G}$. The piecewise property originates from the discontinuity of signal allowed by less aggressive ℓ_1 penalty, with adaptively chosen knot nodes or knot edges. Note that the smoothers induced by GTF are not linear smoothers and cannot be simply represented by linear transformation of the input signal.

5.3 Algorithm

In this section, we first propose a new graph signal denoising estimator. Then we develop an efficient optimization algorithm for solving the denoising problem and introduce a novel, general and efficient message passing scheme, i.e., Elastic Message Passing (EMP), for graph signal smoothing. Finally, the integration of the proposed message passing scheme and deep neural networks leads to Elastic GNNs.

5.3.1 Elastic Graph Signal Estimator

To combine the advantages of ℓ_1 and ℓ_2-based graph smoothing, we propose the following elastic graph signal estimator:

$$\arg\min_{\mathbf{F}\in\mathbb{R}^{n\times d}} \underbrace{\lambda_1\|\Delta\mathbf{F}\|_1}_{g_1(\Delta\mathbf{F})} + \underbrace{\frac{\lambda_2}{2}\mathrm{tr}(\mathbf{F}^\top\mathbf{L}\mathbf{F}) + \frac{1}{2}\|\mathbf{F}-\mathbf{X}_{\text{in}}\|_F^2}_{f(\mathbf{F})} \tag{5.5}$$

where $\mathbf{X}_{\text{in}} \in \mathbb{R}^{n\times d}$ is the d-dimensional input signal of n nodes. The first term can be written in an edge-centric way: $\|\Delta^{(1)}\mathbf{F}\|_1 = \sum_{(v_i,v_j)\in\mathcal{E}} \|\mathbf{F}_i - \mathbf{F}_j\|_1$, which penalizes the absolute difference across connected nodes in graph $\mathcal{G}$. Similarly, the second term penalizes the difference quadratically via $\mathrm{tr}(\mathbf{F}^\top\mathbf{L}\mathbf{F}) = \sum_{(v_i,v_j)\in\mathcal{E}} \|\mathbf{F}_i - \mathbf{F}_j\|_2^2$. The last term is the fidelity term which preserves the similarity with the input signal. The regularization coefficients λ_1 and λ_2 control the balance between ℓ_1 and ℓ_2-based graph smoothing.

Remark 11. *It is potential to consider higher-order graph differences in both the ℓ_1-based and ℓ_2-based smoothers. But, in this work, we focus on the* 0*-th order graph difference operator* Δ*, since we assume the piecewise constant prior for graph representation learning.*

Normalization. In existing GNNs, it is beneficial to normalize the Laplacian matrix for better numerical stability, and the normalization trick is also crucial for achieving superior performance. Therefore, for the ℓ_2-based graph smoothing, we follow the common normalization trick in GNNs: $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$, where $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$, $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\hat{\mathbf{D}}_{ii} = d_i = \sum_j \hat{\mathbf{A}}_{ij}$. It leads to a degree normalized penalty

$$\operatorname{tr}(\mathbf{F}^\top \tilde{\mathbf{L}}\mathbf{F}) = \sum_{(v_i,v_j)\in\mathcal{E}} \left\| \frac{\mathbf{F}_i}{\sqrt{d_i+1}} - \frac{\mathbf{F}_j}{\sqrt{d_j+1}} \right\|_2^2.$$

In the literature of graph total variation and graph trend filtering, the normalization step is often overlooked and the graph difference operator is directly used as in GTF [111, 108]. To achieve better numerical stability and handle diverse node degrees in real-world graphs, we propose to normalize each column of the incident matrix by the square root of node degrees for the ℓ_1-based graph smoothing as follows[1]:

$$\tilde{\Delta} = \Delta\hat{\mathbf{D}}^{-\frac{1}{2}}.$$

It leads to a degree normalized total variation penalty [2]

$$\|\tilde{\Delta}\mathbf{F}\|_1 = \sum_{(v_i,v_j)\in\mathcal{E}} \left\| \frac{\mathbf{F}_i}{\sqrt{d_i+1}} - \frac{\mathbf{F}_j}{\sqrt{d_j+1}} \right\|_1.$$

Note that this normalized incident matrix maintains the relation with the normalized Laplacian matrix as in the unnormalized case

$$\tilde{\mathbf{L}} = \tilde{\Delta}^\top\tilde{\Delta} \tag{5.6}$$

given that

$$\tilde{\mathbf{L}} = \hat{\mathbf{D}}^{-\frac{1}{2}}(\hat{\mathbf{D}} - \hat{\mathbf{A}})\hat{\mathbf{D}}^{-\frac{1}{2}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{L}\hat{\mathbf{D}}^{-\frac{1}{2}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\Delta^\top\Delta\hat{\mathbf{D}}^{-\frac{1}{2}}.$$

[1] It naturally supports read-value edge weights if the edge weights are set in the incident matrix Δ.

[2] With the normalization, the piecewise constant prior is up to the degree scaling, i.e., sparsity in $\tilde{\Delta}\mathbf{F}$.

With the normalization, the estimator defined in (5.5) becomes:

$$\arg\min_{\mathbf{F}\in\mathbb{R}^{n\times d}} \underbrace{\lambda_1\|\tilde{\Delta}\mathbf{F}\|_1}_{g_1(\tilde{\Delta}\mathbf{F})} + \underbrace{\frac{\lambda_2}{2}\mathrm{tr}(\mathbf{F}^\top\tilde{\mathbf{L}}\mathbf{F}) + \frac{1}{2}\|\mathbf{F}-\mathbf{X}_{\text{in}}\|_F^2}_{f(\mathbf{F})}. \tag{5.7}$$

Capture correlation among dimensions. The node features in real-world graphs are usually multi-dimensional. Although the estimator defined in (5.7) is able to handle multi-dimensional data since the signal from different dimensions are separable under ℓ_1 and ℓ_2 norm, such estimator treats each feature dimension independently and does not exploit the potential relation between feature dimensions. However, the sparsity patterns of node difference across edges could be shared among feature dimensions. To better exploit this potential correlation, we propose to couple the multi-dimensional features by ℓ_{21} norm, which penalizes the summation of ℓ_2 norm of the node difference

$$\|\tilde{\Delta}\mathbf{F}\|_{21} = \sum_{(v_i,v_j)\in\mathcal{E}} \left\| \frac{\mathbf{F}_i}{\sqrt{d_i+1}} - \frac{\mathbf{F}_j}{\sqrt{d_j+1}} \right\|_2.$$

This penalty promotes the row sparsity of $\tilde{\Delta}\mathbf{F}$ and enforces similar sparsity patterns among feature dimensions. In other words, if two nodes are similar, all their feature dimensions should be similar. Therefore, we define the ℓ_{21}-based estimator as

$$\arg\min_{\mathbf{F}\in\mathbb{R}^{n\times d}} \underbrace{\lambda_1\|\tilde{\Delta}\mathbf{F}\|_{21}}_{g_{21}(\tilde{\Delta}\mathbf{F})} + \underbrace{\frac{\lambda_2}{2}\mathrm{tr}(\mathbf{F}^\top\tilde{\mathbf{L}}\mathbf{F}) + \frac{1}{2}\|\mathbf{F}-\mathbf{X}_{\text{in}}\|_F^2}_{f(\mathbf{F})} \tag{5.8}$$

where $g_{21}(\cdot) = \lambda_1\|\cdot\|_{21}$. In the following subsections, we will use $g(\cdot)$ to represent both $g_1(\cdot)$ and $g_{21}(\cdot)$. We use ℓ_1 to represent both ℓ_1 and ℓ_{21} if not specified.

5.3.2 Elastic Message Passing

For the ℓ_2-based graph smoother, message passing schemes can be derived from the gradient descent iterations of the graph signal denoising problem, as in the case of GCN and APPNP [73]. However, computing the estimators defined by (5.7) and (5.8) is much more challenging because of the nonsmoothness, and the two components, i.e., $f(\mathbf{F})$ and $g(\tilde{\Delta}\mathbf{F})$, are non-separable as they are coupled by the graph difference operator $\tilde{\Delta}$. In the literature, researchers have developed optimization

algorithms for the graph trend filtering problem (5.4) such as Alternating Direction Method of Multipliers (ADMM) and Newton type algorithms [111, 108]. However, these algorithms require to solve the minimization of a non-trivial sub-problem in each single iteration, which incurs high computation complexity. Moreover, it is unclear how to make these iterations compatible with the back-propagation training of deep learning models. This motivates us to design an algorithm which is not only efficient but also friendly to back-propagation training. To this end, we propose to solve an equivalent saddle point problem using a primal-dual algorithm with efficient computations.

Saddle point reformulation. For a general convex function $g(\cdot)$, its conjugate function is defined as

$$g^*(\mathbf{Z}) := \sup_{\mathbf{X}} \langle \mathbf{Z}, \mathbf{X} \rangle - g(\mathbf{X}).$$

By using $g(\tilde{\Delta}\mathbf{F}) = \sup_{\mathbf{Z}} \langle \tilde{\Delta}\mathbf{F}, \mathbf{Z} \rangle - g^*(\mathbf{Z})$, the problem (5.7) and (5.8) can be equivalently written as the following saddle point problem:

$$\min_{\mathbf{F}} \max_{\mathbf{Z}} f(\mathbf{F}) + \langle \tilde{\Delta}\mathbf{F}, \mathbf{Z} \rangle - g^*(\mathbf{Z}). \tag{5.9}$$

where $\mathbf{Z} \in \mathbb{R}^{m \times d}$. Motivated by Proximal Alternating Predictor-Corrector (PAPC) [70, 15], we propose an efficient algorithm with per iteration low computation complexity and convergence guarantee:

$$\bar{\mathbf{F}}^{k+1} = \mathbf{F}^k - \gamma \nabla f(\mathbf{F}^k) - \gamma \tilde{\Delta}^\top \mathbf{Z}^k, \tag{5.10}$$

$$\mathbf{Z}^{k+1} = \mathbf{prox}_{\beta g^*}(\mathbf{Z}^k + \beta \tilde{\Delta} \bar{\mathbf{F}}^{k+1}), \tag{5.11}$$

$$\mathbf{F}^{k+1} = \mathbf{F}^k - \gamma \nabla f(\mathbf{F}^k) - \gamma \tilde{\Delta}^\top \mathbf{Z}^{k+1}, \tag{5.12}$$

where $\mathbf{prox}_{\beta g^*}(\mathbf{X}) = \arg\min_{\mathbf{Y}} \frac{1}{2}\|\mathbf{Y} - \mathbf{X}\|_F^2 + \beta g^*(\mathbf{Y})$. The stepsizes, γ and β, will be specified later. The first step (5.10) obtains a prediction of $\mathbf{F}^{k+1}$, i.e., $\bar{\mathbf{F}}^{k+1}$, by a gradient descent step on primal variable $\mathbf{F}^k$. The second step (5.11) is a proximal dual ascent step on the dual variable $\mathbf{Z}^k$ based on the predicted $\bar{\mathbf{F}}^{k+1}$. Finally, another gradient descent step on the primal variable based on $(\mathbf{F}^k, \mathbf{Z}^{k+1})$ gives next iteration $\mathbf{F}^{k+1}$ (5.12). Algorithm (5.10)–(5.12) can be interpreted as a "predict-correct" algorithm for the saddle point problem (5.9). Next we demonstrate how to compute the proximal operator in Eq. (5.11).

Proximal operators. Using the Moreau's decomposition principle [6]

$$\mathbf{X} = \mathbf{prox}_{\beta g^*}(\mathbf{X}) + \beta \mathbf{prox}_{\beta^{-1} g}(\mathbf{X}/\beta),$$

we can rewrite the step (5.11) using the proximal operator of $g(\cdot)$, that is,

$$\mathbf{prox}_{\beta g^*}(\mathbf{X}) = \mathbf{X} - \beta \mathbf{prox}_{\frac{1}{\beta} g}(\frac{1}{\beta}\mathbf{X}). \tag{5.13}$$

$$\begin{aligned}
&\mathbf{Y}^{k+1} = \gamma \mathbf{X}_{\text{in}} + (1-\gamma)\tilde{\mathbf{A}}\mathbf{F}^k \\
&\bar{\mathbf{F}}^{k+1} = \mathbf{Y}^k - \gamma \tilde{\Delta}^\top \mathbf{Z}^k \\
&\bar{\mathbf{Z}}^{k+1} = \mathbf{Z}^k + \beta \tilde{\Delta} \bar{\mathbf{F}}^{k+1} \\
&\begin{cases}
\mathbf{Z}^{k+1} = \min(|\bar{\mathbf{Z}}^{k+1}|, \lambda_1) \cdot \text{sign}(\bar{\mathbf{Z}}^{k+1}) & \textbf{(Option I: } \ell_1 \text{ norm)} \\
\mathbf{Z}_i^{k+1} = \min(\|\bar{\mathbf{Z}}_i^{k+1}\|_2, \lambda_1) \cdot \frac{\bar{\mathbf{Z}}_i^{k+1}}{\|\bar{\mathbf{Z}}_i^{k+1}\|_2}, \forall i \in [m] & \textbf{(Option II: } \ell_{21} \text{ norm)}
\end{cases} \\
&\mathbf{F}^{k+1} = \mathbf{Y}^k - \gamma \tilde{\Delta}^\top \mathbf{Z}^{k+1}
\end{aligned}$$

Figure 5.1: Elastic Message Passing (EMP). $\mathbf{F}^0 = \mathbf{X}_{\text{in}}$ and $\mathbf{Z}^0 = \mathbf{0}^{m \times d}$.

We discuss the two options for the function $g(\cdot)$ corresponding to the objectives (5.7) and (5.8).

- **Option I** (ℓ_1 norm): $g_1(\mathbf{X}) = \lambda_1 \|\mathbf{X}\|_1$

  By definition, the proximal operator of $\frac{1}{\beta} g_1(\mathbf{X})$ is

  $$\mathbf{prox}_{\frac{1}{\beta} g_1}(\mathbf{X}) = \arg\min_{\mathbf{Y}} \frac{1}{2}\|\mathbf{Y} - \mathbf{X}\|_F^2 + \frac{1}{\beta}\lambda_1 \|\mathbf{Y}\|_1,$$

  which is equivalent to the soft-thresholding operator (component-wise):

  $$\begin{aligned}
  (S_{\frac{1}{\beta}\lambda_1}(\mathbf{X}))_{ij} =& \text{sign}(\mathbf{X}_{ij}) \max(|\mathbf{X}_{ij}| - \frac{1}{\beta}\lambda_1, 0) \\
  =& \mathbf{X}_{ij} - \text{sign}(\mathbf{X}_{ij}) \min(|\mathbf{X}_{ij}|, \frac{1}{\beta}\lambda_1).
  \end{aligned}$$

  Therefore, using (5.13), we have

  $$(\mathbf{prox}_{\beta g_1^*}(\mathbf{X}))_{ij} = \text{sign}(\mathbf{X}_{ij}) \min(|\mathbf{X}_{ij}|, \lambda_1). \tag{5.14}$$

  which is a *component-wise projection* onto the ℓ_∞ ball of radius λ_1.

- **Option II** (ℓ_{21} norm): $g_{21}(\mathbf{X}) = \lambda_1 \|\mathbf{X}\|_{21}$

  By definition, the proximal operator of $\frac{1}{\beta} g_{21}(\mathbf{X})$ is

$$\mathbf{prox}_{\frac{1}{\beta} g_{21}}(\mathbf{X}) = \arg\min_{\mathbf{Y}} \frac{1}{2}\|\mathbf{Y} - \mathbf{X}\|_F^2 + \frac{1}{\beta}\lambda_1 \|\mathbf{Y}\|_{21}$$

  with the i-th row being

$$\left(\mathbf{prox}_{\frac{1}{\beta} g_{21}}(\mathbf{X})\right)_i = \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_2} \max(\|\mathbf{X}_i\|_2 - \frac{1}{\beta}\lambda_1, 0).$$

  Similarly, using (5.13), we have the i-th row of $\mathbf{prox}_{\beta g_{21}^*}(\mathbf{X})$ being

$$\begin{aligned}
&(\mathbf{prox}_{\beta g_{21}^*}(\mathbf{X}))_i \\
&= \mathbf{X}_i - \beta \mathbf{prox}_{\frac{1}{\beta} g_{21}}(\mathbf{X}_i/\beta) \\
&= \mathbf{X}_i - \beta \frac{\mathbf{X}_i/\beta}{\|\mathbf{X}_i/\beta\|_2} \max(\|\mathbf{X}_i/\beta\|_2 - \lambda_1/\beta, 0) \\
&= \mathbf{X}_i - \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_2} \max(\|\mathbf{X}_i\|_2 - \lambda_1, 0) \\
&= \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_2} (\|\mathbf{X}_i\|_2 - \max(\|\mathbf{X}_i\|_2 - \lambda_1, 0)) \\
&= \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_2} \min(\|\mathbf{X}_i\|_2, \lambda_1),
\end{aligned} \tag{5.15}$$

  which is a *row-wise projection* on the ℓ_2 ball of radius λ_1. Note that the proximal operator in the ℓ_1 norm case treats each feature dimension independently, while in the ℓ_{21} norm case, it couples the multi-dimensional features, which is consistent with the motivation to exploit the correlation among feature dimensions.

The Algorithm (5.10)–(5.12) and the proximal operators (5.14) and (5.15) enable us to derive the final message passing scheme. Note that the computation $\mathbf{F}^k - \gamma \nabla f(\mathbf{F}^k)$ in steps (5.10) and (5.12) can be shared to save computation. Therefore, we decompose the step (5.10) into two steps:

$$\begin{aligned}
\mathbf{Y}^k &= \mathbf{F}^k - \gamma \nabla f(\mathbf{F}^k) \\
&= \left((1-\gamma)\mathbf{I} - \gamma\lambda_2 \tilde{\mathbf{L}}\right)\mathbf{F}^k + \gamma \mathbf{X}_{\text{in}}, \qquad (5.16) \\
\bar{\mathbf{F}}^{k+1} &= \mathbf{Y}^k - \gamma \tilde{\Delta}^\top \mathbf{Z}^k. \qquad (5.17)
\end{aligned}$$

In this work, we choose $\gamma = \frac{1}{1+\lambda_2}$ and $\beta = \frac{1}{2\gamma}$. Therefore, with $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$, Eq. (5.16) can be simplified as

$$\mathbf{Y}^{k+1} = \gamma \mathbf{X}_{\text{in}} + (1-\gamma)\tilde{\mathbf{A}}\mathbf{F}^k. \tag{5.18}$$

Let $\bar{\mathbf{Z}}^{k+1} := \mathbf{Z}^k + \beta\tilde{\Delta}\bar{\mathbf{F}}^{k+1}$, then steps (5.11) and (5.12) become

$$\mathbf{Z}^{k+1} = \mathbf{prox}_{\beta g^*}(\bar{\mathbf{Z}}^{k+1}), \tag{5.19}$$

$$\begin{aligned}\mathbf{F}^{k+1} &= \mathbf{F}^k - \gamma\nabla f(\mathbf{F}^k) - \gamma\tilde{\Delta}\mathbf{Z}^{k+1} \\ &= \mathbf{Y}^k - \gamma\tilde{\Delta}^\top\mathbf{Z}^{k+1}.\end{aligned} \tag{5.20}$$

Substituting the proximal operators in (5.19) with (5.14) and (5.15), we obtain the complete elastic message passing scheme (EMP) as summarized in Figure 5.1.

Interpretation of EMP. EMP can be interpreted as the standard message passing (MP) ($\mathbf{Y}$ in Fig. 1) with extra operations (the following steps). The extra operations compute $\tilde{\Delta}^\top\mathbf{Z}$ to adjust the standard MP such that sparsity in $\tilde{\Delta}\mathbf{F}$ is promoted and some large node differences can be preserved. EMP is general and covers some existing propagation rules as special cases as demonstrated in Remark 12.

Remark 12 (Special cases)**.** *If there is only ℓ_2-based regularization, i.e., $\lambda_1 = 0$, then according to the projection operator, we have $\mathbf{Z}^k = \mathbf{0}^{m\times n}$. Therefore, with $\gamma = \frac{1}{1+\lambda_2}$, the proposed message passing scheme reduces to*

$$\mathbf{F}^{k+1} = \frac{1}{1+\lambda_2}\mathbf{X}_{in} + \frac{\lambda_2}{1+\lambda_2}\tilde{\mathbf{A}}\mathbf{F}^k.$$

If $\lambda_2 = \frac{1}{\alpha} - 1$, it recovers the message passing in APPNP:

$$\mathbf{F}^{k+1} = \alpha\mathbf{X}_{in} + (1-\alpha)\tilde{\mathbf{A}}\mathbf{F}^k.$$

If $\lambda_2 = \infty$, it recovers the simple aggregation operation in many GNNs:

$$\mathbf{F}^{k+1} = \tilde{\mathbf{A}}\mathbf{F}^k.$$

Computation Complexity. EMP is efficient and composed by simple operations. The major computation cost comes from four sparse matrix multiplications, include $\tilde{\mathbf{A}}\mathbf{F}^k, \tilde{\Delta}^\top \mathbf{Z}^k, \tilde{\Delta}\bar{\mathbf{F}}^{k+1}$ and $\tilde{\Delta}^\top \mathbf{Z}^{k+1}$. The computation complexity is in the order $O(md)$ where m is the number of edges in graph $\mathcal{G}$ and d is the feature dimension of input signal $\mathbf{X}_{\text{in}}$. Other operations are simple matrix additions and projection.

The convergence of EMP and the parameter settings are justified by Theorem 6.

Theorem 6 (Convergence of EMP)**.** *Under the stepsize setting* $\gamma < \frac{2}{1+\lambda_2\|\tilde{\mathbf{L}}\|_2}$ *and* $\beta \leq \frac{4}{3\gamma\|\tilde{\Delta}\tilde{\Delta}^\top\|_2}$*, the elastic message passing scheme (EMP) in Figure 5.1 converges to the optimal solution of the elastic graph signal estimator defined in* (5.7) *(Option I) or* (5.8) *(Option II). It is sufficient to choose any* $\gamma < \frac{2}{1+2\lambda_2}$ *and* $\beta \leq \frac{2}{3\gamma}$ *since* $\|\tilde{\mathbf{L}}\|_2 = \|\tilde{\Delta}^\top\tilde{\Delta}\|_2 = \|\tilde{\Delta}\tilde{\Delta}^\top\|_2 \leq 2$.

Proof. We first consider the general problem

$$\min_{\mathbf{F}} f(\mathbf{F}) + g(\mathbf{B}\mathbf{F}) \tag{5.21}$$

where f and g are convex functions and $\mathbf{B}$ is a bounded linear operator. It is proved in [70, 15] that the iterations in (5.10)–(5.12) guarantee the convergence of $\mathbf{F}^k$ to the optimal solution of the minimization problem (5.21) if the parameters satisfy $\gamma < \frac{2}{L}$ and $\beta \leq \frac{1}{\gamma\lambda_{\max}(\mathbf{B}\mathbf{B}^\top)}$, where L is the Lipschitz constant of $\nabla f(\mathbf{F})$. These conditions are further relaxed to $\gamma < \frac{2}{L}$ and $\beta \leq \frac{4}{3\gamma\lambda_{\max}(\mathbf{B}\mathbf{B}^\top)}$ in [60].

For the specific problems defined in (5.7) and (5.8), the two function components f and g are both convex, and the linear operator Δ is bounded. The Lipschitz constant of $\nabla f(\mathbf{F})$ can be computed by the largest eigenvalue of the Hessian matrix of $f(\mathbf{F})$:

$$L = \lambda_{\max}(\nabla^2 f(\mathbf{F})) = \lambda_{\max}(\mathbf{I} + \lambda_2\tilde{\mathbf{L}}) = 1 + \lambda_2\|\tilde{\mathbf{L}}\|_2.$$

Therefore, the elastic message passing scheme derived from iterations (5.10)–(5.12) is guaranteed to converge to the optimal solution of problem (5.7) (Option I) or problem (5.8) (Option II) if the stepsizes satisfy $\gamma < \frac{2}{1+\lambda_2\|\tilde{\mathbf{L}}\|_2}$ and $\beta \leq \frac{4}{3\gamma\|\tilde{\Delta}\tilde{\Delta}^\top\|_2}$.

Let $\tilde{\Delta} = \mathbf{U}\Sigma\mathbf{V}^\top$ be the singular value decomposition of $\tilde{\Delta}$ and we derive

$$\|\tilde{\Delta}\tilde{\Delta}^\top\|_2 = \|\mathbf{U}\Sigma\mathbf{V}^\top\mathbf{V}\Sigma\mathbf{U}^\top\|_2 = \|\mathbf{U}\Sigma^2\mathbf{U}^\top\|_2 = \|\mathbf{V}\Sigma^2\mathbf{V}^\top\|_2 = \|\mathbf{V}\Sigma\mathbf{U}^\top\mathbf{U}\Sigma\mathbf{V}^\top\|_2 = \|\tilde{\Delta}^\top\tilde{\Delta}\|_2.$$

The equivalence $\tilde{\mathbf{L}} = \tilde{\Delta}^\top\tilde{\Delta}$ in (5.6) further gives

$$\|\tilde{\mathbf{L}}\|_2 = \|\tilde{\Delta}^\top\tilde{\Delta}\|_2 = \|\tilde{\Delta}\tilde{\Delta}^\top\|_2.$$

Since $\|\tilde{\mathbf{L}}\|_2 \leq 2$ [19], we have $\frac{2}{1+2\lambda_2} \leq \frac{2}{1+\lambda_2\|\tilde{\mathbf{L}}\|_2}$ and $\frac{2}{3\gamma} \leq \frac{4}{3\gamma\|\tilde{\Delta}\tilde{\Delta}^\top\|_2}$. Therefore, $\gamma < \frac{2}{1+2\lambda_2}$ $\beta \leq \frac{2}{3\gamma}$ are sufficient for the convergence of EMP.

