Full-Graph vs. Mini-Batch Training: Comprehensive Analysis from a Batch Size and Fan-Out Size Perspective [Experiment, Analysis & Benchmark]

Mengfan Liu
The University of Hong Kong
Hong Kong SAR, China
ml621@connect.hku.hk

Da Zheng
Ant Technology Research Institute
Hang Zhou, China
zhengda.zheng@antgroup.com

Junwei Su
The University of Hong Kong
Hong Kong SAR, China
junweisu@connect.hku.hk

Chuan Wu
The University of Hong Kong
Hong Kong SAR, China
cwu@cs.hku.hk

ABSTRACT

Graph neural networks (GNNs) are widely used for representational learning on graph-structured data. Full-graph training and mini-batch training have been used to learn GNNs, where full-graph training can be viewed as mini-batch training with the largest possible batch and fan-out sizes. The two training approaches require different system optimizations, and their differences in system efficiency and model performance (i.e., convergence and generalization) remain insufficiently explored. Existing studies have provided overall comparisons of model performance between full-batch training and mini-batch training, but do not capture how key hyperparameters (e.g., batch size and fan-out size) that distinguish these two approaches affect model performance. A detailed comparison is crucial in terms of these key hyperparameters, to inform training design decisions for faster model convergence or better generalization performance. This paper compares full batch vs. mini-batch training of GNNs from the perspective of batch size and fan-out size, addressing two key questions: *1) What are the model training behaviors (i.e., convergence, generalization performance and system efficiency) of each approach? 2) How do graph-based hyperparameters (e.g., batch size and fan-out size) and hardware environments (e.g., cluster size) impact the model performance and system efficiency?*

We present a systematic empirical and theoretical analysis of full-graph and mini-batch GNN training for transductive node classification tasks on homogeneous bidirectional graphs. Our key findings include: 1) larger batch sizes lead to more iterations for convergence of GNNs trained with the mean square error (MSE) loss, different from DNN training behaviors; 2) larger fan-out sizes reduce iterations required for convergence, regardless of whether MSE or the cross entropy (CE) loss is used; 3) Full-graph training, viewed as mini-batch with the largest batch and fan-out sizes, does not always yield superior model performance or system efficiency compared to smaller mini-batch settings. Carefully tuning these graph-based hyperparameters can produce more favorable trade-offs in practice (e.g, for faster convergence or better generalization), balancing generalization performance, convergence speed, and system efficiency.

PVLDB Reference Format:
Mengfan Liu, Da Zheng, Junwei Su, and Chuan Wu. Full-Graph vs. Mini-Batch Training: Comprehensive Analysis from a Batch Size and Fan-Out Size Perspective [Experiment, Analysis & Benchmark]. PVLDB, 14(1): XXX-XXX, 2025.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/LIUMENGFAN-gif/GNN_fullgraph_minibatch_training.

1 INTRODUCTION

Graph neural networks (GNNs) have demonstrated state-of-the-art performance across machine learning tasks involving graph-structured data [31, 65, 75]. In GNN training, features or embeddings from each training node's neighborhood are iteratively aggregated to generate the node's embedding, which is referred to as message passing [20]. GNNs are commonly trained using either *mini-batch* training or *full-graph* training. Each mini-batch contains training nodes and their multi-hop neighbors [23], where the batch size is the number of training nodes sampled in each mini-batch and the fan-out size specifies the number of neighbors chosen per node at each neighbor hop. As graph structures used in training vary according to the batch size and fan-out size, we refer to these two parameters as graph-based hyperparameters in GNN training. We view full-graph training of GNNs as mini-batch training with the largest batch size and fan-out size, analogous to how full-batch training [54] of deep neural networks (DNNs) can be deemed as mini-batch training with the largest batch size. Mini-batch GNN training under different batch and fan-out sizes may exhibit varied convergence and generalization performance.

Full-graph training and mini-batch training give rise to two distinct classes of GNN training systems. When scaling to large graphs, full-graph training systems partition the graph into smaller subgraphs that can each fit in a single GPU, facilitating parallel training using multiple GPUs [27, 61]. These systems render a

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

unique opportunity for reducing communication overhead in the exchange of neighbor node features or embeddings during node feature/embedding aggregation, by asynchronous communication [44, 50], pipeline parallelism [63], boundary node sampling [62] and quantization [61] techniques. Mini-batch training confines node aggregation within a single GPU, processing smaller mini-batches across multiple GPUs [79, 80]. Mini-batch training systems render additional optimization aspects such as distributed node sampling and feature fetching between GPUs and CPUs [9, 11, 40, 81].

Full-graph training designs mainly focus on expediting convergence without compromising test accuracy at the system level [44, 57, 61, 62], while mini-batch training designs develop both system-level convergence acceleration techniques [40, 79, 80] and algorithm-level improvements of convergence and generalization (e.g., sampling methods [9, 11, 13, 23, 49, 73, 83]). A detailed comparison between full-graph training and mini-batch training is important for in-depth understanding of their pros and cons, for better training design decisions that achieve faster model convergence or better generalization performance. Bajaj et al. [3] provides an empirical study comparing full-graph and mini-batch GNN training on convergence and generalization performance. As the only full-graph vs. mini-batch training comparison study so far, they do not consider the impact of key hyperparameters (e.g., batch size and fan-out size) on GNN model performance (e.g., convergence and generalization performance) and system efficiency. These graph-related hyperparameters are crucial in deciding trade-offs among convergence speed, generalization performance and system efficiency, as inspired by the study of batch size in well-developed DNN training analyses [4, 21, 24, 30, 47, 53, 70, 74, 82]. In addition, the existing literature presents conflicting empirical findings regarding which training approach demonstrates superior convergence and generalization performance [3, 6, 28, 62–64, 78, 80]. In-depth analysis and empirical study on the impact of batch size and fan-out size on GNN training performance are needed, for further understanding of pros and cons of full-graph vs. mini-batch GNN training.

Yuan et al. [71] empirically compare GNN generalization and convergence performance across varying batch sizes and fan-out sizes. It lacks theoretical support, and examines hyperparameter values much smaller than those of full-graph training on medium-scale graphs (e.g., ogbn-arxiv [25]). Hu et al. [26] empirically examine how the batch size, ranging from small values to full-batch, affect GNN generalization performance and convergence speed, attempting to use gradient variance to explain these impacts. However, their explanation and empirical findings are inconsistent. None of them provide a valid theoretical analysis.

Existing theoretical analyses of GNN training [2, 56, 60, 67] focus on the degrees and the number of nodes in the training graphs, and overlook the roles of batch size and fan-out size on GNN performance. Although mini-batch training of DNNs enables superior scalability [69] and model performance [30] over full-batch DNN training, performance analyses of DNN training cannot be directly applied to GNN training due to the unknown impact of the message-passing process. Further, recent GNN empirical studies focus on time-based performance metrics for convergence, such as epoch time (training time per epoch) [8, 44, 64] and time-to-accuracy (time to reach a target validation accuracy) [3, 25]. These metrics are highly sensitive to hardware differences (e.g., computational power and communication bandwidth), and the time estimation may not apply in new hardware environments. Moreover, time-based metrics does not distinguish model performance improvement per training iteration (in terms of accuracy or loss) from system efficiency (e.g., the numbder of processed nodes per second). It is desirable to evaluate hardware-independent metrics that can capture model performance improvement with the training progress. This requires more in-depth theoretical exploration of GNN performance behaviors, which can isolate key factors driving training dynamics and provide insights that generalize beyond specific datasets and hardware configurations.

Research questions. These prompt further investigation of the following questions: *1) What are the model performance (i.e., convergence and generalization) and system efficiency achievable by full-graph training and mini-batch training? 2) How do graph-based hyperparameters (e.g., batch size and fan-out size) and hardware environments (e.g., cluster size) impact the model performance and system efficiency?* We seek an integrated empirical and theoretical study to answer the above questions.

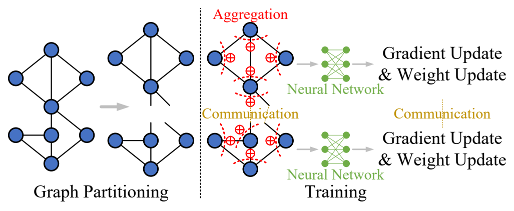
We conduct a systematic study of full-graph and mini-batch GNN training under different batch sizes and fan-out sizes on transductive node classification tasks for bidirectional graphs. Theoretically, we formulate our analysis under practical settings (e.g., non-regular graphs with nodes of varying degrees and GNNs with finite width and non-linear ReLU activations), considering graph structure differences between training and testing datasets. Empirically, we highlight the importance of using additional convergence metrics, such as iteration-to-loss (i.e., the number of iterations needed to achieve a target training loss) and iteration-to-accuracy (i.e., the number of iterations needed to achieve a target validation accuracy), rather than relying solely on time-to-accuracy. These metrics ensure that our comparison results generalize across different hardware environments. This paper utilizes real-world datasets such as reddit [23], ogbn-arxiv [25], ogbn-products [25], and ogbn-papers100M [25]. We make all our scripts and appendix materials publicly available.[1]. Our contributions in this paper can be summarized as follows:

▹ We extend the GNN convergence study from regular graphs to non-regular graphs using the Polyak-Łojasiewicz (PL) condition, considering GNNs trained with non-linear activations and mean square error (MSE), to better align with practical settings. We theoretically identify that for a fixed fan-out size, GNN training under larger batch sizes requires more iterations for convergence, different from the DNN training convergence result. Our empirical results support this theoretical finding.

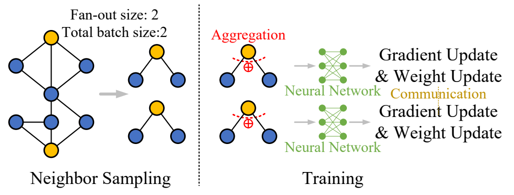
▹ We investigate the impact of fan-out size on GNN convergence, which is a hyperparameter absent in DNNs and remains theoretically unexplored in GNN training. We theoretically show that for a fixed batch size, GNNs with larger fan-out sizes require fewer iterations to convergence. Our empirical results support this finding, regardless of whether MSE or cross entropy (CE) loss is used.

▹ We theoretically use the Wasserstein distance to quantify graph structure differences between training and testing datasets, and analyze GNN generalization under mini-batch training using the PAC-Bayesian framework, focusing on the impact of batch size and fan-out size on generalization.

[1]https://github.com/LIUMENGFAN-gif/GNN_fullgraph_minibatch_training



(a) Full-graph Training



(b) Mini-batch Training

Figure 1: Distributed full-graph and mini-batch training of a one-layer GNN model.

▹ We identify that full-graph training does not always yield superior model performance or system efficiency compared to smaller mini-batch training settings. Carefully tuning these graph-based hyperparameters can produce more favorable trade-offs in practice, balancing model generalization performance, convergence speed and system efficiency. Specially, we find that increasing batch sizes improves generalization performance more effectively than increasing fan-out sizes; conversely, increasing fan-out sizes accelerate convergence more effectively than increasing batch size.

2 BACKGROUND AND MOTIVATION

2.1 Graph Neural Networks

Distinct from DNN training, GNN training adopts message passing with the aggregation of neighbor node features/embeddings and the update of each node's embedding. The embedding $\mathbf{x}_i^l$ of each training node i at l-th layer of a GNN [20] is:

$$\mathbf{x}_i^l = update(\mathbf{x}_i^{l-1}, aggregate(\mathbf{x}_j^{l-1}, j \in neighbor(i))), \qquad (1)$$

where $neighbor(i)$ denotes the neighborhood of node i.

Different aggregation and update operations exist in various GNN models. For example, GCNs [76] use the symmetrically normalized graph Laplacian as the aggregation operation. GraphSAGE [23] employs functions like mean, LSTM, or pooling for aggregation. GAT [59] introduces a self-attention mechanism that assigns weights to neighbors via attention coefficients, enabling weighted feature aggregation. These three models update node features by linear transformation and non-linear activation.

GNNs can be trained using *mini-batch* training or *full-graph* training. With mini-batch training, sampled training nodes in the graph dataset are divided into multiple mini-batches. Each training epoch, for training over the entire graph, is divided into multiple iterations, with each iteration processing the nodes of a mini-batch and their neighbors for message-passing, followed by stochastic gradient descent (SGD) for GNN model update. In this paper, mini-batch training employs uniform neighbor sampling, with a fixed number of multi-hop neighbors randomly selected for each training node [23]. In full-graph training, message passing works on all training nodes across the entire graph in each training epoch, followed by gradient descent (GD) to update model parameters. It can be viewed as mini-batch training with the largest possible batch size and fan-out size, and each training epoch contains only one iteration. We focus on transductive node classification tasks, where the testing and training nodes are in the same graph.

2.2 Full-graph vs. Mini-batch Training Systems

On large-scale graphs, full-graph and mini-batch training may require different data management pipelines for multi-GPU training, presenting different optimization opportunities in system design.

Full-graph training systems. Full-graph training can use graph parallelism to distribute the workload across GPUs [27, 61, 63, 64]: the entire graph is partitioned into multiple subgraphs, each GPU handles one partition with the entire GNN model, and communication occurs across GPUs for both embedding aggregation during message passing and gradient aggregation for model update, as shown in Figure 1(a). To reduce communication overhead, a number of optimization strategies have been exploited asynchronous communication, pipeline parallelism, sampling, and quantization. Asynchronous communication bypasses synchronous node feature/embedding aggregation and uses historical embeddings for computation [44, 57]. With pipeline parallelism, communication and computation overlap, minimizing resource idle time [44, 63]. Sampling selectively communicates among boundary nodes across partitions [62], maintaining synchronous training while reducing message volume. With quantization, messages are compressed into lower-precision integers, reducing both communication and computational overheads [61].

Mini-batch training systems. Mini-batch training can naturally employ data parallelism by distributing sampled mini-batches across multiple GPUs for concurrent computation [79, 80]. Compared to full-graph training, embedding aggregation during message passing in mini-batch training is limited within individual GPUs, and communication across GPUs is only required for gradient aggregation, as illustrated in Figure 1(b). However, each iteration processes different mini-batches and their neighbors, and a single GPU cannot store all mini-batches. Consequently, different from full-graph training that pre-loads all features onto the GPUs, mini-batches are cached in CPUs, necessitating frequent and substantial input feature fetching between the CPU and GPU during mini-batch training. To optimize input feature fetching to GPUs, caching strategies are proposed to reduce the heavy CPU-GPU communication. Some advocate static caching, replicating frequently accessed remote nodes locally [81], storing high out-degree nodes [38], or caching frequently accessed "hot" nodes [45]. BGL [40] implements dynamic caching using FIFO policies. Some [7, 55, 77] cache both graph structure and feature data directly on GPUs to further enhance system efficiency, but this strategy can be limited by fan-out size and batch size.

2.3 Motivations and Challenges

2.3.1 Motivation 1: Unclear performance behaviors. Mini-batch GNN training under different batch sizes and fan-out sizes exhibit varied performance (i.e., convergence and generalization) and system efficiency during training. Understanding the model performance and system efficiency behaviors is essential in choosing the most suitable training approach (mini-batch or full-graph training) and the key hyperparameters (e.g., batch size and fan-out size) for faster convergence or better generalization. These behaviors can guide the design of model optimization strategies as well, e.g., adaptive hyperparameter tuning and optimizer design. The existing literature presents conflicting empirical findings on the behaviors: some studies [6, 62–64] argue that full-graph training achieves higher model accuracy and faster convergence than mini-batch training, while others [3, 28, 78, 80] present contrasting findings.

Research question 1. We propose our first question concerning full-graph vs. mini-batch training of GNNs: "What are the model performance behaviors (i.e., convergence and generalization) and system efficiency of each approach?"

2.3.2 *Motivation 2: Methodology.*

To study full-graph vs. mini-batch GNN training, the analyses of DNN training performance [4, 21, 24, 30, 47, 53, 70, 82] cannot be directly transferred to GNNs due to the uncertain impact of the message-passing process. Besides, previous GNN studies are constrained by their methodology as follows.

Batch size and fan-out size. The sole existing empirical study comparing full-graph and mini-batch training [3] provides an overall comparison on convergence speed and generalization performance. It does not dissect the underlying factors driving the comparison results, nor the individual impact of key hyperparameters (e.g., batch size and fan-out size) on GNN model performance. Consequently, it obscures the trade-offs achieved by tuning these hyperparameters. On the other hand, DNN training performance is well-studied on batch size, with a clear picture on the behaviors of generalization [30, 70], convergence [4, 21, 53, 82] and system efficiency [24, 47]. Inspired by DNN studies, we inspect GNN training on different batch sizes and fan-out sizes, which is relevant to the hardware environment of training as well (e.g., cluster size), as a larger cluster can support a higher batch size.

Current GNN studies are limited on comparing GNN model performance across varying batch sizes and fan-out sizes. Empirically, Yuan et al. [71] show that smaller batch sizes lead to shorter time to convergence but reduce model accuracy; the generalization performance on ogbn-arxiv [25] shows a trend of "first improve and then degrade" as the fan-out size increases while the convergence speed exhibits the opposite. This study uses batch sizes and fan-out sizes much lower than those in full-graph training, and does not provide theoretical support. Hu et al. [26] empirically claim that, as batch size grows from small values to full-batch, the test accuracy first increases and then decreases, and the convergence time increases. They use the gradient variance to explain the impact of batch size on GNN generalization performance; however, their explanatory and empirical findings on optimal batch sizes are misaligned, as their explanation assumes that the minimal degree approximates the average degree, which is often impractical. Further, it does not explore the impact of fan-out size on generalization and convergence. These existing studies draw inconsistent conclusions, lack a valid theoretical analysis, and examine GNN training performance only on medium-scale datasets, such as ogbn-arxiv [25] and ogbn-products [25]. On the other hand, existing theoretical studies on GNN training [2, 56, 60, 67] overlook the batch size and fan-out size, and mainly focus on basic graph properties such as the degrees and the number of training nodes.

Theoretical exploration is required to understand how batch size and fan-out size affect GNN model performance, as empirical observations alone offer limited insights into key factors shaping training dynamics. Empirical comparisons often reveal performance differences without explaining why certain hyperparameter choices are more effective for one dataset but not on another. Theoretical analysis can potentially isolate key factors, providing insights that extend beyond specific datasets and hardware environments.

Metrics. GNN studies widely adopt time-based metrics (e.g., time-to-accuracy [3, 25, 28, 78, 80] and epoch time [6, 62–64]) to evaluate model convergence performance, ignoring the influence of hardware differences. Different hardware environment, e.g., communication bandwidth and computational power (GPUs or CPUs), can skew the comparison results based on time-based metrics, leading to algorithm and system efficiency improvement conclusions that do not generalize to new hardware environments. Hence, we are motivated to employ additional hardware-independent metrics in assessing and validating convergence behaviors predicted by our theoretical analysis.

Research question 2: We propose our second question: "How do graph-based hyperparameters (e.g., batch size and fan-out size) and hardware environment (e.g., cluster size) impact the system efficiency and model performance?" Addressing this question requires both theoretical analyses and empirical studies, with hardware-independent metrics necessary to validate the theoretical findings.

2.3.3 *Challenges.*

Up till now, an integrated empirical and theoretical study is still lacking, that explores the model performance and system efficiency of full-graph and mini-batch GNN training on graph-based hyperparameters. Challenges arise on both empirical and theoretical sides.

Empirical study: Metrics in evaluating convergence performance. Identifying suitable hardware-independent metrics remains a challenge, requiring interpretability and computational efficiency. Some hardware-independent metrics, such as decay of gradient norm [10] and convergence curve [12], are complex to understand, computationally expensive, and ineffective in capturing overall convergence in noisy settings with high variance in gradient estimates or data. In contrast, iteration-to-loss (i.e., the number of iterations needed to achieve a target training loss) and iteration-to-accuracy (i.e., the number of iterations needed to achieve a target validation accuracy) are intuitive to understand and are universal measures of convergence without requiring calculations. Moreover, iteration-based metrics (e.g., iteration-to-loss and iteration-to-accuracy) are directly supported by theoretical analysis, as many theoretical frameworks [2, 4, 21, 37, 53, 66, 68, 82] analyze convergence dynamics in terms of the number of discrete iterations. We adopt iteration-to-loss and iteration-to-accuracy as hardware-independent metrics alongside time-to-accuracy.

Theoretical study: Convergence and Generalization. Existing theoretical analyses of GNN model convergence and generalization do not apply under practical assumptions, such as differing graph structures between training and testing datasets in mini-batch training for generalization, non-regular graphs and finite-width GNNs with non-linear activations for model performance.

For *convergence* analysis, challenges arise from the lack of full-graph training studies that simultaneously consider these practical settings, as well as the absence of mini-batch training studies. For example, Yang et al. [68] and Lin et al. [37] apply the Neural Tangent Kernel (NTK) framework through assuming infinite-width GNNs, where the NTK framework approximates the training dynamic of neural networks using kernel methods. Xu et al. [66] analyze multi-layer linear GNNs, maintaining non-linearity through a non-convex loss function. Awasthi et al. [2] employ PL conditions (i.e., the squared norm of the gradient lower bounds the loss value at any iteration) to study one-round GNNs (i.e., GNNs with one-layer neural network) with ReLU activation, simplifying the analysis to regular graphs. All these analytical studies are on full-graph training. We consider non-regular graphs and apply PL conditions in both full-graph and mini-batch training, accommodating GNNs with activation functions and finite width.

For *generalization* analysis, full-graph GNN training has been studied [14, 17–19, 32, 36, 41, 48, 52, 58] under the well-established frameworks (e.g., PAC-Bayesian framework [43]), while the previous analyses of mini-batch training impractically assume the same graph structures used in training and testing [56, 60]. We observe that inference utilizes all testing neighbors across the entire graph, whereas mini-batch training relies on sampled neighbors within limited hops. This difference among graph structures in training and testing can result in generalization performance degradation or overfitting to graph structures used in training. Instead, we seek a suitable metric to measure the difference between training and testing datasets in generalization analysis.

To *measure dataset differences,* common metrics, such as MMD [22], CMD [72] and KL divergence [33], fail to capture geometric differences between distributions (e.g., the correlations among nodes). Ma et al. [42] address distribution shifts between training and testing datasets using the l_2-norm of node feature differences, but assume the use of all neighbors across the entire graph regardless of whether during training or testing. In contrast, the Wasserstein distance [29], which measures the minimal cost of transforming one distribution into another, is well-suited for handling non-i.i.d. data with complex spatial structures. We adopt a PAC-Bayesian approach and use the Wasserstein distance to measure the differences of graph structures between training and testing datasets for generalization analysis in mini-batch GNN training.

3 PRELIMINARIES

3.1 Notation

Given a homogeneous undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $n = |\mathcal{V}|$ nodes, set n_{train} nodes in the training set and n_{test} nodes in the testing set, where $n = n_{\text{train}} + n_{\text{test}}$. We allow arbitrary subsets of nodes to be selected as the training and testing sets. Each node is an instance $(\mathbf{x}_i, y_i)$ with feature $\mathbf{x}_i$ and label y_i. Let $\mathbf{X} \in \mathbb{R}^{n\times r}$ be the feature matrix where $\mathbf{x}_i$ denotes the i-th row of $\mathbf{X}$ and r denotes the feature size. We apply uniform neighbor sampling in mini-batch training, with the batch size $b \leq n_{\text{train}}$ and the fan-out size $\beta \leq n$.

Let $\mathbf{A}$ represent the adjacency matrix composed by training nodes in the batch of a training iteration, where $\mathbf{A}^{\text{full}}_{\text{train}} \in \mathbb{R}^{n_{\text{train}}\times n}$ is for full-graph training, $\mathbf{A}^{\text{mini}}_{\text{train}} \in \mathbb{R}^{b\times n}$ is for mini-batch training, and $\mathbf{A}_{\text{test}} \in \mathbb{R}^{n_{\text{test}}\times n}$ is for testing inference. Here $\mathbf{A}^{\text{mini}}_{\text{train}}$ is a submatrix of $\mathbf{A}^{\text{full}}_{\text{train}}$. Let $\mathbf{D}^{\text{in}}$ denote a diagonal in-degree matrix where each diagnonal element is the number of incoming edges for the corresponding node. $\mathbf{D}^{\text{in,full}}_{\text{train}} \in \mathbb{R}^{n_{\text{train}}\times n_{\text{train}}}$ is for full-graph training, $\mathbf{D}^{\text{in,mini}}_{\text{train}} \in \mathbb{R}^{b\times b}$ is for mini-batch training, and $\mathbf{D}^{\text{in}}_{\text{test}} \in \mathbb{R}^{n_{\text{test}}\times n_{\text{test}}}$ is for testing. $\mathbf{D}^{\text{out}} \in \mathbb{R}^{n\times n}$ denotes the respective diagonal out-degree matrix. $\tilde{\mathbf{A}} = \left(\mathbf{D}^{\text{in}} + \mathbf{I}\right)^{-\frac{1}{2}} (\mathbf{A} + \mathbf{I}) \left(\mathbf{D}^{\text{out}} + \mathbf{I}\right)^{-\frac{1}{2}}$ is the respective normalized adjacency matrix with self-loops, where self-loops ensure that each node retains its own features during aggregation, preserving node-specific information and improving the model's learning ability. Here $\tilde{\mathbf{a}}_i$ denotes the i-th row of $\tilde{\mathbf{A}}$.

