NODE DUPLICATION IMPROVES COLD-START LINK PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) are prominent in graph machine learning and have shown state-of-the-art performance in Link Prediction (LP) tasks. Nonetheless, recent studies show that GNNs struggle to produce good results on low-degree nodes despite their overall strong performance. In practical applications of LP, like recommendation systems, improving performance on low-degree nodes is critical, as it amounts to tackling the cold-start problem of improving the experiences of users with few observed interactions. In this paper, we investigate improving GNNs' LP performance on low-degree nodes while preserving their performance on high-degree nodes and propose a simple yet surprisingly effective augmentation technique called NODEDUP. Specifically, NODEDUP duplicates low-degree nodes and creates links between nodes and their own duplicates before following the standard supervised LP training scheme. By leveraging a "multi-view" perspective for low-degree nodes, NODEDUP shows significant LP performance improvements on low-degree nodes without compromising any performance on high-degree nodes. Additionally, as a plug-and-play augmentation module, NODEDUP can be easily applied on existing GNNs with very light computational cost. Extensive experiments show that NODEDUP achieves **38.49%**, **13.34%**, and **6.76%** relative improvements on isolated, low-degree, and warm nodes, respectively, on average across all datasets compared to GNNs and state-of-the-art cold-start methods.

1 INTRODUCTION

Link prediction (LP) is a fundamental task of graph-structured data (Liben-Nowell & Kleinberg, 2007; Trouillon et al., 2016), which aims to predict the likelihood of the links existing between two nodes in the network. It has wide-ranging real-world applications across different domains, such as friend recommendations in social media (Sankar et al., 2021; Tang et al., 2022; Fan et al., 2022), product recommendations in e-commerce platforms (Ying et al., 2018; He et al., 2020), knowledge graph completion (Li et al., 2023; Vashishth et al., 2020; Zhang et al., 2020), and chemical interaction prediction (Stanfield et al., 2017; Kovács et al., 2019; Yang et al., 2021).

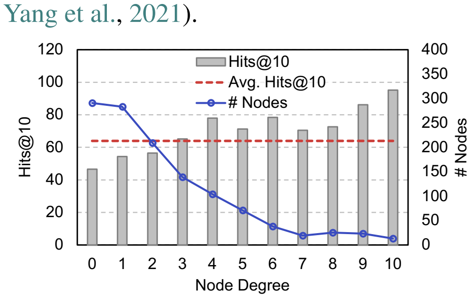


Figure 1: Node Degree Distribution and LP Performance (GSage as an encoder and inner product as a decoder) Distribution w.r.t Nodes Degrees showing reverse trends on `Citeseer` dataset.

In recent years, graph neural networks (GNNs) (Kipf & Welling, 2016a; Veličković et al., 2017; Hamilton et al., 2017) have been widely applied to LP, and a series of cutting-edge models have been proposed (Zhang & Chen, 2018; Zhang et al., 2021; Zhu et al., 2021; Zhao et al., 2022b). Most GNNs follow a message-passing scheme (Gilmer et al., 2017) in which information is iteratively aggregated from neighbors and used to update node representations accordingly. Consequently, the success of GNNs usually heavily relies on having sufficient high-quality neighbors for each node (Zheng et al., 2021; Liu et al., 2021). However, real-world graphs often exhibit long-tailed distribution in terms of node degrees, where a significant fraction of nodes have very few neighbors (Tang et al., 2020b; Ding et al., 2021; Hao et al., 2021). For example, Figure 1 shows the long-tailed degree distribution of the `Citeseer` dataset. Moreover, LP performances w.r.t. node degrees on this dataset also clearly indicate that GNNs struggle to generate satisfactory results for nodes with low or zero degrees. For simplicity, in this paper, we refer to the nodes with low or zero degrees as *cold* nodes and the nodes with higher degrees as *warm* nodes.

To boost GNNs' performance on cold nodes, recent studies have proposed various training strategies (Liu et al., 2020; 2021; Zheng et al., 2021; Hu et al., 2022) and augmentation strategies (Hu et al., 2022; Rong et al., 2019; Zhao et al., 2022b) to improve representation learning quality. For instance, ColdBrew (Zheng et al., 2021) posits that training a powerful MLP can rediscover missing neighbor information for cold nodes; TailGNN (Liu et al., 2021) utilizes a cold-node-specific module to accomplish the same objective. However, such advanced training strategies (e.g., ColdBrew and TailGNN) share a notable drawback: they are trained with a bias towards cold nodes, which then sacrifices performance on warm nodes (empirically validated in Table 1). However, in real-world applications, both cold nodes and warm nodes are critical (Clauset et al., 2009). On the other hand, while augmentation methods such as LAGNN (Liu et al., 2022b) do not have such bias, they primarily focus on improving the overall performance of GNNs in LP tasks, which may be dominated by warm nodes due to their higher connectivity. Additionally, the augmentation methods usually introduce a significant amount of extra computational costs (empirically validated in Figure 5). In light of the existing work discussed above on improving LP performance for cold nodes, we are naturally motivated to explore the following crucial but rather unexplored research question:

Can we improve LP performance on cold nodes without compromising warm node performance?

We observe that cold node LP performance usually suffers because they are under-represented in standard supervised LP training due to their few (if any) connections. Given this observation, in this work, we introduce a simple yet effective augmentation method, NODEDUP, for improving LP performance on cold nodes. Specifically, NODEDUP duplicates cold nodes and establishes edges between each original cold node and its corresponding duplicate. Subsequently, we conduct standard supervised end-to-end training of GNNs on the augmented graph. To better understand why NODEDUP is able to improve LP performance for cold nodes, we thoroughly analyze it from multiple perspectives, during which we discover that this simple technique effectively offers a "multi-view" perspective of cold nodes during training. This "multi-view" perspective of the cold nodes acts similarly to an ensemble and drives performance improvements for these nodes. Additionally, our straightforward augmentation method provides valuable supervised training signals for cold nodes and especially isolated nodes. Furthermore, we also introduce NODEDUP(L), a lightweight variation of NODEDUP that adds only self-loop edges into training edges for cold nodes. NODEDUP(L) empirically offers up to a $1.3\times$ speedup over NODEDUP for the training process and achieves significant speedup over existing augmentation baselines. In our experiments, we comprehensively evaluate our method on seven benchmark datasets. Compared to GNNs and state-of-the-art cold-start methods, NODEDUP achieves **38.49%**, **13.34%**, and **6.76%** relative improvements on isolated, low-degree, and warm nodes, respectively, on average across all datasets. NODEDUP also greatly outperforms augmentation baselines on cold nodes, with comparable warm node performance. Finally, as plug-and-play augmentation methods, our methods are versatile and effective with different LP encoders/decoders. They also achieve significant performance in a more realistic inductive setting. Our code can be found at `https://anonymous.4open.science/r/NodeDup-0241/README.md`.

2 PRELIMINARIES

Notation. Let an attributed graph be $G = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, where $\mathcal{V}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edges where each $e_{vu} \in \mathcal{E}$ indicates nodes v and u are linked. Let $\mathbf{X} \in \mathbb{R}^{N \times F}$ be the node attribute matrix, where F is the attribute dimension. Let $\mathcal{N}_v$ be the set of neighbors of node v, i.e., $\mathcal{N}_v = \{u | e_{vu} \in \mathcal{E}\}$, and the degree of node v is $|\mathcal{N}_v|$. We separate the set of nodes $\mathcal{V}$ into three disjoint sets $\mathcal{V}_{iso}$, $\mathcal{V}_{low}$, and $\mathcal{V}_{warm}$ by their degrees based on the threshold hyperparameter δ[1]. For each node $v \in \mathcal{V}$, $v \in \mathcal{V}_{iso}$ if $|\mathcal{N}_v| = 0$; $v \in \mathcal{V}_{low}$ if $0 < |\mathcal{N}_v| \leq \delta$; $v \in \mathcal{V}_{warm}$ if $|\mathcal{N}_v| > \delta$. For ease of notation, we also use $\mathcal{V}_{cold} = \mathcal{V}_{iso} \cup \mathcal{V}_{low}$ to denote the cold nodes, which is the union of Isolated and Low-degree nodes.

LP with GNNs. In this work, we follow the commonly-used encoder-decoder framework for GNN-based LP (Kipf & Welling, 2016b; Berg et al., 2017; Schlichtkrull et al., 2018; Ying et al., 2018; Davidson et al., 2018; Zhu et al., 2021; Yun et al., 2021; Zhao et al., 2022b), where a GNN encoder learns the node representations and the decoder predicts the link existence probabilities given each

[1]This threshold δ is set as 2 in our experiments, based on observed performance gaps in LP on various datasets, as shown in Figure 1 and Figure 6. Further reasons for this threshold are detailed in Appendix D.1.

pair of node representations. Most GNNs follow the message passing design (Gilmer et al., 2017) that iteratively aggregate each node's neighbors' information to update its embeddings. Without the loss of generality, for each node v, the l-th layer of a GNN can be defined as

$$\boldsymbol{h}_v^{(l)} = \text{UPDATE}\big(\boldsymbol{h}_v^{(l-1)}, \boldsymbol{m}_v^{(l-1)}\big), \text{s.t. } \boldsymbol{m}_v^{(l-1)} = \text{AGG}\big(\{\boldsymbol{h}_u^{(l-1)}\} : \forall u \in \mathcal{N}_v\big), \tag{1}$$

where $\boldsymbol{h}_v^{(l)}$ is the l-th layer's output representation of node v, $\boldsymbol{h}_v^{(0)} = \boldsymbol{x_v}$, AGG$(\cdot)$ is the (typically permutation-invariant) aggregation function, and UPDATE$(\cdot)$ is the update function that combines node v's neighbor embedding and its own embedding from the previous layer. For any node pair v and u, the decoding process can be defined as $\hat{y}_{vu} = \sigma\big(\text{DECODER}(\boldsymbol{h}_v, \boldsymbol{h}_u)\big)$, where $\boldsymbol{h}_v$ is the GNN's output representation for node v and σ is the Sigmoid function. Following existing literature, we use inner product (Wang et al., 2021; Zheng et al., 2021) as the default DECODER.

The standard supervised LP training optimizes model parameters w.r.t. a training set, which is usually the union of all observed M edges and KM no-edge node pairs (as training with all $O(N^2)$ no-edges is infeasible in practice), where K is the negative sampling rate ($K = 1$ usually). We use $\mathcal{Y} = \{0, 1\}^{M+KM}$ to denote the training set labels, where $y_{vu} = 1$ if $e_{vu} \in \mathcal{E}$ and 0 otherwise.

The Cold-Start Problem. The cold-start problem is prevalent in various domains and scenarios. In recommendation systems (Chen et al., 2020; Lu et al., 2020; Hao et al., 2021; Zhu et al., 2019; Volkovs et al., 2017; Liu & Zheng, 2020), cold-start refers to the lack of sufficient interaction history for new users or items, which makes it challenging to provide accurate recommendations. Similarly, in the context of GNNs, the cold-start problem refers to performance in tasks involving cold nodes, which have few or no neighbors in the graph. As illustrated in Figure 1, GNNs usually struggle with cold nodes in LP tasks due to unreliable or missing neighbors' information. In this work, we focus on enhancing LP performance for cold nodes, specifically predicting the presence of links between a cold node $v \in \mathcal{V}_{cold}$ and target node $u \in \mathcal{V}$ (w.l.o.g.). Additionally, we aim to maintain satisfactory LP performance for warm nodes. Prior studies on cold-start problems (Tang et al., 2020b; Liu et al., 2021; Zheng et al., 2021) inspired this research direction.

3 Node Duplication to Improve Cold-start Performance

We make a simple but powerful observation: *cold nodes are strongly under-represented in the LP training.* Given that they have few or even no directly connected neighbors, they hardly participate in the standard supervised LP training as described in Section 2. For example, a model will not see an isolated node unless it is randomly sampled as a negative training edge for another node. In light of such observations, our proposed augmentation technique is simple: we duplicate under-represented cold nodes. By both training and aggregating with the edges connecting the cold nodes with their duplications, cold nodes are able to gain better visibility in the training process, which allows the GNN-based LP models to learn better representations. In this section, we introduce NodeDup in detail, followed by comprehensive analyses of why it works from different perspectives.

3.1 Proposed Method

The implementation of NodeDup can be summarized into four simple steps: (1): duplicate all cold nodes to generate the augmented node set $\mathcal{V}' = \mathcal{V} \cup \mathcal{V}_{cold}$, whose node feature matrix is then $\mathbf{X}' \in \mathbb{R}^{(N+|\mathcal{V}_{cold}|)\times F}$. (2): for each cold node $v \in \mathcal{V}_{cold}$ and its duplication v', add an edge between them and get the augmented edge set $\mathcal{E}' = \mathcal{E} \cup \{e_{vv'} : \forall v \in \mathcal{V}_{cold}\}$. (3): include the augmented edges into the training set and get $\mathcal{Y}' = \mathcal{Y} \cup \{y_{vv'} = 1 : \forall v \in \mathcal{V}_{cold}\}$. (4): proceed with the standard supervised LP training on the augmented graph $G' = \{\mathcal{V}', \mathcal{E}', \mathbf{X}'\}$ with augmented training set $\mathcal{Y}'$. We also summarize this whole process of NodeDup in Algorithm 1 in Appendix C. The effects of duplication nodes and frequency are discussed in Appendix D.3.

