AdaGMLP: AdaBoosting GNN-to-MLP Knowledge Distillation

Weigang Lu
wglu@stu.xidian.edu.cn
Xidian University
Xi'an, China

Ziyu Guan*
ziyuguan@xidian.edu.cn
Xidian University
Xi'an, China

Wei Zhao
ywzhao@mail.xidian.edu.cn
Xidian University
Xi'an, China

Yaming Yang
yym@xidian.edu.cn
Xidian University
Xi'an, China

Abstract

Graph Neural Networks (GNNs) have revolutionized graph-based machine learning, but their heavy computational demands pose challenges for latency-sensitive edge devices in practical industrial applications. In response, a new wave of methods, collectively known as GNN-to-MLP Knowledge Distillation, has emerged. They aim to transfer GNN-learned knowledge to a more efficient MLP student, which offers faster, resource-efficient inference while maintaining competitive performance compared to GNNs. However, these methods face significant challenges in situations with insufficient training data and incomplete test data, limiting their applicability in real-world applications. To address these challenges, we propose AdaGMLP, an AdaBoosting GNN-to-MLP Knowledge Distillation framework. It leverages an ensemble of diverse MLP students trained on different subsets of labeled nodes, addressing the issue of insufficient training data. Additionally, it incorporates a Node Alignment technique for robust predictions on test data with missing or incomplete features. Our experiments on seven benchmark datasets with different settings demonstrate that AdaGMLP outperforms existing G2M methods, making it suitable for a wide range of latency-sensitive real-world applications. We have submitted our code to the GitHub repository (https://github.com/WeigangLu/AdaGMLP-KDD24).

CCS Concepts

• **Computing methodologies → Machine learning**; • **Networks → Network algorithms**.

Keywords

Graph Neural Networks, Knowledge Distillation, GNN-to-MLP Knowledge Distillation

ACM Reference Format:
Weigang Lu, Ziyu Guan, Wei Zhao, and Yaming Yang. 2024. AdaGMLP: AdaBoosting GNN-to-MLP Knowledge Distillation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671699

1 Introduction

Graph Neural Networks (GNNs) [8, 13, 14, 16, 18, 25, 27, 33] have revolutionized the field of graph-based machine learning, enabling state-of-the-art performance in various domains, including social networks [17, 20], recommendation systems [6], and bioinformatics [37]. However, the neighbor-fetching operations in GNNs make it hard for practical industrial applications, particularly when it comes to latency constraints in numerous edge devices.

The quest for more efficient alternatives to GNNs has given rise to a new generation of methods, known as **G**raph Neural Network **to M**ulti-Layer Perceptrons (MLPs) **K**nowledge **D**istillation (**G2M KD**) techniques [2, 24, 28, 32]. The primary idea is to transfer the knowledge learned by a GNN teacher into a MLP student via knowledge distillation [10], which is graph-agnostic. G2M methods enable faster and less resource-intensive inference while maintaining competitive performance compared to GNNs.

Despite their promise, G2M KD methods face two critical challenges that restrict their real-world applicability: **insufficient training data** and **incomplete test data**. In many real-world scenarios, acquiring labeled graph data is a costly and time-consuming process and they often contain nodes with missing or incomplete features, particularly in the context of test (unseen) data. For example, in industries like finance and e-commerce, dealing with insufficient or incomplete data is a daily challenge since many customers refuse to provide (part of) their information. Ensuring the robustness of students in the presence of insufficient training data and incomplete test data is crucial for making informed decisions.

Unfortunately, the above challenges are ignored by existing G2M methods. In the insufficient training data case, traditional G2M methods employing a single MLP student can easily memorize the limited training data rather than learn general patterns from it, inducing degraded performance on test data. It is a more serious challenge on G2M than GNNs since GNNs can at least fetch neighbor information to obtain a more general picture of the graph. In the incomplete test data case, current G2M methods, which are typically designed for complete data, may struggle to make inference over the feature-missing data.

[*] Corresponding Author

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671699

In response to these challenges, we propose AdaGMLP (AdaBoosting GNN-to-MLP Knowledge Distillation), a novel framework designed to address the limitations of existing G2M methods. It draws inspiration from ensemble learning [4, 36] to leverage multiple MLP students for improved distilled knowledge via our developed AdaBoost Knowledge Distillation. Specifically, for each MLP student, we introduce a Random Classification and Node Alignment mechanism to enhance its generalization capabilities. This framework allows us to mitigate overfitting in scenarios with limited training data and ensure robust predictions on test data with missing or incomplete features. Through comprehensive experiments on seven benchmark graph datasets, we demonstrate that AdaGMLP surpasses the performance of state-of-the-art (SOTA) G2M methods across various scenarios, making it a promising solution for deploying efficient and adaptable models in real-world applications.

Our main contributions are summarized as follows:

- **Tackling Real-world Challenges:** We identify two often-neglected challenges of insufficient training data and incomplete test data in current G2M KD methods and present experimental analysis in Sec. 4. These issues are particularly pronounced in G2M contexts, presenting a more serious challenge compared to their impact on GNNs. GNNs inherently leverage message passing to incorporate neighbor information, somewhat mitigating these issues. AdaGMLPintroduces innovative solutions to both issues, which are critically needed for real-world applications.
- **Novel Ensemble Architecture for G2M:** To address the above challenges, we propose AdaGMLPas a novel framework consisting of Random Classification, Node Alignment, and AdaBoost Knowledge Distillation techniques. For the first time within the G2M knowledge distillation domain, our work pioneers the introduction of an ensemble architecture, making a significant departure from existing strategies focused on enhancing G2M through complex modifications or augmentations. The prior efforts, while valuable, have not ventured into establishing a generalizable G2M architecture. We have specifically tailored and extended AdaBoost for G2M, using it as a mechanism to significantly boost the generalization ability of individual MLP students.
- **Comprehensive Empirical Analysis of AdaGMLP:** Extensive experiments reveal that AdaGMLP surpasses SOTA G2M methods in almost all the cases, underscoring its great effectiveness and generalization ability for practical applications.

2 Related Works

In this section, we introduce the works of transferring knowledge from a larger GNN teacher to a smaller student GNN or MLP. Specifically, we represent them as **G2G** (GNN-to-GNN) or **G2M** (GNN-to-MLP) **KD** (Knowledge Distillation), respectively.

Graph-to-Graph Knowledge Distillation. Prior researches [3, 12, 15, 21, 21, 29, 39, 40] have primarily focused on training compact student GNNs from more expansive GNNs using KD techniques [1, 10]. For example, methodologies like LSP [38] and TinyGNN [34] facilitate the transfer of localized structural insights from teacher GNNs to student GNNs. RDD [42] delves into the reliability aspects of nodes and edges to enhance the G2G KD. Although the student model used in CPF [35] is MLP, it additionally leverages label propagation, which still requires latency-inducing neighbor fetching. Nevertheless, these approaches still necessitate neighbor fetching, which can be impractical for applications where latency is a critical concern.

Graph-to-MLP Knowledge Distillation. In response to latency concerns, recent advancements propose employing MLP students, eliminating the need for message passing during inference and showcasing competitive performance against GNN students. A pioneer work, GLNN [41] introduces a general G2M framework without propagations. It trains an MLP student guided by both ground-truth labels and soft labels from the GNN teacher. KRD [30] develops a reliable sampling strategy to train MLPs with confident knowledge. Additionally, NOSMOG [24] combines both structural and attribute features, which serve as inputs to MLPs, thus establishing a structure-aware MLP student. Similarly, GSDN [31] introduces topological information into student training stage. Besides, FF-G2M [28] explores and provides low- and high-frequency knowledge from the graph for the student. While traditional G2M methods have made notable strides in mitigating latency concerns and enabling efficient knowledge transfer, they still exhibit certain limitations, particularly when faced with challenges related to limited training data and feature missing scenarios. We will discuss both limitations in Sec. 4.

3 Preliminaries

Notions. We denote a graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the node set and edge set, respectively. let N represent the total number of nodes. Node features are usually represented by the matrix $\mathbf{X} \in \mathbb{R}^{N\times d}$, where each row $\mathbf{x}_i$ corresponds to the node i's d-dimensional feature vector. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ indicates neighbor connections, where $\mathbf{A}_{ij} = 1$ if there is an edge $(i, j) \in \mathcal{E}$, and 0 otherwise. In this paper, we use capital letters to represent matrices, with corresponding lowercase letters used to denote specific rows within these matrices. For example, $\mathbf{x}_i$ represents the i-th row vector of $\mathbf{X}$.

Node Classification Problem Statement. The label matrix is represented by $\mathrm{Y} \in \mathbb{R}^{N\times C}$ consisting of N one-hot vectors, where C is the number of classes. We use the superscript L and U to divide all the nodes into labeled ($\mathcal{V}^L$, $\mathbf{X}^L$, and $\mathbf{Y}^L$) and unlabeled parts ($\mathcal{V}^U$, $\mathbf{X}^U$, and $\mathbf{Y}^U$). The goal of node classification problem is to predict $\mathbf{Y}^U$ with $\mathbf{A}$, $\mathbf{X}$, and $\mathbf{Y}^L$ available.

