Weak Models Can be Good Teachers: A Case Study on Link Prediction with MLPs

Zongyue Qin[1] **Shichang Zhang**[2] **Mingxuan Ju**[3]

Tong Zhao[3] **Neil Shah**[3] **Yizhou Sun**[1]

[1]University of California Los Angeles, Los Angeles, California, USA
[2]Harvard University, Boston, Massachusetts, USA
[3]*Snap Inc., Bellevue, Washington, USA

Abstract

Link prediction is a crucial graph-learning task. Distilling Graph Neural Network (GNN) teachers into Multi-Layer Perceptron (MLP) students has emerged as an effective approach to achieve strong performance and reducing computational cost by removing graph dependency in the inference stage, especially in applications such as citation prediction and product recommendation where node features are abundant. However, existing distillation methods only use standard GNNs. Do stronger models such as those specially designed for link prediction (e.g., GNN4LP) lead to better students? Are heuristic-based methods (e.g., common neighbors) bad teachers as they are weak models? This paper first explores the impact of different teachers in MLP distillation. Surprisingly, we find that stronger models do not always produce stronger students: MLPs distilled from GNN4LP can underperform those distilled from simpler GNNs, while weaker heuristic methods can teach MLPs to near-GNN performance with drastically reduced training costs. We provide both theoretical and empirical analysis to explain this phenomenon, revealing that a teacher is only as good as its *teachable knowledge*, the portion of its knowledge that can be transferred through the features accessible to the student. Building on these insights, we propose Ensemble Heuristic-Distilled MLPs (EHDM), which eliminates costly GNN training while effectively training complementary MLP predictors via different heuristic teachers. Our extensive experiments show EHDM reduces the total training time by **1.95-3.32**$\times$ while achieve an average **7.93%** improvement over previous GNN-to-MLP approaches, indicating that it is an efficient and effective link prediction method.

1 Introduction

Link prediction is a pivotal task in graph learning, with widespread usage in web applications. It involves estimating the likelihood of a link between two nodes, enabling applications including citation prediction, friend recommendation, knowledge graph completion [1–3], and item recommendation [4–6]. Graph neural networks (GNNs) [7–9] have been widely used for link prediction. This includes standard generic GNNs [10, 11], as well as specifically designed GNNs for link prediction (GNN4LP) which are augmented with additional structural features [12–17] to capture extra link-specific information [18]. However, all of these GNN models suffer from heavy computations in inference time due to their dependency on graphs, which require repeatedly fetching and aggregating neighboring nodes for GNN message passing [13].

*Authors affliated with Snap Inc. served in advisory roles only for this work.

Qin et al., Weak Models Can be Good Teachers: A Case Study on Link Prediction with MLPs . *Proceedings of the Fourth Learning on Graphs Conference (LoG 2025)*, PMLR 338, Arizona State University, Phoenix, USA, December 10–12, 2025.

Researchers have proposed various methods to accelerate GNN inference. There are pruning [19, 20] and quantization [21, 22] methods that accelerate GNN inference to some extent. But the speed-up is limited because they do not resolve the graph dependency issue caused by the message passing. An alternative line of work focuses on distilling knowledge of GNN teachers into Multi-Layer Perceptron (MLP) students that only take node features as inputs [13, 23–28]. By eliminating graph dependency and expensive neighborhood aggregation, MLPs offer up to 70-273$\times$ faster inference [13, 29]. Meanwhile, GNN-distilled MLPs can match or even surpass the performance of GNN teachers in many cases where rich node features are available (e.g., recommendation systems).

Advances in MLP distillation on graphs focus primarily on node classification with only a few works extending the idea to link prediction. For example, LLP [29] distills relational knowledge from standard GNNs to MLPs to handle link prediction. Yet, the link-level GNN-to-MLP distillation remains under-explored. Besides, existing MLP distillation works rely majorly on standard GNNs as teachers. In reality, state-of-the-art link prediction models often go beyond standard GNNs. GNN4LP methods, or even classical heuristic methods (e.g., common neighbors) [30–33], were shown to outperform standard GNNs in many scenarios [34]. This creates an intriguing question: *How does the choice of teacher model influence the MLP student's performance?*

Moreover, although GNN-to-MLP distillation accelerates inference, it still requires training a GNN teacher, which is itself computationally expensive in large-scale applications. Fine-tuning hyper-parameters for GNNs further inflates this cost. Hence, there is a growing incentive to identify alternative teachers that can distill useful link-level information at lower training cost.

In this work, we first explore how different teachers, ranging from GNNs to GNN4LP methods and heuristic-based approaches, affect the performance of the distilled MLP student for link prediction. Adopting the LLP framework [29] as the distillation pipeline, we have two surprising observations:

- *Good models $\neq$ good teachers.* Although GNN4LP models often outperform standard GNNs, the MLPs distilled from GNN4LP can actually underperform those distilled from simpler GNNs.
- *Heuristic methods can be good teachers, even when they underperform.* Classical heuristics can be effective teachers for MLPs to guide them achieve comparable performance to standard GNNs, even if the heuristic's performance is worse than the MLP.

To explain these findings, we provide both empirical and theoretical analyses. The first observation arises because MLPs, with their limited capacity and lack of message passing, struggle to absorb the intricate structural patterns that GNN4LP teachers rely on. In contrast, the second observation highlights that heuristic methods, despite their simplicity and weaker standalone performance, offer complementary signals that are easier for MLPs to learn and align with.

Based on these insights, we propose to distill MLPs from heuristic teachers, which will substantially shorten overall training time. To further improve prediction accuracy, we design a simple yet effective Ensemble Heuristic-Distilled MLPs (EHDM) method that trains one MLP per heuristic, then learns a gater MLP to fuse their predictions at inference time. Our gating mechanism uses *only* node features, thereby preserving the MLP's fast, neighbor-free inference. This allows us to harness multiple heuristics' complementary signals and achieve higher accuracy without reintroducing graph dependencies. To summarize, this work makes the following contributions:

- We show that even underperforming heuristic methods can be surprisingly good teachers to MLPs, substantially reducing the training cost while enabling near-GNN predictive performance.
- We provide empirical and theoretical analysis to this phenomenon, revealing that a teacher is only as good as its *teachable knowledge*, the portion of its knowledge that can be transferred through the features accessible to the student.
- Our work is the first to formally prove that a more expressive model class (GNN4LP) does not inherently offer better teachable knowledge than a simpler model class (GNN). This result highlights an important lesson for future GNN2MLP research: rather than focusing on teacher strength, we should aim to design models that are more teachable.
- We propose an ensemble method EHDM to efficiently and effectively train multiple MLPs that capture different positive links under different heuristic guidance. Experiments on ten datasets show a reduction of training time by **1.95-3.32**$\times$, with an average of **7.93%** performance gain and comparable inference speed over previous GNN-to-MLP methods [29].

2 Preliminaries and Related Work

2.1 Link Prediction on Graphs

Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph with N nodes and observed edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. Node features are represented as $\mathbf{X} \in \mathbb{R}^{N \times F}$ where $\boldsymbol{x}_i$ is the feature vector for node i. Given a pair of nodes (i, j), the goal of link prediction is to estimate the probability of connection, denoted as $P(Y_{ij} = 1)$.

Graph Neural Network (GNN) Solutions. The state-of-the-art solutions are based on GNNs. Early methods directly utilize the embeddings of two target nodes for prediction [1, 5, 10, 11, 35]. Subsequent research revealed that standard GNN methods struggle to capture link-specific information [18]. To overcome this limitation, **GNN4LP models** extend GNNs by explicitly incorporating structural information that is not inherently encoded by GNN architectures [34]. A common example is the inclusion of global node IDs, since vanilla GNN inputs typically contain only node features. These IDs can then be exploited to compute structural signals such as the number of common neighbors [15, 16] and subgraph features [18, 36–38]. Although these methods improve prediction accuracy, they increase the computational cost during training and inference, rendering them less suitable for large-scale web applications.

Heuristic Solutions. Heuristic methods estimate node proximity based on structural or feature similarities [39], categorized as follows:

- **Local Structural Proximity** considers 1-hop neighborhood similarity. We consider three heuristics: Common Neighbors (CN) [31]: $CN(i, j) = |N(i) \cap N(j)|$, Adamic-Adar (AA) [40]: $AA(i, j) = \sum_{k \in N(i) \cap N(j)} \frac{1}{\log(\text{Deg}(k))}$, Resource Allocation (RA) [33]: $RA(i, j) = \sum_{k \in N(i) \cap N(j)} \frac{1}{\text{Deg}(k)}$, where $N(i)$ is the neighbors of node i and $\text{Deg}(\cdot)$ is the node degree.
- **Global Structural Proximity** captures higher-order connectivity, e.g., SimRank [41], Katz [42], and Personalized PageRank [43]. Due to their high computational cost, we adopt Capped Shortest Path (CSP), a simplified shortest-path heuristic [44]: $CSP(i, j) = \frac{1}{\min(\tau, SP(i,j))}$, where $SP(i, j)$ is the shortest path length between i and j, and τ limits the maximum distance considered[2].
- **Feature Proximity** measures similarity in node features, assuming nodes with similar characteristics are more likely to connect [45]. Ma et al. [17] shows MLP is good at capturing feature proximity.