Time Complexity. We discuss complexity of our method in terms of the training process on the augmented graph. We use GSage (Hamilton et al., 2017) and inner product decoder as the default architecture when demonstrating the following complexity (w.l.o.g). With the augmented graph, GSage has a complexity of $O(R^L(N + |\mathcal{V}_{cold}|)D^2)$, where R represents the number of sampled neighbors for each node, L is the number of GSage layers (Wu et al., 2020), and D denotes the size of node representations. In comparison to the non-augmented graph, NodeDup introduces an extra time complexity of $O(R^L|\mathcal{V}_{cold}|D^2)$. For the inner product decoder, we incorporate additionally $|\mathcal{V}_{cold}|$ positive edges and also sample $|\mathcal{V}_{cold}|$ negative edges into the training process, resulting in the extra time complexity of the decoder as $O((M + |\mathcal{V}_{cold}|)D)$. Given that all cold nodes have few

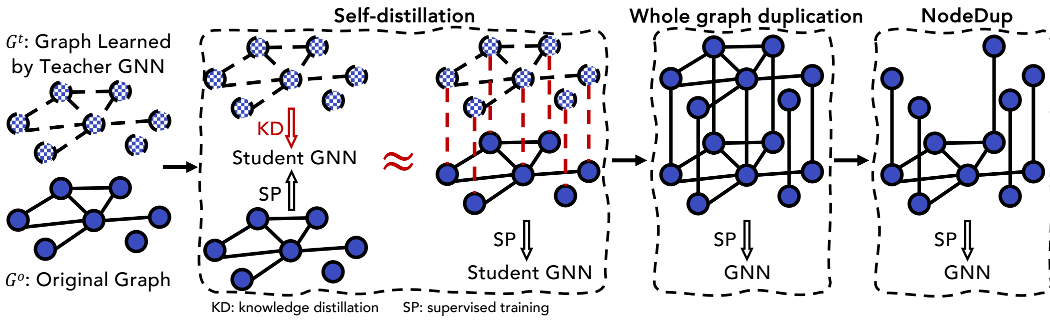


Figure 3: Comparing NODEDUP to self-distillation. The self-distillation process can be approximated by training the student GNN on an augmented graph, which combines G^o, G^t, and edges connecting corresponding nodes in the two graphs. This process can be further improved by replacing G^t with G^o to explore the whole graph duplication. NODEDUP is a lightweight variation of it.

($R \leq 2$ in our experiments) neighbors, and GSage is also always shallow (so L is small) (Zhao & Akoglu, 2019), the overall extra complexity introduced by NODEDUP is $O(|\mathcal{V}_{cold}|D^2 + |\mathcal{V}_{cold}|D)$.

3.2 How does node duplication help cold-start LP?

In this subsection, we analyze how such a simple method can improve cold-start LP from two perspectives: the neighborhood **aggregation** in GNNs and the **supervision** signal during training. In short, NODEDUP leverages the extra information from an additional "view". The existing view is when a node is regarded as the anchor node during message passing, whereas the additional view is when that node is regarded as one of its neighbors thanks to the duplicated node from NODEDUP.

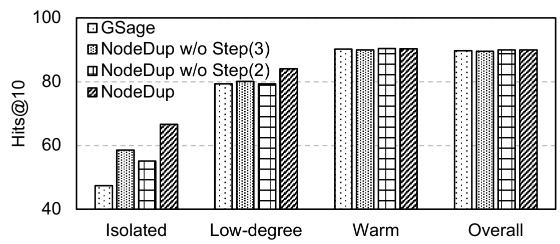


Figure 2: Ablation study of NODEDUP on `Physics`. Step (2) and Step (3) are the steps introduced in Section 3.1. Both steps play an important role in performance improvements of NODEDUP.

Aggregation. As described in Equation (1), when UPDATE$(\cdot)$ and AGG$(\cdot)$ do not share the transformation for node features, GNNs would have separate weights for self-representation and neighbor representations. The separate weights enable the neighbors and the node itself to play distinct roles in the UPDATE step. By leveraging this property, with NODEDUP, the model can leverage the two "views" for each node: first, the existing view is when a node is regarded as the anchor node during message passing, and the additional view is when that node is regarded as one of its neighbors thanks to the duplicated node from NODEDUP. Taking the official PyG (Fey & Lenssen, 2019) implementation of GSage (Hamilton et al., 2017) as an example, it updates node representations using $\boldsymbol{h}_v^{(l+1)} = \mathrm{W}_1\boldsymbol{h}_v^{(l)} + \mathrm{W}_2\boldsymbol{m}_v^{(l)}$. Here, W_1 and W_2 correspond to the self-representation and neighbors' representations, respectively. Without NODEDUP, isolated nodes $\mathcal{V}_{iso}$ have no neighbors, which results with $\boldsymbol{m}_v^{(l)} = \mathbf{0}$. Thus, the representations of all $v \in \mathcal{V}_{iso}$ are only updated by $\boldsymbol{h}_v^{(l+1)} = \mathrm{W}_1\boldsymbol{h}_v^{(l)}$. With NODEDUP, the updating process for isolated node v becomes $\boldsymbol{h}_v^{(l+1)} = \mathrm{W}_1\boldsymbol{h}_v^{(l)} + \mathrm{W}_2\boldsymbol{h}_v^{(l)} = (\mathrm{W}_1 + \mathrm{W}_2)\boldsymbol{h}_v^{(l)}$. It indicates that W_2 is also incorporated into the node updating process for isolated nodes, which offers an additional perspective for isolated nodes' representation learning. Similarly, GAT (Veličković et al., 2017) updates node representations with $\boldsymbol{h}_v^{(l+1)} = \alpha_{vv}\Theta\boldsymbol{h}_v^{(l)} + \sum_{u\in\mathcal{N}_v}\alpha_{vu}\Theta\boldsymbol{h}_u^{(l)}$, where $\alpha_{vu} = \frac{\exp(\mathrm{LeakyReLU}(\boldsymbol{a}^\top[\Theta\boldsymbol{h}_v^{(l)}||\Theta\boldsymbol{h}_u^{(l)}]))}{\sum_{i\in\mathcal{N}_v\cup v}\exp(\mathrm{LeakyReLU}(\boldsymbol{a}^\top[\Theta\boldsymbol{h}_v^{(l)}||\Theta\boldsymbol{h}_i^{(l)}]))}$. Attention scores in $\boldsymbol{a}$ partially correspond to the self-representation $\boldsymbol{h}_v$ and partially to neighbors' representation $\boldsymbol{h}_u$. In this case, neighbor information offers a different perspective compared to self-representation. Such "multi-view" enriches the representations learned for the isolated nodes in a similar way to how ensemble methods work (Allen-Zhu & Li, 2020). Apart from addressing isolated nodes, the same mechanism and multi-view perspective also apply to Low-degree nodes.

Supervision. For LP tasks, besides the aggregation, edges also serve as supervised training signals. Cold nodes have few or no positive training edges connecting to them, potentially leading to out-of-distribution (OOD) issues (Wu et al., 2022), especially for isolated nodes. The additional edges, added by NodeDup to connect cold nodes with their duplicates, serve as additional positive supervision signals for LP. More supervision signals for cold nodes usually lead to better-quality embeddings.

Ablation Study. Figure 2 shows an ablation study on these two designs where NodeDup w/o Step (3) indicates only using the augmented nodes and edges in aggregation but not supervision; NodeDup w/o Step (2) indicates only using the augmented edges in supervision but not aggregation. We can observe that using augmented nodes/edges either in supervision or aggregation can significantly improve the LP performance on Isolated nodes, and NodeDup, by combining them, results in larger improvements. Besides, NodeDup also achieves improvements on Low-degree nodes while not sacrificing Warm nodes' performance.

LP on Warm Nodes. The superior performance on warm nodes is directly tied to our focus on link prediction tasks. Given the substantial number of Warm-Cold node pairs under prediction, these outcomes contribute to the overall performance metrics for both Warm node prediction. Better learning of Cold nodes thus boosts Cold-Warm node pairs link prediction performance, which subsequently elevates the prediction accuracy for warm nodes. A more detailed experimental analysis is provided in Appendix D.2.

3.3 Relation between NodeDup and Self-distillation

Recently, Allen-Zhu & Li (2020) showed that the success of self-distillation, similar to our method, contributes to ensemble learning by providing models with different perspectives on the knowledge. Building on this insight, we show an interesting interpretation of NodeDup, as a simplified and enhanced version of self-distillation for LP tasks for cold nodes, illustrated in Figure 3, in which we draw a connection between self-distillation and NodeDup.

In **self-distillation**, a teacher GNN is first trained to learn the node representations $\mathbf{H}^t$ from original features $\mathbf{X}$. We denote the original graph as G^o, and we denote the graph, where we replace the node features in G^o with $\mathbf{H}^t$, as G^t in Figure 3. The student GNN is then initialized with random parameters and trained with the sum of two loss functions: $\mathcal{L}_{SD} = \mathcal{L}_{SP} + \mathcal{L}_{KD}$, where $\mathcal{L}_{SP}$ denotes the supervised training loss with G^o and $\mathcal{L}_{KD}$ denotes the knowledge distillation loss with G^t. Figure 4 shows that self-distillation outperforms the teacher GNN across all settings.

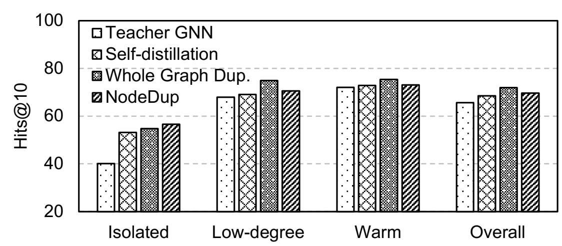


Figure 4: Performance with different training strategies introduced in Figure 3 on `Citeseer`. NodeDup achieves better performance across all settings.

The effect of $\mathcal{L}_{KD}$ is similar to that of creating an additional link connecting nodes in G^o to their corresponding nodes in G^t when optimizing with $\mathcal{L}_{SP}$. This is illustrated by the red dashed line in Figure 3. For better clarity, we show the similarities between these two when we use the inner product as the decoder for LP with the following example. Given a node v with normalized teacher embedding $\boldsymbol{h}_v^t$ and normalized student embedding $\boldsymbol{h}_v$, the additional loss term that would be added for distillation with cosine similarity is $\mathcal{L}_{KD} = -\frac{1}{N}\sum_{v \in \mathcal{V}} \boldsymbol{h}_v \cdot \boldsymbol{h}_v^t$. On the other hand, for the dashed line edges in Figure 3, we add an edge between the node v and its corresponding node v' in G^t with embedding $\boldsymbol{h}_{v'}^t$. When trained with an inner product decoder and binary cross-entropy loss, it results in the following: $\mathcal{L}_{SP} = -\frac{1}{N}\sum y_{vv'} \log(\boldsymbol{h}_v \cdot \boldsymbol{h}_{v'}^t) + (1 - y_{vv'}) \log(1 - \boldsymbol{h}_v \cdot \boldsymbol{h}_{v'}^t)$. Since we always add the edge (v, v'), we know $y_{vv'} = 1$, and can simplify the loss as follows: $\mathcal{L}_{SP} = -\frac{1}{N}\sum \log(\boldsymbol{h}_v \cdot \boldsymbol{h}_{v'}^t)$. Here, we can observe that $\mathcal{L}_{KD}$ and $\mathcal{L}_{SP}$ are positively correlated as $\log(\cdot)$ is a monotonically increasing function.

To further improve this step and mitigate potential noise in G^t, we explore a whole graph duplication technique, where G^t is replaced with an exact duplicate of G^o to train the student GNN. The results in Figure 4 demonstrate significant performance enhancement achieved by **whole graph duplication** compared to self-distillation. NodeDup is a lightweight variation of the whole graph duplication technique, which focuses on duplicating only the cold nodes and adding edges connecting them to their duplicates. From the results, it is evident that **NodeDup** consistently outperforms the teacher GNN and self-distillation in all scenarios. Additionally, NodeDup exhibits superior performance on isolated nodes and is much more efficient compared to the whole graph duplication approach.

3.4 NODEDUP(L): AN EFFICIENT VARIANT OF NODEDUP

Inspired by the above analysis, we further introduce a lightweight variant of NODEDUP for better efficiency, NODEDUP(L). To provide above-described "multi-view" information as well as the supervision signals for cold nodes, NODEDUP(L) simply add additional self-loop edges for the cold nodes into the edge set $\mathcal{E}$, that is, $\mathcal{E}' = \mathcal{E} \cup \{e_{vv} : \forall v \in \mathcal{V}_{cold}\}$. NODEDUP(L) preserves the two essential designs of NODEDUP while avoiding the addition of extra nodes, which further saves time and space complexity. Moreover, NODEDUP differs from NODEDUP(L) since each duplicated node in NODEDUP will provide another view for itself because of dropout layers, which leads to different performance as shown in Section 4.2.

