GIST: Distributed Training for Large-Scale Graph Convolutional Networks

Cameron R. Wolfe[⋆], **Jingkang Yang**[⋆], **Fangshuo Liao**[⋆],
Arindam Chowdhury, Chen Dun, Artun Bayer, Santiago Segarra, Anastasios Kyrillidis
(·[⋆] equal contribution)
Rice University

Abstract

The graph convolutional network (GCN) is a go-to solution for machine learning on graphs, but its training is notoriously difficult to scale both in terms of graph size and the number of model parameters. Although some work has explored training on large-scale graphs, we pioneer efficient training of large-scale GCN models with the proposal of a novel, distributed training framework, called GIST. GIST disjointly partitions the parameters of a GCN model into several, smaller sub-GCNs that are trained independently and in parallel. Compatible with all GCN architectures and existing sampling techniques, GIST $i)$ improves model performance, $ii)$ scales to training on arbitrarily large graphs, $iii)$ decreases wall-clock training time, and $iv)$ enables the training of markedly overparameterized GCN models. Remarkably, with GIST, we train an astonishgly-wide 32,768-dimensional GraphSAGE model, which exceeds the capacity of a single GPU by a factor of $8\times$, to SOTA performance on the Amazon2M dataset.

1 Introduction

Since not all data can be represented in Euclidean space [5], many applications rely on graph-structured data. For example, social networks can be modeled as graphs by regarding each user as a node and friendship relations as edges [34, 37]. Alternatively, in chemistry, molecules can be modeled as graphs, with nodes representing atoms and edges encoding chemical bonds [2, 4].

To better understand graph-structured data, several (deep) learning techniques have been extended to the graph domain [12, 17, 35]. Currently, the most popular one is the graph convolutional network (GCN) [28], a multi-layer architecture that implements a generalization of the convolution operation to graphs. Although the GCN handles node- and graph-level classification, it is notoriously inefficient and unable to support large graphs [8, 9, 14, 23, 51, 54], making practical, large-scale applications difficult to handle.

To deal with these issues, node partitioning methodologies have been developed. These schemes can be roughly categorized into neighborhood sampling [9, 19, 59] and graph partitioning [10, 54] approaches. The goal is to partition a large graph into multiple smaller graphs that can be used as mini-batches for training the GCN. In this way, GCNs can handle larger graphs during training, expanding their potential into the realm of big data. However, the size of the underlying model is still limited by available memory capacity, thus placing further constraints on the scale of GCN experimentation.

Although some papers perform large-scale experiments [10, 54], the models (and data) used in GCN research remain small in the context of deep learning [28, 50], where the current trend is towards incredibly large models and datasets [7, 11]. Despite the widespread moral questioning of this trend [20, 41, 44], the deep learning community continues to push the limits of scale. Overparameterized models yield improvements in tasks like zero/few-shot learning [6, 42], are capable of discovering generalizable solutions [36], and even have desirable theoretical properties [38].

Although GCN models suffer from performance deterioration due to oversmoothing at larger depths [28, 30], one would expect GCNs to similarly benefit from overparameterization, meaning that the use of larger hidden layers may be beneficial. Furthermore, recent work indicates that the impact of overparameterization becomes increasingly pronounced on larger datasets [21], making the exploration of overparameterized models essential as GCNs continue to be applied to practical problems at scale. Moving in this direction, *our work provides an efficient training framework for wide, overparameterized GCN models—beyond the memory capacity of a single GPU—on large-scale datasets.*

This paper. We propose a novel, distributed training methodology that can be used for any GCN architecture and *is compatible with existing node sampling techniques.* This methodology randomly partitions the hidden feature space in each layer, decomposing the global GCN model into multiple, narrow sub-GCNs of equal depth. Sub-GCNs are trained independently for several iterations in parallel prior to having their updates synchronized; see Figure 1. This process of randomly partitioning, independently training, and synchronizing sub-GCNs is repeated until convergence. We call this method **G**raph **I**ndependent **S**ubnetwork **T**raining (`GIST`). `GIST` can scale to arbitrarily large graphs, and significantly reduces the wall-clock time of training large-scale GCNs, allowing larger models and datasets to be explored. We focus specifically on enabling the training of "ultra-wide" GCNs (*i.e.*, GCN models with very large hidden layers), as deeper GCNs are prone to oversmoothing [30]. The contributions of this work are summarized below:

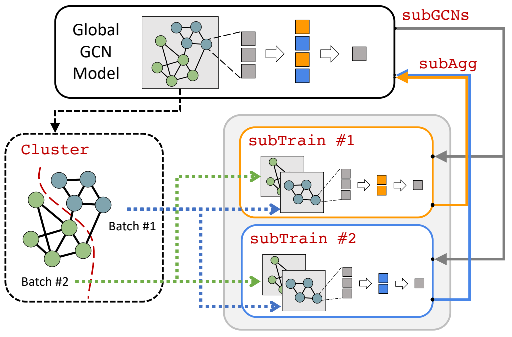

Figure 1: `GIST` pipeline: `subGCNs` divides the global GCN into sub-GCNs. Every sub-GCN is trained by `subTrain` using mini-batches (smaller sub-graphs) generated by `Cluster`. Sub-GCN parameters are intermittently aggregated through `subAgg`.

- We develop a novel, distributed training methodology for arbitrary GCN architectures, based on decomposing the model into independently-trained sub-GCNs. This methodology is compatible with existing techniques for neighborhood sampling and graph partitioning.
- We show that `GIST` can be used to train several GCN architectures to state-of-the-art performance with reduced training time, in comparison to standard methodologies on large-scale datasets.
- We propose a novel Graph Independent Subnetwork Training Kernel (`GIST-K`) that allows a convergence rate to be derived for two-layer GCNs trained with `GIST` in the infinite width regime. Based on `GIST-K`, we provide theory that `GIST` converges linearly –up to an error neighborhood– using distributed gradient descent with local iterations. We show that the radius of the error neighborhood is controlled by the overparameterization parameter, as well as the number of workers in the distributed setting. Such findings reflect practical observations that are made in the experimental section.
- We use `GIST` to enable the training of markedly overparameterized GCN models. In particular, `GIST` is used to train a two-layer GraphSAGE model with a hidden dimension of 32 768 on the Amazon2M dataset. *Such a model exceeds the capacity of a single GPU by* $8\times$.

2 What is the `GIST` of this work?

GCN Architecture. The GCN [28] is arguably the most widely-used neural network architecture on graphs. Consider a graph $\mathcal{G}$ comprised of n nodes with d-dimensional features $\mathbf{X} \in \mathbb{R}^{n\times d}$. The output $\mathbf{Y} \in \mathbb{R}^{n\times d'}$ of a GCN can be expressed as $\mathbf{Y} = \Psi_{\mathcal{G}}(\mathbf{X}; \mathbf{\Theta})$, where $\Psi_{\mathcal{G}}$ is an L-layered architecture with trainable parameters $\mathbf{\Theta}$. If we define $\mathbf{H}_0 = \mathbf{X}$, we then have that $\mathbf{Y} = \Psi_{\mathcal{G}}(\mathbf{X}; \mathbf{\Theta}) = \mathbf{H}_L$, where an intermediate ℓ-th layer of the GCN is given by

$$\mathbf{H}_{\ell+1} = \sigma(\bar{\mathbf{A}}\, \mathbf{H}_\ell\, \mathbf{\Theta}_\ell). \tag{1}$$

In (1), $\sigma(\cdot)$ is an elementwise activation function (*e.g.*, ReLU), $\bar{\mathbf{A}}$ is the degree-normalized adjacency matrix of $\mathcal{G}$ with added self-loops, and the trainable parameters $\mathbf{\Theta} = \{\mathbf{\Theta}_\ell\}_{\ell=0}^{L-1}$ have dimensions

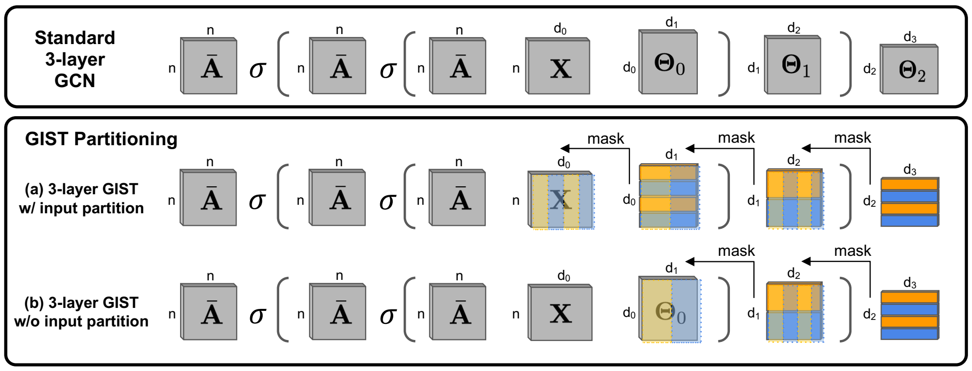

Figure 2: GCN partition into $m = 2$ sub-GCNs. Orange and blue colors depict different feature partitions. Both hidden dimensions (d_1 and d_2) are partitioned. The output dimension (d_3) is not partitioned. Partitioning the input dimension (d_0) is optional, but we do not partition $\mathbf{d}_0$ in GIST.

$\Theta_\ell \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ with $d_0 = d$ and $d_L = d'$. In Figure 2 (top), we illustrate nested GCN layers for $L = 3$, but our methodology extends to arbitrary L. The activation function of the last layer is typically the identity or softmax transformation – we omit this in Figure 2 for simplicity.

Algorithm 1 GIST Algorithm

```
1: **Parameters**: $T$ synchronization iterations, $m$ sub-GCNs,
2: $\zeta$ local iterations, $c$ clusters, $\mathcal{G}$ training graph.
3:
4: $\Psi_{\mathcal{G}}(\,\cdot\,;\Theta) \leftarrow$ randomly initialize GCN
5: $\{\mathcal{G}_{(j)}\}_{j=1}^{c} \leftarrow$ Cluster$(\mathcal{G}, c)$
6: for $t = 0, \ldots, T-1$ do
7:     $\{\Psi_{\mathcal{G}}(\,\cdot\,;\Theta^{(i)})\}_{i=1}^{m} \leftarrow$ subGCNs$(\Psi_{\mathcal{G}}(\,\cdot\,;\Theta), m)$
8:     Distribute each $\Psi_{\mathcal{G}}(\,\cdot\,;\Theta^{(i)})$ to a different worker
9:     for $i = 1, \ldots, m$ do
10:        for $z = 1, \ldots, \zeta$ do
11:            $\Psi_{\mathcal{G}}(\,\cdot\,;\Theta^{(i)}) \leftarrow$ subTrain$(\Theta^{(i)}, \{\mathcal{G}_{(j)}\}_{j=1}^{c})$
12:        end for
13:    end for
14:    $\Psi_{\mathcal{G}}(\,\cdot\,;\Theta) \leftarrow$ subAgg$(\{\Theta^{(i)}\}_{i=1}^{m})$
15: end for
```

GIST **overview.** We overview GIST in Algorithm 1 and present a schematic depiction in Figure 1. We partition our (randomly initialized) global GCN into m smaller, disjoint sub-GCNs with the subGCNs function ($m = 2$ in Figures 1 and 2) by sampling the feature space at each layer of the GCN; see Section 2.1. Each sub-GCN is assigned to a different worker (*i.e.*, a different GPU) for ζ rounds of distributed, independent training through subTrain. Then, newly-learned sub-GCN parameters are aggregated (subAgg) into the global GCN model. This process repeats for T iterations. Our graph domain is partitioned into c sub-graphs through the Cluster function ($c = 2$ in Figure 1). This operation is only relevant for large graphs ($n >$ 50,000), and we omit it ($c = 1$) for smaller graphs that don't require partitioning.[1]

2.1 subGCNs: Constructing Sub-GCNs

GIST partitions a global GCN model into several narrower sub-GCNs of equal depth. Formally, consider an arbitrary layer ℓ and a random, disjoint partition of the feature set $[d_\ell] = \{1, 2, \ldots, d_\ell\}$ into m equally-sized blocks $\{\mathcal{D}_\ell^{(i)}\}_{i=1}^{m}$.[2] Accordingly, we denote by $\Theta_\ell^{(i)} = [\Theta_\ell]_{\mathcal{D}_\ell^{(i)} \times \mathcal{D}_{\ell+1}^{(i)}}$ the matrix obtained by selecting from Θ_ℓ the rows and columns given by the ith blocks in the partitions of $[d_\ell]$ and $[d_{\ell+1}]$, respectively. With this notation in place, we can define m different sub-GCNs $\mathbf{Y}^{(i)} = \Psi_{\mathcal{G}}(\mathbf{X}^{(i)}; \Theta^{(i)}) = \mathbf{H}_L^{(i)}$ where $\mathbf{H}_0^{(i)} = \mathbf{X}_{[n]\times\mathcal{D}_0^{(i)}}$ and each layer is given by:

$$\mathbf{H}_{\ell+1}^{(i)} = \sigma(\bar{\mathbf{A}}\,\mathbf{H}_\ell^{(i)}\,\Theta_\ell^{(i)}). \tag{2}$$

Notably, not all parameters within the global GCN model are partitioned to a sub-GCN. However, by randomly re-constructing new groups of sub-GCNs throughout the training process, all parameters

[1]Though any clustering method can be used, we advocate the use of METIS [25, 26] due to its proven efficiency in large-scale graphs.

[2]For example, if $d_\ell = 4$ and $m = 2$, one valid partition would be given by $\mathcal{D}_\ell^{(1)} = \{1, 4\}$ and $\mathcal{D}_\ell^{(2)} = \{2, 3\}$.

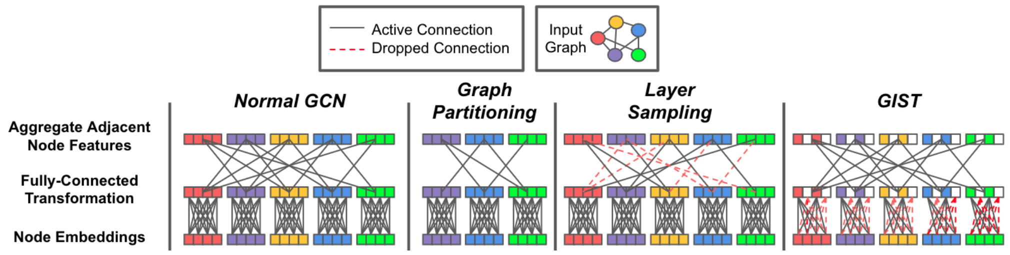

Figure 3: Illustrates the difference between `GIST` and node sampling techniques within the forward pass of a single GCN layer (excluding non-linear activation). While graph partitioning and layer sampling remove nodes from the forward pass (*i.e.*, either completely or on a per-layer basis), `GIST` partitions node feature representations (and, in turn, model parameters) instead of the nodes.

have a high likelihood of being updated. In section 4, we provide theoretical guarantees that the partitioning of model parameters to sub-GCNs does not harm training performance.

Sub-GCN partitioning is illustrated in Figure 2-(a), where $m = 2$. Partitioning the input features is optional (*i.e.*, (a) vs. (b) in Figure 2). *We do not partition the input features within `GIST`* so that sub-GCNs have identical input information (*i.e.*, $\mathbf{X}^{(i)} = \mathbf{X}$ for all i); see Section 5.1. Similarly, we do not partition the output feature space to ensure that the sub-GCN output dimension coincides with that of the global model, thus avoiding any need to modify the loss function. This decomposition procedure (`subGCNs` function in Algorithm 1) extends to arbitrary L.

