Sub-graph Based Diffusion Model for Link Prediction

Hang Li
Michigan State University
lihang4@msu.edu

Wei Jin
Emory University
wei.jin@emory.edu

Geri Skenderi
Bocconi University
geri.skenderi@unibocconi.it

Harry Shomer
Michigan State University
shomerha@msu.edu

Wenzhuo Tang
Michigan State University
tangwen2@msu.edu

Wenqi Fan
The Hong Kong Polytechnic University
wenqi.fan@polyu.edu.hk

Jiliang Tang
Michigan State University
tangjili@msu.edu

Abstract

Denoising Diffusion Probabilistic Models (DDPMs) represent a contemporary class of generative models with exceptional qualities in both synthesis and maximizing the data likelihood. These models work by traversing a forward Markov Chain where data is perturbed, followed by a reverse process where a neural network learns to undo the perturbations and recover the original data. There have been increasing efforts exploring the applications of DDPMs in the graph domain. However, most of them have focused on the generative perspective. In this paper, we aim to build a novel generative model for link prediction. In particular, we treat link prediction between a pair of nodes as a conditional likelihood estimation of its enclosing sub-graph. With a dedicated design to decompose the likelihood estimation process via the Bayes' theorem, we are able to separate the estimation of sub-graph structure and its node features. Such designs allow our model to simultaneously enjoy the advantages of inductive learning and the strong generalization capability. Remarkably, comprehensive experiments across various datasets validate that our proposed method presents numerous advantages: (1) transferability across datasets without retraining, (2) promising generalization on limited training data, and (3) robustness against graph adversarial attacks. The code is available at https://github.com/hzlihang99/SGDiff.git.

1 Introduction

Graphs are ubiquitous data structures, with applications that span from social networks [1–3] to cutting-edge scientific research [4–7]. Link prediction, as one of the most fundamental tasks on graphs, plays an indispensable role in various graph applications in web-related scientific researches such as e-commerce recommendations [8, 9], social network analysis [10], and network security predictions [11]. With the recent rise of graph neural networks (GNNs), a variety of GNN-based methods have been developed, tremendously advancing the performance of link prediction [12–14].

Existing link prediction approaches can be divided into two main categories: discriminative [12, 13, 15] and generative [16, 17]. While discriminative methods are more commonly used, generative approaches, especially diffusion models, remain under-explored. Generative models are known for their superior generalization and robustness, particularly in scenarios with limited labeled data [18] or adversarial conditions [19–21], where they frequently outperform discriminative models. Generative models have increasingly demonstrated their potential in discriminative tasks across domains. In particular, diffusion models [22, 23] have shown remarkable success in image classification [24, 25]

H. Li et al., Sub-graph Based Diffusion Model for Link Prediction. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

and segmentation [26], offering robustness and precision. Inspired by these recent advancements, we explore diffusion model to link prediction, a core task in discriminative graph analysis. This direction not only shows potential for enhancing the generalization and robustness of link prediction but may also pave the way for the broader application of generative models in other graph tasks.

To fully unleash the potential of generative models for link prediction, we leverage a state-of-the-art generative framework, the diffusion model [22, 23], as the backbone for our approach. Additionally, we integrate this model with a sub-graph modeling strategy, proposing the first sub-graph-based generative diffusion model for link prediction, termed SGDIFF. This combination enjoys two key advantages. First, existing generative approaches for link prediction typically model the entire graph structure in a single step [16]. While effective, they result in excessive memory usage, especially for large graphs, due to the exponential growth in adjacency matrix size with increasing node counts. Although autoregressive modeling offers a workaround for this memory constraint [27], it has proven inefficient and high-variance [28], limiting its utility for graph learning tasks. By incorporating sub-graph modeling, SGDIFF can control the size of the sub-graph adjacency matrix, expanding the applicability of generative link prediction models to a broader range of scenarios. Furthermore, sub-graph-based modeling enables SGDIFF to address the automorphic node problem [13], a common limitation in previous global generative models, significantly enhancing SGDIFF's performance over other global-based methods. Second, prior generative models, such as VGAE [16], use both the adjacency matrix and node features to reconstruct the graph structure through a single GNN model. This design limits generalization, as node features across datasets are often incompatible, impeding transferability. Other approaches, such as GraphLP [29], rely solely on adjacency matrices but face limitations in utilizing generative strength due to inconsistencies in low-rank representations across adjacency matrices. In SGDIFF, we apply Bayesian inference to decompose the generation of graph structures and node features into sequential steps, thereby achieving effective cross-dataset transfer capabilities.

2 Related Work

2.1 Sub-graph Based Link Prediction

Due to the limitation of traditional Message Passing Neural Network (MPNN) in capturing the pairwise relations between two individual target nodes, vanilla GNNs often struggle with link prediction problems [30]. To solve this issue, manual feature enhanced models (MFEMs) like NBFNet [15], NCNC [14] and BUDDY [13] proposed a variety of methods trying to fuse the complementary structure information, e.g., heuristic features, with the message passing neural networks. On the other hand, sub-graph GNNs (SGNNs) like SEAL [12] and SUREL [31] transform link prediction into a binary sub-graph classification task and attempts to learn data-driven link prediction heuristics. Compared with fusion-based algorithms, SGNNs do not require complicated heuristic feature fusion designs and have better generous capability to different datasets. Besides, since they use sub-graphs as the sample unit, SGNNs are more flexible to inductive scenarios [32].