Graph Neural Networks. Generally, most GNNs follow the message-passing scheme. That is, The representation $\mathbf{h}_i$ of each node i undergoes iterative updates within each layer by gathering messages from its neighbors, denoted as $\mathcal{N}(i)$. In the l-th layer, $\mathbf{h}_i^{(l)}$ is computed from the representation of the previous layer through an aggregation process denoted as AGGR, which is then followed by an UPDATE operation. This can be formally expressed as:

$$\tilde{\mathbf{h}}_i^{(l)} = \mathrm{AGGR}^{(l)}(\{\mathbf{h}_i^{(l-1)} : i \in \mathcal{N}(i)\}) \tag{1}$$

$$\mathbf{h}_i^{(l)} = \mathrm{UPDATE}^{(l)}(\tilde{\mathbf{h}}_i^{(l)}, \mathbf{h}_i^{(l-1)}). \tag{2}$$

Graph-to-MLP Knowledge Distillation. [10] first introduces the concept of KD to enforce a simple student to mimic a more complex teacher. Notably, [41] proposed a G2M KD framework, wherein GNNs function as teachers and MLPs serve as students. Let $\mathbf{Z}^g \in \mathbb{R}^{N\times C}$ and $\mathbf{Z}^m \in \mathbb{R}^{N\times C}$ represent the final outputs (prior to Softmax) of a GNN and an MLP, respectively. The G2M objective encompasses both the cross-entropy $\mathrm{CE}(\cdot,\cdot)$ between the predictions of the MLP and ground-truth labels:

$$\mathcal{L}_{\mathrm{CE}} = \frac{1}{|\mathcal{V}^L|}\sum_{i\in\mathcal{V}^L}\mathrm{CE}(\sigma(\mathbf{z}_i^m),\mathrm{y}_i), \tag{3}$$

as well as the KL-divergence $\mathcal{D}_{\mathrm{KL}}(\cdot,\cdot)$ calculated between the soft labels generated by the GNN and MLP:

$$\mathcal{L}_{\mathrm{KL}} = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}}\mathcal{D}_{\mathrm{KL}}(\sigma(\mathbf{z}_i^g/\tau),\sigma(\mathbf{z}_i^m/\tau)), \tag{4}$$

where σ is the Softmax function and $\tau \in (0,1]$ is the distillation temperature hyperparameter. Then, the overall objective $\mathcal{L}_{\mathrm{G2M}}$ is defined as follows:

$$\mathcal{L}_{\mathrm{G2M}} = \lambda\mathcal{L}_{\mathrm{CE}} + (1-\lambda)\mathcal{L}_{\mathrm{KL}}, \tag{5}$$

where $\lambda \in (0,1)$ is a weighted parameter.

4 Motivation

4.1 Challenges in Existing G2M KD

Recently, G2M KD methods [24, 30, 41] have demonstrated remarkable results on graph-based tasks, showcasing their superiority over traditional GNNs and G2G methods. This superiority primarily stems from their minimal inference computational overhead. However, these current G2M KD methods face significant challenges, often overlooked but highly relevant in practical scenarios:

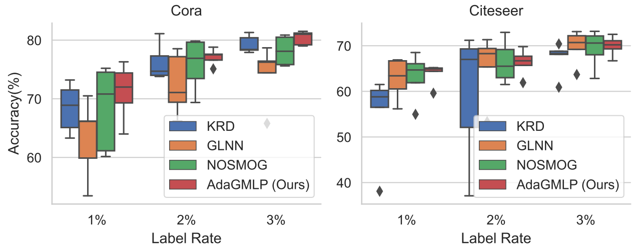


Figure 1: [Challenge 1] Insufficient Training Data. The single-MLP G2M methods with a single MLP student exhibit higher sensitivity to changes in label rates compared to vanilla GNNs. Notably, as the label rate decreases, there is a discernible trend of increasing box heights and the distance between outliers and box boundaries.

[Challenge 1] Insufficient Training Data. GNNs inherently possess strong generalization capabilities, benefiting from their ability to leverage unlabeled nodes via structural relationships for making predictions on unseen data. However, transferring GNNs' knowledge into an MLP becomes problematic when training data is scarce. The first principle of G2M is *"latency comes first."* Therefore, MLP sacrifices the ability of fetching neighbor information so that it can be readily applied to latency-sensitive machines. In scenarios with limited training data, relying solely on a single distilled MLP can lead to overfitting or getting stuck in local optima, resulting in inference bias. This concern motivates us to explore the generalization ability of G2M KD methods, especially in scenarios with limited data, as shown in Figure 1. We evaluate SOTA G2M KD methods, i.e., GLNN [41], KRD [30], and NOSMOG [24] with GCN as the teacher using `Cora` and `Citeseer` datasets under varying label rates. The variability in accuracy within each method, as demonstrated by the height of the boxes and the separation between outliers and the box boundaries, reveals that the present single-MLP G2M methods exhibit higher sensitivity to changes in label rates. In contrast, our `AdaGMLP` gets benefits from an AdaBoost-style ensemble and Random Classification strategy (discussed in Sec. 5) to obtain more stable performance.

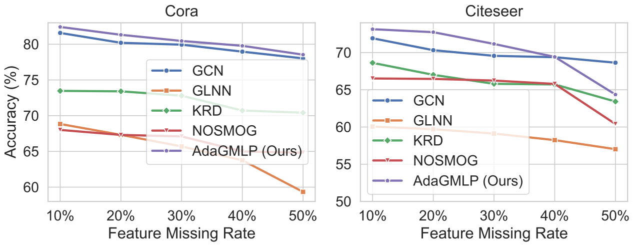


Figure 2: [Challenge 2] Incomplete Test Data. Traditional G2M methods suffer from performance consistent drops when more features are missing. Our `AdaGMLP` consistently maintains a high accuracy level, outperforming other G2M methods as the fraction of missing features increases.

[Challenge 2] Incomplete Test Data. Real-world graph data is frequently incomplete, with missing features in test (new) nodes. However, traditional G2M KD methods ignore such situations and are trained under the complete datasets. When faced with feature-missing test data, they may yield suboptimal results due to the lack of mechanisms to effectively cope with this inherent incomplete features issue. This limitation becomes increasingly critical when making predictions on real-world graphs with incomplete information. In Figure 2, we visualize the performance of different G2M methods under varying levels of missing features on the `Cora` and `Citeseer` datasets. Unlike GCN that achieves a relatively stable performance, the performance of traditional G2M methods gradually decrease as more features are masked since they fail to teach the MLP student how to handle feature-missing situations. Instead, `AdaGMLP` tends to achieve more stable performance than counterparts due to our Node Alignment module (discussed in Sec. 5).

4.2 Towards Addressing these Challenges

To address these aforementioned challenges, we propose an AdaBoosting GNN-to-MLP KD (`AdaGMLP`) framework to address these situations that impede the performance of existing G2M methods. Regarding **Challenge 1**, we tackle this by harnessing an AdaBoost-style [9] ensemble [4, 36] of multiple MLP students trained on different subsets of labeled nodes. This strategy encourages diversity in learned patterns and mitigates the risk of over-reliance on specific subsets during training. Figure 1 shows AdaGMLP's great generalization ability to deal with scarce label resources. To tackle **Challenge 2**, we introduce the Node Alignment technique for each MLP student, aligning representations between labeled nodes with complete and masked features. This mechanism ensures robust predictions on test data with missing or incomplete features, thereby extending its applicability to real-world scenarios. As shown in Figure 2, AdaGMLP maintains a high and consistent accuracy level, demonstrating AdaGMLP's superiority in handling feature-missing data and its potential for real-world applications.

5 Methodology

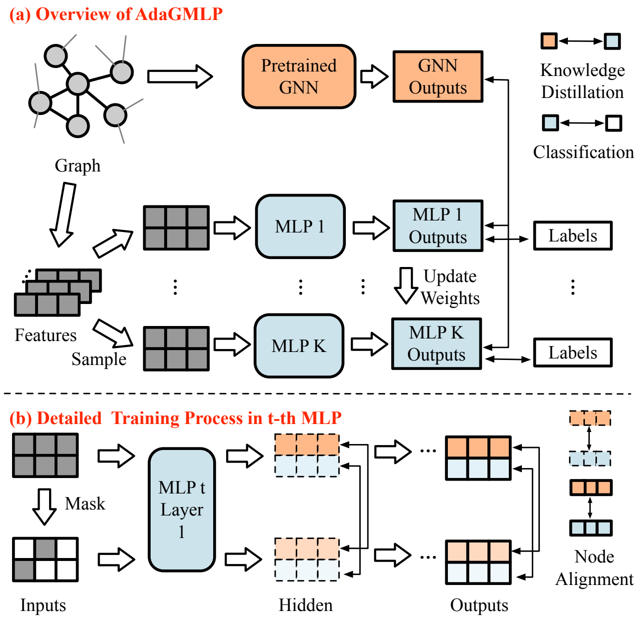


Figure 3: Illustration of AdaGMLP. In (a), for each MLP, we compute the KL loss using node weights, which are determined by the difference between MLP and corresponding GNN outputs (Knowledge Distillation). Additionally, we calculate the CE loss by comparing the sampled labeled nodes with their respective ground-truth labels (Random Classification). In (b), we begin by obtain incomplete nodes with randomly masking the features of the selected nodes and inputting them into the MLP. Subsequently, we employ Mean Squared Error (MSE) loss to align their hidden representations and outputs (Node Alignment).

In this section, we introduce AdaGMLP, a methodology designed to tackle the challenges of G2M distillation while bolstering generalization and model capacity. AdaGMLP consists of a pre-trained GNN as the teacher and a compact student network with K MLPs with L layers. Figure 3 illustrates the architecture, showcasing three fundamental components: Random Classification (RC), Node Alignment (NA), and AdaBoosting Knowledge Distillation (AdaKD).