2.2 `subTrain`: Independently Training Sub-GCNs

Assume $c = 1$ so that the `Cluster` operation in Algorithm 1 is moot and $\{\mathcal{G}_{(j)}\}_{j=1}^{c} = \mathcal{G}$. Because $\mathbf{Y}^{(i)}$ and $\mathbf{Y}$ share the same dimension, sub-GCNs can be trained to minimize the same global loss function. One application of `subTrain` in Algorithm 1 corresponds to a single step of stochastic gradient descent (SGD). Inspired by local SGD [33], multiple, independent applications of `subTrain` are performed in parallel (*i.e.*, on separate GPUs) for each sub-GCN prior to aggregating weight updates. The number of independent training iterations between synchronization rounds, referred to as *local iterations*, is denoted by ζ, and the total amount of training is split across sub-GCNs.[3] Ideally, the number sub-GCNs and local iterations should be increased as much as possible to minimize communication and training costs. In practice, however, such benefits may come at the cost of statistical inefficiency; see Section 5.1.

If $c > 1$, `subTrain` first selects one of the c subgraphs in $\{\mathcal{G}_{(j)}\}_{j=1}^{c}$ to use as a mini-batch for SGD. Alternatively, the union of several sub-graphs in $\{\mathcal{G}_{(j)}\}_{j=1}^{c}$ can be used as a mini-batch for training. Aside from using mini-batches for each SGD update instead of the full graph, *the use of graph partitioning does not modify the training approach outlined above.* Some form of node sampling must be adopted to make training tractable when the full graph is too large to fit into memory. However, *both graph partitioning and layer sampling are compatible with* `GIST` (see Sections 5.2 and **??**). We adopt graph sampling in the main experiments due to the ease of implementation. The novelty of our work lies in the feature partitioning strategy of `GIST` for distributed training, which is an orthogonal technique to node sampling; see Figure 3 and Section 2.4.

2.3 `subAgg`: Aggregating Sub-GCN Parameters

After each sub-GCN completes ζ training iterations, their updates are aggregated into the global model (`subAgg` function in Algorithm 1). Within `subAgg`, each worker replaces global parameter entries within $\mathbf{\Theta}$ with its own, independently-trained sub-GCN parameters $\mathbf{\Theta}^{(i)}$, where no collisions occur due to the disjointness of sub-GCN partitions. Thus, `subAgg` is a basic copy operation that transfers sub-GCN parameters into the global model.

Not every parameter in the global GCN model is updated by `subAgg` because, as previously mentioned, parameters exist that are not partitioned to any sub-GCN by the `subGCNs` operation. For example, focusing on $\mathbf{\Theta}_1$ in Figure 2-(a), one worker will be assigned $\mathbf{\Theta}_1^{(1)}$ (*i.e.*, overlapping orange blocks), while the other worker will be assigned $\mathbf{\Theta}_1^{(2)}$ (*i.e.*, overlapping blue blocks). The rest of $\mathbf{\Theta}_1$ is not

[3]For example, if a global model is trained on a single GPU for 10 epochs, a comparable experiment for `GIST` with two sub-GCNs would train each sub-GCN for only 5 epochs.

considered within subAgg. Nonetheless, since sub-GCN partitions are randomly drawn in each cycle t, one expects all of $\mathbf{\Theta}$ to be updated multiple times if T is sufficiently large.

2.4 What is the value of GIST?

Architecture-Agnostic Distributed Training. GIST is a generic, distributed training methodology that can be used for any GCN architecture. We implement GIST for vanilla GCN, GraphSAGE, and GAT architectures, but GIST is not limited to these models; see Section 5.

Compatibility with Sampling Methods. GIST is **NOT** a replacement for graph or layer sampling. Rather, it is an efficient, distributed training technique that can be used in tandem with node partitioning. As depicted in Figure 3, GIST partitions node feature representations and model parameters between sub-GCNs, while graph partitioning and layer sampling sub-sample nodes within the graph.

Interestingly, we find that GIST's feature and parameter partitioning strategy is compatible with node partitioning—the two approaches can be combined to yield further efficiency benefits. For example, GIST is combined with graph partitioning strategies in Section 5.2 and with layer sampling methodologies in Section **??**.

Enabling Ultra-Wide GCN Training. GIST indirectly updates the global GCN through the training of smaller sub-GCNs, enabling models with hidden dimensions that exceed the capacity of a single GPU to be trained; in our experiments, we show results where GIST allows training of models beyond the capacity of a single GPU by a factor of $8\times$. In this way, GIST allows markedly overparametrized ("ultra-wide") GCN models to be trained on existing hardware. In Section 5.2, *we leverage this capability to train a two-layer GCN model with a hidden dimension of 32,768 on Amazon2M.*

Overparameterization through width could be valuable because deeper GCNs could suffer from oversmoothing [30]. Additionally, the theoretical results provided within Section 4 reveal that the performance of GIST is best as the number of neurons within each hidden layer is increased, which further reveals the benefit of wide, overparameterized layers. We do not explore depth-wise partitions of different GCN layers to each worker, but rather focus solely upon partitioning the hidden neurons within each layer—a strategy that is suited to training wider networks. We consider such a direction as future work.

Improved Model Complexity. Consider a single GCN layer, trained over M machines with input and output dimension of d_{i-1} and d_i, respectively. For one synchronization round, the communication complexity of GIST and standard distributed training is $\mathcal{O}(\frac{1}{M}d_i d_{i-1})$ and $\mathcal{O}(M d_i d_{i-1})$, respectively. GIST reduces communication by only communicating sub-GCN parameters. Existing node partitioning techniques cannot similarly reduce communication complexity because model parameters are never partitioned. Furthermore, the computational complexity of the forward pass for a GCN model trained with GIST and using standard methodology is $\mathcal{O}(\frac{1}{M}N^2 d_i + \frac{1}{M^2}N d_i d_{i-1})$ and $\mathcal{O}(N^2 d_i + N d_i d_{i-1})$, respectively, where N is the number of nodes in the partition being processed.[4] Node partitioning can reduce N by a constant factor but is compatible with GIST.

3 Related Work

GCN training. In spite of their widespread success in several graph related tasks, GCNs often suffer from training inefficiencies [14, 23]. Consequently, the research community has focused on developing efficient and scalable algorithms for training GCNs [8, 9, 10, 19, 54, 59]. The resulting approaches can be divided roughly into two areas: *neighborhood sampling* and *graph partitioning*. However, it is important to note that these two broad classes of solutions are not mutually exclusive, and reasonable combinations of the two approaches may be beneficial.

Neighborhood sampling methodologies aim to sub-select neighboring nodes at each layer of the GCN, thus limiting the number of node representations in the forward pass and mitigating the exponential expansion of the GCNs receptive field. VRGCN [8] implements a variance reduction technique to reduce the sample size in each layer, which achieves good performance with smaller graphs. However, it requires to store all the intermediate node embeddings during training, leading to a memory complexity close to full-batch training. GraphSAGE [19] learns a set of aggregator functions to gather information from a node's local neighborhood. It then concatenates the outputs of these aggregation functions with each node's own representation at each step of the forward pass. FastGCN [9] adopts a Monte Carlo approach to evaluate the GCN's forward pass in practice, which computes each node's

[4]We omit the complexity of applying the element-wise activation function for simplicity.

hidden representation using a fixed-size, randomly-sampled set of nodes. LADIES [59] introduces a layer-conditional approach for node sampling, which encourages node connectivity between layers in contrast to FastGCN [9].

Graph partitioning schemes aim to select densely-connected sub-graphs within the training graph, which can be used to form mini-batches during GCN training. Such sub-graph sampling reduces the memory footprint of GCN training, thus allowing larger models to be trained over graphs with many nodes. ClusterGCN [10] produces a very large number of clusters from the global graph, then randomly samples a subset of these clusters and computes their union to form each sub-graph or mini-batch. Similarly, GraphSAINT [54] randomly samples a sub-graph during each GCN forward pass. However, GraphSAINT also considers the bias created by unequal node sampling probabilities during sub-graph construction, and proposes normalization techniques to eliminate this bias.

As explained in Section 2, `GIST` also relies on graph partitioning techniques (`Cluster`) to handle large graphs. However, the feature sampling scheme at each layer (`subGCNs`) that leads to parallel and narrower sub-GCNs is a hitherto unexplored framework for efficient GCN training.

Distributed training. Distributed training is a heavily studied topic [45, 56]. Our work focuses on synchronous and distributed training techniques [31, 52, 55]. Some examples of synchronous, distributed training approaches include data parallel training, parallel SGD [1, 58], and local SGD [33, 48]. Our methodology holds similarities to model parallel training techniques, which have been heavily explored [3, 15, 18, 29, 40, 49, 57]. More closely, our approach is inspired by independent subnetwork training [53], explored for multi-layer perceptrons.

4 Theoretical Results

We draw upon analysis related to neural tangent kernels (NTK) [24] to derive a convergence rate for two-layer GCNs using gradient descent—as formulated in (1) and further outlined in Appendix C.1—trained with `GIST`. Given the scaled Gram matrix of an infinite-dimensional NTK $\mathbf{H}^{\infty}$, we define the Graph Independent Subnetwork Training Kernel (`GIST-K`) as follows:

$$\mathbf{G}^{\infty} = \bar{\mathbf{A}}\mathbf{H}^{\infty}\bar{\mathbf{A}}.$$

Given the `GIST-K`, we adopt the following set of assumptions related to the underlying graph; see Appendix C.2 for more details.

Assumption 1 *Assume $\lambda_{\min}(\bar{\mathbf{A}}) \neq 0$ and there exists $\epsilon \in (0,1)$ and $p \in \mathbb{Z}_+$ such that $(1-\epsilon)^2 p \leq \mathbf{D}_{ii} \leq (1+\epsilon)^2 p$ for all $i \in [n] = \{1, 2, \ldots, n\}$, where $\mathbf{D}$ is the degree matrix. Additionally, assume that $i)$ input node representations are bounded in norm and not parallel to any other node representation, $ii)$ output node representations are upper bounded, $iii)$ sub-GCN feature partitions are generated at each iteration from a categorical distribution with uniform mean $\frac{1}{m}$.*

Given this set of assumptions, we derive the following result for GCN models trained with `GIST`. Given Assumption 1, if the number of hidden neurons within the two-layer GCN satisfies $d_1 = \Omega\left(\frac{n^3\zeta^2 T^2}{\delta^2\gamma(1-\gamma)^2\lambda_0^4}\left(n + \frac{d}{m^2}\|\bar{\mathbf{A}}^2\|_{1,1}\right)\right)$, then `GIST` with step-size $\eta = O\left(\frac{\lambda_0}{n^2\|\mathbf{A}^2\|_{1,1}}\right)$ converges with probability $1-\delta$ according to

$$\mathbb{E}_{[\mathcal{M}_{t-1}],\mathbf{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y} - \hat{\mathbf{y}}(t)\|_2^2\right] \leq \left(\gamma + (1-\gamma)\left(1 - \frac{\eta\lambda_0}{2}\right)^{\zeta}\right)^t \mathbb{E}_{\mathbf{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y} - \hat{\mathbf{y}}(0)\|_2^2\right] + O\left(\frac{(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1}\right).$$

A full proof of this result is deferred to Appendix C, but a sketch of the techniques used is as follows:

1. We define the `GIST-K` and show that it remains positive definite throughout training given our assumptions and sufficient overparameterization.
2. We show that local sub-GCN training converges linearly, given a positive definite `GIST-K`.
3. We analyze the change in training error when sub-GCNs are sampled (`subGCNs`), locally trained (`subTrain`), and aggregated (`subAgg`).
4. We establish a connection between local and aggregated weight perturbation, showing that network parameters are bounded by a small region centered around the initialization given sufficient overparameterization.

Discussion. Stated intuitively, the result in Theorem 4 shows that, given sufficient width, two-layer GCNs trained using `GIST` converge to approximately zero training error. The convergence rate is

m	d_0	d_1	d_2	Cora	Citeseer	Pubmed	OGBN-Arxiv
Baseline				81.52 ± 0.005	75.02 ± 0.018	75.90 ± 0.003	70.85 ± 0.089
2	✓	✓	✓	80.00 ± 0.010	$\mathbf{75.95} \pm 0.007$	76.68 ± 0.011	65.65 ± 0.700
	✓	✓		78.30 ± 0.011	69.34 ± 0.018	75.78 ± 0.015	65.33 ± 0.347
		✓	✓	$\mathbf{80.82} \pm 0.010$	75.82 ± 0.008	$\mathbf{78.02} \pm 0.007$	$\mathbf{70.10} \pm 0.224$
4	✓	✓	✓	76.78 ± 0.017	70.66 ± 0.011	65.67 ± 0.044	54.21 ± 1.360
	✓	✓		66.56 ± 0.061	68.38 ± 0.018	68.44 ± 0.014	52.64 ± 1.988
		✓	✓	$\mathbf{81.18} \pm 0.007$	$\mathbf{76.21} \pm 0.017$	$\mathbf{76.99} \pm 0.006$	$\mathbf{68.69} \pm 0.579$
8	✓	✓	✓	48.32 ± 0.087	45.42 ± 0.092	54.29 ± 0.029	40.26 ± 1.960
	✓	✓		53.60 ± 0.020	54.68 ± 0.030	51.44 ± 0.002	26.84 ± 7.226
		✓	✓	$\mathbf{79.58} \pm 0.006$	$\mathbf{75.39} \pm 0.016$	$\mathbf{76.99} \pm 0.006$	$\mathbf{65.81} \pm 0.378$

Table 1: Test accuracy of GCN models trained on small-scale datasets with `GIST`. We selectively partition each feature dimension within the GCN model, indicated by a check mark. *Partitioning on all hidden layers except the input layer leads to optimal performance.*

linear and on par with training the full, two-layer GCN model (*i.e.*, without the feature partition utilized in `GIST`), up to an error neighborhood. Such theory shows that the feature partitioning strategy of `GIST` does not cause the model to diverge in training. Additionally, the theory suggests that wider GCN models and a larger number of sub-GCNs should be used to maximize the convergence rate of `GIST` and minimize the impact of the additive term within Theorem 4; though the affect of m on the radius is less significant compared to d_1. Such findings reveal that `GIST` is particularly-suited towards training extremely wide models that cannot be trained using a traditional, centralized approach on a single GPU due to limited memory capacity.

5 Experiments

We use `GIST` to train different GCN architectures on six public, multi-node classification datasets; see Appendix A for details. In most cases, we compare the performance of models trained with `GIST` to that of models trained with standard methods (*i.e.*, single GPU with node partitioning). Comparisons to models trained with other distributed methodologies are also provided in Appendix B. Experiments are divided into small and large scale regimes based upon graph size. The goal of `GIST` is to i) train GCN models to state-of-the-art performance, ii) minimize wall-clock training time, and iii) enable training of very wide GCN models.

5.1 Small-Scale Experiments

In this section, we perform experiments over Cora, Citeseer, Pubmed, and OGBN-Arxiv datasets [43, 22]. For these small-scale datasets, we train a three-layer, 256-dimensional GCN model [28] with `GIST`; see Appendix A.3 for further experimental settings. All reported metrics are averaged across five separate trials. Because these experiments run quickly, we use them to analyze the impact of different design and hyperparameter choices rather than attempting to improve runtime (*i.e.*, speeding up such short experiments is futile).