NODEDUP(L) vs. Self-loop. We remark upon a resemblance between NODEDUP(L) and self-loops in GNNs (e.g., the additional self-connection in the normalized adjacency matrix by GCN) as they both add self-loop edges. However, they differ in two aspects. During **aggregation**: NODEDUP(L) intentionally incorporates the self-representation $\boldsymbol{h}_v^{(l)}$ into the aggregated neighbors' representation $\boldsymbol{m}_v^{(l)}$ by adding additional edges. Taking GSage as an example, the weight matrix W_2 would serve an extra "view" of $\boldsymbol{h}_v^{(l)}$ when updating $\boldsymbol{h}_v^{(l+1)}$, whereas the default self-loops only use information from W_1. Additionally, in the **supervision** signal: unlike the normal self-loops and the self-loops introduced in previous works (Cai et al., 2019; Wang et al., 2020), where self-loops are solely for aggregation, the edges added by NODEDUP(L) also serve as positive training samples for cold nodes.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Datasets and Evaluation Settings. We conduct experiments on 7 benchmark datasets: `Cora`, `Citeseer`, `CS`, `Physics`, `Computers`, `Photos` and `IGB-100K`, with their details specified in Appendix B. We randomly split edges into training, validation, and testing sets. We allocated 10% for validation and 40% for testing in `Computers` and `Photos`, 5%/10% for testing in `IGB-100K`, and 10%/20% in other datasets. We follow the standard evaluation metrics used in the Open Graph Benchmark (Hu et al., 2020) for LP, in which we rank missing references higher than 500 negative reference candidates for each node. The negative references are randomly sampled from nodes not connected to the source node. We use Hits@10 as the main evaluation metric (Han et al., 2022) and also report MRR performance in Appendix D. We follow Guo et al. (2022) and Shiao et al. (2022) for the inductive settings, where new nodes appear after the training process. Additionally, results for large-scale datasets and heterophilic graphs are presented in Appendix D.4 and Appendix D.5.

Baselines. Both NODEDUP and NODEDUP(L) are flexible to integrate with different GNN encoder architectures and LP decoders. For our experiments, we use GSage (Hamilton et al., 2017) encoder and the inner product decoder as the default base LP model. To comprehensively evaluate our work, we compare NODEDUP against three categories of baselines. (1) Base LP models. (2) Cold-start methods: TailGNN (Liu et al., 2021) and Cold-brew (Zheng et al., 2021) primarily aim to enhance the performance on cold nodes. We also compared with Imbalance (Lin et al., 2017), viewing cold nodes as an issue of the imbalance concerning node degrees. (3) Graph data augmentation methods: Augmentation frameworks including DropEdge (Rong et al., 2019), TuneUP (Hu et al., 2022), and LAGNN (Liu et al., 2022b) typically improve the performance while introducing additional preprocessing or training time. Performance comparisons with heuristic methods are in Appendix D.6.

4.2 PERFORMANCE COMPARED TO BASE GNN LP MODELS

Isolated and Low-degree Nodes. We compare our methods with base GNN LP models that consist of a GNN encoder in conjunction with an inner product decoder and are trained with a supervised loss. From Table 1, we observe consistent improvements for both NODEDUP(L) and NODEDUP over the base GSage model across all datasets, particularly in the Isolated and Low-degree node settings. Notably, in the Isolated setting, NODEDUP achieves an impressive 29.6% improvement, on average, across all datasets. These findings provide clear evidence that our methods effectively address the issue of sub-optimal LP performance on cold nodes.

Warm Nodes and Overall. It is encouraging to see that NODEDUP(L) consistently outperforms GSage across all the datasets in the Warm nodes and Overall settings. NODEDUP also outperforms GSage in 13 out of 14 cases under both settings. These findings support the notion that our methods can effectively maintain and enhance the performance of Warm nodes.

Table 1: Performance compared with base GNN model and baselines for cold-start methods(evaluated by Hits@10). The best result is **bold**, and the runner-up is underlined. NodeDup and NodeDup(L) outperform GSage and cold-start baselines almost all the cases.

		GSage	Imbalance	TailGNN	Cold-brew	NodeDup(L)	NodeDup
Cora	Isolated	32.20±3.58	34.51±1.11	36.95±1.34	28.17±0.67	39.76±1.32	**44.27**±3.82
	Low-degree	59.45±1.09	59.42±1.21	61.35±0.79	57.27±0.63	**62.53**±1.03	61.98±1.14
	Warm	61.14±0.78	59.54±0.46	60.61±0.90	56.28±0.81	**62.07**±0.37	59.07±0.68
	Overall	58.31±0.68	57.55±0.67	59.02±0.71	54.44±0.53	**60.49**±0.49	58.92±0.82
Citeseer	Isolated	47.13±2.43	46.26±0.86	37.84±3.36	37.78±4.23	52.46±1.16	**57.54**±1.04
	Low-degree	61.88±0.79	61.90±0.60	62.06±1.73	59.12±9.97	73.71±1.22	**75.50**±0.39
	Warm	71.45±0.52	71.54±0.86	71.32±1.83	65.12±7.82	**74.99**±0.37	74.68±0.67
	Overall	63.77±0.83	63.66±0.43	62.02±1.89	58.03±7.72	70.34±0.35	**71.73**±0.47
CS	Isolated	56.41±1.61	46.60±1.66	55.70±1.38	57.70±0.81	65.18±1.25	**65.87**±1.70
	Low-degree	75.95±0.25	75.53±0.21	73.60±0.70	73.99±0.34	**81.46**±0.57	81.12±0.36
	Warm	84.37±0.46	83.70±0.46	79.86±0.35	78.23±0.28	**85.48**±0.26	84.76±0.41
	Overall	83.33±0.42	82.56±0.40	79.05±0.36	77.63±0.23	**84.90**±0.29	84.23±0.39
Physics	Isolated	47.41±1.38	55.01±0.58	52.54±1.34	64.38±0.85	65.04±0.63	**66.65**±0.95
	Low-degree	79.31±0.28	79.50±0.27	75.95±0.27	75.86±0.10	82.70±0.22	**84.04**±0.22
	Warm	90.28±0.23	89.85±0.09	85.93±0.40	78.48±0.14	**90.44**±0.23	90.33±0.05
	Overall	89.76±0.22	89.38±0.09	85.48±0.38	78.34±0.13	**90.09**±0.22	90.03±0.05
Computers	Isolated	9.32±1.44	10.14±0.59	10.63±1.59	9.75±1.24	17.11±1.62	**19.62**±2.63
	Low-degree	57.91±0.97	56.19±0.82	51.21±1.58	49.03±0.94	**62.14**±1.06	61.16±0.92
	Warm	66.87±0.47	65.62±0.21	62.77±0.44	57.52±0.28	68.02±0.41	**68.10**±0.25
	Overall	66.67±0.47	65.42±0.20	62.55±0.45	57.35±0.28	67.86±0.41	**67.94**±0.25
Photos	Isolated	9.25±2.31	10.80±1.72	13.62±1.00	12.86±2.58	**21.50**±2.14	17.84±3.53
	Low-degree	52.61±0.88	50.68±0.57	42.75±2.50	43.14±0.64	**55.70**±1.38	54.13±1.58
	Warm	67.64±0.55	64.54±0.50	61.63±0.73	58.06±0.56	**69.68**±0.87	68.68±0.49
	Overall	67.32±0.54	64.24±0.49	61.29±0.75	57.77±0.56	**69.40**±0.86	68.39±0.48
IGB-100K	Isolated	75.92±0.52	77.32±0.79	77.29±0.34	82.31±0.30	87.43±0.44	**88.04**±0.20
	Low-degree	79.38±0.23	79.19±0.09	80.57±0.14	83.84±0.16	88.37±0.24	**88.98**±0.17
	Warm	86.42±0.24	86.01±0.19	85.35±0.19	82.44±0.21	**88.54**±0.31	88.28±0.20
	Overall	84.77±0.21	84.47±0.14	84.19±0.18	82.68±0.17	**88.47**±0.28	88.39±0.18

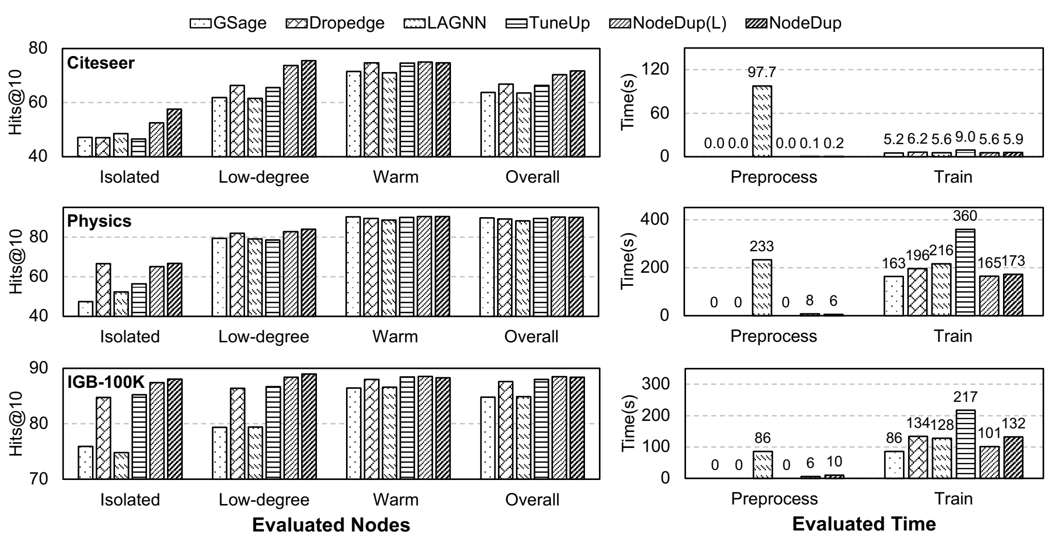


Figure 5: Performance and runtime comparisons of different augmentation methods. The *left* histograms show the performance results, and the *right* histograms show the preprocessing and training time consumption of each method. Our methods consistently achieve significant improvements in both performance for Isolated and Low-degree node settings and runtime efficiency over baselines.

NodeDup vs. NodeDup(L). Furthermore, we observe that NodeDup achieves greater improvements over NodeDup(L) for Isolated nodes. However, NodeDup(L) outperforms NodeDup on 6 out of 7 datasets for Warm nodes. The additional improvements achieved by NodeDup for Isolated nodes can be attributed to the extra view provided to cold nodes through node duplication during aggregation. On the other hand, the impact of node duplication on the original graph structure likely affects the performance of Warm nodes, which explains the superior performance of NodeDup(L) in this setting compared to NodeDup.

4.3 Performance Compare to Cold-start Methods

Table 1 presents the LP performance of various cold-start baselines. For both Isolated and Low-degree nodes, we consistently observe *substantial improvements* of our NodeDup and NodeDup(L) methods compared to other cold-start baselines. Specifically, NodeDup and NodeDup(L) achieve 38.49% and 34.74% improvement for Isolated nodes on average across all datasets, respectively.

In addition, our methods consistently outperform cold-start baselines for Warm nodes across all the datasets, where NodeDup(L) and NodeDup achieve 6.76% and 7.95% improvements on average, respectively. This shows that our methods can successfully overcome issues with degrading performance on Warm nodes in cold-start baselines. Further analyses with other cold-start methods and efficiency comparisons can be found in Appendix D.8 and Appendix D.9.

4.4 Performance Compared to Augmentation Methods

Effectiveness Comparison. Since NodeDup and NodeDup(L) use graph data augmentation techniques, we compare them to other data augmentation baselines. The performance and time consumption results are presented in Figure 5 for three datasets (`Citeseer`, `Physics`, and `IGB-100K`), while the results for the remaining datasets are provided in Appendix D.10 due to the page limit. From Figure 5, we consistently observe that NodeDup outperforms all the graph augmentation baselines for Isolated and Low-degree nodes across all three datasets. Similarly, NodeDup(L) outperforms graph data augmentation baselines on 17/18 cases for Isolated and Low-degree nodes. Not only did our methods perform better for Isolated and Low-degree nodes, NodeDup and NodeDup(L) also perform on par or above baselines for Warm nodes.

Efficiency Comparison. Augmentation methods often come with the trade-off of adding additional run time before or during model training. For example, LAGNN (Liu et al., 2022b) requires extra preprocessing time to train the generative model prior to GNN training. It also takes additional time to generate extra features for each node during training. Although Dropedge (Rong et al., 2019) and TuneUP (Hu et al., 2022) are free of preprocessing, they require additional time to drop edges in each training epoch compared to base GNN training. Furthermore, the two-stage training employed by TuneUP doubles the training time compared to one-stage training methods. For NodeDup methods, duplicating nodes and adding edges is remarkably swift and consumes significantly less preprocessing time than other augmentation methods. As an example, NodeDup(L) and NodeDup are **977.0**× and **488.5**× faster than LAGNN in preprocessing `Citeseer`, respectively. We also observe that NodeDup(L) has the least training time among all augmentation methods and datasets, while NodeDup also requires less training time in 8/9 cases. Additionally, NodeDup(L) achieves significant efficiency benefits compared to NodeDup in Figure 5, especially when the number of nodes in the graph increases substantially. Taking the `IGB-100K` dataset as an example, NodeDup(L) is 1.3× faster than NodeDup for the entire training process.

4.5 Performance under the Inductive Setting

Under the inductive setting (Guo et al., 2022; Shiao et al., 2022), which closely resembles real-world LP scenarios, the presence of new nodes after the training stage adds an additional challenge compared to the transductive setting. We evaluate and present the effectiveness of our methods under this setting in Table 2 for `Citeseer`, `Physics`, and `IGB-100K` datasets. Additional results for other datasets can be found in Appendix D.11. In Table 2, we observe that our methods consistently outperform base GSage across all of the datasets. We also observe significant performance im-

Table 2: Performance in inductive settings (evaluated by Hits@10). The best result is **bold**, and the runner-up is <u>underlined</u>. Our methods consistently outperform GSage.