5.1 Random Classification

We denote each MLP student as $\text{MS}_1, \text{MS}_2, ..., \text{MS}_K$. Their respective outputs are represented as $\mathbf{Z}^{m1}, \mathbf{Z}^{m_2}, ..., \mathbf{Z}^{m_K} \in \mathbb{R}^{N\times C}$. To enhance the student network's generalizability, we introduce randomness into the inputs for $\text{MS}_1, \text{MS}_2, ..., \text{MS}_{K-1}$ by selecting $\lfloor|\mathcal{V}^L|/K\rfloor$ nodes randomly from $\mathcal{V}^L$ *without replacement*. The remaining nodes are used as the input for MS_K, where $\lfloor\cdot\rfloor$ represents the floor function. Assume the labeled node subset of MS_k is $\mathcal{V}_k^L$, the classification objective $\mathcal{L}_{\text{CE}}^{(k)}$ for MS_k can be written as:

$$\mathcal{L}_{\text{CE}}^{(k)} = \frac{1}{|\mathcal{V}_k^L|}\sum_{i\in\mathcal{V}_k^L} \text{CE}(\sigma(\mathbf{z}_i^{m_k}), \mathbf{y}_i). \tag{6}$$

By training different MLP students on different subsets of labeled nodes, this encourages the student network to capture various patterns present in the dataset and avoids over-reliance on a specific subset of labeled nodes, leading to improved and stable performance. The Random Classification objective, $\mathcal{L}_{\text{RC}}$, is presented as:

$$\mathcal{L}_{\text{RC}} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{\text{CE}}^{(k)} \tag{7}$$

5.2 Node Alignment

The primary idea behind Node Alignment is to align the representations of nodes with complete features (labeled nodes) and those with masked features (masked nodes) since we often encounter datasets where labeled nodes have complete feature information, while unlabeled nodes have missing features. To illustrate this, let $\mathbf{x}_i \in \mathbb{R}^d$ represent a complete node, and $\tilde{\mathbf{x}}_i$ signify a corrupted node with a fraction ρ of its features randomly masked, where $\rho \in (0, 1)$. Consequently, we obtain outputs $\mathbf{z}_i^{m_k}$ and $\tilde{\mathbf{z}}_i^{m_k}$ as well as hidden representations $\mathbf{h}_i^{m_k,l}$ and $\mathbf{h}_i^{m_k,l}$, where $l \in \{1, 2, \cdots, L-1\}$. Now, let's delve into the two critical aspects of Node Alignment.

Output Alignment (NA-O). In the NA-O phase, our objective is to ensure that AdaGMLP's predictions on labeled nodes with completed and masked features are consistent. By minimizing the squared L2 norm between the predictions for complete and masked features, as expressed by the loss $\mathcal{L}_{\text{NA-O}}$ which is expressed as:

$$\mathcal{L}_{\text{NA-O}} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{\text{NA-O}}^{(k)} = \sum_{k=1}^{K}\frac{\sum_{i\in\mathcal{V}_k^L}\|\mathbf{z}_i^{m_k} - \tilde{\mathbf{z}}_i^{m_k}\|^2}{K|\mathcal{V}_k^L|}. \tag{8}$$

NA-O encourages the model to produce similar predictions for both labeled and masked nodes. This consistency contributes to stable model behavior and facilitates robust predictions.

Hidden Representation Alignment (NA-H). In the NA-H phase, we focus on aligning the hidden representations of nodes at different layers of the model. Similar to NA-O, we minimize the squared L2 norm between the hidden representations for complete and masked features for each layer:

$$\mathcal{L}_{\text{NA-H}} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{\text{NA-H}}^{(k)} = \frac{\sum_{l=1}^{L-1}\sum_{i\in\mathcal{V}_k^L}\|\mathbf{h}_i^{m_k,l} - \tilde{\mathbf{h}}_i^{m_k,l}\|^2}{K|\mathcal{V}_k^L|(L-1)}. \tag{9}$$

This consistency ensures that the model maintains a coherent understanding of nodes with varying feature completeness across different layers.

By incorporating both NA-O and NA-H with a controlling parameter $\lambda_{\text{NA}} \in (0, 1)$ to optimize the following objective $\mathcal{L}_{\text{NA}}$:

$$\mathcal{L}_{\text{NA}} = \lambda_{\text{NA}}\mathcal{L}_{\text{NA-O}} + (1 - \lambda_{\text{NA}})\mathcal{L}_{\text{NA-H}}, \tag{10}$$

AdaGMLP achieves the dual goals of producing consistent predictions and maintaining coherent representations across nodes with varying feature completeness. This results in a more robust, generalizable, and stable model.

5.3 AdaBoosting Knowledge Distillation

We leverage AdaBoosting to obtain the collective power of multiple MLP students, further enhancing MLP students's generalization and performance. To achieve this, we adapt SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function) algorithm [9], which is an extension of the standard two-class AdaBoost, to propose the KD-SAMME algorithm for combining MLP students in the context of G2M.

In KD-SAMME, we compute weighted error $e^{(k)}$, relying on KL-divergence for quantifying knowledge point (node) dissimilarity. The divergence between each node pair is denoted as $d_i^{(k)} = \mathcal{D}_{\text{KL}}(\sigma(\mathbf{z}_i^g), \sigma(\mathbf{z}_i^{m_k}))$. The error $e^{(k)}$ is determined as:

$$e^{(k)} = \frac{\sum_{i=1}^{N} w_i \left(1 - \exp(-\beta d_i^{(k)})\right)}{\sum_{i=1}^{N} w_i}, \tag{11}$$

where w denotes the weight of i-th node and $\beta > 0$ controls the sensitivity to divergence between knowledge point pairs. This divergence captures the dissimilarity between individual knowledge points extracted from both the teacher and student models.

Subsequently, we leverage this error information to compute a corresponding combining weight $\alpha^{(k)}$ for each MLP student as:

$$\alpha^{(k)} = \max\{\log \frac{1 - e^{(k)}}{e^{(k)}}, \epsilon\}, \tag{12}$$

where ϵ is an extremely small. value Further, node weights w_i are updated by adjusting them based on $\alpha^{(k)}$ and $e^{(k)}$:

$$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(k)}\left(1 - e^{-\beta d_i^{(k)}}\right)\right), i = 1, \cdots, N \tag{13}$$

Then, node weights w_i are normalized. The KD objective for MS_k can be written as:

$$\mathcal{L}_{\text{KL}}^{(k)} = \sum_{i \in \mathcal{V}} w_i \mathcal{D}_{\text{KL}}(\sigma(\mathbf{z}_i^g/\tau), \sigma(\mathbf{z}_i^{m_k}/\tau)). \tag{14}$$

In summary, we obtain the AdaBoosting KD objective $\mathcal{L}_{\text{AdaKD}}$:

$$\mathcal{L}_{\text{AdaKD}} = \frac{1}{K}\sum_{k=1}^{K} \mathcal{L}_{\text{KL}}^{(k)}, \tag{15}$$

5.4 Training and Inference

Training. We define the overall AdaGMLP objective $\mathcal{L}_{\text{AdaGMLP}}$ as:

$$\mathcal{L}_{\text{AdaGMLP}} = \lambda\mathcal{L}_{\text{RC}} + (1 - \lambda)\mathcal{L}_{\text{AdaKD}} + \mathcal{L}_{\text{NA}}, \tag{16}$$

where $\lambda \in (0, 1)$ is a parameter to control the weight between RC and AdaKD. We also present the algorithm of AdaGMLP in Algorithm 1.

Algorithm 1 AdaGMLP Algorithm (Transductive)

```
Input: GNN teacher's output $\mathbf{Z}^m$, hyperparameters $\tau$, $\beta$, $\lambda$, and $\lambda_{\text{NA}}$
Initialize Node weights $w_i = \frac{1}{N}$ for all $i \in \mathcal{V}$, combining weights $\alpha^{(k)} = 1$ for $k = 1, 2, \ldots, K$
for $t = 1$ to $T$ do
    // Student MLP training
    for $k = 1$ to $K$ do
        // Random Classification
        Sample labeled nodes to obtain a subset $\mathcal{V}_k^L$
        Train $\text{MS}_k$ with $\mathcal{V}_k^L$ via RC objective Eq. (6)
        // Node Alignment
        Randomly mask features of nodes in $\mathcal{V}_k^L$
        Train $\text{MS}_k$ using nodes with completed and partially masked features via NA objective Eq. (10)
        // AdaBoost Knowledge Distillation
        Train $\text{MS}_k$ with $\mathbf{Z}^m$ and $w^{(k)}$ via AdaKD objective Eq. (14)

        Calculate the error $e^{(k)}$ via Eq. (11)
        Calculate the combining weight $\alpha^{(k)}$ via Eq. (12)
        Update node weights $w_i$ via Eq. (13)
    end for
    Normalize node weights $w_i \leftarrow \frac{w_i}{\sum_{i=1}^{N} w_i}$
end for
Obtain final prediction $p_i$ for node $i$ via Eq. (17)
```

Inference. After the training process, we obtain a student network comprising K MLPs, each associated with corresponding weights $\alpha^{(1)}, \alpha^{(2)}, \cdots, \alpha^{(K)}$. We aggregate predictions from these distinct MLPs in an AdaBoost-like manner to generate the final predicted label $\hat{\mathbf{y}}_i$ for the i-th node:

$$\hat{\mathbf{y}}_i = \arg\max_c \sum_{k=1}^{K} \bar{\alpha}^{(k)} \sigma(\mathbf{z}_i^{m_k}) \tag{17}$$

Here, $\arg\max_c$ denotes the selection of the class with the highest value among all classes and $\bar{\alpha}^{(k)}$ is the normalized version of $\alpha^{(k)}$. Larger β values emphasize these under-distilled instances more, effectively making them "stronger" in knowledge transferring.

5.5 Complexity

AdaGMLP's computational complexity primarily derives from the multiple MLPs in the ensemble and the operations involved in Node Alignment and AdaBoosting techniques. Assuming each MLP in the ensemble comprises two layers, including a transformation from m-dimensional input features to d-dimensional hidden representations and a projection from these hidden dimensions to c-dimensional outputs, the computational complexity for each MLP is $O(md + dc)$. For K MLPs, the combined complexity for Node Alignment amounts to $O(2Kd(m + c))$. Furthermore, the AdaBoosting process, which updates weights and combines predictions across MLPs, contributes additional complexity. This aspect of the process is proportional to the number of nodes and the ensemble size, represented as $O(nK)$, where n is the number of nodes. Therefore, the time complexity of

Table 1: Classification accuracy ± std (%) in the `Transductive Setting` and `Inductive Setting`.

