Linkless Link Prediction via Relational Distillation

Zhichun Guo [*1] **William Shiao** [2] **Shichang Zhang** [3] **Yozen Liu** [4] **Nitesh V. Chawla** [1] **Neil Shah** [4] **Tong Zhao** [4]

Abstract

Graph Neural Networks (GNNs) have shown exceptional performance in the task of link prediction. Despite their effectiveness, the high latency brought by non-trivial neighborhood data dependency limits GNNs in practical deployments. Conversely, the known efficient MLPs are much less effective than GNNs due to the lack of relational knowledge. In this work, to combine the advantages of GNNs and MLPs, we start with exploring direct knowledge distillation (KD) methods for link prediction, i.e., predicted logit-based matching and node representation-based matching. Upon observing direct KD analogs do not perform well for link prediction, we propose a relational KD framework, *Linkless Link Prediction* (LLP), to distill knowledge for link prediction with MLPs. Unlike simple KD methods that match independent link logits or node representations, LLP distills *relational* knowledge that is centered around each (anchor) node to the student MLP. Specifically, we propose rank-based matching and distribution-based matching strategies that complement each other. Extensive experiments demonstrate that LLP boosts the link prediction performance of MLPs with significant margins, and even outperforms the teacher GNNs on 7 out of 8 benchmarks. LLP also achieves a 70.68× speedup in link prediction inference compared to GNNs on the large-scale OGB dataset.

1. Introduction

Graph neural networks (GNNs) have been widely used for machine learning on graph-structured data (Kipf & Welling, 2016a; Hamilton et al., 2017). They have shown significant performance in various applications, such as node classification (Veličković et al., 2017; Chen et al., 2020), graph classification (Zhang et al., 2018), graph generation (You et al., 2018; Shiao & Papalexakis, 2021), and link prediction (Zhang & Chen, 2018; Shiao et al., 2023).

Of these, link prediction is a notably critical problem in the graph machine learning community, which aims to predict the likelihood of any two nodes forming a link. It has broad practical applications such as knowledge graph completion (Schlichtkrull et al., 2018; Nathani et al., 2019; Vashishth et al., 2020), friend recommendation on social platforms (Sankar et al., 2021; Tang et al., 2022; Fan et al., 2022) and item recommendation for users on service and commerce platforms (Koren et al., 2009; Ying et al., 2018a; He et al., 2020). With the rising popularity of GNNs, state-of-the-art link prediction methods adopt encoder-decoder style models, where encoders are GNNs, and decoders are applied directly on pairs of node representations learned by the GNNs (Kipf & Welling, 2016b; Zhang & Chen, 2018; Cai & Ji, 2020; Zhao et al., 2022b).

The success of GNNs is typically attributed to the explicit use of contextual information from nodes' surrounding neighborhoods (Zhang et al., 2020e). However, this induces a heavy reliance on neighborhood fetching and aggregation schemes, which can lead to high time cost in training and inference compared to tabular models, such as multi-layer perceptrons (MLPs), especially owing to neighbor explosion (Zhang et al., 2020b; Jia et al., 2020; Zhang et al., 2021b; Zeng et al., 2019). Compared to GNNs, MLPs do not require any graph topology information, making them more suitable for new or isolated nodes (e.g., for cold-start settings), but usually resulting in worse general task performance as encoders, which we also empirically validate in Section 4. Nonetheless, having no graph dependency makes the training and inference time for MLPs negligible when comparing with those of GNNs. Thus, in large-scale applications where fast real-time inference is required, MLPs are still a leading option (Zhang et al., 2021b; Covington et al., 2016; Gholami et al., 2021).

Given these speed-performance tradeoffs, several recent works propose to transfer the learned knowledge from GNNs to MLP using knowledge distillation (KD) techniques (Hinton et al., 2015; Zhang et al., 2021b; Zheng et al.,

[*]This work was done during first author's internship at Snap Inc. [1]Department of Computer Science and Engineering, University of Notre Dame, IN, USA [2]Department of Computer Science and Engineering, University of California, Riverside, CA, USA [3]Department of Computer Science, University of California, Los Angeles, CA, USA [4]Snap Inc., CA, USA. Correspondence to: Zhichun Guo <zguo5@nd.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

2021; Hu et al., 2021), to take advantage of both GNN's performance benefits and MLP's speed benefits. Specifically, in this way, the student MLP can potentially obtain the graph-context knowledge transferred from the GNN teacher via KD to not only perform better in practice, but also enjoy model latency benefits compared to GNNs, e.g. in production inference settings. However, these works focus on node- or graph-level tasks. Given that KD on link prediction tasks have not been explored, and the massive scope of recommendation systems contexts that are posed as link prediction problems, our work aims to bridge a critical gap. Specifically, we ask:

Can we effectively distill link prediction-relevant knowledge from GNNs to MLPs?

In this work, we focus on exploring, building upon, and proposing cross-model (GNN to MLP) distillation techniques for link prediction settings. We start with exploring two direct KD methods of aligning student and teacher: (i) logit-based matching of predicted link existence probabilities (Hinton et al., 2015), and (ii) representation-based matching of the generated latent node representations (Gou et al., 2021). However, empirically we observe that neither the logit-based matching nor the representation-based matching are powerful enough to distill sufficient knowledge for the student model to perform well on link prediction tasks. We hypothesize that the reason of these two KD approaches not performing well is that link prediction, unlike node classification, heavily relies on *relational* graph topological information (Martínez et al., 2016; Zhang & Chen, 2018; Yun et al., 2021; Zhao et al., 2022b), which is not well-captured by direct methods.

To address this issue, we propose a relational KD framework, namely LLP: our key intuition is that instead of focusing on matching individual node pairs or node representations, we focus on matching the relationships between each (anchor) node with respect to other (context) nodes in the graph. Given the relational knowledge centered at the anchor node, i.e., the teacher model's predicted link existence probabilities between the anchor node and each context node, LLP distills it to the student model via two matching methods: (i) rank-based matching, and (ii) distribution-based matching. More specifically, rank-based matching equips the student model with a ranking loss over the relative ranks of all context nodes w.r.t the anchor node, preserving crucial ranking information that are directly relevant to downstream link prediction use-cases, e.g. user-contextual friend recommendation (Sankar et al., 2021; Tang et al., 2022) or item recommendation (Ying et al., 2018a; He et al., 2020). On the other hand, distribution-based matching equips the student model with the link probability distribution over context nodes, conditioned on the anchor node. Importantly, distribution-based matching is complementary to rank-based matching, as it provides auxiliary information about the relative values of the probabilities and magnitudes of differences. To comprehensively evaluate the effectiveness of our proposed LLP, we conduct experiments on 8 public benchmarks. In addition to the standard transductive setting for graph tasks, we also design a more realistic setting that mimics realistic (on-line) use-cases for link prediction, which we call the production setting. LLP consistently outperforms stand-alone MLPs by 18.18 points on average under the transductive setting and 12.01 points under the production setting on all the datasets, and matches or outperforms teacher GNNs on **7/8** datasets under the transductive setting. Promisingly, for cold-start nodes, LLP outperforms teacher GNNs and stand-alone MLPs by **25.29** and **9.42** Hits@20 on average, respectively. Finally, LLP infers drastically faster than GNNs, e.g. **70.68**× faster on the large-scale `Collab` dataset.

2. Related Work and Preliminaries

We briefly discuss related work and preliminaries relevant to contextualizing our methods and contributions. Due to space limit, we defer more related work to Appendix A.

Notation. Let $G = (\mathcal{V}, \mathcal{E})$ denote an undirected graph, where $\mathcal{V}$ denotes the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of observed links. $\mathbf{A} \in \{0,1\}^{N \times N}$ denotes the adjacency matrix, where $\mathbf{A}_{i,j} = 1$ if exists an edge $e_{i,j}$ in $\mathcal{E}$ and 0 otherwise. Let the matrix of node features be denoted by $\mathbf{X} \in \mathbb{R}^{N \times F}$, where each row $\boldsymbol{x}_i$ is the F-dim raw feature vector of node i. Given both $\mathcal{E}$ and $\mathbf{A}$ have the validation and test links masked off for link prediction, we use $a_{i,j}$ (different from $\boldsymbol{A}_{i,j}$) to denote the true label of link existence of nodes i and j, which may or may not be visible during training depending on the setting. We use $\mathcal{E}^- = (\mathcal{V} \times \mathcal{V}) \setminus \mathcal{E}$ to denote the no-edge node pairs that are used as negative samples during model training. We denote node representations by $\mathbf{H} \in \mathbb{R}^{N \times D}$, where D is the hidden dimension. In KD context with multiple models, we use $\boldsymbol{h}_i$ and $\hat{\boldsymbol{h}}_i$ to denote node i's representations learned by the teacher and student models, respectively. Similarly, we use $y_{i,j}$ and $\hat{y}_{i,j}$ to denote the predictions for $a_{i,j}$ by the teacher and the student models, respectively.

Link Prediction with GNNs. In this work, we take the commonly used encoder-decoder framework for the link prediction task (Kipf & Welling, 2016b; Berg et al., 2017; Schlichtkrull et al., 2018; Ying et al., 2018a; Davidson et al., 2018; Zhu et al., 2021; Yun et al., 2021; Zhao et al., 2022b), where the GNN-based encoder learns node representations and the decoder predicts link existence probabilities. Most GNNs operate under the message-passing framework, where the model iteratively updates each node i's representation $\boldsymbol{h}_i$ by fetching its neighbors' information. That is, the node's representation in the l-th layer is learned with an aggregation

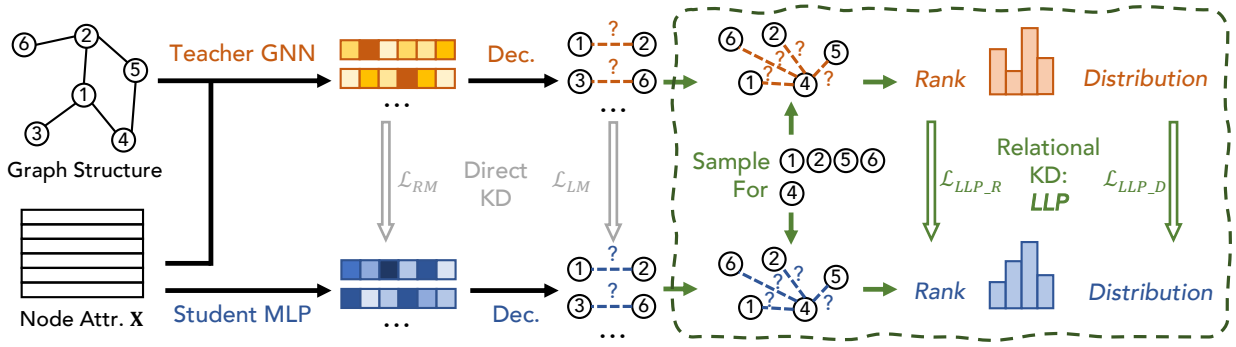

Figure 1: We explore KD methods for link prediction, which distill knowledge from a Teacher GNN to a Student MLP encoder, each with their own decoder. We start by exploring direct KD methods: representation-matching and logit-matching (Section 3.1). Upon observing their drawbacks of not being able to distill relational information, we further propose a relational KD framework: LLP (Section 3.3), which equips the student model with knowledge of each (anchor) node's relationships with other (context) nodes, via our proposed rank-based matching and distribution-based matching objectives.

operation and an update operation:

$$\boldsymbol{h}_i^l = \text{UPDATE}_l\big(\boldsymbol{h}_i^{l-1}, \text{AGGREGATE}_l(\{\boldsymbol{h}_j^{l-1} | e_{i,j} \in \mathcal{E}\})\big), \quad (1)$$

where AGGREGATE combines or pools local neighbor features, UPDATE is a learnable transformation, and $\boldsymbol{h}_i^0 = \boldsymbol{x}_i$. The link prediction decoder takes the node representations from the last layer, i.e., $\boldsymbol{h}_i$ for $i \in \mathcal{V}$, to predict the probability of a link between a node pair i and j:

$$y_{i,j} = \sigma(\text{DECODER}(\boldsymbol{h}_i, \boldsymbol{h}_j)), \quad (2)$$

where σ denotes a Sigmoid operation. In this work, following most state-of-the-art link prediction literature (Zhang et al., 2021a; Tsitsulin et al., 2018; Zhao et al., 2022b; Wang et al., 2021), we take the Hadamard product followed by a MLP as the link prediction DECODER for all methods.