Next we give a formal statement about SGNNs. For a pair of nodes u, v and its enclosing sub-graph G_{uv}, SGNNs produce sub-graph representation $Y_{u,v}$ with GNNs and desired read-out functions $\mathcal{R}$. With the classifier $\mathcal{C}$, the sub-graph representation $Y_{u,v}$ is expected to classified as one if an edge (u, v) exists and zero otherwise. Commonly, node features are augmented with structural features to resolve the automorphic node problem [30]. The global heuristics can be well approximated from sub-graphs that are augmented with structural features with an approximation error that decays exponentially with the number of hops taken to construct the sub-graph [12]. By incorporating the idea of SGNNs, we aim to solve the memory print challenge for generative models on graph learning problems.

2.2 Likelihood Estimation of Diffusion Models

Diffusion models [22, 33] are a contemporary class of generative models. Through an iterative noising (forward) and denoising (reverse) Markov chain, diffusion models aim to learn the distribution of data in an explicit way [33]. Diffusion models enjoy the benefit of having a likelihood-based objective like VAEs [34] as well as high visual sample quality like GANs [35] even on high variability datasets. Recent advances in this area have also shown amazing results in text-to-image generation [36–38], audio synthesis [39, 40] and text-to-3d content creation [41, 42]. Despite being powerful generative

models, diffusion models has also been recently recognized as valid generative classifiers [43, 44]. As using the variational lower bound (VLB) of the log-likelihood as the object function, a well-trained diffusion models could provide accurate estimations to the probability of samples within the data distribution [25]. Furthermore, by incorporating class information as the condition input during the training, the diffusion model can be used to compute class-conditional likelihoods $p_\theta(\mathbf{x}|\mathbf{y})$. Then, by selecting an appropriate prior distribution $p(\mathbf{y})$ and applying Bayes' theorem, predicted class probabilities $p_\theta(\mathbf{y}|\mathbf{x})$ can be easily calculated. Compared to discriminative models, generative classifiers have been shown to generalize better, be more robust, and be better calibrated [20, 21]. In this work, we seek to develop diffusion models for solving discriminative graph problems.

3 Method

Although there are recent works on applying diffusion models for problems on graphs [45, 46], most of them focus on its generative perspective. The usage of the likelihood score of diffusion models to graph problems is relatively underexplored. Therefore, in this work, we take one of the most fundamental problems on graphs (i.e., link prediction) as an example to demonstrate the effectiveness of diffusion models with SGNNs for problems with graph data. Note that our algorithm can be easily extended to other graph problems like node or graph classification and we leave it as one future work. Next, we will first define notations and introduce an overview design of our algorithm SGDIFF. then, we present details about the link likelihood score estimation with the combination of structure and feature diffusion models.

3.1 Notations

In the following, we formally define the notations used in this work. Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of n nodes (vertices) $\mathcal{V}$ and e links (edges), respectively. Let $S = (\mathcal{V}_S \subseteq \mathcal{V}, \mathcal{E}_S \subseteq \mathcal{E})$ be a node-induced sub-graph of G satisfying $(u, v) \in \mathcal{E}_S$ iff $(u, v) \in \mathcal{E}$ for any $u, v \in \mathcal{V}_S$. We use $S^k_{uv} = (\mathcal{V}_{uv}, \mathcal{E}_{uv})$ to denote a k-hop sub-graph enclosing the link (u, v), where $\mathcal{V}_{uv}$ is the union of the k-hop neighbors of u and v and $\mathcal{E}_{uv}$ is the union of the links that can be reached by a k-hop walk originating at u and v. The given features of nodes $\mathcal{V}_{uv}$ are represented by $\mathbf{X}_{uv}$ and the adjacency matrix of S^k_{uv} is $\mathbf{A}_{uv}$. The probability of link (u, v) existing is indicated by $p(y_{uv} = 1)$ and our goal is to estimate $p(y_{uv} = 1|\mathbf{X}_{uv}, \mathbf{A}_{uv})$ with likelihood score generated by diffusion models. As following parts process each sub-graph with the same process, we will omit the subscripts u and v for convenience.

3.2 Design Overview

As mentioned in Section 2.2, by using a prior distribution $p(y)$, the categorical probability $p(y|x)$ can be estimated by applying Bayes' theorem over the class-conditional likelihood $p(x|y)$. However, unlike applying diffusion models to a single input like image or text, performing diffusion on graph data involves two different but related inputs, i.e., node feature ($\mathbf{X}$) and adjacency matrix ($\mathbf{A}$). Therefore, we need dedicated designs to decompose $p(y|\mathbf{A}, \mathbf{X})$. To break the generalization limitation of existing generative approaches and take advantage of the inductive learning capability of SGNNs, we propose the following formulation:

$$p(y|S) = p(y|\mathbf{A}, \mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{A}, y) \cdot p(\mathbf{A}|y) \cdot p(y)}{\Sigma_{c\in\{0,1\}} p(\mathbf{A}, \mathbf{X}|y = c) \cdot p(y = c)} \tag{1}$$

