LINKLESS LINK PREDICTION VIA RELATIONAL DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) have been widely used on graph data and have shown exceptional performance in the task of link prediction. Despite their effectiveness, GNNs often suffer from high latency due to non-trivial neighborhood data dependency in practical deployments. To address this issue, researchers have proposed methods based on knowledge distillation (KD) to transfer the knowledge from teacher GNNs to student MLPs, which are known to be efficient even with industrial scale data, and have shown promising results on node classification. Nonetheless, using KD to accelerate link prediction is still unexplored. In this work, we start with exploring two direct analogs of traditional KD 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). 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 two matching strategies that complement each other: rank-based matching and distribution-based matching. Extensive experiments demonstrate that LLP boosts the link prediction performance of MLPs with significant margins, and even outperforms the teacher GNNs on 6 out of 9 benchmarks. LLP also achieves a 776.37$\times$ speedup in link prediction inference compared to GNNs on the large scale `OGB-Citation2` 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; Ying et al., 2018b), and link prediction (Zhang & Chen, 2018).

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 friend recommendation on social platforms (Sankar et al., 2021; Tang 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., 2022).

The success of GNNs is typically attributed to the explicit use of contextual information from nodes' surrounding neighborhoods (Zhang et al., 2020d). 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., 2020a; 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 Section 4. Nonetheless, having no graph dependency makes the training and inference time for MLPs negligible when comparing

with those of GNNs. Thus, in industrial-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., 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., 2022), 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 9 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 17.13 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 6/9 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. **776.37**$\times$ faster on the large-scale `OGB-Citation2` 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 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; Zhao et al., 2022; Davidson et al., 2018; Zhu et al., 2021; Yun et al., 2021), 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 operation and an update operation:

$$\boldsymbol{h}_i^l = \textsc{Update}_l\big(\boldsymbol{h}_i^{l-1}, \textsc{Aggregate}_l(\{\boldsymbol{h}_j^{l-1} | e_{i,j} \in \mathcal{E}\})\big), \tag{1}$$

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:

$$y_{i,j} = \sigma(\textsc{Decoder}(\boldsymbol{h}_i, \boldsymbol{h}_j)), \tag{2}$$

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., 2022; 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., 2020b; 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 $\hat{y}_{i,j} = \sigma(\textsc{Decoder}(\hat{\boldsymbol{h}}_i, \hat{\boldsymbol{h}}_j))$.

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 preservation of relationships across samples between teacher and student (Park et al., 2019). We next discuss our proposals in more detail, which are also summarized in Figure 1.

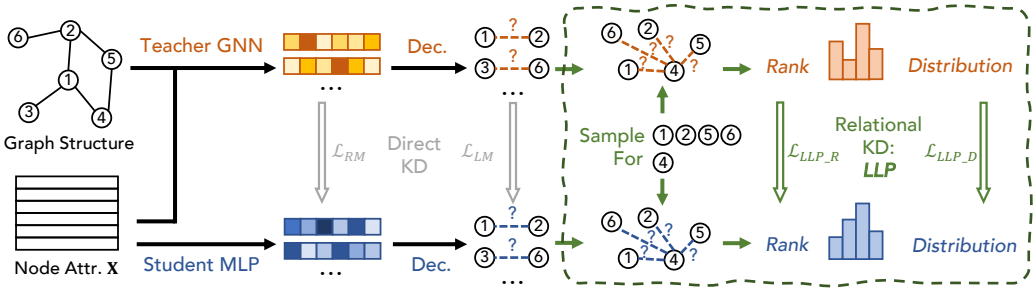

Figure 1: We explore KD methods for link prediction, which distill knowledge from a Teacher GNN to a Student MLP encoder, each with their own decoder. We start by exploring direct KD methods: representation-matching and logit-matching. Upon observing their drawbacks of not being able to distill relational information, we further propose a relational KD framework: LLP, 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.