Knowledge Distillation for GNNs. Knowledge distillation (KD) (Hinton et al., 2015) aims to transfer the knowledge from a high-capacity and highly accurate teacher model to a light-weight student model, and is typically employed in resource-constrained settings. KD methods have shown great promise in significantly reducing model complexity, while sometimes barely (or not) sacrificing task performance (Furlanello et al., 2018; Park et al., 2019). As GNNs are known to be slow due to neighbor aggregation induced by data dependency, graph-based KD methods (Zhang et al., 2021b; Zheng et al., 2021) usually distill GNNs onto MLPs, which are commonly used in large-scale industrial applications due to their significantly improved efficiency and scalability. For example, Zheng et al. (2021) proposed a KD-based framework to rediscover the missing graph structure information for MLPs, which improves the models' generalization of node classification task on cold-start nodes. Existing KD methods on graph data mainly focus on node-level (Zheng et al., 2021; Zhang et al., 2021b; Tian et al., 2022) and graph-level tasks (Ma & Mei, 2019; Zhang et al., 2020c; Deng & Zhang, 2021; Joshi et al., 2021), leaving KD for link prediction yet unexplored. Our work focuses on distilling link prediction-relevant information from the GNN teacher to an MLP student, and investigates various KD strategies to align student and teacher predictions. Specifically, denoting the node representations for nodes i and j learned by the student MLP as $\hat{\boldsymbol{h}}_i$ and $\hat{\boldsymbol{h}}_j$, the link existence prediction by the student model can then be written as $\hat{y}_{i,j} = \sigma(\text{DECODER}(\hat{\boldsymbol{h}}_i, \hat{\boldsymbol{h}}_j))$.

3. Cross-model Knowledge Distillation for Link Prediction

In this section, we propose and discuss several approaches to distill knowledge from a teacher GNN to a student MLP in a cross-model fashion, for the purpose of link prediction. In all cases, we aim to supervise the student MLP with artifacts produced by the GNN teacher, in addition to any available training labels ($a_{i,j}$ w.l.o.g.) about link existence. We start by adapting two *direct* knowledge distillation (KD) methods: logit-matching and representation-matching, on link prediction tasks; we call these methods direct because they involve directly matching sample-wise predictions between teacher and student. Next, we motivate and introduce our proposed *relational* KD framework, LLP, with two matching strategies to distill additional topology-related structural information to the student. We call these methods relational because they call for the preservation of relationships across samples between teacher and student (Park et al., 2019). Figure 1 summarizes our proposals.

3.1. Direct Methods

Logit-matching is one straightforward strategy to distill knowledge from the teacher to the student, where it directly aims to teach the student to generalize as the teacher

does on the downstream task. It was proposed by Hinton et al. (2015), and it is still one of the most widely used KD methods in various tasks (Furlanello et al., 2018; Yang et al., 2020b; Yan et al., 2020). Several works (Phuong & Lampert, 2019; Ji & Zhu, 2020) theoretically analyzed its effectiveness. Moreover, it had also been proved to be effective for knowledge transfer on graph data (Yan et al., 2020; Yang et al., 2021; Zhang et al., 2021b) in recent years. For example, Zhang et al. (2021b) the soft logits generated by the teacher GNNs to help supervise the student MLP and achieved strong performance on node classification tasks. In a similar vein, we generate the soft score $y_{i,j}$ for the candidate edge (i, j) with the teacher GNN model, and train the student to match its prediction $\hat{y}_{i,j}$ on this target:

$$\mathcal{L}_{LM} = \sum_{(i,j)\in\mathcal{E}\cup\mathcal{E}^-} \Big(\lambda\mathcal{L}_{sup}(\hat{y}_{i,j}, a_{i,j}) + (1-\lambda)\mathcal{L}_{match}(\hat{y}_{i,j}, y_{i,j})\Big), \tag{3}$$

where $\mathcal{L}_{sup}$ is the supervised link prediction loss (e.g., binary cross entropy) that directly trains the student model, $\mathcal{L}_{match}$ is the loss for matching the student's prediction with the teacher's prediction, and λ is a hyper-parameter that mediates the importance of the ground-truth labels and logit-matching signals. Note that multiple implementation choices exist for $\mathcal{L}_{match}$. For example, mean-squared error (MSE), Kullback-Leibler (KL) divergence, or cosine similarity. In the experiments, we opt for the empirical best choice for fair comparison across methods.

Representation-matching is another direct distillation method in which we aim to align the student's learned latent node embedding space with the teacher's. As this KD training signal only optimizes the encoder part of the student model, it must be used with $\mathcal{L}_{sup}$ so that the student decoder receives a gradient and can also be optimized:

$$\mathcal{L}_{RM} = \sum_{(i,j)\in\mathcal{E}\cup\mathcal{E}^-} \lambda\mathcal{L}_{sup}(\hat{y}_{i,j}, a_{i,j}) + (1-\lambda)\sum_{i\in\mathcal{V}} \mathcal{L}_{match}(\hat{\boldsymbol{h}}_i, \boldsymbol{h}_i). \tag{4}$$

Unlike logit-matching, representation-matching involves directly aligning node-level artifacts, which is similar to object representation matching in computer vision (Romero et al., 2014; Kim et al., 2018; Wang et al., 2020b; Chen et al., 2021a).

3.2. Link Prediction with Relational Distillation

Motivation. The above direct methods ask the student model to directly match node-level or link-level artifacts. However, one might ask: are matching these sufficient for link prediction tasks? This is especially relevant considering that most link prediction applications involve tasks where ranking target nodes with respect to a source, or anchor node, is the task of interest, i.e. ranking relevant candidate users or items with respect to a seed user (Huang et al., 2005; Trouillon et al., 2016). These contexts involve reasoning over multiple *relations* or link-level samples simultaneously, suggesting that matching across these relations could be more aligned with the target link prediction task, compared to the direct node-level or link-level methods.

Furthermore, several works (Zhang & Chen, 2018; Yun et al., 2021) suggest that graph structure information is critically important for link prediction tasks. For example, heuristic link prediction methods commonly show competitive performance compared to GNNs (Zhang & Chen, 2018) and have long-served as a cornerstone for accurate link prediction even prior to neural graph methods (Martínez et al., 2016). Most heuristic methods measure the score of the target node pairs only based on the graph structure information (Barabási & Albert, 1999; Brin & Page, 2012), such as common neighbors and shortest path. In addition, several recent works (Zhang & Chen, 2017; 2018; Li et al., 2020; Zhao et al., 2022b) also show that enclosing topology information such as local subgraph, distances with anchor nodes, or augmented links can largely improve GNNs' performance on link-level tasks. Observing that most successful methods in link prediction involve using relational information other than just the two nodes in question, we also adopt this intuition in the distillation context and propose our relational KD for link prediction. We elaborate next.

3.3. Proposed Framework: Linkless Link Prediction

In accordance with our intuition regarding preservation of relational knowledge, we propose a novel relational distillation framework, called *Linkless Link Prediction*, or LLP. Instead of focusing on matching individual node pair scores or node representations, LLP focuses on distilling knowledge about the relationships of each node to other nodes in the graph; we call the former node an *anchor* node, and the latter nodes *context* nodes. For each node in the graph, when it serves as the anchor node, we aim to equip the student MLP model with knowledge of its relationships with a set of context nodes. Each node can serve as both an anchor node, as well as a context node (for other anchor nodes).

Let v denote the anchor node and $\mathcal{C}_v$ denote the corresponding set of context nodes of v. We denote the teacher model's predicted probabilities of v and each node in $\mathcal{C}_v$ as $\mathcal{Y}_v = \{y_{v,i} | i \in \mathcal{C}_v\}$. Similarly, we denote the student model's predictions on those as $\hat{\mathcal{Y}}_v = \{\hat{y}_{v,i} | i \in \mathcal{C}_v\}$. To effectively distill the relational knowledge from $\mathcal{Y}_v$ to $\hat{\mathcal{Y}}_v$, we proposed two relational matching objectives to train LLP: *rank-based matching* and *distribution-based matching*, which we introduce next.

Rank-based Matching. As aforementioned in Section 3.2, link prediction is often considered a ranking task, requiring the model to rank relevant candidates w.r.t. a seed node, e.g. in a user-item graph setting, the predictor must rank over a set of candidate items from the perspective of a user. Thus, we reason that unlike matching individual and independent logits, matching the ranking induced by the teacher can more straightforwardly teach the student relational knowledge about context nodes w.r.t. the anchor node, e.g. for a specific user, item A should be ranked higher than item C, which should be ranked higher than item B. To adopt this rank-based intuition into a training objective, we adopt a modified margin-based ranking loss that trains the student with the rank of the logits from the teacher GNN. Specifically, we enumerate all pairs of predicted probabilities in $\hat{\mathcal{Y}}_v$ and supervise it with the corresponding pairs in $\mathcal{Y}_v$. That is,

$$\mathcal{L}_{\text{LLP_}R} = \sum_{v \in \mathcal{V}} \sum_{\{\hat{y}_{v,i}, \hat{y}_{v,j}\} \in \hat{\mathcal{Y}}_v} \max(0, -r \cdot (\hat{y}_{v,i} - \hat{y}_{v,j}) + \delta), \tag{5}$$

$$\text{where } r = \begin{cases} 1, & \text{if } y_{v,i} - y_{v,j} > \delta; \\ -1, & \text{if } y_{v,i} - y_{v,j} < -\delta; \\ 0, & \text{otherwise,} \end{cases}$$

where δ is the margin hyper-parameter, which is usually a very small value (e.g. 0.05). Note that the above loss differs from the conventional margin-based ranking loss, because it has a condition for $r = 0$ (inducing constant loss) on the logits pairs that the teacher GNN gives similar probabilities, i.e., $|y_{v,i} - y_{v,j}| < \delta$. This design effectively prevents the student model from trying to differentiate minuscule differences in probabilities which the teacher may produce owing to noise; without this condition, the loss would pass binary information regardless of how small the difference is. We also empirically show the necessity of this design in Table 16 in Appendix D.

Distribution-based Matching. While the rank-based matching can effectively teach the student model relational rank information, we observe that it does not fully make use of the value information from $\mathcal{Y}_v$, e.g. for a specific user, item A should be ranked *much* higher than item C, which should only be ranked *marginally* higher than item B. Although the logit-matching introduced in Section 3.1 might seem suitable here, we observe that its link-level matching strategy only facilitates matching information on scattered node pairs, rather than focusing on the relationships conditioned on an anchor node – empirically, we also find that it has limited effectiveness. Therefore, to enable relational value-based matching centered on the anchor nodes, we further propose a distribution-based matching scheme which utilizes the KL divergence between the teacher predictions $\mathcal{Y}_v$ and student predictions $\hat{\mathcal{Y}}_v$, centered on each anchor node v. Specifically, we define it as

$$\mathcal{L}_{\text{LLP_}D} = \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{C}_v} \frac{\exp(y_{v,i}/\tau)}{\sum_{j \in \mathcal{C}_v} \exp(y_{v,j}/\tau)} \log\left(\frac{\exp(\hat{y}_{v,i}/\tau)}{\sum_{j \in \mathcal{C}_v} \exp(\hat{y}_{v,j}/\tau)}\right), \tag{6}$$

where τ is a temperature hyper-parameter which controls the softness of the softmaxed distribution. By also asking the student to match relative values within the probability distribution over context nodes conditioned on each anchor node, the distribution-based matching scheme complements rank-based matching by providing auxiliary information about the magnitudes of differences.