where $p(y)$ is the prior distribution of node-pair's connectivity over the graph. $p(\mathbf{A}|y)$ denotes the graph structure probability given the condition of nodes u and v being connected. $p(\mathbf{X}|\mathbf{A}, y)$ represents the feature probability that is conditioned on the observed structure and connectivity. An overview of our framework is shown in Fig. 1. By splitting the generation of graph structure and node features into sequential steps, SGDIFF can be used for various link prediction settings, with or without node features. More importantly, since it integrates the idea of SGNNs, the structure diffusion model of SGDIFF can easily be transferred across datasets without involving any re-training procedure. In our experiments (Section 4.2), we find that small datasets can benefit from the learnt knowledge of larger datasets. Meanwhile, since SGDIFF has the feature component, we can design an independent feature diffusion model for each dataset to model diverse node features. In other words, the structure

diffusion of SGDIFF provides a shareable basis for different datasets and the feature diffusion acts as an adjusting head which adapts the whole model to specific datasets. Lastly, generative models are well-known for their robustness against adversarial attacks [24, 47]. By modeling the likelihood scores of both the graph structure and node features, we expect that SGDIFF should be more robust against graph adversarial attacks as compared to existing discriminative approaches. We empirically verify this assumption in Section 4.4. Next, we detail the major components of SGDIFF.

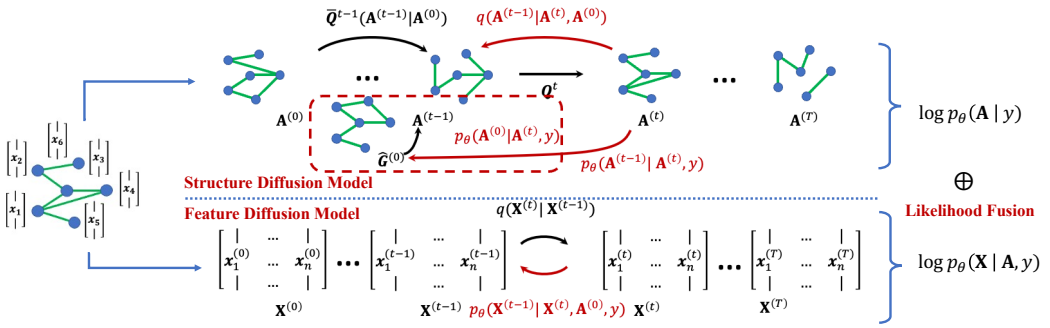


Figure 1: An overview of our proposed framework. $\mathbf{Q}^t$ and q are diffusion kernels for structure and feature diffusion models, respectively. The calculation of log-likelihood scores $\log P_\theta(\mathbf{A}|y)$ and $\log P_\phi(\mathbf{X}|\mathbf{A}, y)$ is based on fitted denoising models, $p_\phi(\mathbf{A}^{(0)}|\mathbf{A}^{(t)}, y)$ and $p_\epsilon(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t-)}, \mathbf{A}^{(0)}, y)$, respectively.

3.3 Structure Diffusion Model

The estimation of $p(\mathbf{A}|y)$ with diffusion models involves the discrete input $\mathbf{A}$. Following DiGress [45], we use discrete status transition noise [48] to maintain both the sparsity of the adjacency matrix as well as graph theoretic notions such as connectivity during the diffusion process. In addition to the adjacency matrix $\mathbf{A}$, we further include the orbit features of each node $\mathbf{X}'$ in the diffusion process. This is because we are estimating the likelihood score of a sub-graph S under the connection condition y of the sub-graph's center nodes u and v. The orbit features indicate the relative distance of each node toward the center nodes thereby better distinguishing the sub-graphs with similar adjacency matrix but different center node locations. There are many choices for the orbit features. We use the Double Radius Node Labeling (DRNL) [30] as we empirically find that it performs well on most of the datasets. To be concise, we define the forward of the structure diffusion model as:

$$\begin{aligned} q(S^{(t)}|S^{(t-1)}) &= (\mathbf{A}^{(t-1)}\mathbf{Q}_A^t,\ \mathbf{X}'^{(t-1)}\mathbf{Q}_X^t), \\ q(S^{(t)}|S^{(0)}) &= (\mathbf{A}^{(0)}\bar{\mathbf{Q}}_A^t,\ \mathbf{X}'^{(0)}\bar{\mathbf{Q}}_X^t), \end{aligned} \tag{2}$$

where $\mathbf{Q}_A^t$ and $\mathbf{Q}_X^t$ are the transition probability matrices at the t-th step for discrete edge and node features. $\bar{\mathbf{Q}}_A^t = \prod_{i=1}^t \mathbf{Q}_A^i$ and $\bar{\mathbf{Q}}_X^t = \prod_{i=1}^t \mathbf{Q}_X^i$. The backward process can be stated as:

$$\begin{aligned} p_\theta(S^{(t-1)}|S^{(t)}, y) = (&\mathbf{A}^{(t)}(\mathbf{Q}_A^t)' \odot \phi_\theta(\mathbf{A}^{(t)}, y, t)\bar{\mathbf{Q}}_A^{t-1}, \\ &\mathbf{X}^{(t)}(\mathbf{Q}_X^t)' \odot \phi_\theta(\mathbf{X}^{(t)}, y, t)\bar{\mathbf{Q}}_X^{t-1}) \end{aligned} \tag{3}$$

where $\odot$ denotes a element-wise product and $(\mathbf{Q}_A^t)'$ and $(\mathbf{Q}_X^t)'$ are the transpose of $\mathbf{Q}_A^t$ and $\mathbf{Q}_X^t$, respectively. ϕ_θ is the denoising diffusion model, which takes timestep t, t-th step noisy sample $S^{(t)}$ and connection condition y as inputs. It further outputs the distribution of categorical features in the clean graph $S^{(0)}$. We use a transformer-based neural network for ϕ_θ and train it following prior work [45]. The conditional information y is concatenated to every node and edge feature during the pre-processing step.