Which layers should be partitioned? We investigate whether models trained with `GIST` are sensitive to the partitioning of features within certain layers. Although the output dimension d_3 is never partitioned, we selectively partition dimensions d_0, d_1, and d_2 to observe the impact on model performance; see Table 1. Partitioning input features (d_0) significantly degrades test accuracy because sub-GCNs observe only a portion of each node's input features (*i.e.*, this becomes more noticeable with larger m). However, other feature dimensions cause no performance deterioration when partitioned between sub-GCNs, **leading us to partition all feature dimensions other than** d_0 **and** d_L within the final `GIST` methodology; see Figure 2-(b).

How many Sub-GCNs to use? Using more sub-GCNs during `GIST` training typically improves runtime because sub-GCNs i) become smaller, ii) are each trained for fewer epochs, and iii) are trained in parallel. We find that **all models trained with `GIST` perform similarly for practical settings of** m; see Table 1. One may continue increasing the number sub-GCNs used within `GIST` until all GPUs are occupied or model performance begins to decrease.

`GIST` **Performance.** Models trained with `GIST` often exceed the performance of models trained with standard, single-GPU methodology; see Table 1. Intuitively, we hypothesize that the random feature partitioning within `GIST`, which loosely resembles dropout [47], provides regularization benefits

		Reddit Dataset					
L	m	GraphSAGE			GAT		
		F1	Time	Speedup	F1	Time	Speedup
2	-	96.09	105.78s	1.00×	89.57	1.19hr	1.00×
	2	96.40	70.29s	1.50×	90.28	0.58hr	2.05×
	4	96.16	68.88s	1.54×	90.02	0.31hr	3.86×
	8	95.46	76.68s	1.38×	89.01	0.18hr	6.70×
3	-	96.32	118.37s	1.00×	89.25	2.01hr	1.00×
	2	96.36	80.46s	1.47×	89.63	0.95hr	2.11×
	4	95.76	78.74s	1.50×	88.82	0.48hr	4.19×
	8	94.39	88.54s	(1.34×)	70.38	0.26hr	(7.67×)
4	-	96.32	120.74s	1.00×	88.36	2.77hr	1.00×
	2	96.01	91.75s	1.32×	87.97	1.31hr	2.11×
	4	95.21	78.74s	(1.53×)	78.42	0.66hr	(4.21×)
	8	92.75	88.71s	(1.36×)	66.30	0.35hr	(7.90×)

		Amazon2M Dataset					
L	m	GraphSAGE (d_i = 400)			GraphSAGE (d_i = 4096)		
		F1	Time	Speedup	F1	Time	Speedup
2	-	89.90	1.81hr	1.00×	91.25	5.17hr	1.00×
	2	88.36	1.25hr	(1.45×)	90.70	1.70hr	3.05×
	4	86.33	1.11hr	(1.63×)	89.49	1.13hr	(4.57×)
	8	84.73	1.13hr	(1.61×)	88.86	1.11hr	(4.65×)
3	-	90.36	2.32hr	1.00×	91.51	9.52hr	1.00×
	2	88.59	1.56hr	(1.49×)	91.12	2.12hr	4.49×
	4	86.46	1.37hr	(1.70×)	89.21	1.42hr	(6.72×)
	8	84.76	1.37hr	(1.69×)	86.97	1.34hr	(7.12×)
4	-	90.40	3.00hr	1.00×	91.61	14.20hr	1.00×
	2	88.56	1.79hr	(1.68×)	91.02	2.77hr	5.13×
	4	87.53	1.58hr	(1.90×)	89.07	1.65hr	(8.58×)
	8	85.32	1.56hr	(1.93×)	87.53	1.55hr	(9.13×)

Table 2: Performance of models trained with `GIST` on Reddit and Amazon2M. Parenthesis are placed around speedups achieved at a cost of >1 deterioration in F1 and m ="-" refers to the baseline. *Models trained with `GIST` train more quickly and achieve comparable F1 score to those trained with standard methodology. The benefits of `GIST` become more pronounced for wider models.*

during training, but some insight into the favorable performance of `GIST` is also provided by the theoretical guarantees outlined in Section 4.

5.2 Large-Scale Experiments

For large-scale experiments on Reddit and Amazon2M, the baseline model is trained on a single GPU and compared to models trained with `GIST` in terms of F1 score and training time. All large-scale graphs are partitioned into 15,000 sub-graphs during training.[5] Graph partitioning is mandatory because the training graphs are too large to fit into memory. One could instead use layer sampling to make training tractable (see Section A.6), but we adopt graph partitioning in most experiments because the implementation is simple and performs well.

Reddit Dataset. We perform tests with 256-dimensional GraphSAGE [19] and GAT [50] models with two to four layers on Reddit; see Appendix A.4 for more details. As shown in Table 2, utilizing `GIST` significantly accelerates GCN training (*i.e.*, a 1.32× to 7.90× speedup). `GIST` performs best in terms of F1 score with $m = 2$ sub-GCNs (*i.e.*, $m = 4$ yields further speedups but F1 score decreases). Interestingly, *the speedup provided by `GIST` is more significant for models and datasets with larger compute requirements*. For example, experiments with the GAT architecture, which is more computationally expensive than GraphSAGE, achieve a near-linear speedup with respect to m.

[5]Single-GPU training with graph partitioning via METIS is the same approach adopted by ClusterGCN [10], making our single-GPU baseline a ClusterGCN model. We adopt the same number of sub-graphs as proposed in this work.

L	m	F1 Score (Time in hours)				
		$d_i = 400$	$d_i = 4096$	$d_i = 8192$	$d_i = 16384$	$d_i = 32768$
2	-	89.38 (1.81)	90.58 (5.17)	OOM	OOM	OOM
	2	87.48 (1.25)	90.09 (1.70)	90.87 (2.76)	90.94 (9.31)	90.91 (32.31)
	4	84.82 (1.11)	88.79 (1.13)	89.76 (1.49)	90.10 (2.24)	90.17 (5.16)
	8	82.56 (1.13)	87.16 (1.11)	88.31 (1.20)	88.89 (1.39)	89.46 (1.76)
3	-	89.73 (2.32)	90.99 (9.52)	OOM	OOM	OOM
	2	87.79 (1.56)	90.40 (2.12)	90.91 (4.87)	91.05 (17.7)	OOM
	4	85.30 (1.37)	88.51 (1.42)	89.75 (2.07)	90.15 (3.44)	OOM
	8	82.84 (1.37)	86.12 (1.34)	88.38 (1.37)	88.67 (1.88)	88.66 (2.56)
4	-	89.77 (3.00)	91.02 (14.20)	OOM	OOM	OOM
	2	87.75 (1.79)	90.36 (2.77)	91.08 (6.92)	91.09 (26.44)	OOM
	4	85.32 (1.58)	88.50 (1.65)	89.76 (2.36)	90.05 (4.93)	OOM
	8	83.45 (1.56)	86.60 (1.55)	88.13 (1.61)	88.44 (2.30)	OOM

Table 3: Performance of GraphSAGE models of different widths trained with GIST on Amazon2M. $m =$"-" refers to the baseline and "OOM" marks experiments that cause out-of-memory errors. *GIST enables training of higher-performing, ultra-wide models.*

Amazon2M Dataset. Experiments are performed with two, three, and four-layer GraphSAGE models [19] with hidden dimensions of 400 and 4096 (we refer to these models as "narrow" and "wide", respectively). We compare the performance (*i.e.*, F1 score and wall-clock training time) of GCN models trained with standard, single-GPU methodology to that of models trained with GIST; see Table 2. Narrow models trained with GIST have a lower F1 score in comparison to the baseline, but training time is significantly reduced. For wider models, GIST provides a more significant speedup (*i.e.*, up to $7.12\times$) and tends to achieve comparable F1 score in comparison to the baseline, revealing that *GIST works best with wider models*.

Within Table 2, models trained with GIST tend to achieve a wall-clock speedup at the cost of a lower F1 score (*i.e.*, observe the speedups marked with parenthesis in Table 2). When training time is analyzed with respect to a fixed F1 score, we observe that the baseline takes significantly longer than GIST to achieve a fixed F1 score. For example, when $L = 2$, a wide GCN trained with GIST ($m = 8$) reaches an F1 score of 88.86 in ~4,000 seconds, *while models trained with standard methodology take ~10,000 seconds to achieve a comparable F1 score.* As such, GIST significantly accelerates training relative to model performance.

5.3 Training Ultra-Wide GCNs

We use GIST to train GraphSAGE models with widths as high as 32K (*i.e.*, **8**$\times$ beyond the capacity of a single GPU); see Table 3 for results and Appendix A.5 for more details. Considering $L = 2$, the best-performing, single-GPU GraphSAGE model ($d_i = 4096$) achieves an F1 score of 90.58 in 5.2 hours. With GIST ($m = 2$), we achieve a higher F1 score of 90.87 in 2.8 hours (*i.e.*, a $1.86\times$ speedup) using $d_i = 8192$, which is beyond single GPU capacity. Similar patterns are observed for deeper models. Furthermore, we find that utilizing larger hidden dimensions yields further performance improvements, revealing the utility of wide, overparameterized GCN models. *GIST, due to its feature partitioning strategy, is unique in its ability to train models of such scale to SOTA performance*.

6 Conclusions and Future Work

We present GIST, a distributed training approach for GCNs that enables the exploration of larger models and datasets. GIST is compatible with existing sampling approaches and leverages a feature-wise partition of model parameters to construct smaller sub-GCNs that are trained independently and in parallel. We have shown that GIST achieves remarkable speed-ups over large graph datasets and even enables the training of GCN models of unprecedented size. We hope GIST can empower the exploration of larger, more powerful GCN architectures within the graph community.

Several interesting avenues of analysis remain that are relevant to GIST. Given that GIST can be applied to any GCN architecture, using models that exploit edge features within the graph could yield further performance improvements. For example, the EGNN model [16] injects edge information into the GCN model via the adjacency matrix at each layer, which—similarly to node partitioning strategies—operates orthogonally to the feature partitioning strategy of GIST and is therefore compatible. Furthermore, the analysis of GIST could be extended to alternative tasks (*e.g.*, link prediction) and larger-scale datasets. However, performing experiments over datasets larger than Amazon2M is difficult due to the lack of moderately-large-scale graphs that are available publicly. For example, the only graph larger than Amazon2M provided via the Open Graph Benchmark [22] is Papers100M, which requires 256 Gb of CPU RAM to load.

References

[1] Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

[2] Alexandru T Balaban. Applications of Graph Theory in Chemistry. *Journal of Chemical Information and Computer Sciences*, 1985.

[3] Tal Ben-Nun and Torsten Hoefler. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. *ACM Computing Surveys (CSUR)*, 2019.

[4] Gil Benkö, Christoph Flamm, and Peter F Stadler. A graph-based toy model of chemistry. *Journal of Chemical Information and Computer Sciences*, 2003.

[5] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 2017.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Tom B. Brown et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[8] Jianfei Chen, Jun Zhu, and Le Song. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[9] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[10] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019.

[11] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised Cross-lingual Representation Learning at Scale. *arXiv preprint arXiv:1911.02116*, 2019.

[12] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

[13] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks, 2019.

[14] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-Scale Learnable Graph Convolutional Networks. *arXiv preprint arXiv:1808.03965*, 2018.

[15] Amir Gholami, Ariful Azad, Peter Jin, Kurt Keutzer, and Aydin Buluc. Integrated Model, Batch and Domain Parallelism in Training Neural Networks. *arXiv preprint arXiv:1712.04432*, 2017.

[16] Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9211–9219, 2019.

[17] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, 2005.

[18] S. Günther, L. Ruthotto, J. B. Schroder, E. C. Cyr, and N. R. Gauger. Layer-Parallel Training of Deep Residual Neural Networks. *arXiv preprint arXiv:1812.04352*, 2018.

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[20] Karen Hao. Training a single ai model can emit as much carbon as five cars in their lifetimes, June 2019.

[21] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[22] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv e-prints*, page arXiv:2005.00687, May 2020.

[23] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive Sampling Towards Fast Graph Representation Learning. *arXiv preprint arXiv:1809.05343*, 2018.

[24] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[25] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 1998.

[26] George Karypis and Vipin Kumar. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 1998.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*, 2016.

[29] Andrew C. Kirby, Siddharth Samsi, Michael Jones, Albert Reuther, Jeremy Kepner, and Vijay Gadepally. Layer-Parallel Training with GPU Concurrency of Deep Residual Neural Networks via Nonlinear Multigrid. *arXiv preprint arXiv:2007.07336*, 2020.

[30] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[31] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[32] Fangshuo Liao and Anastasios Kyrillidis. On the convergence of shallow neural network training with randomly masked neurons, 2021.

[33] Tao Lin, Sebastian U. Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't Use Large Mini-Batches, Use Local SGD. *arXiv preprint arXiv:1808.07217*, 2018.

[34] Dean Lusher, Johan Koskinen, and Garry Robins. *Exponential random graph models for social networks: Theory, methods, and applications*. Cambridge University Press, 2013.

[35] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2015.

[36] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep Double Descent: Where Bigger Models and More Data Hurt. *arXiv preprint arXiv:1912.02292*, 2019.

[37] Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 2002.

[38] Samet Oymak and Mahdi Soltanolkotabi. Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(1):84–105, 2020.

[39] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[40] J. Gregory Pauloski, Zhao Zhang, Lei Huang, Weijia Xu, and Ian T. Foster. Convolutional Neural Network Training with Distributed K-FAC. *arXiv preprint arXiv:2007.00784*, 2020.

[41] Tony Peng and Michael Sarazen. The staggering cost of training sota ai models, June 2019.

[42] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[43] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29:93–93, 2008.

[44] Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*, 2020.

[45] Shaohuai Shi, Zhenheng Tang, Xiaowen Chu, Chengjian Liu, Wei Wang, and Bo Li. A Quantitative Survey of Communication Optimizations in Distributed Deep Learning. *arXiv preprint arXiv:2005.13247*, 2020.

[46] Zhao Song and Xin Yang. Quadratic suffices for over-parametrization via matrix chernoff bound, 2020.

[47] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 2014.

[48] Sebastian U. Stich. Local SGD converges fast and communicates little. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[49] Sanket Tavarageri, Srinivas Sridharan, and Bharat Kaul. Automatic Model Parallelism for Deep Neural Networks with Compiler and Hardware Support. *arXiv preprint arXiv:1906.08168*, 2019.

[50] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[51] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. L2-gcn: Layer-wise and learned efficient training of graph convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2127–2135, 2020.

[52] Kwangmin Yu, Thomas Flynn, Shinjae Yoo, and Nicholas D'Imperio. Layered sgd: A decentralized and synchronous sgd algorithm for scalable deep neural network training. *arXiv preprint arXiv:1906.05936*, 2019.

[53] Binhang Yuan, Anastasios Kyrillidis, and Christopher M. Jermaine. Distributed Learning of Deep Neural Networks using Independent Subnet Training. *arXiv preprint arXiv:1810.01392*, 2019.

[54] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph Sampling Based Inductive Learning Method. *arXiv preprint arXiv:1907.04931*, 2019.

[55] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[56] Z. Zhang, L. Yin, Y. Peng, and D. Li. A quick survey on large scale distributed deep learning systems. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, 2018.

[57] Wentao Zhu, Can Zhao, Wenqi Li, Holger Roth, Ziyue Xu, and Daguang Xu. LAMP: Large Deep Nets with Automated Model Parallelism for Image Segmentation. *arXiv preprint arXiv:2006.12575*, 2020.

[58] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 2595–2603, 2010.

[59] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks. *arXiv preprint arXiv:1911.07323*, 2019.

A Experimental Details

A.1 Datasets

The details of the datasets utilized within `GIST` experiments in Section 5 are provided in Table 4. Cora, Citeseer, PubMed and OGBN-Arxiv are considered "small-scale" datasets and are utilized within experiments in Section 5.1. Reddit and Amazon2M are considered "large-scale" datasets and are utilized within experiments in Section 5.2.