Let $\mathbf{W} \in \mathbb{R}^{h\times r}$ be the learnable model parameters of the GNN model and $\mathbf{W}^* \in \mathbb{R}^{h\times r}$ be the ground truth of $\mathbf{W}$, where the i-th row of $\mathbf{W}$ is denoted by $\mathbf{w}_i$ and h is the hidden dimension. In the transductive learning setting, our task is to predict the labels of nodes $\{\mathbf{x}_i\}_{i=n_{\text{train}}+1}^{n}$ by the GNN model trained on $\{\mathbf{x}_i\}_{i=1}^{n} \cup \{\mathbf{y}_i\}_{i=1}^{n_{\text{train}}}$, where $\mathbf{y}_i \in \mathbb{R}^{1\times K}$ is the one-hot form of the ground truth label y_i for node i and K is the number of label categories. We assume that node features are fixed, while node labels are independently sampled from distributions conditioned on the node features. This setting is widely adopted in the node classification task.

We assume that the GNN model is a one-round GNN with the ReLU activation, generating outcomes in a training iteration as:

$$\hat{\mathbf{y}}_i = \sigma\left(\tilde{\mathbf{a}}_{\text{train},i}\mathbf{X}\mathbf{W}^\top\right), \forall i \in \text{training set}, \tag{2}$$

where $\sigma(x) = \sqrt{2}\max(x, 0)$ is the ReLU activation function, and $h = K$ for one-round GNNs. The term $\tilde{\mathbf{a}}_{\text{train},i}\mathbf{X}$ represents the embedding aggregation on node i. Similarly, the GNN generates outcomes in inference as $\hat{\mathbf{y}}_i = \sigma\left(\tilde{\mathbf{a}}_{\text{test},i}\mathbf{X}\mathbf{W}^\top\right), \forall i \in \text{test set}$.

The empirical training risk for a given training iteration can be defined by MSE across the nodes:

$$\hat{L}^{\text{full}}_{\text{train}}\left(\mathbf{W}^{\text{full}}\right) = \frac{1}{2n_{\text{train}}} \sum_{i\in\text{train set}} \left\|\hat{\mathbf{y}}^{\text{full}}_i - \mathbf{y}_i\right\|_F^2 \tag{3a}$$

$$\hat{L}^{\text{mini}}_{\text{train}}\left(\mathbf{W}^{\text{mini}}\right) = \frac{1}{2b} \sum_{i\in\text{sampled nodes}} \left\|\hat{\mathbf{y}}^{\text{mini}}_i - \mathbf{y}_i\right\|_F^2 \tag{3b}$$

Similarly, the empirical testing risk can be given by $\hat{L}_{\text{test}}(\mathbf{W}) = \frac{1}{2n_{\text{test}}}\sum_{i\in\text{test set}} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_F^2$. The training loss of mini-batch training on the entire training set can be expressed as $\hat{L}_{\text{train}}\left(\mathbf{W}^{\text{mini}}\right) = \frac{1}{n_{\text{train}}}\sum_{i\in\text{train set}} \left\|\hat{\mathbf{y}}^{\text{mini}}_i - \mathbf{y}_i\right\|_F^2$, corresponding to the loss discrepancy between training and testing datasets. Practically, we use the empirical risks $\hat{L}$ to estimate the expected risks L, defined as $L(\mathbf{W}) = \mathbb{E}\left[\hat{L}(\mathbf{W})\right]$.

The GNN model parameters are updated using the GD (SGD) of the training error:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_t \nabla L_t(\mathbf{W}_t), \tag{4}$$

where η_t is the learning rate at the t-th training iteration, and $\nabla L_t(\mathbf{W})$ denotes the gradient of $L_t(\mathbf{W}_t)$ with respect to $\mathbf{W}_t$.

In this paper, we use $\|\cdot\|_2$, $\|\cdot\|$ and $\|\cdot\|_F$ to denote the 2-norm of vector, spectral norm of matrix and frobenius norm of vector, respectively. $Tr(\cdot)$ represents the trace of a matrix. The notation table is in Appendix A.

3.2 Assumptions

We make the following standard assumptions in this paper.

Assumption 3.1. The node feature $\mathbf{x}_i$ is drawn i.i.d from $N(0, \mathbf{I}_{r\times r})$ for all $i \in \mathcal{V}$.

Assumption 3.2. The rows of ground truth parameters satisfy $\|\mathbf{w}_i^*\|_2 = 1$ for all $i \in \{1, \ldots, h\}$.

Assumption 3.3. Entries of matrix $\mathbf{W}$ are initialized i.i.d. from a Gaussian distribution $N(0, \kappa^2\mathbf{I})$.

Assumption 3.4. There exists a constant $v_w > 0$ such that $\|\mathbf{W}^{\text{mini}}\|^2 \leq v_w$.

Assumption 3.5. There exists a constant $v_x > 0$ such that $\|\mathbf{X}\|^2 \leq v_x$.

Assumption 3.1 sets the distribution of node features, Assumption 3.2 limits the norm of ground truth parameters for mini-batch and full-graph training, and Assumption 3.3 assumes the distribution of initialized parameters for both training approaches, which are common assumptions in convergence analysis of GNNs [2]. Assumption 3.4 requires the parameters to be upper-bounded during mini-batch training, and Assumption 3.5 specifies the input features are bounded, which are common assumptions in generalization analysis of GNNs [19, 36, 56].

Definition 1. (Expected Loss Discrepancy [42]). *For any constant $v_u > 0$, define the expected loss discrepancy between training and testing datasets with respect to v_u before GNN training as:*

$$U(v_u) = ln\mathbb{E}_{\mathbf{W}^{\text{mini}}\sim\mathcal{P}} e^{v_u\left(L_{\text{test}}\left(\mathbf{W}^{\text{mini}}\right) - L_{\text{train}}\left(\mathbf{W}^{\text{mini}}\right)\right)}, \tag{5}$$

where the constant v_u is from the lemma 9 in Appendix B.2.1, and $\mathcal{P}$ represents the prior distribution over hypothesis space that includes all possible models that a learning algorithm can select.

The expected loss discrepancy $U(v_u)$ is used in generalization analysis, related to the difference between training and testing datasets.

Definition 2. (Distance to Training set and Testing Set [29]). *Define the distance from the training set to the testing set as the Wasserstein distance given by:*

$$\begin{aligned}\Delta(\beta, b) &= \left\{\inf_{\theta\in\Theta[\rho_{\text{train}},\rho_{\text{test}}]} \sum_{i\in\text{train set}} \sum_{j\in\text{test set}} \theta_{i,j}\delta(y_i, y_j, \beta, b)\right\} \\ &= \left\{\sup_{f(\cdot),g(\cdot)} \sum_{i\in\text{train set}} f(y_i)\rho_{\text{train}}(y_i) + \sum_{j\in\text{test set}} g(y_j)\rho_{\text{test}}(y_j)\right\},\end{aligned} \tag{6}$$

where $\rho_{train}(y_i)$ and $\rho_{test}(y_i)$ denote the probability of y_i appearing in training and testing sets, respectively. $\Theta[\rho_{train}, \rho_{test}]$ is the joint probability of ρ_{train} and ρ_{test}, and $f(\mathbf{y}_i)$ and $g(\mathbf{y}_i)$ are functions of $\mathbf{y}_i$ with $i \in \mathcal{V}$. The infimum in the first equality is conditioned on $\sum_{j\in test\ set}\theta_{i,j} = \rho_{train}(y_i)$, $\sum_{i\in train\ set}\theta_{i,j} = \rho_{test}(y_j)$, $\theta_{i,j} \geq 0$, and the supremum in the second equality is conditioned on $f(y_i) + g(y_j) \leq \delta(y_i, y_j, \beta, b)$. $\delta(\mathbf{y}_i, \mathbf{y}_j, \beta, b)$ is the distance function of any two points from training and testing sets, respectively.

We set $\delta(\mathbf{y}_i, \mathbf{y}_j, \beta, b) = \frac{v_x(v_w+1)}{2n_{\min}}\left(\delta_{i,j}^{\text{full}} + \delta_{i,j}^{\text{full-mini}}\right)$ with $n_{\min} = \min\{n_{\text{train}}, n_{\text{test}}\}$. $\delta_{i,j}^{\text{full}} = Tr\left(\left|\left(\tilde{\mathbf{a}}_{\text{test},j}^{\text{full}}\right)^\top \tilde{\mathbf{a}}_{\text{test},j}^{\text{full}} - \left(\tilde{\mathbf{a}}_{\text{train},i}^{\text{full}}\right)^\top \tilde{\mathbf{a}}_{\text{train},i}^{\text{full}}\right|\right)$, as a constant, capturing the difference of distributions between the training and testing data in full-graph training. Here $\delta_{i,j}^{\text{full-mini}} = Tr\left(\left|\left(\tilde{\mathbf{a}}_{\text{train},i}^{\text{full}}\right)^\top \tilde{\mathbf{a}}_{\text{train},i}^{\text{full}} - \left(\tilde{\mathbf{a}}_{\text{train},i}^{\text{mini}}\right)^\top \tilde{\mathbf{a}}_{\text{train},i}^{\text{mini}}\right|\right)$ reflects the difference of graph structures between full-graph training data and mini-batch training data per training iteration.

The Wasserstein distance effectively measures differences in non-i.i.d. data, particularly regarding geometric variations. A dual representation is provided in Eq (6).

4 THEORETICAL ANALYSIS

We next represent our theoretical analysis of GNN convergence and generalization. In convergence analysis (Sec 4.1), we extend regular graphs to non-regular graphs. We consider practical GNN settings by utilizing the PL condition on GNNs with finite width and ReLU activation. We then discuss the impact of message-passing process in GNN convergence from the commonly adopted perspective of gradient variance. In generalization analysis (Sec 4.2), we consider the difference of graph structures between training and testing datasets and use Wasserstein distance to measure this difference. We then apply PAC-Bayesian framework to inspect GNN generalization.

4.1 Convergence

Loss expression on non-regular graphs. We first simplify expressions for the loss and the gradient in both full-graph and mini-batch training on non-regular graphs. We use Γ to rewrite the sum of $\hat{\mathbf{y}}_i^2$ in MSE defined in Eq (3), where Γ depends on $\tilde{\mathbf{a}}_{\text{train},i}\mathbb{1}$. Note that the term $\tilde{\mathbf{a}}_{\text{train},i}\mathbb{1}$ extends the analysis from regular graphs to non-regular graphs, as it directly captures the information from each node and accommodates the diverse node connections in non-regular graphs. This term $\tilde{\mathbf{a}}_{\text{train},i}\mathbb{1}$ is applicable only to GNNs using message passing, as it is derived from the embedding aggregation, different from DNNs. See Appendix B.1.2 for detailed proof.

PL condition. We then demonstrate that the iterates $\mathbf{w}_t$ satisfy the PL inequality [51], which means that the squared norm of the gradient at any iteration is lower bounded by the loss value scaled by a factor μ, such that $\|\nabla L_{\text{train}}\|^2 \geq \mu L_{\text{train}}$. Here $\mu^{\text{mini}} \geq v_1\frac{\epsilon^2\Gamma}{h^2b^2}$ and $\mu^{\text{full}} \geq v_2\frac{\epsilon^2\Gamma}{h^2n_{\text{train}}^2}$ with $\epsilon \in (0,1)$ and absolute constants v_1, v_2. Furthermore, we prove that the GD and SGD can escape quickly from an initial phase when the PL condition does not hold. See Appendix B.1.2 for proof.

4.1.1 *Convergence results.*

Finally, we prove the convergence theorems for full-graph and mini-batch training, respectively. For clarity, the informal theorems are presented in this section, with detailed proofs provided in Appendix B.1.3.

Informal Theorem 1. (Convergence of Full-graph Training). *If the conditions outlined in Appendix B.1.3 hold, then the training loss satisfies $L_{train}^{full}\left(\mathbf{W}_T^{full}\right) \leq \epsilon^2$, provided that the number of iterations $T \geq \log\frac{h}{\epsilon}O\left(\frac{h^2o(n_{train}^{3/2})}{\epsilon^2n^{1/2}}\right)$ for any $\epsilon \in (0,1)$, in full-graph GNN training.*

Informal Theorem 2. (Convergence of Mini-batch Training) *If the conditions outlined in Appendix B.1.3 hold, then the training* *loss satisfies* $L_{train}^{mini}\left(\mathbf{W}_T^{mini}\right) \leq \epsilon^2$, *provided that the number of iterations* $T \geq \log \frac{h}{\epsilon} O\left(\frac{h^2 o(b^{3/2})}{\epsilon^2 \beta^{1/2}}\right)$ *for any* $\epsilon \in (0, 1)$, *in mini-batch GNN training.*

We observe that when the fan-out size β reaches n and the batch size b reaches n_{train}, the lower bound on the number of iterations to convergence in mini-batch training matches that of full-graph training.

Takeaway 4.1. Informal Theorem 1 shows that an increase in the number of training nodes n_{train} necessitates more iterations to convergence in full-graph training with MSE.

Takeaway 4.2. Informal Theorem 2 shows that, for a fixed fan-out size, an increase in batch size b necessitates more iterations to convergence in mini-batch training with MSE.

Takeaway 4.3. Informal Theorem 2 shows that, for a fixed batch size, a decrease in fan-out size β necessitates more iterations to convergence in mini-batch training with MSE.

4.1.2 Discussion. Using the results in Sec 4.1.1, we analyze the impact of the message-passing process on GNN convergence.

Discussion 4.1: batch size and fan-out size. In Informal Theorem 2, we observe opposite convergence trends with respect to batch size and fan-out size in mini-batch training using MSE. The common intuitive explanation posits that increasing batch size and fan-out size reduces gradient variance, resulting in fewer iterations to convergence [15, 26, 35, 39, 84]. This explanation does not fully account for the observed trends, necessitating additional consideration of the impact of message passing on the loss and gradient.

With a fixed fan-out size, increasing the batch size leads to greater overlap among the neighbors of training nodes due to dependencies in the graph structure. This overlap may cause over-smoothing and information redundancy, particularly when using MSE. Since the gradient of MSE is linearly related to the error, over-smoothing reduces the effective gradient and weakens parameter updates during training. Hence, GNNs with a larger batch size requires more iterations for convergence when using MSE. This explains why our findings diverge from expectations based solely on gradient variance considerations.

Conversely, with a fixed batch size, increasing the fan-out size allows each node to aggregate more neighbors, enriching the node's embedding and enhancing the effective gradient even when using MSE. Therefore, increased fan-out size with the reduced gradient variance leads to fewer iterations for GNN convergence.

Discussion 4.2: MSE v.s. CE in GNNs. Different from the MSE, the gradient of cross entropy (CE) is nonlinearly related to the error. Therefore, the gradient magnitude of CE remains large enough to maintain effective updates, even with the over-smoothing caused by increased batch sizes for a fixed fan-out size. Consequently, as larger batch sizes or larger fan-out sizes reduce gradient variance, GNNs trained with CE require fewer iterations for convergence.

Discussion 4.3: DNNs v.s. GNNs in convergence. Different from GNNs, DNNs do not require a message-passing process to aggregate neighbors. Therefore, increasing the batch size in DNNs only reduces gradient variance and stabilizes the optimization process, leading to fewer iterations required for convergence.

Takeaway 4.4. Discussion 4.2 concludes that a decrease in batch size b or fan-out size β necessitates more iterations for the convergence of mini-batch GNN training with CE.

Takeaway 4.5. Discussions 4.1-4.3 highlight that the message-passing process affects GNN convergence trends, resulting in a divergence from DNN convergence behavior under varying batch sizes when using MSE.

Moreover, we provide a brief analysis of these discussions under our theoretical framework in Appendix B.1.4.

4.2 Generalization of Mini-batch Training

Bound for expected loss discrepancy. We first demonstrate that the expected loss discrepancy $U(v_u)$ is bounded by the Wasserstein distance $\Delta(\beta, b)$ from the training graph to the testing graph, such that $U(v_u) \leq v_u \Delta(\beta, b)$. This bound suggests that the more similar the training and testing graph structures are, the smaller the expected loss discrepancy is. Moreover, we observe that the Wasserstein distance $\Delta(\beta, b)$ decreases as the fan-out size β or the batch size b increases. See Appendix B.2.1 for proof.

We then apply the PAC-Bayesian framework to prove the generalization bound by involving the expected loss discrepancy for mini-batch GNN training.

Informal Theorem 3. (Generalization bound for GNNs). *If the conditions outlined in Appendix B.2.1 hold, for any* $v_u > 0$, *after sufficient iterative updates, we have:*

$$L_{\text{test}}\left(\mathbf{W}^{\text{mini}}; Q\right) - \hat{L}_{\text{train}}\left(\mathbf{W}^{\text{mini}}; Q\right)$$
$$\leq \frac{1}{v_u}\left(1 + \frac{h}{\kappa^2} o\left(\frac{1}{b}\right) + ln\frac{1}{v_G} + \frac{v_u^2}{4n_{\text{train}}} + v_u \Delta(\beta, b)\right),$$

where Q *represents the posterior distribution over hypothesis space, and* $v_G \in (0, 1)$ *is a constant.*

In Informal Theorem 3., we obtain the difference between the prior and posterior distributions over the hypothesis space as $1 + \frac{h}{\kappa^2} o\left(\frac{1}{b}\right)$, capturing the change in model parameters distribution before and after training. This difference, along with the terms $\ln \frac{1}{v_G}$ and $\frac{v_u^2}{4n_{\text{train}}}$, are common factors in PAC-Bayesian generalization analysis for i.i.d. data.

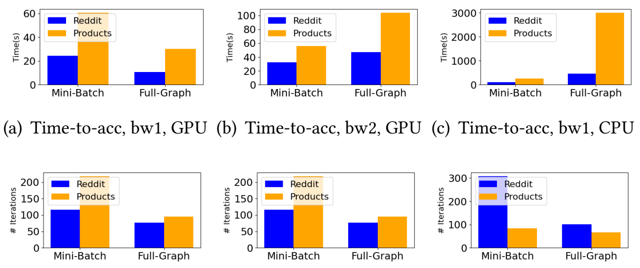
Takeaway 4.6. Informal Theorem 4 shows that an increase in batch size b, fan-out size β, or the number of training nodes n_{train} results in a decreased generalization gap.

Discussion 4.5: GNNs v.s. DNNs in Generalization. GNNs and DNNs exhibit the same generalization trend at small batch sizes [69, 70]. However, performance degradation occurs in DNNs at very large batch sizes [69, 70], as large batches tend to converge to sharper minima, which impairs generalization [5, 30]. Therefore, we remain cautious about the validity of Takeaway 4.6 at very large batch and fan-out sizes when considering performance degradation. We conduct the experiments to study this problem.

5 EXPERIMENTS

We conduct experiments to compare the model performance (i.e., convergence and generalization) and system efficiency of full-graph and mini-batch GNN training, targeting answering the two research questions in Sec 1.

We first explain the rationale for using the metrics (e.g., iteration-to-accuracy) in Sec. 5.2. Next, we compare mini-batch and full-graph training in model performance and system efficiency by incrementally increasing the batch size and fan-out size (Sec. 5.3). Then, we



(a) Time-to-acc, bw1, GPU (b) Time-to-acc, bw2, GPU (c) Time-to-acc, bw1, CPU

(d) Iter-to-acc, bw1, GPU (e) Iter-to-acc, bw2, GPU (f) Iter-to-acc, bw1, CPU

Figure 2: Time-to-acc and iteration-to-acc in mini-batch and full-graph training with varying bandwidths (i.e., two inter-GPU bandwidth values: bw1=infinity > bw2=900GB/s) and computational capacities (i.e., GPU and CPU with 512GB of host memory).

study the impact of cluster sizes on convergence and system efficiency (Sec. 5.4), as well as validate our theoretical analysis results across various graph properties (i.e., the number of training nodes) and graph-based hyperparameters (Sec. 5.5). Finally, we provide suggestions on selecting batch size and fan-out size to balance model performance and system efficiency (Sec. 5.6).

5.1 Experiment setup

Testbed. The experiments, except those on the ogbn-papers100M, are conducted on a machine with 512GB of host memory and four NVIDIA A100 GPUs, each with 40GB of memory, inter-connected via 900GB/s NVLink 4.0. The experiments on ogbn-papers100M are run on two machines without GPUs, each equipped with 1024GB of host memory and an interconnect bandwidth of 50 Gbps.

Models and datasets. We conduct experiments on four real-world datasets: reddit [23], ogbn-arxiv [25], ogbn-products [25] and ogbn-papers100M [25], with the basic information and train/ validate/test splits presented in Appendix C. We train three representative GNN models: GCN [76], GraphSAGE [23] with mean aggregation, and GAT [59] with 2 heads for ogbn-papers100M and 4 heads for the other datasets.

Metrics. We evaluate convergence performance using three metrics: iteration-to-loss, iteration-to-accuracy, and time-to-accuracy. These metrics measure training progress towards a target convergence point in terms of training loss or validation accuracy. For all GNN models and datasets except ogbn-papers100M, the target training loss is defined as the maximum loss observed over 100 consecutive iterations at the smallest batch size, provided that the variance of these loss values is below 5×10^{-4}. Similarly, the target validation accuracy is defined as the minimum accuracy over 100 consecutive iterations at the smallest batch size, provided that the variance of these accuracies is below 4×10^{-4}. Note that the defined target training loss and the defined target validation accuracy are applied across all hyperparameter settings for the specific model and dataset. The smallest batch size is chosen because training losses and evaluation accuracies exhibit the most significant fluctuations under the smallest batch size. For ogbn-papers100M, training is limited to 200 iterations due to the extremely large graph size and training time constraints. For generalization, test accuracy is used as the metric in the training iteration. For system efficiency, we measure the training throughput in terms of the number of nodes processed per second (number of nodes/s). More training settings are in Appendix C.

Table 1: Best test accuracies of full-graph and mini-batch training of multi-layer GraphSAGE model without dropout layers after graph-based hyperparameter tuning.

Datasets	Full-graph	Mini-batch
Reddit	96.13	**96.32**
Ogbn-arxiv	70.96	**71.16**
Ogbn-products	77.92	**78.80**
Ogbn-papers100M	**59.54**	58.52

5.2 Metrics: iteration-to-accuracy

This section explains the rationale for using iteration-to-accuracy as a convergence metric instead of time-to-accuracy. Since iteration-to-loss is derived from theoretical analysis, we do not provide further explanation here.

Figure 2 illustrates time-to-accuracy and iteration-to-accuracy with the two training approaches under different bandwidths and computational capacities. The vanilla distributed system (i.e., the standard implementation without any specialized optimizations or modifications) is used for full-graph training, and the Distributed Data Parallel (DDP) technique in PyTorch [34] is applied for mini-batch training with a total batch size of 2000 per iteration. We train a 3-layer GraphSAGE model on Reddit and a 3-layer GCN model on ogbn-products. To simulate infinite bandwidth (i.e., bw1), we use a single GPU or CPU. For limited bandwidth (i.e., bw2), we use two GPUs on the same machine interconnected via 900GB/s NVLink.

For time-to-accuracy, mini-batch training underperforms full-graph training under a single GPU, whereas mini-batch training performs better than full-graph training under two GPUs or a single CPU. In contrast, iteration-to-accuracy results remain consistent across different hardware environments. Specifically, we observe that the variation in iteration-to-accuracy across hardware environments is no more than 41.28%, while the variation in time-to-accuracy can reach up to 2787.05%. This indicates that convergence performance measured on time-to-accuracy cannot generalize across different hardware environments.