2.2 GNN Acceleration

Inference acceleration has been explored through both hardware and algorithmic optimizations. On the algorithmic side, techniques such as pruning [19, 20, 46] and quantization [21, 22, 47] have been widely studied to reduce model complexity and computational cost. While these techniques improve inference efficiency, they do not eliminate neighbor-fetching operation, which remains a fundamental bottleneck in GNN inference. Because GNNs rely on message passing across graph structures, even optimized models still suffer from high data dependency and irregular memory access patterns.

2.3 GNN-to-MLP Distillation

To further improve inference efficiency in graph applications, GLNN [13] pioneered GNN-to-MLP distillation, eliminating the need for neighbor fetching and significantly accelerating inference. Notably, **the MLPs only take node features as inputs**, but they can learn to infer the structural information from the node features during distillation. Subsequent works have refined distillation techniques [23–27], but mainly for node classification. Wang et al. [28] introduced a self-supervised approach applicable to node classification, link prediction, and graph classification. However, their method is restricted to GNN teachers and does not generalize to GNN4LP models or heuristic methods. The first extension of GNN-to-MLP distillation for link prediction was LLP [29], but like prior methods, it only explores GNN teachers. Notably, LLP's framework is applicable beyond standard GNNs, as it only requires prediction from the teachers. So we adopt LLP's framework while extending it to other teacher options. The details of LLP are introduced in Appendix A.

[2]With bi-directional breadth first search, we need to retrieve $\tau/2$-hops neighborhood to compute CSP.

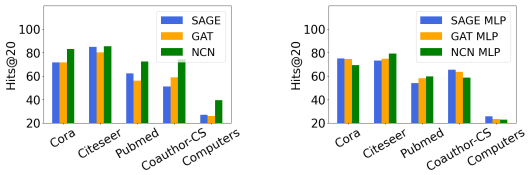
3 Key Observations and Theoretical Analysis

In this section, we present key observations that motivate the use of heuristic methods to distill MLP. First, we compare GNN and GNN4LP models as teachers and find that GNN4LP, despite achieving better accuracy, does not always serve as a superior teacher. We then highlight the advantages of using heuristics as teachers: (1) they significantly enhance MLP performance and (2) they offer better training efficiency.

3.1 Better Models $\neq$ Better Teachers

Unlike node and graph classification tasks, state-of-the-art GNN4LP models for link prediction fall outside the scope of standard message-passing neural networks (MPNNs). While they generally outperform standard GNNs, our results surprisingly show that this advantage does not translate into better MLP students. As shown in Figure 1, while NCN [16] substantially outperforms SAGE [8] and GAT [9], NCN-distilled MLPs exhibit suboptimal performance in three out of five datasets.

This aligns with prior observations that stronger teachers do not always produce better students [26, 48–50], often due to a capacity mismatch. GNN4LP models exploit structural information that standard GNNs cannot capture. Since MLPs rely solely on node features, they cannot inherit the performance gains of GNN4LPs when those gains come from feature-independent structural patterns.



(a) Hits@20 of teachers **(b)** Hits@20 of students

Figure 1: Hits@20 of standard GNNs (SAGE, GAT), a GNN4LP model (NCN), and their student MLPs across five datasets.

To formalize this intuition, we introduce a concept called "teachable knowledge", which captures the part of a teacher's knowledge that can be learned by a MLP limited to node features.

Definition 1 (Teachable Knowledge)**.** *Let $\boldsymbol{x}_i, \boldsymbol{x}_j$ be the features of node i, j. Given a teacher model F, its teachable knowledge to a MLP student is the conditional expectation $\mathbb{E}(F|\boldsymbol{x}_i, \boldsymbol{x}_j)$, which represents the component of the teacher's predictions that is explainable by node features.*

The teachable knowledge represents the best possible approximation of the teacher's output by any MLP that only has access to node features under MSE or KL-divergence. Let s_i be latent random variables representing the structural information of node i. The following Lemma formalizes the teacher can be replaced by its teachable knowledge without changing the training objective under KL-divergence. The proof is given in Appendix B.

Lemma 3.1. *Let $F(y \mid \boldsymbol{x}_i, \boldsymbol{x}_j, s_i, s_j)$ be a teacher model, and let $g(y \mid \boldsymbol{x}_i, \boldsymbol{x}_j)$ be a student model. Suppose distillation is performed using KL divergence as the loss. Then,*

$$\mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}_j, s_i, s_j} KL(F, g) = \mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}_j, s_i, s_j} KL(\mathbb{E}[F|\boldsymbol{x}_i, \boldsymbol{x}_j], g) \tag{1}$$

Therefore, if two teachers have the same teachable knowledge, they yield the same distillation loss when training the student MLP, i.e., their effectiveness as teachers is theoretically equivalent in the GNN2MLP setting.

Next, we consider the teachable knowledge of GNN and GNN4LP teachers. Let us decompose the latent variable s_i into two components: $s_i = (s_i^{\text{GNN}}, s_i^{\text{extra}})$, where s_i^{GNN} contains structural information that can be captured by GNNs, which is usually K-hop neighborhood of node i[3], and s_i^{extra} is the extra structural information beyond GNNs' capacity (e.g., the unique node IDs of node i's K-hop neighborhood[4]). GNN4LP models achieve higher accuracy than conventional GNNs precisely because they incorporate s_i^{extra} into their predictions [18, 36]. However, the following theorem formalizes why s_i^{extra} does not help the student MLPs. The proof is given in Appendix C.

Theorem 3.2 (GNN4LP models are not better teachers)**.** *Let a standard GNN be written as $F_{GNN}(\boldsymbol{x}_i, \boldsymbol{x}_j, s_i^{GNN}, s_j^{GNN})$. And a GNN4LP model as $F_{GNN4LP}(\boldsymbol{x}_i, \boldsymbol{x}_j, s_i^{GNN}, s_j^{GNN}, s_i^{extra}, s_j^{extra})$. For any F_{GNN4LP}, there exists a corresponding F_{GNN} such that the teachable knowledge from either teacher is the same:*

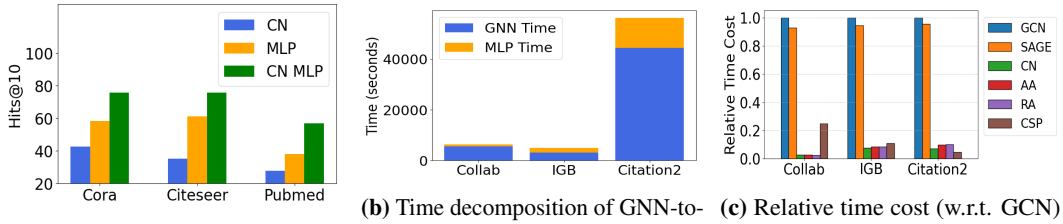
$$\mathbb{E}_{s^{GNN}, s^{extra}} \left[F_{GNN4LP} \mid \boldsymbol{x}_i, \boldsymbol{x}_j\right] = \mathbb{E}_{s^{GNN}} \left[F_{GNN} \mid \boldsymbol{x}_i, \boldsymbol{x}_j\right] \tag{2}$$

[3]It only contains node feautres, but not node IDs

[4]NCN uses node IDs to identify common neighbors between two target nodes

In other words, there always exists a GNN teacher that can impart the same information to MLP, so that its distilled MLP matches the performance of any MLP distilled from a GNN4LP teacher.

3.2 Heuristic Methods are good teachers



(a) Hits@10 of CN, MLP and CN-distilled MLP.

(b) Time decomposition of GNN-to-MLP distillation spent on GNN and MLP.

(c) Relative time cost (w.r.t. GCN) to generate guidance with different teachers on large-scale datasets.

Figure 2: Exploration of heuristic methods as teachers to distill MLPs. (a) Hits@10 comparison of CN, MLP, and CN-distilled MLP, demonstrating performance improvements through heuristic distillation even when the heuristic method underperforms. (b) Time breakdown of GNN-to-MLP distillation, showing the computational burden of GNN training. (c) Relative time cost for generating guidance with different teachers, illustrating the efficiency advantage of heuristic methods.