Dataset	n	# Edges	# Labels	d
Cora	2,708	5,429	7	1,433
CiteSeer	3,312	4,723	6	3,703
Pubmed	19,717	44,338	3	500
OGBN-Arxiv	169,343	1.2M	40	128
Reddit	232,965	11.6 M	41	602
Amazon2M	2.5 M	61.8 M	47	100

Table 4: Details of relevant datasets.

A.2 Implementation Details

We provide an implementation of `GIST` in PyTorch [39] using the NCCL distributed communication package for training GCN [28], GraphSAGE [19] and GAT [50] architectures. Our implementation is centralized, meaning that a single process serves as a central parameter server. From this central process, the weights of the global model are maintained and partitioned to different worker processes (including itself) for independent training. Experiments are conducted with 8 NVIDIA Tesla V100-PCIE-32G GPUs, a 56-core Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz, and 256 GB of RAM.

A.3 Small-Scale Experiments

Small-scale experiments in Section 5.1 are performed using Cora, Citeseer, Pubmed, and OGBN-Arxiv datasets [43, 22]. `GIST` experiments are performed with two, four, and eight sub-GCNs in all cases. We find that the performance of models trained with `GIST` is relatively robust to the number of local iterations ζ, but test accuracy decreases slightly as ζ increases; see Figure 5. Based on the results in Figure 5, we adopt $\zeta = 20$ for Cora, Citeseer, and Pubmed, as well as $\zeta = 100$ for OGBN-Arxiv.

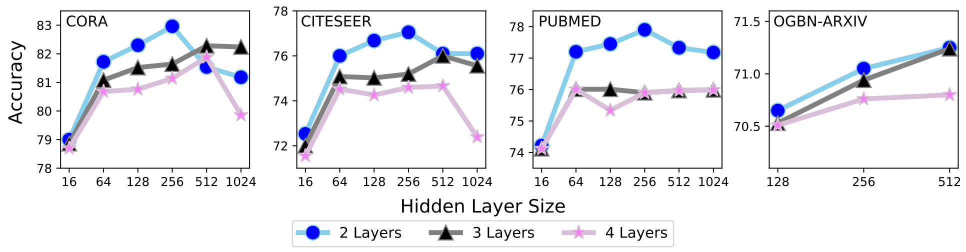

Figure 4: Test accuracy for different sizes (*i.e.*, varying depth and width) of GCN models trained with standard, single-GPU methodology on small-scale datasets. *We adopt three-layer, 256-dimensional GCN models as our baseline architecture.*

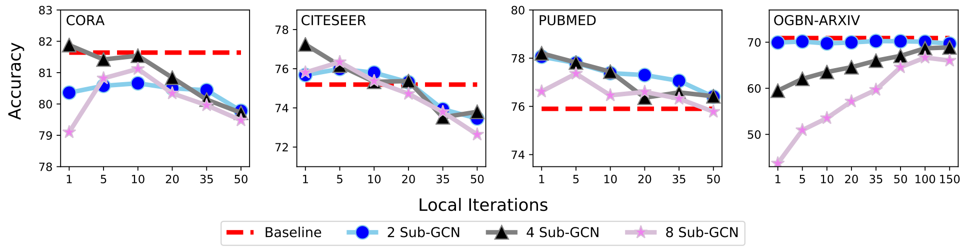

Figure 5: Test accuracy of GCN models trained on small-scale datasets with `GIST` using different numbers of local iterations and sub-GCNs. *Models trained with `GIST` are surprisingly robust to the number of local iterations used during training, no matter the number of sub-GCNs.*

Experiments are run for 400 epochs with a step learning rate schedule (*i.e.*, $10\times$ decay at 50% and 75% of total epochs). A vanilla GCN model, as described in [28], is used. The model is trained in a full-batch manner using the Adam optimizer [27]. No node sampling techniques are employed because the graph is small enough to fit into memory. All reported results are averaged across five trials with different random seeds. For all models, d_0 and d_L are respectively given by the number of features and output classes in the dataset. The size of all hidden layers is the same, but may vary across experiments.

We first train baseline GCN models of different depths and hidden dimensions using a single GPU to determine the best model depth and hidden dimension to be used in small-scale experiments. The results are shown in Figure 4. Deeper models do not yield performance improvements for small-scale datasets, but test accuracy improves as the model becomes wider. Based upon the results in Figure 4, we adopt a three-layer GCN with a hidden dimension of $d_1 = d_2 = 256$ as the underlying model used in small-scale experiments. Though two-layer models seem to perform best, we use a three-layer model within Section 5.1 to enable more flexibility in examining the partitioning strategy of `GIST`.

A.4 Large-Scale Experiments

Reddit Dataset. For experiments on Reddit, we train 256-dimensional GraphSAGE and GAT models using both `GIST` and standard, single-GPU methodology. During training, the graph is partitioned into 15,000 sub-graphs. Training would be impossible without such partitioning because the graph is too large to fit into memory. The setting for the number of sub-graphs is the optimal setting proposed in previous work [10]. Models trained using `GIST` and standard, single-GPU methodologies are compared in terms of F1 score and training time.

All tests are run for 80 epochs with no weight decay, using the Adam optimizer [27]. We find that $\zeta = 500$ achieves consistently high performance for models trained with `GIST` on Reddit. We adopt a batch size of 10 sub-graphs throughout the training process, which is the optimal setting proposed in previous work [10].

L	# Sub-GCNs	GIST + LADIES		
		F1 Score	Time	Speedup
2	Baseline	89.73	3359.91s	1.00×
	2	89.29	1834.59s	1.83×
	4	88.42	1158.51s	2.90×
3	Baseline	89.57	4803.88s	1.00×
	2	86.52	2635.18s	1.82×
	4	86.72	1605.32s	3.00×

Table 5: Performance of GCN models trained with a combination of GIST and LADIES [59] on Reddit. Here, the baseline represents models trained with LADIES in a standard, single-GPU manner. *Combining GIST with layer sampling leads to further improvements in wall-clock training time without deteriorating the F1 score.*

Amazon2M Dataset. For experiments on Amazon2M, we train two to four layer GraphSAGE models with hidden dimensions of 400 and 4096 using both GIST and standard, single-GPU methodology. We follow the experimental settings of [10]. The training graph is partitioned into $15{,}000$ sub-graphs and a batch size of 10 sub-graphs is used. We find that using $\zeta = 5000$ performs consistently well. Models are trained for 400 total epochs with the Adam optimizer [27] and no weight decay.

A.5 Training Ultra-Wide GCNs

All settings for ultra-wide GCN experiments in Section 5.3 are adopted from the experimental settings of Section 5.2; see Appendix A.4 for further details. For $d_i > 4096$ evaluation must be performed on graph partitions (not the full graph) to avoid memory overflow. As such, the graph is partitioned into $5{,}000$ sub-graphs during testing and F1 score is measured over each partition and averaged. All experiments are performed using a GraphSAGE model, and the hidden dimension of the underlying model is changed between different experiments.

A.6 GIST with Layer Sampling

As previously mentioned, some node partitioning approach must be adopted to avoid memory overflow when the underlying training graph is large. Although graph partitioning is used within most experiments (see Section 5.2), GIST is also compatible with other node partitioning strategies. To demonstrate this, we perform training on Reddit using GIST combined with a recent layer sampling approach [59] (*i.e.*, instead of graph partitioning).

As shown in Table 5, combining GIST with layer sampling enables training on large-scale graphs, and the observed speedup actually exceeds that of GIST with graph partitioning. For example, GIST with layer sampling yields a $1.83\times$ speedup when $L = 2$ and $m = 2$, in comparison to a $1.50\times$ speedup when graph partitioning is used within GIST (see Table 2). As the number of sub-GCNs is increased beyond $m = 2$, GIST with layer sampling continues to achieve improvements in wall-clock training time (*e.g.*, speedup increases from $1.83\times$ to $2.90\times$ from $m = 2$ to $m = 4$ for $L = 2$) without significant deterioration to model performance. Thus, although node partitioning is needed to enable training on large-scale graphs, the feature partitioning strategy of GIST is compatible with numerous sampling strategies (*i.e.*, not just graph sampling).

Experiments adopt the same experimental settings as Section 5.2 for the Reddit dataset; see Appendix A.4 for further details. Within these experiments, we combine GIST with LADIES [59], a recent layer sampling approach for efficient GCN training. LADIES is used instead of graph partitioning. Any node sampling approach can be adopted—some sampling approach is just needed to avoid memory overflow.

We train 256-dimensional GCN models with either two or three layers. We utilize a vanilla GCN model within this section (as opposed to GraphSAGE or GAT) to simplify the implementation of GIST with LADIES, which creates a disparity in F1 score. Experiments compare the performance of the same models trained either with GIST or using standard, single-GPU methodology. In this case, the single-GPU model is just a GCN trained with LADIES.

# Machines	Method	F1 Score	Training Time
2	Local SGD	96.37	137.17s
	GIST	**96.40**	**108.67s**
4	Local SGD	95.00	127.63s
	GIST	**96.16**	**116.56s**
8	Local SGD	93.40	129.58s
	GIST	**95.46**	**123.83s**

Table 6: Performance of GraphSAGE models trained using local SGD and GIST on Reddit. We adopt settings described in Section 5.2, but use 100 local iterations for both GIST and local SGD. *Models trained with GIST outperform those trained with local SGD in terms of test F1 score and wall-clock training time in all cases.*

B GIST vs. Other Distributed Training Methods

Although GIST has been shown to provide benefits in terms of GCN performance and training efficiency in comparison to standard, single-GPU training, other choices for the distributed training of GCNs exist. Within this section, we compare GIST to other natural choices for distributed training, revealing that GCN models trained with GIST achieve favorable performance in comparison to those trained with other common distributed training techniques.

B.1 Local SGD

A simple version of local SGD [33] can be implemented for distributed training of GCNs by training the full model on each separate worker for a certain number of local iterations and intermittently averaging local updates. In comparison to such a methodology, GIST has better computational and communication efficiency because $i)$ it communicates only a small fraction of model parameters to each machine and $ii)$ locally training narrow sub-GCNs is faster than locally training the full model. We perform a direct comparison between local SGD and GIST on the Reddit dataset using a two-layer, 256-dimensional GraphSAGE model; see Table 6. As can be seen, GCN models trained with GIST have lower wall-clock training time and achieve better performance than those trained with local SGD in all cases.

# Machines	Method	F1 Score	Inference Time
2	Ensemble	96.31	3.59s
	GIST	**96.40**	**1.81s**
4	Ensemble	96.10	6.38s
	GIST	**96.16**	**1.81s**
8	Ensemble	95.28	11.95s
	GIST	**95.46**	**1.81s**

Table 7: Performance of GraphSAGE models trained both with GIST and as ensembles of shallow sub-GCNs on Reddit. *Models trained with GIST perform better and do not suffer from increased inference time as the number of sub-GCNs is increased.*

B.2 Sub-GCN Ensembles

As previously mentioned, increasing the number of local iterations (*i.e.*, ζ in Algorithm 1) decreases communication requirements given a fixed amount of training. When taken to the extreme (*i.e.*, $\zeta \to \infty$), one could minimize communication requirements by never aggregating sub-GCN parameters, thus forming an ensemble of independently-trained sub-GCNs. We compare GIST to such a methodology[6] in Table 7 using a two-layer, 256-dimensional GraphSAGE model on the Reddit dataset. Though training ensembles of sub-GCNs minimizes communication, Table 7 reveals that $i)$

[6]For each sub-GCN, we measure validation accuracy throughout training and add the highest-performing model into the ensemble.

models trained with GIST achieve better performance and $ii)$ inference time for sub-GCN ensembles becomes burdensome as the number of sub-GCNs is increased.

C Theoretical Results

C.1 Formulation of GIST for One-Hidden-Layer GCNs

In our analysis, we consider a GCN with one hidden-layer and a ReLU activation. We assume that the GCN outputs a scalar value $\tilde{y}_i$ for each node in the graph. Denoting $\tilde{\mathbf{y}} = [\tilde{y}_1, \dots, \tilde{y}_n]$, we can write the output of the GCN as

$$\tilde{\mathbf{y}} = \frac{1}{\sqrt{d_1}} \bar{\mathbf{A}} \sigma(\bar{\mathbf{A}} \mathbf{X} \boldsymbol{\Theta}) \mathbf{a}$$

where $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{d_1}] \in \mathbb{R}^{n \times d_1}$ is the weights within the GCN's first layer and $\mathbf{a} = [a_1, \dots, a_{d_1}] \in \mathbb{R}^{d_1}$ is the weights within the GCN's second layer. To simplify the analysis, we denote $\hat{\mathbf{X}} = \bar{\mathbf{A}}\mathbf{X} = [\hat{\mathbf{x}}_1, \dots \hat{\mathbf{x}}_n]$. Then, we have $\hat{\mathbf{x}}_i = \sum_{i'=1}^{n} \bar{\mathbf{A}}_{ii'} \mathbf{x}_{i'}$ and the output of each node within the graph can be written as

$$\tilde{y}_i = \frac{1}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r=1}^{d_1} \bar{\mathbf{A}}_{ii'} a_r \sigma(\langle \hat{\mathbf{x}}_{i'}, \boldsymbol{\theta}_r \rangle)$$

As in previous convergence analysis for training neural networks, we assume that second-layer weights $\mathbf{a}$ are fixed and only the first layer weights $\boldsymbol{\Theta}$ are trainable. Following the GIST feature partitioning strategy, we only partition the hidden layer. Specifically, in global iteration t, sub-GCNs are constructed by sampling a set of masks $\mathcal{M}_t \in \mathbb{R}^{m \times d_1}$. We denote the jth column of $\mathcal{M}_t$ as $\mathcal{M}_t^{(j)} \in \mathbb{R}^m$, the rth row of $\mathcal{M}_t$ as $\mathcal{M}_{t,r} \in \mathbb{R}^{d_1}$, and the entry in the rth row and jth column as $\mathcal{M}_{t,r}^{(j)}$. Each $\mathcal{M}_{t,r}^{(j)}$ is a binary values: $\mathcal{M}_{t,r}^{(j)} = 1$ if neuron r is active in sub-GCN j, and $\mathcal{M}_{t,r}^{(j)} = 0$ otherwise. Using this mask notation, the output for node i within sub-GCN j can be written as

$$\hat{y}_i^{(j)}(t,k) = f_{\mathcal{M}_t^{(j)}}(\boldsymbol{\Theta}_{t,k}^{(j)}, \mathbf{X})_i = \frac{1}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r=1}^{d_1} \bar{\mathbf{A}}_{ii'} \mathcal{M}_{t,r}^{(j)} a_r \sigma\left(\left\langle \hat{\mathbf{x}_{i'}}, \boldsymbol{\theta}_{t,k,r}^{(j)} \right\rangle\right)$$

t and k denote the current global and local iterations, respectively. We assume that each $\mathcal{M}_{t,r}$ is sampled from a one-hot categorical distribution. We formally define the random variables $\mathcal{M}_{t,r}^{(j)}$ as follows: Let each $\hat{m}_{t,r}$ be a uniform random variable on the index set $[m] = \{1, \dots, m\}$, *i.e.*, $\mathcal{P}(\hat{m}_{t,r} = j) = \frac{1}{m}$ for $j \in [m]$. Then, we define each mask entry as $\mathcal{M}_{t,r}^{(j)} = \mathbb{I}\{\hat{m}_{k,r} = j\}$. Masks sampled in such a fashion have the following properties

- $\mathcal{P}(\mathcal{M}_{t,r}^{(j)} = 1) = \frac{1}{m}$
- $\mathcal{P}(\mathcal{M}_{t,r}^{(j)} = 0) = 1 - \frac{1}{m}$
- $\sum_{j=1}^{m} \mathcal{M}_{t,r}^{(j)} = 1$
- $\mathcal{M}_{t,r}^{(j)} \mathcal{M}_{t,r}^{(j')} = 0$ if $j' \neq j$.