3.1 Direct Methods

Logit-matching is one straightforward strategy to distill knowledge from the teacher to the student, and has been shown to be extremely effective on node classification tasks. For example, Zhang et al. (2021b) utilizes the soft prediction targets generated by GNNs to help supervise the student MLP. In a similar vein, we generate the soft score $y_{i,j}$ for the node pair or candidate edge (i, j) with the teacher GNN model, and train the student to match its prediction $\hat{y}_{i,j}$ on this target:

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

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 in conjunction 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). In other words, 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., 2022) 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. In practice, 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 $\hat{\mathcal{Y}}_v = \{\hat{y}_{v,i} | i \in \mathcal{S}_v\}$. 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 9 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

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

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 $\mathcal{S}_v$ for each anchor node v to limit its size. First, to preserve the local structure around v, we sample p nearby nodes via fixed-length random walks, denoted as $\mathcal{S}_v^N$. On the other hand, we also randomly sample q nodes from G (which are likely to be far-away from v) to form $\mathcal{S}_v^R$, which additionally preserves the global structure w.r.t. v in the graph. p and q are hyper-parameters. Finally, we make $\mathcal{S}_v$ as the union of the nearby samples and random samples, i.e., $\mathcal{S}_v = \mathcal{S}_v^N \cup \mathcal{S}_v^R$. We note that LLP can easily adopt more sophisticated sampling strategies for further improvements; we leave this to future exploration.

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 9 benchmark datasets, including 7 commonly used link prediction benchmarks (`Cora`, `Citeseer`, `Pubmed`, `Computers`, `Photos`, `CS`, and `Physics`) and 2 larger-scale link prediction benchmarks from OGB (Hu et al., 2020) (`OGB-Collab` and `OGB-Citation2`). Further details and statistics of the datasets are summarized in Appendix B.

Evaluation Settings and Metrics. 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; Mikolov et al., 2013). 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 two OGB datasets are already temporally split in their public releases. We further elaborate the details of the production setting as well as the statistics in Appendix C.

Table 1: Link prediction performance under transductive setting. 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	<u>78.06</u>±1.50	74.72±4.27	75.75±1.51	**78.82**±1.74	3.07	0.76	4.44
Citeseer	<u>73.89</u>±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	<u>51.98</u>±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	<u>61.10</u>±2.83	**68.62**±1.46	7.52	34.61	9.11
Physics	<u>66.74</u>±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	<u>31.66</u>±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	<u>49.32</u>±2.64	10.85	21.49	-2.18
OGB-Collab	**48.69**±0.87	36.95±1.37	35.97±0.96	36.86±0.45	<u>45.27</u>±0.79	8.41	8.32	-3.42
OGB-Citation2	**82.56**±0.04	40.63±0.00	38.42±0.01	42.50±0.01	<u>53.20</u>±1.20	10.70	12.57	-29.36

For OGB datasets, we use their official metric (Hits@50 for OGB-Collab and Mean reciprocal Rank (MRR) for OGB-Citation2) following the public leaderboard[1]. For other datasets, following previous works (Yun et al., 2021; Zhang et al., 2021a; Zhao et al., 2022), 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.

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-based matching (Equation (3)); "$\mathcal{L}_{RM}$" refers to MLP trained with node representation-based 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 Figure 5 (in Appendix D.3).

4.2 Link Prediction Results

Transductive Setting. Table 1 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 **17.13** points and **10.17** 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 6 out of 9 datasets, demonstrating that our proposed rank-based and distribution-based matching are able to effectively distill the relational knowledge for link prediction.

Production Setting. Table 2 shows the link prediction performance of the proposed LLP with GNN, MLP, and the direct KD methods in the production setting. For ease of comparison, we also stratify each method's performance on the three different categories of the test edges. We observe that LLP is still able to consistently outperform MLP and direct KD methods by large margins for all test categories. Specifically, LLP achieves **12.01** and **6.67** on Hits@20 improvements over MLP and direct KD methods averaged over datasets, respectively. Moreover, LLP is at or above par with the teacher GNN on 3 out of the 6 datasets. From the 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 improvements of LLP over MLP is valuable for large-scale industrial applications, given their popularity in practice.

[1] https://ogb.stanford.edu/docs/leader_linkprop/

Table 2: The performance of production setting. Overall shows the overall results; Old-Old shows the results on Old-Old node pairs; Old-New shows results on Old-New node pairs, and New-New is the result on New-New node pairs.

	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

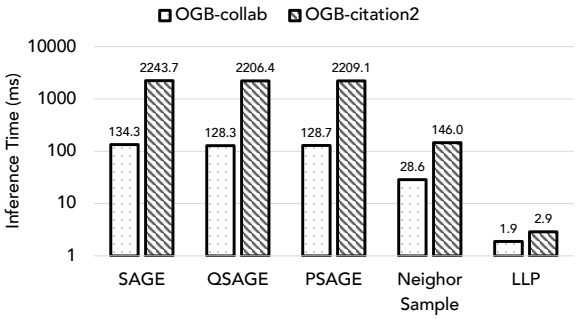

Figure 2: Inference time comparison of LLP with GNN inference acceleration methods in log scale.

