DBLP: Boosting GNN in link prediction via its decoder

Yule Liu [1], Beiyuan Yang [1], and Jie Zheng [1,2]

[1]School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China, [2] Shanghai Institute of Materia Medica, Chinese Academy of Sciences,China

Abstract. Although graph neural network (GNN) has shown its superiority in many graph tasks, its limited ability to distinguish automorphic nodes (those sharing identical structural roles) poses challenges, potentially resulting in poor link prediction performance. Existing methods focus on designing robust GNN encoders that apply node labeling tricks or restrict computation to subgraphs enclosing a link, thus converting the link prediction task into binary subgraph classification. However, the importance of the decoder, which maps the node pair representations to predictions, has been largely overlooked in predicting links on a graph. In this paper, we propose a universal framework named DBLP (Decoder Boosting for Link Prediction), which focuses on solving the automorphic problems in link prediction by boosting the decoder. We show that a simple graph encoder embedded in DBLP can have similar or better performance compared to advanced baselines. DBLP reveals the potential of GNNs in link prediction tasks by exploiting a better decoder. Extensive experiment results show the universality and the effectiveness of our framework.

1 Introduction

Graph Neural Networks (GNNs) have demonstrated their superior ability in various graph learning tasks like node classification, graph classification, link prediction, etc. [26, 14, 42, 43, 47, 17]. Among these tasks, link prediction (i.e. predicting whether two nodes in a network are likely to have a link) is one of the most important due to its wide applications in practice [49], including recommender systems [17, 46], knowledge graphs [29], social networks [15], drug target interaction [34], and biological networks [31]. GNNs can learn node representations via the message passing mechanism [42], which effectively incorporates graph topology and node or edge features and is a main reason behind the great success of GNNs in link prediction.

Although GNNs represent a powerful learning framework, they still suffer from a widely-discussed intrinsic problem [36, 8, 48, 49], which is referred to as the *automorphic node problem* [8]. To generate the link representation, GNNs learn permutation-equivariant structural node representations [30, 36] and use a readout function that maps from the representations of two nodes to the probability of existing a link between the two nodes. As a result, all nodes having identical structural roles induced by the graph automorphism group will have equal representations. If two nodes belong to different clusters while their topological environments are the same, the predictor may assert there will be a link between them which is not the case. The automorphic problem limits the prediction power of GNNs [8]. A detailed explanation of the automorphic problem and the concept comparison between automorphism and isomorphism are shown in Appendix A.

To solve the aforementioned problem, current methods focus on designing robust GNNs that apply node labeling tricks [48, 49] or restrict computation to subgraphs enclosing a link, thereby transforming the link prediction into the binary subgraph classification [45, 5]. However, as an important part of link prediction, the utility of the decoder, which maps the representation of the node pair to the result, was largely ignored in link prediction tasks. Since a key to relieving the problem is to specialize the nodes that are in the same topological environment, we add a special mark (denoted by t), which is a binary variable, to the edge whose two nodes are in the same topological environment (i.e. t will be 1 in this case and 0 otherwise). Then the decoder will predict the edge whose $t = 1$ with new parameters. In this way, the nodes that are in the same topological environment will be treated differently because it use separate parameters for the special pairs. We will show that such a simple modification can greatly improve the performance on different real-world datasets.

Despite the improved expressiveness in many datasets, when the marks are selected based on the observational node representations, a main challenge in working with the imbalanced marks distribution $p(t)$, unlike in randomized cases (i.e. when the marks are independent of the embeddings derived from GNNs), is that edges with different marks can not be assumed equivalent This is because marks are selected based on whether the two nodes are in the same topological environment, which is presented by a distribution shift between $p(Z|t = 1)$ and $p(Z|t = 0)$ [22]. A visual example is demonstrated in Appendix B. If not adjusted for, this imbalance may result in inflated variance in the estimation of potential outcomes [33, 50]. A widely-used method is to use re-weighting tricks to obtain an unbiased estimation of the risk from an empirical sample [35, 9]. However, instead of learning directly from the original features, our decoder learns from the embeddings learned by GNNs. Inspired by the balancing techniques on representation learning [21, 2], we seek a balancing weights decoder for link prediction that is both predictive of potential outcomes and balanced across different marks.

In the paper, we propose DBLP (**De**coder **B**oosting for **L**ink **P**rediction), a universal framework, which can be used by most GNNs for link prediction. In DBLP, we do not modify the GNN architecture for embedding learning, instead, the difference between DBLP and other frameworks lies in how to utilize the embeddings learned by GNNs. Particularly, we focus on the automorphism problem in the embeddings learned by GNNs and try to find an efficient solution from the utilization of the embeddings compared to most works that develop a new GNN network to learn better embeddings.

Our contributions can be summarized as follows:

- It is the first time to focus on the utility of the decoders for link prediction and our study demonstrates that purposeful enhancement of the decoder will lead to much better performance of prediction. We propose a universal framework, DBLP, which improves the performance of most GNNs.
- We show that a simple graph encoder embedded in DBLP can have similar or better performance compared to advanced baselines. DBLP reveals the potential of GNNs in link prediction tasks by exploiting a better decoder.

2 Related Work

2.1 Graph automorphism problem

Link prediction is the task of predicting future relationships between entities or identifying missing connections within a dynamic network. Although GNNs represent a powerful learning framework, they still suffer from a widely-discussed intrinsic problem [36], which is referred to as *automorphic node problem* [8]. Various methods have been proposed to relieve this problem. Some use node labeling tricks [48, 47, 49], which assign labels to different nodes such that nodes in the same topological environment will have different representations, leading to correct prediction. Some restrict computation to subgraphs enclosing a link, transforming link prediction into binary subgraph classification [49, 45, 5]. Compared to the methods that design complex GNNs, we try to exploit the potential impact of the decoder, which has been greatly ignored.

2.2 Balancing weights and causal reasoning

The goal of balancing weight estimators is to alleviate systematic differences in baseline covariates across treatment groups, aiming to improve the individual treatment effect, because the imbalance may result in inflated variance in the estimation of potential outcomes. [24]. Currently, many works try to construct re-weighting estimators that explicitly minimizing the discrepancy between two groups [2, 4, 6]. They rely on assigning a prior to the data representation and seek a hypothesis that is both predictive of potential outcomes and balanced across different groups [1].