Teacher	**Student**	Cora	Citeseer	Pubmed	Photo	CS	Physics	ogbn-arxiv	$\bar{\Delta}_{GLNN}$
Transductive Setting									
MLPs	-	56.66$_{\pm2.02}$	59.88$_{\pm0.59}$	71.94$_{\pm1.24}$	78.16$_{\pm2.76}$	87.17$_{\pm1.04}$	87.24$_{\pm0.61}$	53.60$_{\pm1.31}$	-
GCN	-	82.02$_{\pm0.98}$	71.88$_{\pm0.34}$	77.24$_{\pm0.23}$	90.60$_{\pm2.15}$	89.73$_{\pm0.67}$	92.29$_{\pm0.58}$	71.22$_{\pm0.18}$	-
	GLNN	82.08$_{\pm1.14}$	73.46$_{\pm0.47}$	80.40$_{\pm0.59}$	91.44$_{\pm2.23}$	92.39$_{\pm0.53}$	93.16$_{\pm0.63}$	67.23$_{\pm0.68}$	-
	NOSMOG	82.65$_{\pm1.31}$	73.47$_{\pm1.49}$	80.95$_{\pm2.21}$	92.39$_{\pm1.95}$	93.71$_{\pm0.63}$	93.49$_{\pm0.42}$	71.07$_{\pm0.24}$	↑ 1.41%
	KRD	82.42$_{\pm1.19}$	74.24$_{\pm0.75}$	81.44$_{\pm0.58}$	91.76$_{\pm2.46}$	93.77$_{\pm0.23}$	94.13$_{\pm0.39}$	70.12$_{\pm0.37}$	↑ 1.42%
	AdaGMLP (ours)	**84.26**$_{\pm0.83}$	**75.42**$_{\pm0.39}$	**81.88**$_{\pm0.53}$	**92.60**$_{\pm0.37}$	**93.79**$_{\pm0.33}$	**94.38**$_{\pm0.27}$	**71.45**$_{\pm0.10}$	↑ **2.51**%
GraphSAGE	-	82.04$_{\pm1.33}$	70.66$_{\pm0.31}$	78.30$_{\pm0.58}$	90.24$_{\pm2.13}$	89.28$_{\pm0.34}$	91.99$_{\pm1.03}$	70.91$_{\pm0.26}$	-
	GLNN	82.24$_{\pm1.11}$	71.90$_{\pm0.76}$	79.78$_{\pm1.46}$	91.44$_{\pm2.23}$	92.86$_{\pm0.28}$	93.28$_{\pm0.94}$	68.63$_{\pm0.12}$	-
	NOSMOG	82.74$_{\pm1.53}$	71.95$_{\pm1.39}$	80.70$_{\pm1.31}$	92.09$_{\pm1.62}$	93.04$_{\pm0.93}$	93.92$_{\pm1.29}$	70.57$_{\pm0.41}$	↑ 0.89%
	KRD	83.50$_{\pm0.96}$	72.62$_{\pm0.59}$	81.08$_{\pm0.53}$	91.57$_{\pm2.67}$	93.99$_{\pm0.17}$	94.03$_{\pm0.77}$	71.20$_{\pm0.52}$	↑ 1.44%
	AdaGMLP (ours)	**84.10**$_{\pm0.46}$	**73.26**$_{\pm0.29}$	**81.18**$_{\pm0.60}$	**92.55**$_{\pm2.31}$	**94.06**$_{\pm0.21}$	**94.17**$_{\pm0.57}$	**71.46**$_{\pm0.53}$	↑ **1.93**%
GAT	-	80.24$_{\pm1.34}$	71.24$_{\pm0.73}$	77.20$_{\pm0.68}$	86.98$_{\pm5.76}$	90.93$_{\pm0.23}$	92.39$_{\pm0.80}$	71.10$_{\pm0.10}$	-
	GLNN	81.06$_{\pm1.70}$	69.42$_{\pm3.37}$	80.78$_{\pm0.37}$	86.64$_{\pm9.86}$	93.34$_{\pm0.12}$	93.63$_{\pm0.77}$	68.40$_{\pm0.16}$	-
	NOSMOG	81.30$_{\pm1.24}$	70.52$_{\pm1.47}$	80.42$_{\pm2.25}$	92.92$_{\pm1.13}$	94.20$_{\pm0.17}$	93.98$_{\pm0.52}$	71.47$_{\pm0.18}$	↑ 2.07%
	KRD	82.58$_{\pm1.31}$	69.00$_{\pm3.38}$	81.13$_{\pm0.58}$	89.06$_{\pm2.46}$	94.12$_{\pm0.15}$	94.23$_{\pm0.30}$	71.46$_{\pm0.14}$	↑ 1.49%
	AdaGMLP (ours)	**83.78**$_{\pm0.72}$	**72.30**$_{\pm1.01}$	**81.68**$_{\pm0.59}$	**93.00**$_{\pm1.62}$	**94.35**$_{\pm0.16}$	**94.33**$_{\pm0.28}$	**71.70**$_{\pm0.33}$	↑ **3.23**%
Inductive Setting									
MLPs	-	60.20$_{\pm0.44}$	60.00$_{\pm0.30}$	72.80$_{\pm0.71}$	77.20$_{\pm3.19}$	88.97$_{\pm1.12}$	90.16$_{\pm0.42}$	56.39$_{\pm0.56}$	-
GCN	-	**79.20**$_{\pm0.46}$	**71.88**$_{\pm0.36}$	77.36$_{\pm0.71}$	88.67$_{\pm1.22}$	89.55$_{\pm0.48}$	92.47$_{\pm0.45}$	**70.80**$_{\pm0.48}$	-
	GLNN	72.80$_{\pm0.21}$	70.34$_{\pm0.60}$	78.22$_{\pm0.55}$	88.53$_{\pm2.84}$	91.72$_{\pm0.73}$	93.17$_{\pm0.70}$	61.03$_{\pm0.25}$	-
	NOSMOG	74.55$_{\pm1.74}$	70.94$_{\pm0.49}$	80.83$_{\pm2.49}$	88.93$_{\pm1.93}$	92.93$_{\pm1.93}$	**93.97**$_{\pm0.78}$	68.60$_{\pm0.24}$	↑ **3.09**%
	KRD	73.52$_{\pm0.21}$	70.36$_{\pm0.65}$	80.72$_{\pm1.26}$	88.16$_{\pm2.02}$	92.09$_{\pm0.61}$	93.79$_{\pm0.48}$	60.41$_{\pm0.26}$	↑ 0.55%
	AdaGMLP (ours)	75.02$_{\pm0.44}$	70.84$_{\pm0.28}$	**81.10**$_{\pm0.15}$	**91.15**$_{\pm1.11}$	**93.28**$_{\pm0.28}$	93.96$_{\pm0.51}$	64.30$_{\pm0.21}$	↑ 2.62%
GraphSAGE	-	**80.32**$_{\pm0.16}$	70.44$_{\pm0.42}$	77.40$_{\pm0.32}$	89.40$_{\pm1.66}$	88.94$_{\pm0.54}$	91.89$_{\pm1.67}$	**70.86**$_{\pm0.40}$	-
	GLNN	70.56$_{\pm1.54}$	70.16$_{\pm1.00}$	79.44$_{\pm1.06}$	88.55$_{\pm2.69}$	91.19$_{\pm0.35}$	92.89$_{\pm1.26}$	61.08$_{\pm0.38}$	-
	NOSMOG	71.27$_{\pm2.58}$	70.38$_{\pm1.41}$	80.91$_{\pm2.79}$	89.37$_{\pm1.90}$	91.32$_{\pm1.90}$	93.16$_{\pm1.08}$	68.48$_{\pm0.20}$	↑ 2.38%
	KRD	70.90$_{\pm1.38}$	70.26$_{\pm0.47}$	80.08$_{\pm0.44}$	89.32$_{\pm1.47}$	92.67$_{\pm0.47}$	93.55$_{\pm1.12}$	61.05$_{\pm0.18}$	↑ 0.65%
	AdaGMLP (ours)	74.78$_{\pm0.30}$	**70.47**$_{\pm0.13}$	**81.34**$_{\pm0.24}$	**91.77**$_{\pm0.43}$	**93.99**$_{\pm0.46}$	**93.98**$_{\pm0.12}$	65.16$_{\pm0.26}$	↑ **2.70**%
GAT	-	**80.24**$_{\pm0.90}$	69.72$_{\pm0.61}$	77.00$_{\pm0.68}$	89.97$_{\pm1.86}$	90.22$_{\pm0.94}$	89.95$_{\pm2.29}$	**70.52**$_{\pm0.47}$	-
	GLNN	71.66$_{\pm1.20}$	69.38$_{\pm1.21}$	79.24$_{\pm1.83}$	89.55$_{\pm1.62}$	91.07$_{\pm1.30}$	92.09$_{\pm1.92}$	60.91$_{\pm0.45}$	-
	NOSMOG	72.68$_{\pm2.23}$	70.50$_{\pm2.46}$	81.43$_{\pm3.38}$	89.31$_{\pm1.14}$	91.31$_{\pm1.24}$	93.34$_{\pm1.98}$	68.72$_{\pm0.49}$	↑ **2.85**%
	KRD	71.44$_{\pm1.31}$	69.26$_{\pm1.53}$	80.52$_{\pm1.36}$	89.49$_{\pm2.85}$	91.68$_{\pm0.36}$	92.83$_{\pm1.38}$	60.95$_{\pm0.60}$	↑ 0.37%
	AdaGMLP (ours)	73.92$_{\pm0.68}$	**71.72**$_{\pm0.94}$	**81.86**$_{\pm0.32}$	**91.44**$_{\pm1.18}$	**91.78**$_{\pm0.75}$	**93.98**$_{\pm0.36}$	63.82$_{\pm0.32}$	↑ 2.79%

our AdaGML is $O(2Kd(m+c)+nK)$ for training and $O(Kd(m+c)+nK)$ for inference.