□

5.3.3 Elastic GNNs

Incorporating the elastic message passing scheme from the elastic graph signal estimator (5.7) and (5.8) into deep neural networks, we introduce a family of GNNs, namely Elastic GNNs. In this work, we follow the decoupled way as proposed in APPNP [49], where we first make predictions from node features and aggregate the prediction through the proposed EMP:

$$\mathbf{Y}_{\text{pre}} = \mathbf{EMP}\left(h_\theta(\mathbf{X}_{\text{fea}}), K, \lambda_1, \lambda_2\right). \tag{5.22}$$

$\mathbf{X}_{\text{fea}} \in \mathbb{R}^{n\times d}$ denotes the node features, $h_\theta(\cdot)$ is any machine learning model, such as multilayer perceptrons (MLPs), θ is the learnable parameters in the model, and K is the number of message passing steps. The training objective is the cross entropy loss defined by the final prediction $\mathbf{Y}_{\text{pre}}$ and labels for training data. Elastic GNNs also have the following nice properties:

- In addition to the backbone neural network model, Elastic GNNs only require to set up three hyperparameters including two coefficients λ_1, λ_2 and the propagation step K, but they do not introduce any learnable parameters. Therefore, it reduces the risk of overfitting.
- The hyperparameters λ_1 and λ_2 provide better smoothness adaptivity to Elastic GNNs depending on the smoothness properties of the graph data.
- The message passing scheme only entails simple and efficient operations, which makes it friendly to the efficient and end-to-end back-propagation training of the whole GNN model.

5.4 Experiment

In this section, we conduct experiments to validate the effectiveness of the proposed Elastic GNNs. We first introduce the experimental settings. Then we assess the performance of Elastic GNNs and investigate the benefits of introducing ℓ_1-based graph smoothing into GNNs with semi-supervised learning tasks under normal and adversarial settings. In the ablation study, we validate the local adaptive smoothness, sparsity pattern, and convergence of EMP.

5.4.1 Experimental Settings

Datasets. We conduct experiments on 8 real-world datasets including three citation graphs, i.e., Cora, Citeseer, Pubmed [93], two co-authorship graphs, i.e., Coauthor CS and Coauthor Physics [95], two co-purchase graphs, i.e., Amazon Computers and Amazon Photo [95], and one blog graph, i.e., Polblogs [2]. In Polblogs graph, node features are not available so we set the feature matrix to be a $n \times n$ identity matrix. The data statistics for the benchmark datasets used in Section 5.4.2 are summarized in Table 5.1. The data statistics for the adversarially attacked graph used in Section 5.4.3 are summarized in Table 5.2.

Table 5.1: Statistics of benchmark datasets.

Dataset	Classes	Nodes	Edges	Features	Training Nodes	Validation Nodes	Test Nodes
Cora	7	2708	5278	1433	20 per class	500	1000
CiteSeer	6	3327	4552	3703	20 per class	500	1000
PubMed	3	19717	44324	500	20 per class	500	1000
Coauthor CS	15	18333	81894	6805	20 per class	30 per class	Rest nodes
Coauthor Physics	5	34493	247962	8415	20 per class	30 per class	Rest nodes
Amazon Computers	10	13381	245778	767	20 per class	30 per class	Rest nodes
Amazon Photo	8	7487	119043	745	20 per class	30 per class	Rest nodes

Table 5.2: Dataset Statistics for adversarially attacked graph.

	$\mathbf{N_{LCC}}$	$\mathbf{E_{LCC}}$	Classes	Features
Cora	2,485	5,069	7	1,433
CiteSeer	2,110	3,668	6	3,703
Polblogs	1,222	16,714	2	/
PubMed	19,717	44,338	3	500

Baselines. We compare the proposed Elastic GNNs with representative GNNs including GCN [48], GAT [109], ChebNet [23], GraphSAGE [33], APPNP [49] and SGC [115]. For all models, we use 2 layer neural networks with 64 hidden units.

Parameter settings. For each experiment, we report the average performance and the standard variance of 10 runs. For all methods, hyperparameters are tuned from the following search space: 1) learning rate: $\{0.05, 0.01, 0.005\}$; 2) weight decay: {5e-4, 5e-5, 5e-6}; 3) dropout rate: {0.5, 0.8}. For APPNP, the propagation step K is tuned from $\{5, 10\}$ and the parameter α is tuned from $\{0, 0.1, 0.2, 0.3, 0.5, 0.8, 1.0\}$. For Elastic GNNs, the propagation step K is tuned from $\{5, 10\}$ and parameters λ_1 and λ_2 are tuned from $\{0, 3, 6, 9\}$. As suggested by Theorem 1, we set $\gamma = \frac{1}{1+\lambda_2}$ and $\beta = \frac{1}{2\gamma}$ in the proposed elastic message passing scheme. Adam optimizer [47] is used in all experiments.

5.4.2 Performance on Benchmark Datasets

On commonly used datasets including Cora, CiteSeer, PubMed, Coauthor CS, Coauthor Physics, Amazon Computers and Amazon Photo, we compare the performance of the proposed Elastic GNN ($\ell_{21} + \ell_2$) with representative GNN baselines on the semi-supervised learning task. The classification accuracy are showed in Table 5.3. From these results, we can make the following observations:

- Elastic GNN outperforms GCN, GAT, ChebNet, GraphSAGE and SGC by significant margins on all datasets. For instance, Elastic GNN improves over GCN by 3.1%, 2.0% and 1.8% on Cora, CiteSeer and PubMed datasets. The improvement comes from the global and local smoothness adaptivity of Elastic GNN.
- Elastic GNN ($\ell_{21} + \ell_2$) consistently achieves higher performance than APPNP on all datasets. Essentially, Elastic GNN covers APPNP as a special case when there is only ℓ_2 regularization, i.e., $\lambda_1 = 0$. Beyond the ℓ_2-based graph smoothing, the ℓ_{21}-based graph smoothing further enhances the local smoothness adaptivity. This comparison verifies the benefits of introducing ℓ_{21}-based graph smoothing in GNNs.

Table 5.3: Classification accuracy (%) on benchmark datasets with 10 times random data splits.

Model	**Cora**	**CiteSeer**	**PubMed**	**CS**	**Physics**	**Computers**	**Photo**
ChebNet	76.3 ± 1.5	67.4 ± 1.5	75.0 ± 2.0	91.8 ± 0.4	OOM	**81.0 ± 2.0**	90.4 ± 1.0
GCN	79.6 ± 1.1	68.9 ± 1.2	77.6 ± 2.3	91.6 ± 0.6	93.3 ± 0.8	79.8 ± 1.6	90.3 ± 1.2
GAT	80.1 ± 1.2	68.9 ± 1.8	77.6 ± 2.2	91.1 ± 0.5	93.3 ± 0.7	79.3 ± 2.4	89.6 ± 1.6
SGC	80.2 ± 1.5	68.9 ± 1.3	75.5 ± 2.9	90.1 ± 1.3	93.1 ± 0.6	73.0 ± 2.0	83.5 ± 2.9
APPNP	82.2 ± 1.3	70.4 ± 1.2	78.9 ± 2.2	**92.5 ± 0.3**	93.7 ± 0.7	80.1 ± 2.1	90.8 ± 1.3
GraphSAGE	79.0 ± 1.1	67.5 ± 2.0	77.6 ± 2.0	91.7 ± 0.5	92.5 ± 0.8	80.7 ± 1.7	90.9 ± 1.0
ElasticGNN	**82.7 ± 1.0**	**70.9 ± 1.4**	**79.4 ± 1.8**	**92.5 ± 0.3**	**94.2 ± 0.5**	80.7 ± 1.8	**91.3 ± 1.3**

Table 5.4: Classification accuracy (%) under different perturbation rates of adversarial graph attack.

Dataset	Ptb Rate	Basic GNN		Elastic GNN				
		GCN	GAT	ℓ_2	ℓ_1	ℓ_{21}	$\ell_1 + \ell_2$	$\ell_{21} + \ell_2$
Cora	0%	83.5±0.4	84.0±0.7	**85.8±0.4**	85.1±0.5	85.3±0.4	**85.8±0.4**	**85.8±0.4**
	5%	76.6±0.8	80.4±0.7	81.0±1.0	**82.3±1.1**	81.6±1.1	81.9±1.4	82.2±0.9
	10%	70.4±1.3	75.6±0.6	76.3±1.5	76.2±1.4	77.9±0.9	78.2±1.6	**78.8±1.7**
	15%	65.1±0.7	69.8±1.3	72.2±0.9	73.3±1.3	75.7±1.2	76.9±0.9	**77.2±1.6**
	20%	60.0±2.7	59.9±0.6	67.7±0.7	63.7±0.9	70.3±1.1	67.2±5.3	**70.5±1.3**
Citeseer	0%	72.0±0.6	73.3±0.8	73.6±0.9	73.2±0.6	73.2±0.5	73.6±0.6	**73.8±0.6**
	5%	70.9±0.6	72.9±0.8	72.8±0.5	72.8±0.5	72.8±0.5	**73.3±0.6**	72.9±0.5
	10%	67.6±0.9	70.6±0.5	70.2±0.6	70.8±0.6	70.7±1.2	72.4±0.9	**72.6±0.4**
	15%	64.5±1.1	69.0±1.1	70.2±0.6	68.1±1.4	68.2±1.1	71.3±1.5	**71.9±0.7**
	20%	62.0±3.5	61.0±1.5	**64.9±1.0**	64.7±0.8	64.7±0.8	64.7±0.8	64.7±0.8
Polblogs	0%	95.7±0.4	95.4±0.2	95.4±0.2	**95.8±0.3**	**95.8±0.3**	**95.8±0.3**	**95.8±0.3**
	5%	73.1±0.8	83.7±1.5	82.8±0.3	78.7±0.6	78.7±0.7	82.8±0.4	**83.0±0.3**
	10%	70.7±1.1	76.3±0.9	73.7±0.3	75.2±0.4	75.3±0.7	81.5±0.2	**81.6±0.3**
	15%	65.0±1.9	68.8±1.1	68.9±0.9	72.1±0.9	71.5±1.1	77.8±0.9	**78.7±0.5**
	20%	51.3±1.2	51.5±1.6	65.5±0.7	68.1±0.6	68.7±0.7	77.4±0.2	**77.5±0.2**
Pubmed	0%	87.2±0.1	83.7±0.4	**88.1±0.1**	86.7±0.1	87.3±0.1	**88.1±0.1**	**88.1±0.1**
	5%	83.1±0.1	78.0±0.4	**87.1±0.2**	86.2±0.1	87.0±0.1	**87.1±0.2**	**87.1±0.2**
	10%	81.2±0.1	74.9±0.4	86.6±0.1	86.0±0.2	86.9±0.2	86.3±0.1	**87.0±0.1**
	15%	78.7±0.1	71.1±0.5	85.7±0.2	85.4±0.2	86.4±0.2	85.5±0.1	**86.4±0.2**
	20%	77.4±0.2	68.2±1.0	85.8±0.1	85.4±0.1	86.4±0.1	85.4±0.1	**86.4±0.1**

5.4.3 Robustness Under Adversarial Attack

Locally adaptive smoothness makes Elastic GNNs more robust to adversarial attack on graph structure. This is because the attack tends to connect nodes with different labels, which fuzzes the cluster structure in the graph. But EMP can tolerate large node differences along these wrong edges,

and maintain the smoothness along correct edges.

To validate this, we evaluate the performance of Elastic GNNs under untargeted adversarial graph attack, which tries to degrade GNN models' overall performance by deliberately modifying the graph structure. We use the MetaAttack [135] implemented in DeepRobust [58][3], a PyTorch library for adversarial attacks and defenses, to generate the adversarially attacked graphs based on four datasets including Cora, CiteSeer, Polblogs and PubMed. We randomly split 10%/10%/80% of nodes for training, validation and test. Note that following the works [134, 135, 27, 41], we only consider the largest connected component (LCC) in the adversarial graphs. Therefore, the results in Table 5.4 are not directly comparable with the results in Table 5.3. We focus on investigating the robustness introduced by ℓ_1-based graph smoothing but not on adversarial defense so we don't compare with defense strategies. Existing defense strategies can be applied on Elastic GNNs to further improve the robustness against attacks.

Variants of Elastic GNNs. To make a deeper investigation of Elastic GNNs, we consider the following variants: (1) ℓ_2 ($\lambda_1 = 0$); (2) ℓ_1 ($\lambda_2 = 0$, Option I); (3) ℓ_{21} ($\lambda_2 = 0$, Option II); (4) $\ell_1 + \ell_2$ (Option I); (5) $\ell_{21} + \ell_2$ (Option II). To save computation, we fix the learning rate as 0.01, weight decay as 0.0005, dropout rate as 0.5 and $K = 10$ since this setting works well for the chosen datasets and models. Only λ_1 and λ_2 are tuned. The classification accuracy under different perturbation rates ranging from 0% to 20% is summarized in Table 5.4. From the results, we can make the following observations:

- All variants of Elastic GNNs outperforms GCN and GAT by significant margins under all perturbation rates. For instance, when the pertubation rate is 15%, Elastic GNN ($\ell_{21} + \ell_2$) improves over GCN by 12.1%, 7.4%, 13.7% and 7.7% on the four datasets being considered. This is because Elastic GNN can adapt to the change of smoothness while GCN and GAT can not adapt well when the perturbation rate increases.
- ℓ_{21} outperforms ℓ_1 in most cases, and $\ell_{21} + \ell_2$ outperforms $\ell_1 + \ell_2$ in almost all cases. It demonstrates the benefits of exploiting the correlation between feature channels by coupling

[3] https://github.com/DSE-MSU/DeepRobust

multi-dimensional features via ℓ_{21} norm.

- ℓ_{21} outperforms ℓ_2 in most cases, which suggests the benefits of local smoothness adaptivity. When ℓ_{21} and ℓ_2 is combined, the Elastic GNN (ℓ_{21} + ℓ_2) achieves significantly better performance than solely ℓ_2, ℓ_{21} or ℓ_1 variant in almost all cases. It suggests that ℓ_1 and ℓ_2-based graph smoothing are complementary to each other, and combining them provides significant better robustness against adversarial graph attacks.

5.4.4 Ablation Study

We provide ablation study to further investigate the adaptive smoothness, sparsity pattern, and convergence of EMP in Elastic GNN, based on three datasets including Cora, CiteSeer and PubMed. In this section, we fix $\lambda_1 = 3, \lambda_2 = 3$ for Elastic GNN, and $\alpha = 0.1$ for APPNP. We fix learning rate as 0.01, weight decay as 0.0005 and dropout rate as 0.5 since this setting works well for both methods.

Adaptive smoothness. It is expected that ℓ_1-based smoothing enhances local smoothness adaptivity by increasing the smoothness along correct edges (connecting nodes with same labels) while lowering smoothness along wrong edges (connecting nodes with different labels). To validate this, we compute the average adjacent node differences (based on node features in the last layer) along wrong and correct edges separately, and use the ratio between these two averages to measure the smoothness adaptivity. The results are summarized in Table 5.5. It is clearly observed that for all datasets, the ratio for ElasticGNN is significantly higher than ℓ_2 based method such as APPNP, which validates its better local smoothness adaptivity.

Sparsity pattern. To validate the piecewise constant property enforced by EMP, we also investigate the sparsity pattern in the adjacent node differences, i.e., $\tilde{\Delta}\mathbf{F}$, based on node features in the last layer. Node difference along edge e_i is defined as sparse if $\|(\tilde{\Delta}\mathbf{F})_i\|_2 < 0.1$. The sparsity ratios for ℓ_2-based method such as APPNP and ℓ_1-based method such as Elastic GNN are summarized in Table 5.6. It can be observed that in Elastic GNN, a significant portion of $\tilde{\Delta}\mathbf{F}$ are sparse for all datasets. While in APPNP, this portion is much smaller. This sparsity pattern validates the

piecewise constant prior as designed.

Table 5.5: Ratio between average node differences along wrong and correct edges.

Model	Cora	CiteSeer	PubMed
ℓ_2 (APPNP)	1.57	1.35	1.43
ℓ_{21}+ℓ_2 (ElasticGNN)	2.03	1.94	1.79

Table 5.6: Sparsity ratio (i.e., $\|(\tilde{\Delta}\mathbf{F})_i\|_2 < 0.1$) in node differences $\tilde{\Delta}\mathbf{F}$.

Model	Cora	CiteSeer	PubMed
ℓ_2 (APPNP)	2%	16%	11%
ℓ_{21}+ℓ_2 (ElasticGNN)	37%	74%	42%

Convergence of EMP. We provide two additional experiments to demonstrate the impact of propagation step K on classification performance and the convergence of message passing scheme. Figure 5.2 shows that the increase of classification accuracy when the propagation step K increases. It verifies the effectiveness of EMP in improving graph representation learning. It also shows that a small number of propagation step can achieve very good performance, and therefore the computation cost for EMP can be small. Figure 5.3 shows the decreasing of the objective value defined in Eq. (5.8) during the forward message passing process, and it verifies the convergence of the proposed EMP as suggested by Theorem 6.

5.5 Related Work

The design of GNN architectures can be majorly motivated in spectral domain [48, 23] and spatial domain [33, 109, 91, 29]. The message passing scheme [29, 74] for feature aggregation is one central component of GNNs. Recent works have proven that the message passing in GNNs can be regarded as low-pass graph filters [84, 126]. Generally, it is recently proved that message passing in many GNNs can be unified in the graph signal denosing framework [73, 86, 129, 16]. We point out that they intrinsically perform ℓ_2-based graph smoothing and typically can be represented as linear smoothers.

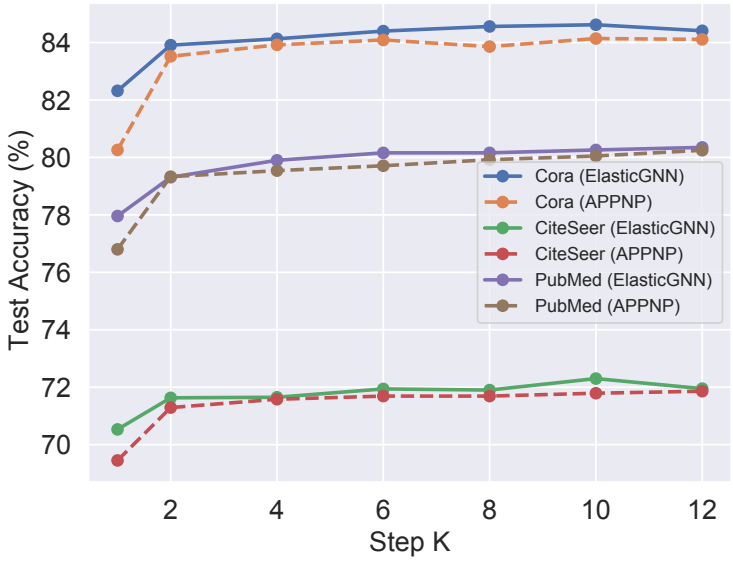


Figure 5.2: Classification accuracy under different propagation steps.

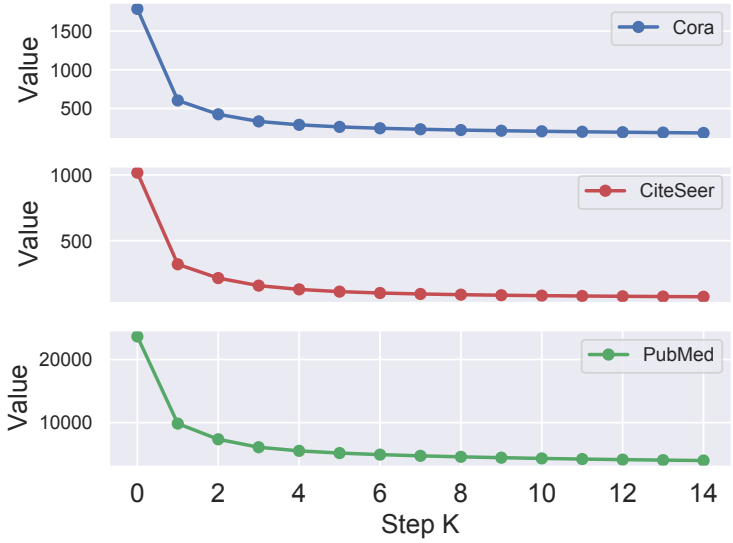


Figure 5.3: Convergence of the objective value for the problem in Eq. (5.8) during message passing.

ℓ_1-based graph signal denoising has been explored in graph trend filtering [111, 108] which tends to provide estimators with k-th order piecewise polynomials over graphs. Graph total variation has also been utilized in semi-supervised learning [83, 44, 43, 5], spectral clustering [12, 11] and graph cut problems [100, 10]. However, it is unclear whether these algorithms can be used to design GNNs. To the best of our knowledge, we make first such investigation in this work.

5.6 Conclusion

In this work, we propose to enhance the smoothness adaptivity of GNNs via ℓ_1 and ℓ_2-based graph smoothing. Through the proposed elastic graph signal estimator, we derive a novel, efficient and general message passing scheme, i.e., elastic message passing (EMP). Integrating the proposed message passing scheme and deep neural networks leads to a family of GNNs, i.e., Elastic GNNs. Extensitve experiments on benchmark datasets and adversarially attacked graphs demonstrate the benefits (e.g., intrinsic robustness) of introducing ℓ_1-based graph smoothing in the design of GNNs. The empirical study suggests that ℓ_1 and ℓ_2-based graph smoothing is complementary to each other, and the proposed Elastic GNNs has better smoothnesss adaptivity owning to the integration of ℓ_1 and ℓ_2-based graph smoothing. We hope the proposed elastic message passing scheme can inspire more powerful GNN architecture design and more general smoothness assumptions such as low homophily [72] can be made in the future. In addition, we also demonstrate the significant advantages of elastic message passing (EMP) in capturing reliable user-item interactions in noisy recommendation systems through the proposed framework of Graph Trend Filtering Networks [28].

CHAPTER 6

CONCLUSION

In this chapter, we summarize the research results in this dissertation and their broader impact, and discuss promising research directions.

6.1 Summary

In this dissertation, we proposed fours solutions to solve the efficiency and security challenges in machine learning - (1) centralized distributed optimization algorithm with bidirectional communication compression; (2) decentralized distributed optimization algorithm with communication compression; (3) graph neural networks with adaptive message passing that is robust to adversarial features; (4) graph neural networks with elastic message passing that is robust to adversarial graph structures.

To fundamentally improve the efficiency of distributed ML systems, I proposed a series of innovative algorithms to break through the communication bottleneck. In particular, when the communication network is a start network, I proposed DORE [68], a double residual compression algorithm, to compress the bi-directional communication between client devices and the server such that over 95% of the communication bits can be reduced. This is the first algorithm that reduces that much communication cost while maintaining the superior convergence complexities (e.g., linear convergence) as the uncompressed counterpart, both theoretically and numerically.

When the communication network is of any general topology (as long as it is connected), I proposed LEAD [69], the first linear convergent decentralized optimization algorithm with communication compression, which only requires point-to-point compressed communication between neighboring devices over communication networks. Theoretically, we prove that under certain compression ratios, the convergence complexity of the proposed algorithm does not depend on the compression operator. In other words, it achieves better communication efficiency for free.

These algorithms significantly improve the efficiency and scalability of large-scale ML systems

with solid theoretical guarantees and remarkable empirical performance. They have the great potential to accelerate scientific discovery through machine learning and data science.

To design intrinsically secure ML models against feature attacks, I investigate to denoise the hidden features in neural network layers caused by the adversarial perturbation using the graph structural information. This is achieved by the proposed AirGNN [66] in which the adaptive message passing denoises perturbed features by feature aggregations and maintains feature separability by adaptive residuals. The proposed algorithm has a clear design principle and interpretation as well strong as performance both in the clean and adversarial data settings. This points out a promising direction of achieving adversarial robustness through feature denoising in hidden layers.

To design intrinsically secure ML models against graph structure attacks, I investigate a new prior knowledge of smoothness in the design of graph neural networks. In particular, we derive an elastic message passing scheme to model the piecewise constant signal in graph data. We demonstrate its stronger resilience to adversarial structure attacks and superior performance when the data is clean through a comprehensive empirical study on the proposed model ElasticGNN [67].

These secure ML models immensely boost the security of ML models under potential adversarial threats. They might not only be applied in safety-critical applications but also inspire further research in this emerging direction.

6.2 Future Direction

Large-scale ML and secure ML are active areas of exploration. Below we discuss some promising research directions:

- **Distributed machine learning under heterogeneous environments:** Our study in centralized learning and decentralized learning suggests that the convergence properties of distributed optimization algorithms might be sensitive to heterogeneous environments - (1) data heterogeneity; (2) network heterogeneity; (3) computation heterogeneity. Due to data heterogeneity, the data distributions from multiple computation devices might significantly differ from each other, which causes direction conflict for model updating if synchronization is not timely.

Due to network heterogeneity, the network bandwidths and conditions are often uneven in large distributed systems. Therefore, it is critical to take the potential communication decay into consideration. Due to computation heterogeneity, different computation devices might have diverse computation power. To fully utilize the computation power, it is vital to design algorithms that support flexible computation tasks and avoid idle time.

- **Secure machine learning for more data types.** Our study in designing secure graph neural networks suggests a new promising direction for designing intrinsically robust ML models through feature denoising in hidden layers of deep neural networks. Therefore, it is promising to generalize these ideas to general data types such as images, videos, and text where the graph structure information is not explicitly available but can be constructed from the data. Moreover, it is also promising to consider more advanced and flexible smoothing assumptions beyond homophily graphs in the design of GNN models.

APPENDICES

APPENDIX A

A DOUBLE RESIDUAL COMPRESSION ALGORITHM FOR DISTRIBUTED LEARNING

A.1 Additional Experiments

A.1.1 Communication Efficiency

To make an explicit comparison of communication efficiency, we report the training loss convergence with respect to communication bits in Figure A.1, A.2 and A.3 for the experiments on synthetic data, MNIST and CIFAR10 dataset respectively. These results are independent of the system architectures and network bandwidth. It suggests that the proposed DORE reduce the communication cost significantly while maintaining good convergence speed.