The likelihood score of sub-graph S can then be estimated by applying the evidence lower bound (ELBO) to the integration result of the joint probability $p_\theta(S^{(0:T)}) = (\prod_{i=0}^T p_\theta(S^{(t-1)}|S^{(t)})) \cdot$

$P(S^{(T)})$ over different trajectories $S^{(1:T)}$. We calculate $p_\theta(S|y)$ as follows:

$$\log p_\theta(S|y) \geq \log p(n_S|y) + \underbrace{D_{KL}[q(S^{(T)}|S)||q_X(n_S|y) \times q_E(n_S|y)]}_{\text{Prior loss}} + \underbrace{\sum_{t=2}^{T} L_t(S|y)}_{\text{Diffusion loss}} + \underbrace{\mathbb{E}_{q(S^{(1)}|S)}[\log p_\theta(S|S^{(1)}, y)]}_{\text{Reconstruction loss}} \tag{4}$$

with

$$L_t(S|y) = \mathbb{E}_{q(S^{(t)}|S)}[D_{KL}[q(S^{(t-1)}|S^{(t)}, S)||p_\theta(S^{(t-1)}|S^{(t)}, y)]] \tag{5}$$

where $\log p(n_S|y)$ is the probability of sub-graph size n_S under condition y. We note that since most link prediction problems are on undirected graphs, the above diffusion process is only applied to the upper-triangular of $\mathbf{A}$. Additionally, as the (u, v)-th element of adjacency matrix $\mathbf{A}^{(0)}$ will be unknown during test, we make $\mathbf{A}^{(0)}_{u,v} = 0$ during pre-processing.

3.4 Node Diffusion Model

Since node features are typically continuous variables, we estimate $p_\theta(\mathbf{X}|\mathbf{A}, y)$ using Gaussian noise as its diffusion kernel in a similar manner to DDPM [22]. Similar to Eq. (2), the forward process of feature diffusion model can be written as:

$$q(\mathbf{X}^{(t)}|\mathbf{X}^{(t-1)}) = \mathcal{N}(\mathbf{X}^{(t)}; \sqrt{1-\beta_t}\mathbf{X}^{(t-1)}, \beta_t\mathbf{I}), \tag{6}$$

$$q(\mathbf{X}^{(t)}|\mathbf{X}^{(0)}) = \mathcal{N}(\mathbf{X}^{(t)}; \sqrt{1-\bar{\alpha}_t}\mathbf{X}^{(0)}, \bar{\alpha}_t\mathbf{I}), \tag{7}$$

where β_t is the variance schedule, which transitions from 0 to 1, and $\bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_t)$. The reverse process under condition c is defined as:

$$p_\theta(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t)}, c) = \mathcal{N}(\mathbf{X}^{(t-1)}; \tilde{\mu}_t, \tilde{\beta}_t)$$
$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(\mathbf{X}^{(t)} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{X}^{(t)}, t, c),\ \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t \tag{8}$$

where $\alpha_t = 1 - \beta_t$. $\epsilon_\theta(\mathbf{X}^{(t)}, t, y)$ is our denoising diffusion models and $\tilde{\beta}_t$ is only correlated with the β_t. Through applying the ELBO trick over the integral of joint distribution $q(\mathbf{X}^{(0:T)}|y)$, we can write the conditioned log-likelihood score of node features as:

$$\begin{aligned}
&\log p_\theta(\mathbf{X}|c) \geq \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{X}^{(0:T)}, c)}{q(\mathbf{X}^{(1:T)}|\mathbf{X}^{(0)})}\right] \\
&= \mathbb{E}_q[\underbrace{D_{KL}(q(\mathbf{X}^{(T)}|\mathbf{X}^{(0)})||p_\theta(\mathbf{X}^{(T)}))}_{\text{Prior loss}} - \underbrace{\log p_\theta(\mathbf{X}^{(0)}|\mathbf{X}^{(1)}, c)}_{\text{Reconstruction loss}} \\
&+ \underbrace{\Sigma_{t=2}^{T} D_{KL}(q(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t)}, \mathbf{X}^{(0)})||p_\theta(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t)}, c))}_{\text{Diffusion loss}}],
\end{aligned} \tag{9}$$

where the prior and reconstruction losses are nullified as their value is much smaller than the diffusion loss. To calculate $D_{KL}(q|p_\theta)$, we use the simplified form proposed by Ho et al. [22] producing the final expression:

$$-\mathbb{E}_{t,\epsilon}[||\epsilon - \epsilon_\theta(\mathbf{X}^{(t)}, c)||^2] \quad \text{with} \quad \mathbf{X}^{(t)} = \sqrt{\bar{\alpha}_t}\mathbf{X}^{(0)} + \sqrt{1-\bar{\alpha}_t}\epsilon, \tag{10}$$