A work similar to ours is *CFLP* [51]. They estimated counterfactual links on a graph and proposed *CFLP*, a framework trained with counterfactually augmented data aiming to learn representations of the causal structure of tasks. Although we have similar marking strategies (treatment in CFLP) and optimization targets, our work is distinguished from *CFLP* in the following aspects. First, we differ in motivation. *CFLP* augments the graph encoder with the additional estimated counterfactual data. By contrast, our work aims to solve the automorphism problem in link prediction with imbalanced distribution via the decoder enhancement. Secondly, although we have similar training objectives with *CFLP*, the targets to balance are different. *CFLP* tries to balance the distributions between the factual data and the estimated counterfactual data, whereas we are balancing the data units across different mark groups, which are all factual. Thirdly, *CFLP* uses similarity-based estimation to obtain the counterfactual data points, which is a slow process with a time complexity of $\mathcal{O}(N^4)$. Our algorithm does not need the counterfactual data and thus saves the time for the estimation. As a result, *DBLP* runs much faster.

3 Problem Definition and Notation

We will use the following problem definition and notations to propose our method. Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph , where $\mathcal{V} = \{v_i : i \in [n]\}$ is the set of n nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. We denote the adjacency matrix as $\mathbf{A}$ and the node feature matrix as $\mathbf{X} \in \mathbb{R}^{n \times d}$, where d is the dimension of feature embeddings. A graph neural network typically takes $(\mathbf{A}, \mathbf{X})$ as input, and tries to learn an encoder $\Phi : \mathbb{R}^d \to \mathbb{R}^h$, which transforms the original node feature space to the representation space. And We will use $\mathbf{Z} = \Phi(\mathbf{X})$ and an additional discriminator $h : \mathbb{R}^{2h} \to \{0, 1\}$ over the edge representation (concatenation of the related embeddings of $(\mathbf{Z}_{v_i}, \mathbf{Z}_{v_j}) : i \in [n], j \in [n]$) to give out the result whether there is a link between the node pair (v_i, v_j).

To introduce the link prediction task with enhancement on the decoder, we need some extra notations and assumptions following [33]. Let the space of edge representation be a bounded subset $\mathcal{X} \subset (\mathbb{R}^d \times \mathbb{R}^d)$ and the outcome space be $\mathcal{Y} \subset \mathbb{R}$. We assume that for a unit with features $x \in \mathcal{X}$ and the edge-marking $t \in \{0, 1\}$ (we will detail introduce how to construct the mark in the next section), there are two potential outcomes: $y = Y_0$ and $y = Y_1$. For each data point, we are able to get one of the potential outcomes, depending on the assignment of the edge marking t: if $t = 0$ we observe $y = Y_0$, if $t = 1$, we observe $y = Y_1$; this is known as the consistency assumption.

Furthermore, we will adopt a commonly accepted important "no-hidden confounders" assumption. We formalize the assumption by using the ignorability and overlap [20, 32]:

$$\forall t \in \{0, 1\} : \underbrace{Y(t)T \mid \Phi(X)}_{\text{Ignorability}} \quad \text{and}$$

$$\forall z \in \mathcal{Z} : \underbrace{p(T = t \mid \Phi(X) = z) > 0}_{\text{Overlap}}$$

Then we could assume that there exists a joint distribution $\mathbb{D}(x, t, Y_0, Y_1)$ which satisfies the above assumptions. Let $\Phi : \mathcal{X} \to \mathbb{R}^h$ be a representation function (with a slight abuse of notation), and $h : \mathbb{R}^h \times \{0, 1\} \to \mathbb{R}$ be a hypothesis over the representation space $\mathbb{R}^h$. And denote $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ as a loss function. We will define the risk of the hypothesis $h(x, t)$

Definition 1. The expected point-wise loss for the data pair (x, t) is given by:

$$\ell_{h,\Phi}(x, t) = \mathbb{E}_{Y(t)|X}[L\left(Y_t, h(\Phi(x), t)\right)|X] \tag{1}$$

Definition 2. Let the expected loss for a single data unit be $\ell_{h,\Phi}(x, t)$. Then we can derive the marginal risk $\epsilon(h, \Phi)$ of a hypothesis w.r.t a population $p(X)$ and the group risk $\epsilon_t(h, \Phi)$ w.r.t $p(X|T = t)$:

$$\epsilon(h, \Phi) = \mathbb{E}_X[\ell_{h,\Phi}(x, t)] \tag{2}$$

$$\epsilon_t(h, \Phi) = \mathbb{E}_{X|T}[\ell_{h,\Phi}(x, t)|T = t] \tag{3}$$

Furthermore, our framework relies on *Integral Probability Metric* (IPM), which is a class of metrics between probability distributions [11, 12]:

Definition 3. For a function family $\mathcal{G}$ of functions $g : \mathcal{S} \to \mathbb{R}$, we have that

$$\mathrm{IPM}_{\mathrm{G}}(p, q) = \sup_{g \in \mathrm{G}} \left| \int_{\mathcal{S}} g(s)(p(s) - q(s))ds \right| \tag{4}$$

Unlike KL-divergence, IPM is a symmetric metric that obeys the triangle inequality. Also, it is monotone, which means the higher the similarity of the two distributions, the smaller the values IPM will output [37].

4 Method

In this section, we will describe our model in two parts. First, we will introduce a novel and simple framework for link prediction that takes the automorphic node pairs into account. Second, we introduce a balancing weights estimator which alleviates the covariant imbalance problems in training.

4.1 Jointly training framework

Due to the automorphism problem [36, 8], graph neural networks may perform poorly in the cases shown in the previous section. Because GNN will assign the same color to the nodes whose topological environments are the same [36], it will assert there will exist a link between node 1 and node 2 which is obviously unreasonable (Figure. 1). In this part, we will introduce additional marks on node pairs and enhance the decoder to solve the problem.

Weisfeiler-Lehman test and marking The classical Weisfeiler-Lehman (WL) algorithm [41], a graph-isomorphism test based on color refinement, became relevant to the study of graph neural networks [42] in recent years. The first-order WL test [41, 19], keeps a state for each node that is updated by aggregating their neighbors' color. In this process, two nodes will be given the same color if their topological environments are the same, which satisfies the definition of automorphic nodes. Because of the equivalence between the WL test and message-passing-based GNNs, the two nodes that have the same color can be regarded as error-prone pairs in link prediction using GNNs.