Since stronger GNN4LP models are not better teachers due to *capacity mismatch*, we explore using simple heuristic methods as teachers. Unlike GNNs, heuristic methods typically capture a single type of structural information (e.g., common neighbors) [39], making them easier for distillation. However, their performance is highly data-dependent, and may even fall below that of MLPs. Surprisingly, we find that utilizing heuristic teachers not only offers substantial training time savings, but also produces effective students, even when heuristic methods themselves perform poorly.

Observation 1: Heuristic methods are much faster than GNNs. Despite their effectiveness, GNNs and GNN4LP models suffer from high computational costs during both training and inference. Figure 2b illustrates the time breakdown of GNN-to-MLP distillation on three large-scale datasets: `OGBL-Collab`, `OGBL-Citation2` [51], and `IGB` [52]. The total GNN time includes both GNN training and guidance generation, while the MLP time refers to the MLP distillation process based on GNN guidance. We observe that 62.2% to 87.1% of the total time is spent on GNNs, making it the primary bottleneck in the distillation pipeline.

In contrast, heuristic methods offer a far more efficient alternative as teachers, as heuristics require no training and involve only simple calculations. Figure 2c compares the relative time costs of two GNN models (GCN and SAGE) against four heuristic methods. The time cost of GNN models includes both training and guidance generation, while the time cost of heuristic methods only includes guidance generation. The results indicate that using heuristic methods as teachers is an order of magnitude faster than using GNNs.

Observation 2: Heuristic methods are effective teachers for MLPs, even when they underperform. Figure 2a shows that CN-distilled MLPs significantly outperform non-distilled MLPs, despite CN itself having lower accuracy. This suggests that CN provides useful information that MLPs alone cannot capture. Additional results in Section 5 further confirm that heuristic-distilled MLPs achieve near-GNN performance across various datasets. These findings provide strong empirical evidence supporting the effectiveness of heuristic teachers.

This leads to a natural question: *why can heuristic methods improve MLPs even when they have lower accuracy?* We argue that heuristic methods provide complementary information to MLPs. Mao et al. [39] show that structural and feature proximity identify different positive node pairs, and propose the following model:

$$P(Y_{ij} = 1) = \begin{cases} \dfrac{1}{1 + e^{\alpha(d_{ij} - \max\{r_i, r_j\})}}, & d_{ij} \leq \max\{r_i, r_j\} \\ \beta_{ij}, & d_{ij} > \max\{r_i, r_j\} \end{cases} \tag{3}$$

where d_{ij} is the structural proximity between node i, j, and $\beta_{i,j}$ is the feature proximity. r_i and r_j are connecting threshold parameters for node i and j. The model illustrates that when two nodes

have close structure proximity, the likelihood of connection depends only on the structure proximity. Otherwise, the likelihood of connection only depends on the feature proximity.

Previous studies [17, 39] show that MLPs primarily capture feature proximity, but exhibit low correlation with structure proximity, an intuitive result since MLPs rely solely on node features. In contrast, heuristic methods primarily capture structure proximity [39]. As a result, heuristic methods identify positive node pairs that MLPs would typically miss. Moreover, because each heuristic encodes a specific structural pattern, it is easier for an MLP to learn how these patterns relate to node features, thus improving performance beyond what feature proximity alone allows. To validate our claim, we compare the positive edges identified by heuristic methods, MLPs, and heuristic-distilled MLPs using the Hits@K protocol from [17] (K=5 for Citeseer, K=10 for Cora and Pubmed). A positive edge is selected if it ranks among the top-K predictions when compared to sampled negatives. We then compute the subset ratio, the proportion of heuristic-identified positives also captured by the MLP. As shown in Figure 3, this ratio increases significantly after distillation, indicating that heuristic-distilled MLPs effectively capture extra positive edges and learn complementary information.

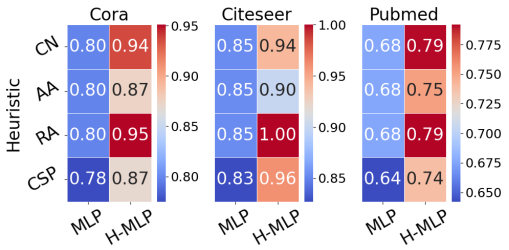


Figure 3: Subset ratio of positive edges identified by heuristic methods that are also recognized by MLPs (without distillation) and heuristic-distilled MLPs (H-MLP) across three datasets. The ratio indicates the proportion of positive edges identified by heuristic methods that are also recognized by MLPs.

Another important question is: *why are heuristic methods better teachers than GNNs?* Our key insight is that although GNNs might have better accuracy, their predictions often rely heavily on structural information that is inaccessible to the student. As a result, the teachable knowledge they offer may be worse than that of heuristic methods.

Formally, let $F(\boldsymbol{x}_i, \boldsymbol{x}_j, s_i, s_j)$ be a teacher model that approximates the ground-truth distribution $P(Y_{ij} = 1) = p(\boldsymbol{x}_i, \boldsymbol{x}_j, s_i, s_j)$. The teachable knowledge of F is defined as $\mathbb{E}F = \mathbb{E}_{s_i,s_j}\left(F(\boldsymbol{x}_i, \boldsymbol{x}_j, s_i, s_j)|\boldsymbol{x}_i, \boldsymbol{x}_j\right)$. Consider the KL Divergence

$$KL(p\|\mathbb{E}F) = \underbrace{KL(p\|F)}_{\text{teacher error}} + \underbrace{\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,s_i,s_j}\mathbb{E}_p\left(\log \mathbb{E}F - \log F\right)}_{\text{information lost}} \tag{4}$$

Eq. 4 shows that the error of teachable knowledge consists of (1) The teacher's error; (2) the information loss incurred when compressing the teacher's predictions into a form that depends only on node features. Even if a GNN has lower teacher error, the second term may be large if its predictions depend heavily on structural variables. In contrast, a heuristic that produces simpler, more feature-aligned predictions can yield lower total KL divergence after distillation.

We now present a toy example to concretely illustrate this phenomenon. Let all random variables x_i, x_j, s_i, s_j all be binary. And $Y_{ij} = 1$ if and only if $s_i = s_j = 1$. Let $x_i, x_j \sim \text{Bern}(0.8)$, and let the structural variables follow a Bernoulli distribution conditioned on features:

$$p_x := P(s = 1|x) = \begin{cases} 0.5, & x_i = x_j = 1, \\ 0.6. & \text{otherwise}, \end{cases} \tag{5}$$

We compare two teacher models:

- $F_1(x_i, x_j, s_i, s_j) = 1$ if and only if $s_i = s_j = 1$, and 0.1 otherwise. The teachable knowledge is $\mathbb{E}[F_1|x_i, x_j] = 0.1 + 0.9p_x^2$.
- $F_2(x_i, x_j, s_i, s_j) = 0.3$ (a constant predictor). The teachable knowledge is also a constant 0.3.

By computing the expected negative log-likelihood (detailed in Appendix D), we find

$$\begin{aligned} \mathbb{E}_{x,s}\left[\text{NLL}(F_1)\right] &\approx 0.07 && (F_1 \text{ effectiveness}) \\ \mathbb{E}_{x,s}\left[\text{NLL}(F_2)\right] &\approx 0.60 && (F_1 \text{ effectiveness}) \\ \mathbb{E}_{x}\left[\text{NLL}(\mathbb{E}[F_1 \mid x])\right] &\approx 0.61 && (\text{Teachable knowledge of } F_1 \text{ effectiveness}) \\ \mathbb{E}_{x}\left[\text{NLL}(\mathbb{E}[F_2 \mid x])\right] &\approx 0.60 && (\text{Teachable knowledge of } F_2 \text{ effectiveness}) \end{aligned} \tag{6}$$

Although F_1 achieves high accuracy, its teachable knowledge is no better than that of a constant teacher, due to its strong dependence on structural variables that are inaccessible to the student. This highlights the central point: *a model with better overall accuracy can still be a worse teacher if its predictions are not well aligned with the student's input space.* Simpler models like heuristics may offer better guidance for feature-only students like MLPs.

4 Ensemble Heuristic-Distilled MLPs

Section 3.2 highlights the advantages of using heuristics as teachers. However, since each heuristic captures only a specific type of structural proximity [39], an MLP distilled from a single heuristic is inherently limited. To improve effectiveness, we aim to enable MLPs to leverage multiple heuristics.

Ma et al. [17] demonstrated that combining multiple heuristics generally improves performance. So a straightforward approach is to ensemble heuristics and distill a single MLP from the combined signals. However, as discussed in Section 3.1, stronger models do not always yield better students. Unfortunately, this holds true for mixing heuristics. Figure 4 shows that while CN+CSP outperforms CN and CSP individually, its distilled MLP performs worse. We hypothesize that mixing heuristics increases structural complexity, making it harder for the MLP to learn. These results suggest that direct distillation from combined heuristics is not effective.