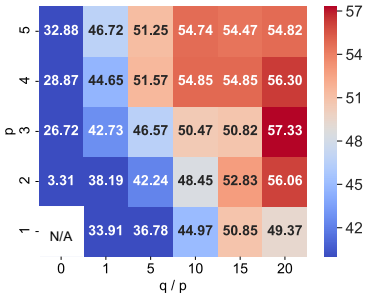

Figure 3: Link prediction performance of LLP on Pubmed with different number of samples for the context nodes.

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 10 randomly chosen nodes in the graph. We evaluate against 4 common GNN inference acceleration 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 datasets. We can observe that LLP can obtain **71.04**× and **776.37**× speedup

comparing with on SAGE on OGB-Collab and OGB-Citation2 datasets, respectively. Comparing with the best acceleration method Neighbor Sampling (which reduces graph dependency, but does not eliminate it like LLP), LLP still achieves **15.12**× and **50.51**× speedup on the two datasets, respectively. This is because all these inference acceleration methods still rely on the graph structure.

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 nodes are isolated (see Appendix C for more details). Table 3 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.

Table 3: Link pred. results 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

4.5 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 4 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. 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.

Table 4: Ablation study on 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

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 low number of random samples show 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-based and representation-based 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 **776.37**× over GNNs), while also improving link prediction performance over MLPs by **17.13** and **12.01** points in transductive and production settings, matching or outperforming the teacher GNN in 6/9 datasets in transductive setting and 3/9 datasets in production setting, and with notable **25.29** on Hits@20 improvements on cold-start nodes.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our experiments, we provide the source code for LLP in the supplementary materials. Moreover, we will publically open-source our code later after we clean up our code package and add proper documentation for it. The hyper-parameters that are required to reproduce our experiments are provided in Appendix E.

REFERENCES

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 2003.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 1999.

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

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. *arXiv preprint arXiv:2110.02910*, 2021.

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

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

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

Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 2019.

Defang Chen, Jian-Ping Mei, Yuan Zhang, Can Wang, Zhe Wang, Yan Feng, and Chun Chen. Cross-layer distillation with semantic calibration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021a.

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

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, 2020.

Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning*, pp. 1695–1706. PMLR, 2021c.

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

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

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

Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318*, 2021.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pp. 1607–1616. PMLR, 2018.

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

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

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

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 2021.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 2017.

Yu Hao, Xin Cao, Yixiang Fang, Xike Xie, and Sibo Wang. Inductive link prediction for nodes having only attribute information. *arXiv preprint arXiv:2007.08053*, 2020.

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

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

Yang Hu, Haoxuan You, Zhecan Wang, Zhicheng Wang, Erjin Zhou, and Yue Gao. Graph-mlp: node classification without message passing in graph. *arXiv preprint arXiv:2106.04051*, 2021.

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

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

Zhihao Jia, Sina Lin, Rex Ying, Jiaxuan You, Jure Leskovec, and Alex Aiken. Redundancy-free computation for graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Chaitanya K Joshi, Fayao Liu, Xu Xun, Jie Lin, and Chuan-Sheng Foo. On representation knowledge distillation for graph neural networks. *arXiv preprint arXiv:2111.04964*, 2021.

Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. *Advances in neural information processing systems*, 2018.

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

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

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. From zero-shot learning to cold-start recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, 2019.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

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

Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)*, 49(4):1–33, 2016.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

S Yu Philip, Jiawei Han, and Christos Faloutsos. *Link mining: Models, algorithms, and applications*. 2010.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. Graph neural networks for friend ranking in large-scale social platforms. In *Proceedings of the Web Conference*, pp. 2535–2546, 2021.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018.

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

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

Xianfeng Tang, Yozen Liu, Xinran He, Suhang Wang, and Neil Shah. 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.

Yijun Tian, Chuxu Zhang, Zhichun Guo, Xiangliang Zhang, and Nitesh V Chawla. Nosmog: Learning noise-robust and structure-aware mlps on graphs. *arXiv preprint arXiv:2208.10010*, 2022.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pp. 2071–2080. PMLR, 2016.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 world wide web conference*, pp. 539–548, 2018.

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

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

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 2020a.

Xiaobo Wang, Tianyu Fu, Shengcai Liao, Shuo Wang, Zhen Lei, and Tao Mei. Exclusivity-consistency regularized knowledge distillation for face recognition. In *European Conference on Computer Vision*, 2020b.