Furthermore, we also test the running time of ResNet18 trained on CIFAR10 dataset under two different network bandwidth configurations, i.e. 1Gbps and 200Mbps, as showed in Figure A.4 and A.5. Due to its superior communication efficiency, the proposed DORE runs faster in both configurations. Moreover, when the network bandwidth reduces from 1Gbps to 200Mbps, the running time of DORE only increases slightly, which indicates that DORE is more robust to network bandwidth change and can work more efficiently under limited bandwidth. These results clearly suggest the advantages of the proposed algorithm.

All the experiments in this section are under the exactly same setting as described in Section 2.5. The running time is tested in a High Performance Computing Cluster with NVIDIA Tesla K80 GPUs and the computing nodes are connected by Gigabit Ethernet interfaces and we use mpi4py as the communication backend. All algorithms in this work are implemented with PyTorch.

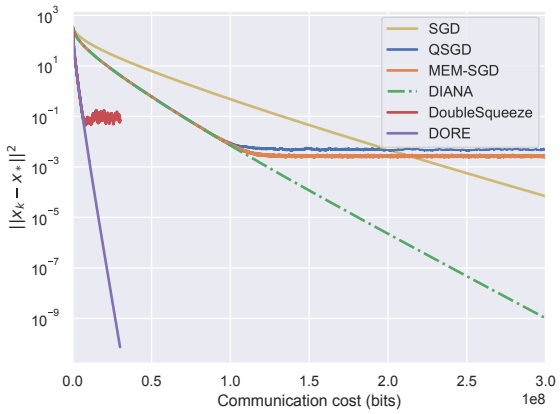


Figure A.1: Linear regression on synthetic data.

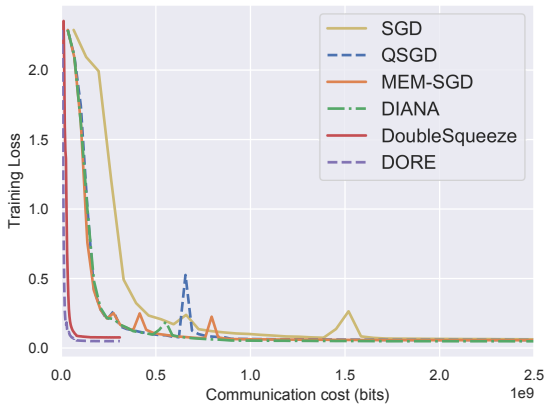


Figure A.2: LeNet trained on MNIST dataset.

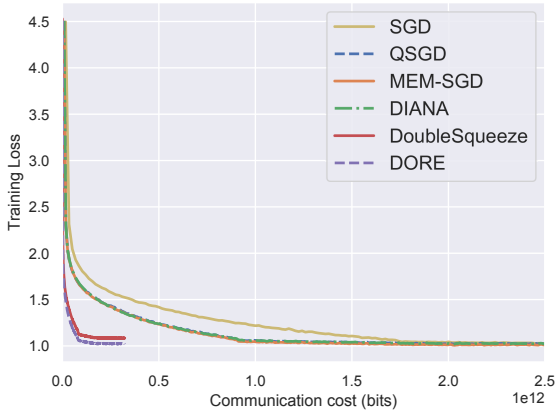


Figure A.3: Resnet18 trained on CIFAR10 dataset.

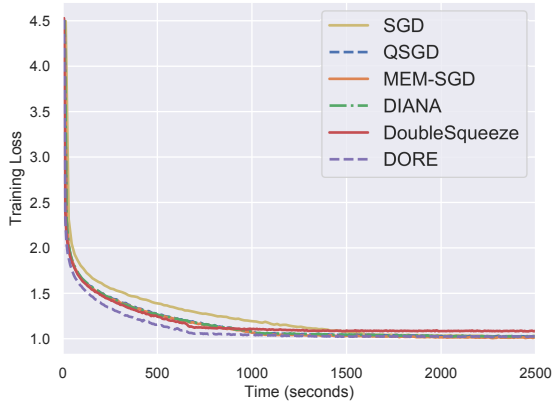


Figure A.4: Resnet18 trained on CIFAR10 dataset with 1Gbps network bandwidth.

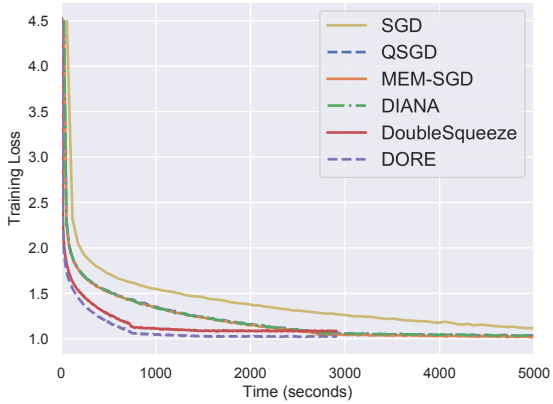
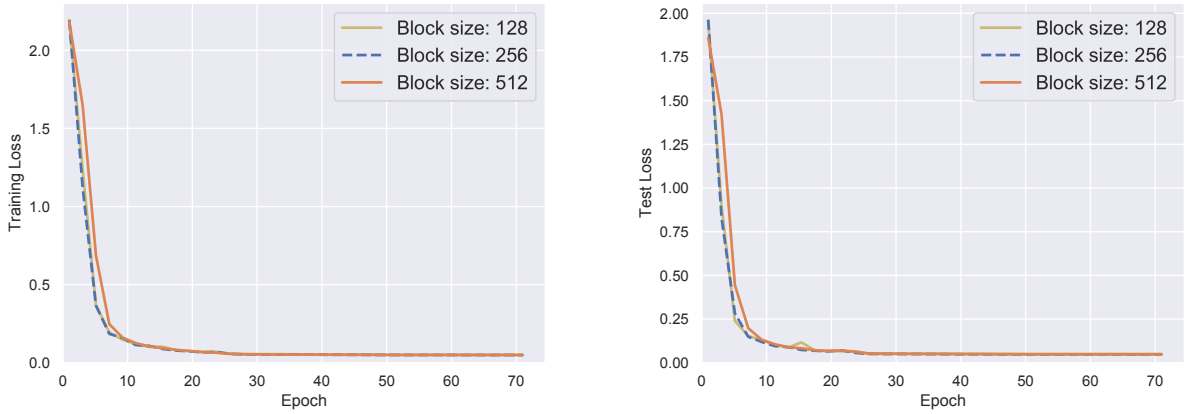


Figure A.5: Resnet18 trained on CIFAR10 dataset with 200Mbps network bandwidth.

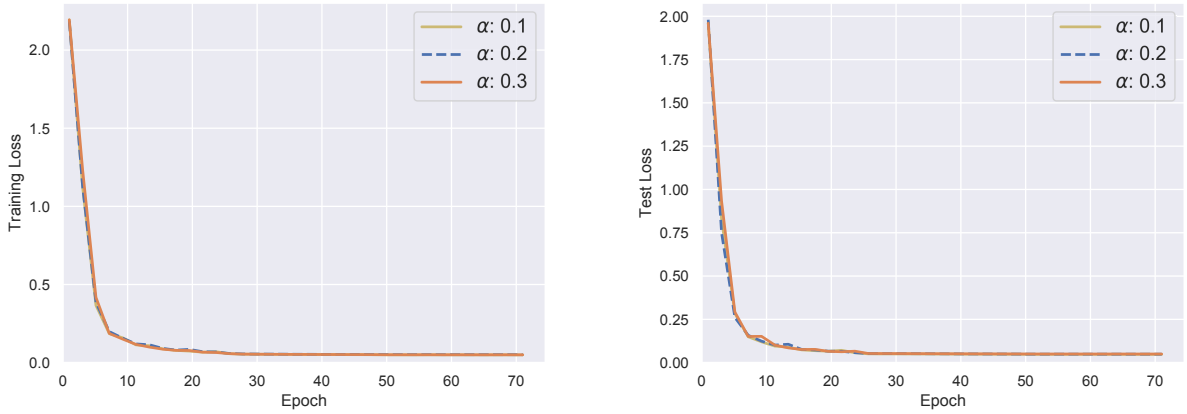
A.1.2 Parameter sensitivity

Continuing the MNIST experiment in Section 2.5, we further conduct parameter analysis on DORE. The basic setting for block size, learning rate, α, β and η are 256, 0.1, 0.1, 1, 1, respectively. We change each parameter individually. Figures A.6, A.7, A.8, and A.9 demonstrate that DORE performs consistently well under different parameter settings.



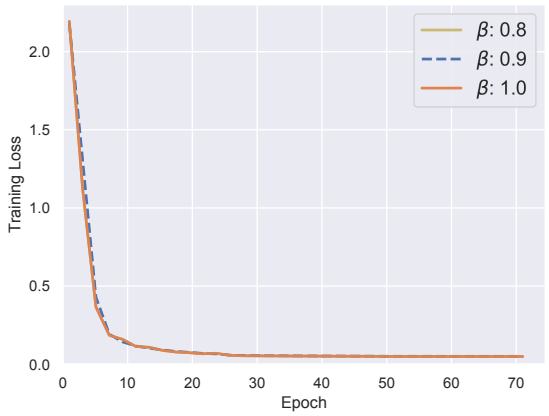
(a) Training loss (b) Test loss

Figure A.6: Training under different compression block sizes.

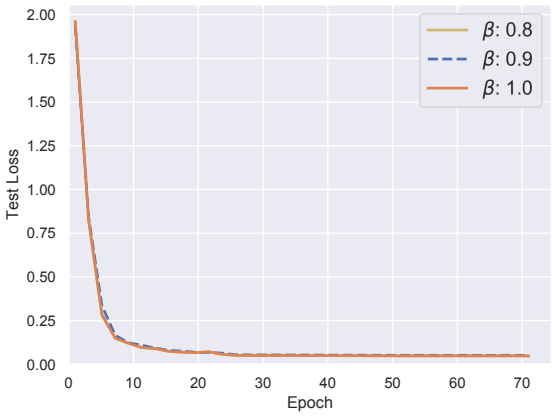


(a) Training loss (b) Test loss

Figure A.7: Training under different α.

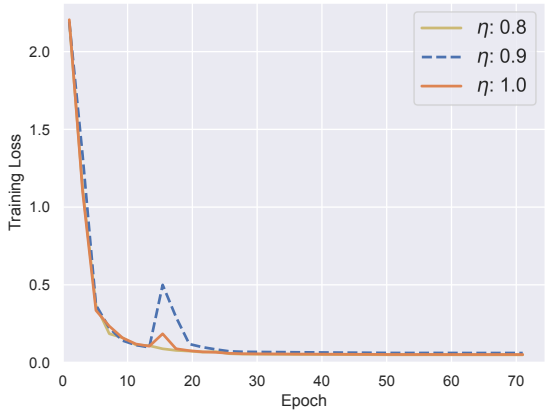


(a) Training loss

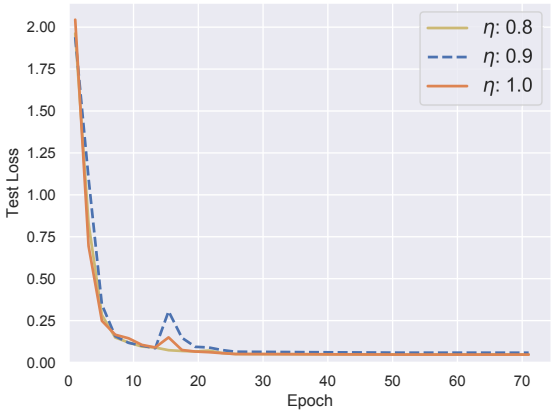


(b) Test loss

Figure A.8: Training under different β.



(a) Training loss



(b) Test loss

Figure A.9: Training under different η.

A.2 Proofs of the theorems

A.2.1 Proof of Theorem 1

We first provide two lemmas. We define $\mathbb{E}_Q$, $\mathbb{E}_k$, and $\mathbb{E}$ be the expectation taken over the quantization, the kth iteration based on $\hat{\mathbf{x}}^k$, and the overall expectation, respectively.

Lemma 7. *For every i, we can estimate the first two moments of $\mathbf{h}_i^{k+1}$ as*

$$\mathbb{E}_Q \mathbf{h}_i^{k+1} = (1-\alpha)\mathbf{h}_i^k + \alpha \mathbf{g}_i^k, \tag{A.1}$$

$$\mathbb{E}_Q\|\mathbf{h}_i^{k+1} - \mathbf{s}_i\|^2 \leq (1-\alpha)\|\mathbf{h}_i^k - \mathbf{s}_i\|^2 + \alpha\|\mathbf{g}_i^k - \mathbf{s}_i\|^2 + \alpha[(C_q+1)\alpha - 1]\|\Delta_i^k\|^2. \tag{A.2}$$

Proof. The first equality follows from lines 5-7 of Algorithm 1 and Assumption 1. For the second equation, we have the following variance decomposition

$$\mathbb{E}\|X\|^2 = \|\mathbb{E}X\|^2 + \mathbb{E}\|X - \mathbb{E}X\|^2 \tag{A.3}$$

for any random vector X. By taking $X = \mathbf{h}_i^{k+1} - \mathbf{s}_i$, we get

$$\mathbb{E}_Q\|\mathbf{h}_i^{k+1} - \mathbf{s}_i\|^2 = \|(1-\alpha)(\mathbf{h}_i^k - \mathbf{s}_i) + \alpha(\mathbf{g}_i^k - \mathbf{s}_i)\|^2 + \alpha^2\mathbb{E}_Q\|\hat{\Delta}_i^k - \Delta_i^k\|^2. \tag{A.4}$$

Using the basic equality

$$\|\lambda\mathbf{a} + (1-\lambda)\mathbf{b}\|^2 + \lambda(1-\lambda)\|\mathbf{a} - \mathbf{b}\|^2 = \lambda\|\mathbf{a}\|^2 + (1-\lambda)\|\mathbf{b}\|^2 \tag{A.5}$$

for all $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ and $\lambda \in [0, 1]$, as well as Assumption 1, we have

$$\mathbb{E}_Q\|\mathbf{h}_i^{k+1} - \mathbf{s}_i\|^2 \leq (1-\alpha)\|\mathbf{h}_i^k - \mathbf{s}_i\|^2 + \alpha\|\mathbf{g}_i^k - \mathbf{s}_i\|^2 - \alpha(1-\alpha)\|\Delta_i^k\|^2 + \alpha^2 C_q\|\Delta_i^k\|^2, \tag{A.6}$$

which is the inequality (A.2). □

Next, from the variance decomposition (A.3), we also derive Lemma 8.

Lemma 8. *The following inequality holds*

$$\mathbb{E}[\|\hat{\mathbf{g}}^k - \mathbf{h}^*\|^2] \leq \mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k) - \mathbf{h}^*\|^2 + \frac{C_q}{n^2}\sum_{i=1}^n \mathbb{E}\|\Delta_i^k\|^2 + \frac{\sigma^2}{n}, \tag{A.7}$$

where $\mathbf{h}^* = \nabla f(\mathbf{x}^*) = \frac{1}{n}\sum_{i=1}^n \mathbf{h}_i^*$ *and* $\sigma^2 = \frac{1}{n}\sum_{i=1}^n \sigma_i^2$.

Proof. By taking the expectation over the quantization of $\mathbf{g}$, we have

$$\begin{aligned}\mathbb{E}\|\hat{\mathbf{g}}^k - \mathbf{h}^*\|^2 &= \mathbb{E}\|\mathbf{g}^k - \mathbf{h}^*\|^2 + \mathbb{E}\|\hat{\mathbf{g}}^k - \mathbf{g}^k\|^2 \\ &\leq \mathbb{E}\|\mathbf{g}^k - \mathbf{h}^*\|^2 + \frac{C_q}{n^2}\sum_{i=1}^n \mathbb{E}\|\Delta_i^k\|^2,\end{aligned} \tag{A.8}$$

where the inequality is from Assumption 1.

For $\|\mathbf{g}^k - \mathbf{h}^*\|$, we take the expectation over the sampling of gradients and derive

$$\begin{aligned}\mathbb{E}\|\mathbf{g}^k - \mathbf{h}^*\|^2 &= \mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k) - \mathbf{h}^*\|^2 + \mathbb{E}\|\mathbf{g}^k - \nabla f(\hat{\mathbf{x}}^k)\|^2 \\ &\leq \mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k) - \mathbf{h}^*\|^2 + \frac{\sigma^2}{n} \end{aligned} \tag{A.9}$$

by Assumption 2.

Combining (A.8) with (A.9) gives (A.7). □

Proof of Theorem 1. We consider $\mathbf{x}^{k+1} - \mathbf{x}^*$ first. Since $\mathbf{x}^*$ is the solution of (2.1), it satisfies

$$\mathbf{x}^* = \mathbf{prox}_{\gamma R}(\mathbf{x}^* - \gamma \mathbf{h}^*). \tag{A.10}$$

Hence

$$\begin{aligned}\mathbb{E}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 =& \mathbb{E}\|\mathbf{prox}_{\gamma R}(\hat{\mathbf{x}}^k - \gamma \hat{\mathbf{g}}^k) - \mathbf{prox}_{\gamma R}(\mathbf{x}^* - \gamma \mathbf{h}^*)\|^2 \\ \leq& \mathbb{E}\|\hat{\mathbf{x}}^k - \mathbf{x}^* - \gamma(\hat{\mathbf{g}}^k - \mathbf{h}^*)\|^2 \\ =& \mathbb{E}\|\hat{\mathbf{x}}^k - \mathbf{x}^*\|^2 - 2\gamma\mathbb{E}\langle \hat{\mathbf{x}}^k - \mathbf{x}^*, \hat{\mathbf{g}}^k - \mathbf{h}^*\rangle + \gamma^2\mathbb{E}\|\hat{\mathbf{g}}^k - \mathbf{h}^*\|^2 \\ =& \mathbb{E}\|\hat{\mathbf{x}}^k - \mathbf{x}^*\|^2 - 2\gamma\mathbb{E}\langle \hat{\mathbf{x}}^k - \mathbf{x}^*, \nabla f(\hat{\mathbf{x}}^k) - \mathbf{h}^*\rangle + \gamma^2\mathbb{E}\|\hat{\mathbf{g}}^k - \mathbf{h}^*\|^2, \end{aligned} \tag{A.11}$$

where the inequality comes from the non-expansiveness of the proximal operator and the last equality is derived by taking the expectation of the stochastic gradient $\hat{\mathbf{g}}^k$. Combining (A.7) and (A.11), we have

$$\begin{aligned}\mathbb{E}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \leq& \mathbb{E}\|\hat{\mathbf{x}}^k - \mathbf{x}^*\|^2 - 2\gamma\mathbb{E}\langle \hat{\mathbf{x}}^k - \mathbf{x}^*, \nabla f(\hat{\mathbf{x}}^k) - \mathbf{h}^*\rangle \\ &+ \frac{\gamma^2}{n}\sum_{i=1}^n \mathbb{E}\|\nabla f_i(\hat{\mathbf{x}}^k) - \mathbf{h}_i^*\|^2 + \frac{C_q\gamma^2}{n^2}\sum_{i=1}^n \mathbb{E}\|\Delta_i^k\|^2 + \frac{\gamma^2}{n}\sigma^2. \end{aligned} \tag{A.12}$$

Then we consider $\mathbb{E}\|\hat{\mathbf{x}}^{k+1} - \mathbf{x}^*\|^2$. According to Algorithm 1, we have:

$$\begin{aligned}\mathbb{E}_Q[\hat{\mathbf{x}}^{k+1} - \mathbf{x}^*] &= \hat{\mathbf{x}}^k + \beta\mathbf{q}^k - \mathbf{x}^* \\ &= (1-\beta)(\hat{\mathbf{x}}^k - \mathbf{x}^*) + \beta(\mathbf{x}^{k+1} - \mathbf{x}^* + \eta\mathbf{e}^k) \end{aligned} \tag{A.13}$$

where the expectation is taken on the quantization of $\mathbf{q}^k$.

By variance decomposition (A.3) and the basic equality (A.5),

$$\begin{aligned}
&\mathbb{E}\|\hat{\mathbf{x}}^{k+1}-\mathbf{x}^*\|^2\\
\leq&(1-\beta)\mathbb{E}\|\hat{\mathbf{x}}^k-\mathbf{x}^*\|^2+\beta\mathbb{E}\|\mathbf{x}^{k+1}+\eta\mathbf{e}^k-\mathbf{x}^*\|^2-\beta(1-\beta)\mathbb{E}\|\mathbf{q}^k\|^2+\beta^2C_q^m\mathbb{E}\|\mathbf{q}^k\|^2\\
\leq&(1-\beta)\mathbb{E}\|\hat{\mathbf{x}}^k-\mathbf{x}^*\|^2+(1+\eta^2\epsilon)\beta\mathbb{E}\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2-\beta(1-(C_q^m+1)\beta)\mathbb{E}\|\mathbf{q}^k\|^2\\
&+(\eta^2+\frac{1}{\epsilon})\beta C_q^m\mathbb{E}\|\mathbf{q}^{k-1}\|^2,
\end{aligned}\tag{A.14}$$

where ϵ is generated from Cauchy inequality of inner product. For convenience, we let $\epsilon=\frac{1}{\eta}$.

Choose a β such that $0<\beta\leq\frac{1}{1+C_q^m}$. Then we have

$$\begin{aligned}
&\beta(1-(C_q^m+1)\beta)\mathbb{E}\|\mathbf{q}^k\|^2+\mathbb{E}\|\hat{\mathbf{x}}^{k+1}-\mathbf{x}^*\|^2\\
\leq&(1-\beta)\mathbb{E}\|\hat{\mathbf{x}}^k-\mathbf{x}^*\|^2+(1+\eta)\beta\mathbb{E}\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2+(\eta^2+\eta)\beta C_q^m\mathbb{E}\|\mathbf{q}^{k-1}\|^2.
\end{aligned}\tag{A.15}$$

Letting $\mathbf{s}_i=\mathbf{h}_i^*$ in (A.2), we have

$$\begin{aligned}
&\frac{(1+\eta)c\beta\gamma^2}{n}\sum_{i=1}^n\mathbb{E}\|\mathbf{h}_i^{k+1}-\mathbf{h}_i^*\|^2\\
\leq&\frac{(1+\eta)(1-\alpha)c\beta\gamma^2}{n}\sum_{i=1}^n\|\mathbf{h}_i^k-\mathbf{h}_i^*\|^2+\frac{(1+\eta)\alpha c\beta\gamma^2}{n}\sum_{i=1}^n\|\mathbf{g}_i^k-\mathbf{h}_i^*\|^2\\
&+\frac{(1+\eta)\alpha[(C_q+1)\alpha-1]c\beta\gamma^2}{n}\sum_{i=1}^n\|\Delta_i^k\|^2.
\end{aligned}\tag{A.16}$$

Then we let $\mathbf{R}^k=\beta(1-(C_q^m+1)\beta)\mathbb{E}\|\mathbf{q}^k\|^2$ and define

$$\mathbf{V}^k=\mathbf{R}^{k-1}+\mathbb{E}\|\hat{\mathbf{x}}^k-\mathbf{x}^*\|^2+\frac{(1+\eta)c\beta\gamma^2}{n}\sum_{i=1}^n\mathbb{E}\|\mathbf{h}_i^k-\mathbf{h}_i^*\|^2.$$

Thus, we obtain

$$\begin{aligned}
\mathbf{V}^{k+1}\leq&(\eta^2+\eta)\beta C_q^m\mathbb{E}\|\mathbf{q}^{k-1}\|^2+(1+\eta\beta)\mathbb{E}\|\hat{\mathbf{x}}^k-\mathbf{x}^*\|^2-2(1+\eta)\beta\gamma\mathbb{E}\langle\hat{\mathbf{x}}^k-\mathbf{x}^*,\nabla f(\hat{\mathbf{x}}^k)-\mathbf{h}^*\rangle\\
&+\frac{(1+\eta)(1-\alpha)c\beta\gamma^2}{n}\sum_{i=1}^n\mathbb{E}\|\mathbf{h}_i^k-\mathbf{h}_i^*\|^2\\
&+\frac{(1+\eta)\beta\gamma^2}{n^2}\Big[nc(C_q+1)\alpha^2-nc\alpha+C_q\Big]\sum_{i=1}^n\mathbb{E}\|\Delta_i^k\|^2\\
&+\frac{(1+\eta)(1+c\alpha)}{n}\beta\gamma^2\sum_{i=1}^n\mathbb{E}\|\nabla f_i(\hat{\mathbf{x}}^k)-\mathbf{h}_i^*\|^2+\frac{(1+\eta)(1+nc\alpha)}{n}\beta\gamma^2\sigma^2.
\end{aligned}\tag{A.17}$$

The $\mathbb{E}\|\Delta_i^k\|^2$-term can be ignored if $nc(C_q+1)\alpha^2 - nc\alpha + C_q \leq 0$, which can be guaranteed by $c \geq \frac{4C_q(C_q+1)}{n}$ and

$$\alpha \in \left(\frac{1-\sqrt{1-\frac{4C_q(C_q+1)}{nc}}}{2(C_q+1)}, \frac{1+\sqrt{1-\frac{4C_q(C_q+1)}{nc}}}{2(C_q+1)} \right).$$

Given that each f_i is L-Lipschitz differentiable and μ-strongly convex, we have

$$\mathbb{E}\langle \nabla f(\hat{\mathbf{x}}^k) - \mathbf{h}^*, \hat{\mathbf{x}}^k - \mathbf{x}^* \rangle \geq \frac{\mu L}{\mu+L}\mathbb{E}\|\hat{\mathbf{x}}^k - \mathbf{x}^*\|^2 + \frac{1}{\mu+L}\frac{1}{n}\sum_{i=1}^n \mathbb{E}\|\nabla f_i(\hat{\mathbf{x}}^k) - \mathbf{h}_i^*\|^2. \tag{A.18}$$