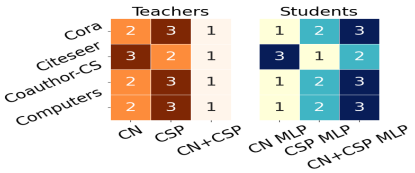
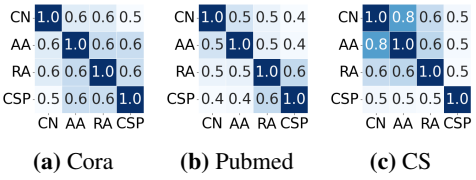


Figure 4: Accuracy rankings (Hits@20) of three teachers (CN, CSP, CN+CSP) and corresponding student MLPs across four datasets.

An alternative approach is to ensemble multiple MLPs. To effectively and efficiently ensemble multiple MLPs, our key insight is that: *Since different heuristics capture different sets of positive links, their distilled MLPs should also capture different sets of positive links.* To validate this, we compute the overlapping ratio of positive edges identified by different distilled MLPs. We use Hits@10 to define the positive edge set for each distilled MLP. More details about positive edge set can be found in Section 3.2. The overlapping ratio is then computed as the Jaccard index between two positive edge sets. As shown in Figure 5, the overlap among different heuristic-distilled MLPs is relatively low, confirming our hypothesis that each MLP captures complementary link information.

So a simple yet effective solution is to train a gating function that dynamically selects which heuristic-distilled MLP to use for each input node pair. Ma et al. [17] introduced a similar idea by training a gating network to ensemble different GNN4LP models. However, their gating network is not suitable for our setting, as it takes heuristic values as inputs, requiring neighbor-fetching operations, which would compromise the inference efficiency of MLPs.



(a) Cora **(b)** Pubmed **(c)** CS

Figure 5: Overlap ratio of student MLPs distilled from different heuristic methods.

To address this, we propose a gating MLP network that takes only the feature vectors of the target node pairs as input and outputs a weight for each distilled MLP. Our rationale is that node features alone should be sufficient for the gating function, since GNN-to-MLP distillation only works when node features correlate with structural information. The gating function is expected to learn which heuristic-derived MLP should be used based on the input node features.

The overall architecture is illustrated in Figure 6. Given an input node pair, the distilled MLPs and the gating MLP operate in parallel. The final prediction is then computed as a weighted sum of the predictions from the different MLPs. This approach ensures that: (1) No additional neighborhood fetching operations are required; (2) All MLPs can run simultaneously, minimizing inference overhead.

To train the gating function, we fix the parameters of the distilled MLPs and optimize the parameters of the gating function using the BCE loss. Additionally, we incorporate L1-regularization on the gate

weights to encourage sparsity and improve generalization. The final loss function for training the gating MLP is:

$$-y_{ij}\log(\sum_{h\in\mathcal{H}} w_h q_{i,j}^{(h)}) - (1-y_{ij})\log(1-\sum_{h\in\mathcal{H}} w_h q_{i,j}^{(h)}) + \lambda \sum_{h\in\mathcal{H}} |w_h| \tag{7}$$

where y_{ij} is the ground-truth label, $\mathcal{H}$ is the heuristic set, and $q^{(h)}$ is the prediction of the MLP distilled from heuristic h, w_h is the output weight of the gating function, and λ is a hyper-parameter controlling the weight of L1-regularization.

Compared to other ensemble methods for link prediction [17], the core novelty of EHDM is an efficient strategy to obtain a set of diverse and effective MLPs that complement each other by capturing different types of positive links. Simply training multiple MLPs via bootstrapping or random initialization is insufficient, as individual MLPs trained directly tend to perform poorly and focus mainly on feature proximity, capturing similar sets of positive links. And training multiple MLPs via boosting algorithms will increase the total training time, as they cannot be trained in parallel.

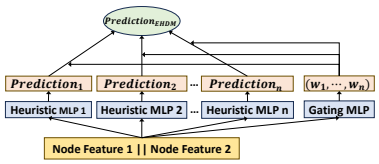


Figure 6: Illustration of Ensemble Heuristic-Distilled MLPs.

5 Experiments

Following previous works [29], we include ten datasets with rich node features for evaluation. Dataset statistics are summarized in Table 6 in the Appendix. For each dataset, we train three teacher GNNs: GCN [7], SAGE [8], and GAT [9], and report the highest-performing one, denoted as "*Best GNN*". Similarly, for the LLP baseline [29], we distill three MLPs from GCN, SAGE, and GAT, respectively, and report the best-performing one, denoted as "*Best LLP*". We include four heuristic methods in the experiments: CN [31], AA [40], RA [33], and CSP [44].

Table 1: Performance of GNN teacher and heuristic-based teachers across different datasets. For Collab we report Hits@50, for IGB we report Hits@100, for Citation2 we report MRR, for other datasets we report Hits@20. **Bold** numbers mark the best performance in each row. Underlined numbers represent the 2nd best performance in each row.

Dataset	MLP	Best GNN	CN	AA	RA	CSP
Cora	73.21±3.28	**73.96±1.23**	42.69±0.00	42.69±0.00	42.69±0.00	42.69±0.00
Citeseer	68.26±1.92	**85.10±2.25**	35.16±0.00	35.16±0.00	35.16±0.00	58.02±0.00
Pubmed	49.40±3.53	**68.83±2.84**	27.93±0.00	27.93±0.00	27.93±0.00	27.93±0.00
CS	39.36±0.99	64.24±3.52	53.65±0.00	**74.32±0.00**	74.01±0.00	0.00±0.00
Physics	21.60±3.09	65.85±2.82	61.40±0.00	78.10±0.00	**79.80±0.00**	0.00±0.00
Computers	17.53±1.25	27.07±4.45	20.38±0.00	27.54±0.00	**32.67±0.00**	0.00±0.00
Photos	31.81±2.82	**49.79±8.87**	34.37±0.00	42.63±0.00	45.39±0.00	0.00±0.00
Collab	44.38±3.47	59.14±1.64	61.37±0.00	**64.17±0.00**	63.81±0.00	46.49±0.00
IGB	19.13±1.34	**20.47±1.39**	7.78±0.00	7.78±0.00	7.78±0.00	1.00±0.00
Citation2	39.17±0.44	**84.90±0.06**	74.30±0.00	75.96±0.00	76.04±0.00	0.28±0.00

5.1 How do Heuristic-Distillation Compare to GNN-Distillation

First, we comprehensively evaluate the effectiveness of heuristic teachers for MLP distillation. Table 1 compares the teacher performance, while Table 2 reports the performance of their corresponding student models. We observe that despite heuristic methods having significantly lower performance on certain datasets, heuristic-distilled MLPs consistently match or surpass the performance of the best GNN-distilled MLPs. These results empirically demonstrate that heuristic methods can serve as effective teachers for training MLPs.

Table 2: Performance of MLP students distilled from GNN and heuristic-based methods across different datasets. For Collab we report Hits@50, for IGB we report Hits@100, for Citation2 we report MRR, for other datasets we report Hits@20. **Bold** numbers mark the best performance, underlined numbers mark the 2nd best performance.

Dataset	MLP	Best LLP	CN MLP	AA MLP	RA MLP	CSP MLP
Cora	73.21±3.28	**76.62±2.00**	75.75±2.45	74.84±3.48	75.94±4.35	74.16±4.52
Citeseer	68.26±1.92	76.53±4.88	75.69±1.14	70.90±2.24	75.38±1.36	**77.10±2.10**
Pubmed	49.40±3.53	59.95±1.58	56.98±2.47	54.95±2.91	**60.21±3.71**	57.82±3.36
CS	39.36±0.99	66.64±1.80	67.53±5.26	**69.56±2.97**	69.39±2.97	67.41±1.34
Physics	21.60±3.09	60.19±2.93	57.09±4.13	**61.10±3.15**	52.87±4.12	50.80±2.09
Computers	17.53±1.25	25.80±4.80	27.55±2.88	**30.45±4.27**	25.50±1.97	21.97±1.26
Photos	31.81±2.82	39.66±2.94	39.17±4.18	**40.23±4.07**	30.84±3.25	38.49±5.06
Collab	44.38±3.47	**49.30±0.79**	48.23±0.89	47.33±1.02	48.93±0.66	48.99±0.69
IGB	19.13±1.34	**25.12±1.14**	24.38±0.11	24.20±0.90	24.11±1.25	24.75±0.83
Citation2	39.17±0.44	42.78±0.10	43.05±0.23	**43.30±0.08**	42.90±0.12	43.17±0.11

Table 3: Comparison between EHDM and baselines across datasets. Δ_{LLP} and Δ_{GNN} denote the relative improvement (in %) of our ensemble method with respect to the best LLP and best GNN, respectively. **Bold** numbers mark the best performance, and underline numbers mark the 2nd best performance. For Collab we report Hits@50, for IGB we report Hits@100, for Citation2 we report MRR, for other datasets we report Hits@20.