Practical Implementation of LLP. In practical implementation, given the large number of nodes in the graph, it is infeasible for LLP to use all other nodes as the set of context nodes, especially for the rank-based matching which enumerates pairs of probabilities in $\hat{\mathcal{Y}}_v$. Hence, we opt for simplicity and adopt two straightforward sampling strategies for the constructing $\mathcal{C}_v$ for each anchor node v to limit its size. First, to keep the local structure around the anchor node, we follow previous works (Perozzi et al., 2014; Hamilton et al., 2017) to sample p nearby nodes by repeating fixed-length random walks several times, denoted as $\mathcal{C}_v^N$. Secondly, we randomly sample q nodes from the whole graph G (which are likely to be far-away from v) to form $\mathcal{C}_v^R$, which additionally preserves the global structure w.r.t. v in the graph. The context nodes for each anchor node are the union of the nearby samples and random samples. p and q are hyper-parameters. Finally, we make $\mathcal{C}_v$ as the union of the nearby samples and random samples, i.e., $\mathcal{C}_v = \mathcal{C}_v^N \cup \mathcal{C}_v^R$. We conduct experiments to show the impact of the selection strategy of context nodes $\mathcal{C}_v$ for each anchor node, which are presented in Section 4.6 and Appendix D.5.

While training LLP, we jointly optimize both the rank-based and distribution-based matching losses in addition to the ground-truth label loss. Therefore, the overall training loss which LLP adopts for the student is

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{sup} + \beta \cdot \mathcal{L}_{\text{LLP_}R} + \gamma \cdot \mathcal{L}_{\text{LLP_}D} \tag{7}$$

where α, β, and γ are hyper-parameters which mediate the strengths of each loss term.

4. Experiments

4.1. Experimental Setup

Datasets. We conduct the experiments using 8 commonly used benchmark datasets for link prediction: `Cora`, `Citeseer`, `Pubmed`, `Computers`, `Photos`, `CS`, `Physics`, and `Collab`. The statistics of the datasets are shown in Table 1 with further details provided in Appendix B.

Table 1: 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
CS	18,333	163,788	6,805
Physics	34,493	495,924	8,415
Computers	13,752	491,722	767
Photos	7,650	238,162	745
Collab	235,868	1,285,465	128

Evaluation Settings. To comprehensively evaluate our proposed LLP and baseline methods on the link prediction tasks, we conduct experiments on both transductive and production settings. For the transductive setting, all the nodes in the graph can be observed for train/validation/test sets. Following previous works (Zhang & Chen, 2018; Chami et al., 2019; Cai et al., 2021) we randomly sample 5%/15% of the links with the same number of no-edge node pairs from the graph as the validation/test sets on the non-OGB datasets. And the validation/test links are masked off from the training graph. For the OGB datasets, we follow their official train/validation/test splits (Wang et al., 2020a). In addition to transductive setting, we also design a more realistic setting that mimics practical link prediction use-cases, which we call the production setting. In the production setting, new nodes would appear in the test set, while training and validation sets only observe previously existing nodes. Thus, this setting entails three categories of node pairs (edges or no-edges) in the test set: existing – existing, existing – new, and new – new, where the first category is similar to the test edges in the transductive setting, and the latter two categories together are similar to the inductive setting used in a few recent works (Bojchevski & Günnemann, 2017; Hao et al., 2020; Chen et al., 2021b). Nonetheless, all three types of these edges appear with varying proportions in practical use-cases, e.g. growth of a social network or online platform, hence we evaluate on all three types. Note that we only conduct production setting experiments on non-OGB datasets, because the OGB dataset is already temporally split in their public releases. We further elaborate the details of the production setting in Appendix C.

For Collab, we use its official metric (Hits@50 for Collab) following their public leaderboard. For other datasets, following previous works (Yun et al., 2021; Zhang et al., 2021a; Zhao et al., 2022b), we use Hits@20 as the main metric, which is also one of the main metrics on OGB datasets. We also report AUC performance in Appendix D. For all experiments, we report the averaged test performance (with early-stopping on validation) along with its standard deviation over 10 runs with different random initializations.

Reproducibility. To ensure the reproducibility of LLP, our implementation is publically available at https://github.com/snap-research/linkless-link-prediction/.

Methods. In the remainder of this section: "GNN" indicates the teacher GNN that was trained with $\mathcal{L}_{sup}$; "MLP" refers to the stand-alone MLP that was trained with $\mathcal{L}_{sup}$; "$\mathcal{L}_{LM}$" refers to MLP trained with logit matching (Equation (3)); "$\mathcal{L}_{RM}$" refers to MLP trained with node representation matching (Equation (4)); "LLP" refers to MLP trained with our proposed relational KD (Equation (7)). For the main experiments, we opt for simplicity and use SAGE (Hamilton et al., 2017) as the teacher GNN in all settings. We also include further experiments of different teacher GNN models in Appendices D.3 and D.4.

4.2. Link Prediction Results

Transductive Setting. Table 2 shows the link prediction performance of the proposed LLP with GNN, MLP, and the direct KD methods (as introduced in Section 3.1) in the transductive setting. We observe that LLP consistently outperforms MLP and direct KD methods across all datasets with large margins. Specifically, LLP achieves **18.18** points and **10.59** points improvements over MLP and direct KD methods averaged on datasets, respectively. On the Physics dataset, LLP achieves **40.75** points and **19.67** points absolute improvements over MLP and direct KD, respectively. Moreover, LLP achieves better performance than the teacher GNN model on 7 out of 8 datasets, demonstrating that our proposed rank-based and distribution-based matching are able to effectively distill the knowledge for link prediction.

From Table 2, we also observe significant performance improvements of LLP over the teacher GNNs on some datasets. We hypothesize that there are two reasons leading to such improvements. The one is that the student MLP model already has significant learning ability when the node features are informative enough for link prediction. For example, on the Cora dataset, MLP already can produce better prediction performance than GNN. The other reason is relational KD provides relational structure knowledge from GNNs to MLPs, which provides extra valuable knowledge to MLPs. Recent studies on knowledge distillation (Allen-Zhu & Li, 2020; Guo et al., 2022) also have similar findings that combining different views from different models could help improve the model's performance.

Production Setting. Table 3 shows the link prediction performance of the proposed LLP with GNN, MLP, and the direct KD methods in the production setting. From this table, we observe that LLP is still able to consistently outperform MLP and direct KD methods by large margins for all benchmarks. Specifically, LLP achieves **12.01** and **6.67** on Hits@20 improvements over MLP and direct KD methods averaged over datasets, respectively. Moreover,

Table 2: Link prediction performance under **transductive** setting. For Collab, we report Hits@50. For other datasets, we report Hits@20. Best and second best performances are marked with bold and underline, respectively. $\Delta_{DirectKD}$, Δ_{MLP}, and Δ_{GNN} represent differences between LLP and these methods.

	GNN	MLP	$\mathcal{L}_{LM}$	$\mathcal{L}_{RM}$	LLP	$\Delta_{DirectKD}$	Δ_{MLP}	Δ_{GNN}
Cora	74.38±1.54	78.06±1.50	74.72±4.27	75.75±1.51	**78.82**±1.74	3.07	0.76	4.44
Citeseer	73.89±0.95	71.21±3.22	72.44±1.52	65.19±5.54	**77.32**±2.42	4.88	6.11	3.43
Pubmed	51.98±5.25	42.89±1.67	42.78±3.15	44.44±2.40	**57.33**±2.42	12.89	14.44	5.35
CS	59.51±7.34	34.01±9.37	40.69±5.12	61.10±2.83	**68.62**±1.46	7.52	34.61	9.11
Physics	66.74±1.53	31.26±9.12	52.11±2.44	52.34±3.78	**72.01**±1.89	19.67	40.75	5.27
Computers	31.66±3.08	20.19±1.58	12.81±1.80	21.75±1.96	**35.32**±2.28	13.57	15.31	3.66
Photos	**51.50**±4.48	27.83±4.90	24.24±2.79	38.47±2.76	49.32±2.64	10.85	21.49	-2.18
Collab	48.69±0.87	36.95±1.37	35.97±0.96	36.86±0.45	**49.10**±0.57	12.24	12.15	0.41

Table 3: Link prediction performance measured by Hits@20 under **production** setting. Best and second best performances are marked with bold and underline, respectively.

	GNN	MLP	$\mathcal{L}_{LM}$	$\mathcal{L}_{RM}$	LLP	$\Delta_{DirectKD}$	Δ_{MLP}	Δ_{GNN}
Cora	27.80±2.11	22.90±2.22	22.65±2.51	22.24±0.55	**27.87**±1.24	5.22	4.97	0.07
Citeseer	**38.78**±2.59	31.21±3.75	29.35±2.55	26.23±1.08	34.75±2.45	5.40	3.54	-4.03
Pubmed	52.71±1.81	38.01±1.67	39.03±4.21	43.27±3.12	**53.48**±1.52	10.21	15.47	0.77
CS	60.69±3.17	38.15±10.78	48.07±2.39	58.90±1.32	**60.74**±1.41	1.84	22.59	0.05
Physics	**55.82**±2.43	29.99±1.96	22.74±1.03	36.32±2.29	52.83±1.50	16.51	22.84	-2.99
Computers	**34.38**±1.41	19.43±0.82	12.79±1.43	20.28±1.01	24.58±3.33	4.30	5.15	-9.80
Photos	**51.03**±6.05	34.29±2.49	24.63±2.20	40.58±1.63	43.79±1.27	3.21	9.50	-7.24

LLP is at or above par with the teacher GNN on 5 out of the 7 datasets, which supports deploying LLP is effective to distill the relational knowledge about link prediction from GNNs to MLPs. For ease of comparison, we also stratify each method's performance on the three different categories of the test edges in Table 8 (in Appendix D.1).

In Table 3, we also observe that LLP achieves the in-stable performance across datasets, which is a recurring issue that plagues many related works (Zhang et al., 2021b; Zheng et al., 2021) that study GNN to MLP distillation, especially under production or inductive settings. Under the transductive setting, LLP is able to impart a significant amount of relational knowledge to the student with respect to the nodes that already exist, and the evaluation will also be performed on those existing nodes. However, new nodes will emerge during the evaluation process under the production setting. In this setting, the efficacy of our approach depends on the quality of the original node features. The more informative the node features, the better our method will perform. This phenomenon is also aligned with Table 3 of GLNN (Zhang et al., 2021b), which demonstrates a similar trend under the inductive setting for node classification tasks.

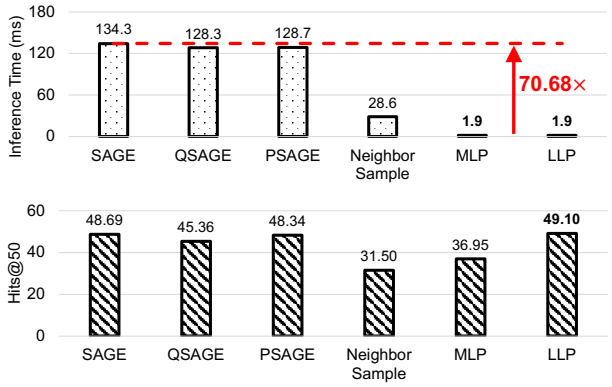

Figure 2: Inference time and prediction performance comparison of LLP with GNN and GNN inference acceleration methods on Collab.

4.3. Inference Acceleration Comparison

We evaluate LLP in comparison to other common GNN inference acceleration methods, which mainly focus on the hardware and algorithm to reduce the computation consuming, such as pruning (Zhou et al., 2021) and quantization (Zhao et al., 2020; Tailor et al., 2020). Following the experimental settings in Zhang et al. (2021b), we measure the inductive inference time on in the graph. We evaluate against 4 common GNN inference acceleration

Table 4: Link prediction performance measured by Hits@20 on **cold start nodes**.

	GNN	MLP	Ours	Δ_{MLP}	Δ_{GNN}
Cora	6.39	17.92	**22.01**	4.09	15.62
Citeseer	11.04	29.33	**32.09**	2.76	21.05
Pubmed	4.63	22.74	**37.68**	14.94	33.05
CS	9.46	29.09	**46.83**	17.74	37.37
Physics	5.46	20.22	**39.37**	19.15	33.91
Computers	1.53	10.72	**14.64**	3.92	13.11
Photos	0.87	20.44	**23.79**	3.35	22.92

Table 5: The statistics of large-scale datasets and the link prediction performances on them. "Citation2-s" indicates the down-sampled Citation2 graph.