Zhitao Wang, Yong Zhou, Litao Hong, Yuanhang Zou, and Hanjing Su. Pairwise learning for neural link prediction. *arXiv preprint arXiv:2112.02936*, 2021.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pp. 5453–5462. PMLR, 2018.

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

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

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

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 2016.

Haoteng Yin, Muhan Zhang, Yanbang Wang, Jianguo Wang, and Pan Li. Algorithm and system co-design for efficient subgraph-based graph representation learning. *arXiv preprint arXiv:2202.13538*, 2022.

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

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 2018b.

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

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-saint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Dalong Zhang, Xin Huang, Ziqi Liu, Zhiyang Hu, Xianzheng Song, Zhibang Ge, Zhiqiang Zhang, Lin Wang, Jun Zhou, Yang Shuang, et al. Agl: a scalable system for industrial-purpose graph machine learning. *arXiv preprint arXiv:2003.02454*, 2020a.

Hanlin Zhang, Shuai Lin, Weiyang Liu, Pan Zhou, Jian Tang, Xiaodan Liang, and Eric P Xing. Iterative graph self-distillation. *arXiv preprint arXiv:2010.12609*, 2020b.

Muhan Zhang and Yixin Chen. 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.

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

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

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

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

Wentao Zhang, Xupeng Miao, Yingxia Shao, Jiawei Jiang, Lei Chen, Olivier Ruas, and Bin Cui. Reliable data distillation on graph convolutional network. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020c.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020d.

Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any gnn with local structure awareness. *arXiv preprint arXiv:2110.03753*, 2021.

Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. Learning from counterfactual links for link prediction. In *International Conference on Machine Learning*, pp. 26911–26926. PMLR, 2022.

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

Wenqing Zheng, Edward W Huang, Nikhil Rao, Sumeet Katariya, Zhangyang Wang, and Karthik Subbian. Cold brew: Distilling graph node representations with incomplete or missing neighborhoods. *arXiv preprint arXiv:2111.04840*, 2021.

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

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

A FURTHER RELATED WORK

In this section, we discuss other work related to LLP.

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; Ma et al., 2021) paradigm. Different GNN customizations include degree normalization (Kipf & Welling, 2016a), neighbor sampling and neighbor separation (Hamilton et al., 2017), self-attention (Veličković et al., 2017), residual connections (Xu et al., 2018), and many more. Moreover, there are subgraph-based methods (Bevilacqua et al., 2021; Zhao et al., 2021), and tensor-based methods (Maron et al., 2019; Geerts & Reutter, 2022) for more expressive GNNs.

Link Prediction Methods. Link prediction has achieved great attention from the research community, considering its wide applications. Heuristic methods (Philip et al., 2010) are 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 are proposed to further improve the effectiveness. Link prediction with GNN is another important direction for link prediction (Yun et al., 2021), which is based on the learned embeddings. Besides the popular node-wise encoder and link-wise decoder strategy discussed in Section 2, another line of work casts link prediction tasks as classification problems on the enclosing subgraphs around each link (Zhang & Chen, 2018; Cai & Ji, 2020; Cai et al., 2021). Although these methods can improve task performance, they are usually extremely computationally expensive and cannot scale well in practical use-cases (Yin et al., 2022). Similarly, Zhu et al. (2021) proposes a GNN link prediction paradigm by encoding information of all paths between two nodes, which is also very expensive. Our work focuses on accelerating the more general and faster methods discussed in Section 2.

GNN Inference Acceleration. Pruning (Zhou et al., 2021; Chen et al., 2021c) and quantization (Zhao et al., 2020; Tailor et al., 2020) strategies are 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) and try to accelerate GNNs by distilling them to MLP. These works focus on KD for node classification tasks, whereas we focus on link prediction tasks.

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, some work (Park et al., 2019; Tung & Mori, 2019; Joshi et al., 2021) propose relational KD, which corresponds to instance-to-instance KD while preserving metrics among representations of similar instances. For GNNs, Yang et al. (2020) shares this idea by using 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 relationship of a pair of nodes among other node pairs and make better link prediction.

GNN KD. Existing GNN KD work are mostly based on the logit-based KD (Hinton et al., 2015) to obtain light-weight models (Zhang et al., 2020c; Zheng et al., 2021; Yang et al., 2021). TinyGNN (Yan et al., 2020) proposes to train a student GNN with fewer parameters using KD. Yang et al. (2021) improve the designed student 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., 2020) and G-CRD (Joshi et al., 2021) propose 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.