Takeaway 5.1. Iteration-to-accuracy as a metric in convergence evaluation is more hardware-independent than time-based metrics (e.g., time-to-accuracy). Iteration-to-accuracy can isolate pure algorithmic efficiency (how effectively each iteration improves accuracy) from system-level factors such as computational power (e.g., CPU or GPU) and communication bandwidth, enabling a clearer understanding of GNN convergence behaviors. Furthermore, when scaling to large graphs, with known iteration-to-accuracy, convergence time can be estimated on any new hardware environment by simply profiling the time of one training iteration.

5.3 Full-graph vs. Mini-batch Training

Table 1 presents an overall comparison of generalization performance between full-graph and mini-batch training after tuning graph-based hyperparameters using grid search. The specific hyperparameter values are represented in the Sec 5.3.1. We train a three-layer GraphSAGE model with hiddien dimension 256 except for ogbn-papers100M. Due to limited resources, we use two hidden layers with hidden dimension 128 for the ogbn-papers100M dataset, which may constrain the representation capacity, especially in mini-batch training with small fan-out sizes. Note that dropout layers are excluded as they randomly set a fraction of neurons to zero during forward pass, obscuring the true effect of graph-based hyperparameters. We find that the best accuracy achieved by mini-batch training closely approximates that of full-graph training, with a difference of no more than 1.74%. This suggests that both approaches enable comparable generalization performance.

We further investigate the generalization by incrementally increasing the batch size and fan-out size until mini-batch training transitions into full-graph training. We examine only a subset of fan-out sizes for the ogbn-papers100M due to resource constraints.

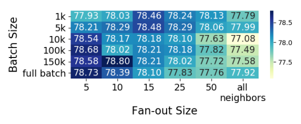
5.3.1 Generalization. Figure 3 shows that when using CE, test accuracies increase with batch size for fixed small fan-out sizes. However, for fixed large fan-out sizes (typically more than 15 on these datasets), test accuracies initially increase with batch size and then decrease. Similarly, for fixed small batch sizes, test accuracies first increase and then decrease with increasing fan-out sizes; with fixed large batch sizes (exceeding half of the training nodes), test accuracies consistently decrease as fan-out sizes increase.

The performance degradation occurs because large-batch sizes tend to lead to convergence to sharp minima, resulting in poorer generalization [30]. Since both larger batch and fan-out sizes reduce gradient variance, we infer that large fan-out sizes lead to similar generalization issues. Moreover, we observe that performance degradation is more severe with large fan-out sizes than with large batch sizes, since performance begins to degrade when fan-out sizes reach large values even at fixed small batch sizes, whereas the performance still improves with larger batch sizes for fixed small fan-out sizes. This occurs because aggregating information from too many neighbors can lead to overfitting to the training data, reducing the model's ability to generalize to unseen data.

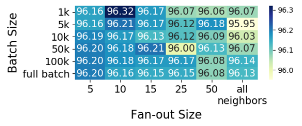
Figure 4 shows that when using MSE, test accuracies increase with both batch size and fan-out size, consistent with our theoretical results. The performance degradation does not occur because MSE produces flatter minima due to weaker gradients near prediction boundaries, resulting in flatter loss surfaces with more stationary points [5]. However, the accuracies with MSE are consistently lower than those achieved with CE, as the loss landscape of CE is more searchable for classification tasks [5].

Takeaway 5.2. Generalization performance degradation is more severe under large fan-out sizes compared to large batch sizes for GNNs trained with CE. Here the large batch sizes refer to more than half of the training nodes in a mini-batch and the large fan-out sizes are more than 15 for datasets with an average degree of less than 50. These findings suggest that increasing batch size improves generalization more effectively than increasing fan-out size. Moreover, CE consistently outperforms MSE in generalization.

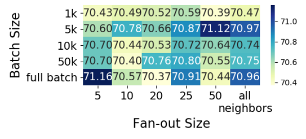
5.3.2 Convergence. Figure 5 shows that when using CE, iteration-to-loss for convergence decreases with increased batch sizes at fixed fan-out sizes or with increased fan-out sizes at fixed batch sizes. When using MSE, iteration-to-loss decreases with smaller batch sizes at fixed fan-out sizes or with larger fan-out sizes at fixed batch



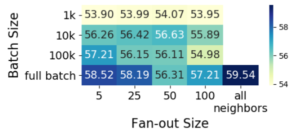
(a) Products



(b) Reddit

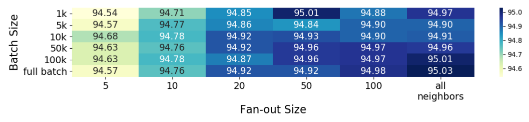


(c) Arxiv

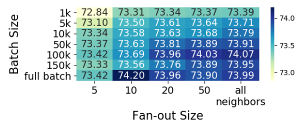


(d) Papers100M

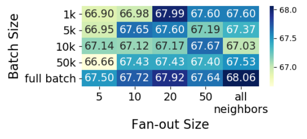
Figure 3: Test accuracies of GraphSAGE trained with *CE* across varying batch sizes and fan-out sizes.



(a) Reddit



(b) Products



(c) Arxiv

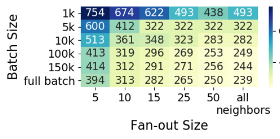
Figure 4: Test accuracies of GraphSAGE trained with *MSE* across varying batch sizes and fan-out sizes.

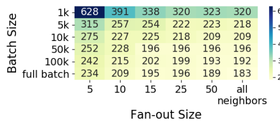
sizes, as shown in Figure 6. Both trends align with our theoretical analysis in Takeaways 4.3-4.4 in Sec. 4.

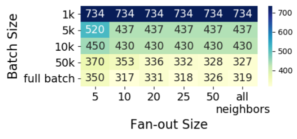
Reaching a target validation accuracy (e.g., iteration-to-accuracy and time-to-accuracy) is more complex than reaching a target training loss (e.g., iteration-to-loss), as these metrics also depend on generalization performance. For MSE, iteration-to-accuracy shows no clear trend in Figure 8, since smaller fan-out sizes can accelerate training convergence but reduce generalization. The trend for time-to-accuracy in Figure 10 is initially unstable due to inconsistent iteration-to-accuracy results but increases with larger batch and fan-out sizes due to more computational resource demand. When using CE, iteration-to-accuracy in Figure 7 follows the same trend as iteration-to-loss, while time-to-accuracy in Figure 9 decreases initially and then rises with increasing batch and fan-out sizes.

Takeaway 5.3. Larger fan-out sizes can accelerate GNN convergence under sufficient computational capacity and large communication bandwidth, regardless of whether MSE or CE is used. However, larger batch sizes slow down convergence of GNNs trained with MSE, different from the DNN convergence behaviors. These findings imply that increasing fan-out size is generally more effective in expediting GNN convergence than increasing batch size.

5.3.3 System efficiency. Figures 11 and 12 show the training throughput under both loss functions, when the GraphSage model is trained on a single GPU, except for the ogbn-papers100M dataset where two machines are used. We observe that the throughput increases with larger batch sizes given the fixed cluster size, as the fixed


(a) Products


(b) Reddit


(c) Arxiv

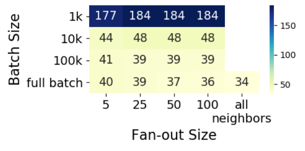
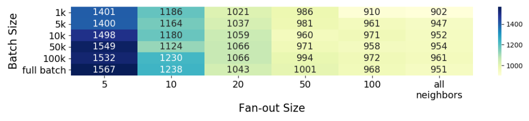
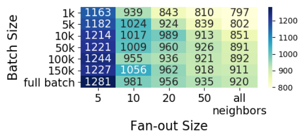

(d) Papers100M

Figure 5: Iteration-to-loss of GraphSAGE trained with *CE* across varying batch sizes and fan-out sizes.


(a) Reddit


(b) Products

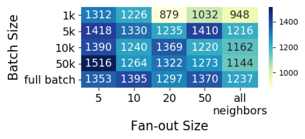
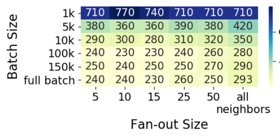
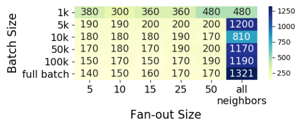
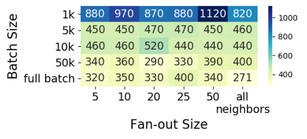

(c) Arxiv

Figure 6: Iteration-to-loss of GraphSAGE trained with *MSE* across varying batch sizes and fan-out sizes.


(a) Products


(b) Reddit


(c) Arxiv

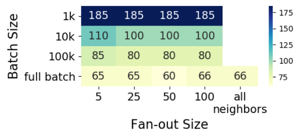
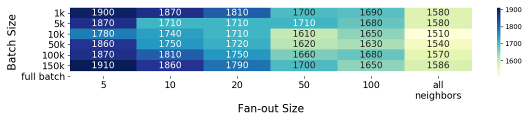
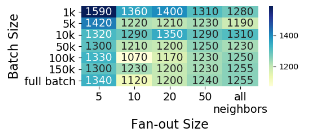

(d) Papers100M

Figure 7: Iteration-to-acc of GraphSAGE trained with *CE* across varying batch sizes and fan-out sizes.


(a) Reddit


(b) Products

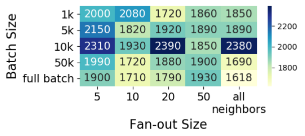
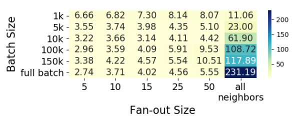
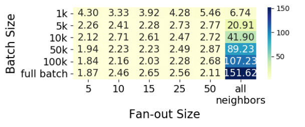
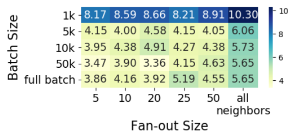

(c) Arxiv

Figure 8: Iteration-to-acc of GraphSAGE trained with *MSE* across varying batch sizes and fan-out sizes.


(a) Products


(b) Reddit


(c) Arxiv

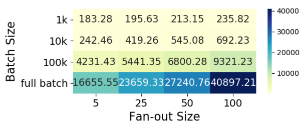
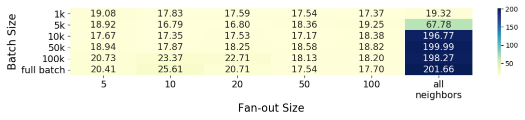
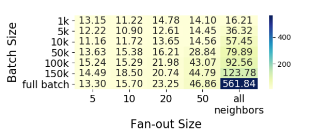

(d) Papers100M

Figure 9: Time-to-accuracy (s) of GraphSAGE trained with *CE* across varying batch sizes and fan-out sizes.


(a) Reddit


(b) Products

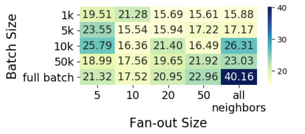
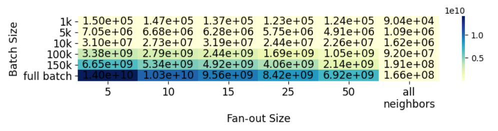

(c) Arxiv

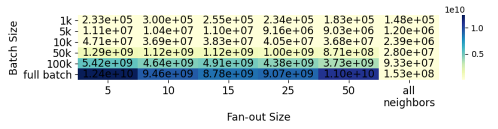
Figure 10: Time-to-accuracy (s) of GraphSAGE trained with *MSE* across varying batch sizes and fan-out sizes.

computation (e.g., parameter updates and gradient computations) is distributed across more data. However, the throughput decreases as fan-out size increases due to the higher computational demands in message passing. Overall, mini-batch training outperforms full-graph training in this system efficiency.

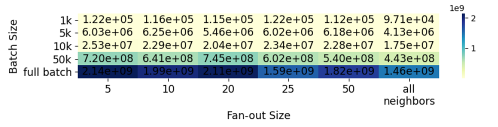
Takeaway 5.4. Full-graph training, viewed as mini-batch training with the largest batch and fan-out sizes, does not always yield superior performance compared to smaller mini-batch settings. Carefully tuning these graph-based hyperparameters can produce more favorable trade-offs in real-world scenarios, balancing generalization, convergence, and system efficiency. The GNN performance behaviors can be summarized as follows: for fixed small batch sizes, larger *fan-out sizes* improve convergence and generalization but reduce system efficiency, with very large fan-out sizes degrading generalization; for fixed small fan-out sizes, larger *batch sizes* slow down convergence with MSE while enhancing generalization and improving system efficiency given fixed cluster size with sufficient resources, with very large batch sizes degrading generalization.



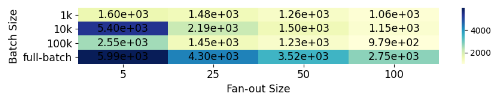
(a) Products



(b) Reddit

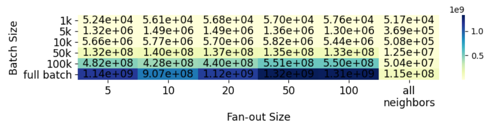


(c) Arxiv

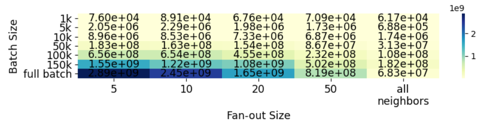


(d) Papers100M

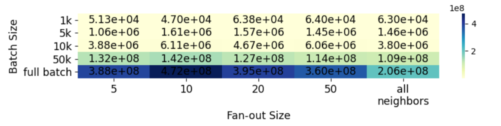
Figure 11: Training throughput (# nodes/s) of GraphSAGE trained with *CE* across varying batch sizes and fan-out sizes.



(a) Reddit



(b) Products



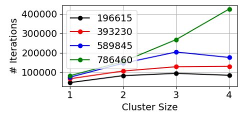
(c) Arxiv

Figure 12: Training throughput (# nodes/s) of GraphSAGE trained with *MSE* across varying batch sizes and fan-out sizes.

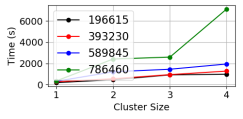
5.4 Cluster Size

We examine the impact of cluster sizes on convergence and system efficiency. Note that the related generalization performance has been thoroughly studied in previous works [3, 42, 56]. We run DDP [34] on four A100 GPUs, with each handling a batch size of 1,000.

Figure 13 shows the convergence and Figure 14 illustrates the throughput across varying numbers of training nodes and cluster sizes on GraphSAGE trained with MSE. As cluster size increases, the time-to-accuracy and iteration-to-accuracy increase, and the gaps in time and iterations to convergence widen among different numbers of training nodes. This indicates that for larger training graphs, increasing cluster size slows down convergence, especially under time-based metrics. As a larger cluster size indicates a larger overall batch size, these results provide further evidence for Takeaway 5.3., demonstrating that increasing the batch size is less effective than increasing the fan-out size. Moreover, we observe that for fixed fan-out sizes, larger batch sizes with a larger cluster size decreases the system efficiency under sufficient resources, complementing the study of system efficiency given varying batch size in Sec. 5.3.3.



(a) Iteration-to-acc



(b) Time-to-acc (s)

Figure 13: Convergence of GraphSAGE on ogbn-products in mini-bacth training on different numbers of training nodes when varying cluster sizes.

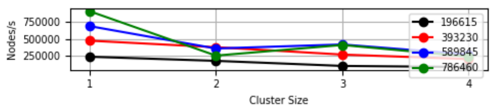


Figure 14: Training throughput (number of nodes/s) of GraphSAGE on ogbn-products in mini-bacth training on different numbers of training nodes when varying cluster sizes.

5.5 Graph property and Graph-based Hyperparameters

We further conduct ablation studies on different graph properties (i.e., number of training nodes) and graph-based hyperparameters (i.e., batch size and fan-out size) in mini-batch training with MSE, as described in Takeaways 4.1-4.3 in Sec. 4.1 and Takeaway 4.6 in Sec. 4.2. All experiments here are conducted on one-layer GNN models (GraphSAGE, GCN and GAT). Due to space limit, the figures and experimental details are in Appendix C.

5.5.1 Number of training nodes. Figure 15 and Figure 16 in Appendix C show iteration-to-loss and time-to-accuracy for different models on ogbn-product dataset in mini-batch and full-graph training across different *numbers of training nodes*. For a fixed fan-out size of 5, the batch size is set to 1000 for iteration-to-loss evaluation and 500 for time-to-accuracy. We observe that both mini-batch training and full-graph training require more iterations when the number of training nodes increases, consistent with the theoretical analysis. The impact on full-graph training is more pronounced than on mini-batch training, as the batch size remains fixed per iteration in mini-batch training, while the number of training nodes increases per epoch of full-graph training.

5.5.2 Batch Size. Figure 17 in Appendix C illustrates the iteration-to-loss for different models on different datasets and learning rates when varying *batch sizes*. We observe that larger batch sizes result in more iterations to reach the target training loss, consistent with our theoretical analysis. Figure 18 in Appendix C illustrates the test accuracies for different models on different real-world datasets and learning rates when varying *batch sizes*. We observe that larger batch sizes improve accuracy, aligning with our theoretical analysis. In addition, GAT exhibits a slightly different generalization trend, as its attention mechanism is sensitive to noise in larger batch sizes.

5.5.3 Fan-out Size. Figure 19 in Appendix C shows the iteration-to-loss for different models on different datasets and learning rates when varying *fan-out sizes*. We observe that mini-batch training with larger fan-out sizes generally requires fewer iterations, aligning with our theoretical analysis. Figure 20 in Appendix C shows the test accuracies for different models on different real-world datasets and learning rates when varying *fan-out sizes*. Test accuracies increase with larger fan-out sizes, consistent with our theoretical analysis. In addition, we observe that on medium datasets (e.g., reddit, ogbn-products, ogbn-arxiv), the benefit of increasing fan-out size on generalization plateaus at 15.

5.6 Discussion

Based the observations above, we provide suggestions on selecting batch size and fan-out size to balance model performance and system efficiency.

Accelerating convergence. Increasing fan-out size is more effective than increasing batch size. Moreover, enhancing memory and computational capacity of a single GPU is more advantageous in expediting convergence than increasing cluster size. Using larger fan-out sizes for fixed small batch sizes can improve convergence speed, regardless of whether MSE or CE is used, but requires more resources to mitigate reduced throughput. In contrast, increased batch sizes and larger cluster sizes significantly slow down convergence when using MSE.

Improving generalization. Increasing batch size is more effective than increasing fan-out size, as generalization begins to degrade when fan-out sizes reach large values even at fixed small batch sizes, whereas generalization continues to improve with larger batch sizes given fixed small fan-out sizes. However, larger batch sizes may require larger cluster sizes, necessitating more system optimization to address high communication overhead.

Training skills. Adaptive adjustments of fan-out sizes can be advantageous in GNN model performance, e.g., starting with a large fan-out size to accelerate convergence and then reducing it to prevent generalization degradation. Further, optimizers of GNNs can consider the impact of the message-passing, especially the fan-out size. For example, we can scale the learning rate proportionally to the fan-out size, or use node-wise adaptive learning rates based on each node's actual fan-out size, as sparse graphs have nodes with fewer neighbors than the specified fan-out size.

Practical graph-based hyperparameter selection. We recommend keeping the batch size below half of the training nodes and the fan-out size under 15 for datasets with an average degree less than 50, to avoid generalization degradation and balance the trade-offs in system efficiency and model performance.

6 FUTURE WORK

Sampling methods. We focus on uniform neighbor sampling in mini-batch training. Many other sampling methods [9, 11, 23, 73, 83] have been proposed at the layer- or subgraph-level to enhance performance, leading to different graph-based hyperparameters. Further research can explore how these sampling methods perform compared to full-graph training, potentially offering fresh insights.

Multi-layer GNN models with activation functions. We focus on a one-layer GNN with ReLU activation in theoretical analysis. For generalization, our findings can be easily extended by re-deriving the difference between prior and posterior distributions over hypothesis space for multi-layer GNNs. For convergence, future work can explore the convergence of multi-layer GNNs under practical settings, particularly with non-linear activations.

Link prediction tasks. We focus on node classification tasks in GNN training, which can be easily extended to graph classification. Different from node classification, link prediction tasks use node pairs (connected and unconnected) for edge prediction. Training data typically includes existing edges as positive samples and nonexistent edges or invalid node pairs as negative samples. Future work can explore link prediction using both training methods.

Inductive GNN tasks. We focus on transductive GNN tasks. Unlike transductive tasks, inductive tasks apply different graphs between testing and training. For convergence, our analysis can be applied to inductive tasks without considering the testing graphs. For generalization, our analysis can be easily extended to inductive tasks by revising $\delta_{i,j}^{\text{full}}$ in the Wasserstein distance to consider graph structure differences between testing and training graphs.

Heterogeneous graphs. Different from homogeneous graphs, heterogeneous graphs require specialized handling to address different types of nodes and edges, involving distinct aggregation and transformation functions for each type, such as using separate neural networks for different edge types. This can be explored.

7 CONCLUSION

We provide systematic empirical study and theoretical analysis of full-graph and mini-batch GNN training, focusing on graph-based hyperparameters (e.g., batch size and fan-out size) and hardware environments (e.g., cluster size) on homogeneous bidirectional graphs. Theoretically, we analyze GNN convergence and generalization under practical settings (e.g., non-regular graphs and GNNs with finite width and non-linear ReLU activations), using the Wasserstein distance to measure graph structure differences between training and testing graphs. Empirically, we highlight the importance of using iteration-to-loss and iteration-to-accuracy, to ensure our results generalize across various hardware environments.

This study demonstrates that larger batch sizes slow down GNN training convergence with MSE, differing from DNN training. We find that for a fixed batch size, GNNs with larger fan-out sizes require fewer iterations to convergence, regardless of whether CE or MSE is used. Furthermore, we identify full-graph training, viewed as mini-batch with the largest batch and fan-out sizes, does not always yield superior model performance or system efficiency compared to smaller mini-batch settings. Carefully tuning these graph-based hyperparameters can produce more favorable trade-offs in real-world scenarios, balancing generalization, convergence, and system efficiency. Our work highlights the importance of tuning graph-based hyperparameters in balancing system efficiency and model performance in GNN training. These findings are crucial for advancing the development of both training systems.

REFERENCES

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2019. A convergence theory for deep learning via over-parameterization. In *International conference on machine learning*. PMLR, n.p., 242–252.

[2] Pranjal Awasthi, Abhimanyu Das, and Sreenivas Gollapudi. 2021. A convergence analysis of gradient descent on graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 20385–20397.

[3] Saurabh Bajaj, Hui Guan, and Marco Serafini. 2024. Graph Neural Network Training Systems: A Performance Comparison of Full-Graph and Mini-Batch. *arXiv preprint arXiv:2406.00552* n.v., n.p. (2024), n.p.

[4] Raef Bassily, Mikhail Belkin, and Siyuan Ma. 2018. On exponential convergence of sgd in non-convex over-parametrized learning. *arXiv preprint arXiv:1811.02564* (2018).

[5] Anna Sergeevna Bosman, Andries Engelbrecht, and Mardé Helbig. 2020. Visualising basins of attraction for the cross-entropy and the squared error neural network loss functions. *Neurocomputing* 400 (2020), 113–136.

[6] Zhenkun Cai, Xiao Yan, Yidi Wu, Kaihao Ma, James Cheng, and Fan Yu. 2021. DGCL: An efficient communication library for distributed GNN training. In *Proceedings of the Sixteenth European Conference on Computer Systems*. n.p., n.p., 130–144.

[7] Zhenkun Cai, Qihui Zhou, Xiao Yan, Da Zheng, Xiang Song, Chenguang Zheng, James Cheng, and George Karypis. 2023. DSP: Efficient GNN training with multiple GPUs. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*. n.p., n.p., 392–404.

[8] Jingji Chen, Zhuoming Chen, and Xuehai Qian. 2023. GNNPipe: Accelerating Distributed Full-Graph GNN Training with Pipelined Model Parallelism. *arXiv preprint arXiv:2308.10087* n.v., n.p. (2023), n.p.

[9] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247* n.v., n.p. (2018), n.p.

[10] Zhaodong Chen, Lei Deng, Bangyan Wang, Guoqi Li, and Yuan Xie. 2020. A comprehensive and modularized statistical framework for gradient norm equality in deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 1 (2020), 13–31.

[11] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. n.p., n.p., 257–266.

[12] Daeyoung Choi, Hyunghun Cho, and Wonjong Rhee. 2018. On the difficulty of DNN hyperparameter optimization using learning curve prediction. In *TENCON 2018-2018 IEEE Region 10 Conference*. IEEE, 0651–0656.

[13] Weilin Cong, Rana Forsati, Mahmut Kandemir, and Mehrdad Mahdavi. 2020. Minimal variance sampling with provable guarantees for fast training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1393–1403.