Hence

$$\begin{aligned}
\mathbf{V}^{k+1} \leq & \rho_1 \mathbf{R}^{k-1} + (1+\eta\beta)\mathbb{E}\|\hat{\mathbf{x}}^k - \mathbf{x}^*\|^2 - 2(1+\eta)\beta\gamma\mathbb{E}\langle \hat{\mathbf{x}}^k - \mathbf{x}^*, \nabla f(\hat{\mathbf{x}}^k) - \mathbf{h}^* \rangle \\
& + \frac{(1+\eta)(1-\alpha)c\beta\gamma^2}{n}\sum_{i=1}^n \mathbb{E}\|\mathbf{h}_i^k - \mathbf{h}_i^*\|^2 + \frac{(1+\eta)(1+c\alpha)}{n}\beta\gamma^2\sum_{i=1}^n \mathbb{E}\|\nabla f_i(\hat{\mathbf{x}}^k) - \mathbf{h}_i^*\|^2 \\
& + \frac{(1+\eta)(1+nc\alpha)}{n}\beta\gamma^2\sigma^2 \\
\leq & \rho_1 \mathbf{R}^{k-1} + \left[1 + \eta\beta - \frac{2(1+\eta)\beta\gamma\mu L}{\mu+L}\right]\mathbb{E}\|\hat{\mathbf{x}}^k - \mathbf{x}^*\|^2 + \frac{(1+\eta)(1-\alpha)c\beta\gamma^2}{n}\sum_{i=1}^n \mathbb{E}\|\mathbf{h}_i^k - \mathbf{h}_i^*\|^2 \\
& + \left[(1+\eta)(1+c\alpha)\beta\gamma^2 - \frac{2(1+\eta)\beta\gamma}{\mu+L}\right]\frac{1}{n}\sum_{i=1}^n \mathbb{E}\|\nabla f_i(\hat{\mathbf{x}}^k) - \mathbf{h}_i^*\|^2 + \frac{(1+\eta)(1+nc\alpha)}{n}\beta\gamma^2\sigma^2 \\
\leq & \rho_1 \mathbf{R}^{k-1} + \rho_2\mathbb{E}\|\hat{\mathbf{x}}^k - \mathbf{x}^*\|^2 + \frac{(1+\eta)(1-\alpha)c\beta\gamma^2}{n}\sum_{i=1}^n \mathbb{E}\|\mathbf{h}_i^k - \mathbf{h}_i^*\|^2 + \frac{(1+\eta)(1+nc\alpha)}{n}\beta\gamma^2\sigma^2
\end{aligned} \tag{A.19}$$

where

$$\begin{aligned}
\rho_1 = & \frac{(\eta^2+\eta)C_q^m}{1-(C_q^m+1)\beta}, \\
\rho_2 = & 1 + \eta\beta - \frac{2(1+\eta)\beta\gamma\mu L}{\mu+L}.
\end{aligned}$$

Here we let $\gamma \leq \frac{2}{(1+c\alpha)(\mu+L)}$ such that $(1+\eta)(1+c\alpha)\beta\gamma^2 - \frac{2(1+\eta)\beta\gamma}{\mu+L} \leq 0$ and the last inequality holds. In order to get $\max(\rho_1, \rho_2, 1-\alpha) < 1$, we have the following conditions

$$\begin{aligned}
0 \leq (\eta^2+\eta)C_q^m \leq & 1 - (C_q^m+1)\beta, \\
\eta < & \frac{2(1+\eta)\gamma\mu L}{\mu+L}.
\end{aligned}$$

Therefore, the condition for γ is

$$\frac{\eta(\mu+L)}{2(1+\eta)\mu L} \leq \gamma \leq \frac{2}{(1+c\alpha)(\mu+L)},$$

which implies an additional condition for η. Therefore, the condition for η is

$$\eta \in \left[0, \min\left(\frac{-C_q^m + \sqrt{(C_q^m)^2 + 4(1-(C_q^m+1)\beta)}}{2C_q^m}, \frac{4\mu L}{(\mu+L)^2(1+c\alpha) - 4\mu L}\right)\right).$$

where $\eta \leq \frac{4\mu L}{(\mu+L)^2(1+c\alpha)-4\mu L}$ is to ensure $\frac{\eta(\mu+L)}{2(1+\eta)\mu L} \leq \frac{2}{(1+c\alpha)(\mu+L)}$ such that we don't get an empty set for γ.

If we define $\rho = \max\{\rho_1, \rho_2, 1-\alpha\}$, we obtain

$$\mathbf{V}^{k+1} \leq \rho \mathbf{V}^k + \frac{(1+\eta)(1+nc\alpha)}{n}\beta\gamma^2\sigma^2 \tag{A.20}$$

and the proof is completed by applying (A.20) recurrently. □

A.2.2 Proof of Theorem 2

Proof. In Algorithm 2, we can show

$$\begin{aligned}\mathbb{E}\|\hat{\mathbf{x}}^{k+1} - \hat{\mathbf{x}}^k\|^2 &= \beta^2\mathbb{E}\|\hat{\mathbf{q}}^k\|^2 = \beta^2\mathbb{E}\|\mathbb{E}\hat{\mathbf{q}}^k\|^2 + \beta^2\mathbb{E}\|\hat{\mathbf{q}}^k - \mathbb{E}\hat{\mathbf{q}}^k\|^2 \\ &= \beta^2\mathbb{E}\|\mathbf{q}^k\|^2 + \beta^2\mathbb{E}\|\hat{\mathbf{q}}^k - \mathbf{q}^k\|^2 \\ &\leq (1+C_q^m)\beta^2\mathbb{E}\|\mathbf{q}^k\|^2.\end{aligned} \tag{A.21}$$

and

$$\mathbb{E}\|\mathbf{q}^k\|^2 = \mathbb{E}\|-\gamma\hat{\mathbf{g}}^k + \eta\mathbf{e}^k\|^2 \leq 2\gamma^2\mathbb{E}\|\hat{\mathbf{g}}^k\|^2 + 2\eta^2\mathbb{E}\|\mathbf{e}^k\|^2 \leq 2\gamma^2\mathbb{E}\|\hat{\mathbf{g}}^k\|^2 + 2C_q^m\eta^2\mathbb{E}\|\mathbf{q}^{k-1}\|^2. \tag{A.22}$$

Using (A.21)(A.22) and the Lipschitz continuity of $\nabla f(\mathbf{x})$, we have

$$\begin{aligned}&\mathbb{E}f(\hat{\mathbf{x}}^{k+1}) + (C_q^m+1)L\beta^2\mathbb{E}\|\mathbf{q}^k\|^2 \\ \leq&\mathbb{E}f(\hat{\mathbf{x}}^k) + \mathbb{E}\langle\nabla f(\hat{\mathbf{x}}^k), \hat{\mathbf{x}}^{k+1} - \hat{\mathbf{x}}^k\rangle + \frac{L}{2}\mathbb{E}\|\hat{\mathbf{x}}^{k+1} - \hat{\mathbf{x}}^k\|^2 + (C_q^m+1)L\beta^2\mathbb{E}\|\mathbf{q}^k\|^2 \\ =&\mathbb{E}f(\hat{\mathbf{x}}^k) + \beta\mathbb{E}\langle\nabla f(\hat{\mathbf{x}}^k), -\gamma\hat{\mathbf{g}}^k + \eta\mathbf{e}^k\rangle + \frac{(1+C_q^m)L\beta^2}{2}\mathbb{E}\|\mathbf{q}^k\|^2 + (C_q^m+1)L\beta^2\mathbb{E}\|\mathbf{q}^k\|^2\end{aligned}$$

$$
\begin{aligned}
=&\mathbb{E}f(\hat{\mathbf{x}}^k) + \beta\mathbb{E}\langle\nabla f(\hat{\mathbf{x}}^k), -\gamma\nabla f(\hat{\mathbf{x}}^k) + \eta\mathbf{e}^k\rangle + \frac{3(C_q^m+1)L\beta^2}{2}\mathbb{E}\|\mathbf{q}^k\|^2\\
\leq&\mathbb{E}f(\hat{\mathbf{x}}^k) - \beta\gamma\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2 + \frac{\beta\eta}{2}\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2 + \frac{\beta\eta}{2}\mathbb{E}\|\mathbf{e}^k\|^2\\
&+ 3(C_q^m+1)L\beta^2\left[\gamma^2\mathbb{E}\|\hat{\mathbf{g}}^k\|^2 + C_q^m\eta^2\mathbb{E}\|\mathbf{q}^{k-1}\|^2\right]\\
\leq&\mathbb{E}f(\hat{\mathbf{x}}^k) - \left[\beta\gamma - \frac{\beta\eta}{2} - 3(C_q^m+1)L\beta^2\gamma^2\right]\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2\\
&+ \frac{3C_q(C_q^m+1)L\beta^2\gamma^2}{n^2}\sum_{i=1}^n\mathbb{E}\|\Delta_i^k\|^2 + \frac{3(C_q^m+1)L\beta^2\gamma^2}{n}\sigma^2\\
&+ \left[\frac{\beta\eta C_q^m}{2} + (3C_q^m+1)C_q^mL\beta^2\eta^2\right]\mathbb{E}\|\mathbf{q}^{k-1}\|^2,
\end{aligned}
\tag{A.23}
$$

where the last inequality is from (A.7) with $\mathbf{h}^* = \mathbf{0}$.

Letting $\mathbf{s}_i = \mathbf{0}$ in (A.2), we have

$$
\mathbb{E}_Q\|\mathbf{h}_i^{k+1}\|^2 \leq (1-\alpha)\|\mathbf{h}_i^k\|^2 + \alpha\|\mathbf{g}_i^k\|^2 + \alpha[(C_q+1)\alpha - 1]\|\Delta_i^k\|^2. \tag{A.24}
$$

Due to the assumption that each worker samples the gradient from the full dataset, we have

$$
\mathbb{E}\mathbf{g}_i^k = \mathbb{E}\nabla f(\hat{\mathbf{x}}^k), \quad \mathbb{E}\|\mathbf{g}_i^k\|^2 \leq \mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2 + \sigma_i^2. \tag{A.25}
$$

Define $\Lambda^k = (C_q^m+1)L\beta^2\|\mathbf{q}^{k-1}\|^2 + f(\hat{\mathbf{x}}^k) - f^* + 3c(C_q^m+1)L\beta^2\gamma^2\frac{1}{n}\sum_{i=1}^n\mathbb{E}\|\mathbf{h}_i^k\|^2$, and from (A.23), (A.24), and (A.25), we have

$$
\begin{aligned}
\mathbb{E}\Lambda^{k+1} \leq&\mathbb{E}f(\hat{\mathbf{x}}^k) - f^* + 3(1-\alpha)c(C_q^m+1)L\beta^2\gamma^2\frac{1}{n}\sum_{i=1}^n\mathbb{E}\|\mathbf{h}_i^k\|^2\\
&- \left[\beta\gamma - \frac{\beta\eta}{2} - 3(1+c\alpha)(C_q^m+1)L\beta^2\gamma^2\right]\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2\\
&+ \frac{(C_q^m+1)L\beta^2\gamma^2}{n^2}\left[3nc(C_q+1)\alpha^2 - 3nc\alpha + 3C_q\right]\sum_{i=1}^n\mathbb{E}\|\Delta_i^k\|^2\\
&+ 3(1+nc\alpha)\frac{(C_q^m+1)L\beta^2\gamma^2\sigma^2}{n}\\
&+ \left[\frac{\beta\eta C_q^m}{2} + 3(C_q^m+1)C_q^mL\beta^2\eta^2\right]\mathbb{E}\|\mathbf{q}^{k-1}\|^2.
\end{aligned}
\tag{A.26}
$$

If we let $c = \frac{4C_q(C_q+1)}{n}$, then the condition of α in (2.5) gives $3nc(C_q+1)\alpha^2 - 3nc\alpha + 3C_q \leq 0$ and

$$
\mathbb{E}\Lambda^{k+1} \leq\mathbb{E}f(\hat{\mathbf{x}}^k) - f^* + 3(1-\alpha)c(C_q^m+1)L\beta^2\gamma^2\frac{1}{n}\sum_{i=1}^n\mathbb{E}\|\mathbf{h}_i^k\|^2
$$

$$
\begin{aligned}
&- \left[\beta\gamma - \frac{\beta\eta}{2} - 3(1+c\alpha)(C_q^m+1)L\beta^2\gamma^2\right]\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2 \\
&+ 3(1+nc\alpha)\frac{(C_q^m+1)L\beta^2\gamma^2\sigma^2}{n} \\
&+ [\frac{\beta\eta C_q^m}{2} + 3(C_q^m+1)C_q^m L\beta^2\eta^2]\mathbb{E}\|\mathbf{q}^{k-1}\|^2. \qquad \text{(A.27)}
\end{aligned}
$$

Let $\eta = \gamma$ and $\beta\gamma \leq \frac{1}{6(1+c\alpha)(C_q^m+1)L}$, we have

$$
\beta\gamma - \frac{\beta\eta}{2} - 3(1+c\alpha)(C_q^m+1)L\beta^2\gamma^2 = \frac{\beta\gamma}{2} - 3(1+c\alpha)(C_q^m+1)L\beta^2\gamma^2 \geq 0.
$$

Take $\gamma \leq \min\left\{\frac{-1+\sqrt{1+\frac{48L^2\beta^2(C_q^m+1)^2}{C_q^m}}}{12L\beta(C_q^m+1)}, \frac{1}{6L\beta(1+c\alpha)(C_q^m+1)}\right\}$ will guarantee

$$
\left[\frac{\beta\eta C_q^m}{2} + 3(C_q^m+1)C_q^m L\beta^2\eta^2\right] \leq (C_q^m+1)L\beta^2.
$$

Hence we obtain

$$
\mathbb{E}\Lambda^{k+1} \leq \mathbb{E}\Lambda^k - \left[\frac{\beta\gamma}{2} - 3(1+c\alpha)(C_q^m+1)L\beta^2\gamma^2\right]\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2 + 3(1+nc\alpha)\frac{(C_q^m+1)L\beta^2\gamma^2\sigma^2}{n}. \qquad \text{(A.28)}
$$

Taking the telescoping sum and plugging the initial conditions, we derive (2.12). □

A.2.3 Proof of Corollary 2

Proof. With $\alpha = \frac{1}{2(C_q+1)}$ and $c = \frac{4C_q(C_q+1)}{n}$, $1 + nc\alpha = 1 + 2C_q$ is a constant. We set $\beta = \frac{1}{C_q^m+1}$ and $\gamma = \min\left\{\frac{-1+\sqrt{1+\frac{48L^2}{C_q^m}}}{12L}, \frac{1}{12L(1+c\alpha)(1+\sqrt{K/n})}\right\}$. In general, C_q^m is bounded which makes the first bound negligible, i.e., $\gamma = \frac{1}{12L(1+c\alpha)(1+\sqrt{K/n})}$ when K is large enough. Therefore, we have

$$
\frac{\beta}{2} - 3(1+c\alpha)(C_q^m+1)L\beta^2\gamma = \frac{1-6(1+c\alpha)L\gamma}{2(C_q^m+1)} \leq \frac{1}{4(C_q^m+1)}. \qquad \text{(A.29)}
$$

From Theorem 2, we derive

$$
\begin{aligned}
&\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2 \\
\leq &\frac{4(C_q^m+1)(\mathbb{E}\Lambda^1 - \mathbb{E}\Lambda^{K+1})}{\gamma K} + \frac{12(1+nc\alpha)L\sigma^2\gamma}{n}
\end{aligned}
$$

$$\leq 48L(C_q^m + 1)(1 + c\alpha)(\mathbb{E}\Lambda^1 - \mathbb{E}\Lambda^{K+1})(\frac{1}{K} + \frac{1}{\sqrt{nK}}) + \frac{(1 + nc\alpha)\sigma^2}{(1 + c\alpha)}\frac{1}{\sqrt{nK}}, \tag{A.30}$$

which completes the proof.

□

APPENDIX B

LINEAR CONVERGENT DECENTRALIZED OPTIMIZATION WITH COMPRESSION

B.1 Compression method

B.1.1 p-norm b-bits quantization

Theorem 9 (p-norm b-bit quantization)**.** *Let us define the quantization operator as*

$$Q_p(\mathbf{x}) := \left(\|\mathbf{x}\|_p \text{sign}(\mathbf{x}) 2^{-(b-1)}\right) \cdot \left\lfloor \frac{2^{b-1}|\mathbf{x}|}{\|\mathbf{x}\|_p} + \mathbf{u} \right\rfloor \tag{B.1}$$

where $\cdot$ is the Hadamard product, $|\mathbf{x}|$ is the elementwise absolute value and $\mathbf{u}$ is a random dither vector uniformly distributed in $[0,1]^d$. $Q_p(\mathbf{x})$ is unbiased, i.e., $\mathbb{E}Q_p(\mathbf{x}) = \mathbf{x}$, and the compression variance is upper bounded by

$$\mathbb{E}\|\mathbf{x} - Q_p(\mathbf{x})\|^2 \leq \frac{1}{4}\|\text{sign}(\mathbf{x})2^{-(b-1)}\|^2 \|\mathbf{x}\|_p^2, \tag{B.2}$$

which suggests that ∞-norm provides the smallest upper bound for the compression variance due to $\|\mathbf{x}\|_p \leq \|\mathbf{x}\|_q, \forall \mathbf{x}$ if $1 \leq q \leq p \leq \infty$.

Remark 13. *For the compressor defined in* (B.1)*, we have the following the compression constant*

$$C = \sup_{\mathbf{x}} \frac{\|\text{sign}(\mathbf{x})2^{-(b-1)}\|^2 \|\mathbf{x}\|_p^2}{4\|\mathbf{x}\|^2}.$$

Proof. Let denote $\mathbf{v} = \|\mathbf{x}\|_p \text{sign}(\mathbf{x}) 2^{-(b-1)}$, $s = \frac{2^{b-1}|\mathbf{x}|}{\|\mathbf{x}\|_p}$, $s_1 = \left\lfloor \frac{2^{b-1}|\mathbf{x}|}{\|\mathbf{x}\|_p} \right\rfloor$ and $s_2 = \left\lceil \frac{2^{b-1}|\mathbf{x}|}{\|\mathbf{x}\|_p} \right\rceil$. We can rewrite $\mathbf{x}$ as $\mathbf{x} = s \cdot \mathbf{v}$.

For any coordinate i such that $s_i = (s_1)_i$, we have $Q_p(\mathbf{x}_i) = (s_1)_i \mathbf{v}_i$ with probability 1. Hence $\mathbb{E}Q_p(\mathbf{x})_i = s_i \mathbf{v}_i = \mathbf{x}_i$ and

$$\mathbb{E}(\mathbf{x}_i - Q_p(\mathbf{x})_i)^2 = (\mathbf{x}_i - s_i \mathbf{v}_i)^2 = 0.$$

For any coordinate i such that $s_i \neq (s_1)_i$, we have $(s_2)_i - (s_1)_i = 1$ and $Q_p(\mathbf{x})_i$ satisfies

$$Q_p(\mathbf{x})_i = \begin{cases} (s_1)_i \mathbf{v}_i, & \text{w.p. } (s_2)_i - s_i, \\ (s_2)_i \mathbf{v}_i, & \text{w.p. } s_i - (s_1)_i. \end{cases}$$

Thus, we derive

$$\mathbb{E}Q_p(\mathbf{x})_i = \mathbf{v}_i(s_1)_i(s_2 - s)_i + \mathbf{v}_i(s_2)_i(s - s_1)_i = \mathbf{v}_i s_i(s_2 - s_1)_i = \mathbf{v}_i s_i = \mathbf{x}_i,$$

and

$$\begin{aligned}
\mathbb{E}[\mathbf{x}_i - Q_p(\mathbf{x})_i]^2 &= (\mathbf{x}_i - \mathbf{v}_i(s_1)_i)^2(s_2 - s)_i + (\mathbf{x}_i - \mathbf{v}_i(s_2)_i)^2(s - s_1)_i \\
&= (s_2 - s_1)_i\mathbf{x}_i^2 + \big((s_1)_i(s_2)_i(s_1 - s_2)_i + s_i((s_2)_i^2 - (s_1)_i^2)\big)\mathbf{v}_i^2 - 2s_i(s_2 - s_1)_i\mathbf{x}_i\mathbf{v}_i \\
&= \mathbf{x}_i^2 + \big(-(s_1)_i(s_2)_i + s_i(s_2 + s_1)_i\big)\mathbf{v}_i^2 - 2s_i\mathbf{x}_i\mathbf{v}_i \\
&= (\mathbf{x}_i - s_i\mathbf{v}_i)^2 + \big(-(s_1)_i(s_2)_i + s_i(s_2 + s_1)_i - s_i^2\big)\mathbf{v}_i^2 \\
&= (\mathbf{x}_i - s_i\mathbf{v}_i)^2 + (s_2 - s)_i(s - s_1)_i\mathbf{v}_i^2 \\
&= (s_2 - s)_i(s - s_1)_i\mathbf{v}_i^2 \\
&\leq \frac{1}{4}\mathbf{v}_i^2.
\end{aligned}$$

Considering both cases, we have $\mathbb{E}Q(\mathbf{x}) = \mathbf{x}$ and

$$\begin{aligned}
\mathbb{E}\|\mathbf{x} - Q_p(\mathbf{x})\|^2 &= \sum_{\{s_i = (s_1)_i\}} \mathbb{E}[\mathbf{x}_i - Q_p(\mathbf{x})_i]^2 + \sum_{\{s_i \neq (s_1)_i\}} \mathbb{E}[\mathbf{x}_i - Q_p(\mathbf{x})_i]^2 \\
&\leq 0 + \frac{1}{4}\sum_{\{s_i \neq (s_1)_i\}} \mathbf{v}_i^2 \\
&\leq \frac{1}{4}\|\mathbf{v}\|^2 \\
&= \frac{1}{4}\|\mathrm{sign}(\mathbf{x})2^{-(b-1)}\|^2\|\mathbf{x}\|_p^2.
\end{aligned}$$

□

B.1.2 Compression error

To verify Theorem 9, we compare the compression error of the quantization method defined in (B.1) with different norms ($p = 1, 2, 3, \ldots, 6, \infty$). Specifically, we uniformly generate 100 random vectors in $\mathbb{R}^{10000}$ and compute the average compression error. The result shown in Figure B.1 verifies our proof in Theorem 9 that the compression error decreases when p increases. This suggests that ∞-norm provides the best compression precision under the same bit constraint.

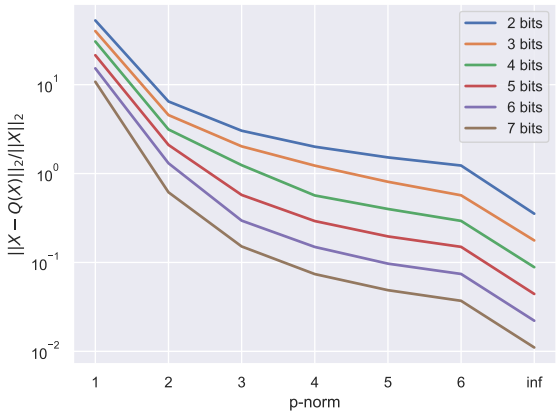


Figure B.1: Relative compression error $\frac{\|\mathbf{x}-Q(\mathbf{x})\|_2}{\|\mathbf{x}\|_2}$ for p-norm b-bit quantization.

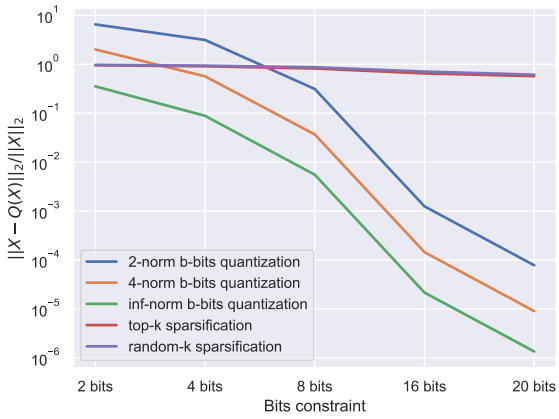


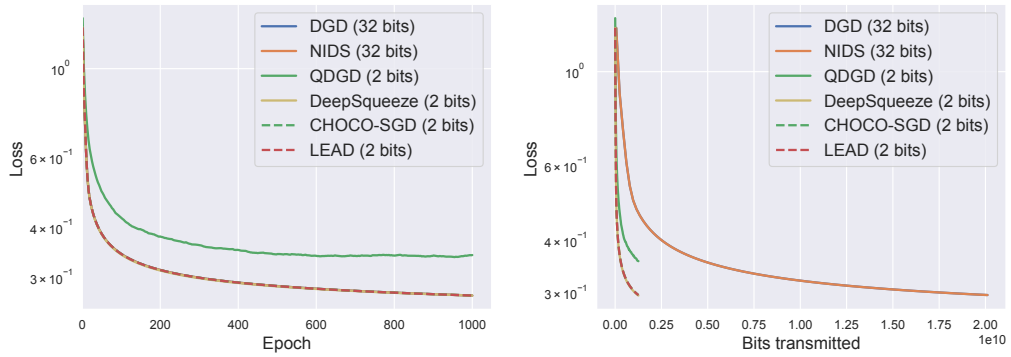
Figure B.2: Comparison of compression error $\frac{\|\mathbf{x}-Q(\mathbf{x})\|_2}{\|\mathbf{x}\|_2}$ between different compression methods.

Under similar setting, we also compare the compression error with other popular compression methods, such as top-k and random-k sparsification. The x-axes represents the average bits needed to represent each element of the vector. The result is showed in Fig. B.2. Note that intuitively top-k methods should perform better than random-k method, but the top-k method needs extra bits to transmitted the index while random-k method can avoid this by using the same random seed. Therefore, top-k method doesn't outperform random-k too much under the same communication budget. The result in Fig. B.2 suggests that ∞-norm b-bits quantization provides significantly better compression precision than others under the same bit constraint.