where $\epsilon \sim \mathcal{N}(0, 1)$. The denoising model ϵ_θ takes as input ϵ_θ, the noisy samples at step t, and the given condition $c = (\mathbf{A}, y)$ and outputs the noise at step t. Since the condition c includes both the adjacency matrix $\mathbf{A}$ and the connection condition y, the predictions are made at node-level. Lastly, we use GCN [49] to model ϵ_θ:

$$\epsilon_\theta(\mathbf{X}^{(t)}, y, t) = \hat{\mathbf{A}}(\sigma(\hat{\mathbf{A}}\mathbf{X}^{(\mathbf{t})}\mathbf{W}_0))\mathbf{W}_1, \tag{11}$$

where σ is an activation function, and $\mathbf{W}_0$ and $\mathbf{W}_1$ are the learnable parameters. $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ with $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ and $\tilde{\mathbf{D}}_{ii} = \Sigma_j\tilde{\mathbf{A}}_{ij}$. The connection status condition y is concatenated to each node feature during the feature pre-processing.

3.5 Connection Probability Estimation

With the estimated log-likelihood scores of the sample's graph structure $\log p(\mathbf{A}|y)$ and node features $\log p(x|\mathbf{A}, y)$, we can estimate the connection probability $P(y|\mathbf{A}, \mathbf{X})$ via Eq. (1). However, directly taking the summation over those two components will be sub-optimal, as the scale of values returned by the two diffusion models are different. Furthermore, the weighting values of the diffusion loss are neglected during the loss calculation for simplification purposes. Because of this, we use the additional learnable parameter set $\{\eta_1, \eta_2, \delta\}$ to flexibly adjust each component during the fusion. The final connection probability calculation can be written as:

$$P(y|\mathbf{A}, \mathbf{X}) = \text{softmax}_y(\log P(\mathbf{A}, \mathbf{X}|y) + \log P(y)), \tag{12}$$

with

$$\log P(\mathbf{A}, \mathbf{X}|y) = \eta_1 \cdot \log \hat{\mathbf{P}}(\mathbf{X}|\mathbf{A}, y) + \eta_2 \cdot \log \hat{\mathbf{P}}(\mathbf{A}|y) + \delta, \tag{13}$$

where $\{\eta_1, \eta_2, \delta\}$ are optimized via gradient descent over the cross entropy loss between true links and the predicted connection probability $\hat{P}(y|\mathbf{A}, \mathbf{X})$. Please check Appendix A.1 for more details.

4 Experiments

In this section we conduct comprehensive experiments to validate the advantages of the proposed framework SGDIFF. In particular, we aim to answer the following questions: **RQ1:** Does SGDIFF enjoy the advantages of both SGNNs and generative models in solving cross-dataset link prediction? **RQ2:** How is the generalization capability of SGDIFF when faced with the challenge of train size limitation? **RQ3:** Does SGDIFF show its strength in robustness against the adversarial attacks on graph structure? Before presenting our experimental results and observations, we first introduce our general experimental settings.

4.1 General Experimental Settings

To demonstrate the effectiveness of SGDIFF, we choose seven representative link prediction algorithms as our baselines. Specifically, our baselines include GCN [49], GAT [50], SAGE [51], NeoGNN [52], VGAE [16], and SEAL [12]. To be noticed, we select VGAE because it is a representative generative model for graph learning. And we collect SEAL and NeoGNNs since both of them are the effective link prediction models sharing the similar sub-graph learning ideas with SGDIFF. For the other baseline methods, we collect them following the prior studies on link prediction tasks [53]. More details about implementations of the baseline and SGDIFF can be found in Appendix A.3. We conduct experiments on six real-world graph datasets, including 3 citation networks: Cora, Citeseer and Pubmed [54] and 3 miscellaneous networks: USAir, NS and Router. The details about each dataset are shown in Table 3. Following prior works [12, 16], we split the existing links in each graph into train/valid/test with the percentages 80%/5%/15%. For evaluation, we randomly sample the same amount of unconnected node pairs as the negative samples. The evaluation metrics used in our experiment are AUC, Average Precision(AP) and Hit@100. All experiments are run over 10 seeds and we report both the mean values of each metric.

4.2 Performance on Cross-data Transferability

In this section, we aim to answer the first question about the cross-data transferablity of SGDIFF. As discussed in Section 3.2, one potential advantage of the structure diffusion model of SGDIFF is the potential to be transferred across datasets without re-training. To validate this advantage, we perform a zero-shot cross dataset transferring experiment, where the model is trained with a source dataset and is tested on other target datasets. As the node features among different datasets are incompatible with each other, we do not add node features for SGDIFF and SEAL. For VGAE, as the training and test graphs have different node numbers, we do not use node-id as input features for VGAE. Instead, we follow prior work [55], which randomly projects the node features into the same dimension and then performs row normalization. For graphs without node features, we draw random vectors from the Gaussian distribution and use it as node features. We test the transferability by setting each of the six graph datasets as the source for training and test the train model overall all six graphs. We report the perforamnce of each model from two perspectives, **Source** and **Target**. To be specific, **Source** averages the test performance on different target datasets of one model trained with one fixed source dataset. And **Target** averages the test performances on one fixed target dataset of six models trained

Table 1: Performance on the cross-data link prediction tasks. The **Rank** displays the average rank of models in different source and target datasets. The best rank value is marked with *, the second best is marked with ‡, and the third best is marked with †.