	IGB-100K	IGB-1M	Citation2	Citation2-s
# Nodes	100K	1M	2,9M	122K
# Edges	547K	12M	30.6M	1.4M
GNN	79.25	40.12	82.56	39.99
MLP	68.83	18.84	40.56	24.34
LLP	79.47	39.78	53.20	29.23

methods: (i) SAGE (Hamilton et al., 2017), (ii) Quantized SAGE (QSAGE) (Zhao et al., 2020; Tailor et al., 2020) from float32 to int8, (iii) SAGE with 50% weights pruned (PSAGE) (Zhou et al., 2021; Chen et al., 2021c), and (iv) SAGE with Neighbor Sampling with fan-out 15. Figure 2 shows the results on the large-scale OGB dataset, Collab. We can observe that LLP can obtain **70.68**× speedup comparing with on SAGE on Collab. Compared with the best acceleration method Neighbor Sampling (which reduces graph dependency, but does not eliminate it like LLP), LLP still achieves **15.05**× speedup. This is because all these inference acceleration methods still rely on the graph structure. From the bottom figure in Figure 2, we can further observe that LLP can outperform both GNN and other inference acceleration methods.

4.4. Link Prediction Results on Cold Start Nodes

Dealing with cold start nodes (newly appeared nodes without edges) is a common challenge in recommendation and information retrieval applications (Li et al., 2019; Zheng et al., 2021; Ding et al., 2021). Without these edges, GNNs cannot perform well as they rely heavily on neighbor information. On the other hand, MLPs, which do not make use of any graph topology information, are arguably more suitable. Here, we simulate the cold-start setting by removing all the new edges during testing stage of the production setting, i.e. all the new appeared nodes are isolated (more details are shown in Appendix C). Table 4 shows the performances of LLP, the stand-alone MLP, and the teacher GNN on the cold-start nodes. We observe that LLP consistently outperforms GNN and MLP by average of **25.29** and **9.42** on Hits@20, respectively. We further compare LLP with another related work on cold-start nodes in Appendix D.8

4.5. Link Prediction Results on Large Scale Datasets

Besides Collab, we also conduct experiments on three recently proposed large-scale graph benchmarks, IGB-100K, IGB-1M (Khatua et al., 2023), and Citation2 (Wang et al., 2020a; Mikolov et al., 2013). The dataset stats and link prediction results (Hits@200) are shown in Table 5. On the IGB datasets, we observe that LLP can produce competitive results on these two datasets, which further demonstrates LLP has the ability to acquire complex link prediction-related knowledge from large-scale graphs. On the other hand, the results on Citation2 show different patterns. Although LLP is able to significantly outperform MLP, its performances still show big gaps when compared with GNNs.

We hypothesize that the different performances pattern on Citation2 is due to the dataset's unique distribution. To validate our hypothesis, we further conduct experiments on a sampled version of Citation2 (the "Citation2-s" column in Table 5). Specifically, we down-sample Citation2 to produce a smaller version of it (with size similar to Collab). We use random walk-based sampling, which has proved ability of property preserving on the original graph (Leskovec & Faloutsos, 2006). From Table 5 we observe very similar patterns on the performances on both Citation2 and Citation2-s, validation our hypothesize that the performance gap between LLP and GNNs are due to the dataset's own distribution rather than other factors such as the size of the graph. Similar observation can also be found when comparing the results of Photos and Collab in Table 2, but in a reversed way: LLP performs better than GNNs on the larger dataset (Collab) but slightly worse on the smaller one (Photos). In summary, the effectiveness of LLP may vary on different datasets, but is not sensitive to the size of the graphs.

4.6. Ablation Study

Effectiveness of $\mathcal{L}_{\text{LLP}_R}$ and $\mathcal{L}_{\text{LLP}_D}$. As our proposed LLP contains two matching strategies, rank-based and distribution-based matching, we evaluate their effectiveness by removing them from LLP. Moreover, we further evaluate by also removing $\mathcal{L}_{sup}$, i.e., using only one of the matching losses as the overall loss for LLP. Table 6 shows the results of these settings compared with the performances of full LLP, stand-alone MLP, and the teacher GNN on Pubmed and CS datasets under both settings. We observe that both rank-based and distribution-based matching contribute significantly for the overall performance. In the transductive setting, both loss terms by themselves (the bottom two rows) can already outperform the teacher GNN.

Table 6: Link prediction performances measured by Hits@20 on different components of LLP.

Setting	Transductive		Production	
Dataset	Pubmed	CS	Pubmed	CS
GNN	51.98	59.51	52.71	60.69
MLP	42.89	40.69	38.01	38.15
LLP	**57.33**	**68.62**	**53.48**	**60.74**
w/o $\mathcal{L}_{\text{LLP}_R}$	55.35	66.61	53.40	60.53
w/o $\mathcal{L}_{\text{LLP}_D}$	54.97	65.17	48.58	60.13
w/o $\mathcal{L}_{\text{LLP}_R}$, $\mathcal{L}_{sup}$	54.86	68.39	39.35	57.35
w/o $\mathcal{L}_{\text{LLP}_D}$, $\mathcal{L}_{sup}$	53.30	68.30	41.43	55.63

In the production setting, the matching losses alone outperform MLP and can achieve comparable performances with GNN after $\mathcal{L}_{sup}$ is added. In conclusion, both rank-based and distribution-based matching can effectively distill the relational knowledge, and they achieve the best performance by complementing each other.

Context Sampling Sensitivity to p and q. Figure 3 shows the link prediction performance of LLP on Pubmed under the transductive setting with different numbers of context node samples (p local samples, and q random samples). For the ease of hyper-parameter tuning, we make q a multiple of p, as shown in the x-axis of Figure 3. We observe that LLP with low number of random samples shows poor link prediction performances, suggesting that preserving global relations are necessary for the proposed relation KD. Generally, the heatmap shows a clear trend, making the optimal values easy to locate.

5. Conclusion

Our work tackled problems related to applying GNNs for link prediction at scale. We note these models have high latency at inference time owing to non-trivial data dependency. In response, we explored applying cross-model distillation methods from teacher GNN to student MLP models, which are advantaged in inference time. We first adopt two direct logit matching and representation matching KD methods to the link prediction context and observe their unsuitability. In response, we introduced a relational KD framework, LLP, which proposed *rank-based matching* and *distribution-based matching* objectives which complement each other to force the student to preserve key information about contextual relationships across anchor nodes. Our experiments demonstrated that LLP achieved MLP-level speedups (up to **70.68**× over GNNs), while also improving link prediction performance over MLPs by **18.18** and **12.01** points in transductive and production settings, matching or outperforming the teacher GNN in **7/8** datasets in transductive setting and 3/6 datasets in production setting, and with notable **25.29** on Hits@20 improvements on cold-start nodes.

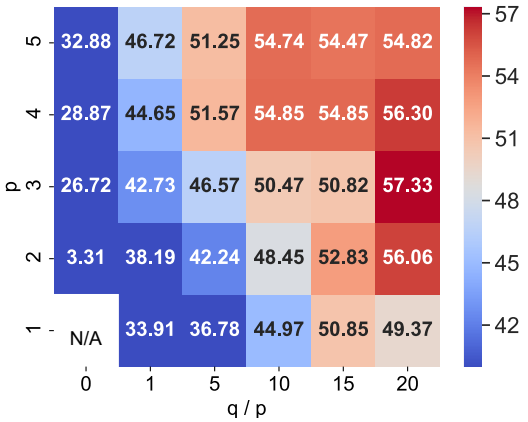

Figure 3: Link prediction performance measured by Hits@20 of LLP on Pubmed with different number of samples for the context nodes.

Limitations

Similar to other MLP-based graph learning methods (Zhang et al., 2021b; Zheng et al., 2021), the main limitation of our proposed LLP is that it relies heavily on feature quality. It can hardly perform well when the node features are unreliable or missing.

Ethical Impact

We do not foresee any negative societal impact or ethical concerns posed by our method. Nonetheless, we note that both positive and negative societal impacts can be made by applications of graph machine learning techniques, which may benefit from the improvements induced by our work. Care must be taken, in general, to ensure positive societal and ethical consequences of machine learning.

Acknowledgement

We appreciate Xiaotian Han from Texas A&M University, Wei Jin from Michigan State University, and Yiwei Wang from National University of Singapore for valuable discussions and suggestions.

References

Adamic, L. A. and Adar, E. Friends and neighbors on the web. *Social networks*, 2003.

Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep

learning. *arXiv preprint arXiv:2012.09816*, 2020.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *science*, 1999.

Berg, R. v. d., Kipf, T. N., and Welling, M. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

Bevilacqua, B., Frasca, F., Lim, D., Srinivasan, B., Cai, C., Balamurugan, G., Bronstein, M. M., and Maron, H. Equivariant subgraph aggregation networks. *arXiv preprint arXiv:2110.02910*, 2021.

Bojchevski, A. and Günnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

Brin, S. and Page, L. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 2012.

Cai, L. and Ji, S. A multi-scale approach for graph link prediction. In *Proceedings of the AAAI conference on artificial intelligence*, 2020.

Cai, L., Li, J., Wang, J., and Ji, S. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136, 2007.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 2019.

Chen, D., Mei, J.-P., Zhang, Y., Wang, C., Wang, Z., Feng, Y., and Chen, C. Cross-layer distillation with semantic calibration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021a.

Chen, J., He, H., Wu, F., and Wang, J. Topology-aware correlations between relations for inductive link prediction in knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021b.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, 2020.

Chen, T., Sui, Y., Chen, X., Zhang, A., and Wang, Z. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning*, pp. 1695–1706. PMLR, 2021c.

Covington, P., Adams, J., and Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pp. 191–198, 2016.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018.

Deng, X. and Zhang, Z. Graph-free knowledge distillation for graph neural networks. *arXiv preprint arXiv:2105.07519*, 2021.

Ding, H., Ma, Y., Deoras, A., Wang, Y., and Wang, H. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318*, 2021.

Fan, W., Liu, X., Jin, W., Zhao, X., Tang, J., and Li, Q. Graph trend filtering networks for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 112–121, 2022.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Fey, M., Lenssen, J. E., Weichert, F., and Leskovec, J. Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings. In *International Conference on Machine Learning*, pp. 3294–3304. PMLR, 2021.

Furlanello, T., Lipton, Z., Tschannen, M., Itti, L., and Anandkumar, A. Born again neural networks. In *International Conference on Machine Learning*, pp. 1607–1616. PMLR, 2018.

Geerts, F. and Reutter, J. L. Expressiveness and approximation properties of graph neural networks. *arXiv preprint arXiv:2204.04661*, 2022.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. Knowledge distillation: A survey. *International Journal of Computer Vision*, 2021.

Guo, Z., Zhang, C., Yu, W., Herr, J., Wiest, O., Jiang, M., and Chawla, N. V. Few-shot graph learning for molecular property prediction. In *WWW*, 2021.

Guo, Z., Zhang, C., Fan, Y., Tian, Y., Zhang, C., and Chawla, N. Boosting graph neural networks via adaptive knowledge distillation. *arXiv preprint arXiv:2210.05920*, 2022.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 2017.

Hao, Y., Cao, X., Fang, Y., Xie, X., and Wang, S. Inductive link prediction for nodes having only attribute information. *arXiv preprint arXiv:2007.08053*, 2020.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. 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*, pp. 639–648, 2020.

Hinton, G., Vinyals, O., Dean, J., et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hu, Y., You, H., Wang, Z., Wang, Z., Zhou, E., and Gao, Y. Graph-mlp: node classification without message passing in graph. *arXiv preprint arXiv:2106.04051*, 2021.

Huang, Z., Li, X., and Chen, H. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pp. 141–142, 2005.

Jeh, G. and Widom, J. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

Ji, G. and Zhu, Z. Knowledge distillation in wide neural networks: Risk bound, data efficiency and imperfect teacher. *Advances in Neural Information Processing Systems*, 2020.

Jia, Z., Lin, S., Ying, R., You, J., Leskovec, J., and Aiken, A. Redundancy-free computation for graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Joshi, C. K., Liu, F., Xun, X., Lin, J., and Foo, C.-S. On representation knowledge distillation for graph neural networks. *arXiv preprint arXiv:2111.04964*, 2021.

Ju, M., Zhao, T., Wen, Q., Yu, W., Shah, N., Ye, Y., and Zhang, C. Multi-task self-supervised graph neural networks enable stronger task generalization. *International Conference on Learning Representations*, 2023.

Kang, S., Hwang, J., Kweon, W., and Yu, H. Topology distillation for recommender system. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 829–839, 2021.