Dataset	MLP	Best GNN	Best LLP	EHDM	Δ_{GNN}	Δ_{LLP}
Cora	73.21±3.28	73.96±1.23	76.62±2.00	**80.49±1.51**	+8.83%	+5.06%
Citeseer	68.26±1.92	**85.10±2.25**	76.53±4.88	79.08±1.90	-7.08%	+3.33%
Pubmed	49.40±3.53	**68.83±2.84**	59.95±1.58	60.75±3.42	-11.74%	+1.33%
CS	39.36±0.99	64.24±3.52	66.64±1.80	**74.33±2.59**	+15.71%	+11.53%
Physics	21.60±3.09	**65.85±2.82**	60.19±2.93	64.44±5.22	-2.14%	+7.06%
Computers	17.53±1.25	27.07±4.45	25.80±4.80	**30.41±2.90**	+12.35%	+17.89%
Photos	31.81±2.82	**49.79±8.87**	39.66±2.94	45.89±1.67	-7.83%	+15.71%
Collab	44.38±3.47	**59.14±1.64**	49.30±0.79	49.27±0.88	-16.70%	-0.07%
IGB	17.74±1.20	20.47±0.82	25.12±1.14	**27.27±0.27**	+33.21%	+8.54%
Citation2	38.12±0.18	**84.90±0.06**	42.78±0.10	46.58±0.10	-45.13%	+8.87%

5.2 Performance of Ensemble Approach

Next, we evaluate the efficiency and effectiveness of our ensemble approach. Figure 7 illustrates the training time decomposition of our ensemble method compared to LLP. The EHDM pipeline consists of three stages: heuristic guidance computation, MLP distillation, and ensembling. Although heuristic guidance must be generated for multiple heuristics, and MLP distillation is performed separately for each heuristic, both of these stages run in parallel. Therefore, we take the maximum time among the four heuristics as the effective time cost for each of these stages. We observe that by eliminating GNN training, our ensemble approach accelerates the entire distillation process by 1.95–3.32× compared to LLP. We also show EHDM has similar inference time as LLP in Appendix G.

Table 3 compares the performance of MLP, GNN, LLP (GNN-distilled MLP), and EHDM. The results show that, despite less training time, EHDM outperforms both MLP and LLP by leveraging the strengths of different heuristic-distilled MLPs. It is on average 7.93% better than the best LLP model. Notably, it achieves at least 90% of the best GNN effectiveness on 7 out of 10 datasets and even surpasses the best GNN on 4 datasets. For the two datasets where EHDM and LLP perform suboptimally (i.e., `Collab` and `Citation2`), we attribute this to their simple node features, as both datasets use only 128-dimensional node embeddings, the smallest feature size among all datasets.

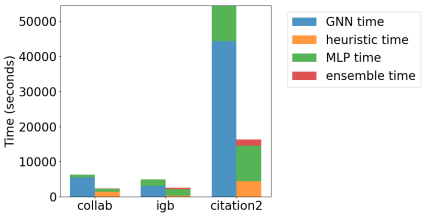


Figure 7: Training time decomposition of SAGE-LLP and EHDM.

6 Conclusion

In this paper, we demonstrate that simple heuristic methods, despite their lower accuracy, can be surprisingly effective teachers for MLPs, enabling competitive link prediction performance while drastically reducing training costs. We provide both empirical and theoretical analysis to explain this observation. Furthermore, we introduce an ensemble approach that aggregates multiple heuristic-distilled MLPs using a gating mechanism. Extensive experiments show that this approach substantially reduces training time while consistently improving prediction accuracy. Our findings have important implications for real-world link prediction tasks, particularly in large-scale web applications.

7 Acknowledgement

This work was partially supported by NSF 2211557, NSF 2119643, NSF 2303037, NSF 2312501, NSF 2531008, SRC JUMP 2.0 Center, Amazon Research Awards, and Snapchat Gifts.

References

[1] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer, 2018. 1, 3

[2] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. *arXiv preprint arXiv:1906.01195*, 2019.

[3] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*, 2019. 1

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. 1

[5] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018. 3

[6] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020. 1

[7] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 1, 8, 15

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 4, 8, 15

[9] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 1, 4, 8, 15

[10] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016. 1, 3

[11] Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018. 1, 3

[12] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018. 1

[13] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old mlps new tricks via distillation. *arXiv preprint arXiv:2110.08727*, 2021. 1, 2, 3

[14] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34:29476–29490, 2021.

[15] Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 34:13683–13694, 2021. 3

[16] Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *The Twelfth International Conference on Learning Representations*, 2024. 3, 4, 18

[17] Li Ma, Haoyu Han, Juanhui Li, Harry Shomer, Hui Liu, Xiaofeng Gao, and Jiliang Tang. Mixture of link predictors on graphs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 1, 3, 6, 7, 8, 18

[18] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems*, 34:9061–9073, 2021. 1, 3, 4, 19

[19] Hongkuan Zhou, Ajitesh Srivastava, Hanqing Zeng, Rajgopal Kannan, and Viktor Prasanna. Accelerating large scale real-time gnn inference using channel pruning. *arXiv preprint arXiv:2105.04528*, 2021. 2, 3

[20] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *International conference on machine learning*, pages 1695–1706. PMLR, 2021. 2, 3

[21] Yiren Zhao, Duo Wang, Daniel Bates, Robert Mullins, Mateja Jamnik, and Pietro Lio. Learned low precision graph neural networks. *arXiv preprint arXiv:2009.09232*, 2020. 2, 3

[22] Shyam A Tailor, Javier Fernandez-Marques, and Nicholas D Lane. Degree-quant: Quantization-aware training for graph neural networks. *arXiv preprint arXiv:2008.05000*, 2020. 2, 3

[23] Yijun Tian, Chuxu Zhang, Zhichun Guo, Xiangliang Zhang, and Nitesh Chawla. Learning mlps on graphs: A unified view of effectiveness, robustness, and efficiency. In *The Eleventh International Conference on Learning Representations*, 2022. 2, 3

[24] Lirong Wu, Haitao Lin, Yufei Huang, Tianyu Fan, and Stan Z Li. Extracting low-/high-frequency knowledge from graph neural networks and injecting it into mlps: An effective gnn-to-mlp distillation framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10351–10360, 2023.

[25] Lirong Wu, Haitao Lin, Yufei Huang, and Stan Z Li. Quantifying the knowledge in gnns for reliable distillation into mlps. In *International Conference on Machine Learning*, pages 37571–37581. PMLR, 2023.

[26] Lirong Wu, Yunfan Liu, Haitao Lin, Yufei Huang, and Stan Z Li. Teach harder, learn poorer: Rethinking hard sample distillation for gnn-to-mlp knowledge distillation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2554–2563, 2024. 4

[27] Weigang Lu, Ziyu Guan, Wei Zhao, and Yaming Yang. Adagmlp: Adaboosting gnn-to-mlp knowledge distillation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2060–2071, 2024. 3

[28] Zehong Wang, Zheyuan Zhang, Chuxu Zhang, and Yanfang Ye. Training mlps on graphs without supervision. In *The 18th ACM International Conference on Web Search and Data Mining*, 2025. 2, 3

[29] Zhichun Guo, William Shiao, Shichang Zhang, Yozen Liu, Nitesh V Chawla, Neil Shah, and Tong Zhao. Linkless link prediction via relational distillation. In *International Conference on Machine Learning*, pages 12012–12033. PMLR, 2023. 2, 3, 8, 15, 16, 19

[30] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020. 2

[31] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001. 3, 8, 16

[32] Yohsuke Murase, Hang-Hyun Jo, János Török, János Kertész, and Kimmo Kaski. Structural transition in social networks: The role of homophily. *Scientific reports*, 9(1):4310, 2019.

[33] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71:623–630, 2009. 2, 3, 8, 16

[34] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 3, 16, 18

[35] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017. 3

[36] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. *arXiv preprint arXiv:2209.15486*, 2022. 3, 4, 18, 19

[37] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34:29476–29490, 2021.