Formally speaking, for the given graph $G = (\mathcal{V}, \mathcal{E})$, we conduct the previous WL test in the training graph and get the embeddings $(c(v_i), c(v_j))$ for all the node pairs (v_i, v_j). The mark t_{ij} will be calculated by $\nVdash(c(v_i), c(v_j))$. The idea of the introduction of the mark is: since we know that the GNNs will perform poorly in the automorphic nodes, we exploit the power of the decoder to deal with this problem by setting separate parameters for the harder case.

In this way, we are able to use a new hypothesis $h(\Phi(x), t)$ instead of the original $h(\Phi(x))$ and take the parameter isolation strategy. If the hypothesis is a multi-layer perception (MLP), the parameter isolation means $h(\Phi(x), 1)$ and $h(\Phi(x), 0)$ will be outputted from two different heads of a shared MLP layer. However, we observed a severe distribution imbalance between distribution $p(t = 1)$ and distribution $p(t = 0)$ in widely used real-world datsets. The imbalance is no surprise because the number of automorphic node pairs is much smaller than the non-automorphic node pairs, or GNNs will not get satisfactory results due to their limitation. If not adjusted for, this imbalance may result in inflated variance in estimates of potential outcomes [33, 22]. In the next part, we will deal with this imbalance.

4.2 Preventing distribution shift

Since the edge representation X and the edge marking T are confounders, the result $Y(t)$ is only accessible for edges that are distributed according to $p(X|T = t)$. As a result, unless marks are independent of the embeddings derived from GNNs, $\epsilon(h, \Phi)$ is different from $\epsilon_t(h, \Phi)$ in general [23]. The optimal hypothesis h for $\epsilon_t(h, \Phi)$ can be different from an optimal hypothesis for $\epsilon(h, \Phi)$, leading to inflated variance in estimates of potential outcomes [35]. To reduce the bias, a widely used method is to use re-weighting tricks to obtain an unbiased estimate of the risk from an empirical sample [35, 9]. The re-weighted marginal risk and re-weighted group risk are defined as follows,

$$\epsilon^{w}(h, \Phi) = \mathbb{E}_X[w(X)\ell_{h,\Phi}(x, t)] \tag{5}$$

$$\epsilon_t^{w}(h, \Phi) = \mathbb{E}_{X|T}[w(X)\ell_{h,\Phi}(x, t)|T = t] \tag{6}$$

where $w(X)$ is a parameter to balance the marks and [3] introduces some principles to chose $w(X)$. Suppose $\pi_t = p(T = t)$, the error gap between the vanilla estimator and the re-weighted estimator is bounded by [10]:

$$\begin{aligned}\epsilon(h, \Phi) = \epsilon_t^{\hat{w}}(h, \Phi) + (1 - \pi_t) \int_{\mathcal{X}} \ell_{h,\Phi}(x, t)(p_{1-t}(x) \\ - p_t^{w}(x))dx \end{aligned} \tag{7}$$

$$\leq \epsilon_t^{\hat{w}}(h, \Phi) + C(1 - \pi_t)\text{IPM}_{\mathcal{L}}(p_{1-t}(x), p_t^{w}(x)) \tag{8}$$

where $\hat{w} = \pi_t + (1 - \pi_t)w(x)$, C is a constant, $\mathcal{L}$ is the assumption for the risk function family. One can minimize the upper bound for the marginal risk to ensure the performance. However, instead of learning directly from the original features, our hypothesis learns from the embeddings learned by GNNs. Inspired by the balancing techniques on representation learning [21, 2], we derive a bound similar to the above when we learn from the low-dimension GNN embeddings $\Phi(X)$.

Lemma 1 Let $\Phi : \mathcal{X} \to \mathbb{R}^h$ be an invertible representation with Ψ as its inverse. Let G be a family of functions $g : \mathbb{R}^h \to \mathbb{R}$, and denote by $\text{IPM}_{\text{G}}(\cdot, \cdot)$ the integral probability metric induced by $\mathcal{G}$. Assume there exists a constant $B_\Phi > 0$, such that for $t = 0, 1$, the function $g(\Phi, h, r, t) := \frac{1}{B_\Phi} \cdot \ell_{h,\Phi}(\Psi(r), t) \in \mathcal{G}$. Then we have:

$$\begin{aligned}\int_{\mathcal{X}} \ell_{h,\Phi}(x, t)(p_{1-t}(x) - p_t^{w}(x))dx \\ \leq B_\Phi \text{IPM}_{\mathcal{G}}(p_{\Phi,1-t}(x), p_{t,\Phi}^{w}(x))\end{aligned} \tag{9}$$

Proof of Lemma 1 will be shown in Appendix. The intuition of the proof is to use the change of variable to transform the calculation on original features into the embeddings learned by GNNs. In this way, the marginal risk is bounded by:

$$\epsilon(h, \Phi) \leq \epsilon_t^{\hat{w}}(h, \Phi) + B_\Phi(1 - \pi_t)\text{IPM}_{\mathcal{G}}(p_{\Phi,1-t}(x), p_{t,\Phi}^{w}(x)) \tag{10}$$

The proof is left in the Appendix.