[14] Weilin Cong, Morteza Ramezani, and Mehrdad Mahdavi. 2021. On provable benefits of depth in training graph convolutional networks. *Advances in Neural Information Processing Systems* 34 (2021), 9936–9949.

[15] Weilin Cong, Morteza Ramezani, and Mehrdad Mahdavi. 2021. On the Importance of Sampling in Training GCNs: Tighter Analysis and Variance Reduction. *arXiv preprint arXiv:2103.02696* (2021).

[16] Amit Daniely, Roy Frostig, and Yoram Singer. 2016. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. *Advances in neural information processing systems* 29 (2016), n.p.

[17] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. 2019. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems* 32 (2019), n.p.

[18] Ran El-Yaniv and Dmitry Pechyony. 2009. Transductive rademacher complexity and its applications. *Journal of Artificial Intelligence Research* 35 (2009), 193–234.

[19] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. 2020. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*. PMLR, n.p., 3419–3430.

[20] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, n.p., 1263–1272.

[21] Noah Golmant, Nikita Vemuri, Zhewei Yao, Vladimir Feinberg, Amir Gholami, Kai Rothauge, Michael W Mahoney, and Joseph Gonzalez. 2018. On the computational inefficiency of large batch sizes for stochastic gradient descent. *arXiv preprint arXiv:1811.12941* (2018).

[22] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. 2006. A kernel method for the two-sample-problem. *Advances in neural information processing systems* 19 (2006), n.p.

[23] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017), n.p.

[24] Johann Hauswald, Yiping Kang, Michael A Laurenzano, Quan Chen, Cheng Li, Trevor Mudge, Ronald G Dreslinski, Jason Mars, and Lingjia Tang. 2015. Djinn and tonic: Dnn as a service and its implications for future warehouse scale computers. *ACM SIGARCH Computer Architecture News* 43, 3S (2015), 27–40.

[25] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[26] Yaochen Hu, Amit Levi, Ishaan Kumar, Yingxue Zhang, and Mark Coates. 2021. On Batch-size Selection for Stochastic Training for Graph Neural Networks. *openreview.net* (2021), n.p.

[27] Zhihao Jia, Sina Lin, Mingyu Gao, Matei Zaharia, and Alex Aiken. 2020. Improving the accuracy, scalability, and performance of graph neural networks with roc. *Proceedings of Machine Learning and Systems* 2 (2020), 187–198.

[28] Tim Kaler, Nickolas Stathas, Anne Ouyang, Alexandros-Stavros Iliopoulos, Tao Schardl, Charles E Leiserson, and Jie Chen. 2022. Accelerating training and inference of graph neural networks with fast sampling and pipelining. *Proceedings of Machine Learning and Systems* 4 (2022), 172–189.

[29] Leonid V Kantorovich. 1960. Mathematical methods of organizing and planning production. *Management science* 6, 4 (1960), 366–422.

[30] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836* (2016).

[31] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* n.v., n.p. (2016), n.p.

[32] Vladimir Koltchinskii. 2001. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory* 47, 5 (2001), 1902–1914.

[33] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

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

[35] Yuanzhi Li and Yingyu Liang. 2018. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in neural information processing systems* 31 (2018).

[36] Renjie Liao, Raquel Urtasun, and Richard Zemel. 2020. A pac-bayesian approach to generalization bounds for graph neural networks. *arXiv preprint arXiv:2012.07690* n.v., n.p. (2020), n.p.

[37] Yucong Lin, Silu Li, Jiaxing Xu, Jiawei Xu, Dong Huang, Wendi Zheng, Yuan Cao, and Junwei Lu. 2023. Graph over-parameterization: Why the graph helps the training of deep graph convolutional network. *Neurocomputing* 534 (2023), 77–85.

[38] Zhiqi Lin, Cheng Li, Youshan Miao, Yunxin Liu, and Yinlong Xu. 2020. Pagraph: Scaling gnn training on large graphs via computation-aware caching. In *Proceedings of the 11th ACM Symposium on Cloud Computing*. n.p., n.p., 401–415.

[39] Chaoyue Liu, Dmitriy Drusvyatskiy, Misha Belkin, Damek Davis, and Yian Ma. 2024. Aiming towards the minimizers: fast convergence of SGD for overparametrized problems. *Advances in neural information processing systems* 36 (2024).

[40] Tianfeng Liu, Yangrui Chen, Dan Li, Chuan Wu, Yibo Zhu, Jun He, Yanghua Peng, Hongzheng Chen, Hongzhi Chen, and Chuanxiong Guo. 2023. {BGL}:{GPU-Efficient}{GNN} training by optimizing graph data {I/O} and preprocessing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. n.p., n.p., 103–118.

[41] Shaogao Lv. 2021. Generalization bounds for graph convolutional neural networks via rademacher complexity. *arXiv preprint arXiv:2102.10234* n.v., n.p. (2021), n.p.

[42] Jiaqi Ma, Junwei Deng, and Qiaozhu Mei. 2021. Subgroup generalization and fairness of graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 1048–1061.

[43] David McAllester. 2003. Simplified PAC-Bayesian margin bounds. In *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*. Springer, n.p., 203–215.

[44] Vasimuddin Md, Sanchit Misra, Guixiang Ma, Ramanarayan Mohanty, Evangelos Georganas, Alexander Heinecke, Dhiraj Kalamkar, Nesreen K Ahmed, and Sasikanth Avancha. 2021. Distgnn: Scalable distributed training for large-scale graph neural networks. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. n.p., n.p., 1–14.

[45] Seung Won Min, Kun Wu, Mert Hidayetoglu, Jinjun Xiong, Xiang Song, and Wen-mei Hwu. 2022. Graph neural network training and data tiering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. n.p., n.p., 3555–3565.

[46] Hesham Mostafa. 2022. Sequential Aggregation and Rematerialization: Distributed Full-batch Training of Graph Neural Networks on Large Graphs. *MLSys* (2022).

[47] Seyed Morteza Nabavinejad, Sherief Reda, and Masoumeh Ebrahimi. 2021. BatchSizer: Power-performance trade-off for DNN inference. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. 819–824.

[48] Kenta Oono and Taiji Suzuki. 2020. Optimization and generalization analysis of transduction through gradient boosting and application to multi-scale graph neural networks. *Advances in Neural Information Processing Systems* 33 (2020), 18917–18930.

[49] Guillaume Papa, Pascal Bianchi, and Stéphan Clémençon. 2015. Adaptive sampling for incremental optimization using stochastic gradient descent. In *International Conference on Algorithmic Learning Theory*. Springer, 317–331.

[50] Jingshu Peng, Zhao Chen, Yingxia Shao, Yanyan Shen, Lei Chen, and Jiannong Cao. 2022. Sancus: sta le n ess-aware c omm u nication-avoiding full-graph decentralized training in large-scale graph neural networks. *Proceedings of the VLDB Endowment* 15, 9 (2022), 1937–1950.

[51] Boris Teodorovich Polyak. 1963. Gradient methods for minimizing functionals. *Zhurnal vychislitel'noi matematiki i matematicheskoi fiziki* 3, 4 (1963), 643–653.

[52] Franco Scarselli, Ah Chung Tsoi, and Markus Hagenbuchner. 2018. The vapnik–chervonenkis dimension of graph and recursive neural networks. *Neural Networks* 108 (2018), 248–259.

[53] SL Smith. 2017. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489* (2017).

[54] Asadi Soodabeh and Manfred Vogel. 2020. A learning rate method for full-batch gradient descent. *Papers on Technical Science* 13 (2020), 174–177.

[55] Jie Sun, Li Su, Zuocheng Shi, Wenting Shen, Zeke Wang, Lei Wang, Jie Zhang, Yong Li, Wenyuan Yu, Jingren Zhou, et al. 2023. Legion: Automatically Pushing the Envelope of {Multi-GPU} System for {Billion-Scale}{GNN} Training. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. n.p., n.p., 165–179.

[56] Huayi Tang and Yong Liu. 2023. Towards understanding generalization of graph neural networks. In *International Conference on Machine Learning*. PMLR, n.p., 33674–33719.

[57] John Thorpe, Yifan Qiao, Jonathan Eyolfson, Shen Teng, Guanzhou Hu, Zhihao Jia, Jinliang Wei, Keval Vora, Ravi Netravali, Miryung Kim, et al. 2021. Dorylus: Affordable, scalable, and accurate {GNN} training with distributed {CPU} servers and serverless threads. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*. 495–514.

[58] Vladimir N Vapnik and A Ya Chervonenkis. 2015. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity: festschrift for alexey chervonenkis*. Springer, n.p., 11–30.

[59] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* n.v., n.p. (2017), n.p.

[60] Saurabh Verma and Zhi-Li Zhang. 2019. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. n.p., n.p., 1539–1548.

[61] Borui Wan, Juntao Zhao, and Chuan Wu. 2023. Adaptive message quantization and parallelization for distributed full-graph gnn training. *Proceedings of Machine Learning and Systems* 5 (2023), n.p.

[62] Cheng Wan, Youjie Li, Ang Li, Nam Sung Kim, and Yingyan Lin. 2022. Bns-gcn: Efficient full-graph training of graph convolutional networks with partition-parallelism and random boundary node sampling. *Proceedings of Machine Learning and Systems* 4 (2022), 673–693.

[63] Cheng Wan, Youjie Li, Cameron R Wolfe, Anastasios Kyrillidis, Nam Sung Kim, and Yingyan Lin. 2022. PipeGCN: Efficient full-graph training of graph convolutional networks with pipelined feature communication. *arXiv preprint arXiv:2203.10428* n.v., n.p. (2022), n.p.

[64] Xinchen Wan, Kaiqiang Xu, Xudong Liao, Yilun Jin, Kai Chen, and Xin Jin. 2023. Scalable and efficient full-graph gnn training for large graphs. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–23.

[65] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* n.v., n.p. (2018), n.p.

[66] Keyulu Xu, Mozhi Zhang, Stefanie Jegelka, and Kenji Kawaguchi. 2021. Optimization of graph neural networks: Implicit acceleration by skip connections and more depth. In *International Conference on Machine Learning*. PMLR, n.p., 11592–11602.

[67] Naganand Yadati. 2022. A convex formulation for graph convolutional training: Two layer case. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, n.p., 1281–1286.

[68] Chenxiao Yang, Qitian Wu, David Wipf, Ruoyu Sun, and Junchi Yan. 2023. How Graph Neural Networks Learn: Lessons from Training Dynamics in Function Space. *arXiv preprint arXiv:2310.05105* n.v., n.p. (2023), n.p.

[69] Yang You, Aydın Buluç, and James Demmel. 2017. Scaling deep learning on GPU and knights landing clusters. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. n.p., n.p., 1–12.

[70] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962* (2019).

[71] Hao Yuan, Yajiong Liu, Yanfeng Zhang, Xin Ai, Qiange Wang, Chaoyi Chen, Yu Gu, and Ge Yu. 2023. Comprehensive Evaluation of GNN Training Systems: A Data Management Perspective. *arXiv preprint arXiv:2311.13279* (2023).

[72] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. 2017. Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811* n.v., n.p. (2017), n.p.

[73] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931* (2019), n.p.

[74] Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. 2019. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems* 32 (2019).

[75] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems* 31 (2018), n.p.

[76] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. 2019. Graph convolutional networks: a comprehensive review. *Computational Social Networks* 6, 1 (2019), 1–23.

[77] Xin Zhang, Yanyan Shen, Yingxia Shao, and Lei Chen. 2023. DUCATI: A dual-cache training system for graph neural networks on giant graphs with the GPU. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–24.

[78] Guoyi Zhao, Tian Zhou, and Lixin Gao. 2021. CM-GCN: A distributed framework for graph convolutional networks using cohesive mini-batches. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, n.p., 153–163.

[79] Da Zheng, Chao Ma, Minjie Wang, Jinjing Zhou, Qidong Su, Xiang Song, Quan Gan, Zheng Zhang, and George Karypis. 2020. DistDGL: Distributed graph neural network training for billion-scale graphs. In *2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*. IEEE, n.p., 36–44.

[80] Da Zheng, Xiang Song, Chengru Yang, Dominique LaSalle, and George Karypis. 2022. Distributed hybrid cpu and gpu training for graph neural networks on billion-scale heterogeneous graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. n.p., n.p., 4582–4591.

[81] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. 2019. Aligraph: A comprehensive graph neural network platform. *arXiv preprint arXiv:1902.08730* n.v., n.p. (2019), n.p.

[82] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. 2020. Gradient descent optimizes over-parameterized deep ReLU networks. *Machine learning* 109 (2020), 467–492.

[83] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. 2019. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems* 32 (2019), n.p.

[84] Difan Zou, Philip M Long, and Quanquan Gu. 2020. On the global convergence of training deep linear resnets. *arXiv preprint arXiv:2003.01094* (2020).

A NOTATIONS

n	Number of nodes of the entire graph
C	Training set
$\mathcal{Z}$	Testing set
n_C / $n_\mathcal{Z}$	Number of nodes in the training set C / the testing set $\mathcal{Z}$
$n_{\min}$	The minimal value between n_C and $n_\mathcal{Z}$
$\mathbf{X}/\mathbf{x}_i$	Node feature matrix / i-th row of node feature matrix
y_i	Ground truth label of node i
$\mathbf{y}_i$ / $\hat{\mathbf{y}}_i$	Ground truth label in one-hot form / estimated outcomes of node i
r	feature size
α	Training method for GNNs
F / M	Full-graph / Mini-batch training method
b	Batch size
β	Fan-out size
$\mathbf{A}_C^\alpha$	Adjacency matrix in each training iteration under method α
s^α	Number of nodes per iteration with $s^F = n_C$ and $b = b$
$\mathbf{D}_C^{\text{in},\alpha}$ / $\mathbf{D}_C^{\text{out},\alpha}$	Diagonal in- / out-degree matrices in each training iteration under method α
$\tilde{\mathbf{A}}_C^\alpha$ / $\tilde{\mathbf{a}}_{C,i}^\alpha$	Normalized adjacency matrix / i-th row of normalized adjacency matrix in a training iteration under method α
$\tilde{\mathbf{A}}_\mathcal{Z}$ / $\tilde{\mathbf{a}}_{\mathcal{Z},i}$	Normalized adjacency matrix / i-th row of Normalized adjacency matrix in testing set
$\mathbf{W}^\alpha$ / $\mathbf{w}_i^\alpha$	Learnable parameters / i-th row of parameters of the GNN under the method α
$\mathbf{W}^{\alpha*}$ / $\mathbf{w}_i^{\alpha*}$	Ground truth / i-th row of ground truth of learnable parameters $\mathbf{W}^\alpha$ under the method α
h	Hidden size
K	Number of label categories
$\sigma(\cdot)$	ReLU activation function
$L_C^\alpha(\cdot)$ / $\hat{L}_C^\alpha(\cdot)$	Expected / empirical training risk
$L_\mathcal{Z}^\alpha(\cdot)$ / $\hat{L}_\mathcal{Z}^\alpha(\cdot)$	Expected / empirical testing risk
η	Learning rate
$\hat{\sigma}(\cdot)$	Dual activation function
$\mathcal{P}/\mathcal{Q}$	Prior / Posterior distribution of model parameters
$U(\cdot)$	Expected loss discrepancy between training set C and testing set $\mathcal{Z}$
$\delta(\cdot)$	Distance function
ρ	Probability
θ	Covariance

Table 2: Notations

B PROOF FROM SECTION 4

B.1 Convergence

B.1.1 Expressions for loss and gradients. **Definition 1.** (Dual activation [16]) *The dual activation of σ is the function $\hat{\sigma}:[-1,1]\to\mathbb{R}$ defined as*

$$\hat{\sigma}(\theta)=\mathbb{E}\left[\sigma(X)\,\sigma(Y)\right], \tag{7}$$

where X and Y are jointly Gaussian random variables with mean zero, variance one, and covariance θ.

Daniely et al. [16] demonstrated that dual activations hold continuity over the interval $[-1,1]$ and convexity within the range $[0,1]$.

We first begin by writing an equivalent expression of $L_C^\alpha(\mathbf{w}_j^\alpha)$ with $j\in\{1,\dots,h\}$ as:

$$L_C^\alpha(\mathbf{w}_j^\alpha)=\frac{1}{2s^\alpha}\left(\mathbb{E}[\sum_{i=1}^{s^\alpha}\sigma(\tilde{\mathbf{a}}_{C,i}^\alpha\mathbf{X}\mathbf{w}_j^{\alpha\top})^2]+\mathbb{E}[\sum_{i=1}^{s^\alpha}\sigma(\tilde{\mathbf{a}}_{C,i}^\alpha\mathbf{X}\mathbf{w}_j^{\alpha*\top})^2]-2\mathbb{E}[\sum_{i,j=1}^{s^\alpha}\sigma(\tilde{\mathbf{a}}_{C,i}^\alpha\mathbf{X}\mathbf{w}_j^{\alpha\top})\sigma(\tilde{\mathbf{a}}_{C,i}^\alpha\mathbf{X}\mathbf{w}_j^{\alpha*\top})]\right) \tag{8}$$

We next compute expressions for each of the three terms above.

$$
\begin{aligned}
&\frac{1}{s^\alpha}\mathbb{E}[\sum_{i=1}^{s^\alpha}\sigma(\tilde{\mathbf{a}}^\alpha_{C,i}\mathbf{X}\mathbf{w}^{\alpha\top}_j)^2] \\
=&\frac{1}{s^\alpha}\mathbb{E}[\sum_{i,k=1}^{s^\alpha}\delta_{ij}\sigma(\tilde{\mathbf{a}}^\alpha_{C,i}\mathbf{X}\mathbf{w}^{\alpha\top}_j)\sigma(\tilde{\mathbf{a}}^\alpha_{C,k}\mathbf{X}\mathbf{w}^{\alpha\top}_j))] \\
=&\frac{1}{s^\alpha}\mathbb{E}[\|\mathbf{w}^\alpha_j\|^2\sum_{i,k=1}^{s^\alpha}\delta_{ij}\sqrt{(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_i(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_k}\sigma(\frac{\tilde{\mathbf{a}}^\alpha_{C,i}\mathbf{X}\mathbf{w}^{\alpha\top}_j}{\sqrt{(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_i}\|\mathbf{w}^\alpha_j\|})\sigma(\frac{\tilde{\mathbf{a}}^\alpha_{C,k}\mathbf{X}\mathbf{w}^{\alpha\top}_j}{\sqrt{(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_k}\|\mathbf{w}^\alpha_j\|})] \\
=&\frac{1}{s^\alpha}\|\mathbf{w}^\alpha_j\|^2\sum_{i,k=1}^{s^\alpha}\delta_{ik}\hat{\sigma}(\frac{\varrho_{i,k}}{\sqrt{\vartheta^\alpha_{i,k}}})\sqrt{\vartheta^\alpha_{i,k}} \qquad \text{(Definition B.1.1 and } \vartheta^\alpha_{i,j}=(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_i(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_j) \\
=&\|\mathbf{w}^\alpha_j\|^2\Gamma^\alpha,
\end{aligned}
\tag{9}
$$

where $\delta_{ij}=1$ if $i=j$ and $\delta_{ij}=0$ if $i\neq j$, $\varrho_{i,j}$ denotes the amount of common messages between node i and node j at a given training iteration, and we have

$$
\Gamma^\alpha = \frac{1}{s^\alpha}\sum_{i,j=1}^{s^\alpha}\delta_{ij}\hat{\sigma}(\frac{\varrho_{i,j}}{\sqrt{\vartheta^\alpha_{i,j}}})\sqrt{\vartheta^\alpha_{i,j}} \tag{10a}
$$

$$
\vartheta^\alpha_{i,j}=(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_i(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_j \tag{10b}
$$

Similarly, we get the second term as:

$$
\frac{1}{s^\alpha}\mathbb{E}[\sum_{i=1}^{s^\alpha}\sigma(\tilde{\mathbf{a}}^\alpha_{C,i}\mathbf{X}\mathbf{w}^{\alpha*\top}_j)^2]=\|\mathbf{w}^{\alpha*}_j\|^2\Gamma^\alpha \tag{11}
$$

Finally we simplify the last term as:

$$
\begin{aligned}
&\frac{1}{s^\alpha}\mathbb{E}[\sum_{i,k=1}^{s^\alpha}\delta_{ik}\sigma(\tilde{\mathbf{a}}^\alpha_{C,i}\mathbf{X}\mathbf{w}^{\alpha\top}_j)\sigma(\tilde{\mathbf{a}}^\alpha_{C,k}\mathbf{X}\mathbf{w}^{\alpha*\top}_j)] \\
=&\frac{1}{s^\alpha}\mathbb{E}[\|\mathbf{w}^\alpha_j\|\|\mathbf{w}^{\alpha*}_j\|\sum_{i,k=1}^{s^\alpha}\delta_{ik}\sqrt{(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_i(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_k}\sigma(\frac{\tilde{\mathbf{a}}^\alpha_{C,i}\mathbf{X}\mathbf{w}^{\alpha\top}_j}{\sqrt{(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_i}\|\mathbf{w}^\alpha_j\|})\sigma(\frac{\tilde{\mathbf{a}}^\alpha_{C,k}\mathbf{X}\mathbf{w}^{\alpha*\top}_j}{\sqrt{(\tilde{\mathbf{A}}^\alpha_C\mathbb{1})_k}\|\mathbf{w}^{\alpha*}_j\|})] \\
=&\frac{1}{s^\alpha}\|\mathbf{w}^\alpha_j\|\|\mathbf{w}^{\alpha*}_j\|\sum_{i,k=1}^{s^\alpha}\delta_{ik}\hat{\sigma}(\frac{\varrho_{i,k}}{\sqrt{\vartheta^\alpha_{i,k}}}\frac{\mathbf{w}^{\alpha\top}_j\mathbf{w}^{\alpha*}_j}{\|\mathbf{w}^\alpha_j\|\|\mathbf{w}^{\alpha*}_j\|})\sqrt{\vartheta^\alpha_{i,k}}
\end{aligned}
\tag{12}
$$

Therefore, we have the expression of $L^\alpha_C(\mathbf{w}^\alpha_j)$ as:

$$
L^\alpha_C(\mathbf{w}^\alpha_j)=\frac{1}{2}(\|\mathbf{w}^\alpha_j\|^2\Gamma^\alpha+\|\mathbf{w}^{\alpha*}_j\|^2\Gamma^\alpha-\frac{2}{s^\alpha}\|\mathbf{w}^\alpha_j\|\|\mathbf{w}^{\alpha*}_j\|\sum_{i,k=1}^{s^\alpha}\delta_{ik}\hat{\sigma}(\frac{\varrho_{i,k}}{\sqrt{\vartheta^\alpha_{i,k}}}\frac{\mathbf{w}^{\alpha\top}_j\mathbf{w}^{\alpha*}_j}{\|\mathbf{w}^\alpha_j\|\|\mathbf{w}^{\alpha*}_j\|})\sqrt{\vartheta^\alpha_{i,k}}) \tag{13}
$$

It is easy to see that if $\mathbf{w}^\alpha_{0,j}$ is the initial value of $\mathbf{w}^\alpha_j$ with $j\in\{1,\dots,h\}$ then each subsequent iteration will be a linear combination of $\mathbf{w}^\alpha_{0,j}$ and $\mathbf{w}^{\alpha*}_j$. Hence we can assume that $\mathbf{w}^\alpha_j=\phi^\alpha\mathbf{w}^{\alpha*}_j+\psi^\alpha\mathbf{w}^{\alpha\perp}_j$, where $\mathbf{w}^\perp$ is a fixed unit vector (depending on the initialization) orthogonal to $\mathbf{w}^{\alpha*}_j$. Then rewriting the loss in terms of ϕ^α, ψ^α and recalling that $\|\mathbf{w}^{\alpha*}_j\|=1$ we get the simplified expression of $L^\alpha_C(\mathbf{w}^\alpha_j)$ for all $\alpha\in\{F,M\}$:

$$
L^\alpha_C(\phi^\alpha,\psi^\alpha)=\frac{1}{2}(\phi^{\alpha 2}+\psi^{\alpha 2}+1)\Gamma^\alpha-\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}\Upsilon^\alpha, \tag{14a}
$$

$$
\Upsilon^\alpha=\frac{1}{s^\alpha}\sum_{i,j=1}^{s^\alpha}\delta_{ij}\hat{\sigma}(\frac{\phi^\alpha}{\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}}\frac{\varrho_{i,j}}{\sqrt{\vartheta^\alpha_{i,j}}})\sqrt{\vartheta^\alpha_{i,j}}. \tag{14b}
$$