In most cases, the hidden dimensionality d often exceeds K, allowing `AdaGMLP`to utilize relatively lighter MLPs with smaller d while still maintaining high performance. This approach not only enhances computational efficiency but also ensures that the model remains robust and effective across various learning scenarios.

6 Experiments

In this section, we conduct a series of experiments to evaluate the performance of `AdaGMLP` on real-world graph datasets, addressing the following questions:

- **Q1:** How does `AdaGMLP` perform in diverse settings (both transductive and inductive), across various real-world graphs, and with different GNN teachers (including GCN, GraphSAGE, and GAT)?
- **Q2:** How does `AdaGMLP` compare to SOTA G2M KD methods when confronted with insufficient training data?
- **Q3:** How effective is `AdaGMLP` in handling incomplete test data compared to SOTA G2M KD methods?
- **Q4:** Is `AdaGMLP` sensitive to the choice of hyper-parameters, i.e., λ, λ_{NA}, β?
- **Q5:** How does the size of the ensemble (K) impact on performance?

- **Q6:** To what extent do the individual components of AdaGMLP contribute to its overall performance?
- **Q7:** Can AdaGMLP fulfill the requirements of real-world applications?

6.1 Experiment Setting

Dataset. Similar to [30], we use six public benchmark graphs, i.e., Cora [22], Citeseer [7], Pubmed [19], Coauthor-CS, Coauthor-Physics, Amazon-Photo [23], and a large-scale graph ogbn-arxiv [11]. The statistics of datasets are provided in Appendix B.

Baselines. There are three types of baselines in this paper: **(1) GNN Teachers** including GCN [13], GraphSAGE [8], and GAT [25]; **(2) SOTA G2M Methods** containing GLNN [41], NOSMOG [24], and KRD [30]; **(3) SOTA G2G Methods** including CPF [35], RDD [42], TinyGNN [34], GNN-SD [3], and LSP [38]. The comparison between AdaGMLP and G2G methods is described in Appendix C.

Implementation. The code of AdaGMLP is built on [30] via DGL library [26] and we implement each MLP student with the same configuration (hidden dimensionality, number of layers) as its GNN teachers. We tune $K \in \{2, 3, 4\}$ for all the experiments except for the hyper-parameter analysis. Due to the space limitation, we present the search spaces of other hyper-parameters in the Appendix A.

6.2 Classification Performance Comparison (Q1)

We evaluate AdaGMLP in both transductive and inductive settings, as shown in Table 1. For all the comparing G2M methods, we evaluate the models with their released parameters. The best metrics are marked by **bold**.

In the transductive setting, AdaGMLP demonstrates superior classification accuracy compared to other G2M methods and even exceeds its GNN teachers on various datasets. In the inductive setting, AdaGMLP competes well with the SOTA methods. The average improvement over GLNN ($\bar{\Delta}_{GLNN}$), which is the representative method, varies across datasets.

It's worth noting that AdaGMLP doesn't consistently outperform NOSMOG in some cases as it dose in the transductive setting. It is because NOSMOG benefits from access to test (unseen) node structural information, which is not typically available in real-world scenarios. Considering this, the strong performance of NOSMOG in the inductive setting should be interpreted with caution. It may not be the best choice for real-world scenarios where structural information about test nodes is unknown. In contrast, AdaGMLP performs competitively in the inductive setting without relying on any information about unseen nodes. This highlights its practical applicability and versatility, as it can handle scenarios where the test node's structure is not available, making it a more robust choice for real-world applications.

6.3 Insufficient Training Data Case (Q2)

We conducted experiments with varying label rates (1%, 2%, and 3%) on the Cora and Citeseer datasets in Table 2. The goal was to assess how well AdaGMLP could perform compared to GCN and other G2M methods in scenarios with limited labeled data.

Table 2: Classification accuracy ± std (%) in the Insufficient Training Data Setting with various label rates.

Dataset	Label	GCN	Student			
	Rate	Teacher	GLNN	KRD	NOSMOG	AdaGMLP
Cora	1%	$67.90_{\pm 4.24}$	$63.24_{\pm 5.94}$	$68.40_{\pm 4.18}$	$68.37_{\pm 7.25}$	$\mathbf{71.20}_{\pm 4.22}$
	2%	$76.81_{\pm 2.15}$	$72.48_{\pm 4.68}$	$76.18_{\pm 3.08}$	$75.84_{\pm 4.45}$	$\mathbf{77.92}_{\pm 1.22}$
	3%	$79.83_{\pm 1.01}$	$74.31_{\pm 4.46}$	$79.26_{\pm 1.50}$	$78.18_{\pm 2.49}$	$\mathbf{80.38}_{\pm 1.05}$
Citeseer	1%	$64.14_{\pm 1.72}$	$62.74_{\pm 4.38}$	$55.02_{\pm 9.64}$	$63.16_{\pm 5.28}$	$\mathbf{63.84}_{\pm 2.13}$
	2%	$67.10_{\pm 1.34}$	$65.54_{\pm 6.39}$	$59.34_{\pm 14.54}$	$66.42_{\pm 4.16}$	$\mathbf{66.46}_{\pm 2.61}$
	3%	$69.06_{\pm 1.82}$	$69.78_{\pm 3.71}$	$67.30_{\pm 3.69}$	$69.35_{\pm 4.13}$	$\mathbf{69.96}_{\pm 1.94}$

Traditional G2M methods struggle to match the performance of the GCN teacher in the low-label-rate settings. This is primarily because these single-student methods might be easily over-fit to limited labeled data. As a result, they tend to show higher standard deviations compared to GCN teacher and our AdaGMLP.

AdaGMLP demonstrates superior adaptability and performance in this setting. Its ability to capture and utilize information efficiently from limited labeled nodes allows it to outperform traditional G2M methods and even the GCN teacher in some cases. Additionally, AdaGMLP's robustness (smaller standard deviation) across different label rates demonstrates its potential in real-world applications where obtaining a large amount of labeled data is challenging or expensive.

6.4 Incomplete Testing Data Case (Q3)

In the feature-missing setting, we conducted extensive experiments on the Cora, Citeseer, and Pubmed datasets to evaluate the performance of AdaGMLP and compare it with GCN, and three G2M methods. We examine the impact of missing rates (10%, 20%, 30%, 40%, and 50%) on classification accuracy. The overall results are provided in Table 3. As the missing rate increases, traditional G2M methods suffer from a significant drop in accuracy. This indicates their vulnerability to missing data, limiting their practicality in

Table 3: Classification accuracy ± std (%) in the Feature-missing Setting with test node features randomly masked according to the missing rate.

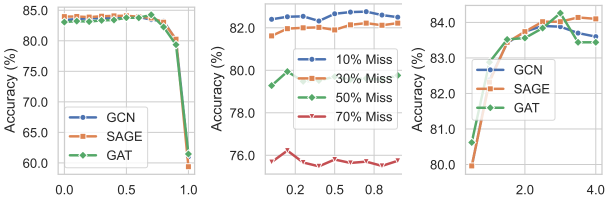
Dataset	Missing	GCN	Student			
	Rate	Teacher	GLNN	KRD	NOSMOG	AdaGMLP
Cora	10%	$81.58_{\pm 1.42}$	$68.84_{\pm 4.71}$	$73.47_{\pm 1.43}$	$68.01_{\pm 3.97}$	$\mathbf{82.42}_{\pm 0.72}$
	20%	$81.20_{\pm 1.45}$	$67.32_{\pm 2.20}$	$73.41_{\pm 1.26}$	$67.31_{\pm 4.34}$	$\mathbf{81.32}_{\pm 1.49}$
	30%	$79.94_{\pm 1.99}$	$65.70_{\pm 3.26}$	$72.80_{\pm 1.93}$	$67.10_{\pm 3.66}$	$\mathbf{80.46}_{\pm 1.32}$
	40%	$78.96_{\pm 2.59}$	$63.76_{\pm 3.55}$	$70.72_{\pm 1.95}$	$65.00_{\pm 7.45}$	$\mathbf{79.78}_{\pm 1.73}$
	50%	$78.02_{\pm 1.73}$	$59.34_{\pm 4.07}$	$70.42_{\pm 1.83}$	$64.82_{\pm 7.14}$	$\mathbf{78.54}_{\pm 1.38}$
Citeseer	10%	$71.92_{\pm 0.84}$	$60.06_{\pm 4.04}$	$68.62_{\pm 1.42}$	$66.52_{\pm 4.30}$	$\mathbf{73.14}_{\pm 0.57}$
	20%	$70.32_{\pm 0.86}$	$59.70_{\pm 3.74}$	$67.01_{\pm 1.96}$	$66.46_{\pm 4.33}$	$\mathbf{72.74}_{\pm 1.25}$
	30%	$69.56_{\pm 1.91}$	$59.12_{\pm 4.13}$	$65.80_{\pm 1.83}$	$66.24_{\pm 4.73}$	$\mathbf{71.16}_{\pm 1.73}$
	40%	$69.38_{\pm 1.69}$	$58.24_{\pm 3.93}$	$65.72_{\pm 0.95}$	$65.79_{\pm 4.85}$	$\mathbf{69.42}_{\pm 1.24}$
	50%	$\mathbf{68.64}_{\pm 1.85}$	$57.02_{\pm 4.10}$	$63.42_{\pm 1.94}$	$60.40_{\pm 3.67}$	$64.36_{\pm 1.45}$
Pubmed	10%	$77.22_{\pm 0.52}$	$67.64_{\pm 3.60}$	$77.94_{\pm 0.73}$	$74.19_{\pm 3.41}$	$\mathbf{80.26}_{\pm 0.28}$
	20%	$76.86_{\pm 0.38}$	$67.24_{\pm 3.48}$	$77.44_{\pm 0.89}$	$73.14_{\pm 3.93}$	$\mathbf{79.37}_{\pm 0.43}$
	30%	$76.34_{\pm 0.35}$	$65.34_{\pm 2.57}$	$76.60_{\pm 1.02}$	$73.09_{\pm 2.23}$	$\mathbf{78.14}_{\pm 1.16}$
	40%	$76.14_{\pm 0.82}$	$64.66_{\pm 4.06}$	$75.42_{\pm 1.14}$	$72.92_{\pm 3.38}$	$\mathbf{77.42}_{\pm 0.56}$
	50%	$76.05_{\pm 0.98}$	$60.01_{\pm 4.55}$	$74.31_{\pm 1.47}$	$72.43_{\pm 4.29}$	$\mathbf{76.48}_{\pm 1.25}$