Here, the first and second properties guarantee that the expected number of neurons active in each sub-GCN is equal. The third and fourth properties guarantee that each neuron is active in one and only one sub-GCN. Within this setup, we consider the GIST training procedure, described as

$$\begin{aligned} \boldsymbol{\theta}_{t,0,r}^{(j)} &= \boldsymbol{\theta}_{t,r} \\ \boldsymbol{\theta}_{t,k+1,r}^{(j)} &= \boldsymbol{\theta}_{t,k,r}^{(j)} - \eta \frac{\partial L(\boldsymbol{\Theta}_{t,k}^{(j)})}{\partial \boldsymbol{\theta}_r} \\ \boldsymbol{\theta}_{t+1,r} &= \boldsymbol{\theta}_{t,r} + \sum_{j=1}^{m} \left(\boldsymbol{\theta}_{t,\zeta,r}^{(j)} - \boldsymbol{\theta}_{t,0,r}^{(j)} \right) \end{aligned} \tag{3}$$

Within this formulation, ζ represents the total number of local iterations performed for each sub-GCN, while $L(\mathbf{\Theta}_{t,k}^{(j)})$ is the loss on the jth sub-GCN during the t-th global and kth local iteration. We can express $L(\mathbf{\Theta}_{t,k}^{(j)})$ as

$$L\left(\mathbf{\Theta}_{t,k}^{(j)}\right) = \left\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\right\|_2^2 = \left\|\mathbf{y} - f_{\mathcal{M}_t^{(j)}}(\mathbf{\Theta}_{t,k}^{(j)}, \mathbf{X})\right\|_2^2$$

and the gradient has the form

$$\frac{\partial L(\mathbf{\Theta}_{t,k}^{(j)})}{\partial \boldsymbol{\theta}_r} = \frac{1}{\sqrt{d_1}} \sum_{i=1}^{n} \sum_{i'=1}^{n} \left(\hat{y}_i^{(j)}(t,k) - y_i\right) \bar{\mathbf{A}}_{ii'} \mathcal{M}_{t,r}^{(j)} a_r \hat{\mathbf{x}}_{i'} \mathbb{I}\left\{\left\langle \boldsymbol{\theta}_{t,k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \geq 0\right\}$$

C.2 Properties of the Transformed Input

The GCN [28] uses a first-degree Chebyshev polynomial to approximate a spectral convolution on the graph, which results in an aggregation matrix of the form

$$\bar{\mathbf{A}} = \mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \tag{4}$$

where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}$ is the degree matrix with $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$. In practice, the re-normalization trick is applied to control the magnitude of the largest eigenvalue of $\bar{\mathbf{A}}$. Here, however, we keep the original formulation of (4) to facilitate our analysis, and our assumption on the depth of the GCN does not lead to numerical instability even if $\lambda_{\max}(\bar{\mathbf{A}}) > 1$. It is a well-known result that $2 = \lambda_{\max}(\bar{\mathbf{A}}) \geq \lambda_{\min}(\bar{\mathbf{A}}) \geq 0$. In particular, the lower bound on the minimum eigenvalue is obtained by considering

$$\mathbf{v}^\top \bar{\mathbf{A}} \mathbf{v} = \sum_{i=1}^{n} v_i^2 + \sum_{(i,j)\in E} \frac{v_i v_j}{\sqrt{\mathbf{D}_{ii}\mathbf{D}_{jj}}} = \sum_{(i,j)\in E} \left(\frac{v_i}{\sqrt{\mathbf{D}_{ii}}} + \frac{v_j}{\sqrt{\mathbf{D}_{jj}}}\right)^2$$

In our analysis, we require the aggregation matrix $\bar{\mathbf{A}}$ to be positive definite. Thus, the following assumption can be made about $\lambda_{\min}(\bar{\mathbf{A}})$.

Assumption 2 $\lambda_{\min}(\bar{\mathbf{A}}) \neq 0$.

Going further, we must make a few more assumptions about the aggregation matrix and the graph itself to satisfy certain properties relevant to the analysis. First, the following property must hold

For all $i \in [n]$, we have $\|\hat{\mathbf{x}}_i\|_2 \leq 1$.

which can be guaranteed by the following assumption.

Assumption 3 *There exists $\epsilon \in (0,1)$ and $p \in \mathbb{Z}_+$ such that*

$$(1-\epsilon)^2 p \leq \mathbf{D}_{ii} \leq (1+\epsilon)^2 p$$

for all $i \in [n]$.

Additionally, we make the following assumption regarding the graph itself

Assumption 4 *For all $i \in [n]$, we have $\|\mathbf{x}_i\|_2 \leq \frac{1-\epsilon}{2}$, and $|y_i| \leq C$ for some constant C. Moreover, for all $j \in [n]$ and $j \neq i$, we have $\mathbf{x}_i \not\parallel \mathbf{x}_j$.*

which, in turn, yields the following property

For all $i, j \in [n]$ such that $i \neq j$, we have $\hat{\mathbf{x}}_i \not\parallel \hat{\mathbf{x}}_j$.

C.3 Full Statement Theorem 4

We now state the full version of Theorem 4 from Section 4, which characterizes the convergence properties of one-hidden-layer GCN models trained with `GIST`. The full proof of this Theorem is provided within Appendix C.5. Suppose assumptions 2-4, and property C.2 hold. Moreover, suppose in each global iteration the masks are generated from a categorical distribution with uniform mean $1/m$. Fix the number of global iterations to T and local iterations to ζ. If the number of hidden neurons satisfies $d_1 = \Omega\left(\frac{n^3\zeta^2T^2}{\delta^2\gamma(1-\gamma)^2\lambda_0^4}\left(n + \frac{d}{m^2}\|\bar{\mathbf{A}}^2\|_{1,1}\right)\right)$, then procedure (3) with constant step size $\eta = O\left(\frac{\lambda_0}{n^2\|\bar{\mathbf{A}}^2\|_{1,1}}\right)$ converges according to

$$\mathbb{E}_{[\mathcal{M}_{t-1}],\mathbf{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right] \leq \left(\gamma + (1-\gamma)\left(1-\frac{\eta\lambda_0}{2}\right)^{\zeta}\right)^{t}\mathbb{E}_{\mathbf{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(0)\|_2^2\right] + O\left(\frac{(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1}\right)$$

with probability at least $1-\delta$.

C.4 `GIST` and Local Training Progress

For a one-hidden-layer MLP, the analysis often depends on the (scaled) Gram Matrix of the infinite-dimensional NTK

$$\mathbf{H}_{ij}^{\infty} = \frac{1}{d_1 m}\langle\hat{\mathbf{x}}_i,\hat{\mathbf{x}}_j\rangle\,\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{N}(0,\mathbf{I})}\left[\mathbb{I}\{\langle\hat{\mathbf{x}}_i,\boldsymbol{\theta}\rangle\geq 0,\langle\hat{\mathbf{x}}_j,\boldsymbol{\theta}\rangle\geq 0\}\right]$$

We can extend this definition of the Gram Matrix to an infinite-width, one-hidden-layer GCN as follows

$$\mathbf{G}^{\infty} = \bar{\mathbf{A}}\mathbf{H}^{\infty}\bar{\mathbf{A}}$$

With property C.2, prior work [13] shows that $\lambda_{\min}(\mathbf{H}) > 0$. Denoting $\lambda_0 = \lambda_{\min}(\mathbf{G}^{\infty})$, since $\bar{\mathbf{A}}$ is also positive definite, we have that $\lambda_0 \geq \lambda_{\min}(\mathbf{H})\lambda_{\min}(\bar{\mathbf{A}}) > 0$. In our analysis, we define the Graph Independent Subnetwork Tangent Kernel (`GIST-K`)

$$\mathbf{G}^{(j)}(t,t',k) = \bar{\mathbf{A}}\mathbf{H}(t,t',k)\bar{\mathbf{A}}$$

where $\mathbf{H}(t,t',k)$ is defined as

$$\mathbf{H}(t,t',k) = \frac{1}{d_1}\langle\hat{\mathbf{x}}_i,\hat{\mathbf{x}}_j\rangle\sum_{r=1}^{d_1}\mathcal{M}_{t,r}^{(j)}\mathbb{I}\left\{\left\langle\hat{\mathbf{x}}_i,\boldsymbol{\theta}_{t',k,r}^{(j)}\right\rangle\geq 0,\left\langle\hat{\mathbf{x}}_j,\boldsymbol{\theta}_{t',k,r}^{(j)}\right\rangle\geq 0\right\}$$

for masks $\mathcal{M}_t$ and weights $\mathbf{\Theta}_{t',k}^{(j)}$. Following previous work [32] on subnetwork theory, the following Lemma can be obtained. Suppose the number of hidden nodes satisfies $d_1 = \Omega\left(\lambda_0^{-1}n^2\log{Tmn/\delta}\right)$. If for all t,k it holds that $\|\boldsymbol{\theta}_{t,k,r}-\boldsymbol{\theta}_{0,r}\|_2 \leq R := \frac{\lambda_0}{48n}$, then with probability at least $1-\delta$, for all $t,t'\in[T]$ we have:

$$\lambda_{\min}(\mathbf{G}^{(j)}(t,t',k)) \geq \tfrac{\lambda_0}{2}.$$

After showing that every `GIST-K` is positive definite, we can then show that the local training of each sub-GCN enjoys a linear convergence rate. Suppose the number of hidden nodes satisfies $d_1 = \Omega\left(\lambda_0^{-1}n^2\log{Tmn/\delta}\right)$. If for all $r\in[d_1]$ it holds that

$$\|\boldsymbol{\theta}_{t,r}-\boldsymbol{\theta}_{0,r}\|_2 + \frac{4T\eta\zeta}{\delta\alpha}\sqrt{\frac{n}{d_1}}\mathbb{E}_{[\mathcal{M}_{t-1}],\mathbf{W}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right]^{\frac{1}{2}} + (T-t)B \leq R \tag{5}$$

with

$$B = \sqrt{\frac{8(m-1)\eta\zeta n}{md_1}}\left(\sqrt{\frac{8(m-1)\|\bar{\mathbf{A}}^2\|_{1,1}d}{\gamma m}} + \sqrt{\frac{\eta\zeta nT}{\delta}}\right); \quad R \leq \frac{\lambda_0}{96n}$$

then we have

$$\left\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k+1)\right\|_2^2 \leq \left(1-\frac{\eta\lambda_0}{2}\right)\left\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k)\right\|_2^2$$

and for all $r \in [d_1], j \in [m]$ it holds that

$$\left\|\boldsymbol{\theta}_{t,\zeta,r}^{(j)}-\boldsymbol{\theta}_{0,r}^{(j)}\right\|_2 \leq \frac{2T\eta\zeta}{\delta}\sqrt{\frac{n}{d_1}}\mathbb{E}_{[\mathcal{M}_{t-1}],\mathbf{W}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right]^{\frac{1}{2}}+\eta\zeta n\sqrt{\frac{8(m-1)T}{md_1\delta}}$$

with probability at least $1-\frac{\delta}{T}$

C.5 Convergence of GIST

We now prove the convergence result for GIST outlined in Appendix C.3. In showing the convergence of GIST, we care about the regression loss $\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2$ with

$$\hat{\mathbf{y}}(t)=f(\boldsymbol{\Theta}_t,\mathbf{X})=\frac{1}{m\sqrt{d_1}}\bar{\mathbf{A}}\sigma(\bar{\mathbf{A}}\mathbf{X}\boldsymbol{\Theta}_t)\mathbf{a}$$

As in previous work [32], we add the scaling factor $\frac{1}{m}$ to make sure that $\mathbb{E}_{\mathcal{M}_t}[\hat{\mathbf{y}}^{(j)}(t,0)]=\hat{\mathbf{y}}(t)$. Moreover, by properties of the masks $\mathcal{M}_t^{(j)}$, we have

$$f(\boldsymbol{\Theta},\mathbf{X})=\sum_{j=1}^{m}f_{\mathcal{M}}^{(j)}(\boldsymbol{\Theta},\mathbf{X})$$

Thus, we can invoke lemmas 13 and 14 from [32]. We state the two key lemmas here in accordance with our own notation. The t-th global step produces squared error satisfying

$$\|\mathbf{y}-\hat{\mathbf{y}}(t+1)\|_2^2=\frac{1}{m}\sum_{j=1}^{m}\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,\zeta)\|_2^2-\frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{j-1}\|\hat{\mathbf{y}}^{(j)}(t,\zeta)-\hat{\mathbf{y}}^{(j')}(t,\zeta)\|_2^2$$

In the t-th global iteration, the sampled subnetwork's deviation from the whole network is given by

$$\sum_{j=1}^{m}\|\hat{\mathbf{y}}(t)-\hat{\mathbf{y}}^{(j)}(t,0)\|_2^2=\frac{1}{m}\sum_{j=1}^{m}\sum_{j'=1}^{j-1}\|\hat{\mathbf{y}}^{(j)}(t,0)-\hat{\mathbf{y}}^{(j')}(t,0)\|_2^2$$

Moreover, lemmas 22 and 23 from [32] show that with probability at least $1-2n\exp(-\frac{m}{32})$, for all $R\leq\frac{1}{2}$, it holds that

$$\|\boldsymbol{\Theta}_0\|_F \leq \sqrt{2d_1d}-\sqrt{d_1}R$$
$$\sum_{r=1}^{m}\langle\boldsymbol{\theta}_{0,r},\hat{\mathbf{x}}_i\rangle \leq d_1n(2-R^2)$$

For convenience, we assume that such an initialization property holds. Then, we can use lemma 24 from [32]: as long as $\|\theta_{t,r}-\theta_{0,r}\|_2\leq R$ for all t,r, then we have

$$\mathbb{E}_{\mathcal{M}_t}\left[\|\hat{\mathbf{y}}(t)-\hat{\mathbf{y}}^{(j)}(t,0)\|_2^2\right]\leq\frac{4n(m-1)}{m^2}$$

Then, applying Markov's inequality gives the following with probability at least $1-\frac{\delta}{2mT}$

$$\|\hat{\mathbf{y}}(t)-\hat{\mathbf{y}}^{(j)}(t,0)\|_2^2\leq\frac{8n(m-1)T}{m\delta}$$

We point out that, within the proof, we use $R = \frac{\lambda_0}{96n}$, which satisfies the condition above. Using lemma C.5 to expand the loss at the $(t+1)$th iteration and invoking lemma C.4 gives

$$\begin{aligned}
\|\mathbf{y} - \hat{\mathbf{y}}(t+1)\|_2^2 &= \frac{1}{m}\sum_{j=1}^{m}\left\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,\zeta)\right\|_2^2 - \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{j-1}\left\|\hat{\mathbf{y}}^{(j)}(t,\zeta) - \hat{\mathbf{y}}^{(j')}(t,\zeta)\right\|_2^2 \\
&\leq \frac{1}{m}\sum_{j=1}^{m}\left(1 - \frac{\eta\lambda_0}{2}\right)^{\zeta}\left\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,0)\right\|_2^2 \\
&\quad - \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{j-1}\left\|\hat{\mathbf{y}}^{(j)}(t,\zeta) - \hat{\mathbf{y}}^{(j')}(t,\zeta)\right\|_2^2 \\
&= \frac{1}{m}\sum_{j=1}^{m}\left\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,0)\right\|_2^2 - \frac{\eta\lambda_0}{2m}\sum_{k=0}^{\zeta-1}\sum_{j=1}^{m}\left(1 - \frac{\eta\lambda_0}{2}\right)^{k}\left\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,0)\right\|_2^2 \\
&\quad - \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{j-1}\left\|\hat{\mathbf{y}}^{(j)}(t,\zeta) - \hat{\mathbf{y}}^{(j')}(t,\zeta)\right\|_2^2
\end{aligned}$$