		GSage	NodeDup(L)	NodeDup
`Citeseer`	Isolated	58.42±0.49	<u>62.42</u>±1.88	**62.94**±1.91
	Low-degree	67.75±1.06	<u>69.93</u>±1.18	**72.05**±1.23
	Warm	72.98±1.15	**75.04**±1.03	<u>74.40</u>±2.43
	Overall	66.98±0.61	<u>69.65</u>±0.83	**70.26**±1.16
`Physics`	Isolated	85.62±0.23	<u>85.94</u>±0.15	**86.90**±0.35
	Low-degree	80.87±0.43	<u>81.23</u>±0.56	**85.56**±0.25
	Warm	90.22±0.36	<u>90.37</u>±0.25	**90.54**±0.14
	Overall	89.40±0.33	<u>89.57</u>±0.23	**89.98**±0.13
`IGB-100K`	Isolated	84.33±0.87	<u>92.94</u>±0.11	**93.95**±0.06
	Low-degree	93.19±0.06	<u>93.33</u>±0.11	**94.00**±0.09
	Warm	90.76±0.13	**91.21**±0.07	<u>91.20</u>±0.08
	Overall	90.31±0.18	<u>91.92</u>±0.05	**92.21**±0.04

Table 3: Performance with different encoders (inner product as the decoder). The best result for each encoder is **bold**, and the runner-up is underlined. Our methods consistently outperform the base models, particularly for Isolated and Low-degree nodes.

		GAT	NodeDup(L)	NodeDup	JKNet	NodeDup(L)	NodeDup
Citeseer	Isolated	37.78±2.36	38.95±2.75	**44.04**±1.03	37.78±0.63	49.06±0.60	**55.15**±0.87
	Low-degree	58.04±2.40	61.93±1.66	**66.73**±0.96	60.74±1.18	71.78±0.64	**75.26**±1.16
	Warm	56.37±2.15	64.55±1.74	**66.61**±1.67	71.61±0.76	74.66±0.47	**75.81**±0.89
	Overall	53.42±1.59	58.89±0.89	**62.41**±0.78	61.73±0.57	68.91±0.38	**71.75**±0.82
Physics	Isolated	38.19±1.23	39.95±1.48	**45.89**±2.82	42.57±1.93	55.47±2.25	**61.11**±2.27
	Low-degree	74.19±0.31	74.77±0.46	**76.36**±0.25	75.36±0.23	79.55±0.21	**81.14**±0.28
	Warm	85.84±0.32	**86.02**±0.45	85.84±0.15	88.24±0.32	**89.42**±0.16	89.24±0.16
	Overall	85.27±0.30	**85.47**±0.45	85.37±0.14	87.64±0.31	**88.96**±0.15	88.87±0.15
IGB-100K	Isolated	75.87±0.48	78.17±0.58	**80.18**±0.31	69.29±0.73	86.60±0.46	**86.85**±0.41
	Low-degree	77.05±0.15	78.50±0.31	**81.00**±0.12	76.90±0.27	86.94±0.15	**87.65**±0.20
	Warm	81.40±0.07	**81.95**±0.25	81.19±0.20	84.93±0.30	**87.41**±0.13	86.19±0.12
	Overall	80.42±0.07	**81.19**±0.25	81.11±0.19	82.91±0.28	**87.29**±0.13	86.47±0.13

provements of our methods on Isolated nodes, where NodeDup and NodeDup(L) achieve 5.50% and 3.57% improvements averaged across the three datasets, respectively. Additionally, NodeDup achieves 5.09% improvements on Low-degree nodes. NodeDup leads to more pronounced improvements on Low-degree/Isolated nodes, making it particularly beneficial for the inductive setting.

4.6 Performance with Different Encoders/Decoders

As a simple plug-and-play augmentation method, NodeDup can work with different GNN encoders and LP decoders. In Tables 3 and 4, we present results with GAT (Veličković et al., 2017) and JKNet (Xu et al., 2018) as encoders, along with a MLP decoder. Due to the space limit, we only report the results of three datasets here and leave the remaining in Appendix D.12. When applying NodeDup to base LP training, with GAT or JKNet as the encoder and inner product as the decoder, we observe significant performance improvements across the board. Regardless of the encoder choice, NodeDup consistently outperforms the base models, particularly for Isolated and Low-degree nodes. From Appendix D.12, we also observe the performance improvements of NodeDup with GCN (Kipf & Welling, 2016a), GraphTransformer (Dwivedi & Bresson, 2020) as encoders.

Table 4: LP performance with MLP decoder (GSage as the encoder). Our methods outperform the base model.

		MLP-Dec.	NodeDup(L)	NodeDup
Citeseer	Isolated	17.16±1.14	37.84±3.06	**51.17**±2.19
	Low-degree	63.82±1.58	68.49±1.19	**71.98**±1.29
	Warm	72.93±1.25	75.33±0.54	**75.72**±0.55
	Overall	59.49±1.21	66.07±0.74	**69.89**±0.65
Physics	Isolated	11.59±1.88	**60.25**±2.54	59.50±1.87
	Low-degree	76.37±0.64	81.74±0.77	**82.58**±0.79
	Warm	91.54±0.33	**91.96**±0.36	91.59±0.22
	Overall	90.78±0.33	**91.51**±0.38	91.13±0.23
IGB-100K	Isolated	3.51±0.32	**82.71**±1.05	82.02±0.73
	Low-degree	75.25±0.49	85.96±0.42	**86.04**±0.26
	Warm	85.06±0.08	**87.89**±0.13	86.87±0.48
	Overall	80.16±0.16	**87.35**±0.21	86.54±0.40

In Table 4, we present the results of our methods applied to the base LP training, where GSage serves as the encoder and MLP as the decoder. Regardless of the decoder, we observe better performance with our methods. These improvements are significantly higher compared to the improvements observed with the inner product decoder. The primary reason for this discrepancy is the inclusion of additional supervised training signals for isolated nodes in our methods, as discussed in Section 3.2. These signals play a crucial role in training the MLP decoder, making it more responsive to the specific challenges presented by isolated nodes. Our methods also improve performance with SEAL (Zhang & Chen, 2018), as shown in Appendix D.12.

5 Conclusion

GNNs in LP encounter difficulties when dealing with cold nodes that lack sufficient or absent neighbors. To address this challenge, we presented a simple yet effective augmentation method (NodeDup) specifically tailored for the cold-start LP problem, which can effectively enhance the prediction capabilities of GNNs for cold nodes while maintaining overall performance. Extensive evaluations demonstrated that both NodeDup and its lightweight variant, NodeDup(L), consistently outperformed baselines on both cold node and warm node settings across 7 benchmark datasets. NodeDup also achieved better runtime efficiency compared to the augmentation baselines.

Ethics Statement. In this work, our simple but effective method enhances the link prediction performance on cold-start nodes, which mitigates the degree bias and advances the fairness of graph machine learning. It can be widely used and beneficial for various real-world applications, such as recommendation systems, social network analysis, and bioinformatics. 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.

REFERENCES

Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

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

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.

Ling Cai, Bo Yan, Gengchen Mai, Krzysztof Janowicz, and Rui Zhu. Transgcn: Coupling transformation assumptions with graph convolutional networks for link prediction. In *Proceedings of the 10th international conference on knowledge capture*, pp. 131–138, 2019.

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

Zhihong Chen, Rong Xiao, Chenliang Li, Gangfeng Ye, Haochuan Sun, and Hongbo Deng. Esam: Discriminative domain adaptation with non-displayed items to improve long-tail performance. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 579–588, 2020.

Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. Node feature extraction by self-supervised multi-scale neighborhood prediction. *arXiv preprint arXiv:2111.00064*, 2021.

Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 2009.

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

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

Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 2022.

Kaiwen Dong, Yijun Tian, Zhichun Guo, Yang Yang, and Nitesh Chawla. Fakeedge: Alleviate dataset shift in link prediction. In *Learning on Graphs Conference*, pp. 56–1. PMLR, 2022.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. Graph trend filtering networks for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020.

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

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*. PMLR, 2017.

Zhichun Guo, William Shiao, Shichang Zhang, Yozen Liu, Nitesh Chawla, Neil Shah, and Tong Zhao. Linkless link prediction via relational distillation. *arXiv preprint arXiv:2210.05801*, 2022.

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

Xiaotian Han, Tong Zhao, Yozen Liu, Xia Hu, and Neil Shah. Mlpinit: Embarrassingly simple gnn training acceleration with mlp initialization. *arXiv preprint arXiv:2210.00102*, 2022.

Bowen Hao, Jing Zhang, Hongzhi Yin, Cuiping Li, and Hong Chen. Pre-training graph neural networks for cold-start users and items representation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021.

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*, 2020.

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.

Weihua Hu, Kaidi Cao, Kexin Huang, Edward W Huang, Karthik Subbian, and Jure Leskovec. Tuneup: A training strategy for improving generalization of graph neural networks. *arXiv preprint arXiv:2210.14843*, 2022.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019.

Arpandeep Khatua, Vikram Sharma Mailthody, Bhagyashree Taleka, Tengfei Ma, Xiang Song, and Wen-mei Hwu. Igb: Addressing the gaps in labeling, features, heterogeneity, and size of public graph datasets for deep learning research, 2023. URL https://arxiv.org/abs/2302.13522.

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.

István A Kovács, Katja Luck, Kerstin Spirohn, Yang Wang, Carl Pollis, Sadie Schlabach, Wenting Bian, Dae-Kyum Kim, Nishka Kishore, Tong Hao, et al. Network-based prediction of protein interactions. *Nature communications*, 2019.

Juanhui Li, Harry Shomer, Jiayuan Ding, Yiqi Wang, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Are message passing neural networks really helpful for knowledge graph completion? *ACL*, 2023.

Jie Liao, Jintang Li, Liang Chen, Bingzhe Wu, Yatao Bian, and Zibin Zheng. Sailor: Structural augmentation based tail node representation learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 1389–1399, 2023.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 2007.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Gang Liu, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Graph rationalization with environment-based augmentations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1069–1078, 2022a.

Siyi Liu and Yujia Zheng. Long-tail session-based recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pp. 509–514, 2020.

Songtao Liu, Rex Ying, Hanze Dong, Lanqing Li, Tingyang Xu, Yu Rong, Peilin Zhao, Junzhou Huang, and Dinghao Wu. Local augmentation for graph neural networks. In *International Conference on Machine Learning*, pp. 14054–14072. PMLR, 2022b.

Zemin Liu, Wentao Zhang, Yuan Fang, Xinming Zhang, and Steven CH Hoi. Towards locality-aware meta-learning of tail node embeddings on networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 975–984, 2020.

Zemin Liu, Trung-Kien Nguyen, and Yuan Fang. Tail-gnn: Tail-node graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

Zemin Liu, Trung-Kien Nguyen, and Yuan Fang. On generalized degree fairness in graph neural networks. *arXiv preprint arXiv:2302.03881*, 2023.

Yuanfu Lu, Yuan Fang, and Chuan Shi. Meta-learning on heterogeneous information networks for cold-start recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Youzhi Luo, Michael McThrow, Wing Yee Au, Tao Komikado, Kanji Uchino, Koji Maruhash, and Shuiwang Ji. Automated data augmentations for graph classification. *arXiv preprint arXiv:2202.13248*, 2022.

Zihan Luo, Hong Huang, Jianxun Lian, Xiran Song, Xing Xie, and Hai Jin. Cross-links matter for link prediction: rethinking the debiased gnn from a data perspective. *Advances in Neural Information Processing Systems*, 36, 2024.

Hyeonjin Park, Seunghun Lee, Sihyeon Kim, Jinyoung Park, Jisu Jeong, Kyung-Min Kim, Jung-Woo Ha, and Hyunwoo J Kim. Metropolis-hastings data augmentation for graph neural networks. *Advances in Neural Information Processing Systems*, 34:19010–19020, 2021.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Foster Provost. Machine learning from imbalanced data sets 101. 2000.

Jiawei Ren, Cunjun Yu, Xiao Ma, Haiyu Zhao, Shuai Yi, et al. Balanced meta-softmax for long-tailed visual recognition. *Advances in neural information processing systems*, 33:4175–4186, 2020.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

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 2021*, 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.

William Shiao, Zhichun Guo, Tong Zhao, Evangelos E Papalexakis, Yozen Liu, and Neil Shah. Link prediction with non-contrastive learning. *arXiv preprint arXiv:2211.14394*, 2022.

Zachary Stanfield, Mustafa Coşkun, and Mehmet Koyutürk. Drug response prediction as a link prediction problem. *Scientific reports*, 2017.

Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11662–11671, 2020.

Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *Advances in Neural Information Processing Systems*, 33:1513–1524, 2020a.

Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. Investigating and mitigating degree-related biases in graph convoltuional networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020b.

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*, 2022.

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

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2020.

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

Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. Dropoutnet: Addressing cold start in recommender systems. *Advances in neural information processing systems*, 30, 2017.

Zhitao Wang, Yu Lei, and Wenjie Li. Neighborhood attention networks with adversarial learning for link prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3653–3663, 2020.

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

Jun Wu, Jingrui He, and Jiejun Xu. Net: Degree-specific graph neural networks for node and graph classification. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 406–415, 2019.

Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. *arXiv preprint arXiv:2202.02466*, 2022.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

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.

Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, and Chao Chen. Link prediction with persistent homology: An interactive view. In *International conference on machine learning*, pp. 11659–11669. PMLR, 2021.

Chaoqi Yang, Cao Xiao, Fenglong Ma, Lucas Glass, and Jimeng Sun. Safedrug: Dual molecular graph encoders for recommending effective and safe drug combinations. In *IJCAI*, pp. 3735–3741, 2021.