real-world scenarios where data completeness cannot be guaranteed. It is mainly due to three reasons: **(1) Lack of Mechanisms for Feature-missing Data:** existing G2M methods are typically trained on complete datasets where all features are available with the lack of mechanisms to effectively cope with missing features; **(2) Limited Feature Information:** these G2M methods, relying on fixed feature vectors for prediction, cannot generalize well in the feature-missing test data.

AdaGMLP consistently outperforms other G2M methods across all missing rates and even exhibits better performance over GCN in almost all the cases. It can be attributed to the Node Alignment module that teaches each MLP student to align feature-missing nodes and complete nodes. Additionally, the AdaBoost-style ensemble approach encourages each student to collectively compensate for the missing information by aggregating diverse knowledge from different subsets, resulting in more robust predictions.This robustness demonstrates AdaGMLP's ability to handle real-world scenarios with incomplete data effectively.

6.5 Hyper-parameters Analysis (Q4)

In this section, we provide comprehensive analysis on Cora dataset to probe into three hyper-parameters in AdaGMLP, i.e., balance weight λ of $\mathcal{L}_{\text{RC}}$ and $\mathcal{L}_{\text{AdaKD}}$, balance weight λ_{NA} of NA-H and NA-O, and sensitivity weight β of knowledge point pairs divergence. To obtain more focused analysis, we remove the Node Alignment module when the interested hyper-parameter is not λ_{NA}.



(a) λ (b) λ_{NA} (c) β

Figure 4: Hyper-parameter Analysis on λ, λ_{NA}, and β.

In Figure 4(a), we tune λ from 0 to 1 with interval of 0.1 using various GNN teachers. We can observe a noticeable and consistent drop in performance across all teacher models when λ exceeds a certain large threshold, e.g., 0.9. This phenomenon can be explained by considering the role of λ in balancing classification loss $\mathcal{L}_{\text{RC}}$ and knowledge distillation loss $\mathcal{L}_{\text{AdaKD}}$. When λ is set to be very large, the model places an overwhelming emphasis on minimizing the classification error during training. This may lead to overfitting on the training data and the teacher's knowledge not being effectively distilled into the student. More interestingly, AdaGMLP maintains high performance at $\lambda = 0$ (complete knowledge distillation). It can be attributed to our AdaBoost Knowledge Distillation, which allows students to effectively transfer valuable knowledge from GNNs.

In Figure 4(b), we observe a notable phenomenon: in high feature missing settings (e.g., 70% missing rate), smaller values of λ_{NA} lead to better results, while in low feature missing scenarios (e.g., 10% missing rate), larger values of λ_{NA} are more effective. With a higher feature missing rate, retaining information through NA-H becomes crucial since the limited available features in test nodes can be hardly classified. Smaller λ_{NA} values emphasize NA-H and allow the model to focus more on preserving hidden representations, which are essential in recovering information from incomplete features, thereby obtaining higher performance. Conversely, with a substantial portion of features available, there is ample feature information available for most nodes. Consequently, the model can exploit this rich data to generate meaningful outputs. A larger λ_{NA} value allocates more importance to the alignment of nodes based on their output representations, which acts like consistency regularization over label information [13, 43], to obtain more robust predictions.

In Figure 4(c), we explore the sensitivity of different teacher models to varying values of β from 0.5 to 4 with interval of 0.5. The parameter β plays a significant role in AdaBoost Knowledge Distillation, as it controls the importance of individual instances in the ensemble. Larger β values make student more sensitive to under-distilled node pairs. The analysis suggest that we should avoid using extremely small β.

6.6 Ensemble Size Analysis (Q5)

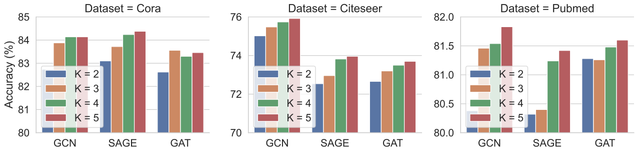


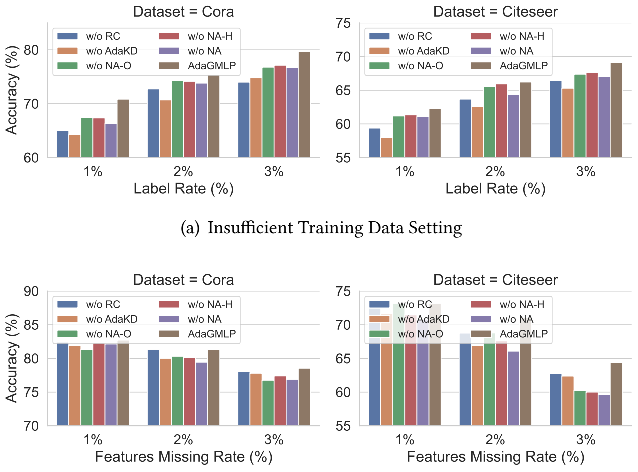
Figure 5: Ensemble Size (K) Analysis.

In this ensemble size (K) ablation experiment conducted using AdaGMLP across various datasets and teacher models, we aim to explore the sensitivity of K to model performance. The results reveal following noteworthy insights.

Across all datasets and teacher models, we observe that as K increases, the classification accuracy generally improves. This suggests that increasing the ensemble size contributes positively to the model's performance. However, it's essential to note that the improvement tends to saturate as K becomes larger. It indicates that there is an optimal point beyond which further increasing K may not significantly benefit the model's performance. The sensitivity of K to model performance suggests that AdaGMLP can benefit from larger K. Researchers can tailor the ensemble size based on their available computational resources and the dataset at hand. Smaller K may suffice for some cases, while others may require larger ensembles to maximize accuracy.

6.7 Ablation Study (Q6)

In this ablation experiment, we investigate the impact of different modules within AdaGMLP under two different settings, including insufficient training data and incomplete test data. We use GCN as the teacher model and set $\lambda = 0.5, \lambda_{\text{NA}} = 0.5, K = 2, \beta = 3$ as the default setting. Different modules (RC, AdaKD, NA-O, NA-H, NA) within AdaGMLP are systematically disabled to analyze their



(a) Insufficient Training Data Setting

(b) Incomplete Test Data Setting

Figure 6: Ablation Study.

individual contributions. The results (shown in Figure 6) provide insights into the role of each module.

Random Classification (RC). Removing the RC module, which involves randomly sampling training data for each student, leads to a significant drop in accuracy across all label rates and datasets. The decline is more obvious in insufficient training data setting, as shown in Figure 6(a). This is because, in the absence of RC, students are trained on a fixed subset of data, potentially leading to overfitting. The randomness introduced by RC helps mitigate overfitting and ensures that students see diverse examples during training. It is essential for improving generalization and robustness in the presence of insufficient training data.

AdaBoosting Knowledge Distillation (AdaKD). Eliminating AdaKD results in a noticeable performance decrease in all the cases. AdaKD contributes to improving the student's knowledge by boosting its ability of knowledge transferring. Its role is vital for maintaining high accuracy. Moreover, it has a significant impact on performance with insufficient training data. This is because when there is limited supervision, AdaKD can help student learn from the teacher's soft labels and provide additional supervision.

Node Alignment (NA). The NA module, formed by integrating NA-H and NA-O, is effective in maintaining model performance, especially under the incomplete test data setting, as shown in Figure 6(b). Removing both NA-H and NA-O leads to more pronounced performance drops, highlighting the value of their synergy within the NA module. These modules enable students to recover representations from the corrupted nodes, which is vital when dealing with incomplete test data. Without this alignment, students struggle to make predictions on unseen or partially observed nodes.

In summary, these modules serve complementary roles, and their removal impacts performance differently based on the specific challenges posed by insufficient training data or incomplete test data.

6.8 Efficiency Analysis (Q7)

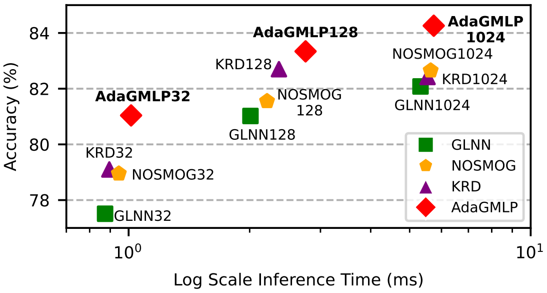


Figure 7: Accuracy *vs.* Inference Time (ms).