[38] Kaiwen Dong, Zhichun Guo, and Nitesh V Chawla. Pure message passing can estimate common neighbor for link prediction. *arXiv preprint arXiv:2309.00976*, 2023. 3

[39] Haitao Mao, Juanhui Li, Harry Shomer, Bingheng Li, Wenqi Fan, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Revisiting link prediction: A data perspective. *arXiv preprint arXiv:2310.00793*, 2023. 3, 5, 6, 7, 18

[40] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3): 211–230, 2003. 3, 8, 16, 19

[41] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543, 2002. 3

[42] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953. 3, 18

[43] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998. 3

[44] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, 2003. 3, 8, 16

[45] Kazi Zainab Khanam, Gautam Srivastava, and Vijay Mago. The homophily principle in social network analysis: A survey. *Multimedia Tools and Applications*, 82(6):8811–8854, 2023. 3

[46] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015. 3

[47] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR, 2015. 3

[48] Aref Jafari, Mehdi Rezagholizadeh, Pranav Sharma, and Ali Ghodsi. Annealing knowledge distillation. *arXiv preprint arXiv:2104.07163*, 2021. 4

[49] Yichen Zhu and Yi Wang. Student customized knowledge distillation: Bridging the gap between student and teacher. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5057–5066, 2021.

[50] Zengyu Qiu, Xinzhu Ma, Kunlin Yang, Chunya Liu, Jun Hou, Shuai Yi, and Wanli Ouyang. Better teacher better student: Dynamic prior knowledge for knowledge distillation. *arXiv preprint arXiv:2206.06067*, 2022. 4

[51] 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. *Advances in neural information processing systems*, 33:22118–22133, 2020. 5, 15

[52] Arpandeep Khatua, Vikram Sharma Mailthody, Bhagyashree Taleka, Tengfei Ma, Xiang Song, and Wen-mei Hwu. Igb: Addressing the gaps in labeling, features, heterogeneity, and size of public graph datasets for deep learning research. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4284–4295, 2023. 5, 15

[53] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016. 15

[54] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018. 15

[55] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015. 15

[56] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020. 15

A LLP framework

LLP captures relational knowledge by defining an anchor node v and a context set C_v, training the student MLP to learn the relative ranking of context nodes based on their connection probability with v. Let $p_{v,i}$ and $q_{v,i}$ denote the teacher GNN's and student MLP's predicted probabilities of link (v, i), respectively. In addition to cross entropy loss with ground-truth edge labels, LLP employs two distillation losses:

- **Ranking-Based Loss** (L_R), ensuring the student MLP preserves the teacher's ranking order:

$$L_R = \sum_{v \in \mathcal{V}} \sum_{i,j \in C_v} \max(0, -r \cdot (q_{v,i} - q_{v,j}) + \delta) \tag{8}$$

  where r is defined as:

$$r = \begin{cases} 1, & \text{if } p_{v,i} - p_{v,j} > \delta \\ -1, & \text{if } p_{v,i} - p_{v,j} < -\delta \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

  with hyperparameter δ controlling ranking sensitivity.
- **Distribution-Based Loss** (L_D), aligning student MLP predictions with the teacher GNN with temperature hyperparameter t:

$$L_D = \sum_{v \in \mathcal{V}} \sum_{i \in C_v} \frac{\exp(\frac{p_{v,i}}{t})}{\sum_{j \in C_v} \exp(\frac{p_{v,j}}{t})} \log \frac{exp(\frac{q_{v,i}}{t})}{\sum_{j \in C_v} \exp(\frac{q_{v,j}}{t})} \tag{10}$$

B Proof of Lemma 3.1

Proof. Expanding the first objective using the definition of KL divergence:

$$\begin{aligned}
&\mathbb{E}_{x_i,x_j,s_i,s_j}\left[\mathrm{KL}(F \,\|\, g)\right] \\
&= \mathbb{E}_{x_i,x_j,s_i,s_j}\left[\sum_y F(y \mid x_i, x_j, s_i, s_j) \log \frac{F(y \mid x_i, x_j, s_i, s_j)}{g(y \mid x_i, x_j)}\right] \\
&= \mathbb{E}_{x_i,x_j}\left[\sum_y \left(\mathbb{E}_{s_i,s_j|x_i,x_j}[F(y \mid x_i, x_j, s_i, s_j)] \cdot \log \frac{\mathbb{E}_{s_i,s_j|x_i,x_j}[F(y \mid \cdot)]}{g(y \mid x_i, x_j)}\right)\right] + C \\
&= \mathbb{E}_{x_i,x_j}\left[\mathrm{KL}\left(\mathbb{E}[F|x_i, x_j] \,\|\, g(y \mid x_i, x_j)\right)\right] + C,
\end{aligned}$$

where C is a constant independent of g, arising from the entropy terms of F.

Meanwhile, since $\mathbb{E}[F|x_i, x_j]$ are g are both independent to s_i, s_j, so

$$\mathbb{E}_{x_i,x_j,s_i,s_j} KL(\mathbb{E}[F|x_i, x_j], g) = \mathbb{E}_{x_i,x_j} KL(\mathbb{E}[F|x_i, x_j], g)$$

Therefore, the two objectives are equivalent. □

C Proof of Theorem 3.2

Proof. The teachable knowledge of $F_{\text{GNN4LP-MLP}}$ is

$$\mathbb{E}_{s^{\text{GNN}}, s^{\text{extra}}}\left[F_{\text{GNN4LP}} \mid \boldsymbol{x}_i, \boldsymbol{x}_j\right] \tag{11}$$

Let a GNN function be

$$F_{\text{GNN}}(\boldsymbol{x}_i, \boldsymbol{x}_j, s_i^{\text{GNN}}, s_j^{\text{GNN}}) = \mathbb{E}[F_{GNN4LP}|\boldsymbol{x}_i, \boldsymbol{x}_j, s_i, s_j]$$

Then its teachable knowledge

$$\mathbb{E}[F_{GNN}|\boldsymbol{x}_i, \boldsymbol{x}_j] = \mathbb{E}\{\mathbb{E}[F_{GNN4LP}|\boldsymbol{x}_i, \boldsymbol{x}_j, s_i, s_j] \mid \boldsymbol{x}_i, \boldsymbol{x}_j\}$$

With the tower property of the expectation, i.e. $\mathbb{E}(f|X) = \mathbb{E}(\mathbb{E}(f|X, Y)|X)$, we know that

$$\mathbb{E}[F_{GNN}|\boldsymbol{x}_i, \boldsymbol{x}_j] = \mathbb{E}[F_{GNN4LP}|\boldsymbol{x}_i, \boldsymbol{x}_j] \tag{12}$$

Therefore we prove that for any GNN4LP we can find a GNN teacher to have the same teachable knowledge. □

D Computation Steps of Toy Example

Let x_i, x_j, s_i, s_j all be binary random variables. And $Y_{ij} = 1$ if and only if $s_i = s_j = 1$. Let $x_i, x_j \overset{iid}{\sim} \text{Bern}(0.8)$. Let $s|x \sim \text{Bern}(p_x)$, with

$$p_x = \begin{cases} 0.5, & x_i = x_j = 1, \\ 0.6. & \text{otherwise}, \end{cases} \tag{13}$$

Let us consider two teacher functions. The first teacher $F_1(x_i, x_j, s_i, s_j) = 1$ if and only if $s_i = s_j = 1$, and $F_1 = 0.1$ otherwise. The second teacher F_2 is a constant predictor and $F_2 = 0.3$. So the teachable knowledge of F_1 is $\mathbb{E}[F_1|x_i, x_j] = 0.1 + 0.9p_x^2$. And the teachable knowledge of F_2 is $\mathbb{E}[F_2|x,x_j] = 0.3$.

For a Bernoulli(p_x^2) target and a predictor q, the conditional expected NLL is

$$\mathbb{E}\big[-\ln q(Y) \mid x\big] = p_x^2\big(-\ln q\big) \;+\; (1-p_x^2)\big(-\ln(1-q)\big).$$

The following computation steps are illustrated in Table 4 and Table 5.

Table 4: Conditional expected NLL given x. Here $p_x = 0.5$ if $(x_i, x_j) = (1, 1)$ (prob. 0.64), else $p_x = 0.6$ (prob. 0.36).