Then we compute the gradient of the objective w.r.t. $\mathbf{w}$ or equivalently w.r.t. ϕ, ψ.

$$\begin{aligned}\frac{\partial L^{\alpha}(\phi^{\alpha},\psi^{\alpha})}{\partial \phi^{\alpha}} &= \phi^{\alpha}\Gamma^{\alpha} - \frac{\phi^{\alpha}\Upsilon^{\alpha}}{\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}} + \frac{1}{s^{\alpha 2}}\Big(\frac{\psi^{\alpha 2}}{\phi^{\alpha 2}+\psi^{\alpha 2}}\sum_{i,j=1}^{s^{\alpha}}\delta_{ij}\varrho_{i,j}\hat{\sigma}'\big(\frac{\phi^{\alpha}}{\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}}\frac{\varrho_{i,j}}{\sqrt{\vartheta^{\alpha}_{i,j}}}\big)\Big)\\ &= \phi^{\alpha}\Gamma^{\alpha} - \frac{\phi^{\alpha}\Upsilon^{\alpha}}{\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}} + \frac{1}{s^{\alpha 2}}\Big(\frac{\psi^{\alpha 2}}{\phi^{\alpha 2}+\psi^{\alpha 2}}\sum_{i,j=1}^{s^{\alpha}}\delta_{ij}\varrho_{i,j}\hat{\sigma}_{step}\big(\frac{\phi^{\alpha}}{\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}}\frac{\varrho_{i,j}}{\sqrt{\vartheta^{\alpha}_{i,j}}}\big)\Big),\\ &= \phi^{\alpha}\Gamma^{\alpha} - \frac{\phi^{\alpha}\Upsilon^{\alpha}}{\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}} + \frac{\psi^{\alpha 2}\Xi^{\alpha}}{\phi^{\alpha 2}+\psi^{\alpha 2}}\end{aligned} \tag{15}$$

where in the second equality we use $\hat{\sigma}' = \hat{\sigma'}$ and $\hat{\sigma'} = \sqrt{2}\mathbb{1}(x \geq 0) = \sigma_{step}(x)$, and σ_{step} is the step function. We have

$$\Xi^{\alpha} = \frac{1}{s^{\alpha}}\sum_{i,j=1}^{s^{\alpha}}\delta_{ij}\varrho_{i,j}\hat{\sigma}_{step}\big(\frac{\phi^{\alpha}}{\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}}\frac{\varrho_{i,j}}{\sqrt{\vartheta^{\alpha}_{i,j}}}\big). \tag{16}$$

Similarly, we get that

$$\frac{\partial L^{\alpha}(\phi^{\alpha},\psi^{\alpha})}{\partial \psi^{\alpha}} = \psi^{\alpha}\Gamma^{\alpha} - \frac{\psi^{\alpha}\Upsilon^{\alpha}}{\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}} + \frac{\phi^{\alpha}\psi^{\alpha}\Xi^{\alpha}}{\phi^{\alpha 2}+\psi^{\alpha 2}}. \tag{17}$$

B.1.2 Lemmas. **Lemma 1.** *If the in-degree of each node is $O(\beta)$ and the out-degree of each node is $O(s^{\alpha})$, then we have that $\frac{1}{\pi} \leq \frac{1}{\pi s^{\alpha}}\|\mathbf{A}^{\alpha}_{C,t}\mathbb{1}\|_1 \leq \Gamma^{\alpha}_t \leq 1$, and $|\Upsilon^{\alpha}_t| \leq \Gamma^{\alpha}_t$, where $\|\mathbf{A}^{M}_{C,t}\mathbb{1}\|_1 = \Theta(\beta^{1/2})\Omega(b^{1/2})$ and $\|\mathbf{A}^{F}_{C,t}\mathbb{1}\|_1 = \Theta(n^{1/2})\Omega(n_C^{1/2})$.*

Proof. We first recall that $\hat{\sigma}(x) \geq \frac{1}{\pi}$ whenever $x \geq 0$ [16]. Hence, the bound on Γ^{α}_t follows as:

$$\begin{aligned}\Gamma^{\alpha}_t =& \frac{1}{s^{\alpha}}\sum_{i,j=1}^{s^{\alpha}}\delta_{ij}\hat{\sigma}\big(\frac{\varrho_{i,j}}{\sqrt{\vartheta^{\alpha}_{i,j}}}\big)\sqrt{\vartheta^{\alpha}_{i,j}}\\ \geq& \frac{1}{\pi s^{\alpha}}\sum_{i,j=1}^{s^{\alpha}}\delta_{ij}\sqrt{\vartheta^{\alpha}_{i,j}} = \frac{1}{\pi s^{\alpha}}\sum_{i,j=1}^{s^{\alpha}}\delta_{ij}\sqrt{(\tilde{\mathbf{A}}^{\alpha}_{C}\mathbb{1})_i(\tilde{\mathbf{A}}^{\alpha}_{C}\mathbb{1})_j}\\ =& \frac{1}{\pi s^{\alpha}}\sum_{i=1}^{s^{\alpha}}(\tilde{\mathbf{A}}^{\alpha}_{C}\mathbb{1})_i,\\ =& \frac{1}{\pi s^{\alpha}}\|\mathbf{A}^{\alpha}_{C,t}\mathbb{1}\|_1.\end{aligned} \tag{18}$$

To bound Υ^{F}_t, we notice that $|\hat{\sigma}(x)| \leq \hat{\sigma}(|x|)$, and $\hat{\sigma}(\cdot)$ is a non-decreasing function in $[0,1]$. Hence, we get $|\Upsilon^{F}_t| \leq \Gamma^{F}_t$.

we have the normalized adjacency matrix of a graph with s nodes as:

$$\tilde{\mathbf{A}} = \begin{bmatrix}\frac{1}{\sqrt{d^{\text{in}}_1}} & & \\ & \ddots & \\ & & \frac{1}{\sqrt{d^{\text{in}}_b}}\end{bmatrix}\begin{bmatrix}a_{11} & \cdots & a_{1n}\\ \vdots & \ddots & \vdots\\ a_{b1} & \cdots & a_{bn}\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{d^{\text{out}}_1}} & & \\ & \ddots & \\ & & \frac{1}{\sqrt{d^{\text{out}}_n}}\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{d^{\text{in}}_1}\sqrt{d^{\text{out}}_1}}a_{11} & \cdots & \frac{1}{\sqrt{d^{\text{in}}_1}\sqrt{d^{\text{out}}_n}}a_{1n}\\ \vdots & \ddots & \vdots\\ \frac{1}{\sqrt{d^{\text{in}}_1}\sqrt{d^{\text{out}}_b}}a_{s1} & \cdots & \frac{1}{\sqrt{d^{\text{in}}_b}\sqrt{d^{\text{out}}_n}}a_{nn}\end{bmatrix}, \tag{19}$$

where $a_{ij} \in \{0,1\}$ represents if node i connects with node j (1) or not (0), and $d^{\text{in}}_i = O(\beta)$ represents the in-degree of node i and $d^{\text{out}}_i = O(b)$ represents the out-degree of node i. Then we have $\tilde{\mathbf{A}}_{ij} = \Omega(\frac{1}{\beta^{1/2}b^{1/2}})$. Adding at most β terms, we can have $(\tilde{\mathbf{A}}^{M}_{C}\mathbb{1})_i = \Theta(\beta^{1/2})\Omega(b^{-1/2})$. Similarly, $(\tilde{\mathbf{A}}^{F}_{C}\mathbb{1})_i = \Theta(n^{1/2})\Omega(n_C^{-1/2})$.

For $\|\mathbf{A}^{M}_{C,t}\mathbb{1}\|_1$, we have $\tilde{\mathbf{A}}^{M}_{C,t}\mathbb{1} \in \mathbb{R}^{b\times 1}$, thereby $\|\mathbf{A}^{M}_{C,t}\mathbb{1}\|_1 = \Theta(\beta^{1/2})\Omega(b^{-1/2})b = \Theta(\beta^{1/2})\Omega(b^{1/2})$. Similarly, $\|\mathbf{A}^{F}_{C,t}\mathbb{1}\|_1 = \Theta(n^{1/2})\Omega(n_C^{1/2})$.

Lemma 2. *If the in-degree of each node is $O(\beta)$ in mini-batch training or $O(n)$ in full-graph training, and the out-degree of each node is $O(s^{\alpha})$, then we have that $|\Xi^{\alpha}_t| = o(\Gamma^{\alpha}_t)$, and, when $\phi^{\alpha}_t \geq -\frac{1}{100}$ and $\sqrt{\phi^{\alpha 2}_t + \psi^{\alpha 2}_t} \geq 1 - o(1)$, then $\Xi^{\alpha}_t \geq \frac{\Gamma^{\alpha}_t}{2s^{\alpha}}$.*

Proof. We analyze the upper bound of Ξ^{α}_t first. Each term in summation is non-zero only when $\varrho_{i,j} \neq 0$. Hence, there are at most s^{α} non-zero terms in the summation, and Ξ^{α}_t is upper bounded by $\frac{1}{s^{\alpha}}\|\mathbf{A}^{\alpha}_{C,t}\mathbb{1}\|_1$ since $|\hat{\sigma}(x)| \leq 1$. Hence, the upper bound on Ξ^{α}_t follows $\Xi^{\alpha}_t = o(\Gamma^{\alpha}_t)$.

We recall that $\hat{\sigma}_{step}(x) \geq \frac{1}{2}$ whenever $|x| \leq \frac{1}{50}$ [16], which is ensured by $\phi_t \geq -\frac{1}{100}$ and $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \geq 1 - o(1)$. Hence, in this case, each term in the summation in the expression of Ξ_t^α will be non-negative. Hence in this case $\Xi_t^\alpha \geq \frac{1}{s^\alpha}\frac{1}{2} \geq \frac{\Gamma_t^\alpha}{2s^\alpha}$ (since $\Gamma_t^\alpha \leq 1$).

Lemma 3. *If $\mathbf{w}_{0,j}^\alpha \sim N(0, \kappa^2 \mathbf{I})$ and the learning rate $\eta_t \leq 0.25$, then with probability at least $1 - e^{-O(r)}$, it holds that for all $t > 0$, $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \leq 2\kappa\sqrt{r} + 4\pi + 1$, and $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} > 0$ for all $t \geq 1$.*

Proof. We have

$$\begin{aligned}
{\phi_{t+1}^\alpha}^2 + {\psi_{t+1}^\alpha}^2 =&(\phi_t^\alpha - \eta_t \frac{\partial L_{C,t}^\alpha(\phi_t^\alpha, \psi_t^\alpha)}{\partial \phi_t^\alpha})^2 + (\psi_t^\alpha - \eta_t \frac{\partial L_{C,t}^\alpha(\phi_t^\alpha, \psi_t^\alpha)}{\partial \psi_t^\alpha})^2 \\
=&(\phi_t^\alpha - \eta_t \phi_t^\alpha \Gamma_t^\alpha + \eta_t \frac{\phi_t^\alpha}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}} \Upsilon_t^\alpha + \eta_t \frac{\psi_t^{\alpha 2}}{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \Xi_t^\alpha)^2 \\
&+ (\psi_t^\alpha - \eta_t \psi_t^\alpha \Gamma_t^\alpha + \eta_t \frac{\psi_t^\alpha}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}} \Upsilon_t^\alpha + \eta_t \frac{\phi_t^\alpha \psi_t^\alpha}{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \Xi_t^\alpha)^2 \\
=&(\phi_t^{\alpha 2} + \psi_t^{\alpha 2}) + \eta_t^2 \Gamma_t^{\alpha 2}(\phi_t^{\alpha 2} + \psi_t^{\alpha 2}) + \eta_t^2 \Upsilon_t^{\alpha 2} + \eta_t^2 \frac{\psi_t^{\alpha 2}}{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \Xi_t^{\alpha 2} \\
&- 2\eta_t \Gamma_t^\alpha (\phi_t^{\alpha 2} + \psi_t^{\alpha 2}) + 2\eta_t \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \Upsilon_t^\alpha - 2\eta_t^2 \Gamma_t^\alpha \Upsilon_t^\alpha \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \\
\leq&(\phi_t^{\alpha 2} + \psi_t^{\alpha 2}) + \eta_t^2 (\phi_t^{\alpha 2} + \psi_t^{\alpha 2}) + \eta_t^2 + \eta_t^2 \frac{\psi_t^{\alpha 2}}{(4s^\alpha)^2 (\phi_t^{\alpha 2} + \psi_t^{\alpha 2})} \\
&- \frac{2}{\pi} \eta_t (\phi_t^{\alpha 2} + \psi_t^{\alpha 2}) + 2\eta_t \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} + 2\eta_t^2 \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}
\end{aligned} \tag{20}$$

Hence, if the learning rate $\eta_t \leq 0.25$, we have

$$
{\phi_{t+1}^\alpha}^2 + {\psi_{t+1}^\alpha}^2 \leq (\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}(1 - \frac{1}{16\pi}))^2 + \frac{1}{16}. \tag{21}
$$

Then, for all $t \geq 1$, $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \leq \sqrt{\phi_0^{\alpha 2} + \psi_0^{\alpha 2}} + 4\pi + 1$. Furthermore, we also have that if $\Upsilon_t^\alpha > 0$, then

$$\begin{aligned}
{\phi_{t+1}^\alpha}^2 + {\psi_{t+1}^\alpha}^2 \geq&(\phi_t^{\alpha 2} + \psi_t^{\alpha 2})(1 - \eta_t \Gamma_t^\alpha)^2 + 2\eta_t \Upsilon_t^\alpha \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}(1 - \eta_t \Gamma_t^\alpha) + \eta_t^2 \Upsilon_t^{\alpha 2} \\
>&\eta_t^2 \Upsilon_t^{\alpha 2} \geq 0
\end{aligned} \tag{22}$$

Finally, with probability at least $1 - e^{O(r)}$, we will have $\sqrt{\phi_0^{\alpha 2} + \psi_0^{\alpha 2}} = O(\kappa\sqrt{r})$.

We observe that the upper bound of the learning rate decreases as the batch size increases, which aligns with the convergence analysis in DNNs [1].

Lemma 4. *If the conditions in Lemma 1. and Lemma 3. hold, then for all $t \geq 1$ and any $v \in [0, 1]$ such that $(\phi^\alpha, \psi^\alpha) = (1 - v)(\phi_t^\alpha, \psi_t^\alpha) + v(\phi_{t+1}^\alpha, \psi_{t+1}^\alpha)$, we have that,*

$$
\lambda_{max}(\nabla^2 L_C^\alpha(\phi^\alpha, \psi^\alpha)) \leq 4(1 + \sqrt{2\kappa\sqrt{r} + 4\pi + 1} + o(\frac{1}{s^\alpha})),
$$

where λ_{max} is the maximum eigenvalue of the population Hessian denoted by $\nabla^2 L_C^\alpha(\phi^\alpha, \psi^\alpha)$.

Proof. From Lemma 3. , we immediately have $\sqrt{\phi^{\alpha 2} + \psi^{\alpha 2}} \leq 2\kappa\sqrt{r} + 4\pi + 1$.

Next, we analyze the upper bound of $\lambda_{max}(\nabla^2 L_C^\alpha(\phi^\alpha, \psi^\alpha))$. We have:

$$
\lambda_{max}(\nabla^2 L_C^\alpha(\phi^\alpha, \psi^\alpha)) \leq |\frac{\partial^2 L_C^\alpha(\phi^\alpha, \psi^\alpha)}{\partial \phi^{\alpha 2}}| + |\frac{\partial^2 L_C^\alpha(\phi^\alpha, \psi^\alpha)}{\partial \psi^{\alpha 2}}| + |\frac{\partial^2 L_C^\alpha(\phi^\alpha, \psi^\alpha)}{\partial \phi^\alpha \partial \psi^\alpha}| + |\frac{\partial^2 L_C^\alpha(\phi^\alpha, \psi^\alpha)}{\partial \psi^\alpha \partial \phi^\alpha}|. \tag{23}
$$

Taking the second derivatives, we get:

$$|\frac{\partial^2 L_C^\alpha(\phi^\alpha,\psi^\alpha)}{\partial\phi^{\alpha 2}}| = \Gamma^\alpha - \frac{\phi^\alpha}{\|\mathbf{w}_j^\alpha\|}\frac{\partial\Upsilon^\alpha}{\partial\phi^\alpha} - \frac{\psi^{\alpha 2}}{\|\mathbf{w}_j^\alpha\|^{3/2}}\Upsilon^\alpha - \frac{\psi^{\alpha 2}}{\|\mathbf{w}_j^\alpha\|^2}\frac{\partial\Xi^\alpha}{\partial\phi^\alpha} + \frac{2\phi^\alpha\psi^{\alpha 2}}{\|\mathbf{w}_j^\alpha\|^4}\Xi^\alpha \tag{24a}$$

$$|\frac{\partial^2 L_C^\alpha(\phi^\alpha,\psi^\alpha)}{\partial\psi^{\alpha 2}}| = \Gamma^\alpha - \frac{\psi^\alpha}{\|\mathbf{w}_j^\alpha\|}\frac{\partial\Upsilon^\alpha}{\partial\psi^\alpha} + \frac{\phi^\alpha\psi^\alpha}{\|\mathbf{w}_j^\alpha\|^{3/2}}\Upsilon^\alpha + \frac{\phi^\alpha\psi^\alpha}{\|\mathbf{w}_j^\alpha\|^2}\frac{\partial\Xi^\alpha}{\partial\phi^\alpha} + \frac{\phi^\alpha(\phi^{\alpha 2}-\psi^{\alpha 2})}{\|\mathbf{w}_j^\alpha\|^4}\Xi^\alpha \tag{24b}$$

$$|\frac{\partial^2 L_C^\alpha(\phi^\alpha,\psi^\alpha)}{\partial\phi^\alpha\partial\psi^\alpha}| = -\frac{\psi^\alpha}{\|\mathbf{w}_j^\alpha\|}\frac{\partial\Upsilon^\alpha}{\partial\phi^\alpha} + \frac{\phi^\alpha\psi^\alpha}{\|\mathbf{w}_j^\alpha\|^{3/2}}\Upsilon^\alpha + \frac{\phi^\alpha\psi^\alpha}{\|\mathbf{w}_j^\alpha\|^2}\frac{\partial\Xi^\alpha}{\partial\phi^\alpha} + \frac{\phi^\alpha\psi^\alpha}{\|\mathbf{w}_j^\alpha\|^2}\frac{\partial\Xi^\alpha}{\partial\phi^\alpha} + \frac{\psi^\alpha(\psi^{\alpha 2}-\phi^{\alpha 2})}{\|\mathbf{w}_j^\alpha\|^4}\Xi^\alpha \tag{24c}$$

$$|\frac{\partial^2 L_C^\alpha(\phi^\alpha,\psi^\alpha)}{\partial\psi^\alpha\partial\phi^\alpha}| = -\frac{\phi^\alpha}{\|\mathbf{w}_j^\alpha\|}\frac{\partial\Upsilon^\alpha}{\partial\psi^\alpha} + \frac{\phi^\alpha\psi^\alpha}{\|\mathbf{w}_j^\alpha\|^{3/2}}\Upsilon^\alpha + \frac{\phi^\alpha\psi^\alpha}{\|\mathbf{w}_j^\alpha\|^2}\frac{\partial\Xi^\alpha}{\partial\phi^\alpha} - 2\frac{\phi^\alpha\psi^{\alpha 2}}{\|\mathbf{w}_j^\alpha\|^4}\Xi^\alpha \tag{24d}$$

Next we have

$$|\frac{\partial\Upsilon^\alpha}{\partial\phi^\alpha}| = |\frac{1}{s^\alpha}\sum_{i,k=1}^{s^\alpha}\delta_{ik}\varrho_{i,k}\hat{\sigma}_{step}(\frac{\varrho_{i,k}}{\sqrt{\vartheta_{i,k}^\alpha}}\frac{\phi^\alpha}{\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}})\cdot\frac{\psi^{\alpha 2}}{\|\mathbf{w}_j^\alpha\|^{2/3}}| \le \frac{1}{s^\alpha}\sqrt{\|\mathbf{w}_j^\alpha\|}|\Xi^\alpha| = o(\frac{\Gamma^\alpha}{s^\alpha}) \tag{25a}$$

$$|\frac{\partial\Upsilon^\alpha}{\partial\psi^\alpha}| = |\frac{1}{s^\alpha}\sum_{i,k=1}^{s^\alpha}\delta_{ik}\varrho_{i,k}\hat{\sigma}_{step}(\frac{\varrho_{i,k}}{\sqrt{\vartheta_{i,k}^\alpha}}\frac{\phi^\alpha}{\sqrt{\phi^{\alpha 2}+\psi^{\alpha 2}}})\cdot\frac{\phi^\alpha\psi^\alpha}{\|\mathbf{w}_j^\alpha\|^{2/3}}| \le \frac{1}{s^\alpha}\sqrt{\|\mathbf{w}_j^\alpha\|}|\Xi^\alpha| = o(\frac{\Gamma^\alpha}{s^\alpha}) \tag{25b}$$

where we use $|\Xi_t^\alpha| = o(\Gamma_t^\alpha)$ in the Lemma 2.