Khatua, A., Mailthody, V. S., Taleka, B., Ma, T., Song, X., and Hwu, W.-m. Igb: Addressing the gaps in labeling, features, heterogeneity, and size of public graph datasets for deep learning research. In *In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

Kim, J., Park, S., and Kwak, N. Paraphrasing complex network: Network compression via factor transfer. *Advances in neural information processing systems*, 2018.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 2009.

Leskovec, J. and Faloutsos, C. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

Li, J., Jing, M., Lu, K., Zhu, L., Yang, Y., and Huang, Z. From zero-shot learning to cold-start recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, 2019.

Li, P., Wang, Y., Wang, H., and Leskovec, J. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

Liu, G., Zhao, T., Xu, J., Luo, T., and Jiang, M. Graph rationalization with environment-based augmentations. In *Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2022.

Liu, X., Jin, W., Ma, Y., Li, Y., Liu, H., Wang, Y., Yan, M., and Tang, J. Elastic graph neural networks. In *International Conference on Machine Learning*, pp. 6837–6849. PMLR, 2021.

Ma, J. and Mei, Q. Graph representation learning via multi-task knowledge distillation. *arXiv preprint arXiv:1911.05700*, 2019.

Ma, Y., Liu, X., Zhao, T., Liu, Y., Tang, J., and Shah, N. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Martínez, V., Berzal, F., and Cubero, J.-C. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)*, 49(4):1–33, 2016.

McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

Nathani, D., Chauhan, J., Sharma, C., and Kaul, M. Learning attention-based embeddings for relation prediction in knowledge graphs. *arXiv preprint arXiv:1906.01195*, 2019.

Park, W., Kim, D., Lu, Y., and Cho, M. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

Philip, S. Y., Han, J., and Faloutsos, C. *Link mining: Models, algorithms, and applications*. Springer, 2010.

Phuong, M. and Lampert, C. Towards understanding knowledge distillation. In *International Conference on Machine Learning*. PMLR, 2019.

Reddi, S., Pasumarthi, R. K., Menon, A., Rawat, A. S., Yu, F., Kim, S., Veit, A., and Kumar, S. Rankdistil: Knowledge distillation for ranking. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Sankar, A., Liu, Y., Yu, J., and Shah, N. Graph neural networks for friend ranking in large-scale social platforms. In *Proceedings of the Web Conference*, pp. 2535–2546, 2021.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Shiao, W. and Papalexakis, E. E. Adversarially generating rank-constrained graphs. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2021.

Shiao, W., Guo, Z., Zhao, T., Papalexakis, E. E., Liu, Y., and Shah, N. Link prediction with non-contrastive learning. In *International Conference on Learning Representations*, 2023.

Tailor, S. A., Fernandez-Marques, J., and Lane, N. D. Degree-quant: Quantization-aware training for graph neural networks. *arXiv preprint arXiv:2008.05000*, 2020.

Tang, X., Liu, Y., He, X., Wang, S., and Shah, N. Friend story ranking with edge-contextual local graph convolutions. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 1007–1015, 2022.

Tian, Y., Zhang, C., Guo, Z., Zhang, X., and Chawla, N. V. Nosmog: Learning noise-robust and structure-aware mlps on graphs. *arXiv preprint arXiv:2208.10010*, 2022.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. Complex embeddings for simple link prediction. In *International conference on machine learning*, pp. 2071–2080. PMLR, 2016.

Tsitsulin, A., Mottin, D., Karras, P., and Müller, E. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 world wide web conference*, pp. 539–548, 2018.

Tung, F. and Mori, G. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

Vashishth, S., Sanyal, S., Nitin, V., and Talukdar, P. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*, 2020.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wang, K., Shen, Z., Huang, C., Wu, C.-H., Dong, Y., and Kanakia, A. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 2020a.

Wang, X., Fu, T., Liao, S., Wang, S., Lei, Z., and Mei, T. Exclusivity-consistency regularized knowledge distillation for face recognition. In *European Conference on Computer Vision*, 2020b.

Wang, Z., Zhou, Y., Hong, L., Zou, Y., and Su, H. Pairwise learning for neural link prediction. *arXiv preprint arXiv:2112.02936*, 2021.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pp. 5453–5462. PMLR, 2018.

Yan, B., Wang, C., Guo, G., and Lou, Y. Tinygnn: Learning efficient graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Yang, C., Liu, J., and Shi, C. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *Proceedings of the Web Conference 2021*, 2021.

Yang, Y., Qiu, J., Song, M., Tao, D., and Wang, X. Distilling knowledge from graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020a.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 2016.

Yang, Z., Shou, L., Gong, M., Lin, W., and Jiang, D. Model compression with two-stage multi-teacher knowledge distillation for web question answering system. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020b.

Yin, H., Zhang, M., Wang, Y., Wang, J., and Li, P. Algorithm and system co-design for efficient subgraph-based graph representation learning. *arXiv preprint arXiv:2202.13538*, 2022.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983, 2018a.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 2018b.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*. PMLR, 2018.

Yun, S., Kim, S., Lee, J., Kang, J., and Kim, H. J. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 2021.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Zhang, C., Yao, H., Huang, C., Jiang, M., Li, Z., and Chawla, N. V. Few-shot knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020a.

Zhang, D., Huang, X., Liu, Z., Hu, Z., Song, X., Ge, Z., Zhang, Z., Wang, L., Zhou, J., Shuang, Y., et al. Agl: a scalable system for industrial-purpose graph machine learning. *arXiv preprint arXiv:2003.02454*, 2020b.

Zhang, H., Lin, S., Liu, W., Zhou, P., Tang, J., Liang, X., and Xing, E. P. Iterative graph self-distillation. *arXiv preprint arXiv:2010.12609*, 2020c.

Zhang, M. and Chen, Y. Weisfeiler-lehman neural machine for link prediction. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 575–583, 2017.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 2018.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

Zhang, M., Li, P., Xia, Y., Wang, K., and Jin, L. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems*, 34:9061–9073, 2021a.

Zhang, S., Liu, Y., Sun, Y., and Shah, N. Graph-less neural networks: Teaching old mlps new tricks via distillation. *arXiv preprint arXiv:2110.08727*, 2021b.

Zhang, W., Miao, X., Shao, Y., Jiang, J., Chen, L., Ruas, O., and Cui, B. Reliable data distillation on graph convolutional network. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020d.

Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020e.

Zhao, L., Jin, W., Akoglu, L., and Shah, N. From stars to subgraphs: Uplifting any gnn with local structure awareness. *arXiv preprint arXiv:2110.03753*, 2021a.

Zhao, T., Liu, Y., Neves, L., Woodford, O., Jiang, M., and Shah, N. Data augmentation for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021b.

Zhao, T., Jin, W., Liu, Y., Wang, Y., Liu, G., Günneman, S., Shah, N., and Jiang, M. Graph data augmentation for graph machine learning: A survey. *arXiv preprint arXiv:2202.08871*, 2022a.

Zhao, T., Liu, G., Wang, D., Yu, W., and Jiang, M. Learning from counterfactual links for link prediction. In *International Conference on Machine Learning*, pp. 26911–26926. PMLR, 2022b.

Zhao, T., Tang, X., Zhang, D., Jiang, H., Rao, N., Song, Y., Agrawal, P., Subbian, K., Yin, B., and Jiang, M. Autogda: Automated graph data augmentation for node classification. In *The First Learning on Graphs Conference*, 2022c.

Zhao, Y., Wang, D., Bates, D., Mullins, R., Jamnik, M., and Lio, P. Learned low precision graph neural networks. *arXiv preprint arXiv:2009.09232*, 2020.

Zheng, W., Huang, E. W., Rao, N., Katariya, S., Wang, Z., and Subbian, K. Cold brew: Distilling graph node representations with incomplete or missing neighborhoods. *arXiv preprint arXiv:2111.04840*, 2021.

Zhou, H., Srivastava, A., Zeng, H., Kannan, R., and Prasanna, V. Accelerating large scale real-time gnn inference using channel pruning. *arXiv preprint arXiv:2105.04528*, 2021.

Zhu, Z., Zhang, Z., Xhonneux, L.-P., and Tang, J. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 2021.

A. Further Related Work

Graph Neural Networks (GNNs). Many GNN architectures have been proposed in recent years to model attributed graph data; most architectures follow the message passing (Gilmer et al., 2017; Ying et al., 2018b; Guo et al., 2021; Ma et al., 2021; Liu et al., 2021; 2022; Ju et al., 2023) paradigm. Different GNN customizations include degree normalization (Kipf & Welling, 2016a), neighbor sampling and neighbor separation (Hamilton et al., 2017; Zhao et al., 2021b), self-attention (Veličković et al., 2017), residual connections (Xu et al., 2018), and more. Alon & Yahav (2020) proposed to use a fully-adjacent layer at the end of GNN to deal with the bottleneck problem of GNNs. Moreover, researchers also proposed subgraph-based methods (Bevilacqua et al., 2021; Zhao et al., 2021a), tensor-based methods (Maron et al., 2019; Geerts & Reutter, 2022), and augmentation methods (Zhao et al., 2022a; Liu et al., 2022; Zhao et al., 2022c) for improving GNNs.

Link Prediction. Link prediction has achieved great attention from the research community, considering its wide applications. Heuristic methods (Philip et al., 2010) were proposed to make the link prediction by measuring the link scores based on the structure information, such as the common neighbors and the shortest path. 2-order (Adamic & Adar, 2003) and high-order (Brin & Page, 2012; Jeh & Widom, 2002) heuristic methods were proposed to further improve the effectiveness. In recent years, GNN-based methods (Zhang & Chen, 2018; Yun et al., 2021; Zhao et al., 2022b) showed their promising performances for link prediction. One line of work follows the node embedding-based strategy, as previously discussed in Section 2, where the GNN-based encoder learns node representations and the decoder predicts whether the link exists. It is worth mentioning that knowledge graph completion follows this strategy to predict not only the link existence but also the type of the link (Schlichtkrull et al., 2018; Nathani et al., 2019; Vashishth et al., 2020; Zhang et al., 2020a). The knowledge graph completion methods mainly use heterogeneous graph neural networks, which are sensitive to different edge types.

Another line of work casts link prediction tasks to binary classification problems on the enclosing subgraphs around each node pair (Zhang & Chen, 2018; Cai & Ji, 2020; Cai et al., 2021). Although these methods can improve task performance, they are usually computationally expensive and cannot scale well in practical use-cases (Yin et al., 2022). Similarly, Zhu et al. (2021) proposed a GNN link prediction paradigm by encoding information of all paths between two nodes, which is also very expensive.

GNN Inference Acceleration. Pruning (Zhou et al., 2021; Chen et al., 2021c) and quantization (Zhao et al., 2020; Tailor et al., 2020) strategies were proposed for accelerating GNN inference. These methods do accelerate GNNs, but they rely on graph data for message passing and thus leave much room for speed improvement. We note that these approaches are complementary to cross-model distillation, and can be employed together with KD for additional inference time improvements. Other than the above acceleration methods, Hu et al. (2021) and Zhang et al. (2021b) accelerated GNNs by distilling them to MLP. These works focus on KD for node classification tasks, whereas we focus on link prediction tasks. GNNAutoScale (Fey et al., 2021) proposed an effective method to accelerate the training process of GNNs. It also reduces the inference to a constant factor by directly using historical embeddings stored offline. However, in this case, all the methods in Figure 2 can share the same inference time benefits. Moreover, GNNAutoScale is not suitable for the production setting, where new nodes (without historical embeddings) appear frequently after the training process. So we did not include it as a baseline in this work.

Knowledge Distillation (KD). Logit-based (Hinton et al., 2015; Furlanello et al., 2018; Zhang et al., 2021b) and representation-based (Romero et al., 2014; Gou et al., 2021) matching are two common KD methods, which match final-layer and intermediate-layer predicted logits between the teacher and the student, respectively. Our work is the first to adapt and evaluate these approaches in the link prediction setting, to the best of our knowledge.

For representation-based KD, several work (Park et al., 2019; Tung & Mori, 2019; Joshi et al., 2021) proposed relational KD, which corresponds to instance-to-instance KD while preserving metrics among representations of similar instances. For GNNs, (Yang et al., 2020a) used knowledge of the neighboring nodes to teach the student to better classify the center node. In contrast, our KD strategies focus on transferring relational knowledge between each pair of nodes from teacher to student. Both the rank-matching and distribution-matching strategies help the student to better capture the relational graph topology information and make better link predictions.