In Figure 7, we study the trade-off between performance and efficiency (inference time cost) of AdaGMLP and SOTA G2M methods on Pubmed dataset. For fair comparison, we use 3-layer GCN with 1024 hidden units as the teacher model and tune the hidden units in $\{32, 128, 1024\}$ for all the student model(s). We fix K at 3 for AdaGMLP.

Despite a slight increase in inference time compared to other methods, AdaGMLP offers significantly better accuracy. This trade-off is often acceptable in real-world applications, where predictive performance is paramount. Besides, AdaGMLP achieves impressive accuracy (83.34%) even with a relatively low hidden dimension of 128. Therefore, the increase in inference time cost due to the use of multiple MLPs is counterbalanced because of the compact student model design. The MLPs in AdaGMLPare designed to be compact, with fewer parameters compared to the potentially other MLP students which demand more parameters to maintain the expressive ability. This design choice significantly reduces the computational load for each MLP. Another essential practical advantage of AdaGMLP is its inherent parallelizability. AdaGMLP's architecture allows for efficient parallel computation across multiple student models. This feature can significantly reduce inference time in scenarios where parallel processing is feasible.

7 Conclusion

In this work, we introduce AdaGMLP, a novel ensemble framework for GNN-to-MLP Knowledge Distillation. Furthermore, we highlight the significant impact of limited training data and insufficient test data in G2M contexts, which pose even greater challenges. Through an extensive series of experiments, we shed light on AdaGMLP's strengths by evaluating it on various scenarios, demonstrating its great potential for real-world applications.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 62133012, 61936006, 62073255, and 62303366, in part by the Innovation Capability Support Program of Shaanxi under Grant 2021TD-05, in part by the Fundamental Research Funds for the Central Universities under Grants QTZX23039, XJSJ23030, and in part by the Innovation Fund of Xidian University under Grant YJSJ24015.

References

[1] Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? *Advances in neural information processing systems* 27 (2014).

[2] Jie Chen, Shouzhen Chen, Mingyuan Bai, Junbin Gao, Junping Zhang, and Jian Pu. 2022. SA-MLP: Distilling Graph Knowledge from GNNs into Structure-Aware MLP. *arXiv preprint arXiv:2210.09609* (2022).

[3] Yuzhao Chen, Yatao Bian, Xi Xiao, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. On self-distilling graph neural network. *arXiv preprint arXiv:2011.02255* (2020).

[4] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer, 1–15.

[5] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer, 1–15.

[6] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[7] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. 1998. CiteSeer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*. 89–98.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[9] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. 2009. Multi-class adaboost. *Statistics and its Interface* 2, 3 (2009), 349–360.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[11] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687* (2020).

[12] Chaitanya K Joshi, Fayao Liu, Xu Xun, Jie Lin, and Chuan Sheng Foo. 2022. On representation knowledge distillation for graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

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

[14] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).

[15] Carlos Lassance, Myriam Bontonou, Ghouthi Boukli Hacene, Vincent Gripon, Jian Tang, and Antonio Ortega. 2020. Deep geometric knowledge distillation with graphs. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 8484–8488.

[16] Weigang Lu, Ziyu Guan, Wei Zhao, Yaming Yang, and Long Jin. 2024. NodeMixup: Tackling Under-Reaching for Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 13 (Mar. 2024), 14175–14183. https://doi.org/10.1609/aaai.v38i13.29328

[17] Weigang Lu, Ziyu Guan, Wei Zhao, Yaming Yang, Yuanhai Lv, Lining Xing, Baosheng Yu, and Dacheng Tao. 2023. Pseudo contrastive learning for graph-based semi-supervised learning. *arXiv preprint arXiv:2302.09532* (2023).

[18] Weigang Lu, Yibing Zhan, Binbin Lin, Ziyu Guan, Liu Liu, Baosheng Yu, Wei Zhao, Yaming Yang, and Dacheng Tao. 2024. SkipNode: On Alleviating Performance Degradation for Deep Graph Convolutional Networks. *IEEE Transactions on Knowledge and Data Engineering* (2024), 1–14. https://doi.org/10.1109/TKDE.2024.3374701

[19] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.

[20] Shengjie Min, Zhan Gao, Jing Peng, Liang Wang, Ke Qin, and Bo Fang. 2021. STGSN—A Spatial–Temporal Graph Neural Network framework for time-evolving social networks. *Knowledge-Based Systems* 214 (2021), 106746.

[21] Yating Ren, Junzhong Ji, Lingfeng Niu, and Minglong Lei. 2021. Multi-task Self-distillation for Graph-based Semi-Supervised Learning. *arXiv preprint arXiv:2112.01174* (2021).

[22] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[23] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[24] Yijun Tian, Chuxu Zhang, Zhichun Guo, Xiangliang Zhang, and Nitesh Chawla. 2022. Learning mlps on graphs: A unified view of effectiveness, robustness, and efficiency. In *The Eleventh International Conference on Learning Representations*.

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

[26] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[27] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[28] Lirong Wu, Haitao Lin, Yufei Huang, Tianyu Fan, and Stan Z Li. 2023. Extracting Low-/High-Frequency Knowledge from Graph Neural Networks and Injecting it into MLPs: An Effective GNN-to-MLP Distillation Framework. *arXiv preprint arXiv:2305.10758* (2023).

[29] Lirong Wu, Haitao Lin, Yufei Huang, and Stan Z Li. 2022. Knowledge distillation improves graph structure augmentation for graph neural networks. *Advances in Neural Information Processing Systems* 35 (2022), 11815–11827.

[30] Lirong Wu, Haitao Lin, Yufei Huang, and Stan Z Li. 2023. Quantifying the Knowledge in GNNs for Reliable Distillation into MLPs. *arXiv preprint arXiv:2306.05628* (2023).

[31] Lirong Wu, Jun Xia, Haitao Lin, Zhangyang Gao, Zicheng Liu, Guojiang Zhao, and Stan Z Li. 2022. Teaching Yourself: Graph Self-Distillation on Neighborhood for Node Classification. *arXiv preprint arXiv:2210.02097* (2022).

[32] Taiqiang Wu, Zhe Zhao, Jiahao Wang, Xingyu Bai, Lei Wang, Ngai Wong, and Yujiu Yang. 2023. Edge-free but Structure-aware: Prototype-Guided Knowledge Distillation from GNNs to MLPs. *arXiv preprint arXiv:2303.13763* (2023).

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

[34] Bencheng Yan, Chaokun Wang, Gaoyang Guo, and Yunkai Lou. 2020. Tinygnn: Learning efficient graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1848–1856.

[35] Cheng Yang, Jiawei Liu, and Chuan Shi. 2021. Extract the Knowledge of Graph Neural Networks and Go Beyond It: An Effective Knowledge Distillation Framework. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 1227–1237. https://doi.org/10.1145/3442381.3450068

[36] Jing Yang, Xiaoqin Zeng, Shuiming Zhong, and Shengli Wu. 2013. Effective neural network ensemble approach for improving generalization performance. *IEEE transactions on neural networks and learning systems* 24, 6 (2013), 878–887.

[37] Yaming Yang, Ziyu Guan, Wei Zhao, Weigang Lu, and Bo Zong. 2022. Graph substructure assembling network with soft sequence and context attention. *IEEE Transactions on Knowledge and Data Engineering* 35, 5 (2022), 4894–4907.

[38] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. 2020. Distilling knowledge from graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7074–7083.

[39] Hanlin Zhang, Shuai Lin, Weiyang Liu, Pan Zhou, Jian Tang, Xiaodan Liang, and Eric P Xing. 2023. Iterative graph self-distillation. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[40] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. 2019. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3713–3722.

[41] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. 2021. Graph-less Neural Networks: Teaching Old MLPs New Tricks Via Distillation. In *International Conference on Learning Representations*.

[42] Wentao Zhang, Xupeng Miao, Yingxia Shao, Jiawei Jiang, Lei Chen, Olivier Ruas, and Bin Cui. 2020. Reliable Data Distillation on Graph Convolutional Network. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1399–1414. https://doi.org/10.1145/3318464.3389706

[43] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 912–919.

Table 4: Datasets Statics.

Dataset	# Nodes	# Edges	# Features	# Classes	Label Rate
Cora	2,708	5,278	1,433	7	5.2%
Citeseer	3,327	4,614	3,703	6	3.6%
Pubmed	19,717	44,324	500	3	0.3%
Photo	7,650	119,081	745	8	2.1%
CS	18,333	81,894	6,805	15	1.6%
Physics	34,493	247,962	8,415	5	0.3%
ogbn-arxiv	169,343	1,166,243	128	40	53.7%

A Implement Details

Hyper-parameters. We set the max training epochs at 500 for all the trails. The search space of the hyper-parameters is as follows:

- Hidden Dimensionality $F = \{128, 256, 512, 1024, 2048\}$
- Number of Layer $L = \{2, 3\}$
- Ensemble Size $K = \{2, 3\}$
- Balance Parameter $\lambda, \lambda_{\text{NA}} = \{0.1, 0.2, \cdots, 0.9\}$
- Divergence Sensitivity Parameter $\beta = \{0.5, 1, 2, 3, 4\}$

For masking rate ρ, we fix it at 0.1 in the normal setting and set to the same value as the feature missing rate in the incomplete test data setting.

Hardware and Software. AdaGMLP is implemented based on the DGL library [26] and PyTorch 1.7.1 with Intel(R) Core(TM) i9-10980XE CPU @ 3.00GHz and 2 NVIDIA TITAN RTX GPUs.