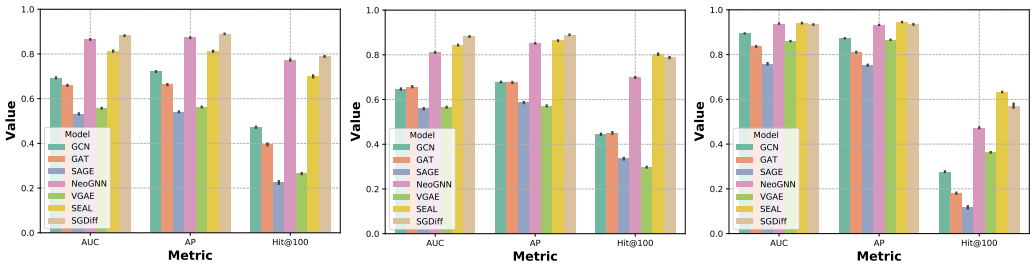
Model	Rank ↓	Source						Target					
		Cora	Citeseer	Pubmed	Router	NS	USAir	Cora	Citeseer	Pubmed	Router	NS	USAir
AUC ↑													
GCN	5.3	82.25	75.38	83.43	74.87	50.84	68.71	70.52	71.12	71.68	55.75	85.02	81.39
GAT	4.7	79.31	77.89	80.63	75.12	67.59	68.15	74.76	74.19	73.99	50.43	89.32	85.99
SAGE	4.0	82.41	81.28	84.17	76.30	70.63	68.14	73.33	73.01	79.54	65.45	85.81	85.78
NeoGNN	3.9†	80.26	74.49	85.99	81.09	63.83	78.21	82.67	78.05	82.48	53.37	90.15	77.16
VGAE	6.8	71.89	73.58	75.46	64.56	62.34	65.25	67.24	66.27	69.18	54.04	80.72	75.64
SEAL	1.9‡	89.09	84.55	88.84	87.98	86.55	75.03	84.88	83.14	80.02	78.45	94.86	90.69
SGDiff	1.4*	85.94	90.49	92.07	87.99	87.98	83.80	86.93	86.23	90.78	88.62	91.62	84.09
AP ↑													
GCN	5.5	83.75	79.37	85.94	74.88	58.78	69.11	73.67	73.00	74.37	62.85	85.83	82.11
GAT	4.4	81.44	80.68	83.57	80.10	72.31	72.21	78.27	79.40	76.66	58.39	92.88	84.72
SAGE	4.6	83.14	81.59	85.70	77.03	69.85	66.24	74.13	74.69	77.87	65.76	87.47	83.63
NeoGNN	3.3†	85.50	80.36	89.71	85.35	72.63	80.79	86.38	83.19	86.63	65.55	93.57	79.01
VGAE	6.9	71.21	74.21	76.21	64.81	61.52	65.07	67.56	66.01	69.38	57.23	78.35	74.49
SEAL	1.8‡	90.54	87.06	91.51	89.02	88.16	78.76	87.50	86.89	82.25	81.32	95.92	91.16
SGDiff	1.5*	87.79	91.36	92.65	86.54	88.66	85.17	87.93	86.98	90.90	89.62	91.56	85.19
Hit@100 ↑													
GCN	5.3	71.26	63.37	72.54	56.10	35.33	48.31	52.78	55.62	31.88	36.34	85.26	85.04
GAT	4.6	64.85	63.18	68.19	60.84	51.22	49.22	58.91	63.11	27.92	27.36	87.73	92.47
SAGE	4.8	68.06	65.35	73.38	57.45	48.49	43.62	54.19	58.28	26.50	39.79	85.16	92.44
NeoGNN	3.3†	72.09	65.13	77.77	70.12	52.23	63.56	74.96	69.35	48.35	41.76	89.78	76.70
VGAE	6.8	54.08	55.38	60.89	42.81	39.45	43.30	43.00	43.39	22.12	25.92	82.15	79.34
SEAL	1.8‡	78.42	74.46	82.31	77.71	73.46	59.26	77.16	76.14	39.09	62.92	94.96	95.35
SGDiff	1.5*	74.99	83.37	86.02	76.82	75.01	74.51	78.95	80.08	50.43	78.67	93.84	88.75

by different source datasets. Overall, the cross-data transferring results are shown as Table 1, and we calculate the average rank of each model under different source and target dataset as the indicator for model's cross-data transferability. Detailed performance of models can be found in Appendix. A.5.