RankDistill (Reddi et al., 2021) and Topology Distillation (Kang et al., 2021) are designed to transfer ranking knowledge from the teacher to the student. Different from our work which distill the relational information in a graph context, they distill ranking in a non-graph context between teacher and student. We adopt different sampling and matching methods based on our different motivations. Further analysis is shown in Appendix D.9.

Knowledge Distillation on GNNs. Existing GNN-based KD work are mostly based on the logit-based KD (Hinton et al., 2015) to obtain light-weight models (Zhang et al., 2020d; Zheng et al., 2021; Yang et al., 2021). Yan et al. (2020) proposed to train a student GNN with fewer parameters using KD. Yang et al. (2021) improved the designed stu-

Table 7: Detailed statistics of datasets splits under production setting.

	Nodes		Testing Edges		
	# Existing	# New	# Existing – Existing	# Existing – New	# New – New
Cora	1,896	812	765	675	142
Citeseer	2,329	998	673	568	124
Pubmed	15,774	3,943	5,648	2,858	358
CS	14,666	3,667	10,482	5,221	675
Physics	27,594	6,899	31,399	16,126	2,067
Computers	11,002	2,750	31,095	16,033	2,043
Photos	6,120	1,530	15,248	7,618	950

dent model, which consists of label propagation and feature-based prior knowledge, using the pre-trained teacher GNN. Different from the above work, LSP (Yang et al., 2020a) and G-CRD (Joshi et al., 2021) proposed structure-preserving KD methods, which are specifically designed for GNN. Both of these work follow the original relational KD to preserve the metrics among node representations and are applied on node classification tasks.

B. Additional Datasets Details

Here we present the details of the datasets used in the experiments. Cora, Citeseer, and Pubmed (Yang et al., 2016) are all representative citation network datasets, where the nodes and edges represent papers and citations, respectively. CS, Physics (Shchur et al., 2018) and Collab (Wang et al., 2020a) are all collaboration networks based on MAG, where the nodes represent authors and the edges indicate the collaboration for the paper. Computers and Photos (Shchur et al., 2018) are two well-known co-purchased graphs (McAuley et al., 2015), where the nodes represent goods and the edges indicate two items were bought together.

C. Additional Evaluation Setting Details

C.1. Transductive Setting

The transductive setting is a standard setting for link prediction (Kipf & Welling, 2016b; Zhang & Chen, 2017; 2018; Yun et al., 2021; Zhao et al., 2022b), where the nodes in training/validation/testing are all visible in the training graph, but subsets of positive links are masked out for validation and test sets.

C.2. Production Setting

In this work, we design a new production setting to resemble the real-world link prediction scenario. This setting mimics practical link prediction use cases. For example, user friend recommendations on social platforms where new users (nodes) and friendships (links) appear frequently. Under the production setting, the newly occurred nodes and edges that can not be seen during the training stage would appear in the graph at inference time.

Specifically, the following are the detailed procedures of splitting the datasets into the production setting:

- **Split all nodes:** Given the graph $G = (\mathcal{V}, \mathcal{E})$, we randomly sample 10% of nodes from $\mathcal{V}$ as the new nodes $\mathcal{V}^N$ and remove them from the training graph. We denote the remaining nodes by $\mathcal{V}^E$, where superscripts E stands for *Existing* and N stands for *New*. Note that for Cora and Citeseer, we sample 30% nodes as new nodes because these two datasets are too small.
- **Split all edges:** We then split the edges $\mathcal{E}$ according to the node splits into three sets: $\mathcal{E}^{E-E}$, $\mathcal{E}^{E-N}$, and $\mathcal{E}^{N-N}$, denoting the links between existing–existing, existing–new, and new–new node pairs, respectively.
- **Split edges in $\mathcal{E}^{E-E}$:** For the existing–existing node pairs, we split it into three sets following an 80/10/10 splitting ratio: 80% as training edges, 10% as new visible edges for message passing, and 10% as testing edges. Note that validation set contains only existing nodes $\mathcal{V}^E$ as the new nodes are not visible during training.
- **Split edges in $\mathcal{E}^{E-N}$ and $\mathcal{E}^{N-N}$:** We follow the same ratio and split these two sets following with 90/10 splitting ratio: 90% as newly visible edges (used only for message passing during testing inference), and 10% as testing edges.
- **Message passing edges during training:** During training, the GNN model can only utilize the 80% existing-existing training edges for message passing.
- **Message passing edges for inferencing:** During inference, the GNN model can conduct message passing on all edges except the testing ones. Specifically, the training and testing (total of 90%) sets of $\mathcal{E}^{E-E}$, and the 90% of newly visible message passing edges in $\mathcal{E}^{E-N}$ and $\mathcal{E}^{N-N}$.

Table 8: Detailed link prediction performance measured by Hits@20 under **production** setting. Best and second best performances are marked with bold and underline, respectively.

	GNN	MLP	$\mathcal{L}_{LM}$	$\mathcal{L}_{RM}$	LLP	$\Delta_{DirectKD}$	Δ_{MLP}	Δ_{GNN}
				Overall				
Cora	27.80±2.11	22.90±2.22	22.65±2.51	22.24±0.55	**27.87**±1.24	5.22	4.97	0.07
Citeseer	**38.78**±2.59	31.21±3.75	29.35±2.55	26.23±1.08	34.75±2.45	5.40	3.54	-4.03
Pubmed	52.71±1.81	38.01±1.67	39.03±4.21	43.27±3.12	**53.48**±1.52	10.21	15.47	0.77
CS	60.69±3.17	38.15±10.78	48.07±2.39	58.90±1.32	**60.74**±1.41	1.84	22.59	0.05
Physics	**55.82**±2.43	29.99±1.96	22.74±1.03	36.32±2.29	52.83±1.50	16.51	22.84	-2.99
Computers	**34.38**±1.41	19.43±0.82	12.79±1.43	20.28±1.01	24.58±3.33	4.30	5.15	-9.80
Photos	**51.03**±6.05	34.29±2.49	24.63±2.20	40.58±1.63	43.79±1.27	3.21	9.50	-7.24
				Existing – Existing				
Cora	28.81±2.01	28.00±2.70	27.66±3.01	27.03±0.65	**33.31**±1.29	5.65	5.31	4.5
Citeseer	**38.10**±2.70	33.88±3.50	32.24±2.89	27.52±0.94	37.50±2.43	5.26	3.62	-0.60
Pubmed	52.67±1.78	41.58±1.61	42.57±4.32	46.32±3.08	**57.16**±1.34	10.84	15.58	4.49
CS	61.52±3.10	40.27±11.69	50.78±2.50	62.17±1.45	**63.99**±1.36	1.82	23.72	2.47
Physics	**56.56**±2.42	32.32±2.32	23.88±1.14	38.74±2.50	56.04±1.47	17.30	23.72	-0.52
Computers	**35.13**±1.48	21.46±1.08	13.81±1.56	22.78±1.17	26.89±3.60	4.11	5.43	-8.24
Photos	**51.90**±6.24	37.47±2.73	26.54±2.55	44.51±2.10	48.38±1.30	3.87	10.91	-3.52
				Existing – New				
Cora	**25.78**±2.33	19.47±2.09	19.11±2.03	18.58±1.28	23.08±1.51	3.97	3.61	-2.7
Citeseer	**38.73**±2.37	30.77±4.07	28.77±2.70	26.65±1.52	34.30±2.40	5.53	3.53	-4.43
Pubmed	**53.98**±2.29	23.70±2.09	24.91±4.00	32.21±3.38	38.94±2.44	6.73	15.24	-15.04
CS	**56.78**±3.57	29.25±7.05	36.60±2.17	45.28±0.93	47.05±1.72	1.77	17.80	-9.73
Physics	**52.90**±2.44	20.61±1.01	18.23±0.74	26.57±1.81	39.73±1.75	13.16	19.12	-13.17
Computers	**31.07**±1.17	11.00±1.37	8.53±1.16	9.85±0.54	14.88±2.58	5.03	3.88	-16.19
Photos	**47.42**±5.18	21.00±1.65	16.75±0.92	24.10±1.38	24.27±2.07	0.17	3.27	-23.15
				New – New				
Cora	**31.97**±6.65	11.69±2.19	12.54±2.83	13.80±1.37	16.90±5.50	3.10	5.21	-15.07
Citeseer	**42.74**±4.49	18.71±4.54	16.29±3.80	17.26±3.54	21.94±4.39	4.68	3.23	-20.8
Pubmed	**33.18**±1.24	5.45±1.24	4.55±4.55	11.36±4.82	15.00±6.35	3.64	9.55	-18.18
CS	**64.10**±3.55	26.27±8.79	33.73±3.81	38.07±2.90	42.89±1.83	4.82	16.62	-21.21
Physics	**48.96**±3.53	13.20±1.62	12.56±1.85	18.88±2.22	32.80±1.55	13.92	19.6	-16.16
Computers	**32.61**±1.89	5.55±1.56	6.72±0.66	3.87±1.24	10.25±1.41	3.53	4.7	-22.36
Photos	**43.54**±6.6	10.09±3.99	8.14±1.15	12.21±1.31	14.87±3.09	2.66	4.78	-28.67

- **Testing edges:** We test all methods on the above-mentioned three separate testing edge sets (10% of each) sampled from $\mathcal{E}^{E-E}$, $\mathcal{E}^{E-N}$, and $\mathcal{E}^{N-N}$, respectively.

Table 7 shows the detailed statistics of different datasets under this setting.

C.3. Cold Start Setting

Followed by the production setting, we remove all the new edges appearing newly in the inference time. Then the new nodes will be the strict cold start nodes with no neighbor information for the model to predict. The experimental results shown in Section 4.4 are conducted with this setting.

D. Additional Experimental Results

D.1. Detailed Hits@20 Results under Production Setting

Here, we stratify each method's performance on the three different categories of the test edges in Table 8. From these more stratified test performances, we observe that LLP can generally achieve similar performance with GNN on the existing–existing category, but much worse on the other two categories that involve newly appeared nodes. We hypothesize that this is because GNN neighbor aggregation improves generalization for low-degree nodes. We also observe that the performance gaps between the teacher GNN and stand-alone MLP on new–new and existing–new are much larger than that of the existing–existing category, which also evidences that GNNs have better inductive bias than MLPs on graph data. Nonetheless, we note that such a significant and consistent performance improvement of LLP over MLP is valuable for large-scale industrial applications, given their popularity in practice.

Table 9: Link prediction performance measured by AUC under **transductive** setting.

	GNN	MLP	$\mathcal{L}_{LM}$	$\mathcal{L}_{RM}$	LLP	$\Delta_{DirectKD}$	Δ_{MLP}	Δ_{GNN}
Cora	95.03±0.37	94.80±0.44	94.67±0.58	94.05±0.17	**95.23**±0.49	0.56	0.43	0.20
Citeseer	95.15±0.58	93.11±1.21	94.11±0.21	92.88±0.37	**95.32**±0.21	1.21	2.21	0.17
Pubmed	93.84±0.31	97.89±0.07	97.82±0.06	**97.96**±0.02	97.90±0.09	-0.06	0.01	4.06
CS	97.43±0.23	97.61±0.52	98.05±0.14	**98.33**±0.05	98.06±0.04	-0.27	0.45	0.63
Physics	98.80±0.02	98.71±0.05	98.36±0.07	98.96±0.02	**99.10**±0.02	0.14	0.39	0.30
Computers	98.76±0.03	98.46±0.08	98.11±0.14	98.66±0.06	**98.84**±0.09	0.18	0.38	0.08
Photos	98.98±0.02	98.71±0.08	98.51±0.06	98.95±0.04	**99.03**±0.06	0.08	0.32	0.05

Table 10: Link prediction performance measured by AUC under **production** setting.