Region	Predictor	q	$\mathbb{E}[-\ln q(Y) \mid x]$
$(1,1)$, $p_x^2 = 0.25$	F_1	$q_1 = 1\,(Y=1),\ 0.1\,(Y=0)$	$(1-0.25)(-\ln 0.9) \approx 0.0790$
	F_2	$q_2 = 0.3$	$0.25(-\ln 0.3) + 0.75(-\ln 0.7) \approx 0.5685$
	$\mathbb{E}F_1$	$q_{g1} = 0.1 + 0.9 \cdot 0.25 = 0.325$	≈ 0.5758
	$\mathbb{E}F_2$	$q_{g2} = 0.3$	≈ 0.5685
otherwise, $p_x^2 = 0.36$	F_1	$q_1 = 0.1$	$(1-0.36)(-\ln 0.9) \approx 0.0674$
	F_2	$q_2 = 0.3$	$0.36(-\ln 0.3) + 0.64(-\ln 0.7) \approx 0.6617$
	$\mathbb{E}F_1$	$q_{g1} = 0.1 + 0.9 \cdot 0.36 = 0.424$	≈ 0.6619
	$\mathbb{E}F_2$	$q_{g2} = 0.3$	≈ 0.6617

Table 5: Unconditional expected NLL: $\Pr((1,1)) = 0.64,\ \Pr(\text{otherwise}) = 0.36.$

Predictor	$\mathbb{E}[-\ln q(Y)]$
F_1	$0.64 \cdot 0.0790 + 0.36 \cdot 0.0674 \approx 0.0748$
F_2	$0.64 \cdot 0.5685 + 0.36 \cdot 0.6617 \approx 0.6021$
$\mathbb{E}F_1$	$0.64 \cdot 0.5758 + 0.36 \cdot 0.6619 \approx 0.6068$
$\mathbb{E}F_2$	same as Teacher 2: ≈ 0.6021

E Experiment Setting

Datasets. Following previous works [29], we include ten datasets for a comprehensive evaluation, i.e., `Cora`, `Citeseer`, `Pubmed` [53], `Coauthor-CS`, `Coauthor-Physics` [54], `Amazon-Computers`, `Amazon-photos` [54, 55], `OGBL-Collab`, `OGBL-Citation2` [56], and `IGB` [52]. Dataset statistics are summarized in Table 6 in the appendix. Notably, `OGBL-Collab` contains over a million edges, while `IGB` and `OGBL-Citation2` have over a million nodes and tens of millions of edges.

Evaluation Protocol. For `Collab` and `Citation2` datasets, we use the same train/test split as in the OGBL benchmark [51], where the edges are split according to time to simulate real-world production settings. For all other datasets, we randomly sample 5%/15% of the edges along with an equal number of non-edge node pairs, as validation/test sets. Unlike LLP [29], which trains the teacher GNN with ten different random seeds and selects the model with the highest validation accuracy, we train the teacher GNN only once for distillation. Our approach better reflects practical constraints, as training a GNN ten times is infeasible for real-world applications. For each dataset, we train three teacher GNNs: GCN [7], SAGE [8], and GAT [9], and report the highest-performing one, denoted as "*Best GNN*". Similarly, for the LLP baseline [29], we distill three MLPs from GCN, SAGE, and GAT, respectively, and report the best-performing one, denoted as "*Best LLP*". We include four

heuristic methods: CN [31], AA [40], RA [33], and CSP [44]. For each dataset, all MLPs have same architecture. For Collab we report Hits@50, for IGB we report Hits@100, for Citation2 we report MRR, for other datasets we report Hits@20.

F Hyper-parameters

Table 6: Statistics of datasets.

Dataset	# Nodes	# Edges	# Features
Cora	2,708	5,278	1,433
Citeseer	3,327	4,552	3,703
Pubmed	19,717	44,324	500
Coauthor-CS	18,333	163,788	6,805
Coauthor-Physics	34,493	495,924	8,415
Computers	13,752	491,722	767
Photos	7,650	238,162	745
OGBL-Collab	235,868	1,285,465	128
IGB	1M	12M	1,024
OGBL-Citation2	2.9M	30.6M	128

For capped shortest path (CSP) heuristic, we set the upper bound $\tau = 6$ for seven smaller datasets and $\tau = 2$ for the three larger datasets. When distilling MLPs from teacher models (GNNs or heuristic methods), we follow the same hyperparameter configurations as in [29] for sampling context nodes and adopt the configurations from [34] for training teacher GNNs and non-distilled MLPs. We use 3-layer MLPs for `OGBL-Collab`, `IGB`, and `OGBL-Citation2`, and 2-layer MLPs for all other datasets. Let α, β denote the loss weights of L_R and L_D, respectively. After distillation, we perform a grid search to determine the loss weights α and β, where $\alpha, \beta \in \{0, 0.001, 1, 10\}$. Additionally, we use grid search to optimize the margin δ in L_R (Eq. 8), where $\delta \in \{0.05, 0.1, 0.2\}$. For training ensemble models, we conduct a grid search to find the optimal weight λ for L1 regularization (Eq. 7), where $\lambda \in \{0, 0.1, 1\}$.

G Additional Experiment Results

G.1 Inference Speed

One of the primary advantages of our approach is its fast inference training speed. Figure 7 in our paper has shown its advantage in training speed. And prior work, such as LLP, has demonstrated that GNN-to-MLP methods can achieve up to 70 times faster inference compared to GNNs. Our experimental results, as shown in Table 7, confirm that EHDM maintains similar inference speed advantages.

Table 7: Inference time (in ms) for different models on `Collab` dataset.

Model	SAGE	SAGE-LLP	EHDM
Inference Time (ms)	134.3	2.1	3.6

G.2 Breakdown of Best GNN and Best LLP

Table 8 and Table 9 shows the detailed breakdowns of "best GNN" and "best LLP", respectively. We observe that different datasets have different best GNNs and different best LLPs. So when compared with a fixed teacher GNN, our proposed method will have even better improvement than the number reported in the main paper.

Table 8: Breakdown of teacher performance across datasets, showing GCN, SAGE, GAT, and the best-performing GNN. All values are given as mean ± standard deviation.

Dataset	GCN	SAGE	GAT	best GNN
cora	73.96 ± 1.23	71.80 ± 3.08	71.77 ± 1.44	73.96 ± 1.23
citeseer	79.91 ± 2.52	85.10 ± 2.25	80.48 ± 2.21	85.10 ± 2.25
pubmed	68.83 ± 2.84	62.45 ± 3.11	56.15 ± 3.09	68.83 ± 2.84
cs	64.24 ± 3.52	51.20 ± 2.61	59.17 ± 4.15	64.24 ± 3.52
physics	62.25 ± 3.75	65.85 ± 2.82	45.03 ± 6.55	65.85 ± 2.82
computers	27.07 ± 4.45	26.00 ± 3.09	9.82 ± 1.82	27.07 ± 4.45
photos	49.37 ± 1.18	49.79 ± 8.87	44.45 ± 2.13	49.79 ± 8.87
collab	56.75 ± 1.39	59.14 ± 1.64	55.90 ± 1.22	59.14 ± 1.64
IGB	20.47 ± 0.82	20.38 ± 1.39	OOM	20.47 ± 0.82
citation2	84.90 ± 0.06	82.92 ± 0.22	OOM	84.90 ± 0.06

Table 9: Breakdown of student MLP performance distilled from GCN, SAGE, and GAT, along with the best-performing LLP (best LLP). All values are mean ± standard deviation.

Dataset	GCN-LLP	SAGE-LLP	GAT-LLP	best LLP
cora	76.62 ± 2.00	75.26 ± 3.22	74.54 ± 4.34	76.62 ± 2.00
citeseer	76.53 ± 4.88	73.27 ± 4.11	74.95 ± 3.14	76.53 ± 4.88
pubmed	59.95 ± 1.58	54.07 ± 4.95	58.30 ± 3.66	59.95 ± 1.58
cs	66.64 ± 1.80	65.50 ± 3.91	63.65 ± 1.97	66.64 ± 1.80
physics	60.19 ± 2.93	58.46 ± 1.85	56.83 ± 3.81	60.19 ± 2.93
computers	23.66 ± 1.61	25.80 ± 4.80	23.51 ± 2.14	25.80 ± 4.80
photos	39.66 ± 2.94	34.72 ± 3.34	35.86 ± 4.65	39.66 ± 2.94
collab	49.30 ± 0.79	47.99 ± 0.62	48.45 ± 0.82	49.30 ± 0.79
IGB	25.12 ± 1.14	24.33 ± 0.48	NA	25.12 ± 1.14
citation2	42.78 ± 0.10	42.62 ± 0.08	NA	42.78 ± 0.10

G.3 Comparision of GNN4LP and GNN methods

Below we show a more comprehensive comparison between NCN and standard GNNs across seven datasets, including two million-scale graphs. As shown in the Table 10 and Table 11, while NCN consistently achieves the highest accuracy as a teacher, the MLPs distilled from standard GNNs outperform those distilled from NCN in 6 out of 7 cases. Furthermore, our proposed EHDM model consistently achieves the best student performance, while also training 3 times faster.