5 Algorithm for link prediction

We propose a simple yet effective framework DBLP (**De**coder **B**oosting for **L**ink **P**rediction) based on the above analysis. The input of DBLP is a graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, and the output is the logit of link prediction in G. The model consists of three parts: node labeling, graph encoder, and edge predictor. For node labeling, we mark every node pair according to the principle described in the previous part. And since DBLP is a universal framework for link prediction, the graph encoder Φ can be any GNN model. Here we use the most popular graph neural network GCN [26], and in each layer, it follows that:

$$m_{ij}^{(k)} = \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}}(W^{(k-1)}h_i^{(k-1)}))$$

$$h_i^{(k)} = ReLU(W \cdot MEAN(h_i^{(k-1)}, \sum_{j \in \mathcal{N}(i)} m_{ij}))$$

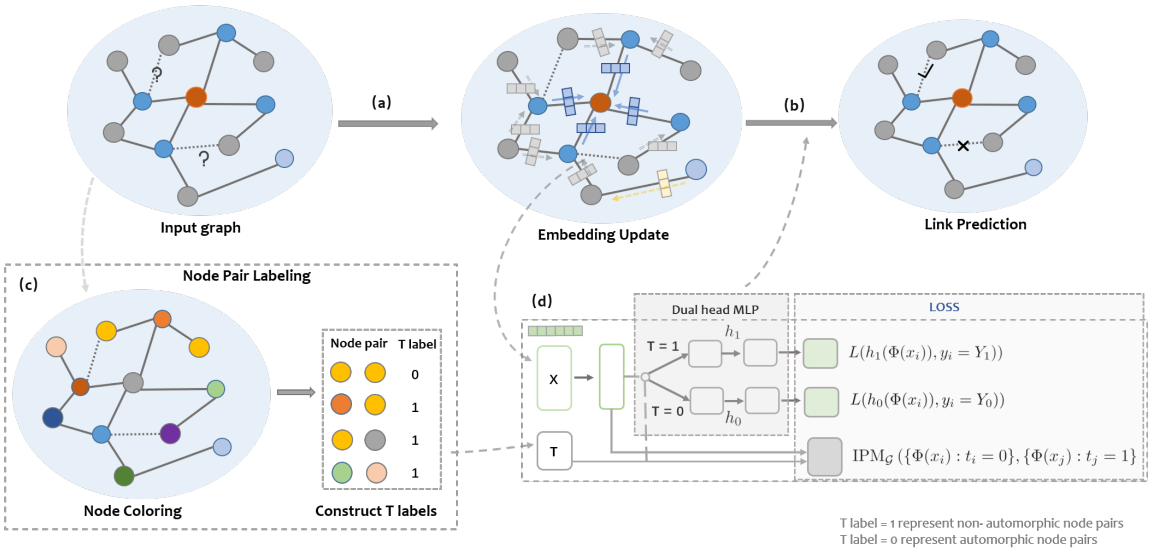

Figure 1. DBLP framework: Leverage the WL-test outcomes to assign colors to node pairs, and term them as T labels. Employ the dual-head MLP designated for T=0 and T=1 labels as the decoder for the graph neural networks during the link prediction task. The loss function comprises two distinct binary cross-entropy losses and an IPM loss, where the IPM loss is implemented to address the distribution shift.

where $h_i^{(k)} \in \mathbf{H}^{(K)}$ and $\mathbf{H}^{(1)} = \mathbf{X}$, $MEAN$ is the element-wise mean pooling and $ReLU$ is the nonlinear activation function.

In the third part, we set up a two-head MLP with a shared layer to implement the parameter isolation strategy. Given an embedding X learned by GNNs and the corresponding mark t, if $t = 1$, we use the first head to predict the logit and use the other if the mark is zero.

Training During the training of DBLP, we will seek an encoder Φ and h by optimizing the following objective:

$$\min_{h,\Phi} \frac{1}{n} \sum_{i=1}^{n} L(h(\Phi(x_i), t_i), y_i) + \lambda \|\mathbf{W}\|_2 + \alpha \, \mathrm{IPM}_{\mathcal{G}} \left(\{\Phi(x_i) : t_i = 0\}, \{\Phi(x_j) : t_j = 1\} \right)$$

where $\|\mathbf{W}\|$ is the L2 regularization term and α, λ are the hyper-parameters. Note that for most IPMs, we cannot compute the factor B_Φ in Equation 10, but we can treat it as part of the hyperparameter α. Also, we only sample part of the representations instead of all the representations to estimate IPM for the sake of efficiency. We train our models by minimizing the objective above using Adam [25] with a zig-zag learning rate scheduler, where we backpropagate the error through both the hypothesis and representation networks.

6 Experiment

In this section, we first describe the experimental setup. Then we show the performance of DBLP, followed by a comparative study and ablation studies. We focus on several topics: 1) How does the proposed framework perform if we adopt it to popular baselines? And how does our method fare against the more advanced models in link prediction tasks? 2) What is the effect of using the edge marking strategy? Do we relieve the node automorphic problem? 3) What is the effect of the balancing weight estimator? How does the proposed estimator influence the distribution of the edge embeddings?

6.1 *Experiment setup*

Dataset In this study, we use the benchmark datasets including citation networks CiteSeer, Pubmed [44], social networks Facebook[28], and biological networks including drug-target interaction (DTI) network KIBA [38] and standard drug-drug interaction OGB-DDI network [18]. We randomly select $70\%/10\%$ of the node pairs as training/validation samples and the remaining 20% as test samples. And the links in the validation and test sets are masked from the training graph. For OGB-DDI dataset, we follow the official setup reported in [18]. All the datasets used in this work are publicly available. Further information on the datasets is listed in the Appendix (table D.1).

Metric We use Hit@20 and Hit@50 as the evaluation metrics for the model, where Hit@k is a performance metric that measures the percentage of relevant items successfully retrieved within the top k candidates suggested by the algorithm [18]. The results are average with deviation over 10 runs with different parameter initialization.

Baseline methods We compared our model with several recently proposed baseline methods for link prediction tasks. These methods include the position embedding methods *Node2Vec* [13], GNN-based methods *GCN* [26], *SAGE* [14], *GAT*, [40] *VGAE* [27], *JKNet* [43]), and other models designed for the node automorphic problems *SEAL* [48], and *LGLP* [7]. Also, we compare our model with *CFLP* [51] which has similar optimization targets to us.

Implement details We show the hyperparameter settings and other implement details for each dataset in the Appendix. To ensure the fairness and consistency of the experiment, some of the results are referenced from OGB-leaderboard [18]. We trained and tested our model on 1 NVIDIA V100 GPU.

6.2 *Experiment results*

Link prediction performance In this section, we report the performance of DBLP in two aspects: Firstly, we aim to ascertain DBLP's universality by comparing the performance of the most popular graph encoders with and without the *DBLP* framework, assessing its adaptability to most GNNs. Secondly, we report the performance com-

Table 1. Performance comparison of DBLP with baselines under metric Hit@20. The best performance is marked in bold and underlined, respectively.