Using the fact that $\mathbb{E}_{\mathcal{M}_t}[\hat{\mathbf{y}}^{(j)}(t,0)] = \hat{\mathbf{y}}(t)$ we have

$$\mathbb{E}_{\mathcal{M}_t}\left[\left\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,0)\right\|_2^2\right] = \|\mathbf{y} - \hat{\mathbf{y}}(t)\|_2^2 + \mathbb{E}_{\mathcal{M}_t}\left[\left\|\hat{\mathbf{y}}(t) - \hat{\mathbf{y}}^{(j)}(t,0)\right\|_2^2\right]$$

Then, using lemma C.5 to rewrite the last term in the equation above and plugging in gives

$$\begin{aligned}
&\mathbb{E}_{\mathcal{M}_t}\left[\|\mathbf{y} - \hat{\mathbf{y}}(t+1)\|_2^2\right] \\
&\quad \leq \|\mathbf{y} - \hat{\mathbf{y}}(t)\|_2^2 - \frac{\eta\lambda_0}{2m}\sum_{k=0}^{\zeta-1}\sum_{j=1}^{m}\left(1 - \frac{\eta\lambda_0}{2}\right)^{k}\mathbb{E}_{\mathcal{M}_t}\left[\left\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,0)\right\|_2^2\right] \\
&\qquad + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{j-1}\mathbb{E}_{\mathcal{M}_t}\left[\|\hat{\mathbf{y}}^{(j)}(t,0) - \hat{\mathbf{y}}^{(j')}(t,0)\|_2^2 - \left\|\hat{\mathbf{y}}^{(j)}(t,\zeta) - \hat{\mathbf{y}}^{(j')}(t,\zeta)\right\|_2^2\right]
\end{aligned}$$

We denote the last term within the equation above as ι_t

$$\iota_t = \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{j-1}\mathbb{E}_{\mathcal{M}_t}\left[\|\hat{\mathbf{y}}^{(j)}(t,0) - \hat{\mathbf{y}}^{(j')}(t,0)\|_2^2 - \left\|\hat{\mathbf{y}}^{(j)}(t,\zeta) - \hat{\mathbf{y}}^{(j')}(t,\zeta)\right\|_2^2\right]$$

The following lemma shows the bound on ι_t As long as $\left\|\boldsymbol{\theta}_{t,k,r}^{(j)} - \boldsymbol{\theta}_{0,r}\right\|_2 \leq R$ for all t, k, j, and the initialization satisfies $\|\boldsymbol{\Theta}_0\|_F \leq \sqrt{2d_1 d} - \sqrt{d_1}R$, then we have

$$\iota_t \leq \frac{\eta\gamma\lambda_0}{2m}\sum_{j=1}^{n}\sum_{k=0}^{\zeta-1}\mathbb{E}_{\mathcal{M}_t}\left[\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2^2\right] + \frac{64\eta(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1} + \iota_t'$$

with $\mathbb{E}_{\mathbf{\Theta}_0,\mathbf{a}_r}\left[\iota_t'\right] = 0$, for all $\gamma \in (0,1)$. Therefore, we can derive the following using lemma C.5

$$
\begin{aligned}
&\mathbb{E}_{\mathcal{M}_t}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t+1)\|_2^2\right] \\
&\quad \leq \|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2 - \frac{\eta\lambda_0}{2m}\sum_{k=0}^{\zeta-1}\sum_{j=1}^{m}\left(1-\frac{\eta\lambda_0}{2}\right)^k \mathbb{E}_{\mathcal{M}_t}\left[\left\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,0)\right\|_2^2\right] \\
&\qquad + \frac{\eta\gamma\lambda_0}{2m}\sum_{j=1}^{n}\sum_{k=0}^{\zeta-1}\mathbb{E}_{\mathcal{M}_t}\left[\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k)\|_2^2\right] \\
&\qquad + \frac{64\eta(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1} + \iota_t' \\
&\quad \leq \|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2 - \frac{(1-\gamma)\eta\lambda_0}{2m}\sum_{k=0}^{\zeta-1}\sum_{j=1}^{m}\left(1-\frac{\eta\lambda_0}{2}\right)^k \mathbb{E}_{\mathcal{M}_t}\left[\left\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,0)\right\|_2^2\right] \\
&\qquad + \frac{64\eta(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1} + \iota_t' \\
&\quad \leq \|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2 - \frac{(1-\gamma)\eta\lambda_0}{2}\sum_{k=0}^{\zeta-1}\left(1-\frac{\eta\lambda_0}{2}\right)^k \|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2 \\
&\qquad + \frac{64\eta(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1} + \iota_t' \\
&\quad = \left(\gamma + (1-\gamma)\left(1-\frac{\eta\lambda_0}{2}\right)^\zeta\right)\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2 \\
&\qquad + \frac{64\eta(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1} + \iota_t'
\end{aligned}
$$

Starting from here, we use α to denote the global convergence rate

$$\alpha = (1-\gamma)\left(1-\left(1-\frac{\eta\lambda_0}{2}\right)^\zeta\right)$$

Since $\zeta \geq 1$, we have that $\alpha \geq \frac{\eta\lambda_0}{2}(1-\gamma)$. Then, the convergence rate above yields the following

$$\mathbb{E}_{[\mathcal{M}_{t-1}],\mathbf{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right] \leq \mathbb{E}_{\mathbf{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(0)\|_2^2\right] + O\left(\frac{(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1}\right)$$

Lastly, we provide a bound on weight perturbation using overparameterization. In particular, we can show that hypothesis 5 holds for iteration $t+1$

$$\|\boldsymbol{\theta}_{t+1,r}-\boldsymbol{\theta}_{0,r}\|_2 + \frac{4T\eta\zeta}{\delta\alpha}\sqrt{\frac{n}{d_1}}\mathbb{E}_{[\mathcal{M}_t],\mathbf{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t+1)\|_2^2\right]^{\frac{1}{2}} + (T-t-1)B \leq R$$

under the assumption that it holds in iteration t

$$\|\boldsymbol{\theta}_{t,r}-\boldsymbol{\theta}_{0,r}\|_2 + \frac{4T\eta\zeta}{\delta\alpha}\sqrt{\frac{n}{d_1}}\mathbb{E}_{[\mathcal{M}_{t-1}],\mathbf{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right]^{\frac{1}{2}} + (T-t)B \leq R$$

and given the global convergence result

$$\mathbb{E}_{\mathcal{M}_t}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t+1)\|_2^2\right] \leq (1-\alpha)\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2 + \frac{64\eta(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1} + \iota_t'$$

Thus, it suffices to show that

$$
\begin{aligned}
&\|\boldsymbol{\theta}_{t+1,r}-\boldsymbol{\theta}_{0,r}\|_2 - \|\boldsymbol{\theta}_{t,r}-\boldsymbol{\theta}_{0,r}\|_2 \\
&\quad \leq \left(\mathbb{E}_{[\mathcal{M}_{t-1}],\mathbf{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right]^{\frac{1}{2}} - \mathbb{E}_{[\mathcal{M}_t],\mathbf{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t+1)\|_2^2\right]^{\frac{1}{2}}\right) \\
&\qquad \cdot \frac{4T\eta\zeta}{\delta\alpha}\sqrt{\frac{n}{d_1}} + B
\end{aligned}
$$

Using Jensen's inequality, we derive the following

$$\begin{aligned}
&\mathbb{E}_{[\mathcal{M}_t],\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t+1)\|_2^2\right]^{\frac{1}{2}} \\
&\quad\leq \left((1-\alpha)\,\mathbb{E}_{[\mathcal{M}_{t-1}],\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right] + \frac{64\eta(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1}\right)^{\frac{1}{2}} \\
&\quad\leq \left(1-\frac{\alpha}{2}\right)\mathbb{E}_{[\mathcal{M}_{t-1}],\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right]^{\frac{1}{2}} + \frac{8(m-1)}{m}\sqrt{\frac{\eta\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma d_1}}
\end{aligned}$$

It then suffices to show that

$$\begin{aligned}
\|\boldsymbol{\theta}_{t+1,r}-\boldsymbol{\theta}_{0,r}\|_2 - \|\boldsymbol{\theta}_{t,r}-\boldsymbol{\theta}_{0,r}\|_2 &\leq \frac{2T\eta\zeta}{\delta}\sqrt{\frac{n}{d_1}}\mathbb{E}_{[\mathcal{M}_t],\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t+1)\|_2^2\right]^{\frac{1}{2}} \\
&\qquad + B - \frac{8(m-1)}{m}\sqrt{\frac{\eta\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma d_1}} \\
&= \frac{2T\eta\zeta}{\delta}\sqrt{\frac{n}{d_1}}\mathbb{E}_{[\mathcal{M}_t],\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t+1)\|_2^2\right]^{\frac{1}{2}} \\
&\qquad + \eta\zeta n\sqrt{\frac{8(m-1)T}{md_1\delta}}
\end{aligned}$$

Fix $r \in [d_1]$ and let $\hat{j}$ be the index of the sub-GCN in which r is active. Indeed, we have

$$\begin{aligned}
\|\boldsymbol{\theta}_{t+1,r}-\boldsymbol{\theta}_{0,r}\|_2 &\leq \|\boldsymbol{\theta}_{t,r}-\boldsymbol{\theta}_{0,r}\|_2 + \|\boldsymbol{\theta}_{t+1,r}-\boldsymbol{\theta}_{0,r}\|_2 \\
&= \|\boldsymbol{\theta}_{t,r}-\boldsymbol{\theta}_{0,r}\|_2 + \|\boldsymbol{\theta}_{t,\zeta,r}^{(\hat{j})}-\boldsymbol{\theta}_{t,r}\|_2 \\
&\leq \|\boldsymbol{\theta}_{t,r}-\boldsymbol{\theta}_{0,r}\|_2 + \frac{2T\eta\zeta}{\delta}\sqrt{\frac{n}{d_1}}\mathbb{E}_{[\mathcal{M}_{t-1}],\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right]^{\frac{1}{2}} \\
&\quad + \eta\zeta n\sqrt{\frac{8(m-1)T}{md_1\delta}}
\end{aligned}$$

What remains is to prove hypothesis 5 for $t=0$. In that case, we need

$$\frac{4T\eta\zeta}{\delta\alpha}\sqrt{\frac{n}{d_1}}\mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(0)\|_2^2\right]^{\frac{1}{2}} + B \leq R = O\left(\frac{\lambda_0}{n}\right)$$

Finally, we have the following lemma bounding $\mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(0)\|_2^2\right]$ It holds that

$$\mathbb{E}\left[\|\mathbf{y}-\hat{\mathbf{y}}(0)\|_2^2\right] \leq C^2 n + \frac{d}{m^2}\|\bar{\mathbf{A}}^2\|_{1,1}$$

Thus, the bound above boils down to

$$\begin{aligned}
\frac{T\eta\zeta n}{\delta\alpha\sqrt{d_1}} &= O\left(\frac{\lambda_0}{n}\right) \\
\frac{T\eta\zeta}{\delta\alpha m}\sqrt{\frac{nd}{d_1}}\|\bar{\mathbf{A}}^2\|_{1,1}^{\frac{1}{2}} &= O\left(\frac{\lambda_0}{n}\right) \\
\frac{8(m-1)T}{m}\sqrt{\frac{\eta\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma d_1}} &= O\left(\frac{\lambda_0}{n}\right) \\
\eta\zeta n\sqrt{\frac{8(m-1)T^3}{md_1\delta}} &= O\left(\frac{\lambda_0}{n}\right)
\end{aligned}$$

Plugging in the value of B and using $\alpha \geq \frac{\eta\lambda_0}{2}(1-\gamma)$ to solve for d_1 gives

$$d_1 = \Omega\left(\frac{n^3\zeta^2T^2}{\delta^2\gamma(1-\gamma)^2\lambda_0^4}\left(n + \frac{d}{m^2}\|\bar{\mathbf{A}}^2\|_{1,1}\right)\right)$$

C.6 Proof of Lemmas

We now provide all proofs for the major properties and lemmas utilized in deriving the convergence results for GIST. [Proof of Property C.2] Under assumption 3, we have that for all $i, i' \in [n]$

$$\left(\frac{1-\epsilon}{1+\epsilon}\right)^2 \leq \frac{\mathbf{D}_{ii}}{\mathbf{D}_{i'i'}} \leq \left(\frac{1+\epsilon}{1-\epsilon}\right)^2$$

Therefore, we can write

$$\begin{aligned}
\|\hat{\mathbf{x}}_i\|_2 &= \left\| \sum_{i'=1}^{n} \bar{\mathbf{A}}_{ii'} \mathbf{x}_i \right\|_2 \\
&= \left\| \sum_{i'=1}^{n} \left(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}\right)_{ii'} \mathbf{x}_i \right\|_2 \\
&= \left\| \mathbf{x}_i + \mathbf{D}_{ii}^{-\frac{1}{2}} \sum_{i' \neq i} \mathbf{A}_{ii'} \mathbf{D}_{i'i'}^{-\frac{1}{2}} \mathbf{x}_i \right\|_2 \\
&\leq \frac{1-\epsilon}{2} + \mathbf{D}_{ii}^{-\frac{1}{2}} \sum_{i' \neq i} \mathbf{A}_{ii'} \mathbf{D}_{i'i'}^{-\frac{1}{2}} \left(\frac{1-\epsilon}{2}\right) \\
&\leq \frac{1-\epsilon}{2} + \mathbf{D}_{ii}^{-\frac{1}{2}} \sum_{i' \neq i} \mathbf{A}_{ii'} \mathbf{D}_{ii}^{-\frac{1}{2}} \left(\frac{1+\epsilon}{1-\epsilon}\right) \left(\frac{1-\epsilon}{2}\right) \\
&= 1
\end{aligned}$$

where the first inequality follows from assumption 4.

[Proof of Lemma C.4] Fix some $R > 0$. Following Theorem 2 by [32], we have that with probability at least $1 - 2n^2 e^{-2d_1 t^2}$ it holds that

$$\|\mathbf{H}^{(j)}(t, 0, 0) - \mathbf{H}^{\infty}\|_2 \leq nt$$

and with probability at least $1 - n^2 e^{-\frac{d_1 R}{10m}}$ it holds that

$$\|\mathbf{H}^{(j)}(t, t', k) - \mathbf{H}^{(j)}(k, 0, 0)\|_2 \leq \frac{3nR}{m}$$

Choosing $t = \frac{\lambda_0}{16n}$ and $R = \frac{\lambda_0}{48n}$ gives

$$\|\mathbf{G}^{(j)}(t, t', k) - \mathbf{G}^{\infty}\|_2 \leq \|\bar{\mathbf{A}}\|^2 \|\mathbf{H}^{(j)}(t, t', k) - \mathbf{H}^{\infty}\|_2 = \|\bar{\mathbf{A}}\|^2 \cdot \frac{\lambda_0}{8} \leq \frac{\lambda_0}{2}$$

with probability at least $1 - n^2 \left(2\exp\left(-\frac{d_1 \lambda_0^2}{128n^2}\right) + \exp\left(-\frac{d_1 \lambda_0}{480mn}\right)\right)$. Taking a union bound over all values of t' and j, then plugging in the requirement $d_1 = \Omega\left(\lambda_0^{-1} n^2 \log {}^{Tmn}/_{\delta}\right)$ gives the desired result.