	GNN	MLP	$\mathcal{L}_{LM}$	$\mathcal{L}_{RM}$	LLP	$\Delta_{DirectKD}$	Δ_{MLP}	Δ_{GNN}
Overall								
Cora	72.59±1.63	73.41±2.04	70.67±1.62	64.62±0.51	**78.22**±1.14	7.55	4.81	5.63
Citeseer	69.15±1.82	77.36±3.38	75.04±3.20	67.67±0.59	**80.13**±0.98	5.09	2.77	10.98
Pubmed	90.45±0.45	96.07±0.13	96.13±0.26	**96.74**±0.05	94.30±0.34	-2.44	-1.77	3.85
CS	**97.08**±0.16	95.96±1.19	96.59±0.08	96.76±0.03	96.87±0.03	0.11	0.91	-0.21
Physics	98.60±0.02	97.70±0.04	97.46±0.08	98.00±0.01	**98.75**±0.11	0.75	1.05	0.15
Computers	**98.67**±0.05	97.85±0.04	97.59±0.07	97.95±0.03	97.89±0.04	-0.06	0.04	-0.78
Photos	**98.78**±0.14	97.97±0.08	97.85±0.06	98.18±0.04	98.05±0.03	-0.13	0.08	-0.73
Existing – Existing								
Cora	70.80±2.14	74.42±2.70	70.69±2.00	64.82±0.75	**78.43**±1.44	7.74	4.01	7.63
Citeseer	67.34±1.81	76.83±3.41	73.79±3.12	68.00±2.03	**78.36**±1.41	4.57	1.53	11.02
Pubmed	90.44±0.46	96.69±0.13	96.72±0.21	**97.24**±0.05	95.17±0.33	-2.07	-1.52	4.73
CS	**97.01**±0.16	96.08±1.11	96.70±0.08	96.91±0.03	97.00±0.03	0.09	0.92	-0.01
Physics	98.60±0.02	97.96±0.05	97.65±0.09	98.20±0.02	**98.76**±0.16	0.56	0.80	0.16
Computers	**98.70**±0.05	98.27±0.05	97.95±0.09	98.41±0.03	98.51±0.04	0.10	0.24	-0.19
Photos	**98.80**±0.14	98.33±0.09	98.20±0.07	98.57±0.07	98.61±0.04	0.04	0.28	-0.19
Existing – New								
Cora	72.61±1.50	72.06±1.55	70.18±1.41	64.07±0.58	**77.65**±1.12	7.47	5.59	5.04
Citeseer	69.90±1.88	77.58±3.48	76.05±3.51	67.13±1.74	**81.23**±0.71	5.18	3.65	11.33
Pubmed	90.82±0.38	93.67±0.23	93.82±0.54	**94.83**±0.14	90.97±0.74	-3.86	-2.70	0.15
CS	**97.31**±0.20	95.46±1.53	96.18±0.12	96.18±0.08	96.31±0.10	0.13	0.85	-1.00
Physics	**98.57**±0.04	96.64±0.08	96.66±0.09	97.17±0.04	95.72±0.27	-1.45	-0.92	-2.85
Computers	**98.60**±0.05	96.23±0.07	96.22±0.07	96.19±0.08	95.42±0.08	-0.80	-0.81	-3.18
Photos	**98.69**±0.14	96.53±0.03	96.45±0.09	96.60±0.08	95.76±0.16	-0.84	-0.77	-2.93
New – New								
Cora	**82.10**±1.57	74.46±1.40	72.85±1.92	66.12±1.39	79.85±1.30	7.00	5.39	-2.25
Citeseer	75.48±1.67	79.23±3.08	77.13±3.24	68.36±2.67	**84.68**±0.89	7.55	5.45	9.20
Pubmed	84.30±0.90	87.97±1.02	88.72±1.19	**89.95**±0.50	83.54±2.57	-6.41	-4.43	-0.76
CS	**97.99**±0.23	95.03±1.34	95.39±0.44	95.22±0.22	95.97±0.34	0.58	0.94	-2.02
Physics	**98.84**±0.12	95.72±0.30	96.43±0.20	96.80±0.24	94.78±0.34	-2.02	-0.94	-4.06
Computers	**98.07**±0.10	93.22±0.16	93.30±0.32	92.96±0.39	92.09±0.73	-1.21	-1.13	-5.98
Photos	**98.35**±0.16	94.21±0.28	94.69±0.09	93.79±0.43	92.09±0.59	-2.60	-2.12	-6.26

Nonetheless, while obtaining accurate link prediction for new nodes (e.g. recommending friends to new users) is very important in the production scenario, it is equally and perhaps even more important to achieve stronger prediction performance on the existing nodes (e.g. recommending friends to existing users in the real-world link prediction scenario). This is because, in practice, we have many large graphs (e.g. social graphs) which grow slowly, but which require frequent predictions for existing nodes (e.g. still have a large and active user-base to serve) – for example, Facebook's user growth[1] reveals that the platform added 51 million new users in 2022, but still had a total of 2.96 billion users that year. Additionally, Twitter experienced a decline in its user base in 2022[2]. Given these circumstances, "Overall" and "Existing-Existing" still matter a lot in practical settings because they account for this large cohort.

[1] https://www.oberlo.com/statistics/how-many-users-does-facebook-have

[2] https://www.statista.com/statistics/238729/twitters-annual-growth-rate-in-the-us/

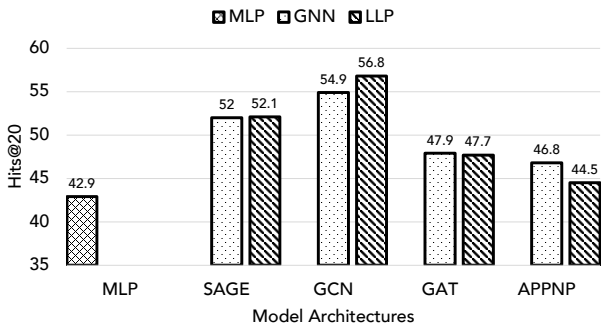

Figure 4: Link prediction performance measured by Hits@20 on Pubmed under transductive setting with different GNN teachers (SAGE, GAT, GCN, and APPNP).

Table 11: Link prediction performance measured by Hits@20 of SAGE, PLNLP (Wang et al., 2021) and LLP. LLP (SAGE) indicates LLP with SAGE as the teacher. LLP (PLNLP) indicates LLP with PLNLP as the teacher.

	SAGE	LLP (SAGE)	PLNLP	LLP (PLNLP)
CS	59.51	68.62	63.42	69.76
Physics	66.74	72.01	67.18	71.71
Computers	31.66	35.32	32.92	35.55
Photos	51.50	49.32	52.01	51.97
Collab	48.69	49.10	54.60	51.14

D.2. Link Prediction Results Measured by AUC under Transductive and Production Settings

We present AUC results on all the non-OGB datasets under the transductive setting in Table 9 and production setting in Table 10. In Table 9, we observe that our method outperforms both MLP and the teacher GNN on all the datasets under the transductive setting. For the production setting (Table 10), our method achieves better performances than the teacher GNN on 4/7 datasets.

D.3. Performance with Different Teachers

In our experiments, we use SAGE as the teacher GNN under both transductive and production settings. We further evaluate LLP's performance with other GNNs as teachers, such as GCN, GAT, and APPNP. In Figure 4, we observe that our method consistently outperforms MLP and the corresponding teacher GNNs. However, absolute performance is closely related to the teacher GNNs.

D.4. Comparison among SAGE, PLNLP, and LLP

Effectiveness Comparison. Here we extend the experiments in Figure 4 with more datasets and also a more advanced teacher model PLNLP (Wang et al., 2021), which is ranked #6 on the official Collab leaderboard. Table 11 shows the performance of different GNNs and LLP with different GNNs as teacher models. We can observe that LLP with PLNLP as teacher model achieves higher link prediction performances than the LLP with SAGE as teacher model, showing LLP is able to learn from more advanced teacher GNNs.

Table 12: Number of model parameters, inference time, and prediction performance comparison of SAGE, PLNLP (Wang et al., 2021) and LLP on Collab.

	SAGE	LLP (SAGE)	PLNLP	LLP (PLNLP)
# Params	263,169	2,232,321	60,644,864	2,232,321
Inference Time	156.3	8.1	143.2	8.1
Hits@50	48.69	49.1	54.6	51.14

Table 13: Link prediction performance measured by Hits@20 of LLP with different fixed-length and repeating times of random walks to sample the nearby nodes for the context nodes on CS. The default setting is 3 of 3-step random walks for the nearby nodes ($p = 9$) and 120 randomly sampled nodes ($q/p = 20$). RW indicates the random walk.

RW Length	1	3	5	10	20
Hits@20	64.7	67.82	66.18	66.67	67.1
Repeat RW Times	1	2	3	4	5
Hits@20	65.89	66.23	67.82	67.43	67.52

Efficiency Comparison. Due to the dependency on the graph structure, the inference time required for GNN methods can be very large when compared with that of MLPs. To illustrate, we show empirical results on Collab, in which we compare SAGE, PLNLP and LLP in terms of their number of model parameters, inference time, and prediction performance (Hits@50). In Table 12, we found that the two GNNs have high delay during inference (over 100-millisecond for a single node). In many real-time applications, such high latency would make the deployment of GNNs infeasible given business or engineering constraints, considering that many such queries may occur simultaneously in industrial settings. On the other hand, we observe that LLP can preserve most of the performance with significantly less time, indicating LLP is more suitable for certain contexts in which time constraints are important.

D.5. Analysis of Context Nodes Selection Strategy.

Sensitivity Analysis of p and q for $\mathcal{L}_{\text{LLP}_R}$ and $\mathcal{L}_{\text{LLP}_D}$. To analyze the influence of the context nodes on $\mathcal{L}_{\text{LLP}_R}$ and $\mathcal{L}_{\text{LLP}_D}$, we plot two heat maps to show their individual performance on Pubmed under the transductive setting, as shown in Figure 5. These two figures show different patterns with the context nodes. $\mathcal{L}_{\text{LLP}_D}$ (left figure) shows that the performance becomes better with more nearby nodes (p)

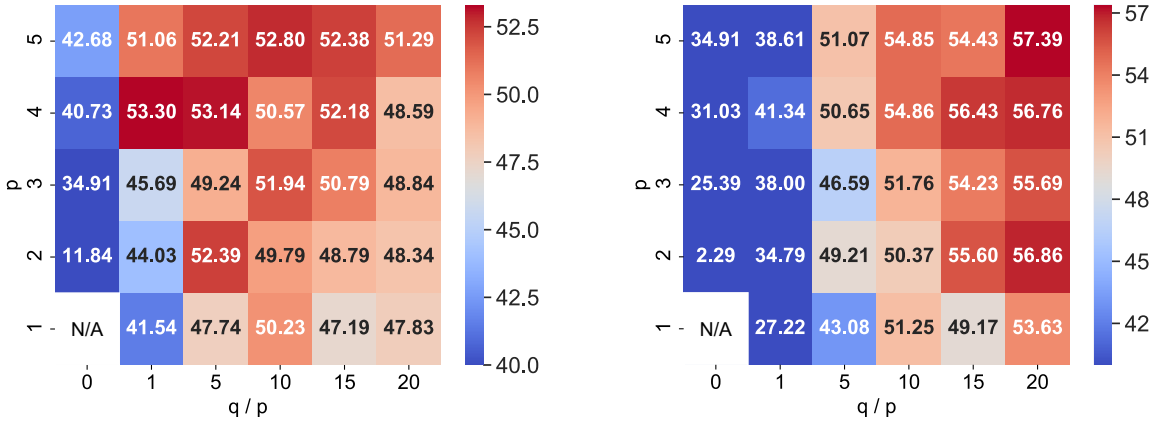

Figure 5: Link prediction performance measured by Hits@20 of $\mathcal{L}_{\text{LLP}_R}$ (left) and $\mathcal{L}_{\text{LLP}_D}$ (right) on Pubmed with different p and q.

and a higher random sampled rate(q/p). And random sampling rate can lead to a much better performance than nearby nodes. However, in $\mathcal{L}_{\text{LLP}_R}$, we find that the results on the diagonal perform consistently better than those around, which means the random sampling rate should match the nearby nodes to work well. Besides, we can also observe that $\mathcal{L}_{\text{LLP}_R}$ is more sensitive with a smaller number of context nodes than $\mathcal{L}_{\text{LLP}_D}$. $\mathcal{L}_{\text{LLP}_R}$ matches the performance by the relative ranking of the context nodes w.r.t. the anchor nodes. However, it becomes difficult for $\mathcal{L}_{\text{LLP}_R}$ to learn well when there are many context nodes. In contrast, $\mathcal{L}_{\text{LLP}_D}$ matches the distribution, and more context nodes provide a clearer picture about the link-related structure around the anchor node.