B.2 Experiments

B.2.1 Experiments in homogeneous setting

The experiments on logistic regression problem in homogeneous case are showed in Fig. B.3 and Fig. B.4. It shows that DeepSqueeze, CHOCO-SGD and LEAD converges similarly while DeepSqueeze and CHOCO-SGD require to tune a smaller γ for convergence as showed in the parameter setting in Section B.2.2. Generally, a smaller γ decreases the model propagation between agents since γ changes the effective mixing matrix and this may cause slower convergence. However, in the setting where data from different agents are very similar, the models move to close directions such that the convergence is not affected too much.

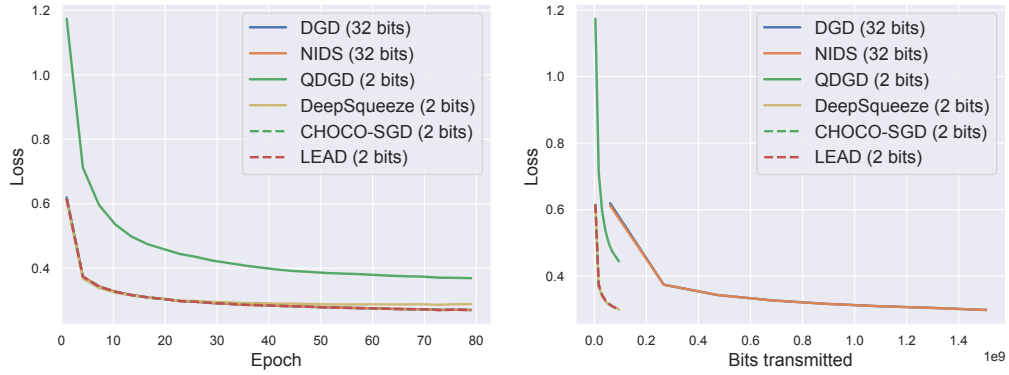


(a) Loss $f(\overline{\mathbf{X}}^k)$ (b) Loss $f(\overline{\mathbf{X}}^k)$

Figure B.3: Logistic regression in the homogeneous case (full-batch gradient).

B.2.2 Parameter settings

The best parameter settings we search for all algorithms and experiments are summarized in Tables B.1– B.4. QDGD and DeepSqueeze are more sensitive to γ and CHOCO-SGD is slight more robust. LEAD is most robust to parameter settings and it works well for the setting $\alpha = 0.5$ and $\gamma = 1.0$ in all experiments in this work.



(a) Loss $f(\overline{\mathbf{X}}^k)$ (b) Loss $f(\overline{\mathbf{X}}^k)$

Figure B.4: Logistic regression in the homogeneous case (mini-batch gradient).

Algorithm	η	γ	α
DGD	0.1	-	-
NIDS	0.1	-	-
QDGD	0.1	0.2	-
DeepSqueeze	0.1	0.2	-
CHOCO-SGD	0.1	0.8	-
LEAD	0.1	1.0	0.5

Table B.1: Parameter settings for the linear regression problem.

Algorithm	η	γ	α
DGD	0.1	-	-
NIDS	0.1	-	-
QDGD	0.1	0.4	-
DeepSqueeze	0.1	0.4	-
CHOCO-SGD	0.1	0.6	-
LEAD	0.1	1.0	0.5

Homogeneous case

Algorithm	η	γ	α
DGD	0.1	-	-
NIDS	0.1	-	-
QDGD	0.1	0.2	-
DeepSqueeze	0.1	0.6	-
CHOCO-SGD	0.1	0.6	-
LEAD	0.1	1.0	0.5

Heterogeneous case

Table B.2: Parameter settings for the logistic regression problem (full-batch gradient).

B.3 Proofs of the theorems

B.3.1 Illustrative flow

The following flow graph depicts the relation between iterative variables and clarifies the range of conditional expectation. $\{\mathcal{G}_k\}_{k=0}^{\infty}$ and $\{\mathcal{F}_k\}_{k=0}^{\infty}$ are two $\sigma-$algebras generated by the gradient

Algorithm	η	γ	α
DGD	0.1	-	-
NIDS	0.1	-	-
QDGD	0.05	0.2	-
DeepSqueeze	0.1	0.6	-
CHOCO-SGD	0.1	0.6	-
LEAD	0.1	1.0	0.5

Homogeneous case

Algorithm	η	γ	α
DGD	0.1	-	-
NIDS	0.1	-	-
QDGD	0.05	0.2	-
DeepSqueeze	0.1	0.6	-
CHOCO-SGD	0.1	0.6	-
LEAD	0.1	1.0	0.5

Heterogeneous case

Table B.3: Parameter settings for the logistic regression problem (mini-batch gradient).

Algorithm	η	γ	α
DGD	0.1	-	-
NIDS	0.1	-	-
QDGD	0.05	0.1	-
DeepSqueeze	0.1	0.2	-
CHOCO-SGD	0.1	0.6	-
LEAD	0.1	1.0	0.5

Homogeneous case

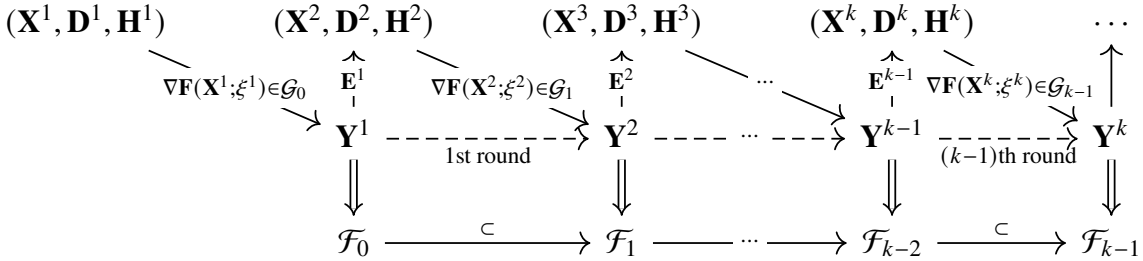
Algorithm	η	γ	α
DGD	0.05	-	-
NIDS	0.1	-	-
QDGD	*	*	-
DeepSqueeze	*	*	-
CHOCO-SGD	*	*	-
LEAD	0.1	1.0	0.5

Heterogeneous case

Table B.4: Parameter settings for the deep neural network. (* means divergence for all options we try).

sampling and the stochastic compression respectively. They satisfy

$$\mathcal{G}_0 \subset \mathcal{F}_0 \subset \mathcal{G}_1 \subset \mathcal{F}_1 \subset \cdots \subset \mathcal{G}_k \subset \mathcal{F}_k \subset \cdots$$



The solid and dashed arrows in the top flow illustrate the dynamics of the algorithm, while in the bottom, the arrows stand for the relation between successive $\mathcal{F}$-σ-algebras. The downward arrows determine the range of $\mathcal{F}$-σ-algebras. E.g., up to $\mathbf{E}^k$, all random variables are in $\mathcal{F}_{k-1}$ and up to $\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)$, all random variables are in $\mathcal{G}_{k-1}$ with $\mathcal{G}_{k-1} \subset \mathcal{F}_{k-1}$. Throughout the appendix, without specification, $\mathbb{E}$ is the expectation conditioned on the corresponding stochastic estimators given the

context.

B.3.2 Two central Lemmas

Lemma 10 (Fundamental equality)**.** *Let* $\mathbf{X}^*$ *be the optimal solution,* $\mathbf{D}^* := -\nabla\mathbf{F}(\mathbf{X}^*)$ *and* $\mathbf{E}^k$ *denote the compression error in the* k*th iteration, that is* $\mathbf{E}^k = \mathbf{Q}^k - (\mathbf{Y}^k - \mathbf{H}^k) = \hat{\mathbf{Y}}^k - \mathbf{Y}^k$*. From Alg. 3, we have*

$$\begin{aligned}
&\|\mathbf{X}^{k+1} - \mathbf{X}^*\|^2 + (\eta^2/\gamma)\|\mathbf{D}^{k+1} - \mathbf{D}^*\|_{\mathbf{M}}^2 \\
=&\|\mathbf{X}^k - \mathbf{X}^*\|^2 + (\eta^2/\gamma)\|\mathbf{D}^k - \mathbf{D}^*\|_{\mathbf{M}}^2 - (\eta^2/\gamma)\|\mathbf{D}^{k+1} - \mathbf{D}^k\|_{\mathbf{M}}^2 - \eta^2\|\mathbf{D}^{k+1} - \mathbf{D}^*\|^2 \\
&- 2\eta\langle\mathbf{X}^k - \mathbf{X}^*, \nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \nabla\mathbf{F}(\mathbf{X}^*)\rangle + \eta^2\|\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \nabla\mathbf{F}(\mathbf{X}^*)\|^2 + 2\eta\langle\mathbf{E}^k, \mathbf{D}^{k+1} - \mathbf{D}^*\rangle,
\end{aligned}$$

where $\mathbf{M} := 2(\mathbf{I} - \mathbf{W})^\dagger - \gamma\mathbf{I}$ *and* $\gamma < 2/\lambda_{\max}(\mathbf{I} - \mathbf{W})$ *ensures the positive definiteness of* $\mathbf{M}$ *over range*$(\mathbf{I} - \mathbf{W})$.

Lemma 11 (State inequality)**.** *Let the same assumptions in Lemma 10 hold. From Alg. 3, if we take the expectation over the compression operator conditioned on the* k*-th iteration, we have*

$$\begin{aligned}
\mathbb{E}\|\mathbf{H}^{k+1} - \mathbf{X}^*\|^2 \le\ & (1-\alpha)\|\mathbf{H}^k - \mathbf{X}^*\|^2 + \alpha\mathbb{E}\|\mathbf{X}^{k+1} - \mathbf{X}^*\|^2 + \alpha\eta^2\mathbb{E}\|\mathbf{D}^{k+1} - \mathbf{D}^k\|^2 \\
&+ \frac{2\alpha\eta^2}{\gamma}\mathbb{E}\|\mathbf{D}^{k+1} - \mathbf{D}^k\|_{\mathbf{M}}^2 + \alpha^2\mathbb{E}\|\mathbf{E}^k\|^2 - \alpha\gamma\mathbb{E}\|\mathbf{E}^k\|_{\mathbf{I}-\mathbf{W}}^2 - \alpha(1-\alpha)\|\mathbf{Y}^k - \mathbf{H}^k\|^2.
\end{aligned}$$

B.3.3 Proof of Lemma 10

Before proving Lemma 10, we let $\mathbf{E}^k = \hat{\mathbf{Y}}^k - \mathbf{Y}^k$ and introduce the following three Lemmas.

Lemma 12. *Let* $\mathbf{X}^*$ *be the consensus solution. Then, from Line 4-7 of Alg. 3, we obtain*

$$\frac{\mathbf{I} - \mathbf{W}}{2\eta}(\mathbf{X}^{k+1} - \mathbf{X}^*) = \left(\frac{I}{\gamma} - \frac{\mathbf{I} - \mathbf{W}}{2}\right)(\mathbf{D}^{k+1} - \mathbf{D}^k) - \frac{\mathbf{I} - \mathbf{W}}{2\eta}\mathbf{E}^k. \tag{B.3}$$

Proof. From the iterations in Alg. 3, we have

$$\mathbf{D}^{k+1} = \mathbf{D}^k + \frac{\gamma}{2\eta}(\mathbf{I} - \mathbf{W})\hat{\mathbf{Y}}^k \qquad \text{(from Line 6)}$$

$$
\begin{aligned}
&= \mathbf{D}^k + \frac{\gamma}{2\eta}(\mathbf{I}-\mathbf{W})(\mathbf{Y}^k + \mathbf{E}^k) \\
&= \mathbf{D}^k + \frac{\gamma}{2\eta}(\mathbf{I}-\mathbf{W})(\mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \eta\mathbf{D}^k + \mathbf{E}^k) \qquad \text{(from Line 4)} \\
&= \mathbf{D}^k + \frac{\gamma}{2\eta}(\mathbf{I}-\mathbf{W})(\mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \eta\mathbf{D}^{k+1} - \mathbf{X}^* + \eta(\mathbf{D}^{k+1}-\mathbf{D}^k) + \mathbf{E}^k) \\
&= \mathbf{D}^k + \frac{\gamma}{2\eta}(\mathbf{I}-\mathbf{W})(\mathbf{X}^{k+1} - \mathbf{X}^*) + \frac{\gamma}{2}(\mathbf{I}-\mathbf{W})(\mathbf{D}^{k+1}-\mathbf{D}^k) + \frac{\gamma}{2\eta}(\mathbf{I}-\mathbf{W})\mathbf{E}^k,
\end{aligned}
$$

where the fourth equality holds due to $(\mathbf{I}-\mathbf{W})\mathbf{X}^* = \mathbf{0}$ and the last equality comes from Line 7 of Alg. 3. Rewriting this equality, and we obtain (B.3). □

Lemma 13. *Let* $\mathbf{D}^* = -\nabla\mathbf{F}(\mathbf{X}^*) \in \mathbf{span}\{\mathbf{I}-\mathbf{W}\}$*, we have*

$$
\langle \mathbf{X}^{k+1} - \mathbf{X}^*, \mathbf{D}^{k+1} - \mathbf{D}^k \rangle = \frac{\eta}{\gamma}\|\mathbf{D}^{k+1} - \mathbf{D}^k\|_{\mathbf{M}}^2 - \langle \mathbf{E}^k, \mathbf{D}^{k+1} - \mathbf{D}^k \rangle, \tag{B.4}
$$

$$
\langle \mathbf{X}^{k+1} - \mathbf{X}^*, \mathbf{D}^{k+1} - \mathbf{D}^* \rangle = \frac{\eta}{\gamma}\langle \mathbf{D}^{k+1} - \mathbf{D}^k, \mathbf{D}^{k+1} - \mathbf{D}^* \rangle_{\mathbf{M}} - \langle \mathbf{E}^k, \mathbf{D}^{k+1} - \mathbf{D}^* \rangle, \tag{B.5}
$$

where $\mathbf{M} = 2(\mathbf{I}-\mathbf{W})^\dagger - \gamma\mathbf{I}$ *and* $\gamma < 2/\lambda_{\max}(\mathbf{I}-\mathbf{W})$ *ensures the positive definiteness of* $\mathbf{M}$ *over* $\operatorname{span}\{\mathbf{I}-\mathbf{W}\}$.

Proof. Since $\mathbf{D}^{k+1} \in \mathbf{span}\{\mathbf{I}-\mathbf{W}\}$ for any k, we have

$$
\begin{aligned}
&\langle \mathbf{X}^{k+1} - \mathbf{X}^*, \mathbf{D}^{k+1} - \mathbf{D}^k \rangle \\
=&\langle (\mathbf{I}-\mathbf{W})(\mathbf{X}^{k+1} - \mathbf{X}^*), (\mathbf{I}-\mathbf{W})^\dagger(\mathbf{D}^{k+1} - \mathbf{D}^k) \rangle \\
=&\left\langle \frac{\eta}{\gamma}(2\mathbf{I} - \gamma(\mathbf{I}-\mathbf{W}))(\mathbf{D}^{k+1} - \mathbf{D}^k) - (\mathbf{I}-\mathbf{W})\mathbf{E}^k, (\mathbf{I}-\mathbf{W})^\dagger(\mathbf{D}^{k+1} - \mathbf{D}^k) \right\rangle \qquad \text{(from (B.3))} \\
=&\left\langle \frac{\eta}{\gamma}\Big(2(\mathbf{I}-\mathbf{W})^\dagger - \gamma\mathbf{I}\Big)(\mathbf{D}^{k+1} - \mathbf{D}^k) - \mathbf{E}^k, \mathbf{D}^{k+1} - \mathbf{D}^k \right\rangle \\
=&\frac{\eta}{\gamma}\|\mathbf{D}^{k+1} - \mathbf{D}^k\|_{\mathbf{M}}^2 - \langle \mathbf{E}^k, \mathbf{D}^{k+1} - \mathbf{D}^k \rangle.
\end{aligned}
$$

Similarly, we have

$$
\begin{aligned}
&\langle \mathbf{X}^{k+1} - \mathbf{X}^*, \mathbf{D}^{k+1} - \mathbf{D}^* \rangle \\
=&\langle (\mathbf{I}-\mathbf{W})(\mathbf{X}^{k+1} - \mathbf{X}^*), (\mathbf{I}-\mathbf{W})^\dagger(\mathbf{D}^{k+1} - \mathbf{D}^*) \rangle \\
=&\left\langle \frac{\eta}{\gamma}(2\mathbf{I} - \gamma(\mathbf{I}-\mathbf{W}))(\mathbf{D}^{k+1} - \mathbf{D}^k) - (\mathbf{I}-\mathbf{W})\mathbf{E}^k, (\mathbf{I}-\mathbf{W})^\dagger(\mathbf{D}^{k+1} - \mathbf{D}^*) \right\rangle
\end{aligned}
$$

$$
\begin{aligned}
&= \left\langle \frac{\eta}{\gamma}(2(\mathbf{I}-\mathbf{W})^{\dagger}-\mathbf{I})(\mathbf{D}^{k+1}-\mathbf{D}^{k}) - \mathbf{E}^{k}, \mathbf{D}^{k+1}-\mathbf{D}^{*} \right\rangle \\
&= \frac{\eta}{\gamma}\langle \mathbf{D}^{k+1}-\mathbf{D}^{k}, \mathbf{D}^{k+1}-\mathbf{D}^{*}\rangle_{\mathbf{M}} - \langle \mathbf{E}^{k}, \mathbf{D}^{k+1}-\mathbf{D}^{*}\rangle.
\end{aligned}
$$

To make sure that $\mathbf{M}$ is positive definite over $\mathbf{span}\{\mathbf{I}-\mathbf{W}\}$, we need $\gamma < 2/\lambda_{\max}(\mathbf{I}-\mathbf{W})$. □

Lemma 14. *Taking the expectation conditioned on the compression in the kth iteration, we have*

$$
\begin{aligned}
2\eta\mathbb{E}\langle \mathbf{E}^{k}, \mathbf{D}^{k+1}-\mathbf{D}^{*}\rangle &= 2\eta\mathbb{E}\left\langle \mathbf{E}^{k}, \mathbf{D}^{k} + \frac{\gamma}{2\eta}(\mathbf{I}-\mathbf{W})\mathbf{Y}^{k} + \frac{\gamma}{2\eta}(\mathbf{I}-\mathbf{W})\mathbf{E}^{k} - \mathbf{D}^{*}\right\rangle \\
&= \gamma\mathbb{E}\langle \mathbf{E}^{k}, (\mathbf{I}-\mathbf{W})\mathbf{E}^{k}\rangle = \gamma\mathbb{E}\|\mathbf{E}^{k}\|^{2}_{\mathbf{I}-\mathbf{W}}, \\
2\eta\mathbb{E}\langle \mathbf{E}^{k}, \mathbf{D}^{k+1}-\mathbf{D}^{k}\rangle &= 2\eta\mathbb{E}\left\langle \mathbf{E}^{k}, \frac{\gamma}{2\eta}(\mathbf{I}-\mathbf{W})\mathbf{Y}^{k} + \frac{\gamma}{2\eta}(\mathbf{I}-\mathbf{W})\mathbf{E}^{k}\right\rangle \\
&= \gamma\mathbb{E}\langle \mathbf{E}^{k}, (\mathbf{I}-\mathbf{W})\mathbf{E}^{k}\rangle = \gamma\mathbb{E}\|\mathbf{E}^{k}\|^{2}_{\mathbf{I}-\mathbf{W}}.
\end{aligned}
$$

Proof. The proof is straightforward and omitted here. □

Proof of Lemma 10. From Alg. 3, we have

$$
\begin{aligned}
&2\eta\langle \mathbf{X}^{k}-\mathbf{X}^{*}, \nabla\mathbf{F}(\mathbf{X}^{k};\xi^{k}) - \nabla\mathbf{F}(\mathbf{X}^{*})\rangle \\
=&2\langle \mathbf{X}^{k}-\mathbf{X}^{*}, \eta\nabla\mathbf{F}(\mathbf{X}^{k};\xi^{k}) - \eta\nabla\mathbf{F}(\mathbf{X}^{*})\rangle \\
=&2\langle \mathbf{X}^{k}-\mathbf{X}^{*}, \mathbf{X}^{k}-\mathbf{X}^{k+1} - \eta(\mathbf{D}^{k+1}-\mathbf{D}^{*})\rangle \qquad \text{(from Line 7)} \\
=&2\langle \mathbf{X}^{k}-\mathbf{X}^{*}, \mathbf{X}^{k}-\mathbf{X}^{k+1}\rangle - 2\eta\langle \mathbf{X}^{k}-\mathbf{X}^{*}, \mathbf{D}^{k+1}-\mathbf{D}^{*}\rangle \\
=&2\langle \mathbf{X}^{k}-\mathbf{X}^{*}, \mathbf{X}^{k}-\mathbf{X}^{k+1}\rangle - 2\eta\langle \mathbf{X}^{k}-\mathbf{X}^{k+1}, \mathbf{D}^{k+1}-\mathbf{D}^{*}\rangle - 2\eta\langle \mathbf{X}^{k+1}-\mathbf{X}^{*}, \mathbf{D}^{k+1}-\mathbf{D}^{*}\rangle \\
=&2\langle \mathbf{X}^{k}-\mathbf{X}^{*} - \eta(\mathbf{D}^{k+1}-\mathbf{D}^{*}), \mathbf{X}^{k}-\mathbf{X}^{k+1}\rangle - 2\eta\langle \mathbf{X}^{k+1}-\mathbf{X}^{*}, \mathbf{D}^{k+1}-\mathbf{D}^{*}\rangle \\
=&2\langle \mathbf{X}^{k+1}-\mathbf{X}^{*} + \eta(\nabla\mathbf{F}(\mathbf{X}^{k};\xi^{k}) - \nabla\mathbf{F}(\mathbf{X}^{*})), \mathbf{X}^{k}-\mathbf{X}^{k+1}\rangle - 2\eta\langle \mathbf{X}^{k+1}-\mathbf{X}^{*}, \mathbf{D}^{k+1}-\mathbf{D}^{*}\rangle \text{ (from Line 7)} \\
=&2\langle \mathbf{X}^{k+1}-\mathbf{X}^{*}, \mathbf{X}^{k}-\mathbf{X}^{k+1}\rangle + 2\eta\langle \nabla\mathbf{F}(\mathbf{X}^{k};\xi^{k}) - \nabla\mathbf{F}(\mathbf{X}^{*}), \mathbf{X}^{k}-\mathbf{X}^{k+1}\rangle \\
&- 2\eta\langle \mathbf{X}^{k+1}-\mathbf{X}^{*}, \mathbf{D}^{k+1}-\mathbf{D}^{*}\rangle. \qquad \text{(B.6)}
\end{aligned}
$$

Then we consider the terms on the right hand side of (B.6) separately. Using $2\langle \mathbf{A}-\mathbf{B}, \mathbf{B}-\mathbf{C}\rangle = \|\mathbf{A}-\mathbf{C}\|^{2} - \|\mathbf{B}-\mathbf{C}\|^{2} - \|\mathbf{A}-\mathbf{B}\|^{2}$, we have

$$
2\langle \mathbf{X}^{k+1}-\mathbf{X}^{*}, \mathbf{X}^{k}-\mathbf{X}^{k+1}\rangle = 2\langle \mathbf{X}^{*}-\mathbf{X}^{k+1}, \mathbf{X}^{k+1}-\mathbf{X}^{k}\rangle
$$

$$= \|\mathbf{X}^k - \mathbf{X}^*\|^2 - \|\mathbf{X}^{k+1} - \mathbf{X}^k\|^2 - \|\mathbf{X}^{k+1} - \mathbf{X}^*\|^2. \tag{B.7}$$

Using $2\langle \mathbf{A}, \mathbf{B}\rangle = \|\mathbf{A}\|^2 + \|\mathbf{B}\|^2 - \|\mathbf{A} - \mathbf{B}\|^2$, we have

$$\begin{aligned}
&2\eta\langle \nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \nabla \mathbf{F}(\mathbf{X}^*), \mathbf{X}^k - \mathbf{X}^{k+1}\rangle \\
=&\eta^2 \|\nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \nabla \mathbf{F}(\mathbf{X}^*)\|^2 + \|\mathbf{X}^k - \mathbf{X}^{k+1}\|^2 - \|\mathbf{X}^k - \mathbf{X}^{k+1} - \eta(\nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \nabla \mathbf{F}(\mathbf{X}^*))\|^2 \\
=&\eta^2 \|\nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \nabla \mathbf{F}(\mathbf{X}^*)\|^2 + \|\mathbf{X}^k - \mathbf{X}^{k+1}\|^2 - \eta^2 \|\mathbf{D}^{k+1} - \mathbf{D}^*\|^2. \qquad \text{(from Line 7)}
\end{aligned} \tag{B.8}$$