	OGB-DDI	CiteSeer	PubMed	Facebook	KIBA
Node2Vec	23.26±2.09	47.78±1.72	39.19±1.02	24.24±3.02	59.03±1.94
VGAE	11.71±1.96	***44.04±4.86***	23.73±1.61	37.01±0.63	48.38±0.58
SEAL	30.56±3.86	40.90±3.68	28.45±3.81	40.89±5.70	59.70±1.18
LGLP	-	57.43±3.71		37.86±2.13	56.07±4.09
CFLP	86.08±1.98	68.09±1.49	44.90±2.00	55.22±1.29	60.94±1.56
GCN	37.07±5.07	55.56±1.32	21.84±3.87	*53.89±2.14*	51.71±1.91
GCN-DBLP	*58.45±01.03*	*62.45±2.69*	*39.42±1.52*	51.06±2.71	*59.04±1.36*
SAGE	53.90±4.74	53.67±2.94	39.13±4.41	45.51±3.22	52.53±2.69
SAGE-DBLP	*72.67±00.51*	*62.64±1.17*	*40.48±3.27*	*53.01±2.87*	*59.50±2.17*
GIN	64.00±0.42	30.84±7.33	18.95±4.63	39.24±1.95	*54.20±4.07*
GIN-DBLP	*70.11±0.35*	*39.16±3.05*	*23.44±2.21*	*40.32±0.56*	44.59±0.48
GAT	46.85±3.28	36.08±7.38	15.59±2.85	*27.36±1.49*	11.36±0.76
GAT-DBLP	34.09±1.79	*54.14±1.75*	13.15±4.47	21.68±04.03	*25.99*±13.53
JKNet	60.56±8.69	55.60±2.17	25.64±4.11	52.25±1.48	55.21±0.32
JKNet-DBLP	***87.83±1.44***	***72.45±1.11***	43.96±4.14	***56.63±1.65***	***61.11±0.08***

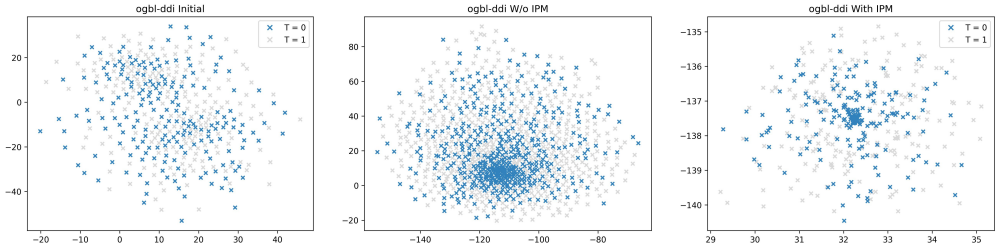

Figure 2. We first sample the 200 node pairs with mark 1 and mark 0 on the DDI dataset. Then we extract the original edge features, edge embeddings learned by vanilla GNNs, and edge embeddings learned by DBLP and use t-SNE to project the above embeddings into two dimensions (edge embeddings are derived by point-wise product of the two nodes embeddings).

parison with other advanced baselines. We show the effectiveness of using DBLP in solving the link prediction problem.

We follow the implementation provided by OGB [18] as baselines and embed them in DBLP. We used Hit@20 and Hit@50 as evaluation metrics, and the average results from five experiments were taken as the outcomes. The Hit@20 results are presented in Table 1. Additional results are shown in Appendix Table 4. LGLP on Pubmed and OGB-DDI is missing due to the out-of-memory (OOM) error. With DBLP embedded, the simple GNNs can have better performance than their original versions in most of the datasets. JKNet shows the best compatibility with DBLP and outperforms all baselines. One exception is Hit@20 in the PubMed dataset. However, we have better performance on a more challenging Hit@50 task.

For one thing, the results indicate that the performance of almost all GNNs is significantly improved by incorporating DBLP, which shows the universality of DBLP. For another, We show that a sim- ple graph encoder embedded in DBLP can have similar or better performance compared to advanced baselines. DBLP reveals the potential of GNNs in link prediction tasks by exploiting a better decoder.

Effect of edge marking Since we propose the edge marking strategy aiming to solve the node automorphic problem, we conduct experiments to show to which extent DBLP could relieve this intrinsic problem in GNNs. In Figure 3, we illustrate the result of the prediction on the test set whose marks are all 1, i.e., the set of all the error-prone node pairs. The results show that DBLP could better handle these edges than the baselines. The full results will be included in the Appendix Figure 5. Furthermore, to show marking edges in the proposed way is good enough, we conduct an ablation study using different mark strategies. This experiment result supports our original motivation that relieving the node automorphic problem is the key to having better performance in link prediction tasks.

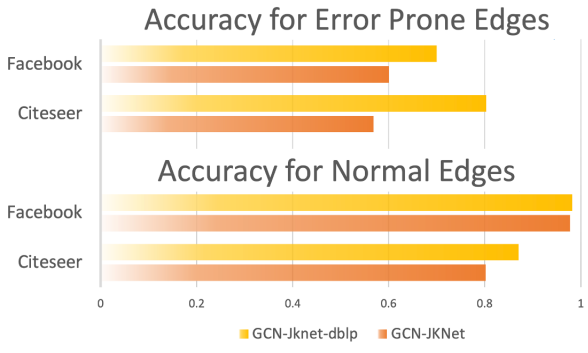

Figure 3. Accuracy results of tests performed on T=1 (normal) and T=0 (error-prone) edge datasets, respectively

Effect of the balancing weight estimator Since we observed dis-

tribution shift in all the datasets and existing works [33, 22] point out that the shift might lead to the inflated variance of potential outcomes, we conducted several experiments to show the effect of the proposed balancing weight estimator. First, we show the effect on the learned embeddings. We first sample the 200 node pairs with mark 1 and mark 0. Then we extract the original edge features (Figure 2-left), edge embeddings learned by vanilla GNNs (Figure 2-mid), and edge embeddings learned by DBLP (Figure 2-right) and use t-SNE [39] to project the above embeddings into two dimensions (edge embeddings are derived by point-wise product of the two nodes embeddings). We find several interesting results. The results of other datasets are in the Appendix.