From Table 1 we have the following observations: (1) the link prediction performance of all baseline models is always much better than the random guess, e.g., AUC greater than 50%. This fact indicates that different graph datasets actually share some similar structure patterns for link prediction task and it will be possible to develop a unified link prediction model across different graph datasets. (2) SGDIFF achieves best average ranking performance across different transferring scenarios, which supports our claim that SGDIFF takes the advantages of both SGNNs and generative model in generalization. In addition, we find that SGDIFF consistently benefits from transferring from a larger source dataset. For instance, when trained with different source datasets, SGDIFF receives the best performance with Pubmed, which indicates that fitting on Pubmed tends to produce the best transferability. This observation is followed by the other five source datasets, Cora, Router, Citeseer, NS and USAir, where USAir is the smallest. This phenomenon encourages us to explore an unified pre-training framework for link prediction as one promising future direction. (3) It is important to note that SGDIFF is outperformed by other baseline methods, such as SEAL, in certain transfer scenarios—most notably when the target is USAir. In Appendix A.4, we provide a comprehensive case study examining structural feature distribution shifts between the source and target datasets. Our findings confirm that SGDIFF faces challenges when the distribution shift between the source and target datasets is substantial.

4.3 Performance with Train Size Constraint

In this subsection, we further explore the generalization capability of SGDIFF and answer the second question by applying low availability limitations on the size of training set. In this setting, we shrink the training sample size of each dataset to only 1%. To make the result comparable with other experiments, only the size of training data is decreased while keeping the validation and test sets. Furthermore, we use random sampling to create smaller training sets. To be noticed, as we intend to explore the performance change caused by decreasing training sample size, but not the completeness of the graph, we do not mask the remaining 99% training edges from the original graph during the enclosing sub-graph generation process. We only control the number of sub-graphs used for training SEAL and SGDIFF. And for VGAE, we use the same adjacency matrix as other experiments but we mask 99% of the cross-entropy loss over the adjacency matrix during the back propagation. The performance of SGDIFF and baseline models are shown in Figure 2. We observe that as we limit



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

(d) Router **(e)** NS **(f)** USAir

Figure 2: Model Performance on Cora / Citeseer / Pubmed / Router / NS / USAir datasets under the limited (1%) training set scenario.

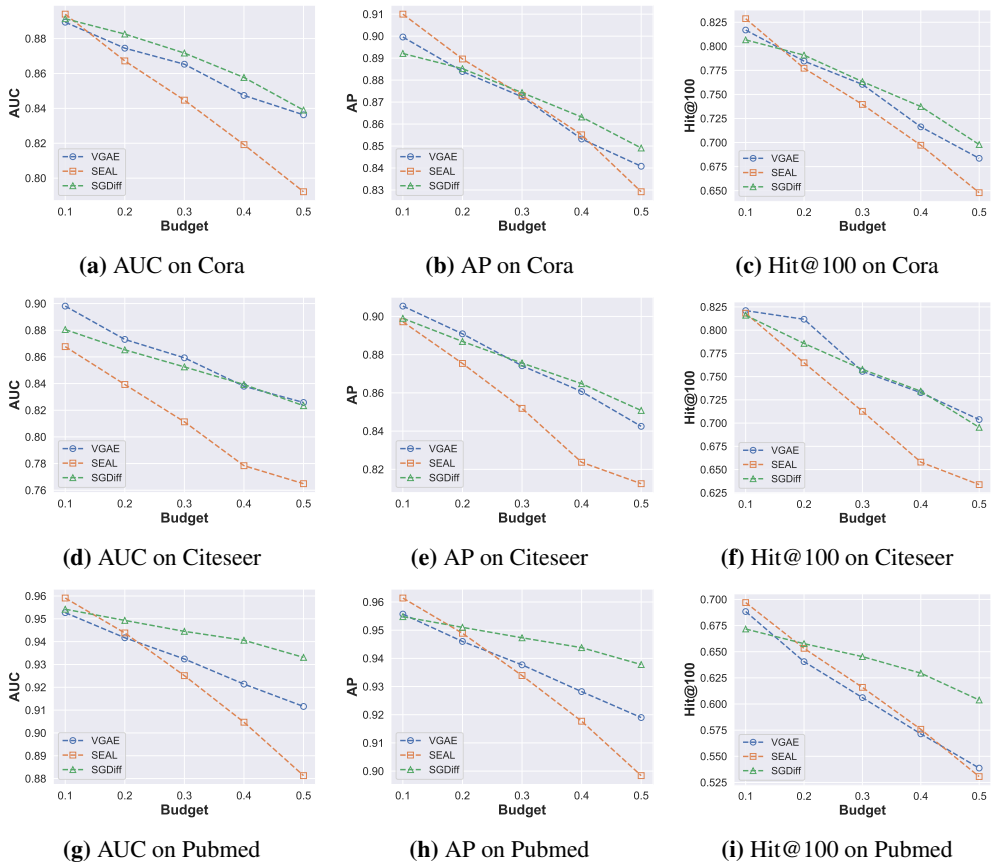
Table 2: Robustness against random flip (RF) and node embedding (NE) Attacks on Cora / Citeseer / Pubmed. The **Rank** displays the average rank of models in different settings. The best rank value is marked with *, the second best is marked with ‡, and the third best is marked with †.