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

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*, 2018.

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.

Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Zhenhui Li, and Nitesh V Chawla. Few-shot knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 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*, 2021.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence*, 2021.

Tong Zhao, Wei Jin, Yozen Liu, Yingheng Wang, Gang Liu, Stephan Günneman, Neil Shah, and Meng Jiang. Graph data augmentation for graph machine learning: A survey. *arXiv preprint arXiv:2202.08871*, 2022a.

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

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.

Yu Zhu, Jinghao Lin, Shibi He, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. Addressing the item cold-start problem by attribute-driven active learning. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

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.

CONTENTS OF APPENDIX

A RELATED WORK

LP with GNNs. Over the past few years, GNN architectures (Kipf & Welling, 2016a; Gilmer et al., 2017; Hamilton et al., 2017; Veličković et al., 2017; Xu et al., 2018) have gained significant attention and demonstrated promising outcomes in LP tasks. There are two primary approaches to applying GNNs in LP. The first approach involves a node-wise encoder-decoder framework, which we discussed in Section 2. The second approach reformulates LP tasks as enclosing subgraph classification tasks (Zhang & Chen, 2018; Cai & Ji, 2020; Cai et al., 2021; Dong et al., 2022). Instead of directly predicting links, these methods perform graph classification tasks on the enclosing subgraphs sampled around the target link. These methods can achieve even better results compared to node-wise encoder-decoder frameworks by assigning node labels to indicate different roles within the subgraphs. However, constructing subgraphs poses challenges in terms of efficiency and scalability, requiring substantial computational resources. Our work focuses on the encoder-decoder framework for LP, circumventing the issues associated with subgraph construction.

Methods for Cold-start Nodes. Recently, several GNN-based methods (Wu et al., 2019; Liu et al., 2020; Tang et al., 2020b; Liu et al., 2021; Zheng et al., 2021) have explored degree-specific transformations to address robustness and cold-start node issues. Tang et al.(Tang et al., 2020b) introduced a degree-related graph convolutional network to mitigate degree-related bias in node classification tasks. Liu et al.(Liu et al., 2021) proposed a transferable neighborhood translation model to address missing neighbors for cold-start nodes. Zheng et al.(Zheng et al., 2021) tackled the cold-start nodes problem by recovering missing latent neighbor information. These methods require cold-start-node-specific architectural components, unlike our approach, which does not necessitate any architectural modifications. Additionally, other studies have focused on long-tail scenarios in various domains, such as cold-start recommendation(Chen et al., 2020; Lu et al., 2020; Hao et al., 2021). Imbalance tasks present another common long-tail problem, where there are long-tail instances within small classes (Lin et al., 2017; Ren et al., 2020; Tan et al., 2020; Kang et al., 2019; Tang et al., 2020a). Approaches like (Lin et al., 2017; Ren et al., 2020; Tan et al., 2020) address this issue by adapting the loss for different samples. However, due to the different problem settings, it is challenging to directly apply these methods to our tasks. We only incorporate the balanced cross entropy introduced by Lin et al. (Lin et al., 2017) as one of our baselines.

Graph Data Augmentation. Graph data augmentation expands the original data by perturbing or modifying the graphs to enhance the generalizability of GNNs (Zhao et al., 2022a; Ding et al., 2022). Existing methods primarily focus on semi-supervised node-level tasks(Rong et al., 2019; Feng et al., 2020; Zhao et al., 2021; Park et al., 2021) and graph-level tasks (Liu et al., 2022a; Luo et al., 2022). However, the exploration of graph data augmentation for LP remains limited (Zhao et al., 2022b). CFLP (Zhao et al., 2022b) proposes the creation of counterfactual links to learn representations from both observed and counterfactual links. Nevertheless, this method encounters scalability issues due to the high computational complexity associated with finding counterfactual links. Moreover, there exist general graph data augmentation methods (Liu et al., 2022b; Hu et al., 2022) that can be applied to various tasks. LAGNN (Liu et al., 2022b) proposed to use a generative model to provide additional neighbor features for each node. TuneUP (Hu et al., 2022) designs a two-stage training strategy, which trains GNNs twice to make them perform well on both warm nodes and cold-start nodes. These augmentation methods come with the trade-off of introducing extra runtime either before or during the model training. Unlike TLC-GNN (Yan et al., 2021), which necessitates extracting topological features for each node pair, and GIANT (Chien et al., 2021), which requires pre-training of the text encoder to improve node features, our methods are more streamlined and less complex.

B ADDITIONAL DATASETS DETAILS

This section provides detailed information about the datasets used in our experiments. We consider various types of networks, including citation networks, collaboration networks, and co-purchase networks. The datasets we utilize are as follows:

- Citation Networks: `Cora` and `Citeseer` originally introduced by Yang et al. (2016), consist of citation networks where the nodes represent papers and the edges represent citations between papers. `IGB-100K` (Khatua et al., 2023) is a recently-released benchmark citation network with high-quality node features and a large dataset size.

Table 5: Detailed statistics of data splits under the transductive and inductive setting.

Datasets	Original Graph #Nodes	Original Graph #Edges	Testing Isolated #Nodes	Testing Isolated #Edges	Testing Low-degree #Nodes	Testing Low-degree #Edges	Testing Warm #Nodes	Testing Warm #Edges
Transductive Setting								
`Cora`	2,708	5,278	135	164	541	726	662	1,220
`Citeseer`	3,327	4,552	291	342	492	591	469	887
`CS`	18,333	163,788	309	409	1,855	2,687	10,785	29,660
`Physics`	34,493	495,924	275	397	2,062	3,188	25,730	95,599
`Computers`	13,752	491,722	218	367	830	1,996	11,887	194,325
`Photos`	7,650	238,162	127	213	516	1,178	6,595	93,873
`IGB-100K`	100,000	547,416	1,556	1,737	6,750	7,894	23,949	35,109
Inductive Setting								
`Cora`	2,708	5,278	149	198	305	351	333	505
`Citeseer`	3,327	4,552	239	265	272	302	239	339
`CS`	18,333	163,788	1,145	1,867	1,202	1,476	6,933	13,033
`Physics`	34,493	495,924	2,363	5,263	1,403	1,779	17,881	42,548
`Computers`	13,752	491,722	1,126	4,938	239	302	9,235	43,928
`Photos`	7,650	238,162	610	2,375	169	212	5,118	21,225
`IGB-100K`	100,000	547,416	5,507	9,708	8,706	13,815	24,903	41,217

- Collaboration Networks: `CS` and `Physics` are representative collaboration networks. In these networks, the nodes correspond to authors and the edges represent collaborations between authors.
- Co-purchase Networks: `Computers` and `Photos` are co-purchase networks, where the nodes represent products and the edges indicate the co-purchase relationship between two products.

Why there are no OGB (Hu et al., 2020) datasets applied? OGB benchmarks that come with node features, such as OGB-collab and OGB-citation2, lack a substantial number of isolated or low-degree nodes, which makes it challenging to yield convincing results for experiments focusing on the cold-start problem. This is primarily due to the split setting adopted by OGB, where the evaluation is centered around a set of the most recent papers with high degrees. Besides, considering these datasets have their fixed splitting settings based on time, it will lead to inconsistent problems to compared with the leaderboard results if we use our own splitting method to ensure we have a reasonable number of isolated/low-degree nodes. Given these constraints, we opted for another extensive benchmark dataset, IGB-100K (Khatua et al., 2023), to test and showcase the effectiveness of our methods on large-scale graphs. We further conducted the experiments on `IGB1M`, which are shown in Appendix D.4.

B.1 Transductive Setting

For the transductive setting, we randomly split the edges into training, validation, and testing sets based on the splitting ratio specified in Section 4.1. The nodes in training/validation/testing are all visible during the training process. However, the positive edges in validation/testing sets are masked out for training. After the split, we calculate the degrees of each node using the validation graph. The dataset statistics are shown in Table 5.

B.2 Inductive Setting

The inductive setting is considered a more realistic setting compared to the transductive setting, where new nodes appear after the training process. Following the inductive setting introduced in Guo et al. (2022) and Shiao et al. (2022), we perform node splitting to randomly sample 10% nodes from the original graph as the new nodes appear after the training process. The remaining nodes are considered observed nodes during the training. Next, we group the edges into three sets: observed-observed, observed-new, and new-new node pairs. We select 10% of observed-observed, 10% of observed-new, and 10% of new-new node pairs as the testing edges. We consider the remaining observed-new and new-new node pairs, along with an additional 10% of observed-observed node pairs, as the newly visible edges for the testing inference. The datasets statistics are shown in Table 5.

C NodeDup Algorithm

In this section, we provide a detailed description of our algorithm, which is outlined in Algorithm 1. Compared to the default training of GNNs for LP tasks, NodeDup incorporates additional augmentation steps, denoted as L1-L5 in Algorithm 1.

Algorithm 1: NodeDup.

Require: Graph $G = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, Supervision $\mathcal{Y}$, AGG, UPDATE, GNNs Layer L, DECODER, Supervised loss function $\mathcal{L}_{sup}$.

1: **# Augment the graph by duplicating cold-start nodes $\mathcal{V}_{cold}$.**
2: Identify cold node set $\mathcal{V}_{cold}$ based on the node degree.
3: Duplicate all cold nodes to generate the augmented node set $\mathcal{V}' = \mathcal{V} \cup \mathcal{V}_{cold}$, whose node feature matrix is then $\mathbf{X}' \in \mathbb{R}^{(N+|\mathcal{V}_{cold}|)\times F}$.
4: Add an edge between each cold node $v \in \mathcal{V}_{cold}$ and its duplication v', then get the augmented edge set $\mathcal{E}' = \mathcal{E} \cup \{e_{vv'} : \forall v \in \mathcal{V}_{cold}\}$.
5: Add the augmented edges into the training set and get $\mathcal{Y}' = \mathcal{Y} \cup \{y_{vv'} = 1 : \forall v \in \mathcal{V}_{cold}\}$.
6: **# End-to-end supervised training based on the augmented graph $G' = \{\mathcal{V}', \mathcal{E}', \mathbf{X}'\}$.**
7: **for** $l = 1$ to L **do**
8:     **for** v in $\mathcal{V}'$ **do**
9:         $\boldsymbol{h}_v'^{(l+1)} = \text{UPDATE}\Big(\boldsymbol{h}_v'^{(l)}, \text{AGG}\big(\{\boldsymbol{h}_u'^{(l)}\} : \forall e_{uv} \in \mathcal{E}'\big)\Big)$
10:     **end for**
11: **end for**
12: **for** (i, j) in $\mathcal{Y}'$ **do**
13:     $\hat{y}'_{ij} = \sigma\big(\text{DECODER}(\boldsymbol{h}'_i, \boldsymbol{h}'_j)\big)$
14: **end for**
15: Loss = $\sum_{(i,j)\in\mathcal{Y}'} \mathcal{L}_{sup}(\hat{y}'_{ij}, y_{ij})$

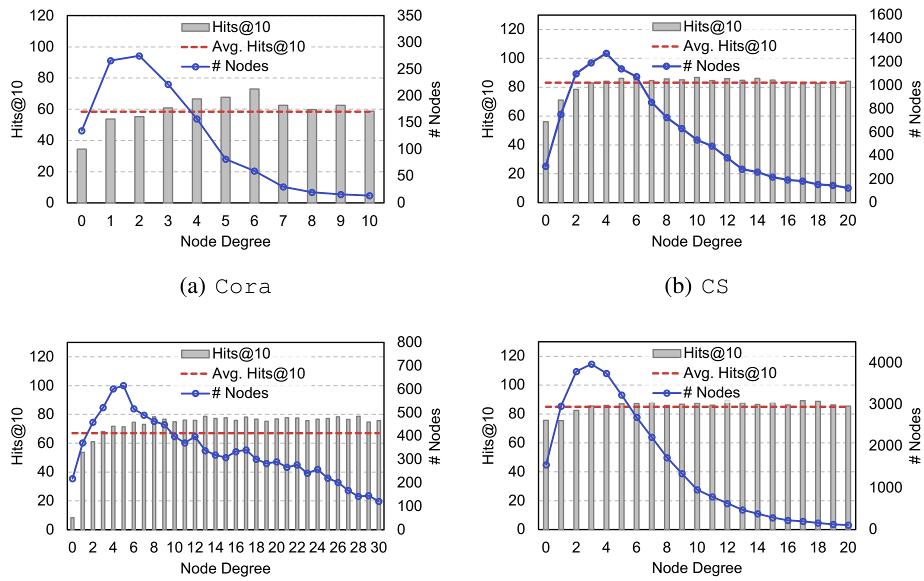
D Further Experimental Results

D.1 Selection of the threshold δ.

Our decision to set the threshold δ at 2 is grounded in data-driven analysis, as illustrated in Figure 1 and Figure 6. These figures reveal that nodes with degrees not exceeding 2 consistently perform below the average Hits@10 across all datasets, and higher than 2 will outperform the average results. Besides, our choice aligns with methodologies in previous studies (Liu et al., 2020; 2021), where cold nodes are identified using a fixed threshold across all the datasets. In addition, we conduct experiments with different thresholds δ on `Cora` and `Citeseer` datasets. The results are shown in Table 6. Our findings were consistent across different thresholds, with similar observations at δ = 1, δ = 2, and δ = 3. This indicates that our method's effectiveness is not significantly impacted by changes in this threshold.

D.2 Performance on Warm-Warm and Warm-Cold links.