B Dataset Statics

Table 4 presents a summary of the statistical characteristics of these datasets. Data splitting strategies differ depending on the dataset's scale:

- For the three small-scale datasets, Cora, Citeseer, and Pubmed, we adopt the data splitting strategy outlined in [13].
- For Coauthor-CS, Coauthor-Physics, and Amazon-Photo, we follow the procedures from [35, 41] to perform random data splits into training, validation, and test sets.
- For the large-scale dataset, ogbn-arxiv, we strictly follow the publicly available data splits in [11].

C Performance Comparison with G2G Methods

We compare our AdaGMLP with SOTA GNN-to-GNN (G2G) methods, i.e., CPF [35], RDD [42], TinyGNN [34], GNN-SD [3], and LSP [38], in Table 5. All the methods use GCN as the teacher model. We reuse the results of G2G methods from [30]. $\bar{\Delta}_{GCN}$ is the average improvements across all the datasets over GCN.

We can observe that AdaGMLP consistently shows better performance across the majority of the datasets. The improvements in accuracy are most notable in the Cora, Citeseer, and Pubmed datasets. On Coauthor-CS and ogbn-arxiv, AdaGMLP still demonstrates competitive performance, although not the top performer. It maintains robust results but with a slightly lower margin compared to the top performer (RDD and FreeKD).

AdaGMLP not only enhances efficiency but also maintains competitive accuracy. It achieves higher accuracy than G2G methods across multiple datasets. Unlike G2G methods, which require message propagation during inference, AdaGMLP operates without this need. This efficiency is crucial in real-world applications, especially in scenarios with latency constraints and resource limitations, making AdaGMLP an optimal choice for such settings. This balance between efficiency and accuracy is a significant advantage for practical applications where both factors are essential.

D Performance Comparison with Ensemble Methods

In this section, we compare our AdaGMLPagainst some two well-known ensemble strategies. i.e., Vote, Bagging [5], and a simple average ensemble strategy that uses the average predictions from each MLP student. All the strategies use the same configuration. We conduct experiments under three different settings, including the transductive setting, insufficient training data setting, and incomplete test data setting on Cora and Citeseer. The results are provided in Table 6

In the transductive setting, our AdaBoost strategies achieve the highest accuracy on both the Cora and Citeseer datasets. Other strategies, such as average, vote, and bagging, perform relatively close to the baseline GLNN method but fall short of surpassing AdaBoost. This is attributed to AdaBoost's adaptive weighting of each student and emphasis on unaligned knowledge points, allowing it to focus on the difficult-to-extract knowledge and improve overall predictive performance.

In the insufficient training data setting, we can see that simple ensemble strategies can also achieve better performance compared to GLNN in some cases. However, there is still a performance gap between them and AdaBoost. It indicates that the AdaBoost's ability to adaptively weigh weak learners is particularly effective in tackling the challenges posed by limited labeled data.

In the incomplete test data setting, we can observe that simple ensemble strategies can also bring performance improvement when test data is corrupted. It demonstrates that combining multiple MLP students is a promising and simple way to mitigate the incomplete test data issue.

Overall, the results show that AdaBoost outperforms other ensemble strategies in various settings. Its adaptability, emphasis on challenging knowledge points, and weighting mechanism contribute to its superior performance. Additionally, the experiments highlight the potential benefits of ensemble methods for improving performance of G2M.

Table 5: Performance comparison with G2G methods.

Method	Cora	Citeseer	Pubmed	Photo	CS	Physics	ogbn-arxiv	$\bar{\Delta}_{GCN}$
MLPs	$56.66_{\pm 2.02}$	$59.88_{\pm 0.59}$	$71.94_{\pm 1.24}$	$78.16_{\pm 2.76}$	$87.17_{\pm 1.04}$	$87.24_{\pm 0.61}$	$53.60_{\pm 1.31}$	-
GCN	$82.02_{\pm 0.98}$	$71.88_{\pm 0.34}$	$77.24_{\pm 0.23}$	$90.60_{\pm 2.15}$	$89.73_{\pm 0.67}$	$92.29_{\pm 0.58}$	$71.22_{\pm 0.18}$	-
LSP	$82.70_{\pm 0.43}$	$72.68_{\pm 0.62}$	$80.86_{\pm 0.50}$	$91.74_{\pm 1.42}$	$92.56_{\pm 0.45}$	$92.85_{\pm 0.46}$	$71.57_{\pm 0.25}$	↑ 1.73%
GNN-SD	$82.54_{\pm 0.36}$	$72.34_{\pm 0.55}$	$80.52_{\pm 0.37}$	$91.83_{\pm 1.58}$	$91.92_{\pm 0.51}$	$93.22_{\pm 0.66}$	$70.90_{\pm 0.23}$	↑ 1.41%
TinyGNN	$83.10_{\pm 0.53}$	$73.24_{\pm 0.72}$	$81.20_{\pm 0.44}$	$92.03_{\pm 1.49}$	$93.78_{\pm 0.38}$	$93.70_{\pm 0.56}$	$72.18_{\pm 0.27}$	↑ 2.47%
RDD	$83.68_{\pm 0.40}$	$73.64_{\pm 0.50}$	$81.74_{\pm 0.44}$	$92.18_{\pm 1.45}$	$\mathbf{94.20}_{\pm 0.48}$	$94.14_{\pm 0.39}$	$72.34_{\pm 0.17}$	↑ 2.94%
FreeKD	$83.84_{\pm 0.47}$	$73.92_{\pm 0.47}$	$81.48_{\pm 0.38}$	$92.38_{\pm 1.54}$	$93.65_{\pm 0.43}$	$93.87_{\pm 0.48}$	$\mathbf{72.50}_{\pm 0.29}$	↑ 2.91%
AdaGMLP (ours)	$\mathbf{84.26}_{\pm 0.83}$	$\mathbf{75.42}_{\pm 0.39}$	$\mathbf{81.88}_{\pm 0.53}$	$\mathbf{92.60}_{\pm 0.37}$	$93.79_{\pm 0.33}$	$\mathbf{94.38}_{\pm 0.27}$	$71.45_{\pm 0.10}$	↑ 3.28%

Table 6: Performance comparison with Ensemble methods.

Ensemble Method	Cora			Citeseer			$\bar{\Delta}_{GLNN}$
Transductive Setting							
None (GLNN)	$82.08_{\pm 1.14}$			$73.46_{\pm 0.47}$			-
Average	$81.62_{\pm 0.97}$			$72.12_{\pm 0.40}$			↓ 1.19%
Vote	$82.04_{\pm 1.17}$			$72.44_{\pm 0.37}$			↓ 0.71%
Bagging	$82.68_{\pm 0.92}$			$73.06_{\pm 0.52}$			↑ 0.09%
AdaGMLP	$\mathbf{84.26}_{\pm 0.82}$			$\mathbf{75.42}_{\pm 0.39}$			↑ **2.66%**
Insufficient Training Data Setting							
Label Rate	1%	2%	3%	1%	2%	3%	
None (GLNN)	$63.24_{\pm 5.94}$	$72.48_{\pm 4.68}$	$74.31_{\pm 4.46}$	$62.74_{\pm 4.38}$	$65.54_{\pm 6.39}$	$69.78_{\pm 3.71}$	-
Average	$62.36_{\pm 5.97}$	$72.68_{\pm 2.72}$	$75.26_{\pm 1.64}$	$57.16_{\pm 6.92}$	$64.12_{\pm 2.49}$	$67.28_{\pm 1.08}$	↓ 2.14%
Vote	$61.40_{\pm 5.53}$	$73.90_{\pm 1.99}$	$76.94_{\pm 2.48}$	$56.36_{\pm 7.34}$	$63.74_{\pm 2.48}$	$67.20_{\pm 0.75}$	↓ 2.07%
Bagging	$61.98_{\pm 5.72}$	$74.40_{\pm 1.73}$	$77.28_{\pm 2.51}$	$56.88_{\pm 6.70}$	$64.06_{\pm 2.28}$	$68.03_{\pm 0.94}$	↓ 1.31%
AdaGMLP	$\mathbf{71.20}_{\pm 4.22}$	$\mathbf{77.92}_{\pm 1.22}$	$\mathbf{80.38}_{\pm 1.05}$	$\mathbf{63.84}_{\pm 2.13}$	$\mathbf{66.46}_{\pm 2.61}$	$\mathbf{69.96}_{\pm 1.91}$	↑ **5.58%**
Incomplete Test Data Setting							
Feature Missing Rate	10%	30%	50%	10%	30%	50%	
None (GLNN)	$68.84_{\pm 4.71}$	$65.70_{\pm 3.26}$	$59.34_{\pm 4.07}$	$60.06_{\pm 4.04}$	$59.12_{\pm 4.13}$	$57.02_{\pm 4.10}$	-
Average	$71.90_{\pm 0.82}$	$68.00_{\pm 0.56}$	$61.94_{\pm 0.82}$	$61.34_{\pm 0.41}$	$60.30_{\pm 0.90}$	$58.10_{\pm 0.61}$	↑ 3.05%
Vote	$71.82_{\pm 0.58}$	$67.98_{\pm 0.43}$	$61.92_{\pm 0.92}$	$61.02_{\pm 0.21}$	$59.86_{\pm 0.56}$	$57.84_{\pm 0.54}$	↑ 2.73%
Bagging	$71.94_{\pm 1.10}$	$68.00_{\pm 0.58}$	$62.03_{\pm 1.14}$	$61.28_{\pm 0.09}$	$59.99_{\pm 0.47}$	$57.98_{\pm 0.57}$	↑ 2.95%
AdaGMLP	$\mathbf{82.42}_{\pm 0.72}$	$\mathbf{80.46}_{\pm 1.32}$	$\mathbf{78.54}_{\pm 1.38}$	$\mathbf{73.14}_{\pm 0.57}$	$\mathbf{71.16}_{\pm 1.73}$	$\mathbf{64.36}_{\pm 1.45}$	↑ **21.59%**