To differentiate Ξ^α, we employ $\hat{\sigma}(\theta) = 1 - \frac{arccos(\theta)}{\pi}$ [16] and $arccos'(\theta) = -\frac{1}{\sqrt{1-\theta^2}}$ to get

$$|\frac{\partial\Xi^\alpha}{\partial\phi^\alpha}| =|\frac{1}{s^\alpha}\sum_{i,k=1}^{s^\alpha}\delta_{ik}\frac{\varrho_{i,k}^2}{\vartheta_{i,k}^\alpha}\frac{\|\mathbf{w}_j^\alpha\|}{\psi^\alpha}\frac{\psi^{\alpha 2}}{\|\mathbf{w}_j^\alpha\|^{2/3}}| \le \frac{1}{s^\alpha}\sqrt{\|\mathbf{w}_j^\alpha\|}|\frac{\varrho_{i,k}^2}{\vartheta_{i,k}^\alpha} = o(\frac{\Gamma^\alpha}{s^\alpha}) \tag{26a}$$

$$|\frac{\partial\Xi^\alpha}{\partial\psi^\alpha}| =|\frac{1}{s^\alpha}\sum_{i,k=1}^{s^\alpha}\delta_{ik}\frac{\varrho_{i,k}^2}{\vartheta_{i,k}^\alpha}\frac{\|\mathbf{w}_j^\alpha\|}{\psi^\alpha}\frac{\phi^\alpha\psi^\alpha}{\|\mathbf{w}_j^\alpha\|^{2/3}}| \le \frac{1}{s^\alpha}\sqrt{\|\mathbf{w}_j^\alpha\|}|\frac{\varrho_{i,k}^2}{\vartheta_{i,k}^\alpha} = o(\frac{\Gamma^\alpha}{s^\alpha}) \tag{26b}$$

Using $\Gamma^\alpha \le 1$, we get:

$$|\frac{\partial^2 L_C^\alpha(\phi^\alpha,\psi^\alpha)}{\partial\phi^{\alpha 2}}| \le \Gamma^\alpha + \sqrt{\|\mathbf{w}_j^\alpha\|}|\Gamma^\alpha + o(\frac{\Gamma^\alpha}{s^\alpha}) \le \Gamma^\alpha(1+\sqrt{2\kappa\sqrt{r}+4\pi+1}+o(\frac{1}{s^\alpha})) \le 1+\sqrt{2\kappa\sqrt{r}+4\pi+1}+o(\frac{1}{s^\alpha}) \tag{27a}$$

$$|\frac{\partial^2 L_C^\alpha(\phi^\alpha,\psi^\alpha)}{\partial\psi^{\alpha 2}}| \le \Gamma^\alpha + \sqrt{\|\mathbf{w}_j^\alpha\|}|\Gamma^\alpha + o(\frac{\Gamma^\alpha}{s^\alpha}) \le \Gamma^\alpha(1+\sqrt{2\kappa\sqrt{r}+4\pi+1}+o(\frac{1}{s^\alpha})) \le 1+\sqrt{2\kappa\sqrt{r}+4\pi+1}+o(\frac{1}{s^\alpha}) \tag{27b}$$

$$|\frac{\partial^2 L_C^\alpha(\phi^\alpha,\psi^\alpha)}{\partial\phi^\alpha\partial\psi^\alpha}| \le \sqrt{\|\mathbf{w}_j^\alpha\|}|\Gamma^\alpha + o(\frac{\Gamma^\alpha}{s^\alpha}) \le \Gamma^\alpha(\sqrt{2\kappa\sqrt{r}+4\pi+1}+o(\frac{1}{s^\alpha})) \le \sqrt{2\kappa\sqrt{r}+4\pi+1}+o(\frac{1}{s^\alpha}) \tag{27c}$$

$$|\frac{\partial^2 L_C^\alpha(\phi^\alpha,\psi^\alpha)}{\partial\psi^\alpha\partial\phi^\alpha}| \le \sqrt{\|\mathbf{w}_j^\alpha\|}|\Gamma^\alpha + o(\frac{\Gamma^\alpha}{s^\alpha}) \le \Gamma^\alpha(\sqrt{2\kappa\sqrt{r}+4\pi+1}+o(\frac{1}{s^\alpha})) \le \sqrt{2\kappa\sqrt{r}+4\pi+1}+o(\frac{1}{s^\alpha}) \tag{27d}$$

Lemma 5. *If* $\mathbf{w}_{0,j}^\alpha \sim N(0,\kappa^2\mathbf{I})$ *and the learning rate* $\eta_t \le 0.25$*, then with at least* $1-1/h^2$*, it holds that for all* $t \ge v_1\log(\kappa\log h)$, $\sqrt{\phi_t^{\alpha 2}+\psi_t^{\alpha 2}} \ge 1-o(1)$*, where* $v_1 > 0$ *is an absolute constant.*

Proof. Due to random initialization, with probability at least $1-\frac{1}{h^2}$, we have that $\|\mathbf{w}_{0,j}^\alpha\| = o(\kappa\sqrt{r})$ and $\phi_0^\alpha \ge -v'\kappa\sqrt{\log h}$ with a constant $v' > 0$. Furthermore, we have the following updates:

$$\phi_{t+1}^{\alpha}{}^2 + \psi_{t+1}^{\alpha}{}^2 =(\sqrt{\phi_t^{\alpha 2}+\psi_t^{\alpha 2}}(1-\eta_t\Gamma_t^\alpha+\eta_t\Upsilon_t^\alpha))^2 + \eta_t^2\frac{\psi_t^{\alpha 2}}{\phi_t^{\alpha 2}+\psi_t^{\alpha 2}}\Xi_t^{\alpha 2} \tag{28a}$$

$$\phi_{t+1}^\alpha =\phi_t^\alpha(1-\eta_t\Gamma_t^\alpha) + \eta_t\frac{\phi_t^\alpha}{\sqrt{\phi_t^{\alpha 2}+\psi_t^{\alpha 2}}}\Upsilon_t^\alpha + \eta_t\frac{\psi_t^{\alpha 2}}{\phi_t^{\alpha 2}+\psi_t^{\alpha 2}}\Xi_t^\alpha. \tag{28b}$$

(28c)

Since $\Upsilon_t^\alpha > 0$ and is bounded by Γ_t^α and $\Xi_t^\alpha = o(\Gamma_t^\alpha)$, if $\phi_t^\alpha < 0$ and $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \geq 2$, we have:

$$\sqrt{{\phi_{t+1}^\alpha}^2 + {\psi_{t+1}^\alpha}^2} \geq \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}(1 - \eta_t \Gamma_t^\alpha) \tag{29a}$$

$$\phi_{t+1}^\alpha \geq \phi_t^\alpha (1 - \frac{\eta_t}{2}\Gamma_t^\alpha - \eta_t o(\Gamma_t^\alpha)). \tag{29b}$$

Hence, after $t \geq v_1 \log(\kappa \log h)$ steps, we have that $\phi_t^\alpha \geq -\frac{1}{100}$ and $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \geq 2$.

Next we show that from this point on ϕ_t^α and $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}$, the conditions in this Lemma continue to be satisfied. We have:

$$\sqrt{{\phi_{t+1}^\alpha}^2 + {\psi_{t+1}^\alpha}^2} \geq \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} - \eta_t \Gamma_t^\alpha (\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} - 1) + \eta_t (\Upsilon_t^\alpha - \Gamma_t^\alpha), \tag{30}$$

where

$$\Upsilon_t^\alpha - \Gamma_t^\alpha = \frac{1}{s^\alpha} \sum_{i,j:\varrho_{i,j} \neq 0} \delta_{ij} \sqrt{\vartheta_{i,j}} (\hat{\sigma}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}} \frac{\phi_t^\alpha}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}) - \hat{\sigma}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}})). \tag{31}$$

Once $\phi_t^\alpha \geq -\frac{1}{100}$, we have:

$$|\hat{\sigma}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}} \frac{\phi_t^\alpha}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}) - \hat{\sigma}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}})| \leq 2 \frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}}. \tag{32}$$

Hence, we have that if $\phi_t^\alpha \geq -\frac{1}{100}$, then $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \geq 1 - o(1)$.

Next we discuss that ϕ_t^α continues to be larger than $-\frac{1}{100}$. First, if $\phi_t^\alpha > 0$, then $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \geq 1 - o(1)$ remains. Furthermore, if $\phi_t^\alpha \in [-\frac{1}{100}, 0)$, then Ξ_t^α is non negative and is at least $\frac{1}{4s^\alpha} \sum_{i,j,\varrho_{i,j} \neq 0} \delta_{ij} \frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}}$. Hence, we have:

$$\phi_{t+1}^\alpha \geq \phi_t^\alpha - \eta_t \Gamma_t^\alpha \phi_t^\alpha (1 - \frac{1}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}) + \eta_t \frac{\phi_t^\alpha}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}} (\Upsilon_t^\alpha - \Gamma_t^\alpha) + \eta_t \frac{\psi_t^{\alpha 2}}{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \frac{\sum_{i,j,\varrho_{i,j} \neq 0} \delta_{ij} \frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}}}{4s^\alpha}. \tag{33}$$

Using $|\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} - 1| = O(1)$ and the fact that if $\phi_t^\alpha \in [-\frac{1}{100}, 0)$ and $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \geq 1 - o(1)$, then $|\frac{\psi_t^\alpha}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}| \geq \frac{1}{2}$, we have that $\phi_{t+1}^\alpha \geq \phi_t^\alpha$.

Lemma 6. *If the conditions in Lemma 1 and Lemma 3 hold, then there is an absolute constant v_1, such that for all $t \geq v_1 \log(\kappa \log h)$, either $|\psi_t^\alpha| \leq \frac{\epsilon}{4h}$ and $\|\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} - 1\| \leq \frac{\epsilon}{4h}$ or we have that*

$$\|\nabla L_{C,t}^\alpha (\phi_t^\alpha, \psi_t^\alpha)\|^2 \geq \mu_t^\alpha L_{C,t}^\alpha (\phi_t^\alpha, \psi_t^\alpha),$$

where $\mu_t^\alpha \geq v_2 \frac{\epsilon \Gamma_t^\alpha}{s^{\alpha 2} h^2}$ and $v_2 > 0$ is a constant.

Proof. We have

$$\begin{aligned} \|\nabla L_{C,t}^\alpha (\phi_t^\alpha, \psi_t^\alpha)\|^2 =& (\phi_t^\alpha \Gamma_t^\alpha - \frac{\phi_t^\alpha}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}} \Upsilon_t^\alpha - \frac{\psi_t^{\alpha 2}}{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \Xi_t^\alpha)^2 + (\psi_t^\alpha \Gamma_t^\alpha - \frac{\psi_t^\alpha}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}} \Upsilon_t^\alpha - \frac{\phi_t^\alpha \psi_t^\alpha}{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \Xi_t^\alpha)^2 \\ =& (\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \Gamma_t^\alpha - \Upsilon_t^\alpha)^2 + \frac{\psi_t^{\alpha 2}}{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \Xi_t^{\alpha 2}. \end{aligned} \tag{34}$$

On the other hand, the loss $L_{C,t}^\alpha (\phi_t^\alpha, \psi_t^\alpha)$ can be written as :

$$L_{C,t}^\alpha (\phi_t^\alpha, \psi_t^\alpha) = \frac{1}{2}(\phi_t^{\alpha 2} + \psi_t^{\alpha 2} + 1)\Gamma_t^\alpha - \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \Upsilon_t^\alpha \leq \frac{1}{2}(\phi_t^{\alpha 2} + \psi_t^{\alpha 2} + 1)\Gamma_t^\alpha. \tag{35}$$

Hence, we have:

$$\frac{\|\nabla L_{C,t}^\alpha (\phi_t^\alpha, \psi_t^\alpha)\|^2}{L_{C,t}^\alpha (\phi_t^\alpha, \psi_t^\alpha)} \geq \frac{(\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \Gamma_t^\alpha - \Upsilon_t^\alpha)^2}{\Gamma_t^\alpha} + 2 \frac{\psi_t^{\alpha 2} \Xi_t^{\alpha 2}}{(\phi_t^{\alpha 2} + \psi_t^{\alpha 2})(\phi_t^{\alpha 2} + \psi_t^{\alpha 2} + 1)\Gamma_t^\alpha}. \tag{36}$$

If $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} - 1 > \frac{\epsilon}{4h}$, then the first term above combined with $\Upsilon_t^\alpha \leq \Gamma_t^\alpha$ leads to

$$\begin{aligned}\frac{\|\nabla L_{C,t}^\alpha(\phi_t^\alpha, \psi_t^\alpha)\|^2}{L_{C,t}^\alpha(\phi_t^\alpha, \psi_t^\alpha)} &\geq \frac{(\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}\Gamma_t^\alpha - \Upsilon_t^\alpha)^2}{\Gamma_t^\alpha} \\ &\geq \frac{\Gamma_t^{\alpha 2}\epsilon^2}{16h^2\Gamma_t^\alpha} \\ &\geq \frac{\Gamma_t^\alpha \epsilon^2}{16h^2}.\end{aligned} \tag{37}$$

If $|\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} - 1| \leq \frac{\epsilon}{4h} \leq 2$ and $|\psi_t^\alpha| > \frac{\epsilon}{4h}$, then the second term leads to

$$\begin{aligned}\frac{\|\nabla L_{C,t}^\alpha(\phi_t^\alpha, \psi_t^\alpha)\|^2}{L_{C,t}^\alpha(\phi_t^\alpha, \psi_t^\alpha)} &\geq 2\frac{\psi_t^{\alpha 2}\Xi_t^{\alpha 2}}{(\phi_t^{\alpha 2} + \psi_t^{\alpha 2})(\phi_t^{\alpha 2} + \psi_t^{\alpha 2} + 1)\Gamma_t^\alpha} \\ &\geq 2\frac{\psi_t^{\alpha 2}\Xi_t^{\alpha 2}}{v_w^{\alpha 2}(v_w^{\alpha 2} + 1)\Gamma_t^\alpha} \\ &\geq 2\frac{\Xi_t^{\alpha 2}}{v_w^{\alpha 2}(v_w^{\alpha 2} + 1)\Gamma_t^\alpha}\frac{\epsilon^2}{16h^2} \\ &\geq 2\frac{\Gamma_t^{\alpha 2}}{4s^{\alpha 2}v_w^{\alpha 2}(v_w^{\alpha 2} + 1)\Gamma_t^\alpha}\frac{\epsilon^2}{16h^2} \\ &\geq 2\frac{\Gamma_t^\alpha}{4s^{\alpha 2}v_w^{\alpha 2}(v_w^{\alpha 2} + 1)}\frac{\epsilon^2}{16h^2} \\ &= \frac{\Gamma_t^\alpha \epsilon^2}{32s^{\alpha 2}v_w^{\alpha 2}(v_w^{\alpha 2} + 1)h^2}\end{aligned} \tag{38}$$

Finally, consider the case when $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} \leq 1 - \frac{\epsilon}{4h}$. We can assume that $|\Upsilon_t^\alpha - \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}\Gamma_t^\alpha| \leq \frac{\epsilon}{8h}\Gamma_t^\alpha$ since otherwise we get the same bound as in (37). In this case, we show that $\|\psi_t^\alpha\|$ must be at least $\frac{\epsilon}{4h}$ and hence the bound of (38) can be applicable. Using $\hat{\sigma}(\cdot)$ is convex in $[0, 1]$, we can get

$$s^\alpha \delta_{ij}(\sqrt{\vartheta_{i,j}}(\hat{\sigma}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}}\frac{\phi_t^\alpha}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}) - \hat{\sigma}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}}))) \geq \delta_{ij}\varrho_{i,j}\frac{\phi_t^\alpha - \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}\hat{\sigma}_{step}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}}). \tag{39}$$

Summing over i, j, we have

$$s^\alpha(\Upsilon_t^\alpha - \Gamma_t^\alpha) \geq \frac{\phi_t^\alpha - \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}\sum_{i,j}\delta_{ij}\varrho_{i,j}\hat{\sigma}_{step}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}}). \tag{40}$$

Substituting $\Upsilon_t^\alpha = \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}\Gamma_t^\alpha \pm \frac{\epsilon\Gamma_t^\alpha}{8h}$, we have

$$s^\alpha((\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} - 1)\Gamma_t^\alpha \pm \frac{\epsilon\Gamma_t^\alpha}{8h}) \geq \frac{\phi_t^\alpha - \sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}\sum_{i,j}\delta_{ij}\varrho_{i,j}\hat{\sigma}_{step}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}}). \tag{41}$$

Using the bound on $\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}$, the above implies that

$$\frac{\epsilon s^\alpha \Gamma_t^\alpha}{8h} \leq \frac{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}} - \phi_t^\alpha}{\sqrt{\phi_t^{\alpha 2} + \psi_t^{\alpha 2}}}\sum_{i,j}\delta_{ij}\varrho_{i,j}\hat{\sigma}_{step}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}}) \tag{42}$$

Noticing that $\Gamma_t^\alpha \geq \frac{1}{\pi s^\alpha}\|\tilde{\mathbf{A}}_{C,t}^\alpha \mathbb{1}\|_2$, we have

$$\frac{\sqrt{\phi_t^{\alpha 2}+\psi_t^{\alpha 2}}-\phi_t^\alpha}{\sqrt{\phi_t^{\alpha 2}+\psi_t^{\alpha 2}}} \geq \frac{\epsilon\|\tilde{\mathbf{A}}_{C,t}^\alpha \mathbb{1}\|_2}{8\pi h\sum_{i,j}\delta_{ij}\varrho_{i,j}\hat{\sigma}_{step}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}})} \tag{43}$$

Therefore, we have

$$\begin{aligned}
\psi_t^\alpha &\geq \sqrt{\phi_t^{\alpha 2}+\psi_t^{\alpha 2}}-\phi_t^\alpha \\
&\geq \frac{\epsilon\|\tilde{\mathbf{A}}_{C,t}^\alpha \mathbb{1}\|_2}{8\pi h\sum_{i,j}\delta_{ij}\varrho_{i,j}\hat{\sigma}_{step}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}})}\sqrt{\phi_t^{\alpha 2}+\psi_t^{\alpha 2}} \\
&\geq \frac{\epsilon\|\tilde{\mathbf{A}}_{C,t}^\alpha \mathbb{1}\|_2}{8\pi h\sum_{i,j}\delta_{ij}\varrho_{i,j}\hat{\sigma}_{step}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}})} \\
&\geq \frac{\epsilon\|\tilde{\mathbf{A}}_{C,t}^\alpha \mathbb{1}\|_2}{4hs^\alpha} \\
&> \frac{\epsilon}{4h}
\end{aligned} \tag{44}$$

where the second last inequality uses $\sum_{i,j}\delta_{ij}\varrho_{i,j}\hat{\sigma}_{step}(\frac{\varrho_{i,j}}{\sqrt{\vartheta_{i,j}}}) \leq s^\alpha$.

B.1.3 Convergence results. **Theorem 1.** (Convergence of Full-graph Training.) *In full-graph training, let $\mathbf{W}^{F^*}$ be the unknown parameter for a one-round GNN with $\|\mathbf{w}_j^{F^*}\| = 1$ for $j \in \{1,\ldots,h\}$, and let $L_C^F(\mathbf{W}^F)$ denote the population loss at $\mathbf{W}^F$. If the in-degree of each node is $O(n_C)$ and the out-degree of each node is $O(n)$, then for any $\epsilon \in (0,1)$, if $\mathbf{W}_0^F \sim N(0,\mathbf{I})$, with probability at least $1-he^{-v^F r}$, we have that $L_C^F(\mathbf{W}_T^F) \leq \epsilon^2$ provided $T \geq \log\frac{h}{\epsilon}O(\frac{h^2 o(n_C)}{\epsilon^2})$ with the absolute constants v^F.*

Proof. We analyze an arbitrary $j \in \{1,\ldots,h\}$ and the iterates of the corresponding $\mathbf{w}_j^F$ vector. Setting $\kappa = 1$, we have from Lemma 4 that the smoothness parameter v_L^F of the loss function is

$$v_L^F \leq 4(1+\sqrt{4\pi+3}+o(\frac{1}{n_C})) \tag{45}$$

Hence, for any $t > 0$,

$$\begin{aligned}
L_{C,t+1}^F(\mathbf{w}_{j,t+1}^F) \leq& L_{C,t}^F(\mathbf{w}_{j,t}^F) + \nabla L_{C,t}^F(\mathbf{w}_{j,t}^F)(\mathbf{w}_{j,t+1}^F-\mathbf{w}_{j,t}^F) + \frac{v_L^F}{2}\|\mathbf{w}_{j,t+1}^F-\mathbf{w}_{j,t}^F\|^2 \\
\leq& L_{C,t}^F(\mathbf{w}_{j,t}^F) - \eta\|\nabla L_{C,t}^F(\mathbf{w}_{j,t}^F)\|^2 + \frac{\eta^2 v_L^F}{2}\|\nabla L_{C,t}^F(\mathbf{w}_{j,t}^F)\|^2 \\
=& L_{C,t}^F(\mathbf{w}_{j,t}^F) - \eta\|\nabla L_{C,t}^F(\mathbf{w}_{j,t}^F)\|^2(1-\frac{\eta v_L^F}{2}).
\end{aligned} \tag{46}$$

Restrict further the range of η in Lemma 2 as $\eta \leq \frac{1}{4(1+\sqrt{4\pi+3}+o(\frac{1}{n_C}))}$, namely $\eta = O(\omega(n_C))$, we have $\eta_t v_L^F \leq 1$. Furthermore, using Lemma 6, we have

$$\begin{aligned}
L_{C,t+1}^F(\mathbf{w}_{j,t+1}^F) \leq& L_{C,t}^F(\mathbf{w}_{j,t}^F) + \nabla L_{C,t}^F(\mathbf{w}_{j,t}^F)(\mathbf{w}_{j,t+1}^F-\mathbf{w}_{j,t}^F) + \frac{v_L^F}{2}\|\mathbf{w}_{j,t+1}^F-\mathbf{w}_{j,t}^F\|^2 \\
\leq& L_{C,t}^F(\mathbf{w}_{j,t}^F)(1-\eta\mu^F) \\
\leq& L_{C,0}^F(\mathbf{w}_{j,0}^F)(1-\eta\mu^F)^t.
\end{aligned} \tag{47}$$

Hence after $T \geq \log\frac{h}{\epsilon}O(\frac{h^2 n_C^{5/2} o(1/n_C)}{\epsilon^2 n^{1/2}}) = \log\frac{h}{\epsilon}O(\frac{h^2 o(n_C^{3/2})}{\epsilon^2 n^{1/2}})$ time steps, we either have $L_{C,T}^F(\mathbf{w}_{j,T}^F) \leq \epsilon^2/h$, or that $\psi_t^F \leq \frac{\epsilon}{4h}$ and $\phi_t^{F2}+\psi_t^{F2}-1 \leq \frac{\epsilon}{4h}$. The latter implies that $\|\mathbf{w}_{j,t}^F-\mathbf{w}_{j,t}^{F^*}\|^2 \leq \frac{\epsilon^2}{2h}$. In addition, it is easy to see that $L_{C,t}^F(\mathbf{w}_{j,t}^F) \leq \|\mathbf{w}_{j,t}^F-\mathbf{w}_{j,t}^{F^*}\|^2$. Hence, if the latter happens, then $L_{C,T}^F(\mathbf{w}_{j,t}^F) \leq \epsilon^2/h$. Hence $L_{C,T}^F(\mathbf{W}_T^F) \leq \epsilon^2$.

Theorem 2. (Convergence of Mini-batch Training.) *In mini-batch training, let $\mathbf{W}^{M^*}$ be the unknown parameter for a one-round GNN with $\|\mathbf{w}_j^{M^*}\| = 1$ for $j \in \{1,\ldots,h\}$, let $L_C^M(\mathbf{W}^M)$ denote the population loss at $\mathbf{W}^M$. If the in-degree of each node is $O(\beta)$, the out-degree of each node is $O(b)$, and $\mathbf{W}_0^M \sim N(0,\mathbf{I})$, with probability at least $1-he^{-v^M r}$, then we have that $L_C^M(\mathbf{W}_T^M) \leq \epsilon^2$ provided $T \geq \log\frac{h}{\epsilon}\log\frac{h}{\epsilon}O(\frac{h^2 o(b^{3/2})}{\epsilon^2\beta^{1/2}}))$ time steps for the absolute constant v^M for any $\epsilon \in (0,1)$.*

Proof. Similar to Theorem 1, we have the the smoothness parameter $v_{L,t}^M$ of the loss function is

$$v_{L,t}^M \le 4(1+\sqrt{4\pi+3}+o(\frac{1}{b})) \tag{48}$$

Restrict further the range of η in Lemma 2 as $\eta \le \frac{1}{4(1+\sqrt{4\pi+3}+o(\frac{1}{b}))}$, namely $\eta = O(\omega(b))$, we have $\eta v^M \le 1$.

Hence, for any $t > 0$,

$$\begin{aligned}
L_{C,t+1}^M(\mathbf{w}_{j,t+1}^M) \le& L_{C,t}^M(\mathbf{w}_{j,t}^M) + \nabla L_{C,t}^M(\mathbf{w}_{j,t}^M)(\mathbf{w}_{j,t+1}^M - \mathbf{w}_{j,t}^M) + \frac{v_{L,t}^M}{2}\|\mathbf{w}_{j,t+1}^M - \mathbf{w}_{j,t}^M\|^2 \\
\le& L_{C,t}^M(\mathbf{w}_{j,t}^M) - \eta_t\|\nabla L_{C,t}^M(\mathbf{w}_{j,t}^M)\|^2 + \frac{\eta_t^2 v_L^M}{2}\|\nabla L_{C,t}^M(\mathbf{w}_{j,t}^M)\|^2 \\
=& L_{C,t}^M(\mathbf{w}_{j,t}^M) - \eta_t\|\nabla L_{C,t}^M(\mathbf{w}_{j,t}^M)\|^2(1 - \frac{\eta_t v_{L,t}^M}{2}) \\
\le& L_{C,t}^M(\mathbf{w}_{j,t}^M)(1-\eta_t\mu_t^M) \\
\le& L_{C,0}^M(\mathbf{w}_{j,0}^M)\prod_{\tau=1}^{t}(1-\eta_\tau\mu_\tau^M) \\
\le& L_{C,0}^M(\mathbf{w}_{j,0}^M)(1-\frac{1}{t}\sum_{\tau=1}^{t}\eta_\tau\mu_\tau^M)^t,
\end{aligned} \tag{49}$$

where the last inequality can be proved: $f(x) = \log(1-x)$ is a concave function on $0 < x < 1$, then, for $0 < x_i < 1$ with $i = \{1,\dots,n\}$, we have $f(\frac{1}{n}\sum_{i=1}^n x_i) \ge \frac{1}{n}\sum_{i=1}^n f(x_i)$, which can be written as $\log(1-\frac{1}{n}\sum_{i=1}^n x_i) \ge \frac{1}{n}\sum_{i=1}^n \log(1-x_i)$. Therefore, we have $(1-\frac{1}{n}\sum_{i=1}^n x_i)^n \ge \prod_{i=1}^n(1-x_i)$.

Using $\mu_t^M \ge v_2\frac{\epsilon^2\Gamma_t^M}{h^2b^2} \ge v_2\Theta(\frac{\epsilon^2\beta^{1/2}}{h^2})\Omega(b^{-5/2})$ and $\eta_t = O(\omega(b))$, we have after $T \ge \log\frac{h}{\epsilon}O(\frac{h^2 o(b^{3/2})}{\epsilon^2\beta^{1/2}})$ time steps, $L_{C,T}^F(\mathbf{W}_T^F) \le \epsilon^2$.

B.1.4 Discussions. We analyze the impact of embedding aggregation on GNN convergence. First, we examine the effects of batch size and fan-out size in embedding aggregation. Next, we compare the convergence behaviors of different loss functions to investigate the role of embedding aggregation. Finally, we discuss why GNN convergence differs from DNN convergence by using the embedding aggregation.