To clearly explain the performance improvements of NodeDup on Warm nodes, we first compared the number of Warm-Warm and Warm-Cold links in the testing set. Then, we conducted experiments to compare the performance of our methods on these two sets of links. The results, shown in Table 7, indicate that the number of Warm-Warm links consistently exceeds that of Warm-Cold links across all datasets. This means that Warm-Cold links do not dominate the performance of Warm nodes. Additionally, our methods consistently improve performance on Warm-Cold links while maintaining performance on Warm-Warm links. These findings demonstrate that our methods do not negatively impact the learning ability of Warm nodes. The observed improvement on Warm nodes is primarily due to better learning on Cold nodes, as we demonstrated in Section 3.2.



(a) Cora (b) CS

(c) Computers (d) IGB-100K

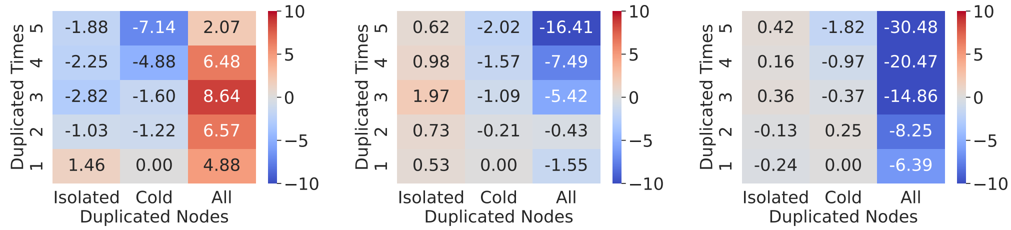
Figure 6: Node Degree Distribution and LP Performance Distribution w.r.t Nodes Degrees showing reverse trends on various datasets.

Table 6: Performance with different thresholds δ on Cora and Citeseer datasets.

		$\delta = 1$		$\delta = 2$		$\delta = 3$	
		Gsage	NodeDup	Gsage	NodeDup	Gsage	NodeDup
Cora	Isolated	31.34±5.60	42.20±2.30	32.20±3.58	44.27±3.82	31.95±1.26	43.17±2.94
	Low-degree	53.98±1.20	57.99±1.34	59.45±1.09	61.98±1.14	59.64±1.01	62.68±0.63
	Warm	61.68±0.29	61.17±0.43	61.14±0.78	59.07±0.68	61.03±0.79	59.91±0.44
	Overall	58.01±0.57	59.16±0.44	58.31±0.68	58.92±0.82	58.08±0.74	59.99±0.53
Citeseer	Isolated	47.25±1.82	56.49±1.72	47.13±2.43	57.54±1.04	47.31±2.17	56.90±1.12
	Low-degree	54.10±0.85	71.09±0.47	61.88±0.79	75.50±0.39	62.97±0.83	75.45±0.40
	Warm	72.41±0.35	74.57±1.04	71.45±0.52	74.68±0.67	73.57±0.46	75.02±0.84
	Overall	64.27±0.45	70.53±0.91	63.77±0.83	71.73±0.47	64.05±0.42	71.80±0.40

Table 7: Distribution and AUC performance of testing Warm-Warm and Warm-Cold links.

	Warm-Warm				Warm-Cold			
	Number	GSage	NodeDup(L)	NodeDup	Number	GSage	NodeDup(L)	NodeDup
Cora	157738	94.92±0.31	95.17±0.19	**95.18±0.18**	16759	77.06±1.40	**81.41±1.18**	80.51±1.72
Citeseer	63266	**97.21±0.09**	97.06±0.21	97.02±0.12	24020	85.40±0.78	87.96±0.79	**88.40±0.92**
CS	4209161	98.31±0.03	98.30±0.02	**98.42±0.02**	91458	87.92±0.19	**91.47±0.35**	90.44±0.84
Physics	11462743	99.01±0.01	99.01±0.02	**99.02±0.00**	103174	86.21±0.33	89.94±0.31	**90.23±0.51**
Photos	2984253	97.85±0.06	**98.03±0.04**	97.87±0.02	104737	59.80±1.33	**68.11±0.43**	64.32±0.73
Computers	5417165	97.58±0.07	**97.60±0.08**	97.54±0.09	217090	46.49±0.75	57.32±0.99	**57.63±0.49**
IGB-100K	6899924	98.70±0.00	**98.71±0.02**	98.64±0.01	1372994	97.14±0.10	**98.63±0.42**	98.23±0.06



(a) Isolated (b) Low-degree (c) Overall

Figure 7: Ablation study on duplication frequency and nodes of NodeDup. (a), (b), (c) show the performance for Isolated nodes, Low-degree nodes, and Overall settings, respectively. The numbers displayed in each block represent the differences compared to duplicating cold nodes once.

Table 8: Link prediction performance with different duplication nodes of NodeDup on Citeseer. "D_*" indicates duplication of "*" group nodes for one time.

	Isolated	Low-degree	Warm	Overall
Supervised	47.13	61.88	71.45	63.77
D_Isolated	54.04	72.28	74.53	69.95
D_Cold	57.54	75.50	74.68	71.73
D_Mid-warm	46.93	61.34	71.84	63.75
D_Warm	47.49	62.20	71.54	63.99
D_Random	54.10	72.39	75.05	70.06
D_All	58.87	76.09	76.01	72.44

D.3 Influence of the duplication frequency and nodes

In our experiments, we duplicate cold nodes once and add one edge for each cold node in NodeDup. In Figure 7, we present the results of our ablation study, focusing on the effects of duplication frequency and duplicated nodes on the performance of NodeDup in terms of Isolated, Low-degree, and Overall settings. The numbers displayed in each block represent the differences compared to duplicating cold nodes once. We observe that increasing the duplication times does not necessarily lead to improvements across all settings, except when duplicating all nodes for Isolated nodes performance. We also notice that duplicating all nodes multiple times can significantly enhance the performance on Isolated nodes. However, this strategy negatively impacts the overall performance due to the increased number of isolated nodes in the graph. As a result, duplicating cold nodes once remains the optimal strategy, consistently yielding strong performance across all settings.

To make our analysis more comprehensive, we further conducted the experiments to show the results with duplicating warm nodes, mid-warm nodes, and randomly sampled nodes for one time, respectively, on Citeseer. The results are shown in Table 8. From the table, we can observe that duplicating mid-warm and warm nodes are not useful for the LP performance for all the settings. It's probably because for the mid-warm and warm nodes, the neighbors' information and supervised training signals are informative enough, therefore NodeDup cannot contribute more. We can also observe that duplicating random nodes is more effective than duplicating warm nodes but less effective than duplicating cold nodes and duplicating all nodes.

D.4 Performance on large-scale datasets

As outlined in Section 3.1, our methods incur a minimal increase in time complexity compared to base GNNs, with the increase being linearly proportional to the number of cold nodes. This ensures scalability. Besides, the effectiveness of our method is also insensitive to dataset size. We extend our experiments to the `IGB1M` dataset, featuring 1 million nodes and 12 million edges. The findings, which we detail in Table 9, affirm the effectiveness of our methods in handling large-scale datasets, consistent with observations from smaller datasets.

Table 9: Performance on the large-scale dataset. The best result is **bold**. Our method consistently outperforms GSage on `IGB1M`.

		GSage	NodeDup
`IGB1M`	Isolated	82.10±0.06	**87.81**±0.40
	Low-degree	84.73±0.06	**90.84**±0.03
	Warm	89.98±0.02	**91.31**±0.02
	Overall	89.80±0.02	**91.29**±0.02

Table 10: Performance on heterophilic datasets. The best result for each dataset is **bold**.

		GSage	NodeDup(L)	NodeDup
Chameleon	Isolated	24.91±6.75	**30.76±4.02**	27.37±2.88
	Low-degree	79.09±1.21	80.11±0.68	**80.91±0.41**
	Warm	94.00±0.23	**94.01±0.12**	93.68±0.44
	Overall	92.77±0.19	**92.88±0.10**	92.57±0.44
Squirrel	Isolated	25.05±3.70	**33.07±3.20**	30.11±1.57
	Low-degree	63.34±2.12	66.61±0.26	**68.05±0.80**
	Warm	93.35±0.22	93.43±0.11	**93.82±0.13**
	Overall	92.89±0.23	93.02±0.11	**93.41±0.13**

Table 11: Performance compared with heuristic methods and DegFairGNN (Liu et al., 2023). The best result is **bold**. NODEDUP, consistently outperforms all the heuristic methods and DegFairGNN.

		CN	AA	RA	DegFairGNN	GSage	NODEDUP
Cora	Isolated	0.00	0.00	0.00	18.70±1.53	32.20±3.58	**44.27±3.82**
	Low-degree	20.30	20.14	20.14	38.43±0.14	59.45±1.09	**61.98±1.14**
	Warm	38.33	38.90	38.90	42.49±1.82	**61.14±0.78**	59.07±0.68
	Overall	25.27	25.49	25.49	39.24±1.10	58.31±0.68	**58.92±0.82**
Citeseer	Isolated	0.00	0.00	0.00	15.50±1.27	47.13±2.43	**57.54±1.04**
	Low-degree	26.86	27.00	27.00	45.06±0.96	61.88±0.79	**75.50±0.39**
	Warm	37.30	39.02	39.02	55.47±1.08	71.45±0.52	**74.68±0.67**
	Overall	30.81	31.85	31.85	44.58±1.03	63.77±0.83	**71.73±0.47**
CS	Isolated	0.00	0.00	0.00	17.93±1.35	56.41±1.61	**65.87±1.70**
	Low-degree	39.60	39.60	39.60	49.83±0.68	75.95±0.25	**81.12±0.36**
	Warm	72.73	72.74	72.72	61.72±0.37	84.37±0.46	**84.76±0.41**
	Overall	69.10	69.11	69.10	60.20±0.37	83.33±0.42	**84.23±0.39**
Physics	Isolated	0.00	0.00	0.00	19.48±2.94	47.41±1.38	**66.65±0.95**
	Low-degree	46.08	46.08	46.08	47.63±0.52	79.31±0.28	**84.04±0.22**
	Warm	85.48	85.74	85.70	62.79±0.82	90.28±0.23	**90.33±0.05**
	Overall	83.87	84.12	84.09	62.13±0.76	89.76±0.22	**90.03±0.05**
Computers	Isolated	0.00	0.00	0.00	9.36±1.81	9.32±1.44	**19.62±2.63**
	Low-degree	28.31	28.31	28.31	18.90±0.81	57.91±0.97	**61.16±0.92**
	Warm	59.67	63.50	62.84	31.44±2.25	66.87±0.47	**68.10±0.25**
	Overall	59.24	63.03	62.37	31.27±2.22	66.67±0.47	**67.94±0.25**
Photos	Isolated	0.00	0.00	0.00	12.99±1.51	9.25±2.31	**17.84±3.53**
	Low-degree	28.44	28.78	28.78	20.18±0.21	52.61±0.88	**54.13±1.58**
	Warm	64.53	67.26	66.88	42.72±0.89	67.64±0.55	**68.68±0.49**
	Overall	63.94	66.64	66.26	42.37±0.87	67.32±0.54	**68.39±0.48**
IGB-100K	Isolated	0.00	0.00	0.00	57.09±21.08	75.92±0.52	**88.04±0.20**
	Low-degree	12.26	12.26	12.26	59.45±21.84	79.38±0.23	**88.98±0.17**
	Warm	30.65	30.65	30.65	65.57±20.43	86.42±0.24	**88.28±0.20**
	Overall	26.22	26.22	26.22	64.16±20.70	84.77±0.21	**88.39±0.18**

D.5 PERFORMANCE ON HETEROPHILY DATASETS

We have conducted experiments on two heterophilic datasets (i.e., Chameleon (Pei et al., 2020) and Squirrel (Pei et al., 2020)), with the results shown in Table 10. Our methods improve GNN performance across all settings on these datasets. Specifically, NodeDup and NodeDup(L) enhance the performance of Isolated nodes by 9.9% and 23.5% on Chameleon, and by 20.2% and 32.0% on Squirrel.

D.6 PERFORMANCE COMPARED WITH HEURISTIC METHODS

We compare our method with traditional link prediction baselines, such as common neighbors (CN), Adamic-Adar(AA), Resource allocation (RA). The results are shown in Table 11. We observe that NODEDUP can consistently outperform these heuristic methods across all the datasets, with particularly significant improvements observed on Isolated nodes.

Table 12: Performance compared with base GNN model and baselines for cold-start methods (evaluated by MRR). The best result is **bold**, and the runner-up is underlined. NodeDup and NodeDup(L) outperform GSage and cold-start baselines almost all the cases.