First, we find out that with vanilla GNNs, the embeddings of error-prone node pairs (node pairs with mark 1) are scattered in space while the node pairs with mark 0 clusters. As we stated in the introduction when marks are selected based on whether the two nodes are in the same topological environment, this is presented by a distribution shift between $p(Z|t=1)$ and $p(Z|t=0)$. We find the error-prone edges distributed differently from the common edges, so it is hard to have a consistent hypothesis for a population consisting of different distributions. With balancing weight estimators, two distributions "look similar", then we can better fit the hypothesis using risk minimization. Second, we conducted an ablation study of w/o the balancing term IPM and illustrated the results in Table 2. The results show the effect of using the balancing weight estimator compared to the estimator w/o balance.

Table 2. Performance comparison on different dataset under without IPM and with IPM Scenario under metric Hit@20

Dataset	W/o IPM	With IPM
OGB-DDI	87.66±1.70	**87.83±1.44**
Citeseer	70.25±0.55	**72.45±1.11**
Pubmed	42.34±1.73	**43.96±4.14**
Facebook	53.83±2.01	**56.63±1.65**
KIBA	61.05±0.77	**61.11±0.80**

7 Conclusion

In this work, we proposed a simple yet effective framework named DBLP (**D**ecoder **B**oosting for **L**ink **P**rediction). We focused on the node automorphic problem in GNNs and tried to solve it via the design of the decoder for a GNN, whose utility is largely ignored by the community. To this end, we proposed the edge marking strategy and balancing weight estimator to enhance the decoder. As a universal framework, extensive experiments demonstrated that DBLP can adapt to different graph encoders and outperform baselines on benchmark datasets. Also, we analyzed the effect of using the edge marking strategy and the effect of the balancing weight estimator. This work suggests that an effective decoder with even a simple graph encoder can greatly improve the performance of graph learning tasks such as link prediction. We note that the difference in distributions of edge representations with different marks leads to poor performance in automorphic problems. Therefore, the use of more sophistically designed balancing weight decoders can lead to larger improvements in link prediction tasks, which can be a valuable future direction for the GNN community.

References

[1] S. Assaad, S. Zeng, C. Tao, S. Datta, N. Mehta, R. Henao, F. Li, and L. Carin. Counterfactual representation learning with balancing weights. In *International Conference on Artificial Intelligence and Statistics*, pages 1972–1980. PMLR, 2021.

[2] S. Athey, G. W. Imbens, and S. Wager. Approximate residual balancing: debiased inference of average treatment effects in high dimensions. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 80(4):597–623, 2018.

[3] P. C. Austin. An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate behavioral research*, 46(3):399–424, 2011.

[4] E. Ben-Michael, A. Feller, D. A. Hirshberg, and J. R. Zubizarreta. The balancing act in causal inference. *arXiv preprint arXiv:2110.14831*, 2021.

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

[6] D. A. Bruns-Smith and A. Feller. Outcome assumptions and duality theory for balancing weights. In *International Conference on Artificial Intelligence and Statistics*, pages 11037–11055. PMLR, 2022.

[7] L. Cai, J. Li, J. Wang, and S. Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5103–5113, 2021.

[8] B. P. Chamberlain, S. Shirobokov, E. Rossi, F. Frasca, T. Markovich, N. Hammerla, M. M. Bronstein, and M. Hansmire. Graph neural networks for link prediction with subgraph sketching. *arXiv preprint arXiv:2209.15486*, 2022.

[9] C. Cortes, Y. Mansour, and M. Mohri. Learning bounds for importance weighting. *Advances in neural information processing systems*, 23, 2010.

[10] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1): 2096–2030, 2016.

[11] A. Gretton, K. Borgwardt, M. J. Rasch, B. Scholkopf, and A. J. Smola. A kernel method for the two-sample problem. *arXiv preprint arXiv:0805.2368*, 2008.

[12] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[13] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[14] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[15] M. A. Hasan and M. J. Zaki. A survey of link prediction in social networks. *Social network data analytics*, pages 243–275, 2011.

[16] T. He, M. Heidemeyer, F. Ban, A. Cherkasov, and M. Ester. Simboost: a read-across approach for predicting drug–target binding affinities using gradient boosting machines. *Journal of cheminformatics*, 9(1):1–14, 2017.

[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[18] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[19] N. T. Huang and S. Villar. A short tutorial on the weisfeiler-lehman test and its variants. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8533–8537. IEEE, 2021.

[20] G. W. Imbens and J. M. Wooldridge. Recent developments in the econometrics of program evaluation. *Journal of economic literature*, 47(1): 5–86, 2009.

[21] F. Johansson, U. Shalit, and D. Sontag. Learning representations for counterfactual inference. In *International conference on machine learning*, pages 3020–3029. PMLR, 2016.

[22] F. D. Johansson, U. Shalit, N. Kallus, and D. Sontag. Generalization bounds and representation learning for estimation of potential outcomes and causal effects. *The Journal of Machine Learning Research*, 23(1): 7489–7538, 2022.

[23] Y. Jung, J. Tian, and E. Bareinboim. Estimating causal effects using weighting-based estimators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10186–10193, 2020.

[24] N. Kilbertus. Beyond traditional assumptions in fair machine learning. *arXiv preprint arXiv:2101.12476*, 2021.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

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

[28] J. Leskovec and J. Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

[29] S. Liu, B. Grau, I. Horrocks, and E. Kostylev. Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding. *Advances in Neural Information Processing Systems*, 34:2034–2045, 2021.

[30] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

[31] J. Peng, G. Lu, and X. Shang. A survey of network representation learning methods for link prediction in biological network. *Current pharmaceutical design*, 26(26):3076–3084, 2020.

[32] C. A. Rolling. *Estimation of Conditional Average Treatment Effects*. PhD thesis, University of Minnesota, 2014.

[33] U. Shalit, F. D. Johansson, and D. Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning*, pages 3076–3085. PMLR, 2017.

[34] N. Shibata, Y. Kajikawa, and I. Sakata. Link prediction in citation networks. *Journal of the American society for information science and technology*, 63(1):78–85, 2012.

[35] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[36] B. Srinivasan and B. Ribeiro. On the equivalence between positional node embeddings and structural graph representations. *arXiv preprint arXiv:1910.00452*, 2019.

[37] B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. Lanckriet. On the empirical estimation of integral probability metrics. 2012.