Combining (B.6), (B.7), (B.8), and (B.4), we obtain

$$\begin{aligned}
&2\eta\langle \mathbf{X}^k - \mathbf{X}^*, \nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \nabla \mathbf{F}(\mathbf{X}^*)\rangle \\
=&\underbrace{\|\mathbf{X}^k - \mathbf{X}^*\|^2 - \|\mathbf{X}^{k+1} - \mathbf{X}^k\|^2 - \|\mathbf{X}^{k+1} - \mathbf{X}^*\|^2}_{2\langle \mathbf{X}^{k+1} - \mathbf{X}^*, \mathbf{X}^k - \mathbf{X}^{k+1}\rangle} \\
&+ \underbrace{\eta^2 \|\nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \nabla \mathbf{F}(\mathbf{X}^*)\|^2 + \|\mathbf{X}^k - \mathbf{X}^{k+1}\|^2 - \eta^2 \|\mathbf{D}^{k+1} - \mathbf{D}^*\|^2}_{2\eta\langle \nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \nabla \mathbf{F}(\mathbf{X}^*), \mathbf{X}^k - \mathbf{X}^{k+1}\rangle} \\
&- \underbrace{\left(\frac{2\eta^2}{\gamma}\langle \mathbf{D}^{k+1} - \mathbf{D}^k, \mathbf{D}^{k+1} - \mathbf{D}^*\rangle_{\mathbf{M}} - 2\eta\langle \mathbf{E}^k, \mathbf{D}^{k+1} - \mathbf{D}^*\rangle\right)}_{2\eta\langle \mathbf{X}^{k+1} - \mathbf{X}^*, \mathbf{D}^{k+1} - \mathbf{D}^*\rangle} \\
=&\|\mathbf{X}^k - \mathbf{X}^*\|^2 - \|\mathbf{X}^{k+1} - \mathbf{X}^k\|^2 - \|\mathbf{X}^{k+1} - \mathbf{X}^*\|^2 \\
&+ \eta^2 \|\nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \nabla \mathbf{F}(\mathbf{X}^*)\|^2 + \|\mathbf{X}^k - \mathbf{X}^{k+1}\|^2 - \eta^2 \|\mathbf{D}^{k+1} - \mathbf{D}^*\|^2 \\
&+ \frac{\eta^2}{\gamma} \underbrace{\left(\|\mathbf{D}^k - \mathbf{D}^*\|_{\mathbf{M}}^2 - \|\mathbf{D}^{k+1} - \mathbf{D}^*\|_{\mathbf{M}}^2 - \|\mathbf{D}^{k+1} - \mathbf{D}^k\|_{\mathbf{M}}^2\right)}_{-2\langle \mathbf{D}^{k+1} - \mathbf{D}^k, \mathbf{D}^{k+1} - \mathbf{D}^*\rangle_{\mathbf{M}}} + 2\eta\langle \mathbf{E}^k, \mathbf{D}^{k+1} - \mathbf{D}^*\rangle,
\end{aligned}$$

where the last equality holds because

$$2\langle \mathbf{D}^k - \mathbf{D}^{k+1}, \mathbf{D}^{k+1} - \mathbf{D}^*\rangle_{\mathbf{M}} = \|\mathbf{D}^k - \mathbf{D}^*\|_{\mathbf{M}}^2 - \|\mathbf{D}^{k+1} - \mathbf{D}^*\|_{\mathbf{M}}^2 - \|\mathbf{D}^{k+1} - \mathbf{D}^k\|_{\mathbf{M}}^2.$$

Thus, we reformulate it as

$$\begin{aligned}
&\|\mathbf{X}^{k+1} - \mathbf{X}^*\|^2 + \frac{\eta^2}{\gamma}\|\mathbf{D}^{k+1} - \mathbf{D}^*\|_{\mathbf{M}}^2 \\
=&\|\mathbf{X}^k - \mathbf{X}^*\|^2 + \frac{\eta^2}{\gamma}\|\mathbf{D}^k - \mathbf{D}^*\|_{\mathbf{M}}^2 - \frac{\eta^2}{\gamma}\|\mathbf{D}^{k+1} - \mathbf{D}^k\|_{\mathbf{M}}^2 - \eta^2 \|\mathbf{D}^{k+1} - \mathbf{D}^*\|^2 \\
&- 2\eta\langle \mathbf{X}^k - \mathbf{X}^*, \nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \nabla \mathbf{F}(\mathbf{X}^*)\rangle + \eta^2 \|\nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \nabla \mathbf{F}(\mathbf{X}^*)\|^2 + 2\eta\langle \mathbf{E}^k, \mathbf{D}^{k+1} - \mathbf{D}^*\rangle,
\end{aligned}$$

which completes the proof. □

B.3.4 Proof of Lemma 11

Proof of Lemma 11. From Alg. 3, we take the expectation conditioned on kth compression and obtain

$$\begin{aligned}
&\mathbb{E}\|\mathbf{H}^{k+1}-\mathbf{X}^*\|^2\\
=&\mathbb{E}\|(1-\alpha)(\mathbf{H}^k-\mathbf{X}^*)+\alpha(\mathbf{Y}^k-\mathbf{X}^*)+\alpha\mathbf{E}^k\|^2 \qquad \text{(from Line 13)}\\
=&\|(1-\alpha)(\mathbf{H}^k-\mathbf{X}^*)+\alpha(\mathbf{Y}^k-\mathbf{X}^*)\|^2+\alpha^2\mathbb{E}\|\mathbf{E}^k\|^2\\
=&(1-\alpha)\|\mathbf{H}^k-\mathbf{X}^*\|^2+\alpha\|\mathbf{Y}^k-\mathbf{X}^*\|^2-\alpha(1-\alpha)\|\mathbf{H}^k-\mathbf{Y}^k\|^2+\alpha^2\mathbb{E}\|\mathbf{E}^k\|^2. \qquad \text{(B.9)}
\end{aligned}$$

In the second equality, we used the unbiasedness of the compression, i.e., $\mathbb{E}\mathbf{E}^k=\mathbf{0}$. The last equality holds because of

$$\|(1-\alpha)\mathbf{A}+\alpha\mathbf{B}\|^2=(1-\alpha)\|\mathbf{A}\|^2+\alpha\|\mathbf{B}\|^2-\alpha(1-\alpha)\|\mathbf{A}-\mathbf{B}\|^2.$$

In addition, by taking the conditional expectation on the compression, we have

$$\begin{aligned}
\|\mathbf{Y}^k-\mathbf{X}^*\|^2=&\|\mathbf{X}^k-\eta\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)-\eta\mathbf{D}^k-\mathbf{X}^*\|^2 \qquad \text{(from Line 4)}\\
=&\mathbb{E}\|\mathbf{X}^{k+1}+\eta\mathbf{D}^{k+1}-\eta\mathbf{D}^k-\mathbf{X}^*\|^2 \qquad \text{(from Line 7)}\\
=&\mathbb{E}\|\mathbf{X}^{k+1}-\mathbf{X}^*\|^2+\eta^2\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|^2+2\eta\mathbb{E}\langle\mathbf{X}^{k+1}-\mathbf{X}^*,\mathbf{D}^{k+1}-\mathbf{D}^k\rangle\\
=&\mathbb{E}\|\mathbf{X}^{k+1}-\mathbf{X}^*\|^2+\eta^2\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|^2\\
&+\frac{2\eta^2}{\gamma}\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|_{\mathbf{M}}^2-2\eta\mathbb{E}\langle\mathbf{E}^k,\mathbf{D}^{k+1}-\mathbf{D}^k\rangle. \qquad \text{(from (B.4))}\\
=&\mathbb{E}\|\mathbf{X}^{k+1}-\mathbf{X}^*\|^2+\eta^2\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|^2\\
&+\frac{2\eta^2}{\gamma}\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|_{\mathbf{M}}^2-\gamma\mathbb{E}\|\mathbf{E}^k\|_{\mathbf{I}-\mathbf{W}}^2. \qquad \text{(from Line 6)} \qquad \text{(B.10)}
\end{aligned}$$

Combing the above two equations (B.9) and (B.10) together, we have

$$\begin{aligned}
&\mathbb{E}\|\mathbf{H}^{k+1}-\mathbf{X}^*\|^2\\
\le&(1-\alpha)\|\mathbf{H}^k-\mathbf{X}^*\|^2+\alpha\mathbb{E}\|\mathbf{X}^{k+1}-\mathbf{X}^*\|^2+\alpha\eta^2\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|^2+\frac{2\alpha\eta^2}{\gamma}\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|_{\mathbf{M}}^2\\
&-\alpha\gamma\mathbb{E}\|\mathbf{E}^k\|_{\mathbf{I}-\mathbf{W}}^2+\alpha^2\mathbb{E}\|\mathbf{E}^k\|^2-\alpha(1-\alpha)\|\mathbf{Y}^k-\mathbf{H}^k\|^2, \qquad \text{(B.11)}
\end{aligned}$$

which completes the proof. □

B.3.5 Proof of Theorem 3

Proof of Theorem 3. Combining Lemmas 10, 11, and 14, we have the expectation conditioned on the compression satisfying

$$
\begin{aligned}
&\mathbb{E}\|\mathbf{X}^{k+1}-\mathbf{X}^*\|^2+\frac{\eta^2}{\gamma}\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^*\|_{\mathbf{M}}^2+a_1\mathbb{E}\|\mathbf{H}^{k+1}-\mathbf{X}^*\|^2\\
\leq&\|\mathbf{X}^k-\mathbf{X}^*\|^2+\frac{\eta^2}{\gamma}\|\mathbf{D}^k-\mathbf{D}^*\|_{\mathbf{M}}^2-\frac{\eta^2}{\gamma}\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|_{\mathbf{M}}^2-\eta^2\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^*\|^2\\
&-2\eta\langle\mathbf{X}^k-\mathbf{X}^*,\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)-\nabla\mathbf{F}(\mathbf{X}^*)\rangle+\eta^2\|\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)-\nabla\mathbf{F}(\mathbf{X}^*)\|^2+\gamma\mathbb{E}\|\mathbf{E}^k\|_{\mathbf{I}-\mathbf{W}}^2\\
&+a_1(1-\alpha)\|\mathbf{H}^k-\mathbf{X}^*\|^2+a_1\alpha\mathbb{E}\|\mathbf{X}^{k+1}-\mathbf{X}^*\|^2+a_1\alpha\eta^2\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|^2\\
&+\frac{2a_1\alpha\eta^2}{\gamma}\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|_{\mathbf{M}}^2+a_1\alpha^2\mathbb{E}\|\mathbf{E}^k\|^2-a_1\alpha\gamma\mathbb{E}\|\mathbf{E}^k\|_{\mathbf{I}-\mathbf{W}}^2-a_1\alpha(1-\alpha)\|\mathbf{Y}^k-\mathbf{H}^k\|^2\\
=&\underbrace{\|\mathbf{X}^k-\mathbf{X}^*\|^2-2\eta\langle\mathbf{X}^k-\mathbf{X}^*,\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)-\nabla\mathbf{F}(\mathbf{X}^*)\rangle+\eta^2\|\nabla\mathbf{F}(\mathbf{X}^k;\xi^k)-\nabla\mathbf{F}(\mathbf{X}^*)\|^2}_{\mathcal{A}}\\
&+a_1\alpha\mathbb{E}\|\mathbf{X}^{k+1}-\mathbf{X}^*\|^2+\frac{\eta^2}{\gamma}\|\mathbf{D}^k-\mathbf{D}^*\|_{\mathbf{M}}^2-\eta^2\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^*\|^2\\
&+a_1(1-\alpha)\|\mathbf{H}^k-\mathbf{X}^*\|^2\underbrace{-(1-2a_1\alpha)\frac{\eta^2}{\gamma}\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|_{\mathbf{M}}^2+a_1\alpha\eta^2\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|^2}_{\mathcal{B}}\\
&+\underbrace{a_1\alpha^2\mathbb{E}\|\mathbf{E}^k\|^2+(1-a_1\alpha)\gamma\mathbb{E}\|\mathbf{E}^k\|_{\mathbf{I}-\mathbf{W}}^2-a_1\alpha(1-\alpha)\|\mathbf{Y}^k-\mathbf{H}^k\|^2}_{C},
\end{aligned}
\tag{B.12}
$$

where a_1 is a non-negative number to be determined. Then we deal with the three terms on the right hand side separately. We want the terms $\mathcal{B}$ and C to be nonpositive. First, we consider $\mathcal{B}$. Note that $\mathbf{D}^k\in\mathbf{Range}(\mathbf{I}-\mathbf{W})$. If we want $\mathcal{B}\leq 0$, then, we need $1-2a_1\alpha>0$, i.e., $a_1\alpha<1/2$. Therefore we have

$$
\begin{aligned}
\mathcal{B}=&-(1-2a_1\alpha)\frac{\eta^2}{\gamma}\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|_{\mathbf{M}}^2+a_1\alpha\eta^2\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|^2\\
\leq&\left(a_1\alpha-\frac{(1-2a_1\alpha)\lambda_{n-1}(\mathbf{M})}{\gamma}\right)\eta^2\mathbb{E}\|\mathbf{D}^{k+1}-\mathbf{D}^k\|^2,
\end{aligned}
$$

where $\lambda_{n-1}(\mathbf{M})>0$ is the second smallest eigenvalue of $\mathbf{M}$. It means that we also need

$$
a_1\alpha+\frac{(2a_1\alpha-1)\lambda_{n-1}(\mathbf{M})}{\gamma}\leq 0,
$$

which is equivalent to

$$a_1\alpha \leq \frac{\lambda_{n-1}(\mathbf{M})}{\gamma + 2\lambda_{n-1}(\mathbf{M})} < 1/2. \tag{B.13}$$

Then we look at C. We have

$$\begin{aligned} C =& a_1\alpha^2\mathbb{E}\|\mathbf{E}^k\|^2 + (1-a_1\alpha)\gamma\mathbb{E}\|\mathbf{E}^k\|^2_{\mathbf{I}-\mathbf{W}} - a_1\alpha(1-\alpha)\|\mathbf{Y}^k - \mathbf{H}^k\|^2 \\ \leq& ((1-a_1\alpha)\beta\gamma + a_1\alpha^2)\mathbb{E}\|\mathbf{E}^k\|^2 - a_1\alpha(1-\alpha)\|\mathbf{Y}^k - \mathbf{H}^k\|^2 \\ \leq& C((1-a_1\alpha)\beta\gamma + a_1\alpha^2)\|\mathbf{Y}^k - \mathbf{H}^k\|^2 - a_1\alpha(1-\alpha)\|\mathbf{Y}^k - \mathbf{H}^k\|^2 \end{aligned}$$

Because we have $1 - a_1\alpha > 1/2$, so we need

$$C((1-a_1\alpha)\beta\gamma + a_1\alpha^2) - a_1\alpha(1-\alpha) = (1+C)a_1\alpha^2 - a_1(C\beta\gamma+1)\alpha + C\beta\gamma \leq 0. \tag{B.14}$$

That is

$$\alpha \geq \frac{a_1(C\beta\gamma+1) - \sqrt{a_1^2(C\beta\gamma+1)^2 - 4(1+C)Ca_1\beta\gamma}}{2(1+C)a_1} =: \alpha_0, \tag{B.15}$$

$$\alpha \leq \frac{a_1(C\beta\gamma+1) + \sqrt{a_1^2(C\beta\gamma+1)^2 - 4(1+C)Ca_1\beta\gamma}}{2(1+C)a_1} =: \alpha_1. \tag{B.16}$$

Next, we look at $\mathcal{A}$. Firstly, by the bounded variance assumption, we have the expectation conditioned on the gradient sampling in kth iteration satisfying

$$\begin{aligned} &\mathbb{E}\|\mathbf{X}^k - \mathbf{X}^*\|^2 - 2\eta\mathbb{E}\langle\mathbf{X}^k - \mathbf{X}^*, \nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \nabla\mathbf{F}(\mathbf{X}^*)\rangle + \eta^2\mathbb{E}\|\nabla\mathbf{F}(\mathbf{X}^k;\xi^k) - \nabla\mathbf{F}(\mathbf{X}^*)\|^2 \\ \leq& \|\mathbf{X}^k - \mathbf{X}^*\|^2 - 2\eta\langle\mathbf{X}^k - \mathbf{X}^*, \nabla\mathbf{F}(\mathbf{X}^k) - \nabla\mathbf{F}(\mathbf{X}^*)\rangle + \eta^2\|\nabla\mathbf{F}(\mathbf{X}^k) - \nabla\mathbf{F}(\mathbf{X}^*)\|^2 + n\eta^2\sigma^2 \end{aligned}$$

Then with the smoothness and strong convexity from Assumptions 8, we have the co-coercivity of $\nabla g_i(\mathbf{x})$ with $g_i(\mathbf{x}) := f_i(\mathbf{x}) - \frac{u}{2}\|\mathbf{x}\|_2^2$, which gives

$$\langle\mathbf{X}^k - \mathbf{X}^*, \nabla\mathbf{F}(\mathbf{X}^k) - \nabla\mathbf{F}(\mathbf{X}^*)\rangle \geq \frac{\mu L}{\mu+L}\|\mathbf{X}^k - \mathbf{X}^*\|^2 + \frac{1}{\mu+L}\|\nabla\mathbf{F}(\mathbf{X}^k) - \nabla\mathbf{F}(\mathbf{X}^*)\|^2.$$

When $\eta \leq 2/(\mu+L)$, we have

$$\langle\mathbf{X}^k - \mathbf{X}^*, \nabla\mathbf{F}(\mathbf{X}^k) - \nabla\mathbf{F}(\mathbf{X}^*)\rangle$$

$$
\begin{aligned}
&= \left(1 - \frac{\eta(\mu+L)}{2}\right) \langle \mathbf{X}^k - \mathbf{X}^*, \nabla \mathbf{F}(\mathbf{X}^k) - \nabla \mathbf{F}(\mathbf{X}^*)\rangle + \frac{\eta(\mu+L)}{2} \langle \mathbf{X}^k - \mathbf{X}^*, \nabla \mathbf{F}(\mathbf{X}^k) - \nabla \mathbf{F}(\mathbf{X}^*)\rangle \\
&\geq \left(\mu - \frac{\eta\mu(\mu+L)}{2} + \frac{\eta\mu L}{2}\right) \|\mathbf{X}^k - \mathbf{X}^*\|^2 + \frac{\eta}{2} \|\nabla \mathbf{F}(\mathbf{X}^k) - \nabla \mathbf{F}(\mathbf{X}^*)\|^2 \\
&= \mu \left(1 - \frac{\eta\mu}{2}\right) \|\mathbf{X}^k - \mathbf{X}^*\|^2 + \frac{\eta}{2} \|\nabla \mathbf{F}(\mathbf{X}^k) - \nabla \mathbf{F}(\mathbf{X}^*)\|^2.
\end{aligned}
$$

Therefore, we obtain

$$
\begin{aligned}
&- 2\eta \langle \mathbf{X}^k - \mathbf{X}^*, \nabla \mathbf{F}(\mathbf{X}^k) - \nabla \mathbf{F}(\mathbf{X}^*)\rangle \\
&\leq - \eta^2 \|\nabla \mathbf{F}(\mathbf{X}^k) - \nabla \mathbf{F}(\mathbf{X}^*)\|^2 - \mu(2\eta - \mu\eta^2) \|\mathbf{X}^k - \mathbf{X}^*\|^2. \qquad \text{(B.17)}
\end{aligned}
$$

Conditioned on the $k$the iteration, (i.e., conditioned on the gradient sampling in kth iteration), the inequality (B.12) becomes

$$
\begin{aligned}
&\mathbb{E}\|\mathbf{X}^{k+1} - \mathbf{X}^*\|^2 + \frac{\eta^2}{\gamma} \mathbb{E}\|\mathbf{D}^{k+1} - \mathbf{D}^*\|_{\mathbf{M}}^2 + a_1 \mathbb{E}\|\mathbf{H}^{k+1} - \mathbf{X}^*\|^2 \\
&\leq \left(1 - \mu(2\eta - \mu\eta^2)\right) \|\mathbf{X}^k - \mathbf{X}^*\|^2 + a_1\alpha \mathbb{E}\|\mathbf{X}^{k+1} - \mathbf{X}^*\|^2 \\
&\quad + \frac{\eta^2}{\gamma} \|\mathbf{D}^k - \mathbf{D}^*\|_{\mathbf{M}}^2 - \eta^2 \mathbb{E}\|\mathbf{D}^{k+1} - \mathbf{D}^*\|^2 + a_1(1-\alpha)\|\mathbf{H}^k - \mathbf{X}^*\|^2 + n\eta^2\sigma^2, \qquad \text{(B.18)}
\end{aligned}
$$

if the step size satisfies $\eta \leq \frac{2}{\mu+L}$. Rewriting (B.18), we have

$$
\begin{aligned}
&(1 - a_1\alpha)\mathbb{E}\|\mathbf{X}^{k+1} - \mathbf{X}^*\|^2 + \frac{\eta^2}{\gamma} \mathbb{E}\|\mathbf{D}^{k+1} - \mathbf{D}^*\|_{\mathbf{M}}^2 + \eta^2 \mathbb{E}\|\mathbf{D}^{k+1} - \mathbf{D}^*\|^2 + a_1 \mathbb{E}\|\mathbf{H}^{k+1} - \mathbf{X}^*\|^2 \\
&\leq \left(1 - \mu(2\eta - \mu\eta^2)\right) \|\mathbf{X}^k - \mathbf{X}^*\|^2 + \frac{\eta^2}{\gamma} \|\mathbf{D}^k - \mathbf{D}^*\|_{\mathbf{M}}^2 + a_1(1-\alpha)\|\mathbf{H}^k - \mathbf{X}^*\|^2 + n\eta^2\sigma^2, \qquad \text{(B.19)}
\end{aligned}
$$

and thus

$$
\begin{aligned}
&(1 - a_1\alpha)\mathbb{E}\|\mathbf{X}^{k+1} - \mathbf{X}^*\|^2 + \frac{\eta^2}{\gamma} \mathbb{E}\|\mathbf{D}^{k+1} - \mathbf{D}^*\|_{\mathbf{M}+\gamma\mathbf{I}}^2 + a_1 \mathbb{E}\|\mathbf{H}^{k+1} - \mathbf{X}^*\|^2 \\
&\leq \left(1 - \mu(2\eta - \mu\eta^2)\right) \|\mathbf{X}^k - \mathbf{X}^*\|^2 + \frac{\eta^2}{\gamma} \|\mathbf{D}^k - \mathbf{D}^*\|_{\mathbf{M}}^2 + a_1(1-\alpha)\|\mathbf{H}^k - \mathbf{X}^*\|^2 + n\eta^2\sigma^2. \qquad \text{(B.20)}
\end{aligned}
$$

With the definition of $\mathcal{L}^k$ in (3.17), we have

$$
\mathbb{E}\mathcal{L}^{k+1} \leq \rho \mathcal{L}^k + n\eta^2\sigma^2, \qquad \text{(B.21)}
$$

with

$$
\rho = \max\left\{\frac{1 - \mu(2\eta - \mu\eta^2)}{1 - a_1\alpha}, \frac{\lambda_{\max}(\mathbf{M})}{\gamma + \lambda_{\max}(\mathbf{M})}, 1 - \alpha\right\}.
$$

where

$$\lambda_{\max}(\mathbf{M}) = 2\lambda_{\max}((\mathbf{I}-\mathbf{W})^{\dagger}) - \gamma.$$

Recall all the conditions on the parameters $a_1,\ \alpha$, and γ to make sure that $\rho < 1$:

$$a_1\alpha \le \frac{\lambda_{n-1}(\mathbf{M})}{\gamma + 2\lambda_{n-1}(\mathbf{M})}, \tag{B.22}$$

$$a_1\alpha \le \mu(2\eta - \mu\eta^2), \tag{B.23}$$

$$\alpha \ge \frac{a_1(C\beta\gamma+1) - \sqrt{a_1^2(C\beta\gamma+1)^2 - 4(1+C)Ca_1\beta\gamma}}{2(1+C)a_1} =: \alpha_0, \tag{B.24}$$

$$\alpha \le \frac{a_1(C\beta\gamma+1) + \sqrt{a_1^2(C\beta\gamma+1)^2 - 4(1+C)Ca_1\beta\gamma}}{2(1+C)a_1} =: \alpha_1. \tag{B.25}$$

In the following, we show that there exist parameters that satisfy these conditions.

Since we can choose any a_1, we let

$$a_1 = \frac{4(1+C)}{C\beta\gamma+2},$$

such that

$$a_1^2(C\beta\gamma+1)^2 - 4(1+C)Ca_1\beta\gamma = a_1^2.$$

Then we have

$$\begin{aligned}
\alpha_0 =& \frac{C\beta\gamma}{2(1+C)} \to 0, \qquad \text{as } \gamma \to 0,\\
\alpha_1 =& \frac{C\beta\gamma+2}{2(1+C)} \to \frac{1}{1+C}, \quad \text{as } \gamma \to 0.
\end{aligned}$$

Conditions (B.24) and (B.25) show

$$a_1\alpha \in \left[\frac{2C\beta\gamma}{C\beta\gamma+2}, 2\right] \to [0,2], \text{ if } C = 0 \text{ or } \gamma \to 0.$$

Hence in order to make (B.22) and (B.23) satisfied, it's sufficient to make

$$\frac{2C\beta\gamma}{C\beta\gamma+2} \le \min\left\{\frac{\lambda_{n-1}(\mathbf{M})}{\gamma+2\lambda_{n-1}(\mathbf{M})}, \mu(2\eta-\mu\eta^2)\right\} = \min\left\{\frac{\frac{2}{\beta}-\gamma}{\frac{4}{\beta}-\gamma}, \mu(2\eta-\mu\eta^2)\right\}. \tag{B.26}$$

where we use $\lambda_{n-1}(\mathbf{M}) = \frac{2}{\lambda_{\max}(\mathbf{I}-\mathbf{W})} - \gamma = \frac{2}{\beta} - \gamma$.

When $C > 0$, the condition (B.26) is equivalent to

$$\gamma \leq \min\left\{\frac{(3C+1)-\sqrt{(3C+1)^2-4C}}{C\beta}, \frac{2\mu\eta(2-\mu\eta)}{[2-\mu\eta(2-\mu\eta)]C\beta}\right\}. \tag{B.27}$$

The first term can be simplified using

$$\frac{(3C+1)-\sqrt{(3C+1)^2-4C}}{C\beta} \geq \frac{2}{(3C+1)\beta}$$

due to $\sqrt{1-x} \leq 1-\frac{x}{2}$ when $x \in (0,1)$.

Therefore, for a given stepsize η, if we choose

$$\gamma \in \left(0, \min\left\{\frac{2}{(3C+1)\beta}, \frac{2\mu\eta(2-\mu\eta)}{[2-\mu\eta(2-\mu\eta)]C\beta}\right\}\right)$$

and

$$\alpha \in \left[\frac{C\beta\gamma}{2(1+C)}, \min\left\{\frac{C\beta\gamma+2}{2(1+C)}, \frac{2-\beta\gamma}{4-\beta\gamma}\frac{C\beta\gamma+2}{4(1+C)}, \mu\eta(2-\mu\eta)\frac{C\beta\gamma+2}{4(1+C)}\right\}\right],$$

then, all conditions (B.22)-(B.25) hold.