Dataset	Setting	GSage	Imbalance	TailGNN	Cold-brew	NodeDup(L)	NodeDup
Cora	Isolated	16.73±1.50	17.12±0.77	20.88±0.97	15.96±1.60	22.83±0.48	**25.61**±1.41
	Low-degree	38.46±0.62	37.93±1.17	40.19±0.96	35.20±0.55	**40.20**±1.02	39.78±0.97
	Warm	36.97±0.60	34.94±0.87	36.39±0.51	31.97±0.31	**36.99**±0.41	35.34±0.32
	Overall	35.91±0.51	34.59±0.81	36.49±0.59	31.84±0.17	**36.89**±0.47	35.82±0.34
Citeseer	Isolated	29.36±2.30	28.35±1.02	22.49±1.67	21.91±5.24	34.19±0.77	**38.26**±1.26
	Low-degree	44.13±0.38	44.67±0.44	43.92±1.55	34.65±10.10	51.58±0.56	**53.71**±0.64
	Warm	46.68±0.48	46.95±1.01	45.93±1.17	36.45±7.50	**48.89**±0.53	48.05±0.42
	Overall	42.60±0.59	42.72±0.52	40.87±1.34	33.13±7.90	47.00±0.44	**48.05**±0.54
CS	Isolated	35.54±0.74	29.61±1.62	30.32±0.92	32.35±0.77	42.22±1.41	**44.94**±0.60
	Low-degree	56.18±0.81	57.44±0.68	46.66±0.61	42.67±0.26	61.20±0.64	**61.65**±0.84
	Warm	58.18±0.84	57.03±0.77	48.32±0.44	43.71±0.41	**59.94**±0.54	58.67±0.72
	Overall	57.73±0.83	56.72±0.73	47.96±0.45	43.48±0.38	**59.83**±0.52	58.74±0.70
Physics	Isolated	27.73±1.10	33.61±0.34	23.17±0.74	30.62±0.30	41.12±1.10	**45.62**±2.45
	Low-degree	61.40±0.52	62.74±0.27	47.05±0.39	41.95±0.15	64.04±0.43	**65.94**±0.21
	Warm	66.72±0.47	66.03±0.09	55.77±0.49	46.06±0.12	**66.94**±0.49	66.83±0.04
	Overall	66.39±0.47	65.80±0.09	55.36±0.49	45.86±0.12	**66.74**±0.49	66.72±0.04
Computers	Isolated	4.50±0.75	5.01±0.71	4.88±0.54	4.07±0.46	8.59±1.45	**9.65**±1.10
	Low-degree	26.65±0.62	26.85±0.31	21.22±0.56	23.40±0.59	28.85±1.13	**29.78**±0.32
	Warm	34.11±0.40	33.77±0.17	31.02±0.34	28.75±0.23	35.11±0.31	**35.63**±0.14
	Overall	33.98±0.40	33.65±0.16	30.88±0.34	28.64±0.23	35.00±0.31	**35.52**±0.13
Photos	Isolated	3.99±0.52	4.79±1.38	5.78±0.94	6.49±0.98	**8.23**±1.10	7.90±1.55
	Low-degree	25.10±1.35	24.60±1.20	20.41±1.29	21.54±0.35	**27.90**±0.90	26.90±1.29
	Warm	34.90±0.57	33.03±0.47	30.79±0.63	29.40±0.23	**36.84**±0.55	35.69±0.43
	Overall	34.71±0.57	32.87±0.47	30.60±0.63	29.26±0.22	**36.66**±0.54	35.52±0.43
IGB-100K	Isolated	53.20±0.24	50.81±0.41	45.25±0.26	48.42±0.25	59.34±0.51	**61.75**±0.47
	Low-degree	55.93±0.28	55.79±0.30	51.11±0.29	51.92±0.15	62.35±0.49	**63.91**±0.26
	Warm	61.31±0.49	60.63±0.40	55.91±0.18	50.88±0.20	**61.56**±0.48	61.24±0.19
	Overall	60.05±0.43	59.40±0.36	54.65±0.20	50.97±0.17	61.61±0.48	**61.73**±0.21

D.7 MRR results compared with the base GNN model and cold-start baselines

Table 12 presents the performance of our methods, evaluated using MRR, compared against the base GNN model and cold-start baselines. We can observe that NodeDup(L) consistently achieves significant improvements over the baseline methods for Isolated and Low-degree nodes across all datasets. NodeDup also outperforms baselines in 13 out of 14 cases on the cold nodes. This further demonstrates the superior effectiveness of our methods in addressing cold node scenarios. Furthermore, we can also observe that our methods consistently perform on par or above baseline methods in Warm nodes and the overall setting. This observation further supports the effectiveness of our methods in maintaining and even improving the performance of Warm nodes.

D.8 Efficiency comparison with the base GNN model and cold-start baselines

The efficiency comparison between our methods and cold-start baselines is presented in Figure 8. We can observe that our methods and Imbalance exhibit similar efficiency, comparable to GSage. However, TailGNN and Cold-brew demand significantly more preprocessing and training time. Cold-brew, in particular, needs the most preprocessing time as it needs to train a teacher model for distillation.

D.9 Performance compared with additional cold-start methods

Upsampling (Provost, 2000). In Section 3, we discussed the issue of under-representation of cold nodes during the training of LP, which is the main cause of their unsatisfactory performance. To tackle this problem, one straightforward and naive approach is upsampling (Provost, 2000), which involves increasing the number of samples in the minority class. In order to further demonstrate the effectiveness of our methods, we conducted experiments where we doubled the edge sampling

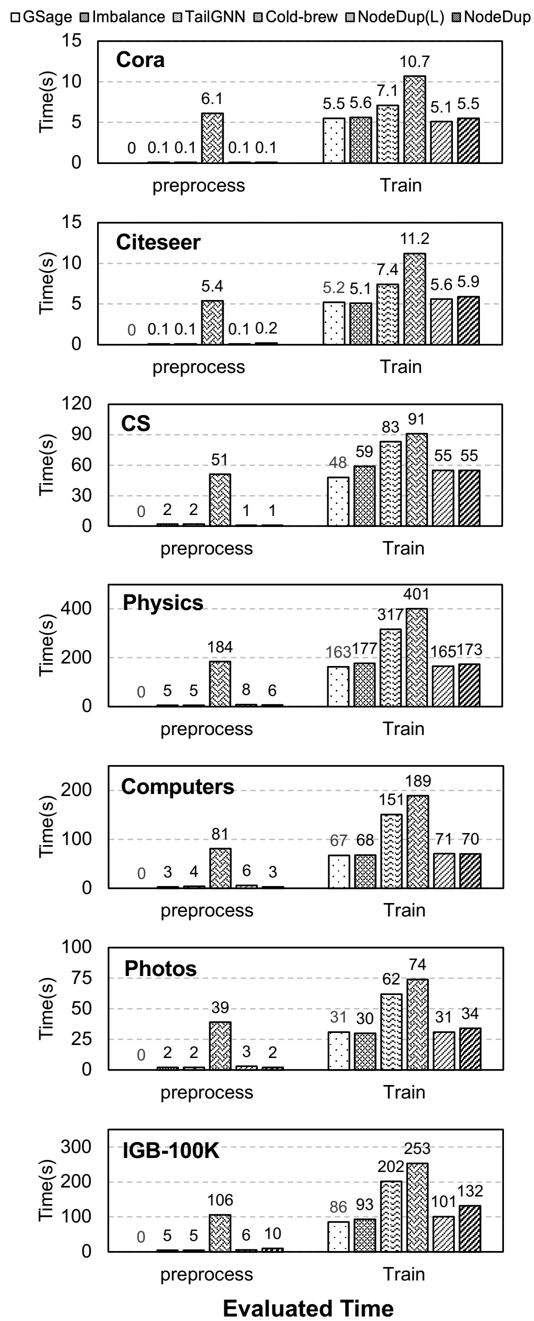


Figure 8: Time-consuming compared with cold-start methods. The histograms show the preprocessing and training time consumption of each method.

probability of code nodes, aiming to enhance their visibility. The results are presented in Table 13. We can observe that NodeDup(L) consistently outperforms upsampling, and NodeDup outperforms upsampling in almost all the cases, except for Warm nodes on `Cora`.

The methods tackling degree bias in GNNs. SAILOR (Liao et al., 2023) proposes a structural augmentation framework to enhance the representation learning of tail nodes. GRADE (Luo et al., 2024) improves structural fairness using graph contrastive learning methods. We used GCN as the encoder for both NodeDup(L) and NodeDup to ensure consistency, as both GRADE and SAILOR used GCN as their encoder. Table 14 shows that our methods outperform these baselines in all

Table 13: Performance compared with upsampling(evaluated by Hits@10). The best result is **bold**, and the runner-up is underlined. NODEDUP(L) consistently outperforms upsampling.

Dataset	Method	Isolated	Low-degree	Warm	Overall
Cora	Upsampling	32.81±2.75	59.57±0.60	<u>60.49</u>±0.81	57.90±0.65
	NODEDUP(L)	<u>39.76</u>±1.32	**62.53**±1.03	**62.07**±0.37	**60.49**±0.49
	NODEDUP	**44.27**±3.82	<u>61.98</u>±1.14	59.07±0.68	<u>58.92</u>±0.82
Citeseer	Upsampling	46.88±0.45	62.32±1.57	71.33±1.35	63.81±0.81
	NODEDUP(L)	<u>52.46</u>±1.16	<u>73.71</u>±1.22	**74.99**±0.37	<u>70.34</u>±0.35
	NODEDUP	**57.54**±1.04	**75.50**±0.39	<u>74.68</u>±0.67	**71.73**±0.47
CS	Upsampling	49.63±2.24	75.62±0.13	83.40±0.73	82.34±0.64
	NODEDUP(L)	<u>65.18</u>±1.25	**81.46**±0.57	**85.48**±0.26	**84.90**±0.29
	NODEDUP	**65.87**±1.70	<u>81.12</u>±0.36	<u>84.76</u>±0.41	<u>84.23</u>±0.39
Physics	Upsampling	52.01±0.97	79.63±0.13	89.41±0.32	89.33±0.46
	NODEDUP(L)	<u>65.04</u>±0.63	<u>82.70</u>±0.22	**90.44**±0.23	**90.09**±0.22
	NODEDUP	**66.65**±0.95	**84.04**±0.22	<u>90.33</u>±0.05	<u>90.03</u>±0.05
Computers	Upsampling	11.36±0.72	58.23±0.88	67.07±0.49	66.87±0.48
	NODEDUP(L)	<u>17.11</u>±1.62	**62.14**±1.06	<u>68.02</u>±0.41	<u>67.86</u>±0.41
	NODEDUP	**19.62**±2.63	<u>61.16</u>±0.92	**68.10**±0.25	**67.94**±0.25
Photos	Upsampling	10.92±2.15	51.67±0.98	65.75±0.73	65.45±0.71
	NODEDUP(L)	**21.50**±2.14	**55.70**±1.38	**69.68**±0.87	**69.40**±0.86
	NODEDUP	<u>17.84</u>±3.53	<u>54.13</u>±1.58	<u>68.68</u>±0.49	<u>68.39</u>±0.48
IGB-100K	Upsampling	75.49±0.90	79.47±0.11	86.54±0.19	84.87±0.14
	NODEDUP(L)	<u>87.43</u>±0.44	<u>88.37</u>±0.24	**88.54**±0.31	**88.47**±0.28
	NODEDUP	**88.04**±0.20	**88.98**±0.17	<u>88.28</u>±0.20	<u>88.39</u>±0.18

Table 14: Performance compared with GRADE (Luo et al., 2024) and SAILOR (Liao et al., 2023). The best result is **bold**.

		GCN	GRADE	SAILOR	NODEDUP(L)	NODEDUP
Cora	Isolated	40.61±3.52	43.29±2.62	45.12±1.29	42.93±2.68	**46.71±1.53**
	Low-degree	63.86±0.78	58.76±1.27	62.98±3.92	**64.63±1.60**	64.10±1.37
	Warm	60.59±0.62	60.00±0.51	57.34±3.80	**61.31±0.43**	60.26±0.70
	Overall	60.16±0.44	56.90±0.71	58.33±3.51	**61.02±0.61**	59.90±0.89
Citeseer	Isolated	45.56±1.30	50.11±2.24	49.29±2.75	47.84±0.94	**50.64±1.10**
	Low-degree	69.37±0.36	59.49±1.13	65.78±1.11	70.15±1.56	**71.13±0.64**
	Warm	**74.68±0.38**	70.01±0.50	72.66±0.37	73.26±0.97	72.93±0.78
	Overall	67.48±0.42	61.11±0.72	64.80±0.66	67.47±0.83	**67.67±0.66**

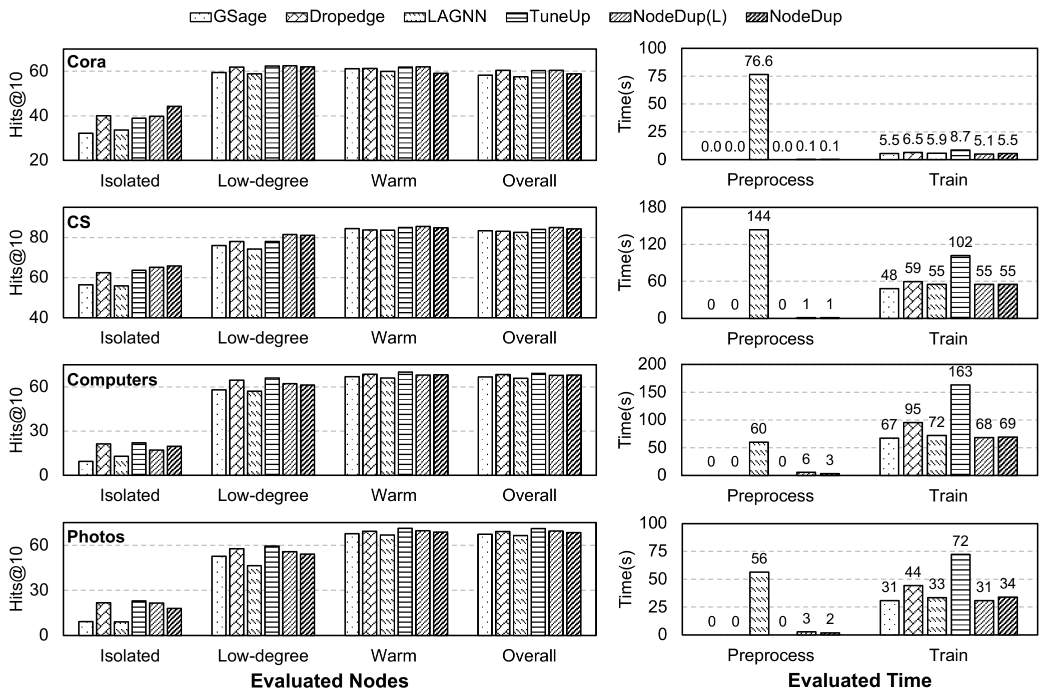


Figure 9: Performance and time-consuming compared with augmentation methods (Remaining results of Figure 5). The *left* histograms show the performance results, and the *right* histograms show the preprocessing and training time consumption of each method.

settings. Additionally, both GRADE and SAILOR perform better than vanilla GCN on Isolated nodes, which is the primary focus of their training. DegFairGNN (Liu et al., 2023) introduces a learnable debiasing function in the GNN architecture to produce fair representations for nodes, aiming for similar predictions for nodes within the same class, regardless of their degrees. Unfortunately, we've found in Table 11 that this approach is not well-suited for link prediction tasks for several reasons: (1) This method is designed specifically for node classification tasks. For example, the fairness loss, which ensures prediction distribution uniformity among low and high-degree node groups, is not suitable for link prediction because there is no defined node class in link prediction tasks. (2) This approach achieves significant performance in node classification tasks by effectively mitigating degree bias. However, in the context of link prediction, the degree trait is crucial. Applying DegFairGNN (Liu et al., 2023) would compromise the model's ability to learn from structural information, such as isomorphism and common neighbors. This, in turn, would negatively impact link prediction performance, as evidenced by references (Zhang & Chen, 2018; Chamberlain et al., 2022).