[38] J. Tang, A. Szwajda, S. Shakyawar, T. Xu, P. Hintsanen, K. Wennerberg, and T. Aittokallio. Making sense of large-scale kinase inhibitor bioactivity data sets: a comparative and integrative analysis. *Journal of Chemical Information and Modeling*, 54(3):735–743, 2014.

[39] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

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

[41] B. Weisfeiler and A. Leman. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968.

[42] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

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

[44] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

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

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

[47] J. You, J. M. Gomes-Selman, R. Ying, and J. Leskovec. Identity-aware graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10737–10745, 2021.

[48] M. Zhang and Y. Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

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

[50] Y. Zhang, A. Bellot, and M. Schaar. Learning overlapping representations for the estimation of individualized treatment effects. In *International Conference on Artificial Intelligence and Statistics*, pages 1005–1014. PMLR, 2020.

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

A Graph automorphic problem

In this section, we will illustrate the drawback of using graph neural networks in link prediction tasks mainly following the analysis in [36, 8]. To learn from a graph, we first need to learn the representations of each node. The structural representation of a node (learned by GNN) [26, 14] shows which nodes have similar roles (structural similarities) on a graph. If two nodes share the same representations, they are often seen as having similar effects in the graph. GNNs learn the representations via the relative relationship of nodes, and this kind of representation is permutation equivariance [42]. Therefore, the link probability $p(u, v)$ is the same for all u in the same topological environment. This is the result of GNN's built-in permutation equivariance.

In this paragraph, we distinguish the concepts of isomorphism and automorphism. Graph isomorphism refers to the situation where two graphs are structurally identical, meaning there exists a bijective mapping (a one-to-one correspondence) between their nodes that preserves the edge relationships. Graph automorphism, on the other hand, involves a mapping from a graph to itself, such that the graph remains unchanged. In other words, an automorphism is an isomorphism from a graph to itself. If a graph has nodes A, B, C, and D, and there is an automorphism, it becomes possible to reassign labels to these nodes in a manner that preserves the overall structure of the graph. While isomorphism compares two different graphs to check if they are structurally the same, automorphism looks at symmetries within a single graph. And our paper focuses on the challenges brought by automorphic nodes in a graph.

B Distribution Shift in Embeddings

To demonstrate the distribution shift led by the confounder, we show a simple linear example. In Figure 4-mid, points represent the data sampled from a linear model. The color (blue and red) means that the points are divided into two different groups according to some features. Given the points, we will do the estimation of the two groups. The regression results are shown in Figure 4-right. Red/Blue solid indicates the mean value of the output in a single group. However, we observe that the mean value of each group is shifted from the original linear model that generates the data points. As a result, if not adjusted for, this shift may result in inflated variance in the estimation of potential outcomes.

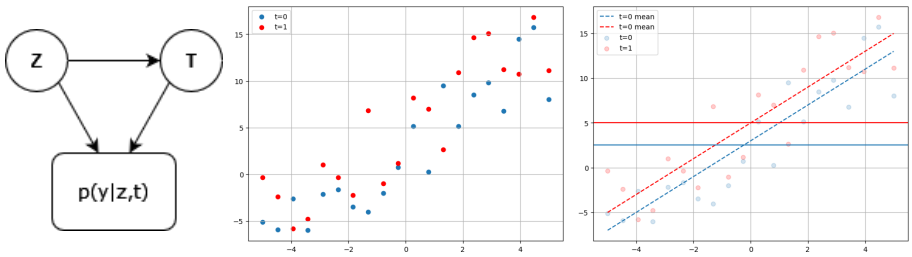

Figure 4. Left: The effect of the confounder. **Z** is the link representation obtained from vanilla GNNs, T is the indicator of whether two nodes are in the same topological environment, and $p(y|z,t)$ is the predicting result of link prediction. **Middle**: A sample dataset D with mark T. **Right**: Fit the data units given $t = 1$ and $t = 0$. The effect of the confounder manifests as a distribution difference between $p(z|t = 1)$ and $p(z|t = 0)$. The mean output in the two groups (red solid, blue solid) is different from the real mean value. If not adjusted for, this shift may result in inflated variance in the estimation of potential outcomes.

C Proof of inequality 9 and 10

Since we learn embeddings derived from GNNs, we are supposed to bound the marginal risk using the learned embeddings instead of the original features. In Lemma 1, we use the change of variables formula to transform the features into the embedding space and obtain the upper bound of the risk. Combining inequality 7 and equality 9, we will directly get the final result.

Proof (The proof is inspired by Lemma 1 in [33])

$$\int_{x\in\mathcal{X}} \ell_{f_t}(x)\left(p_{1-t}(x) - p_t^w(x)\right)dx = \int_{z\in\mathcal{Z}} \left(p_{\Phi,1-t}(z) - p_{\Phi,t}^w(z)\right)\ell_{f_t}(\Psi(z))\left|J_\Psi(z)\right|dz \tag{11}$$

$$= \int_{z\in\mathcal{Z}} \left(p_{\Phi,1-t}(z) - p_{\Phi,t}^w(z)\right)\ell_{\Phi,h_t}(z)\left|J_\Psi(z)\right|dz \tag{12}$$

$$\leq C_\Phi \cdot \sup_{\ell\in\mathcal{L}} \int_{z\in\mathcal{Z}} \ell(z)\left(p_{\Phi,1-t}(z) - p_{\Phi,t}^w(z)\right)dz. \tag{13}$$

$$= C_\Phi \cdot \mathrm{IPM}_\mathcal{L}\left(p_{\Phi,1-t}, p_{\Phi,t}^w\right). \tag{14}$$

where $|J_\Psi(z)|$ is the Jacobian determinant of the representation inverse Ψ. Equality (12) is obtained using the change of variables formula. Inequality (13) holds based on the premise that $\ell \in \mathcal{G}$. Finally, (14) is following the definition of an integral probability metric (IPM).

D Additional Experiment result

D.1 Dataset information

The statistics of the datasets are listed in table D.1, which includes the number of nodes and links of the datasets. All the datasets used in this work are publicly available.