Discussion 1: batch size and fan-out size. In mini-batch training, we bound the in-degree d of each node as $O(\beta)$ and the out-degree e of each node as $O(b)$. For the node i, each element in $\tilde{\mathbf{a}}_{\text{train},i}\mathbb{1}$, derived from the embedding aggregation, becomes $\frac{1}{\sqrt{d}\sqrt{e}} = \Omega\left(\beta^{-1/2}b^{-1/2}\right)$. Adding at most β elements, we have $\tilde{\mathbf{a}}_{\text{train},i}\mathbb{1} = \Theta\left(\beta^{1/2}\right)\Omega(b^{-1/2})$. Hence, batch sizes and fan-out sizes exhibit different trends in the term $\tilde{\mathbf{a}}_{\text{train},i}\mathbb{1}$, affecting the ratio μ^{mini} under PL conditions and leading to Informal Theorem 2.

Discussion 2: MSE v.s. CE in GNNs. We further identify how embedding aggregation impact GNN convergence behaviors under different loss functions. We discuss MSE and CE, two commonly used loss functions in node classification tasks. We analyze the ratio μ_t^{mini} in PL conditions. For simplicity, we consider $\hat{y}_i = \tilde{a}_i x w$ and $\hat{y}_i = \tilde{a}_i x w^*$ with no activations in this discussion.

For the *MSE*, the loss function is $l_{\text{MSE}} = \frac{1}{b}\sum_{i=1}^b(\hat{y}_i - y_i)^2$, with a gradient of $\partial l_{\text{MSE}}/\partial w = \frac{1}{b}\sum_{i=1}^b(\hat{y}_i - y_i)\tilde{a}_i x$. We observe that the embedding aggregation $\tilde{a}_i x$ appears in the numerator of μ^{mini} after the division cancellation of $\mu^{\text{mini}} = \|\partial l_{\text{MSE}}/\partial w\|^2/l_{\text{MSE}}$, where ax_i contains the term $\tilde{\mathbf{a}}_{\text{train},i}\mathbb{1}$ as analyzed in Discussion 4.1. Then the iteration-to-loss T includes $1/(\mu^{mini}\eta)$, where $\eta = O(\omega(b))$ and the variations of η with respect to batch size cannot offset the trend of μ^{mini} driven by batch size. This leads to Informal Theorem 2.

For the *CE*, the loss function is $l_{\text{CE}} = \frac{1}{b}\sum_{i=1}^b(y_i\log\hat{y}_i + (1-y_i)\log(1-\hat{y}_i))$, with a gradient of $\partial l_{\text{CE}}/\partial w = \frac{1}{b}\sum_{i=1}^b(\frac{y_i}{\hat{y}_i} - \frac{1-y_i}{1-\hat{y}_i})\tilde{a}_i x$. We observe that the impact of embedding aggregation on the gradient is mitigated by the division $\tilde{a}_i x/\hat{y}_i$, while its impact on the loss is reduced by the logarithmic function, though to a lesser extent than in the gradient. Consequently, μ^{mini} is less sensitive to fan-out size and batch size compared to MSE. Then the iteration-to-loss T includes $1/(\mu^{mini}\eta)$, where η approximates $O(\omega(b^{1/2}))$ and the variations of η with respect to batch size affect the trend of μ^{mini} driven by batch size. Hence, we derive that iteration-to-loss in GNN convergence with CE decreases as the batch size or the fan-out size grows.

Discussion 3: DNNs v.s. GNNs in convergence. GNN convergence with MSE exhibits an opposite trend to DNN convergence with MSE [39, 84] when the batch size varies, as GNNs consider the impact of embedding aggregation in the gradient and loss value. However, GNNs with CE has the same convergence trend as DNNs with CE [35, 84] as the impact of embedding aggregation is mitigated in the gradient as analyzed in Discussion 4.2.

Hence, embedding aggregation differentiates DNN and GNN convergence, increasing complexity in GNN convergence.

B.2 Generalization of Mini-batch Training

B.2.1 General PAC-Bayesian theorem. **Lemma 7.** ([42]) *For any two distributions $\mathcal{P}$ and $\mathcal{Q}$ defined on the hypothesis space, and any function $f(\cdot) \in \mathbb{R}$ with* **dom**f *in this hypothesis space, we have:*

$$\mathbb{E}_{x\sim \mathcal{Q}} \leq D_{KL}(\mathcal{Q}\|\mathcal{P}) + \mathbb{E}_{x\sim\mathcal{P}} e^{f(x)}. \tag{50}$$

Lemma 8. ([42]) *Suppose $X_1, X_2, \ldots, X_n$ are independent random variables with $a_i \leq X_i \leq b_i$, $\forall i = 1, 2, \ldots, n$. Let $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. Then, for any $v > 0$,*

$$\rho(|\overline{X} - \mathbb{E}\overline{X}| > v) \leq 2e^{-\frac{n^2 v^2}{\sum_{i=1}^{n}(b_i - a_i)^2}}. \tag{51}$$

Lemma 9. ([42]) *If X is a centered random variable, i.e., $\mathbb{E}X = 0$, and if $\exists v_1 > 0$, for any $v_2 > 0$,*

$$\rho(|X| > v_2) \leq 2e^{-v_1 v_2^2}. \tag{52}$$

Then, for any $v_u > 0$,

$$\mathbb{E}e^{v_u X} \leq e^{\frac{v_u^2}{2v_1}}. \tag{53}$$

Theorem 3 (PAC-Bayesian Generalization Theorem). *For any $v_u > 0$, for any "prior" distribution $\mathcal{P}$ of the output hypothesis function of a GNN that is independent of node labels from training dataset C, with probability at least $1 - v_G$, for the distribution $\mathcal{Q}$ of the output hypothesis function of a GNN, we have:*

$$L_{\mathcal{Z}}^{M}(\mathbf{W}^M;\mathcal{Q}) \leq \hat{L}_C(\mathbf{W}^M;\mathcal{Q}) + \frac{1}{v_u}(D_{KL}(\mathcal{Q}\|\mathcal{P}) + ln\frac{1}{v_G} + \frac{v_u^2}{4n_C} + U(v_u)).$$

Proof. We prove the result by upper-bounding the quantity $v_u(L_{\mathcal{Z}}^{M}(\mathbf{W}^M;\mathcal{Q}) - \hat{L}_C(\mathbf{W}^M;\mathcal{Q}))$. First, we have

$$\begin{aligned}
&v_u(L_{\mathcal{Z}}^{M}(\mathbf{W}^M;\mathcal{Q}) - \hat{L}_C(\mathbf{W}^M;\mathcal{Q}))\\
\leq &\mathbb{E}_{\mathbf{W}^M\sim\mathcal{Q}} v_u(L_{\mathcal{Z}}^{M}(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M))\\
\leq &D_{KL}(\mathcal{Q}\|\mathcal{P}) + ln\mathbb{E}_{\mathbf{W}^M\sim\mathcal{P}} e^{(L_{\mathcal{Z}}^{M}(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M))},
\end{aligned} \tag{54}$$

where the last inequality uses Lemma 7.

Next, we upper-bound the second term in the RHS of (54). Here the term $\Lambda = \mathbb{E}_{\mathbf{W}^M\sim\mathcal{P}} e^{(L_{\mathcal{Z}}^{M}(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M))}$ is a random variable with the randomness coming from the sample of node labels in training dataset, and $\mathcal{P}$ is independent of node labels $\mathbf{y}$ from training dataset. Applying Markov's inequality to the term Λ, we have for any $v_G > 0$, with probability at least $1 - v_G$ over $\mathbf{y} \sim C$,

$$\Lambda \leq \frac{1}{v_G}\mathbb{E}_{\mathbf{y}\sim C}\Lambda \tag{55}$$

and hence,

$$ln\Lambda \leq ln\frac{1}{v_G}\mathbb{E}_{\mathbf{y}\sim C}\Lambda = ln\frac{1}{v_G} + ln\mathbb{E}_{\mathbf{y}\sim C}\Lambda. \tag{56}$$

Then we need to upper-bound $ln\mathbb{E}_{\mathbf{y}\sim C}\Lambda$. We can re-write it as:

$$ln\mathbb{E}_{\mathbf{y}\sim C}\Lambda = ln\mathbb{E}_{\mathbf{y}\sim C}\mathbb{E}_{\mathbf{W}^M\sim\mathcal{P}} e^{(L_{\mathcal{Z}}^{M}(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M))} = ln\mathbb{E}_{\mathbf{W}^M\sim\mathcal{P}}\mathbb{E}_{\mathbf{y}\sim C} e^{(L_{\mathcal{Z}}^{M}(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M))}. \tag{57}$$

For a fixed model with model parameters $\mathbf{W}^M$, we have

$$\begin{aligned}
&\mathbb{E}_{\mathbf{y}\sim C} e^{(L_{\mathcal{Z}}^{M}(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M))}\\
=&\mathbb{E}_{\mathbf{y}\sim C} e^{(L_{\mathcal{Z}}^{M}(\mathbf{W}^M) - L_C(\mathbf{W}^M) + L_C(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M))}\\
=&\mathbb{E}_{\mathbf{y}\sim C} e^{(L_{\mathcal{Z}}^{M}(\mathbf{W}^M) - L_C(\mathbf{W}^M))} e^{(L_C(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M))}\\
=&e^{(L_{\mathcal{Z}}^{M}(\mathbf{W}^M) - L_C(\mathbf{W}^M))}\mathbb{E}_{\mathbf{y}\sim C} e^{(L_C(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M))}.
\end{aligned} \tag{58}$$

In the following, we wil give an upper bound on $\mathbb{E}_{\mathbf{y}\sim C} e^{(L_C(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M))}$ that is independent of $\mathbf{W}^M$. For the entire training dataset, $\hat{L}_C(\mathbf{W}^M)$ can be written as:

$$\hat{L}_C(\mathbf{W}^M) = \frac{1}{n_C}\sum_{i\in C} \|\hat{\mathbf{y}}_i^M - \mathbf{y}_i\|_F^2, \tag{59}$$

where the node labels are independently sampled. Hence, $\hat{L}_C(\mathbf{W}^M)$ is the empirical mean of n_C independent Bernoulli random variables and $L_C(\mathbf{W}^M)$ is the expectation of $\hat{L}_C(\mathbf{W}^M)$. By Lemma 8, for any $v_1 > 0$,

$$\rho(|L_C(\mathbf{W}^M) - \hat{L}_C(\mathbf{W}^M)| \geq v_1) \leq 2e^{-2n_C v_1^2}, \tag{60}$$

and hence, by Lemma 9, we have

$$\mathbb{E}_{\mathbf{y}\sim C} e^{v_u(L_C(\mathbf{W}^M)-\hat{L}_C(\mathbf{W}^M))} \leq e^{\frac{v_u^2}{4n_C}}. \tag{61}$$

Therefore, we have

$$\begin{aligned} ln\Lambda \leq & ln\mathbb{E}_{\mathbf{W}^M\sim\mathcal{P}} e^{(L_{\mathcal{Z}}^M(\mathbf{W}^M)-L_C(\mathbf{W}^M))} e^{\frac{v_u^2}{4n_C}} \\ = & U(v_u)) + \frac{v_u^2}{4n_C}. \end{aligned} \tag{62}$$

Finally, we get

$$\begin{aligned} & v_u(L_{\mathcal{Z}}^M(\mathbf{W}^M;Q) - \hat{L}_C(\mathbf{W}^M;Q)) \\ \leq & D_{KL}(Q\|\mathcal{P}) + ln\mathbb{E}_{\mathbf{W}^M\sim\mathcal{P}} e^{(L_{\mathcal{Z}}^M(\mathbf{W}^M)-\hat{L}_C(\mathbf{W}^M))} \\ \leq & D_{KL}(Q\|\mathcal{P}) + ln\frac{1}{v_G} + \frac{v_u^2}{4n_C} + U(v_u). \end{aligned} \tag{63}$$

Hence, we have the final result

$$L_{\mathcal{Z}}^M(\mathbf{W}^M;Q) \leq \hat{L}_C(\mathbf{W}^M;Q) + \frac{1}{v_u}(D_{KL}(Q\|\mathcal{P}) + ln\frac{1}{v_G} + \frac{v_u^2}{4n_C} + U(v_u). \tag{64}$$

Lemma 10 (Restatement of Lemma 4.6.) *For any $v_u > 0$, assume the "prior" $\mathcal{P}$ on hypothesis space is defined by sampling the model parameters. If the in-degree of each node is $O(\beta)$ and the out-degree of each node is $O(b)$, we have:*

$$U(v_u) \leq v_u\Delta(\beta, b),$$

and,

$$\begin{aligned} \Delta(\beta_1, b) \geq & \Delta(\beta_2, b), \beta_1 \leq \beta_2, \\ \Delta(\beta, b_1) \geq & \Delta(\beta, b_2), b_1 \leq b_2, \end{aligned}$$

where β is the fan-out size in uniform neighbor sampling and b is the batch size.

Proof. Recall that

$$U(v_u) = ln\mathbb{E}_{\mathbf{W}^M\sim\mathcal{P}} e^{v_u(L_{\mathcal{Z}}^M(\mathbf{W}^M)-L_C(\mathbf{W}^M))}. \tag{65}$$

First, we focus on the term $L_{\mathcal{Z}}^M(\mathbf{W}^M) - L_C(\mathbf{W}^M)$. Set $l_C^M(\mathbf{y}_i) = \|\hat{\mathbf{y}}_i^M - \mathbf{y}_i\|_F^2, \forall i \in C$ and $l_{\mathcal{Z}}^M(\mathbf{y}_j) = \|\hat{\mathbf{y}}_i^M - \mathbf{y}_j\|_F^2, \forall i \in \mathcal{Z}$. Then we have

$$\begin{aligned} L_{\mathcal{Z}}^M(\mathbf{W}^M) - L_C(\mathbf{W}^M) = & \mathbb{E}_{\mathbf{y}\sim\mathcal{Z}}[\frac{1}{n_{\mathcal{Z}}}\sum_{j\in\mathcal{Z}} l_{\mathcal{Z}}^M(\mathbf{y}_j)] - \mathbb{E}_{\mathbf{y}\sim C}[\frac{1}{n_C}\sum_{i\in C} l_C^M(\mathbf{y}_i)] \\ = & \frac{1}{n_{\mathcal{Z}}}\sum_{j\in\mathcal{Z}} l_{\mathcal{Z}}^M(\mathbf{y}_j)\rho_{\mathcal{Z}}(\mathbf{y}_j) - \frac{1}{n_C}\sum_{i\in C} l_C^M(\mathbf{y}_i)\rho_C(\mathbf{y}_i) \end{aligned} \tag{66}$$

Furthermore, we set $f(\mathbf{y}_i) = -\frac{1}{n_C} l_C^M(\mathbf{y}_i)$ with $\forall i \in C$ and $g(\mathbf{y}_j) = \frac{1}{n_{\mathcal{Z}}} l_{\mathcal{Z}}^M(\mathbf{y}_j)$ with $\forall j \in \mathcal{Z}$ and $n_{min} = \min\{n_C, n_{\mathcal{Z}}\}$. As each element in $\tilde{\mathbf{a}}_{\mathcal{Z},i}^F$ and $\tilde{\mathbf{a}}_{C,i}^M$ is more than 0, we have $\sigma(\tilde{\mathbf{a}}_{\mathcal{Z},j}^F \mathbf{X}\mathbf{W}^{M^\top}) = \tilde{\mathbf{a}}_{\mathcal{Z},j}^F \sigma(\mathbf{X}\mathbf{W}^{M^\top})$, $\sigma(\tilde{\mathbf{a}}_{\mathcal{Z},j}^F \mathbf{X}(\mathbf{W}^{M^*})^\top) = \tilde{\mathbf{a}}_{\mathcal{Z},j}^F \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top)$, $\sigma(\tilde{\mathbf{a}}_{C,i}^M \mathbf{X}\mathbf{W}^{M^\top}) = \tilde{\mathbf{a}}_{C,i}^M \sigma(\mathbf{X}\mathbf{W}^{M^\top})$ and $\sigma(\tilde{\mathbf{a}}_{C,i}^M \mathbf{X}(\mathbf{W}^{M^*})^\top) = \tilde{\mathbf{a}}_{C,i}^M \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top)$.

Hence, we have:

$$
\begin{aligned}
f(\mathbf{y}_i) + g(\mathbf{y}_j) =& \frac{1}{n_{\mathcal{Z}}} l^M_{\mathcal{Z}}(\mathbf{y}_j) - \frac{1}{n_C} l^M_C(\mathbf{y}_i) \\
\leq& \frac{1}{n_{min}} (l^M_{\mathcal{Z}}(\mathbf{y}_j) - l^M_C(\mathbf{y}_i)) \\
=& \frac{1}{n_{min}} (\|\sigma(\tilde{\mathbf{a}}^F_{\mathcal{Z},j}\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\tilde{\mathbf{a}}^F_{\mathcal{Z},j}\mathbf{X}(\mathbf{W}^{M^*})^\top)\|_F^2 - \|\sigma(\tilde{\mathbf{a}}^M_{C,i}\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\tilde{\mathbf{a}}^M_{C,i}\mathbf{X}(\mathbf{W}^{M^*})^\top)\|_F^2) \\
=& \frac{1}{n_{min}} (\|\tilde{\mathbf{a}}^F_{\mathcal{Z},j}(\sigma(\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top))\|_F^2 - \|\tilde{\mathbf{a}}^M_{C,i}(\sigma(\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top))\|_F^2) \\
=& \frac{1}{n_{min}} (Tr((\sigma(\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top))^\top (\tilde{\mathbf{a}}^F_{\mathcal{Z},j})^\top \tilde{\mathbf{a}}^F_{\mathcal{Z},j} (\sigma(\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top))) \\
&- Tr((\sigma(\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top))^\top (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i} (\sigma(\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top)))) \\
=& \frac{1}{n_{min}} Tr((\sigma(\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top))^\top ((\tilde{\mathbf{a}}^F_{\mathcal{Z},j})^\top \tilde{\mathbf{a}}^F_{\mathcal{Z},j} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}) (\sigma(\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top))) \\
=& \frac{1}{n_{min}} Tr(((\tilde{\mathbf{a}}^F_{\mathcal{Z},j})^\top \tilde{\mathbf{a}}^F_{\mathcal{Z},j} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}) (\sigma(\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top)) (\sigma(\mathbf{X}\mathbf{W}^{M^\top}) - \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top))^\top) \\
\leq& \frac{1}{n_{min}} Tr(|(\tilde{\mathbf{a}}^F_{\mathcal{Z},j})^\top \tilde{\mathbf{a}}^F_{\mathcal{Z},j} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}| (\sigma(\mathbf{X}\mathbf{W}^{M^\top}) + \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top)) (\sigma(\mathbf{X}\mathbf{W}^{M^\top}) + \sigma(\mathbf{X}(\mathbf{W}^{M^*})^\top))^\top) \\
\leq& \frac{1}{n_{min}} Tr(|(\tilde{\mathbf{a}}^F_{\mathcal{Z},j})^\top \tilde{\mathbf{a}}^F_{\mathcal{Z},j} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}| \mathbf{X}(\mathbf{W}^{M^\top} + (\mathbf{W}^{M^*})^\top)(\mathbf{W}^{M^\top} + (\mathbf{W}^{M^*})^\top)^\top \mathbf{X}^\top) \\
\leq& \frac{1}{n_{min}} Tr(|(\tilde{\mathbf{a}}^F_{\mathcal{Z},j})^\top \tilde{\mathbf{a}}^F_{\mathcal{Z},j} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}| \|\mathbf{X}\|(\|\mathbf{W}^M\|^2 + \|\mathbf{W}^{M^*}\|^2)) \\
\leq& \frac{v_x(v_w+1)}{n_{min}} Tr(|(\tilde{\mathbf{a}}^F_{\mathcal{Z},j})^\top \tilde{\mathbf{a}}^F_{\mathcal{Z},j} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}|) \\
\leq& \frac{v_x(v_w+1)}{n_{min}} (Tr(|(\tilde{\mathbf{a}}^F_{\mathcal{Z},j})^\top \tilde{\mathbf{a}}^F_{\mathcal{Z},j} - (\tilde{\mathbf{a}}^F_{C,i})^\top \tilde{\mathbf{a}}^F_{C,i}|) + Tr(|(\tilde{\mathbf{a}}^F_{C,i})^\top \tilde{\mathbf{a}}^F_{C,i} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}|)) = \delta(\mathbf{y}_i, \mathbf{y}_j, \beta, b)
\end{aligned} \tag{67}
$$

where the penultimate expression is exactly the distance function defined in Definition 3., and $\delta^F_{i,j} = Tr(|(\tilde{\mathbf{a}}^F_{\mathcal{Z},j})^\top \tilde{\mathbf{a}}^F_{\mathcal{Z},j} - (\tilde{\mathbf{a}}^F_{C,i})^\top \tilde{\mathbf{a}}^F_{C,i}|)$ is a constant based on the split of training and testing.

Hence, we have

$$
\begin{aligned}
L^M_{\mathcal{Z}}(\mathbf{W}^M) - L_C(\mathbf{W}^M) =& \frac{1}{2n_{\mathcal{Z}}} \sum_{j \in \mathcal{Z}} l^M_{\mathcal{Z}}(\mathbf{y}_j)\rho_{\mathcal{Z}}(\mathbf{y}_j) - \frac{1}{2n_C} \sum_{i \in C} l^M_C(\mathbf{y}_i)\rho_C(\mathbf{y}_i) \\
\leq& \Delta_{C,\mathcal{Z}}(\beta, b) = \min \sum_{i \in C} \sum_{j \in \mathcal{Z}} \theta_{i,j} \delta(\mathbf{y}_i, \mathbf{y}_j, \beta, b) \\
=& \min \sum_{i \in C} \sum_{j \in \mathcal{Z}} \theta_{i,j} \frac{v_x(v_w+1)}{2n_{min}} (\delta^F_{i,j} + Tr(|(\tilde{\mathbf{a}}^F_{C,i})^\top \tilde{\mathbf{a}}^F_{C,i} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}|))
\end{aligned} \tag{68}
$$

As the trace is required, we mainly focus on the elements of $|(\tilde{\mathbf{a}}^F_{C,i})^\top \tilde{\mathbf{a}}^F_{C,i} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}|$. Here we have the normalized adjacency matrix of a graph with n nodes as:

$$
\tilde{\mathbf{A}} = \begin{bmatrix} \frac{1}{\sqrt{d^{\text{in}}_1}} & & \\ & \ddots & \\ & & \frac{1}{\sqrt{d^{\text{in}}_b}} \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{b1} & \cdots & a_{bn} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{d^{\text{out}}_1}} & & \\ & \ddots & \\ & & \frac{1}{\sqrt{d^{\text{out}}_n}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{d^{\text{in}}_1}\sqrt{d^{\text{out}}_1}} a_{11} & \cdots & \frac{1}{\sqrt{d^{\text{in}}_1}\sqrt{d^{\text{out}}_n}} a_{1n} \\ \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{d^{\text{out}}_1}\sqrt{d^{\text{in}}_b}} a_{s1} & \cdots & \frac{1}{\sqrt{d^{\text{in}}_b}\sqrt{d^{\text{out}}_n}} a_{bn} \end{bmatrix}, \tag{69}
$$

where $a_{ij} \in \{0, 1\}$ represents if node i connects with node j (1) or not (0), and $d^{\text{in}}_i \geq 0$ represents the degree of node i. Noted that when *beta* increases, we have four situations: a^M_{ij} keeps as 0 given $a^F_{ij} = 0$, a^M_{ij} keeps as 0 given $a^F_{ij} = 1$, a^M_{ij} keeps as 1 given $a^F_{ij} = 1$, as well as a^M_{ij} becomes 1 from 0 given $a^F_{ij} = 1$.