Note that $\gamma < \frac{2}{(3C+1)\beta}$ implies $\gamma < \frac{2}{\beta}$, which ensures the positive definiteness of $\mathbf{M}$ over $\mathbf{span}\{\mathbf{I}-\mathbf{W}\}$ in Lemma 13.

Note that $\eta \leq \frac{2}{\mu+L}$ ensures

$$\mu\eta(2-\mu\eta)\frac{C\beta\gamma+2}{4(1+C)} \leq \frac{C\beta\gamma+2}{2(1+C)}. \tag{B.28}$$

So, we can simplify the bound for α as

$$\alpha \in \left[\frac{C\beta\gamma}{2(1+C)}, \min\left\{\frac{2-\beta\gamma}{4-\beta\gamma}\frac{C\beta\gamma+2}{4(1+C)}, \mu\eta(2-\mu\eta)\frac{C\beta\gamma+2}{4(1+C)}\right\}\right].$$

Lastly, taking the total expectation on both sides of (B.21) and using tower property, we complete the proof for $C > 0$.

□

Proof of Corollary 4. Let's first define $\kappa_f = \frac{L}{\mu}$ and $\kappa_g = \frac{\lambda_{\max}(\mathbf{I}-\mathbf{W})}{\lambda^+_{\min}(\mathbf{I}-\mathbf{W})} = \lambda_{\max}(\mathbf{I}-\mathbf{W})\lambda_{\max}((\mathbf{I}-\mathbf{W})^\dagger)$.

We can choose the stepsize $\eta = \frac{1}{L}$ such that the upper bound of γ is

$$\gamma_{\text{upper}} = \min\left\{\frac{2}{(3C+1)\beta}, \frac{\frac{2}{\kappa_f}\left(2-\frac{1}{\kappa_f}\right)}{\left[2-\frac{1}{\kappa_f}\left(2-\frac{1}{\kappa_f}\right)\right]C\beta}, \frac{2}{\beta}\right\} \geq \min\left\{\frac{2}{(3C+1)\beta}, \frac{1}{\kappa_f C\beta}\right\},$$

due to $\frac{x(2-x)}{2-x(2-x)} \geq \frac{x}{2-x} \geq x$ when $x \in (0,1)$.

Hence we can take $\gamma = \min\{\frac{1}{(3C+1)\beta}, \frac{1}{\kappa_f C\beta}\}$.

The bound of α is

$$\alpha \in \left[\frac{C\beta\gamma}{2(1+C)}, \min\left\{\frac{2-\beta\gamma}{4-\beta\gamma}\frac{C\beta\gamma+2}{4(1+C)}, \frac{1}{\kappa_f}(2-\frac{1}{\kappa_f})\frac{C\beta\gamma+2}{4(1+C)}\right\}\right]$$

When γ is chosen as $\frac{1}{\kappa_f C\beta}$, pick

$$\alpha = \frac{C\beta\gamma}{2(1+C)} = \frac{1}{2(1+C)\kappa_f}. \tag{B.29}$$

When $\frac{1}{(3C+1)\beta} \leq \frac{1}{\kappa_f C\beta}$, the upper bound of α is

$$\begin{aligned}
\alpha_{\text{upper}} &= \min\left\{\frac{2-\beta\gamma}{4-\beta\gamma}\frac{C\beta\gamma+2}{4(1+C)}, \frac{1}{\kappa_f}(2-\frac{1}{\kappa_f})\frac{C\beta\gamma+2}{4(1+C)}\right\} \\
&= \min\left\{\frac{6C+1}{12C+3}, \frac{1}{\kappa_f}(2-\frac{1}{\kappa_f})\right\}\frac{7C+2}{4(C+1)(3C+1)} \\
&\geq \min\left\{\frac{6C+1}{12C+3}, \frac{1}{\kappa_f}\right\}\frac{7C+2}{4(C+1)(3C+1)}.
\end{aligned}$$

In this case, we pick

$$\alpha = \min\left\{\frac{6C+1}{12C+3}, \frac{1}{\kappa_f}\right\}\frac{7C+2}{4(C+1)(3C+1)}. \tag{B.30}$$

Note $\alpha = O\left(\frac{1}{(1+C)\kappa_f}\right)$ since $\frac{6C+1}{12C+3}$ is lower bounded by $\frac{1}{3}$. Hence in both cases (Eq. (B.29) and Eq. (B.30)), $\alpha = O\left(\frac{1}{(1+C)\kappa_f}\right)$, and the third term of ρ is upper bounded by

$$1-\alpha \leq \max\left\{1-\frac{1}{2(1+C)\kappa_f}, 1-\min\left\{\frac{6C+1}{12C+3}, \frac{1}{\kappa_f}\right\}\frac{7C+2}{4(1+C)(3C+1)}\right\}$$

In two cases of γ, the second term of ρ becomes

$$1-\frac{\gamma}{2\lambda_{\max}((\mathbf{I}-\mathbf{W})^{\dagger})} = \max\left\{1-\frac{1}{2C\kappa_f\kappa_g}, 1-\frac{1}{(1+3C)\kappa_g}\right\}$$

Before analysing the first term of ρ, we look at $a_1\alpha$ in two cases of γ.

When $\gamma = \frac{1}{\kappa_f C\beta}$, we have

$$a_1\alpha = \frac{2C\beta\gamma}{C\beta\gamma + 2} = \frac{2}{2\kappa_f + 1} \leq \frac{1}{\kappa_f}.$$

When $\gamma = \frac{1}{(3C+1)\beta}$, we have

$$a_1\alpha = \min\left\{\frac{6C+1}{(12C+3)}, \frac{1}{\kappa_f}\right\} \leq \frac{1}{\kappa_f}.$$

In both cases, $a_1\alpha \leq \frac{1}{\kappa_f}$. Therefore, the first term of ρ becomes

$$\frac{1-\mu\eta(2-\mu\eta)}{1-a_1\alpha} \leq \frac{1-\frac{1}{\kappa_f}(2-\frac{1}{\kappa_f})}{1-\frac{1}{\kappa_f}} = 1 - \frac{1-\frac{1}{\kappa_f}}{\kappa_f - 1} = 1 - \frac{1}{\kappa_f}.$$

To summarize, we have

$$\rho \leq 1 - \min\left\{\frac{1}{\kappa_f}, \frac{1}{2C\kappa_f\kappa_g}, \frac{1}{(1+3C)\kappa_g}, \frac{1}{2(1+C)\kappa_f}, \min\left\{\frac{6C+1}{12C+3}, \frac{1}{\kappa_f}\right\}\frac{7C+2}{4(1+C)(3C+1)}\right\}$$

and therefore

$$\rho = \max\left\{1 - O\Big(\frac{1}{(1+C)\kappa_f}\Big), 1 - O\Big(\frac{1}{(1+C)\kappa_g}\Big), 1 - O\Big(\frac{1}{C\kappa_f\kappa_g}\Big)\right\}.$$

With full-gradient (i.e., $\sigma = 0$), we get $\epsilon-$accuracy solution with the total number of iterations

$$k \geq \widetilde{O}((1+C)(\kappa_f + \kappa_g) + C\kappa_f\kappa_g).$$

When $C = 0$, i.e., there is no compression, the iteration complexity recovers that of NIDS, $\widetilde{O}\left(\kappa_f + \kappa_g\right)$.

When $C \leq \frac{\kappa_f+\kappa_g}{\kappa_f\kappa_g+\kappa_f+\kappa_g}$, the complexity is improved to that of NIDS, i.e., the compression doesn't harm the convergence in terms of the order of the coefficients. □

Proof of Corollary 5. Note that $(\bar{\mathbf{x}}^k)^\top = \overline{\mathbf{X}}^k$ and $\mathbf{1}_{n\times 1}\overline{\mathbf{X}}^* = \mathbf{X}^*$, then

$$\begin{aligned}
\sum_{i=1}^{n} \mathbb{E}\|\mathbf{x}_i^k - \bar{\mathbf{x}}^k\|^2 &= \mathbb{E}\left\|\mathbf{X}^k - \mathbf{1}_{n\times 1}\overline{\mathbf{X}}^k\right\|^2 \\
&= \mathbb{E}\left\|\mathbf{X}^k - \mathbf{X}^* + \mathbf{X}^* - \mathbf{1}_{n\times 1}\overline{\mathbf{X}}^k\right\|^2 \\
&= \mathbb{E}\left\|\mathbf{X}^k - \mathbf{X}^* - \frac{\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^\top}{n}\left(\mathbf{X}^k - \mathbf{X}^*\right)\right\| \\
&\leq \mathbb{E}\|\mathbf{X}^k - \mathbf{X}^*\|^2 \\
&\leq \frac{\rho\mathbb{E}\mathcal{L}^{k-1} + n\eta^2\sigma^2(1-\rho)^{-1}}{1 - a_1\alpha} \\
&\leq 2\rho^k\mathcal{L}^0 + 2\frac{n\eta^2\sigma^2}{1-\rho}. \qquad \text{(B.31)}
\end{aligned}$$

The last inequality holds because we have $a_1\alpha \leq 1/2$. □

Proof of Corollary 3. From the proof of Theorem 3, when $C = 0$, we can set $\gamma = 1$, $\alpha = 1$, and $a_1 = 0$. Plug those values into ρ, and we obtain the convergence rate for NIDS. □

B.3.6 Proof of Theorem 4

Proof of Theorem 4. In order to get exact convergence, we pick diminishing step-size, set $\alpha = \frac{C\beta\gamma}{2(1+C)}$, $a_1\alpha = \frac{2C\beta\gamma_k}{C\beta\gamma_k+2}$, $\theta_1 = \frac{1}{2\lambda_{\max}((\mathbf{I}-\mathbf{W})^\dagger)}$ and $\theta_2 = \frac{C\beta}{2(1+C)}$, then

$$\rho_k = \max\left\{1 - \frac{\mu\eta_k(2-\mu\eta_k) - a_1\alpha}{1-a_1\alpha}, 1-\theta_1\gamma_k, 1-\theta_2\gamma_k\right\}$$

If we further pick diminishing η_k and γ_k such that $\mu\eta_k(2-\mu\eta_k) - a_1\alpha \geq a_1\alpha$, then

$$\frac{\mu\eta_k(2-\mu\eta_k) - a_1\alpha}{1-a_1\alpha} \geq \frac{a_1\alpha}{1-a_1\alpha} = \frac{2C\beta\gamma_k}{2 - C\beta\gamma_k} \geq C\beta\gamma_k.$$

Notice that $C\beta\gamma \leq \frac{2}{3}$ since $(3C+1) - \sqrt{(3C+1)^2 - 4C}$ is increasing in $C > 0$ with limit $\frac{2}{3}$ at ∞.

In this case we only need,

$$\gamma_k \in \left(0, \min\left\{\frac{(3C+1) - \sqrt{(3C+1)^2 - 4C}}{C\beta}, \frac{2\mu\eta_k(2-\mu\eta_k)}{[4-\mu\eta_k(2-\mu\eta_k)]C\beta}, \frac{2}{\beta}\right\}\right). \qquad \text{(B.32)}$$

And

$$\rho_k \leq \max\{1 - C\beta\gamma_k, 1-\theta_1\gamma_k, 1-\theta_2\gamma_k\} \leq 1 - \theta_3\gamma_k$$

if $\theta_3 = \min\{\theta_1, \theta_2\}$ and note that $\theta_2 \leq C\beta$.

We define

$$\mathcal{L}^k \coloneqq (1 - a_1\alpha_k)\|\mathbf{X}^k - \mathbf{X}^*\|^2 + (2\eta_k^2/\gamma_k)\mathbb{E}\|\mathbf{D}^{k+1} - \mathbf{D}^*\|^2_{(\mathbf{I}-\mathbf{W})^\dagger} + a_1\|\mathbf{H}^k - \mathbf{X}^*\|^2.$$

Hence

$$\mathbb{E}\mathcal{L}^{k+1} \leq (1 - \theta_3\gamma_k)\mathbb{E}\mathcal{L}^k + n\sigma^2\eta_k^2.$$

From $a_1\alpha \leq \frac{\mu\eta_k(2-\mu\eta_k)}{2}$, we get

$$\frac{4C\beta\gamma_k}{C\beta\gamma_k + 2} \leq \mu\eta_k(2 - \mu\eta_k).$$

If we pick $\gamma_k = \theta_4\eta_k$, then it's sufficient to let

$$2C\beta\theta_4\eta_k \leq \mu\eta_k(2 - \mu\eta_k).$$

Hence if $\theta_4 < \frac{\mu}{C\beta}$ and let $\eta_* = \frac{2(\mu - C\beta\theta_4)}{\mu^2}$, then $\eta_k = \frac{\gamma_k}{\theta_4} \in (0, \eta_*)$ guarantees the above discussion and

$$\mathbb{E}\mathcal{L}^{k+1} \leq (1 - \theta_3\theta_4\eta_k)\mathbb{E}\mathcal{L}^k + n\sigma^2\eta_k^2.$$

So far all restrictions for η_k are

$$\eta_k \leq \min\left\{\frac{2}{\mu + L}, \eta_*\right\}$$

and

$$\eta_k \leq \frac{1}{\theta_4}\min\left\{\frac{(3C+1) - \sqrt{(3C+1)^2 - 4C}}{C\beta}, \frac{2}{\beta}\right\}$$

Let $\theta_5 = \min\left\{\frac{2}{\mu+L}, \eta_*, \frac{(3C+1)-\sqrt{(3C+1)^2-4C}}{C\beta\theta_4}, \frac{2}{\beta\theta_4}\right\}$, $\eta_k = \frac{1}{Bk+A}$ and $D = \max\left\{A\mathcal{L}^0, \frac{2n\sigma^2}{\theta_3\theta_4}\right\}$, we claim that if we pick $B = \frac{\theta_3\theta_4}{2}$ and some A, by setting $\eta_k = \frac{2}{\theta_3\theta_4 k + 2A}$, we get

$$\mathbb{E}\mathcal{L}^k \leq \frac{D}{Bk + A}.$$

Induction:

When $k = 0$, it's obvious. Suppose previous k inequalities hold. Then

$$\mathbb{E}\mathcal{L}^{k+1} \leq \left(1 - \frac{2\theta_3\theta_4}{\theta_3\theta_4 k + 2A}\right)\frac{2D}{\theta_3\theta_4 k + 2A} + \frac{4n\sigma^2}{(\theta_3\theta_4 k + 2A)^2}.$$

Multiply $M := (\theta_3\theta_4 k + \theta_3\theta_4 + 2A)(\theta_3\theta_4 k + 2A)(2D)^{-1}$ on both sides, we get

$$\begin{aligned}
M\mathbb{E}\mathcal{L}^{k+1} \leq & \left(1 - \frac{2\theta_3\theta_4}{\theta_3\theta_4 k + 2A}\right)(\theta_3\theta_4 k + \theta_3\theta_4 + 2A) + \frac{4n\sigma^2(\theta_3\theta_4 k + \theta_3\theta_4 + 2A)}{2D(\theta_3\theta_4 k + 2A)} \\
= & \frac{2D(\theta_3\theta_4 k + 2A - 2\theta_3\theta_4)(\theta_3\theta_4 k + \theta_3\theta_4 + 2A) + 4n\sigma^2(\theta_3\theta_4 k + \theta_3\theta_4 + 2A)}{2D(\theta_3\theta_4 k + 2A)} \\
= & \frac{2D(\theta_3\theta_4 k + 2A)^2 + 4n\sigma^2(\theta_3\theta_4 k + 2A) - 4D\theta_3\theta_4(\theta_3\theta_4 k + 2A) + 2D\theta_3\theta_4(\theta_3\theta_4 k + 2A)}{2D(\theta_3\theta_4 k + 2A)} \\
& + \frac{-4D(\theta_3\theta_4)^2 + 4n\sigma^2\theta_3\theta_4}{2D(\theta_3\theta_4 k + 2A)} \\
\leq & \theta_3\theta_4 k + 2A.
\end{aligned}$$

Hence

$$\mathbb{E}\mathcal{L}^{k+1} \leq \frac{2D}{\theta_3\theta_4(k+1) + 2A}$$

This induction holds for any A such that η_k is feasible, i.e.

$$\eta_0 = \frac{1}{A} \leq \theta_5.$$

Here we summarize the definition of constant numbers:

$$\theta_1 = \frac{1}{2\lambda_{\max}((\mathbf{I} - \mathbf{W})^\dagger)},\ \theta_2 = \frac{C\beta}{2(1+C)}, \tag{B.33}$$

$$\theta_3 = \min\{\theta_1, \theta_2\},\ \theta_4 \in \left(0, \frac{\mu}{C\beta}\right),\ \eta_* = \frac{2(\mu - C\beta\theta_4)}{\mu^2}, \tag{B.34}$$

$$\theta_5 = \min\left\{\frac{2}{\mu + L}, \eta_*, \frac{(3C+1) - \sqrt{(3C+1)^2 - 4C}}{C\beta\theta_4}, \frac{2}{\beta\theta_4}\right\}. \tag{B.35}$$

Therefore, let $A = \frac{1}{\theta_5}$ and $\eta_k = \frac{2\theta_5}{\theta_3\theta_4\theta_5 k + 2}$, we get

$$\frac{1}{n}\mathbb{E}\mathcal{L}^k \leq \frac{2\max\left\{\frac{1}{n}\mathcal{L}^0, \frac{2\sigma^2\theta_5}{\theta_3\theta_4}\right\}}{\theta_3\theta_4\theta_5 k + 2}.$$

Since $1 - a_1\alpha_k \geq 1/2$, we complete the proof.

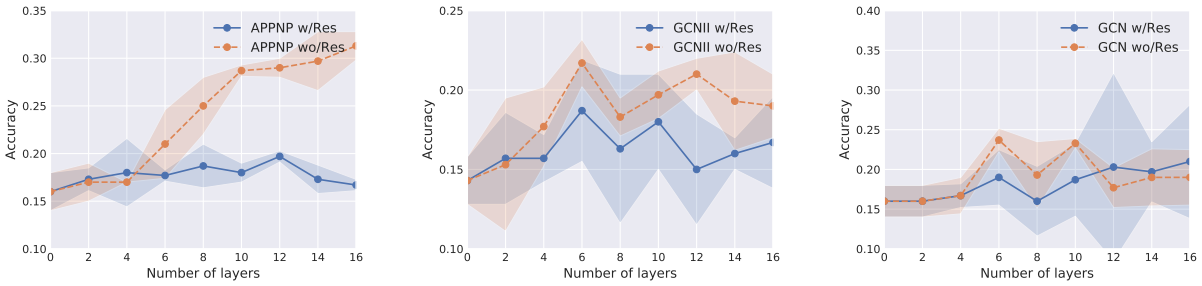
□

APPENDIX C

GRAPH NEURAL NETWORKS WITH ADAPTIVE RESIDUAL

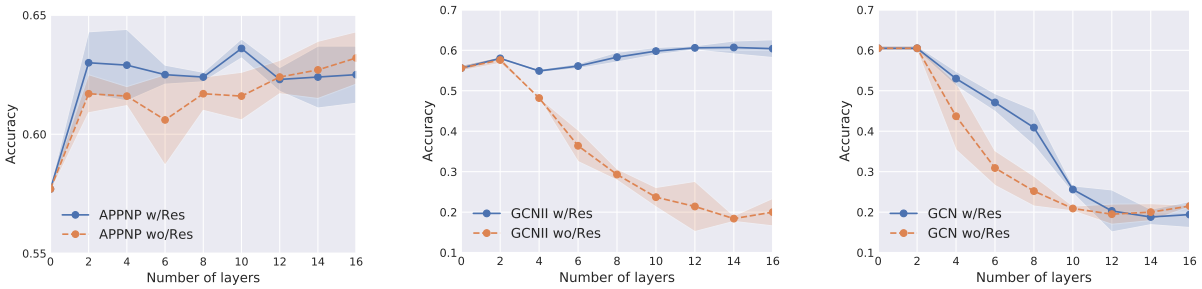
C.1 Additional Results for the Preliminary Study

In this section, we provide additional results on CiteSeer and PubMed datasets for the preliminary study in Section 4.2. The results on these two datasets are showed in Figure C.1, C.2, C.3 and C.4. It can be observed that residual connection helps obtain better performance on normal features but it is detrimental to abnormal features, which aligns with the findings in Section 4.2.



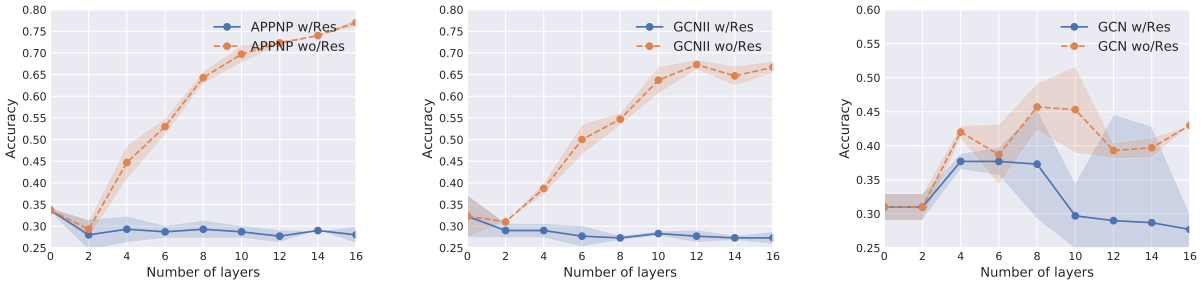
(a) APPNP (b) GCNII (c) GCN

Figure C.1: Node classification accuracy on abnormal nodes (CiteSeer).



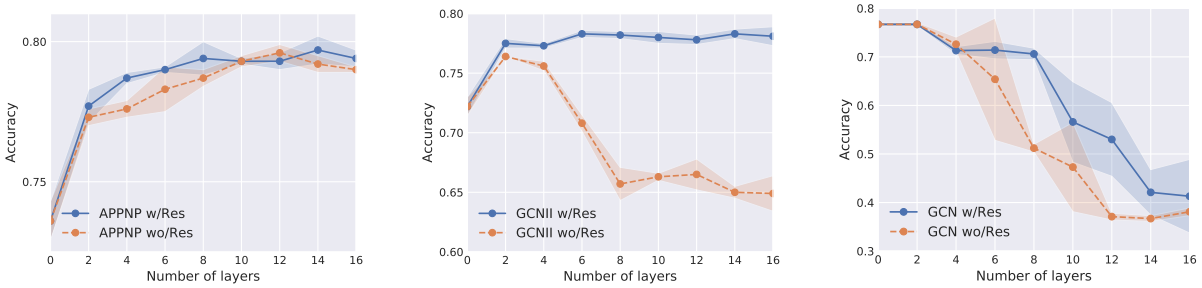
(a) APPNP (b) GCNII (c) GCN

Figure C.2: Node classification accuracy on normal nodes (CiteSeer).



(a) APPNP (b) GCNII (c) GCN

Figure C.3: Node classification accuracy on abnormal nodes (PubMed).



(a) APPNP (b) GCNII (c) GCN

Figure C.4: Node classification accuracy on normal nodes (PubMed).

C.2 Additional Experiments for the Proposed Method

In this section, we provide more experiments and ablation study for the proposed AirGNN.

C.2.1 Experiments on More Datasets

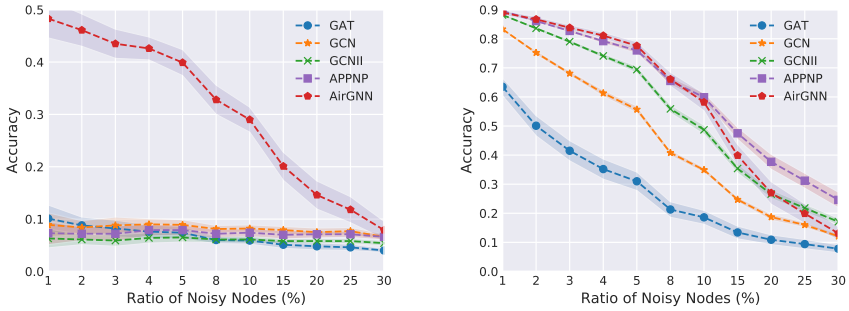
In this subsection, we provide additional experiments for Section 4.4. In particular, we conduct the experiments for the noisy feature scenario on the following 5 datasets: Coauthor CS [95], Coauthor Physics [95], Amazon Computers [95], Amazon Photo [95], and ogbn-arxiv [113]. The node classification accuracy are showed in Figures C.5, C.6, C.7, C.8, and C.9, respectively. Specifically, the accuracy on abnormal nodes and normal nodes are plotted separately in (a) and (b), with respect to the ratio of noisy nodes.

When the ratios of noisy nodes are within a reasonable range, we can observe that (1) AirGNN obtains much better accuracy on abnormal nodes on all datasets, which verifies its stronger resilience

to abnormal features; and (2) AirGNN achieves better or sometimes comparable accuracy on normal nodes in most cases, which shows its capability to maintain good performance for normal nodes.

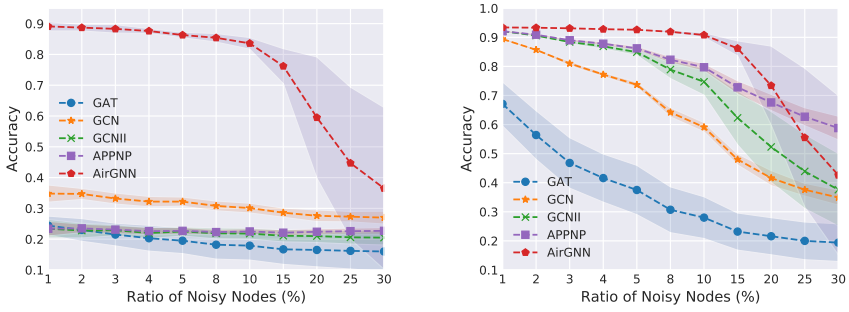
However, when the noise ratio is very high, the performance of AirGNN drops quickly. This is because the modulation hyperparameter λ is tuned based on the clean dataset such that it is far away from being optimal for highly noisy dataset. But it can be significantly improved by adjusting the hyperparameter λ as discussed in next subsection.