Dataset	CORA	CiteSEER	PUBMED	FACEBOOK	KIBA
# nodes	2,708	3,327	19,717	4,039	4,267
# links	5,278	4,552	44,324	88,234	1,334,889
# validation node pairs	1,054	910	8,864	17,646	235,371
# test node pairs	2,110	1,820	17,728	35,292	229,088

In citation networks(Cora, CiteSeer, and PubMed) [44], the nodes are published papers, and features are bag-of-word vectors extracted from the corresponding paper. Links represent the citation relation between papers. The social network dataset (Facebook) is a social network constructed from friends lists from Facebook [28]. The nodes are Facebook users and links indicate the friendship relation on Facebook. And the DTI network(KIBA) [38] encompasses selectivity assays conducted on kinase proteins and their corresponding inhibitors, primarily consisting of KIBA scores. We use the threshold 12.1 following [16] to classify the KIBA dataset.

D.2 Experimental Settings

Table 3. Parameter Settings

	OGB-DDI	KIBA	facebook	pubmed	citeseer
α	2	1	$1e^{-3}$	0.1	$1e^{-3}$
lr	0.01	0.05	0.05	0.1	0.1
neg_rate	1	10	2	40	50
batch_size	1,000	1,024	65,536	12,000	65,536
num_np			5,000		
lr_scheduler			zig-zag		

In this paper, we use the hyper-parameter list above to achieve the corresponding results in Table 3. *Alpha* is the parameter in the training objective. *lr* is the learning rate. *neg_rate* is the rate at which we get the negative samples.*Num_np* is the number of links we sample to estimate the IPM term. And we choose a zig-zag learning rate scheduler for all datasets. The "zig-zag" learning rate scheduler dynamically adjusts the learning rate over 70 training steps, featuring warmup for the initial 50 steps with a gradual increase, followed by annealing over the next 20 steps with a linear decrease, and a subsequent constant stage, aiming to balance between quick convergence and fine-tuning.

D.3 Effect of edge marking

Since we propose the edge marking strategy aiming to solve the node automorphic problem, we conduct experiments to show to which extent DBLP could relieve this intrinsic problem in GNNs. In Figure 5, we illustrate the result of the prediction on the test set whose marks are all 1, i.e., the set of all the error-prone node pairs. The results show that DBLP could better handle these edges than the baselines. We noticed that in Pubmed and KIBA datasets, all the methods get almost perfect results. This is because the number of error-prone edges is small in the test set.

D.4 Additional main comparison experimental results

This section presents supplementary results for the primary experiments. We performed experiments on each of the five datasets, comparing our methods with baseline approaches. The following results showcase the performance based on AUC and Hit@50 as the evaluation metrics.

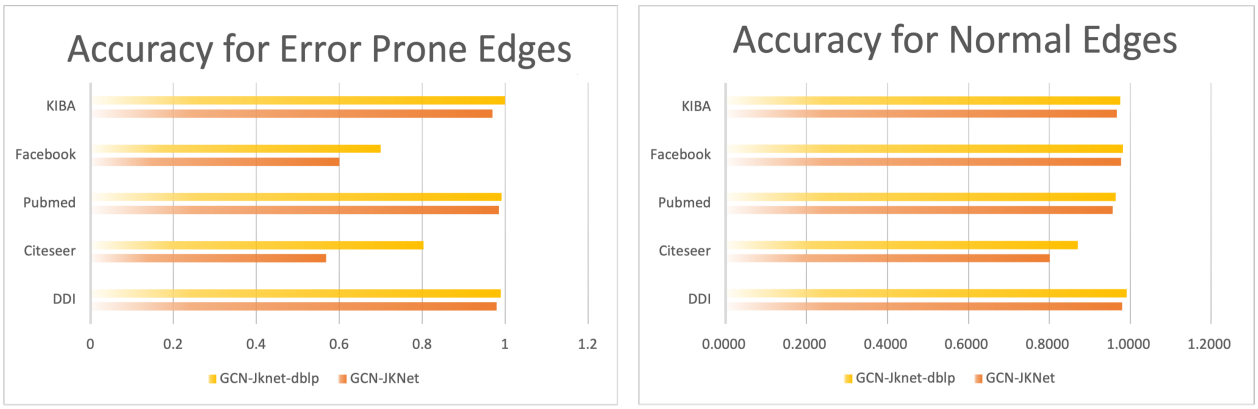

Figure 5. left: The result of the prediction on the test set whose marks are all 1, i.e., the set of all the error-prone node pairs. right: The result of the prediction on the test set whose marks are all 0, i.e., the set of all the non-isomorphic node pairs

Table 4. Performance comparison of DBLP with baselines under metric Hit@50. The best performance is marked in bold and underlined, respectively.

	OGB-DDI	CiteSeer	PubMed	Facebook	KIBA
Node2Vec	24.34±1.67	54.57±1.40	50.73±1.10	43.91±1.03	72.35±1.32
VGAE	23.00±1.66	54.68±3.15	41.98±0.31	51.36±0.93	47.03±0.14
SEAL	40.85±2.97	54.55±1.77	42.85±2.03	57.20±1.85	**77.44±1.87**
LGLP	-	57.43±3.71	–	56.22±0.49	73.31±1.17
CFLP	93.07±1.14	77.01±1.92	58.16±1.40	70.47±0.77	60.94±1.56
GCN	73.70±3.99	63.38±1.73	39.20±6.47	53.89±2.14	51.71±1.91
GCN-DBLP	70.60±0.05	71.72±1.68	57.37±0.66	66.47±0.86	78.00±1.16
SAGE	86.83±3.85	61.71±2.43	54.81±2.67	45.51±3.22	52.53±2.69
SAGE-DBLP	83.12±1.04	70.84±0.19	54.91±5.69	67.69±1.01	78.42±1.59
GIN	70.91±4.66	45.35±4.40		56.15±1.33	62.06±0.43
GIN-DBLP	75.68±0.49	50.36±0.17	34.29±1.33	54.39±0.65	63.41±0.18
GAT	60.23±3.16	51.12±1.14		47.33±0.39	24.61±1.11
GAT-DBLP	55.99±0.34	66.19±1.46	30.82±2.68	41.33±1.12	46.07±1.71
JKNet	91.48±2.41	62.26±2.10	45.16±3.18	52.25±1.48	55.21±0.32
JKNet-DBLP	**94.17±0.75**	**79.01±0.83**	**59.32±1.94**	**70.90±1.07**	76.79±0.28