[Proof of Lemma C.4] We first bound the norm of the gradient as

$$\begin{aligned}
\left\| \frac{\partial L(\mathbf{\Theta}_{t,k}^{(j)})}{\partial \boldsymbol{\theta}_r} \right\|_2 &\leq \frac{1}{\sqrt{d_1}} \sum_{i=1}^{n} \sum_{i'=1}^{n} \bar{\mathbf{A}}_{ii'} \mid \hat{\mathbf{y}}_i^{(j)}(t, k) - \mathbf{y}_i \mid \\
&= \frac{1}{\sqrt{d_1}} \|\bar{\mathbf{A}} \Delta\|_1 \\
&\leq \sqrt{\frac{n}{d_1}} \|\bar{\mathbf{A}}\| \|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t, k)\|_2
\end{aligned}$$

where here $\Delta = \left[| \hat{\mathbf{y}}_1^{(j)}(t,k) - \mathbf{y}_1 |, \ldots, | \hat{\mathbf{y}}_n^{(j)}(t,k) - \mathbf{y}_n | \right]$, and for the last inequality we use $\|\Delta\|_2 = \|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2$. Then, following [46], we first fix $R = \frac{\lambda_0}{\|\bar{\mathbf{A}}\|^2}$, and denote

$$\begin{aligned} S_i &= \{r \in [m] : \neg A_{ir}\} \\ A_{ir} &= \{\exists \boldsymbol{\theta} : \|\boldsymbol{\theta} - \boldsymbol{\theta}_{0,r}\|_2 \leq R, \mathbb{I}\{\langle \boldsymbol{\theta}, \hat{\mathbf{x}}_i \rangle \geq 0\} \neq \mathbb{I}\{\langle \boldsymbol{\theta}_{0,r}, \hat{\mathbf{x}}_i \rangle \geq 0\}\} \\ S_i^{\perp} &= [m] \setminus S_i \\ \hat{s} &= \max_{i \in [n]} | S_i^{\perp} | \end{aligned}$$

Lemma 16 from [32] shows that

$$\mathcal{P}\left(| S_i^{\perp} | \leq 4 d_1 R \right) \geq \exp(-d_1 R)$$

Throughout the proof, we let $\hat{s} = 4 d_1 R$. Moreover, we define

$$\mathbf{H}^{\perp}(t,t',k) = \frac{1}{d} \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_{i'} \rangle \sum_{r \in S_i^{\perp}} \mathcal{M}_{t,r}^{(j)} \mathbb{I}\{\left\langle \boldsymbol{\theta}_{t',k,r}^{(j)}, \hat{\mathbf{x}}_i \right\rangle \geq 0; \left\langle \boldsymbol{\theta}_{t',k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \geq 0\}$$

and let $\mathbf{G}^{\perp}(t,t',k) = \bar{\mathbf{A}} \mathbf{H}^{\perp}(t,t',k) \bar{\mathbf{A}}$. Then, we have

$$\begin{aligned} \left\| \mathbf{H}^{\perp}(t,t',k) \right\|_2^2 &\leq \left\| \mathbf{H}^{\perp}(t,t',k) \right\|_F^2 \\ &\leq \frac{1}{d_1^2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \sum_{r \in S_i^{\perp}} \sum_{r' \in S_i^{\perp}} \mathbb{I}\left\{ \left\langle \boldsymbol{\theta}_{t',k,r}^{(j)}, \hat{\mathbf{x}}_i \right\rangle \geq 0; \left\langle \boldsymbol{\theta}_{t',k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \geq 0 \right\} \cdot \\ &\qquad \mathbb{I}\left\{ \left\langle \boldsymbol{\theta}_{t',k,r'}^{(j)}, \hat{\mathbf{x}}_i \right\rangle \geq 0; \left\langle \boldsymbol{\theta}_{t',k,r'}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \geq 0 \right\} \\ &\leq \frac{n^2 \hat{s}^2}{d_1^2} = 16 n^2 R^2 \end{aligned}$$

which yields the following

$$\left\| \mathbf{G}^{(j)\perp}(t,t',k) \right\| \leq \|\bar{\mathbf{A}}\|^2 \|\mathbf{H}^{(j)\perp}(t,t',k)\| = 16 n R$$

We then expand the loss at iteration $(t, k+1)$ as

$$\begin{aligned} \left\| \mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k+1) \right\|_2^2 = &\left\| \mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k) \right\|_2^2 \\ &- 2 \left\langle \mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k), \hat{\mathbf{y}}^{(j)}(t,k+1) - \hat{\mathbf{y}}^{(j)}(t,k) \right\rangle \\ &+ \left\| \hat{\mathbf{y}}^{(j)}(t,k+1) - \hat{\mathbf{y}}^{(j)}(t,k) \right\|_2^2 \end{aligned}$$

Starting to analyze the second term, we note that

$$\begin{aligned} &\hat{y}_i^{(j)}(t,k+1) - \hat{y}_i^{(j)}(t,k) \\ &\quad = \frac{1}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r=1}^{d_1} \bar{\mathbf{A}}_{ii'} \mathcal{M}_{t,r}^{(j)} a_r \left(\sigma\left(\left\langle \boldsymbol{\theta}_{t,k+1,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \right) - \sigma\left(\left\langle \boldsymbol{\theta}_{t,k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \right) \right) \end{aligned}$$

We decompose $\hat{y}_i^{(j)}(t,k+1) - \hat{y}_i^{(j)}(t,k) = I_{t,k,1}^{(j)} + I_{t,k,2}^{(j)}$ with

$$I_{i,1}^{(j)}(t,k) = \frac{1}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r \in S_{i'}} \bar{\mathbf{A}}_{ii'} \mathcal{M}_{t,r}^{(j)} a_r \left(\sigma\left(\left\langle \boldsymbol{\theta}_{t,k+1,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \right) - \sigma\left(\left\langle \boldsymbol{\theta}_{t,k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \right) \right)$$

$$I_{i,2}^{(j)}(t,k) = \frac{1}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r \in S_{i'}^{\perp}} \bar{\mathbf{A}}_{ii'} \mathcal{M}_{t,r}^{(j)} a_r \left(\sigma\left(\left\langle \boldsymbol{\theta}_{t,k+1,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \right) - \sigma\left(\left\langle \boldsymbol{\theta}_{t,k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \right) \right)$$

where $I_{i,1}^{(j)}(t,k)$ can be further written as

$$\begin{aligned}
I_{i,1}^{(j)}(t,k) &= \frac{1}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r \in S_{i'}} \bar{\mathbf{A}}_{ii'} \mathcal{M}_{t,r}^{(j)} a_r \left\langle \boldsymbol{\theta}_{t,k+1,r}^{(j)} - \boldsymbol{\theta}_{t,k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \mathbb{I}\left\{ \left\langle \boldsymbol{\theta}_{t,k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \geq 0 \right\} \\
&= -\frac{\eta}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r \in S_{i'}} \bar{\mathbf{A}}_{ii'} \mathcal{M}_{t,r}^{(j)} a_r \left\langle \frac{\partial L\left(\boldsymbol{\Theta}_{t,k}^{(j)}\right)}{\partial \boldsymbol{\theta}_r}, \hat{\mathbf{x}}_{i'} \right\rangle \mathbb{I}\left\{ \left\langle \boldsymbol{\theta}_{t,k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \geq 0 \right\} \\
&= \frac{\eta}{d_1} \sum_{i'=1}^{n} \sum_{i_1=1}^{n} \sum_{i_1'=1}^{n} \sum_{r \in S_i} \bar{\mathbf{A}}_{ii'} \bar{\mathbf{A}}_{i_1 i_1'} \mathcal{M}_{t,r}^{(j)} \left(y_{i_1} - \hat{y}_{i_1}^{(j)}(t,k) \right) \left\langle \hat{\mathbf{x}}_{i_1'}, \hat{\mathbf{x}}_{i'} \right\rangle \cdot \\
&\qquad \mathbb{I}\left\{ \left\langle \boldsymbol{\theta}_{t,k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \geq 0; \left\langle \boldsymbol{\theta}_{t,k,r}^{(j)}, \hat{\mathbf{x}}_{i_1'} \right\rangle \geq 0 \right\} \\
&= \eta \sum_{i'=1}^{n} \sum_{i_1=1}^{n} \sum_{i_1'=1}^{n} \bar{\mathbf{A}}_{ii'} \bar{\mathbf{A}}_{i_1 i_1'} \left(y_{i_1} - \hat{y}_{i_1}^{(j)}(t,k) \right) \left(\mathbf{H}^{(j)}(t,t,k)_{i' i_1'} - \mathbf{H}^{(j)\perp}(t,t,k)_{i' i_1'} \right)
\end{aligned}$$

Thus for $\mathbf{I}_{i,1}^{(j)}(t,k) = [I_{1,1}^{(j)}(t,k), \ldots, I_{n,1}^{(j)}(t,k)]$ we have

$$\begin{aligned}
\mathbf{I}_{i,1}^{(j)}(t,k) &= \eta \bar{\mathbf{A}} \left(\mathbf{H}^{(j)}(t,t,k) - \mathbf{H}^{(j)\perp}(t,t,k) \right) \bar{\mathbf{A}} \left(\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k) \right) \\
&= \eta \left(\mathbf{G}^{(j)}(t,t,k) - \mathbf{G}^{(j)\perp}(t,t,k) \right) \left(\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k) \right) \\
&\geq \eta \left(\frac{\lambda_0}{2} - \|\mathbf{G}^{(j)\perp}(t,t,k)\|_2 \right) \left\| \mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k) \right\|_2^2
\end{aligned}$$

For $I_{i,2}^{(j)}(t,k)$ we have

$$\begin{aligned}
\mid I_{i,2}^{(j)}(t,k) \mid &\leq \frac{1}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r \in S_{i'}^{\perp}} \bar{\mathbf{A}}_{ii'} \mid \sigma\left(\left\langle \boldsymbol{\theta}_{t,k+1,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \right) - \sigma\left(\left\langle \boldsymbol{\theta}_{t,k,r}^{(j)}, \hat{\mathbf{x}}_{i'} \right\rangle \right) \mid \\
&\leq \frac{\eta}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r \in S_{i'}^{\perp}} \bar{\mathbf{A}}_{ii'} \left\| \frac{\partial L(\boldsymbol{\Theta}_{t,k}^{(j)})}{\partial \boldsymbol{\theta}_r} \right\|_2 \\
&\leq \frac{\eta \hat{s}}{d_1} \|\bar{\mathbf{A}}\| \|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2 \sum_{i'=1}^{n} \bar{\mathbf{A}}_{ii'}
\end{aligned}$$

which yields the following

$$\begin{aligned}
\mid \left\langle \mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k), \mathbf{I}_{i,2}^{(j)}(t,k) \right\rangle \mid &\leq \sum_{i=1}^{n} \mid y_i - y_i^{(j)}(t,k) \mid \cdot \mid I_{i,2}^{(j)}(t,k) \mid \\
&\leq \frac{\eta \hat{s}}{d_1} \|\bar{\mathbf{A}}\| \|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2 \sum_{i,i'=1}^{n} \bar{\mathbf{A}}_{ii'} \mid y_i - y_i^{(j)}(t,k) \mid \\
&\leq \frac{\eta \hat{s}}{d_1} \|\bar{\mathbf{A}}\|^2 \left\| \mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k) \right\|_2^2 \\
&\leq \frac{4 \eta \hat{s}}{d_1} \left\| \mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k) \right\|_2^2
\end{aligned}$$

Lastly, we have

$$\begin{aligned}
\left(y_i - \hat{y}_i^{(j)}(t,k) \right)^2 &= \frac{1}{d_1} \sum_{i'=1}^{n} \sum_{i''=1}^{n} \sum_{r=1}^{d_1} \sum_{r'=1}^{d_1} \bar{\mathbf{A}}_{ii'} \bar{\mathbf{A}}_{ii''} \left\| \frac{\partial L(\boldsymbol{\Theta}_{t,k}^{(j)})}{\partial \boldsymbol{\theta}_r} \right\|_2 \cdot \left\| \frac{\partial L(\boldsymbol{\Theta}_{t,k}^{(j)})}{\partial \boldsymbol{\theta}_{r'}} \right\|_2 \\
&\leq \eta^2 \|\bar{\mathbf{A}}\|^2 \|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2^2 \sum_{i'=1}^{n} \sum_{i''=1}^{n} \bar{\mathbf{A}}_{ii'} \bar{\mathbf{A}}_{ii''}
\end{aligned}$$

Therefore

$$\begin{aligned}\left\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k)\right\|_2^2 &= \sum_{i=1}^{n}\left(y_i-\hat{y}_i^{(j)}(t,k)\right)^2\\ &= \eta^2\|\bar{\mathbf{A}}\|^2\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k)\|_2^2\sum_{i=1}^{n}\sum_{i'=1}^{n}\sum_{i''=1}^{n}\bar{\mathbf{A}}_{ii'}\bar{\mathbf{A}}_{ii''}\\ &= 4\eta^2\|\bar{\mathbf{A}}^2\|_{1,1}\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k)\|_2^2\end{aligned}$$

Putting things together gives

$$\begin{aligned}\left\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k+1)\right\|_2^2 &\le \eta\left(2\|\mathbf{G}^{(j)\perp}(t,t,k)\|_2+\frac{8\eta\hat{s}}{d_1}+4\eta n^2\|\bar{\mathbf{A}}^2\|_{1,1}-\lambda_0\right)\\ &\quad\cdot\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k)\|_2^2+\left\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k)\right\|_2^2\\ &\le\left(1-\frac{\eta\lambda_0}{2}\right)\left\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k)\right\|_2^2\end{aligned}$$

where the last step follows by plugging in the values of $\|\mathbf{G}^{(j)\perp}(t,t,k)\|$ and $\hat{s}$, then setting $R=\frac{\lambda_0}{96n}$ and $\eta\le\frac{\lambda_0}{n^2\|\bar{\mathbf{A}}^2\|_{1,1}}$. Next, we bound the weight perturbation. First, using Markov's inequality, we have that with probability at least $1-\frac{\delta}{2T}$

$$\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\le\frac{2T}{\delta}\mathbb{E}_{[\mathcal{M}_{t-1}]}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right]$$

Thus, we have

$$\begin{aligned}\left\|\boldsymbol{\theta}_{t,k,r}^{(j)}-\boldsymbol{\theta}_{t,0,r}^{(j)}\right\|_2 &\le\sum_{k'=0}^{k-1}\|\boldsymbol{\theta}_{t,k+1,r}-\boldsymbol{\theta}_{t,k,r}\|_2\\ &\le\eta\sum_{k'=0}^{k-1}\left\|\frac{\partial L\left(\boldsymbol{\Theta}_{t,k}^{(j)}\right)}{\partial\boldsymbol{\theta}_r}\right\|_2\\ &\le\eta\sqrt{\frac{n}{d_1}}\sum_{k'=0}^{k-1}\|\mathbf{y}-\hat{\mathbf{y}}^{(j)}(t,k)\|_2\\ &\le\eta\zeta\sqrt{\frac{n}{d_1}}\left(\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2+\|\hat{\mathbf{y}}(t)-\hat{\mathbf{y}}^{(j)}(t,0)\|_2\right)\\ &\le\frac{2T\eta\zeta}{\delta}\sqrt{\frac{n}{d_1}}\mathbb{E}_{[\mathcal{M}_{t-1}]}\left[\|\mathbf{y}-\hat{\mathbf{y}}(t)\|_2^2\right]^{\frac{1}{2}}+\eta\zeta n\sqrt{\frac{8(m-1)T}{md_1\delta}}\end{aligned}$$