Sensitivity Analysis on the Fixed-Length and Repeating Times of Random Walks for the Context Nodes. We further conduct sensitivity analysis on the fixed lengths and repeating times of random walks for the context nodes on the CS dataset, which is shown in Table 13. For the fixed length of the random walk, we find that even a relatively short length can effectively retain the local structure information. Additionally, we find that repeating the process multiple times may not lead to a significant performance boost beyond thrice, since it may already cover most of the adjacent nodes within three hops.

Ablation Study on Different Sampling Strategies. Moreover, we conduct an ablation study on different efficient sampling strategies for each anchor node: 1) sampling with only repeated random walks 2) using all 3-hops Neighbors as samples 3) only randomly sampling nodes from the whole graph 4) LLP (the combination of the first and third strategies). The results are shown in Table 14. From the table, we observe that LLP yields the best performance by combining the local samples with global random samples. We note that LLP can also incorporate other sampling strategies, where more complex ones can potentially further improve the performances. We leave this as future work.

Table 14: Link prediction performance evaluated by Hits@20 of LLP and different context nodes sampling strategies for each anchor node on CS.

Random Walk only	3-hop Neighbors	Random Sample only	LLP
60.35	62.23	64.92	67.82

D.6. Analysis of $\mathcal{L}_{\text{LLP}_R}$, $\mathcal{L}_{\text{LLP}_D}$ and $\mathcal{L}_{sup}$ for LLP

Importance Analysis of $\mathcal{L}_{\text{LLP}_R}$, $\mathcal{L}_{\text{LLP}_D}$ and $\mathcal{L}_{sup}$. We conduct the ablation study on all non-OGB datasets to analyze the contributions of each component in Equation (7). In each ablation setting, we remove one component independently, as shown in Figure 6. We can observe that the performance drops when any of the three components (i.e., $\mathcal{L}_{\text{LLP}_R}$, $\mathcal{L}_{\text{LLP}_D}$, and $\mathcal{L}_{sup}$) is removed, which shows the importance of each component. It also demonstrates that $\mathcal{L}_{\text{LLP}_R}$ and $\mathcal{L}_{\text{LLP}_D}$ indeed provide complementary link prediction-related information for the student. Other than these two components, we find that the true link label information also contributes, especially under the production setting. In the production setting, as the neighbor information is sparse or absent, the limited true label information becomes critically important.

Sensitivity Analysis of α, β, γ in LLP. Table 15 presents the sensitiveness experiments about α, β, and γ on CS dataset. We can observe that LLP is generally robust to the choices of these parameters. We also note that LLP is slightly more sensitive to α and γ, which means the results more rely on the ground-truth label loss and distribution-based matching loss training on CS dataset.

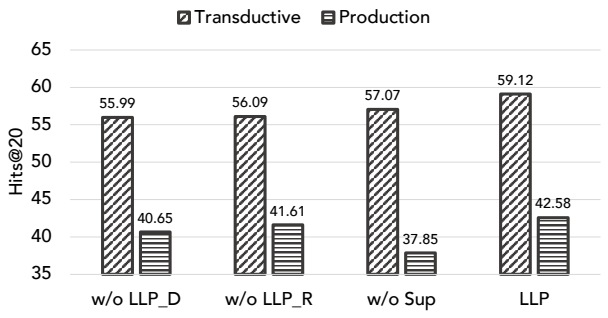

Figure 6: Link prediction performance of LLP by dropping each component in Equation (7). The result shown by each bar is the averaged Hits@20 across all the datasets.

Table 15: Link prediction performance measured by Hits@20 of LLP with different δ, α, β, and γ on CS.

δ	0	0.001	0.05	0.1	0.15	0.2	0.5
Hits@20	64.11	65.97	67.06	68.62	67.94	67.41	67.01
α	0.001	0.01	0.1	1	10	100	1000
Hits@20	63.8	65.1	64.17	65.43	68.62	65.59	58.48
β	0.001	0.01	0.1	1	10	100	1000
Hits@20	67.18	67.85	68.62	67.96	67.48	67.77	65.9
γ	0.001	0.01	0.1	1	10	100	1000
Hits@20	57.45	55.47	57.41	57.86	62.73	68.62	63.43

D.7. Analysis of δ in $\mathcal{L}_{\text{LLP}_R}$

Necessity Analysis of δ in $\mathcal{L}_{\text{LLP}_R}$. To analyze the necessity of δ in $\mathcal{L}_{\text{LLP}_R}$, we conduct experiments on all non-OGB datasets to compare the results using $\mathcal{L}_{\text{LLP}_R}$ with and without δ. The results are shown in Table 16. We observe that the results of $\mathcal{L}_{\text{LLP}_R}$ without δ always approach zero after several training epochs. It demonstrates the effectiveness of δ in avoiding noise and transferring useful knowledge to the student.

Sensitivity Analysis of δ in LLP. We conduct the sensitivity experiments about δ on CS dataset. The results are shown in Table 15. We can observe that LLP is generally robust to the choices of this parameter.

D.8. Further Comparison Results on Cold Start Nodes

We further compare LLP with another related work (Alon & Yahav, 2020), which does not only rely on the connection relationship in the graph either. This paper identifies a bottleneck of graph neural networks, and proposes to modify the last layer to be a fully-adjacent layer (FA) as a simple strategy to circumvent the bottleneck problem. To evaluate its performance in cold-start setting, we modify the last layer of GNNs with a fully-adjacent layer following it. The results compared with LLP are shown in Table 17. We observe that this method is not suitable for large datasets – it results in a dense $N \times N$ adjacency matrix and results in each node receiving messages from N other nodes (which is problematic when N is large). Most datasets we utilized in our paper with a fully-adjacent layer can not fit into an NVIDIA A100 GPU (40GB memory). The performance of GNN+FA on Cora and Citeseer is even worse than vanilla GNNs. The potential reason is link prediction tasks do not heavily depend on long-range information, where the best results were found using only 2-3 layers - we observe that Alon & Yahav (2020) focuses evaluation on smaller datasets that are sensitive to long-range dependencies, which seems to be a mismatch with our intended setting.

D.9. Comparison between LLP and Other Distillation/Rank-based Matching Methods

Compared with Other Distillation Methods. Different from the link prediction tasks like RankDistill (Reddi et al., 2021) and Topology Distillation (Kang et al., 2021) which distill the information in a non-graph context to keep the rank information among entities, we distill in a graph context to keep the graph structure information. **For the sampling methods,** RankDistill samples the nodes to teach the student based on the teacher's results. Same for Topology Distillation, which assigns different entities into different groups and builds a hierarchical topology structure accordingly to distill the topology information. Different from these two methods, LLP samples node pairs independently of the teacher, which is only based on the graph structure. **For the matching methods,** RankDistill keeps the order for "top-K" items and penalizes high scores by the student for "bottom-K" items. Topology Distillation generates fully connected graphs based on entities or groups, where the adjacency matrix is built by the similarity among nodes in the generated graphs. Topology Distillation matches the adjacency matrix generated by the teacher and the student to preserve the topology knowledge. Differently, our method matches both the order and the distribution of the sampled node pairs between teacher and student, which we show are both useful and complementary Table 6 in preserving structure information distillation between the two.

We also conduct experiments to evaluate the impact of the different methods in our prediction task. For Topology Distillation, we pick their full topology distillation (FTD) method for the evaluation, as the authors demonstrate the student model takes more benefits from FTD when there is no significant capacity gap between the teacher and the student. The results are shown in Table 18. We can observe that LLP consistently outperforms RankDistill and Topology Distillation. We believe this is because our sampling and matching methods based on the graph topology structure are more effective in preserving relevant graph structure and graph link prediction-related knowledge than the distillation methods proposed for item recommendation.

Table 16: Link prediction performance measured by Hits@20 of $\mathcal{L}_{\text{LLP}_R}$ with and without δ.

Method	Cora	Citeseer	Pubmed	CS	Physics	Computers	Photos
$\mathcal{L}_{\text{LLP}_R}$	76.52	75.23	53.30	68.30	60.28	25.98	33.33
$\mathcal{L}_{\text{LLP}_R}$ w/o δ	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 17: Link prediction performance measured by Hits@20 of LLP and GNN+FA (Alon & Yahav, 2020) on cold-start nodes.

Method	Cora	Citeseer	Pubmed	CS	Physics	Computers	Photos
GNN	6.39	11.04	4.63	9.46	5.46	1.53	0.87
GNN+FA	2.03	2.89	OOM	OOM	OOM	OOM	OOM
LLP	22.01	32.09	37.68	46.83	39.37	14.64	23.79

Table 18: Link prediction performance measured by Hits@20 of LLP and distillation methods for other link prediction tasks.

	RankDistill	Topology Distillation	LLP
Cora	74.29	75.51	78.82
Citeseer	70.44	71.47	77.32
Pubmed	39.28	37.89	57.33
CS	44.55	60.16	68.62
Physics	49.11	55.32	72.01
Computers	15.64	23.89	35.32
Photos	28.75	38.87	49.32

Table 19: Link prediction performance measured by Hits@20 of LLP and ListNet (Cao et al., 2007).

	ListNet	$\mathcal{L}_{\text{LLP}_R}$
Cora	72.64	76.52
Citeseer	59.82	75.23
Pubmed	18.79	53.3
CS	43.06	68.3
Physics	28.87	60.28
Computers	17.16	25.98
Photos	25.68	33.33

Compared with Other Rank-based Matching Methods. While there exist multiple techniques that can be used to implement the rank-based matching for our task, we note that our task (rank-based matching for preserving graph structural information) is not the same as other rank-matching tasks (i.e. KD for recommendation tasks). Therefore, existing rank-matching methods might not be suitable for KD in link prediction. In Table 19, we compare ListNet (Cao et al., 2007) with our proposed rank-based matching $\mathcal{L}_{\text{LLP}_R}$. We can observe that $\mathcal{L}_{\text{LLP}_R}$ always outperforms ListNet across all datasets. This observation is consistent with the findings presented in Appendix D.7: the conventional margin-based ranking loss cannot work well on our tasks. This is also the reason we introduce the term δ into our rank-matching loss Equation (5). The existing rank-based matching methods match the strict ranking order for all candidate items, as needed by the task of recommendation. But for our use case of preserving graph structural information, our proposed $\mathcal{L}_{\text{LLP}_R}$ is more suitable due to its ability to not differentiate similar nodes within the same hop.

E. Implementation Details

Transductive Setting. Inspired by GLNN (Zhang et al., 2021b), we enlarge the size of the student MLP in our experiment. As suggested by GLNN, this can significantly shorten the gap between the student MLP and the teacher GNN without greatly reducing the timing performance. We set the hidden dimension of student MLP two times larger than the teacher for Physics, Computers, and Photos, and set it four times larger than the teacher for Collab and Citation2. We examine the timing performance of the enlarged students by repeating the inference task ten times. The inference time of LLP increases from 1.9 to 7.1 ms on Collab and from 2.9 to 15.2 ms on Citation2, but it is still 18.9× and 147× faster than SAGE, respectively.

Hyper-parameters. We take a 2-layer SAGE (hidden size is set to 256) as the teacher for most datasets. For Citation2, IGB-100K, and IGB-1M, we set the layer size as 3 for better prediction results. We also take 3-layer MLP as the student on these datasets. For LLP, we conduct the hyperparameter search of the weights for $\mathcal{L}_{sup}$, $\mathcal{L}_{\text{LLP}_R}$ and $\mathcal{L}_{\text{LLP}_D}$ from [0.001, 0.01, 0.1, 1, 10, 100, 1000], the number of the nearby nodes p from [1,2,3,4,5], the random sampling rate q/p from [1, 3, 5, 10, 15], the learning rate from [0.001, 0.005] and the dropout rate from [0, 0.5].

Implementation and Hardware Details. Our implementation is based on PyTorch Geometric (Fey & Lenssen, 2019). We conduct experiments with NVIDIA V100 GPU(16GB memory). For Citation2 and IGB datasets, we run the experiments on NVIDIA A100 GPU with 40GB memory.