Table 10: Teacher Model Performance

Dataset	GCN	SAGE	GAT	NCN
cora	73.96 ± 1.23	71.80 ± 3.08	71.77 ± 1.44	83.22 ± 1.37
citeseer	79.91 ± 2.52	85.10 ± 2.25	80.48 ± 2.21	85.45 ± 0.73
pubmed	68.83 ± 2.84	62.45 ± 3.11	56.15 ± 3.09	72.47 ± 1.86
CS	64.24 ± 3.52	51.20 ± 2.61	59.17 ± 4.15	74.49 ± 2.37
computers	27.07 ± 4.45	27.07 ± 4.45	26.00 ± 3.09	39.49 ± 2.30
coauthor	56.75 ± 1.39	59.14 ± 1.64	55.90 ± 1.22	63.07
citation2	84.90 ± 0.06	82.92 ± 0.22	OOM	89.11

Table 11: Distilled Model Performance

Dataset	GCN-MLP	SAGE-MLP	GAT-MLP	NCN-MLP	EHDM
cora	76.62 ± 2.00	75.26 ± 3.22	74.54 ± 4.34	69.37 ± 2.10	80.49 ± 1.51
citeseer	76.53 ± 4.88	73.27 ± 4.11	74.95 ± 3.14	79.29 ± 2.79	79.08 ± 1.90
pubmed	59.95 ± 1.58	54.07 ± 4.95	58.30 ± 3.66	59.85 ± 1.31	60.75 ± 3.42
CS	66.64 ± 1.80	65.50 ± 3.91	63.65 ± 1.97	58.82 ± 4.01	74.33 ± 2.59
computers	23.66 ± 1.61	25.80 ± 4.80	23.51 ± 2.14	23.01 ± 6.21	30.41 ± 2.90
collab	49.30 ± 0.79	48.00 ± 0.62	48.45 ± 0.82	48.00 ± 1.79	49.27 ± 0.88
citation2	42.78 ± 0.10	42.62 ± 0.08	NA	42.58 ± 2.10	46.58 ± 0.10

G.4 Additional GNN4LP Teachers

Table 12 shows more performance of GNN4LP models and their distilled MLPs. Specifically, we evaluate NCN [16], NCNC [16], and BUDDY [36]. The results further validate our finding that stronger models might not be better teachers.

Table 12: Performance comparison (Hits@10) of different teacher models and their distilled MLP students across datasets.

	Standard GNN			GNN4LP		
Dataset	GCN	SAGE	GAT	NCN	NCNC	BUDDY
Teacher Models						
Cora	66.11 ± 4.93	64.82 ± 4.48	61.97 ± 6.47	74.50 ± 1.64	76.59 ± 3.58	56.24 ± 3.44
Citeseer	73.41 ± 2.89	77.45 ± 3.60	75.91 ± 1.84	80.18 ± 1.76	83.61 ± 1.14	79.34 ± 2.03
Pubmed	54.60 ± 3.98	44.70 ± 6.03	42.18 ± 9.92	61.02 ± 3.59	60.74 ± 2.34	48.25 ± 4.30
Student MLPs						
Cora	68.96 ± 2.29	64.29 ± 8.34	66.15 ± 6.58	57.53 ± 5.95	67.32 ± 3.31	65.09 ± 3.36
Citeseer	68.22 ± 3.18	65.80 ± 4.77	70.68 ± 2.92	70.82 ± 3.58	72.91 ± 1.72	66.02 ± 3.25
Pubmed	47.62 ± 3.22	41.17 ± 5.10	43.89 ± 5.16	46.38 ± 3.09	46.73 ± 4.52	44.42 ± 3.15

G.5 Performance of Ensemble Heuristics

Here, we provide more results showing distilling MLPs from ensemble heuristics is sub-optimal. We trained an MLP as a gating function to ensemble different heuristic methods and distilled MLPs from these ensemble heuristics. As shown in Table 13, although ensemble heuristics generally yield higher Hits@K scores, the student MLP distilled from the ensemble heuristics has lower Hits@K. The results suggest that MLPs, limited by their capacity, struggle to effectively learn complex structural information. Consequently, GNNs and GNN4LP models, which integrate various proximities together [17, 39], may also present challenges for MLPs to fully capture.

G.6 Complicated Heuristic Teachers

Notably, MLPs cannot learn well from Katz [42] heuristic either. Katz [42] is a well-established heuristic for link prediction, widely effective across various datasets [34]. Katz is computed as $\sum_{l=1}^{\infty} \lambda^l |path^{(l)}(i,j)|$, where $\lambda < 1$ is a damping factor and $|path^{(l)}(i,j)|$ is the number of length-l paths between i and j. As shown in Table 14, while Katz outperforms CN as a heuristic, the MLP distilled from Katz performs worse than MLPs trained without distillation. It suggests that MLPs cannot learn well from complicated heuristics.

G.7 Effects of the Number of Trainable Parameters

To investigate whether the improvement of the ensemble model is solely due to an increase in trainable parameters, we conduct an ablation study. Specifically, we train an additional MLP, denoted as MLP*, with the same number of trainable parameters as the ensemble MLP. Table 15 compares the performance of MLP, MLP*, and our ensemble method. The results show that the ensemble

Table 13: Performance comparison of different heuristic methods and their corresponding MLP-distilled versions across datasets. CC denotes "CN+CSP", CARC denotes "CC+AA+RA+CSP". **Bolded numbers** mark the highest performance. Underlined numbers represent the second highest performance.

Dataset	CN	CSP	CC	CARC
Teacher				
Cora	42.69	42.69	60.34	**60.76**
Citeseer	35.16	58.02	59.21	**69.10**
Coauthor-CS	53.65	0	74.90	**77.17**
Computers	20.38	0	22.27	**33.87**
Student MLP				
Cora	**75.75**	74.16	69.60	69.49
Citeseer	75.69	**77.10**	75.74	72.26
Coauthor-CS	**67.53**	67.41	62.21	62.33
Computers	**27.55**	21.97	21.95	19.12

Table 14: Hits@10 of CN, Katz, and MLP models across different datasets.

Dataset	CN	Katz	MLP	CN MLP	Katz MLP
Cora	42.69	51.61	58.48	62.16	55.33
Citeseer	35.16	57.36	61.21	70.51	66.64
Pubmed	27.93	42.17	38.21	48.43	37.45

method consistently outperforms MLP*, demonstrating that its effectiveness is not merely a result of increased parameter capacity.

Table 15: Performance comparison of MLP, MLP*, and Ensemble across different datasets.

Dataset	MLP	MLP*	Ensemble
Cora	73.21 ± 3.28	71.05 ± 2.02	80.49 ± 1.51
Citeseer	68.26 ± 1.92	68.57 ± 3.28	79.08 ± 1.90
Pubmed	49.40 ± 3.53	57.10 ± 2.56	69.75 ± 3.42
IGB	17.74 ± 1.20	20.92 ± 0.92	27.27 ± 0.27

G.8 Problems with "Production" Setting

Guo et al. [29] introduced a "production" setting for splitting train/test data in link prediction tasks, where a random subset of nodes is treated as newly added to the graph. These nodes and their associated edges are removed from the training graph. Their results showed a significant performance drop when transitioning from the "transductive" setting to the "production" setting. Similarly, we observe the same performance decline for heuristic-distilled MLPs, to the extent that heuristic distillation may fail to improve MLP performance.

Our analysis reveals that this performance drop is primarily due to the "production" split significantly altering the graph structure, leading to inaccurate heuristic values in the training graph. For instance, triangle counts play a crucial role in link prediction [18, 36, 40]. As shown in Table 16, the number of triangles in the training graphs is drastically reduced under the "production" setting compared to the "transductive" setting. This discrepancy indicates that the removing random nodes make the graphs more "broken" and that "production" setting introduces a substantial structural mismatch between training and test graphs, explaining the performance degradation observed in [29].

In practice, such drastic structural changes are unlikely in real-world applications. Therefore, we argue that the "production" setting does not accurately reflect real-world scenarios. Instead, datasets like `Collab` and `Citation2`, which split edges based on time, offer a more realistic evaluation. Consequently, we do not adopt the "production" setting in our experiments.

Table 16: Triangle counts and the number of connected components (#CC) in training graph under different split settings.

	Cora		Pubmed	
	Triangle Counts	#CC	Triangle Counts	#CC
Train graph (production)	31	898	3,372	3,407
Train graph (transductive)	991	170	7,633	1,512
Original graph	1,630	78	12,520	1

H Limitations

Since MLPs rely heavily on rich node features to perform effectively, our experiments focus primarily on datasets from social networks and recommendation systems, where such features are abundant. In contrast, biological graphs, such as protein-protein interaction networks, are not included in our evaluation, as they typically lack informative node features, making them less suitable for MLP-based approaches. This limitation is not specific to our method but is inherent to all MLP distillation methods on graphs.