Since $\alpha<1<2$, we have that

$$\frac{2T\eta\zeta}{\delta}\sqrt{\frac{n}{d_1}}\le\frac{4T\eta\zeta}{\delta\alpha}\sqrt{\frac{n}{d_1}}$$

Also we have

$$\eta\zeta n\sqrt{\frac{8(m-1)T}{md_1\delta}}\le B$$

Therefore, by hypothesis 5, we have that

$$\left\|\boldsymbol{\theta}_{t,k,r}^{(j)}-\boldsymbol{\theta}_{0,r}^{(j)}\right\|_2\le\left\|\boldsymbol{\theta}_{t,r}^{(j)}-\boldsymbol{\theta}_{0,r}^{(j)}\right\|_2+\left\|\boldsymbol{\theta}_{t,k,r}^{(j)}-\boldsymbol{\theta}_{t,0,r}^{(j)}\right\|_2\le R$$

[Proof of Lemma (C.5)] For convenience, we denote

$$\sigma_i^{(j)}(t,r)=\sigma\left(\langle\boldsymbol{\theta}_{t,r},\hat{\mathbf{x}}_i\rangle\right);\quad\sigma_i^{(j)}(t+1,r)=\sigma\left(\left\langle\boldsymbol{\theta}_{t,\zeta,r}^{(j)},\hat{\mathbf{x}}_i\right\rangle\right)$$

Using 1-Lipschitzness of ReLU, we have

$$\begin{aligned}
\mid \sigma_i^{(j)}(t,r) - \sigma_i^{(j)}(t+1,r) \mid &\leq \left\| \boldsymbol{\theta}_{t,r} - \boldsymbol{\theta}_{t,\zeta,r}^{(j)} \right\|_2 \\
&\leq \eta \sum_{k=0}^{\zeta-1} \left\| \frac{\partial L(\boldsymbol{\Theta}_{t,k}^{(j)})}{\partial \boldsymbol{\theta}_r} \right\|_2 \\
&\leq 2\sqrt{\frac{n}{d_1}} \sum_{k=0}^{\zeta-1} \|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2
\end{aligned}$$

Also, we have

$$\begin{aligned}
\mid \sigma_i^{(j)}(t,r) + \sigma_i^{(j)}(t+1,r) \mid &\leq \left\| \boldsymbol{\theta}_{t,r} + \boldsymbol{\theta}_{t,\zeta,r}^{(j)} \right\|_2 \\
&\leq 2\|\boldsymbol{\theta}_{0,r}\|_2 + \|\boldsymbol{\theta}_{t,r} - \boldsymbol{\theta}_{0,r}\|_2 + \left\| \boldsymbol{\theta}_{t,\zeta,r}^{(j)} - \boldsymbol{\theta}_{0,r} \right\|_2 \\
&\leq 2\|\boldsymbol{\theta}_{0,r}\|_2 + 2R
\end{aligned}$$

Expanding the difference of squares gives

$$\left(\hat{y}_i^{(j)}(t,0) - \hat{y}_i^{(j')}(t,0)\right)^2 - \left(\hat{y}_i^{(j)}(t,\zeta) - \hat{y}_i^{(j')}(t,\zeta)\right)^2 = \beta_{i,1}\beta_{i,2}$$

with

$$\begin{aligned}
\beta_{i,1}^{(j,j')} &= \hat{y}_i^{(j)}(t,0) - \hat{y}_i^{(j')}(t,0) + \hat{y}_i^{(j)}(t,\zeta) - \hat{y}_i^{(j')}(t,\zeta) \\
\beta_{i,2}^{(j,j')} &= \hat{y}_i^{(j)}(t,0) - \hat{y}_i^{(j')}(t,0) - \hat{y}_i^{(j)}(t,\zeta) + \hat{y}_i^{(j')}(t,\zeta)
\end{aligned}$$

Written in terms of the simplified notation, we have

$$\begin{aligned}
\beta_{i,1}^{(j,j')} = \frac{1}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r=1}^{d_1} \bar{\mathbf{A}}_{ii'} a_r \Bigg(&\mathcal{M}_{t,r}^{(j)} \left(\sigma_{i'}^{(j)}(t,r) + \sigma_{i'}^{(j)}(t+1,r)\right) \\
&- \mathcal{M}_{t,r}^{(j')} \left(\sigma_{i'}^{(j')}(t,r) + \sigma_{i'}^{(j')}(t+1,r)\right) \Bigg) \\
\beta_{i,2}^{(j,j')} = \frac{1}{\sqrt{d_1}} \sum_{i'=1}^{n} \sum_{r=1}^{d_1} \bar{\mathbf{A}}_{ii'} a_r \Bigg(&\mathcal{M}_{t,r}^{(j)} \left(\sigma_{i'}^{(j)}(t,r) - \sigma_{i'}^{(j)}(t+1,r)\right) \\
&- \mathcal{M}_{t,r}^{(j')} \left(\sigma_{i'}^{(j')}(t,r) - \sigma_{i'}^{(j')}(t+1,r)\right) \Bigg)
\end{aligned}$$

Letting

$$\begin{aligned}
\tau_1(i',r) &= \mathcal{M}_{t,r}^{(j)} \left(\sigma_{i'}^{(j)}(t,r) + \sigma_{i'}^{(j)}(t+1,r)\right) - \mathcal{M}_{t,r}^{(j')} \left(\sigma_{i'}^{(j')}(t,r) + \sigma_{i'}^{(j')}(t+1,r)\right) \\
\tau_2(i',r) &= \mathcal{M}_{t,r}^{(j)} \left(\sigma_{i'}^{(j)}(t,r) - \sigma_{i'}^{(j)}(t+1,r)\right) - \mathcal{M}_{t,r}^{(j')} \left(\sigma_{i'}^{(j')}(t,r) - \sigma_{i'}^{(j')}(t+1,r)\right)
\end{aligned}$$

Then we have

$$\begin{aligned}
\beta_{i,1}^{(j,j')}\beta_{i,2}^{(j,j')} &= \frac{1}{d_1} \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \sum_{r=1}^{d_1} \sum_{r'=1}^{d_1} \bar{\mathbf{A}}_{ii_1} \bar{\mathbf{A}}_{ii_2} a_r a_{r'} \tau_1(i_1,r)\tau_2(i_2,r') \\
&= \frac{1}{d_1} \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \sum_{r=1}^{d_1} \bar{\mathbf{A}}_{ii_1} \bar{\mathbf{A}}_{ii_2} \tau_1(i_1,r)\tau_2(i_2,r) + \Delta_{i,t}^{(j,j')}
\end{aligned}$$

with

$$\Delta_{i,t}^{(j,j')} = \frac{1}{d_1} \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \sum_{r=1}^{d_1} \sum_{r'\neq r} \bar{\mathbf{A}}_{ii_1} \bar{\mathbf{A}}_{ii_2} a_r a_{r'} \tau_1(i_1,r)\tau_2(i_2,r')$$

Note that due to the independence of a_r and $a_{r'}$, we have that $\mathbb{E}_{\mathbf{a}}\left[\Delta_{i,t}^{(j,j')}\right] = 0$. Moreover, for $j \neq j'$, we have that $\mathcal{M}_{t,r}^{(j)}\mathcal{M})_{t,r}^{(j')} = 0$. Thus, we have

$$\begin{aligned}
&\mid \tau(i_1, r)\tau_2(i_2, r') \mid \\
&\quad = \mathcal{M}_{t,r}^{(j)} \mid \sigma_{i_1}^{(j)}(t,r) + \sigma_{i_1}^{(j)}(t+1,r) \mid \cdot \mid \sigma_{i_2}^{(j)}(t,r) - \sigma_{i_2}^{(j)}(t+1,r) \mid \\
&\qquad + \mathcal{M}_{t,r}^{(j')} \mid \sigma_{i_1}^{(j')}(t,r) + \sigma_{i_1}^{(j')}(t+1,r) \mid \cdot \mid \sigma_{i_2}^{(j')}(t,r) - \sigma_{i_2}^{(j')}(t+1,r) \mid \\
&\quad \leq 4\eta\sqrt{\frac{n}{d_1}}\left(\|\boldsymbol{\theta}_{0,r}\|_2 + 2R\right)\left(\sum_{k=0}^{\zeta-1}\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2 + \sum_{k=0}^{\zeta-1}\|\mathbf{y} - \hat{\mathbf{y}}^{(j')}(t,k)\|_2\right)
\end{aligned}$$

Thus

$$\begin{aligned}
&\beta_{i,1}^{(j,j')}\beta_{i,2}^{(j,j')} \\
&\quad = \frac{4\eta\sqrt{n}}{d_1^{\frac{3}{2}}}\left(\sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\bar{\mathbf{A}}_{ii_1}\bar{\mathbf{A}}_{ii_2}\right)\left(\sum_{r=1}^{d_1}\left(\|\boldsymbol{\theta}_{0,r}\|_2 + 2R\right)\right)\cdot \\
&\qquad \left(\sum_{k=0}^{\zeta-1}\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2 + \sum_{k=0}^{\zeta-1}\|\mathbf{y} - \hat{\mathbf{y}}^{(j')}(t,k)\|_2\right) + \Delta_{i,t}^{(j,j')} \\
&\quad \leq 4\eta\sqrt{\frac{2nd}{d_1}}\left(\sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\bar{\mathbf{A}}_{ii_1}\bar{\mathbf{A}}_{ii_2}\right)\left(\sum_{k=0}^{\zeta-1}\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2 + \sum_{k=0}^{\zeta-1}\|\mathbf{y} - \hat{\mathbf{y}}^{(j')}(t,k)\|_2\right) \\
&\qquad + \Delta_{i,t}^{(j,j')}
\end{aligned}$$

Thus

$$\begin{aligned}
\iota_t &= \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{j-1}\mathbb{E}_{\mathcal{M}_t}\left[\sum_{i=1}^{n}\beta_{i,1}^{(j,j')}\beta_{i,2}^{(j,j')}\right] \\
&= \frac{4\eta(m-1)}{m^2}\sqrt{\frac{2nd}{d_1}}\|\bar{\mathbf{A}}^2\|_{1,1}\sum_{j=1}^{m}\sum_{k=0}^{\zeta-1}\mathbb{E}_{\mathcal{M}_t}\left[\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2\right] \\
&\quad + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{j-1}\sum_{i=1}^{n}\mathbb{E}_{\mathcal{M}_t^{(j)}}\left[\Delta_{i,t}^{(j,j')}\right] \\
&= \sum_{j=1}^{n}\sum_{k=0}^{\zeta-1}\mathbb{E}_{\mathcal{M}_t}\left[\alpha_{t,k}^{(j)}\right] + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{j-1}\sum_{i=1}^{n}\mathbb{E}_{\mathcal{M}_t^{(j)}}\left[\Delta_{i,t}^{(j,j')}\right]
\end{aligned}$$

with

$$\alpha_{t,k}^{(j)} = \frac{4\eta(m-1)}{m^2}\sqrt{\frac{2nd}{d_1}}\|\bar{\mathbf{A}}^2\|_{1,1}\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2$$

By Cauchy-Schwartz inequality,

$$\begin{aligned}
\alpha_{t,k}^{(j)} &= \frac{\eta}{m}\cdot\left(\sqrt{\frac{\gamma\lambda_0}{2}}\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2\right)\cdot\left(\frac{8(m-1)}{m\sqrt{\gamma}}\|\bar{\mathbf{A}}\|_{1,1}\sqrt{\frac{nd}{d_1}}\right) \\
&\leq \frac{\eta}{m}\left(\frac{\gamma\lambda_0}{2}\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2^2 + \frac{64(m-1)^2\|\bar{\mathbf{A}}\|_{1,1}^2 nd}{\gamma m^2 d_1}\right) \\
&= \frac{\eta\gamma\lambda_0}{2m}\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2^2 + \frac{64\eta(m-1)^2\|\bar{\mathbf{A}}\|_{1,1}^2 nd}{\gamma m^3 d_1}
\end{aligned}$$

Thus

$$\iota_t \leq \frac{\eta\gamma\lambda_0}{2m}\sum_{j=1}^{m}\sum_{k=0}^{\zeta-1}\mathbb{E}_{\mathcal{M}_t}\left[\|\mathbf{y} - \hat{\mathbf{y}}^{(j)}(t,k)\|_2^2\right] + \frac{64\eta(m-1)^2\zeta\|\bar{\mathbf{A}}\|_{1,1}^2 nd}{\gamma m^2 d_1} + \iota_t'$$

with

$$\mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\iota_t'\right] = 0$$

[Proof of Lemma C.5] Note that

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(0)\|_2^2\right] &= \sum_{i=1}^{n}\mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[(y_i-\hat{y}_i(0))^2\right] \\ &= y_i^2 - 2y_i\mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\hat{y}_i(0)\right] + \mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\hat{y}_i(0)^2\right] \\ &\leq C^2 + \mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\hat{y}_i(0)^2\right]\end{aligned}$$

where the last inequality follows from the bound on $|y_i|$ and the fact that $\mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\hat{y}_i(0)^2\right] = 0$. Moreover, we have

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\hat{y}_i(0)^2\right] \\ &\quad= \frac{1}{m^2d_1}\sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\sum_{r_1=1}^{d_1}\sum_{r_2=1}^{d_1}\bar{\mathbf{A}}_{ii_1}\bar{\mathbf{A}}_{ii_2}\mathbb{E}_{\mathbf{a}}[a_{r_1}a_{r_2}]\mathbb{E}_{\boldsymbol{\Theta}_0}\left[\sigma\left(\langle\boldsymbol{\theta}_{0,r_1},\hat{\mathbf{x}}_{i_1}\rangle\right)\sigma\left(\langle\boldsymbol{\theta}_{0,r_2},\hat{\mathbf{x}}_{i_2}\rangle\right)\right] \\ &\quad= \frac{1}{m^2d_1}\sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\sum_{r=1}^{d_1}\bar{\mathbf{A}}_{ii_1}\bar{\mathbf{A}}_{ii_2}\mathbb{E}_{\boldsymbol{\Theta}_0}\left[\sigma\left(\langle\boldsymbol{\theta}_{0,r},\hat{\mathbf{x}}_{i_1}\rangle\right)\sigma\left(\langle\boldsymbol{\theta}_{0,r},\hat{\mathbf{x}}_{i_2}\rangle\right)\right] \\ &\quad\leq \frac{1}{m^2d_1}\sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\sum_{r=1}^{d_1}\bar{\mathbf{A}}_{ii_1}\bar{\mathbf{A}}_{ii_2}\mathbb{E}_{\boldsymbol{\Theta}_0}\left[\|\boldsymbol{\theta}_{0,r}\|_2^2\right] \\ &\quad= \frac{d}{m^2}\sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\bar{\mathbf{A}}_{ii_1}\bar{\mathbf{A}}_{ii_2}\end{aligned}$$

Thus

$$\mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y}-\hat{\mathbf{y}}(0)\|_2^2\right] \leq \sum_{i=1}^{n}\mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\hat{y}_i(0)^2\right] \leq C^2n + \frac{d}{m^2}\|\bar{\mathbf{A}}^2\|_{1,1}$$

Supplementary information All code for this project is publicly-available via github at the following link: `https://github.com/wolfecameron/GIST`