D.10 Additional results compared with augmentation baselines

Figure 9 presents the performance compared with augmentation methods on the remaining datasets. On `Cora` and `CS` datasets, we can consistently observe that NodeDup and NodeDup(L) outperform all the graph augmentation baselines for Isolated and Low-degree nodes. Moreover, for Warm nodes, NodeDup can also perform on par or above baselines. On the `Computers` and `Photos` datasets, our methods generally achieve comparable or superior performance compared to the baselines, except in comparison to TuneUP. However, it is worth noting that both NodeDup and NodeDup(L) exhibit more than $2\times$ faster execution speed than TuneUP on these two datasets.

Table 15: Performance in inductive settings (Remaining results of Table 2). The best result is **bold**, and the runner-up is underlined. Our methods consistently outperform GSage.

		GSage	NodeDup(L)	NodeDup
Cora	Isolated	43.64±1.84	45.31±0.83	**46.06**±0.66
	Low-degree	60.06±0.62	60.46±0.91	**61.94**±2.22
	Warm	60.59±1.13	60.95±1.40	**62.53**±1.23
	Overall	57.23±0.33	57.65±0.82	**59.24**±1.02
CS	Isolated	74.34±0.56	75.42±0.36	**77.80**±0.68
	Low-degree	75.75±0.48	77.02±0.65	**81.33**±0.60
	Warm	82.55±0.27	83.52±0.67	**83.55**±0.50
	Overall	81.00±0.28	82.01±0.59	**82.70**±0.52
Computers	Isolated	66.81±0.72	67.03±0.51	**69.82**±0.63
	Low-degree	64.17±2.01	65.10±1.76	**66.36**±0.69
	Warm	68.76±0.40	68.78±0.39	**70.49**±0.41
	Overall	68.54±0.42	68.59±0.39	**70.40**±0.42
Photos	Isolated	68.29±0.67	69.60±0.75	**70.46**±0.53
	Low-degree	63.02±1.51	64.25±1.31	**68.49**±2.39
	Warm	70.17±0.57	71.05±0.70	**71.61**±0.81
	Overall	69.92±0.57	70.84±0.63	**71.47**±0.77

Table 16: Performance with different encoders (Remaining results of Table 3), where the inner product is the decoder. The best result for each encoder is **bold**, and the runner-up is underlined. Our methods consistently outperform the base models, particularly for Isolated and Low-degree nodes.

		GAT	NodeDup(L)	NodeDup	JKNet	NodeDup(L)	NodeDup
Cora	Isolated	25.61±1.78	30.73±2.54	**36.83**±1.76	30.12±1.02	37.44±2.27	**43.90**±3.66
	Low-degree	54.88±0.84	55.76±0.50	**56.72**±0.81	59.56±0.66	61.93±1.64	**62.89**±1.43
	Warm	55.31±1.14	**55.36**±1.28	53.70±1.26	58.64±0.12	**59.36**±1.00	57.67±1.60
	Overall	52.85±0.91	**53.58**±0.80	53.43±0.49	56.74±0.27	**58.54**±0.83	58.40±1.33
CS	Isolated	33.74±1.98	34.77±0.90	**41.76**±2.99	54.43±1.77	56.38±2.14	**64.79**±1.68
	Low-degree	70.20±0.47	70.90±0.32	**71.92**±0.36	73.97±0.72	76.64±0.38	**77.77**±0.43
	Warm	78.39±0.28	**78.67**±0.33	77.69±0.89	82.38±0.67	**83.29**±0.37	79.20±0.13
	Overall	77.16±0.24	**77.49**±0.30	77.20±0.80	81.35±0.62	**82.41**±0.32	78.91±0.13
Computers	Isolated	12.04±2.08	16.84±2.34	**17.17**±2.22	9.92±3.07	23.81±2.02	**25.50**±1.32
	Low-degree	53.60±1.51	53.62±1.00	**53.65**±2.35	62.29±1.08	67.21±0.99	**68.49**±0.70
	Warm	**60.19**±1.19	58.64±0.81	58.55±1.01	69.96±0.33	**70.90**±0.40	70.66±0.25
	Overall	**60.03**±1.19	58.50±0.80	58.77±1.93	69.77±0.32	**70.78**±0.40	70.55±0.25
Photos	Isolated	15.31±3.46	18.03±2.50	**18.77**±3.33	12.77±2.40	19.44±1.31	**20.56**±1.61
	Low-degree	43.11±9.93	43.40±9.61	**44.21**±9.25	57.27±2.06	59.86±1.09	**60.93**±0.74
	Warm	56.17±8.28	**56.75**±8.33	56.10±8.35	68.35±0.81	69.56±0.69	**69.60**±0.50
	Overall	55.91±9.22	**56.48**±8.26	55.93±8.28	68.09±0.82	69.33±0.68	**69.38**±0.49

D.11 Additional results under the inductive setting

We further evaluate and present the effectiveness of our methods under the inductive setting on the remaining datasets in Table 15. We can observe that both NodeDup and NodeDup(L) consistently outperform GSage for Isolated, Low-degree, and Warm nodes. Compared to NodeDup(L), NodeDup is particularly beneficial for this inductive setting.

D.12 Ablation study

D.12.1 Performance with various encoders and decoders

For the ablation study, we further explored various encoders and decoders on the remaining datasets. The results are shown in Table 16 and Table 17. From these two tables, we can observe that regardless of the encoders or decoders, both NodeDup and NodeDup(L) consistently outperform the base model for Isolated and Low-degree nodes, which further demonstrates the effectiveness of our methods on cold nodes. Furthermore, NodeDup(L) consistently achieves better performance compared to the base model for Warm nodes.

Table 17: Link prediction performance with MLP decoder (Remaining results of Table 4), where GSage is the encoder. Our methods achieve better performance than the base model.

		MLP-Dec.	NodeDup(L)	NodeDup
Cora	Isolated	16.83±2.61	37.32±3.87	**38.41**±1.22
	Low-degree	58.83±1.77	**64.46**±2.13	64.02±1.02
	Warm	58.84±0.86	**61.57**±0.98	58.66±0.61
	Overall	55.57±1.10	**60.68**±0.66	58.93±0.25
CS	Isolated	5.60±1.14	58.68±0.95	**60.20**±0.68
	Low-degree	71.46±1.08	78.82±0.68	**79.58**±0.31
	Warm	84.54±0.32	**85.88**±0.22	85.20±0.24
	Overall	82.48±0.32	**84.96**±0.25	84.42±0.22
Computers	Isolated	6.13±3.63	**27.74**±3.38	26.70±3.98
	Low-degree	62.56±1.34	62.60±3.38	**63.35**±3.64
	Warm	69.72±1.31	**70.01**±2.41	68.43±2.50
	Overall	69.53±1.30	**69.91**±3.11	68.30±2.51
Photos	Isolated	6.34±2.67	18.15±2.02	**18.97**±1.71
	Low-degree	55.63±6.21	**56.13**±6.36	55.93±7.27
	Warm	70.40±6.84	**70.67**±6.30	69.97±5.07
	Overall	69.89±6.81	**69.93**±6.24	69.69±5.07

Table 18: Performance with GCN (Kipf & Welling, 2016a) and GT (Dwivedi & Bresson, 2020) encoders, where the inner product is the decoder. The best result for each encoder is **bold**.

		GCN	GCN+NodeDup(L)	GCN+NodeDup	GT	GT+NodeDup(L)	GT+NodeDup
Cora	Isolated	40.61±3.52	42.93±2.68	**46.71±1.53**	20.93±2.46	**38.82±1.27**	37.40±1.53
	Low-degree	63.86±0.78	**64.63±1.60**	64.10±1.37	58.59±0.29	61.16±1.08	**61.39±0.89**
	Warm	60.59±0.62	**61.31±0.43**	60.26±0.70	58.14±1.15	**59.29±0.84**	59.07±0.05
	Overall	60.16±0.44	**61.02±0.61**	59.90±0.89	55.40±0.43	**58.34±0.19**	58.18±0.42
Citeseer	Isolated	45.56±1.30	47.84±0.94	**50.64±1.10**	36.84±3.26	51.46±1.27	**52.34±1.46**
	Low-degree	69.37±0.36	70.15±1.56	**71.13±0.64**	60.24±1.18	72.98±1.54	**73.77±1.03**
	Warm	**74.68±0.38**	73.26±0.97	72.93±0.78	71.14±1.47	74.48±1.08	**75.08±0.63**
	Overall	67.48±0.42	67.47±0.83	**67.67±0.66**	61.15±1.57	69.67±1.10	**70.38±0.86**

Besides GSage, GAT and JKNet, we also conducted further experiments with convolutional-based GNNs, such as GCN (Kipf & Welling, 2016a) and GT(GraphTransformer) (Dwivedi & Bresson, 2020). The results are shown in Table 18. Our findings indicate that our methods can also improve performance when using GCN and GT as the encoder. However, since GCN uses the same matrix for both self-representations and neighbor representations, our methods only benefit from the supervision aspect. This leads to less pronounced performance improvements on cold nodes compared to using GT and GSage as the encoder. Specifically, NodeDup shows a 13.10% improvement for GCN, 60.38% for GT, and 29.79% for GSage on isolated nodes. Moverover, NodeDup(L) on average improves GCN by 5.4%, GT by 62.58%, and GSage by 17.4%.

D.12.2 Performance with SEAL (Zhang & Chen, 2018)

Considering our methods are flexible to integrate with GNN-based link prediction structures, we conduct the experiments on top of SEAL (Zhang & Chen, 2018) on the `Cora` and `Citeseer` datasets. The results are shown in Table 19. We can observe that adding NodeDup on top of SEAL can consistently improve link prediction performance in the Isolated and Low-degree node settings on these two datasets.

Table 19: Performance with SEAL (Zhang & Chen, 2018) on `Cora` and `Citeseer` datasets.

		SEAL	SEAL + NodeDup
`Cora`	Isolated	62.20±1.06	**70.73**±0.61
	Low-degree	66.80±2.83	**67.70**±4.11
	Warm	**56.69**±2.36	54.87±1.61
	Overall	60.60±2.38	**60.89**±2.36
`Citeseer`	Isolated	56.92±5.53	**66.37**±1.01
	Low-degree	64.13±2.56	**65.54**±1.69
	Warm	58.81±3.22	**60.73**±2.75
	Overall	60.18±2.98	**63.35**±1.43

E Implementation Details

In this section, we introduce the implementation details of our experiments. Our implementation can be found at `https://anonymous.4open.science/r/NodeDup-0241/README.md`.

Parameter Settings. We use 2-layer GNN architectures with 256 hidden dimensions for all GNNs and datasets. The dropout rate is set as 0.5. We report the results over 10 random seeds. Hyperparameters were tuned using an early stopping strategy based on performance on the validation set. We manually tune the learning rate for the final results. For the results with the inner product as the decoder, we tune the learning rate over range: $lr \in \{0.001, 0.0005, 0.0001, 0.00005\}$. For the results with MLP as the decoder, we tune the learning rate over range: $lr \in \{0.01, 0.005, 0.001, 0.0005\}$.

Hardware and Software Configuration All methods were implemented in Python 3.10.9 with Pytorch 1.13.1 and PyTorch Geometric (Fey & Lenssen, 2019). The experiments were all conducted on an NVIDIA P100 GPU with 16GB memory.

F Limitations

In our work, NodeDup and NodeDup(L) are specifically proposed for LP tasks. Although cold-start is a widespread issue in all graph learning tasks, our proposed methods might not be able to generalize to other tasks, such as node classification, due to their unique design. Furthermore, the two heterophily datasets we used for evaluation involve graphs where nodes with similar features are assigned different labels. Our methods may struggle on heterophilic graphs where connected nodes have dissimilar features, such as molecular networks, which are beyond the scope of this study.