We fix the out-degree $d^{\text{out},M}_j$: if $d^{\text{in},M}_i \leq \beta$, then we have $\tilde{\mathbf{a}}^M_{C,ij} = \frac{1}{\sqrt{d^{\text{in},M}_i d^{\text{out},M}_j}} a^M_{ij}$ and $\tilde{\mathbf{a}}^F_{C,ij} = \frac{1}{\sqrt{d^{\text{in},F}_i d^{\text{out},F}_j}} a^F_{ij}$. When β increases and a^M_{ij} changes as analyzed, $|(\frac{1}{\sqrt{d^{\text{in},F}_i d^{\text{out},F}_j}} a^F_{ij})^2 - (\frac{1}{\sqrt{d^{\text{in},M}_i d^{\text{out},M}_j}} a^M_{ij})^2|$ is non-increasing. If $d^{\text{in},M}_i > \beta$, then we have $\tilde{\mathbf{a}}^M_{C,ij} = \frac{1}{\sqrt{d^{\text{out},M}_j \beta}} a^M_{ij}$ and

$\tilde{a}^F_{C,ij} = \frac{1}{\sqrt{d_i^{\text{in},F} d_j^{\text{out},F}}} a^F_{ij}$. When β increases and a^M_{ij} changes as analyzed, $|(\frac{1}{\sqrt{d_j^{\text{out},M}\beta}} a^M_{ij})^2 - (\frac{1}{\sqrt{d_i^{\text{in},F} d_j^{\text{out},F}}} a^F_{ij})^2|$ is non-increasing. Hence, $|(\tilde{\mathbf{a}}^F_{C,i})^\top \tilde{\mathbf{a}}^F_{C,i} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}|$ is non-increasing when β increases.

We fix the in-degree $d_i^{\text{in},M}$: when $d_j^{\text{out},M} = O(b)$ increases and a^M_{ij} changes as analyzed, $|(\frac{1}{\sqrt{d_i^{\text{in},F} d_j^{\text{out},F}}} a^F_{ij})^2 - (\frac{1}{\sqrt{d_i^{\text{in},M} d_j^{\text{out},M}}} a^M_{ij})^2|$ is non-increasing. Hence, $|(\tilde{\mathbf{a}}^F_{C,i})^\top \tilde{\mathbf{a}}^F_{C,i} - (\tilde{\mathbf{a}}^M_{C,i})^\top \tilde{\mathbf{a}}^M_{C,i}|$ is non-increasing when the batch size b increases.

Therefore, $\delta(\mathbf{y}_i, \mathbf{y}_j, \beta, b)$ is non-increasing when β or b increases, thereby the upper bound $\Delta(\beta, b)$ of $L^M_{\mathcal{Z}}(\mathbf{W}^M) - L_C(\mathbf{W}^M)$ keeps the same or decreases when β or b increases.

Finally, Using Assumption 3.4., we have

$$\begin{aligned} U(v_u) =& ln\mathbb{E}_{\mathbf{W}^M \sim \mathcal{P}} e^{v_u(L^M_{\mathcal{Z}}(\mathbf{W}^M) - L_C(\mathbf{W}^M))} \\ \leq& ln\mathbb{E}_{\mathbf{W}^M \sim \mathcal{P}} e^{v_u \Delta(\beta,b)} \\ =& ln(e^{v_u\Delta(\beta,b)}) \\ =& v_u \Delta(\beta, b). \end{aligned} \tag{70}$$

Theorem 4. (Generalization bound for GNNs). *For any $v_u > 0$, assume the "prior" distribution $\mathcal{P}$ over hypothesis space is defined by sampling the model parameters from $\mathcal{N}(0, \kappa^2\mathbf{I})$ for $\kappa > 0$, which is independent of the training data on $\mathcal{S}$. If the in-degree of each node is $O(\beta)$ and the out-degree of each node is $O(b)$, with probability at least $1 - v_G$, for any "posterior" distribution $\mathcal{Q}$ over hypothesis space on one-round GNN under uniform neighbor sampling with the batch size b and the fan-out size β after T iterate updates, we have:*

$$L^M_{\mathcal{Z}}(\mathbf{W}^M; \mathcal{Q}) \leq \hat{L}_C(\mathbf{W}^M; \mathcal{Q}) + \frac{1}{v_u}(1 + \frac{h}{\kappa^2} o(\frac{1}{b}) + ln\frac{1}{v_G} + \frac{v_u^2}{4n_C} + v_u\Delta(\beta, b)),$$

and,

$$\begin{aligned} \Delta_{C,\mathcal{Z}}(\beta_1, b) &\geq \Delta_{C,\mathcal{Z}}(\beta_2, b), \qquad \beta_1 \leq \beta_2, \\ \Delta_{C,\mathcal{Z}}(\beta, b_1) &\geq \Delta_{C,\mathcal{Z}}(\beta, b_2), \qquad b_1 \leq b_2. \end{aligned}$$

Proof. Using Theorem 3., we have

$$L^M_{\mathcal{Z}}(\mathbf{W}^M; \mathcal{Q}) \leq \hat{L}_C(\mathbf{W}^M; \mathcal{Q}) + \frac{1}{v_u}(D_{KL}(\mathcal{Q}\|\mathcal{P}) + ln\frac{1}{v_G} + \frac{v_u^2}{4n_C} + U(v_u). \tag{72}$$

Since both $\mathcal{P}$ and $\mathcal{Q}$ are normal distributions [42], we know that

$$D_{KL}(\mathcal{Q}\|\mathcal{P}) \leq \frac{\|\mathbf{W}^M_T\|_F^2}{2\kappa^2}, \tag{73}$$

where there exist T iterations in the training process.

As the model parameters are updated by $\mathbf{W}^M_T = \mathbf{W}^M_{T-1} - \eta_{T-1}\nabla L^M_{C,T-1}(\mathbf{W}^M_{T-1})$, we have $\mathbf{W}^M_T = \mathbf{W}^M_0 - \sum_{t=0}^{T-1} \eta_t \nabla L^M_{C,t-1}(\mathbf{W}^M_{t-1})$. Then,

$$\begin{aligned} D_{KL}(\mathcal{Q}\|\mathcal{P}) \leq& \frac{\|\mathbf{W}^M_T\|_F^2}{2\kappa^2} \\ =& \frac{\|\mathbf{W}^M_0 - \sum_{t=0}^{T-1} \eta_t \nabla L^M_{C,t-1}(\mathbf{W}^M_{t-1})\|_F^2}{2\kappa^2} \\ \leq& \frac{\|\mathbf{W}^M_0\|_F^2 + \sum_{t=0}^{T-1} \eta_t^2 \|\nabla L^M_{C,t-1}(\mathbf{W}^M_{t-1})\|_F^2}{2\kappa^2} \\ \leq& 1 + \frac{\sum_{t=0}^{T-1} \eta_t^2 Tr((\nabla L^M_{C,t-1}(\mathbf{W}^M_{t-1}))^\top \nabla L^M_{C,t-1}(\mathbf{W}^M_{t-1}))}{2\kappa^2} \end{aligned} \tag{74}$$

For $\nabla L^M_{C,t-1,j}(\mathbf{w}^M_{t-1,j})$ with $j = \{1, \ldots, h\}$, we have

$$\nabla L^M_{C,t-1,j}(\mathbf{w}^M_{t-1,j}) = \begin{bmatrix} \frac{\partial^2 L^M_C(\phi^M, \psi^M)}{\partial \phi^{M^2}} & \frac{\partial^2 L^M_C(\phi^M, \psi^M)}{\partial \phi^M \partial \psi^M} \\ \frac{\partial^2 L^M_C(\phi^M, \psi^M)}{\partial \psi^M \partial \phi^M} & \frac{\partial^2 L^M_C(\phi^M, \psi^M)}{\partial \psi^{M^2}} \end{bmatrix} \tag{75}$$

Then, using the intermediate results in Lemma 3., we have

$$\begin{aligned}
&Tr((\nabla L^M_{C,t-1,j}(\mathbf{w}^M_{t-1,j}))^\top \nabla L^M_{C,t-1,j}(\mathbf{w}^M_{t-1,j})) \\
=&(\frac{\partial^2 L^M_C(\phi^M,\psi^M)}{\partial {\phi^M}^2})^2 + 2(\frac{\partial^2 L^M_C(\phi^M,\psi^M)}{\partial\phi^M \partial\psi^M})^2 + (\frac{\partial^2 L^M_C(\phi^M,\psi^M)}{\partial {\psi^M}^2})^2 \\
\le& 2(\Gamma^M)^2(1+\sqrt{2\kappa\sqrt{r}+4\pi+1}+o(\frac{1}{b}))^2 + 2(\Gamma^M)^2(\sqrt{2\kappa\sqrt{r}+4\pi+1}+o(\frac{1}{b}))^2 \\
=&(\Gamma^M)^2 o(\frac{1}{b^2}) \\
\le& o(\frac{1}{b^2}).
\end{aligned} \tag{76}$$

Then we have $Tr((\nabla L^M_{C,t-1}(\mathbf{W}^M_{t-1}))^\top \nabla L^M_{C,t-1}(\mathbf{W}^M_{t-1})) = \sum_{j=1}^h Tr((\nabla L^M_{C,t-1,j}(\mathbf{w}^M_{t-1,j}))^\top \nabla L^M_{C,t-1,j}(\mathbf{w}^M_{t-1,j})) \le 2h\zeta(b)$. Using the range of η_t in Theorem 2. and T is extremely large, we have

$$\begin{aligned}
D_{KL}(Q\|\mathcal{P}) \le& 1 + \frac{\sum_{t=0}^{T-1}\eta_t Tr((\nabla L^M_{C,t-1}(\mathbf{W}^M_{t-1}))^\top \nabla L^M_{C,t-1}(\mathbf{W}^M_{t-1}))}{2\kappa^2} \\
\le& 1 + \frac{h}{\kappa^2} o(\frac{1}{b})
\end{aligned} \tag{77}$$

Hence,

$$\begin{aligned}
L^M_{\mathcal{Z}}(\mathbf{W}^M;Q) \le& \hat{L}_C(\mathbf{W}^M;Q) + \frac{1}{v_u}(D_{KL}(Q\|\mathcal{P}) + ln\frac{1}{v_G} + \frac{v_u^2}{4n_C} + U(v_u) \\
\le& \hat{L}_C(\mathbf{W}^M;Q) + \frac{1}{v_u}(1 + \frac{h}{\kappa^2}o(\frac{1}{b}) + ln\frac{1}{v_G} + \frac{v_u^2}{4n_C} + v_u\Delta(\beta,b)).
\end{aligned} \tag{78}$$

C EXPERIMENTS

Table 3: Datasets.

Datasets	#Nodes	#Edges	Avg. Degree	#Classes	#Features	Train/Val/Test
Reddit	232,965	11,606,919	50	41	602	152,410/23,699/55,334
Ogbn-arxiv	169,343	1,166,243	7	40	128	90,941/29,799/48,603
Ogbn-products	2,449,029	61,859,140	25	47	100	195,922/48,980/2,204,127
Ogbn-papers100M	111,059,956	1,615,685,872	15	172	128	1,207,179 / 125,265/214,338

C.1 Training settings

We run all implementations using PyTorch 3.8.10 and dgl>=1.0.0. The GNN model parameters are initialized with the same seed. The uniform neighbor sampling is used for mini-batch training.

C.2 Metrics: Iteration-to-loss

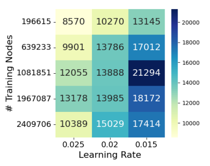
We examine a three-layer GraphSAGE model on Reddit and a three-layer GCN model on ogbn-products. These models include normalization layers and are trained using a cross-entropy loss function and Adam optimizer with a learning rate of 0.01. The target validation accuracy is set at 0.9 for ogbn-products and 0.95 for Reddit. The total batch size is 2000 and the fan-out size is [5,10,15].

C.3 Full-graph vs. Mini-batch Training.

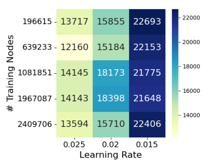
For the comparison at the dimension of batch size and fan-out size, we use 3-layer GraphSAGE models with hidden dimension of 256 for reddit, ogbn-products, and ogbn-arxiv, and 2-layer GraphSAGE models with hidden dimension of 128 for ogbn-papers100M. The activation function is ReLU function. The optimizer is Adam with a learning rate of 0.001 and a weight decay of 0. Due to the extremely large graph size of the ogbn-papers100M dataset and limited computational resources, we use separate machines for full-graph and mini-batch training on this dataset, making it infeasible to compare system efficiency between the two methods.

C.4 Cluster Size.

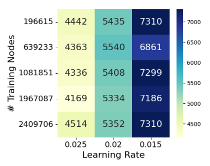
We set the target accuracy as 0.5 for one-layer GraphSAGE on ogbn-products in mini-batch training with MSE. We set the fan-out size as 5 and the learning rate as 0.025. We use DDP [34] is used on four A100 GPUs with each GPU handling a batch size of 1,000 nodes.



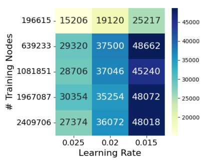
(a) GAT, mini-batch



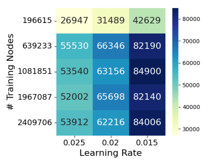
(b) SAGE, mini-batch



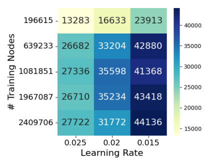
(c) GCN, mini-batch



(d) GAT, full-graph

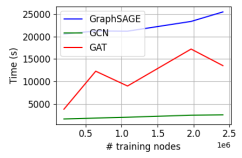


(e) SAGE, full-graph

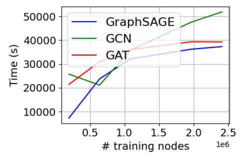


(f) GCN, full-graph

Figure 15: Iteration-to-loss for ogbn-products datasets for GAT, GCN, GraphSAGE across different number of training nodes in mini-batch and full-graph training.



(a) Mini-batch



(b) Full-graph

Figure 16: Time-to-acc for GAT, GCN, GraphSAGE on ogbn-products in full-graph and mini-bacth training across different number of training nodes.

C.5 Graph-based Hyperparameters in Mini-batch Training

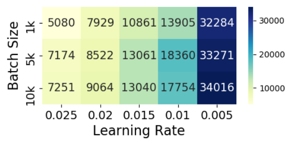
The SAR system [46] is used for full-graph and mini-batch training on ogbn-papers100M via the *gloo* backend, while other datasets are mainly trained on a single GPU. The basic setup includes 1-layer GNN models without drop-out or normalization layers and with ReLU activation, MSE loss function, and SGD optimizer with a learning rate of 0.025 for both full-graph and mini-batch training.

To align with theoretical analysis, we use iteration-to-loss and time-to-accuracy for convergence, and accuracy for generalization. We use different settings for the experiments measuring three metrics:

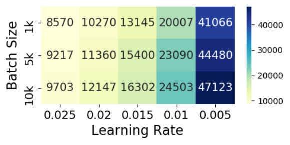
1). For iteration-to-loss, the *target training losses* are 0.09 for two synthetic graphs, 0.0226 for ogbn-arxiv, 0.0225 for reddit and ogbn-products, and [0.005, 0.0054, 0.0065] for ogbn-papers100M on GCN, GraphSAGE, GAT, respectively. The *learning rates* are [0.005, 0.01, 0.015, 0.02, 0.025] reddit, ogbn-arxiv, and ogbn-products, [0.00025, 0.0002] for ogbn-papers100M, and [0.01, 0.02, 0.03] for synthetic graphs. The *batch sizes* with a fan-out size of 5 are [500, 1000, 5000] for ogbn-arxiv and synthetic graphs, [1000, 5000, 10000] for ogbn-products and reddit, as well as [10000, 15000, 100000] for ogbn-papers100M. The *fan-out sizes* with a batch-size of 500 are [1,2,5] for ogbn-arxiv , and [5,10,15] for ogbn-products on all models and two synthetic graphs on GAT model, reddit and two synthetic graphs. The fan-out size with a batch-size of 5000 are [5,10,15] for two synthetic graphs on GraphSAGE and GCN models. The fan-out size with a batch size of 10000 are [5,10,50] for ogbn-papers100M. Full-graph are used for all datasets.

2). For time-to-accuracy, the *target validation accuracies* are [0.75, 0.75, 0.6] for ogbn-products, [0.6, 0.6, 0.5] for reddit, [0.29, 0.29,0.29] for ogbn-arxiv, [0.1179, 0.1284, 0.1739] for ogbn-papers100M on GCN, GraphSAGE, GAT, respectively. The *batch sizes* with a fan-out size of 5 are [500, 1000,5000,10000] for ogbn-arxiv, ogbn-products, and reddits, and [10000, 15000, 100000] for ogbn-papers100M. The *fan-out sizes* with a batch size of 5 are [1,2,5,10,50] for ogbn-arxiv, [5,10,15,30,50] for ogbn-products and reddit, and [5,10,50] for ogbn-papers100M. Full-graph are used for all real-world datasets except ogbn-papers100M.

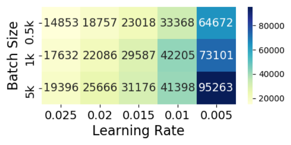
3). For test accuracy, *the numbder of iterations* are 5×10^5 for GraphSAGE and GCN, or 1×10^5 for GAT, for ogbn-arxiv, ogbn-products, and reddit. And the number of iterations are 1×10^4 for ogbn-papers100M across all GNN models. The *learning rates* are [0.015,0.02,0.025] for ogbn-arxiv, ogbn-products, and reddit, and [0.00025, 0.0002] for ogbn-papers100M. The batch sizes and the fan-out sizes are consistent with the settings used in the experiments measuring time-to-accuracy.



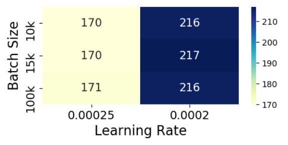
(a) Reddit, GAT



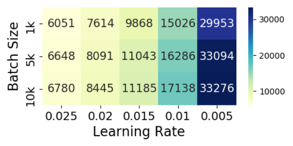
(b) Products, GAT



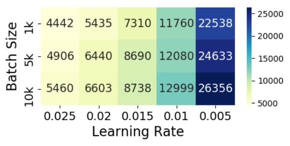
(c) Arxiv, GAT



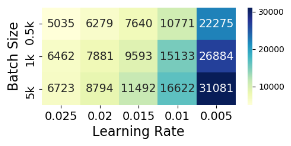
(d) Papers100M, GAT



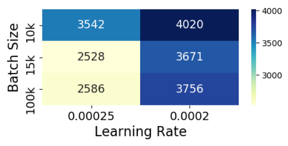
(e) Reddit, GCN



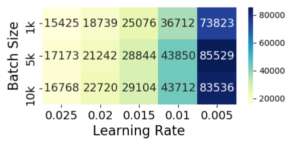
(f) Products, GCN



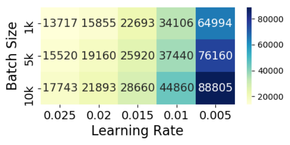
(g) Arxiv, GCN



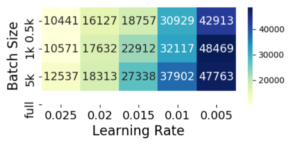
(h) Papers100M, GCN



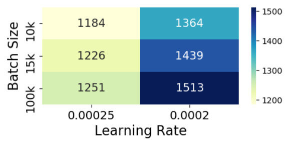
(i) Reddit, SAGE



(j) Products, SAGE

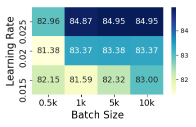


(k) Arxiv, SAGE

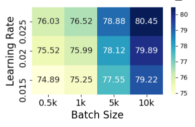


(l) Papers100M, SAGE

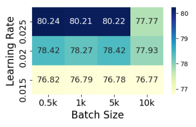
Figure 17: Iteration-to-loss for real-world datasets for GAT, GCN, GraphSAGE across different batch sizes and learning rates.



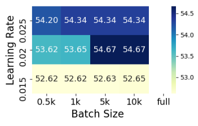
(a) Reddit, SAGE



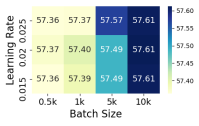
(b) Reddit, GCN



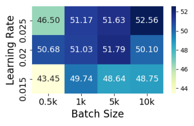
(c) Reddit, GAT



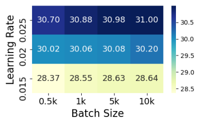
(d) Products, SAGE



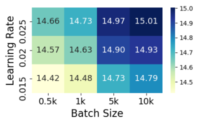
(e) Products, GCN



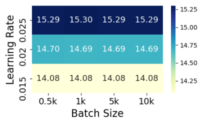
(f) Products, GAT



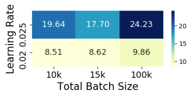
(g) Arxiv, SAGE



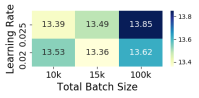
(h) Arxiv, GCN



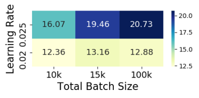
(i) Arxiv, GAT



(j) Papers100M, SAGE

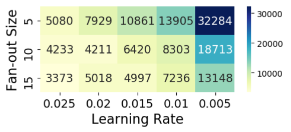


(k) Papers100M, GCN

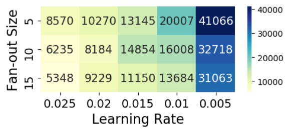


(l) Papers100M, GAT

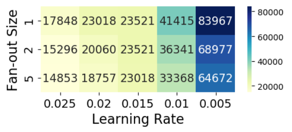
Figure 18: Test accuracy for real-world datasets for GAT, GCN, GraphSAGE across different batch size and learning rate.



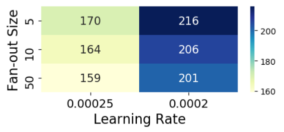
(a) Reddit, GAT



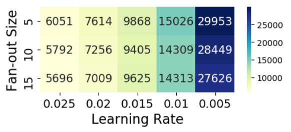
(b) Products, GAT



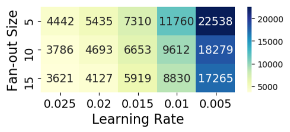
(c) Arxiv, GAT



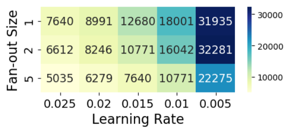
(d) Papers100M, GAT



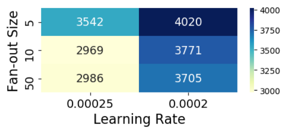
(e) Reddit, GCN



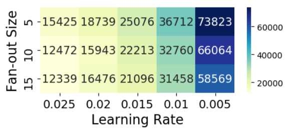
(f) Products, GCN



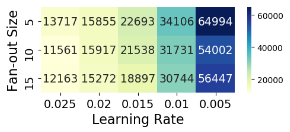
(g) Arxiv, GCN



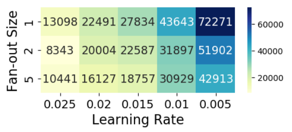
(h) Papers100M, GCN



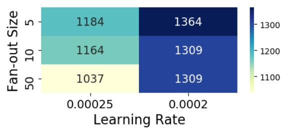
(i) Reddit, SAGE



(j) Products, SAGE

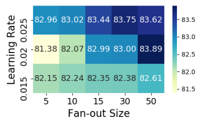


(k) Arxiv, SAGE

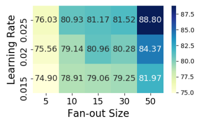


(l) Papers100M, SAGE

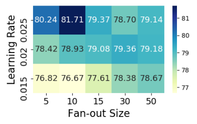
Figure 19: Iteration-to-loss for real-world datasets for GAT, GCN, GraphSAGE across different fan-out sizes and learning rates.



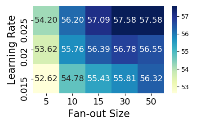
(a) Reddit, SAGE



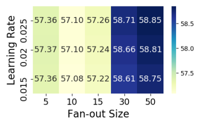
(b) Reddit, GCN



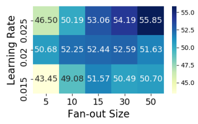
(c) Reddit, GAT



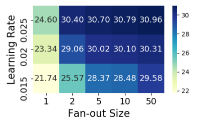
(d) Products, SAGE



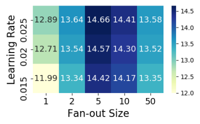
(e) Products, GCN



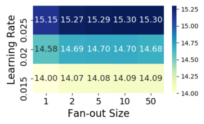
(f) Products, GAT



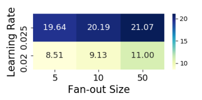
(g) Arxiv, SAGE



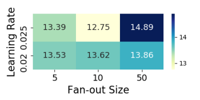
(h) Arxiv, GCN



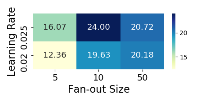
(i) Arxiv, GAT



(j) Papers100M, SAGE



(k) Papers100M, GCN



(l) Papers100M, GAT

Figure 20: Test accuracy for real-world datasets for GAT, GCN, GraphSAGE across different fan-out sizes and learning rates.