These results suggest the significant advantages of adaptive residual in AirGNN, and confirm the conclusion in the main paper. The adversarial attack on larger graphs is computationally expensive so we omit the results on more datasets in the adversarial feature scenario.



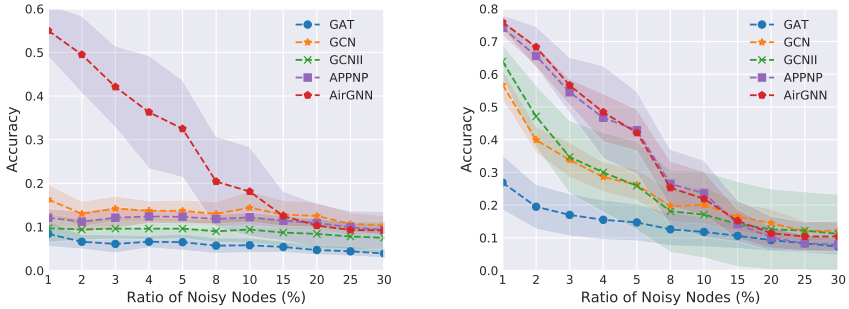
(a) Accuracy on abnormal nodes (b) Accuracy on normal nodes

Figure C.5: Node classification accuracy in noisy features scenario (Coauthor CS).



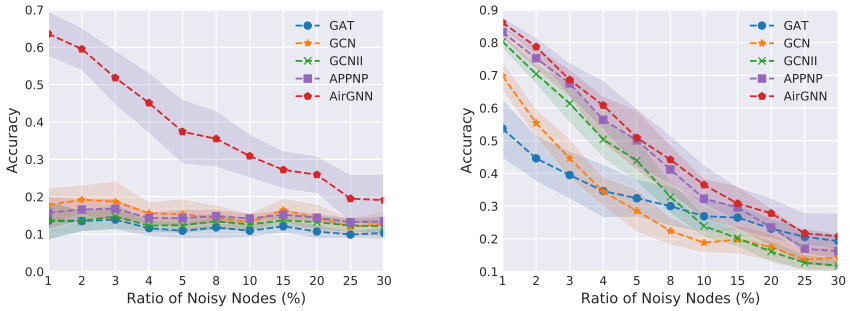
(a) Accuracy on abnormal nodes (b) Accuracy on normal nodes

Figure C.6: Node classification accuracy in noisy features scenario (Coauthor Physics).



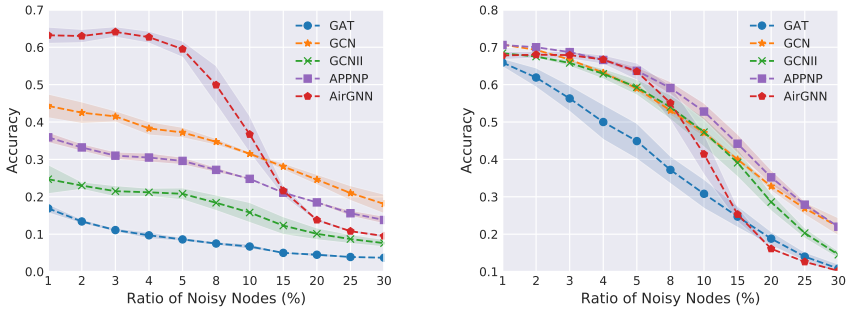
(a) Accuracy on abnormal nodes (b) Accuracy on normal nodes

Figure C.7: Node classification accuracy in noisy features scenario (Amazon Computers).



(a) Accuracy on abnormal nodes (b) Accuracy on normal nodes

Figure C.8: Node classification accuracy in noisy features scenario (Amazon Photo).



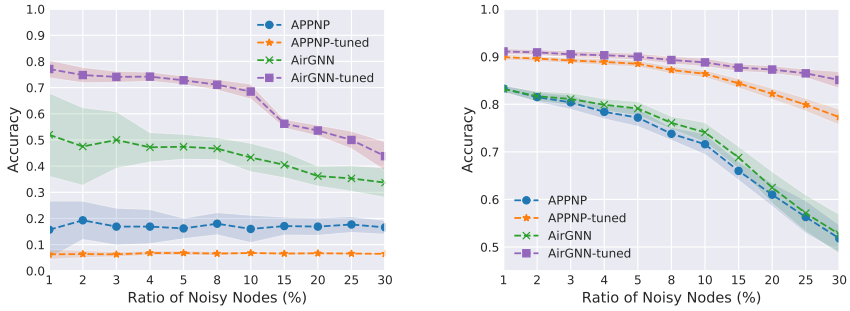
(a) Accuracy on abnormal nodes (b) Accuracy on normal nodes

Figure C.9: Node classification accuracy in noisy features scenario (ogbn-arxiv).

C.2.2 AirGNN with Adjusted λ

Note that in Figures C.5, C.6, C.7, C.8, and C.9, the performance of AirGNN drops significantly when the noise ratio is very large. This is because the modulation hyperparameter λ is tuned based

on the clean dataset such that it is far away from being optimal for highly noisy dataset. In fact, the performance of AirGNN can be significantly improved by adjusting λ during test time according to the performance on the validation set. Taking the Coauthor CS [95] dataset as an example, we compare AirGNN with APPNP and we tune the hyperparameter λ and α for them (denoted as AirGNN-tuned and APPNP-tuned) for a fair comparison as showed in Figure C.10. The result verifies that AirGNN-tuned gets tremendous improvement on both abnormal and normal nodes by adjusting λ. However, APPNP-tuned only focuses on improving global performance and overlooks the abnormal nodes after adjusting α based on validation performance so that the performance on abnormal node are much worse.



(a) Accuracy on abnormal nodes (b) Accuracy on normal nodes

Figure C.10: Node classification accuracy in noisy features scenario with adjustment (Coauthor CS).

C.2.3 Detailed Comparison with APPNP

Figure 4.1 in Section 4.2 shows that APPNP without residual performs well on the noisy nodes. Therefore, in order to demonstrate the advantages of AirGNN, it is of interest to make a detailed comparison between AirGNN and the two variants of APPNP (w/Res and wo/Res). We evaluate their performance on noisy nodes, normal nodes, and overall nodes on Cora dataset, and the results under varying noise ratio are summarized in Table C.1, Table C.2, and Table C.3. We can make the following observations:

- In Table C.1, both AirGNN and APPNP wo/Res significantly outperform APPNP w/Res on noisy nodes, and AirGNN achieves comparable performance with APPNP wo/Res. This verifies

that the residual connection in GNN amplifies the vulnarability to abnormal features, and AirGNN is able to adaptively adjust the residual connections for abnormal nodes to reduce the vulnerability.

- In Table C.2 and Table C.3, AirGNN consistently outperforms APPNP wo/Res, which verifies the importance of residual connections in maintaining good performance on normal nodes. AirGNN exhibits much better performance than APPNP w/Res, which shows the benefits of removing abnormal features by adaptive residual.

- APPNP wo/Res is a special case of AirGNN with $\lambda = 0$. Moreover, as noted in Section C.2.2, the performance of AirGNN in Table C.1, Table C.2, and Table C.3 can be further improved by adjusting the modulation hyperparameter λ for each noise ratio according to validation performance.

As discussed in Section 4.3, in existing GNNs such as APPNP and GCNII, the conflict between feature aggregation and residual connection can only be partially mitigated by adjusting the residual weight α. However, such global adjustment cannot be adaptive to a subset of the nodes, which explains the advantages of AirGNN in above observations. In the adversarial feature setting, we can make similar observations but here we omit the comparison.

Table C.1: Comparison between APPNP and AirGNN on abnormal (noisy) nodes (Cora).

Noisy ratio	5%	10%	15%	20%	25%	30%
APPNP w/Res	0.167 ± 0.034	0.170 ± 0.070	0.170 ± 0.027	0.193 ± 0.031	0.187 ± 0.024	0.178 ± 0.026
APPNP wo/Res	0.469 ± 0.035	0.442 ± 0.062	0.427 ± 0.038	0.381 ± 0.043	0.383 ± 0.045	0.354 ± 0.067
AirGNN	0.474 ± 0.048	0.433 ± 0.055	0.405 ± 0.050	0.362 ± 0.039	0.353 ± 0.050	0.337 ± 0.057

Table C.2: Comparison between APPNP and AirGNN on normal nodes (Cora).

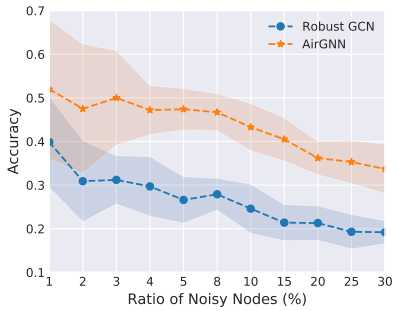
Noisy ratio	5%	10%	15%	20%	25%	30%
APPNP w/Res	0.773 ± 0.015	0.712 ± 0.024	0.669 ± 0.019	0.622 ± 0.024	0.580 ± 0.032	0.530 ± 0.029
APPNP wo/Res	0.761 ± 0.014	0.709 ± 0.025	0.664 ± 0.015	0.599 ± 0.025	0.556 ± 0.035	0.497 ± 0.049
AirGNN	0.791 ± 0.015	0.741 ± 0.021	0.688 ± 0.024	0.625 ± 0.034	0.571 ± 0.039	0.527 ± 0.042

Table C.3: Comparison between APPNP and AirGNN on all nodes (Cora).

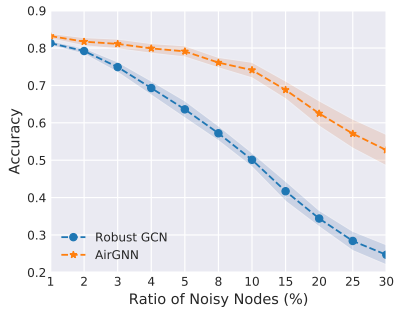
Noisy ratio	5%	10%	15%	20%	25%	30%
APPNP w/Res	0.743 ± 0.015	0.657 ± 0.026	0.594 ± 0.017	0.536 ± 0.024	0.482 ± 0.025	0.425 ± 0.025
APPNP wo/Res	0.746 ± 0.013	0.682 ± 0.026	0.628 ± 0.015	0.556 ± 0.027	0.513 ± 0.034	0.455 ± 0.053
AirGNN	0.775 ± 0.015	0.710 ± 0.021	0.646 ± 0.025	0.572 ± 0.033	0.516 ± 0.038	0.470 ± 0.044

C.2.4 Comparison with Robust Model

To further demonstrate the advantages of the proposed AirGNN, we compare it with a representative robust model, Robust GCN [128]. Tables C.11, C.12 and C.13 show the performance comparison between Robust GCN and AirGNN on Cora, Citeseer and PubMed, respectively. The accuracy on abnormal nodes and normal nodes are plotted separately in (a) and (b), with respect to the ratio of noisy nodes. These figures show that AirGNN achieves significant better performance than Robust GCN on both abnormal and normal nodes in the noisy feature scenario.

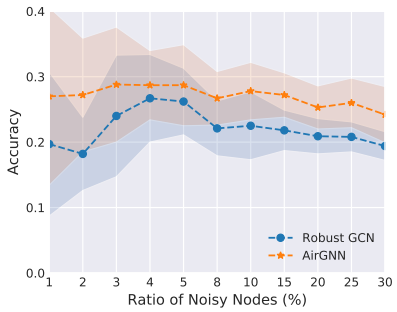


(a) Accuracy on abnormal nodes

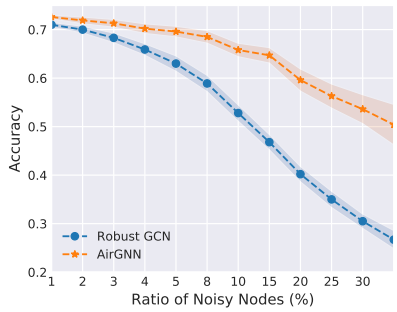


(b) Accuracy on normal nodes

Figure C.11: Node classification accuracy in noisy features scenario (Cora).

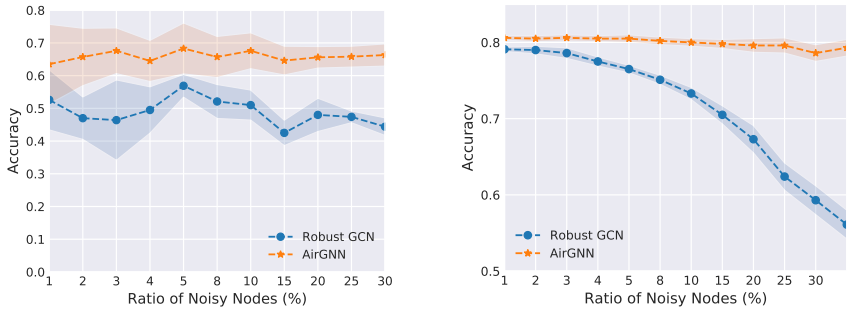


(a) Accuracy on abnormal nodes



(b) Accuracy on normal nodes

Figure C.12: Node classification accuracy in noisy features scenario (CiteSeer).



(a) Accuracy on abnormal nodes (b) Accuracy on normal nodes

Figure C.13: Node classification accuracy in noisy features scenario (PubMed).

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, 2016.

[2] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005.

[3] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 440–445, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[4] 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, 2017.

[5] Angelica I Aviles-Rivero, Nicolas Papadakis, Ruoteng Li, Samar M Alsaleh, Robby T Tan, and Carola-Bibiane Schonlieb. When labelled data hurts: Deep semi-supervised classification with the graph 1-laplacian. *arXiv preprint arXiv:1906.08635*, 2019.

[6] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[7] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. SIGNSGD: compressed optimisation for non-convex problems. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 559–568, 2018.

[8] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.

[9] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[10] Xavier Bresson, Thomas Laurent, David Uminsky, and James H. von Brecht. An adaptive total variation algorithm for computing the balanced cut of a graph, 2013.

[11] Xavier Bresson, Thomas Laurent, David Uminsky, and James H Von Brecht. Multiclass total variation clustering. *arXiv preprint arXiv:1306.1185*, 2013.

[12] Thomas Bühler and Matthias Hein. Spectral clustering based on the graph p-laplacian. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 81–88, 2009.

[13] Ruggero Carli, Fabio Fagnani, Paolo Frasca, and Sandro Zampieri. Gossip consensus algorithms via quantized communication. *Automatica*, 46(1):70–80, 2010.

[14] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1725–1735. PMLR, 13–18 Jul 2020.

[15] Peijun Chen, Jianguo Huang, and Xiaoqun Zhang. A primal–dual fixed point algorithm for convex separable minimization with applications to image restoration. *Inverse Problems*, 29(2):025011, 2013.

[16] Siheng Chen, Yonina C. Eldar, and Lingxiao Zhao. Graph unrolling networks: Interpretable neural networks for graph signal denoising, 2020.

[17] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015.

[18] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in Neural Information Processing Systems*, 33, 2020.

[19] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[20] Laurent Condat. A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms. *Journal of optimization theory and applications*, 158(2):460–479, 2013.

[21] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

[22] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1223–1231, USA, 2012.

[23] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3844–3852, 2016.

[24] Tyler Derr, Yao Ma, Wenqi Fan, Xiaorui Liu, Charu Aggarwal, and Jiliang Tang. Epidemic graph convolutional network. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 160–168, 2020.

[25] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.

[26] P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, March 1975.

[27] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177, 2020.

[28] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. Graph trend filtering networks for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–121, 2022.

[29] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.

[30] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[31] Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. *arXiv preprint arXiv:1901.09401*, 2019.

[32] William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

[33] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

[34] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman Hall/CRC, 2015.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[36] Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Sebastian Stich, and Peter Richtárik. Stochastic distributed learning with gradient quantization and variance reduction. *arXiv preprint arXiv:1904.05115*, 2019.

[37] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34, 2021.

[38] Wei Jin, Xiaorui Liu, Yao Ma, Charu Aggarwal, and Jiliang Tang. Towards feature overcorrelation in deeper graph neural networks. In *Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2022.

[39] Wei Jin, Xiaorui Liu, Yao Ma, Tyler Derr, Charu Aggarwal, and Jiliang Tang. Graph feature gating networks. In *Proceedings of the 30th ACM International Conference on Information amp; Knowledge Management*, CIKM '21, page 813–822, New York, NY, USA, 2021. Association for Computing Machinery.

[40] Wei Jin, Xiaorui Liu, Xiangyu Zhao, Yao Ma, Neil Shah, and Jiliang Tang. Automated self-supervised learning for graphs. In *International Conference on Learning Representations*, 2022.

[41] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 66–74, 2020.

[42] Michael I Jordan, Jason D Lee, and Yun Yang. Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, 114(526):668–681, 2019.

[43] A. Jung, A. O. Hero, III, A. C. Mara, S. Jahromi, A. Heimowitz, and Y. C. Eldar. Semi-supervised learning in network-structured data via total variation minimization. *IEEE Transactions on Signal Processing*, 67(24):6256–6269, 2019.

[44] Alexander Jung, Alfred O Hero III, Alexandru Mara, and Saeed Jahromi. Semi-supervised learning via sparse label propagation. *arXiv preprint arXiv:1612.01414*, 2016.

[45] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Urban Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3252–3261. PMLR, 2019.

[46] Seung-Jean Kim, Kwangmoo Koh, Stephen Boyd, and Dimitry Gorinevsky. ℓ_1 trend filtering. *SIAM review*, 51(2):339–360, 2009.

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[48] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[49] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[50] Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. In *International Conference on Learning Representations*, 2020.

[51] Anastasia Koloskova, Sebastian U. Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3479–3487. PMLR, 2019.

[52] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[53] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[54] 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 *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 583–598, Berkeley, CA, USA, 2014. USENIX Association.

[55] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[56] Yao Li, Xiaorui Liu, Jiliang Tang, Ming Yan, and Kun Yuan. Decentralized composite optimization with compression. *arXiv preprint arXiv:2108.04448*, 2021.

[57] Yao Li and Ming Yan. On linear convergence of two decentralized algorithms. *arXiv preprint arXiv:1906.07225*, 2019.

[58] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses, 2020.

[59] Zhi Li, Wei Shi, and Ming Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, 67(17):4494–4506, 2019.

[60] Zhi Li and Ming Yan. New convergence analysis of a primal dual algorithm with large stepsizes. *Advances in Computational Mathematics*, 2021.

[61] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2737–2745. Curran Associates, Inc., 2015.

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

[63] Qing Ling, Wei Shi, Gang Wu, and Alejandro Ribeiro. DLM: Decentralized linearized alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 63(15):4051–4064, 2015.

[64] Haochen Liu, Yiqi Wang, Wenqi Fan, Xiaorui Liu, Yaxin Li, Shaili Jain, Yunhao Liu, Anil K. Jain, and Jiliang Tang. Trustworthy ai: A computational perspective. *ACM Trans. Intell. Syst. Technol.*, jun 2022. Just Accepted.

[65] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2020.

[66] Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. Graph neural networks with adaptive residual. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[67] Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. Elastic graph neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6837–6849. PMLR, 18–24 Jul 2021.

[68] Xiaorui Liu, Yao Li, Jiliang Tang, and Ming Yan. A double residual compression algorithm for efficient distributed learning. *The 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

[69] Xiaorui Liu, Yao Li, Rongrong Wang, Jiliang Tang, and Ming Yan. Linear convergent decentralized optimization with compression. In *International Conference on Learning Representations*, 2021.

[70] Ignace Loris and Caroline Verhoeven. On a generalization of the iterative soft-thresholding algorithm for the case of non-separable penalty. *Inverse Problems*, 27(12):125007, 2011.

[71] Yucheng Lu and Christopher De Sa. Moniqua: Modulo quantized communication in decentralized SGD. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[72] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2022.

[73] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. A unified view on graph neural networks as graph signal denoising. *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 2021.

[74] Yao Ma and Jiliang Tang. *Deep Learning on Graphs*. Cambridge University Press, 2020.

[75] Sindri Magnússon, Hossein Shokri-Ghadikolaei, and Na Li. On maintaining linear convergence of distributed learning and optimization under limited communication. *IEEE Transactions on Signal Processing*, 68:6101–6116, 2020.

[76] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[77] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

[78] Joao FC Mota, Joao MF Xavier, Pedro MQ Aguiar, and Markus Püschel. D-ADMM: A communication-efficient distributed algorithm for separable optimization. *IEEE Transactions on Signal Processing*, 61(10):2718–2723, 2013.

[79] Angelia Nedic, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[80] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[81] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[82] Lam Nguyen, PHUONG HA NGUYEN, Marten van Dijk, Peter Richtarik, Katya Scheinberg, and Martin Takac. SGD and hogwild! Convergence without the bounded gradients assumption. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3750–3758, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[83] Feiping Nie, Hua Wang, Heng Huang, and Chris Ding. Unsupervised and semi-supervised learning via ℓ_1-norm graph. In *2011 International Conference on Computer Vision*, pages 2268–2273. IEEE, 2011.

[84] Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

[85] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020.

[86] Xuran Pan, Song Shiji, and Huang Gao. A unified framework for convolution-based graph neural networks. *https://openreview.net/forum?id=zUMD–Fb9Bt*, 2020.

[87] Shi Pu and Angelia Nedić. Distributed stochastic gradient tracking methods. *Mathematical Programming*, pages 1–49, 2020.

[88] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani. An exact quantized decentralized gradient descent algorithm. *IEEE Transactions on Signal Processing*, 67(19):4934–4947, 2019.

[89] Amirhossein Reisizadeh, Hossein Taheri, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani. Robust and communication-efficient collaborative learning. In *Advances in Neural Information Processing Systems*, pages 8388–8399, 2019.

[90] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[91] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[92] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and application to data-parallel distributed training of speech DNNs. In *Interspeech 2014*, September 2014.

[93] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[94] James Sharpnack, Aarti Singh, and Alessandro Rinaldo. Sparsistency of the edge lasso over graphs. In *Artificial Intelligence and Statistics*, pages 1028–1036. PMLR, 2012.

[95] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[96] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[97] Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4452–4463, USA, 2018. Curran Associates Inc.

[98] Nikko Strom. Scalable distributed DNN training using commodity GPU cloud computing. In *INTERSPEECH*, pages 1488–1492, 2015.

[99] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[100] Arthur Szlam and Xavier Bresson. Total variation, cheeger cuts. In *ICML*, 2010.

[101] 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.

[102] Hanlin Tang, Xiangru Lian, Shuang Qiu, Lei Yuan, Ce Zhang, Tong Zhang, and Ji Liu. Deepsqueeze: Decentralization meets error-compensated compression. *CoRR*, abs/1907.07346, 2019.

[103] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D^2: Decentralized training over decentralized data. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4848–4856, 2018.

[104] Hanlin Tang, Chen Yu, Xiangru Lian, Tong Zhang, and Ji Liu. DoubleSqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6155–6165, 2019.

[105] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[106] Ryan J Tibshirani et al. Adaptive piecewise polynomial estimation via trend filtering. *Annals of statistics*, 42(1):285–323, 2014.

[107] John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

[108] Rohan Varma, Harlin Lee, Jelena Kovačević, and Yuejie Chi. Vector-valued graph trend filtering with non-convex penalties. *IEEE Transactions on Signal and Information Processing over Networks*, 6:48–62, 2019.

[109] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[110] Jialei Wang, Mladen Kolar, Nathan Srebro, and Tong Zhang. Efficient distributed learning with sparsity. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3636–3645, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[111] Yu-Xiang Wang, James Sharpnack, Alexander J Smola, and Ryan J Tibshirani. Trend filtering on graphs. *Journal of Machine Learning Research*, 17:1–41, 2016.

[112] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 1306–1316, USA, 2018. Curran Associates Inc.

[113] Marinka Zitnik Yuxiao Dong Hongyu Ren Bowen Liu Michele Catasta Jure Leskovec Weihua Hu, Matthias Fey. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[114] 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.

[115] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[116] Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized SGD and its applications to large-scale distributed optimization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5325–5333, 10–15 Jul 2018.

[117] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[118] Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

[119] Han Xu, Xiaorui Liu, Yaxin Li, Anil Jain, and Jiliang Tang. To be robust or to be fair: Towards fairness in adversarial training. In Marina Meila and Tong Zhang, editors, *Proceedings of the*

38th International Conference on Machine Learning, volume 139 of *Proceedings of Machine Learning Research*, pages 11492–11501. PMLR, 18–24 Jul 2021.

[120] Jinming Xu, Ye Tian, Ying Sun, and Gesualdo Scutari. Accelerated primal-dual algorithms for distributed smooth convex optimization over networks. In *International Conference on Artificial Intelligence and Statistics*, pages 2381–2391. PMLR, 2020.

[121] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5453–5462. PMLR, 10–15 Jul 2018.

[122] 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*, ICPP 2018, pages 1:1–1:10, New York, NY, USA, 2018. ACM.

[123] Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

[124] Kun Yuan, Wei Xu, and Qing Ling. Can primal methods outperform primal-dual methods in decentralized dynamic optimization? *arXiv preprint arXiv:2003.00816*, 2020.

[125] Kun Yuan, Bicheng Ying, Xiaochuan Zhao, and Ali H Sayed. Exact diffusion for distributed optimization and learning—part i: Algorithm development. *IEEE Transactions on Signal Processing*, 67(3):708–723, 2018.

[126] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2019.

[127] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. 2004.

[128] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1399–1407, 2019.

[129] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. Interpreting and unifying graph neural networks with an optimization framework, 2021.

[130] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation.

[131] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

[132] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*, 2021.

[133] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J. Smola. Parallelized stochastic gradient descent. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2595–2603. Curran Associates, Inc., 2010.

[134] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *KDD*. ACM, 2018.

[135] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412*, 2019.