Table 5: Statistics of all datasets used in the experiments.

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
OGB-Collab	235,868	1,285,465	128
OGB-Citation2	2,927,963	30,561,187	128

Table 6: Detailed statistics of data 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
OGB-Collab	188,695	47,173	164,022	82,655	10,414

B ADDITIONAL DATASETS DETAILS

Here, we present the details of the datasets used in our experiments. Cora, Citeseer, Pubmed (Yang et al., 2016) and OGB-Citation2 (Wang et al., 2020a; Mikolov et al., 2013) are all representative citation network datasets, where the nodes and edges represent papers and citations, respectively. CS, Physics (Shchur et al., 2018) and OGB-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. The detailed statistics of these datasets are shown in Table 5.

C ADDITIONAL EVALUATION SETTING DETAILS

C.1 TRANSDUCTIVE SETTING

The transductive setting is a standard setting for link prediction, 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

To match realistic applications, we design a new production setting for link prediction tasks, in which we expect there to be new nodes and new edges evident in the graph at inference time. These new nodes and edges are not seen during the training process. Intuitively, this matches a scenario of a growing social network.

We further explain our implementation for this setting in detail: Given the graph $G = (\mathcal{V}, \mathcal{E})$, we randomly sample 10% nodes $\mathcal{V}^N$ from the $\mathcal{V}$ as the new nodes and remove them from the training graph. For Cora and Citeseer, we sample 30% nodes as new nodes because these datasets are very small. The remaining nodes are placed in $\mathcal{V}^O$, which denotes the visible nodes during model

Table 7: 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

training. Then we split the $\mathcal{E}$ into three parts based on the node splits, as $\mathcal{E}^{O-O}$, $\mathcal{E}^{O-N}$ and $\mathcal{E}^{N-N}$, which denote the links connecting existing–existing, existing–new and new–new node pairs. Next we split existing–existing node pairs into three parts followed 80/10/10 splitting ratio to be training edges, new incoming edges (for message passing), and testing edges. We split $\mathcal{E}^{O-N}$ and $\mathcal{E}^{N-N}$ into two parts both following a 90/10 splitting ratio: new coming edges for message passing and testing edges. We will merge the training edges, new incoming edges sampled from $\mathcal{E}^{O-N}$ and $\mathcal{E}^{N-N}$ for the message passing. Then we can test our methods on three separate testing edge sets sampled from $\mathcal{E}^{O-O}$, $\mathcal{E}^{O-N}$ and $\mathcal{E}^{N-N}$ respectively. The detailed statistic of this setting is shown in Table 6.

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 showed in Section 4.4 are conducted with this setting.

D Additional Experimental Results

D.1 AUC results on Transductive and Production Settings.

Here, we present AUC results on all the non-OGB datasets under the transductive setting in Table 7 and production setting in Table 8. In Table 7, we can observe that our method outperforms both MLP and the teacher GNN on all the datasets under the transductive setting. For the production setting, our method performs better than the teacher GNN on 4/7 datasets, as shown in Table 8.

D.2 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 4. 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) 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 provides a clearer picture about the link-related structure around the anchor node.

D.3 Results with Different Teachers.

In our experiments, we use SAGE as the teacher GNN in both transductive and production settings. We further test LLP's performance with other GNNs, such as GCN, GAT, and APPNP. In Figure 5, we find that our model always outperforms MLP with different GNN teachers. However, absolute performance is closely related to the teacher.

Table 8: 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±1.5	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±0.2	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

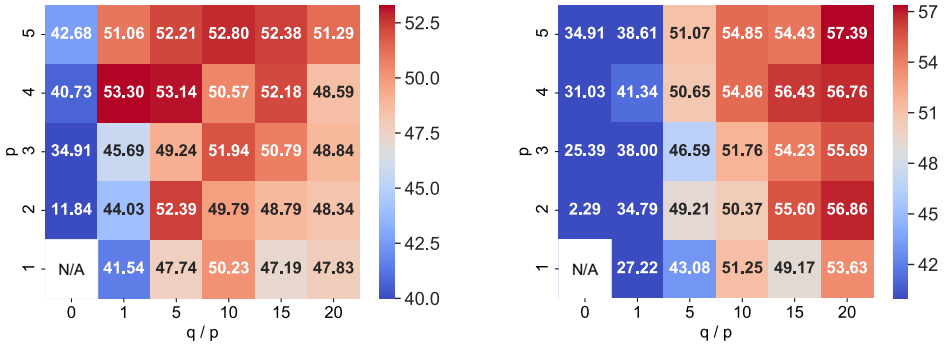

Figure 4: Link prediction performance of $\mathcal{L}_{\text{LLP_}R}$ (left) and $\mathcal{L}_{\text{LLP_}D}$ (right) on Pubmed with different p and q.

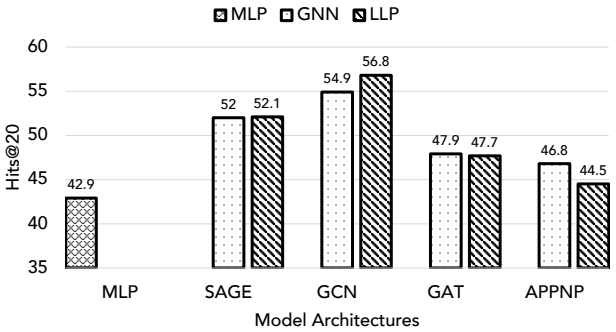

Figure 5: Link prediction performance on Pubmed under the transductive setting with different GNN teachers (SAGE, GAT, GCN, and APPNP).

D.4 Importance Analysis of $\mathcal{L}_{\text{LLP_}R}$, $\mathcal{L}_{\text{LLP_}D}$ and $\mathcal{L}_{sup}$ for LLP.

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.

D.5 Necessity Analysis of δ in $\mathcal{L}_{\text{LLP_}R}$

To analyze the necessity of δ, we designed for $\mathcal{L}_{\text{LLP_}R}$, we conduct the 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 9. We observe that the results of $\mathcal{L}_{\text{LLP_}R}$ without δ always go to zero after several epochs of training. It demonstrates that δ is really important for $\mathcal{L}_{\text{LLP_}R}$ to avoid noise and transfer useful knowledge to the student.

E Implementation Details

Transductive Setting. Under the transductive setting, we originally set the number of layers and the hidden dimension of each layer to be the same as the teacher GNN. With this setting, LLP can highly improve the performance over MLP. However, the gap between LLP and the teacher GNN is still

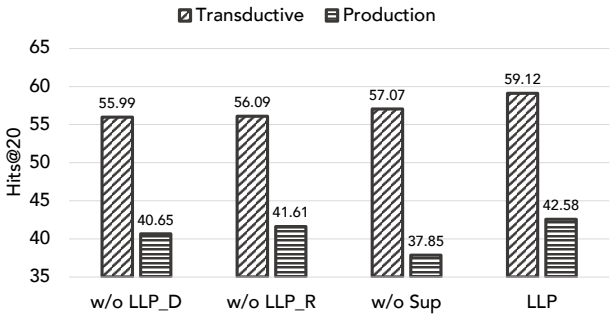

Figure 6: The averaged results across all the datasets by dropping each component in Equation (7).

Table 9: The performance of $\mathcal{L}_{\text{LLP}_R}$ with and without δ.

Methods	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

large. Inspired by GLNN (Zhang et al., 2021b), which observed that enlarging the hidden side of the student MLP can shorten this gap a lot on node classification tasks with very limited time consuming, we also enlarge the hidden dimension of the student model. For `Physics`, `Computers`, and `Photos`, we set the hidden dimension of student MLP two times larger than the teacher and set it four times larger than the teacher. We test the inference time of the enlarged students. We observe that the ten times inference time of LLP on `OGB-Collab` and `OGB-Citation2` change from 1.9 to 7.1 and 2.9 to 15.2 separately, which are still 18.9× and 147× faster than SAGE.

Model Hyper-parameters. We take 2-layer SAGE (hidden size is set to 256) as the teacher for all the non-OGB datasets. For `OGB-Collab` and `OGB-Citation2`, we follow their official implementation to set the layer size as 3. Also, we take 3-layer MLP as the student on these two datasets. We set dropout as 0. For LLP, we do 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], number of the nearby nodes p from [1,2,3,4,5], the random sampling rate q/p from [1, 3, 5, 10, 15] and learning rate from [0.001, 0.005].

Implementation and Hardware Details. Our code is implemented based on PyTorch Geometric (Fey & Lenssen, 2019). We conduct our experiments with NVIDIA V100 GPU(16GB memory). For `OGB-Citation2`, we run the experiments on NVIDIA A100 GPU with 40GB memory.