Model	Rank ↓	Cora				Citeseer				Pubmed			
		RF		EA		RF		EA		RF		EA	
		25%	50%	25%	50%	25%	50%	25%	50%	25%	50%	25%	50%
AUC ↑													
GCN	5.4	84.66	82.29	84.23	84.23	80.55	78.63	80.93	79.60	95.04	93.28	94.10	93.10
GAT	5.2	86.90	83.60	87.53	84.23	84.19	81.31	84.90	82.64	88.92	84.72	88.71	83.86
SAGE	2.7‡	89.63	86.75	87.97	85.54	86.16	84.53	86.87	85.89	94.88	92.43	94.54	91.56
NeoGNN	3.7	89.15	86.59	87.51	84.92	82.59	80.96	83.00	82.19	94.40	93.68	95.30	93.21
VGAE	3.0†	88.87	86.61	87.38	85.05	90.07	87.46	89.96	87.45	94.35	92.42	94.24	92.27
SEAL	6.2	86.84	83.49	87.03	82.84	82.66	78.55	82.70	79.26	91.90	87.39	90.71	86.07
SGDiff	1.8*	88.37	86.88	88.00	85.67	87.13	85.43	86.51	84.82	95.09	94.62	94.76	93.99
AP ↑													
GCN	5.6	85.79	83.37	85.11	85.11	81.33	80.06	81.74	81.36	95.18	93.51	95.25	93.47
GAT	6.1	87.33	84.00	88.41	84.79	85.90	83.32	86.68	84.78	89.28	85.28	89.15	84.40
SAGE	3.1†	89.79	87.98	90.52	89.08	87.25	85.77	87.86	86.99	95.27	93.08	95.04	92.51
NeoGNN	2.5*	90.84	89.25	91.05	89.82	85.77	84.53	86.16	85.36	95.59	94.06	95.54	93.83
VGAE	2.7‡	90.12	88.18	88.75	86.72	91.09	88.95	91.21	89.07	94.94	93.52	94.90	93.44
SEAL	4.9	89.34	86.89	89.72	86.65	86.83	83.73	86.96	84.56	93.15	89.88	92.40	89.28
SGDiff	3.2	88.35	87.10	88.09	86.04	89.17	87.79	88.65	87.46	95.12	94.62	94.97	94.47
Hit@100 ↑													
GCN	6.2	72.73	67.40	71.88	71.88	68.97	64.16	68.78	66.21	64.06	56.44	64.13	57.15
GAT	5.8	77.48	70.34	78.52	71.51	73.73	69.83	74.83	71.01	43.32	33.70	42.72	32.15
SAGE	2.3‡	82.28	78.18	82.37	80.25	77.08	73.45	77.67	75.61	67.65	57.80	66.67	56.47
NeoGNN	3.9	80.44	77.37	81.04	78.27	70.00	69.08	70.52	69.97	63.98	58.61	63.83	59.45
VGAE	2.2*	80.49	76.94	78.59	73.87	84.30	79.50	82.92	79.13	67.58	62.81	68.23	62.86
SEAL	5.1	78.58	72.65	78.80	72.11	74.98	68.22	74.43	68.78	63.87	56.65	63.73	58.49
SGDiff	2.5†	79.55	77.10	78.82	74.53	79.49	76.08	78.11	74.48	65.63	63.27	65.51	64.42

the size of the training data, SGDIFF suffers less performance degradation compared with the other baseline models. This validates the advantages of SGDIFF when little training data is used.

4.4 Performance in Terms of Robustness

In this section, we answer the third question by demonstrating the robustness of SGDIFF. To empirically test this, we adopt three common adversarial attack baselines for link predictions, i.e., random flipping (RF), Embedding Attack (EA) [56] and DICE [57]. To be noticed, as most adversarial attack are proposed for graphs with node features, we conduct the following experiments with three citation networks: Cora, Citeseer and Pubmed. For DICE, as it required node label information as



(a) AUC on Cora **(b)** AP on Cora **(c)** Hit@100 on Cora

(d) AUC on Citeseer **(e)** AP on Citeseer **(f)** Hit@100 on Citeseer

(g) AUC on Pubmed **(h)** AP on Pubmed **(i)** Hit@100 on Pubmed

Figure 3: Models' robustness against the DICE attack on Cora / Citeseer / Pubmed datasets.

the supervised signal to train a surrogate model during the attack process, we apply it only with two baseline models, e.g., VGAE, SEAL, and SGDIFF. The implementation of each attack uses the open source graph attack tool library, DeepRobust [58]. For each type of attacks, we substitute the clean adjacency matrix with an attacked one during the inference process. We then compare each model's performance with different adversarial budgets against its clean performance. During the model training phrase, the node feature will be used if it is available on that dataset. Otherwise, the one-hot node id feature will be used as node features for VGAE. The complete performance of all models against RF and EA is shown in Table 2 and the robustness towards DICE is presented as Figure 3.

From Table 2 and Figure 3, we have the following observations. (1) Generative models, e.g., VGAE and SGDIFF, are more robust compared to most of the discriminative based models. This observation is consistent with the robustness conclusion in prior researches on generative and discriminative methods [19]. (2) SGDIFF achieves dominating leading positions in the relative performance degradation percentage, while keeping leading in the absolute metric values on most of datasets. This phenomenon demonstrates the robustness of SGDIFF. (3) Although SGDIFF does not shown its steady leading position on some datasets like citeseer, but its performance degradation is relatively much smaller than other baselines. And we have reason to believe that SGDIFF will be more robust when face stronger perturbations.

5 Conclusion

In this paper, we aim to adopt the diffusion model to the link prediction problem. With extensive experiments over the model's generalization, robustness and cross-data transfer capability, we successfully demonstrate the advantages of applying generative models toward graph learning tasks. Additionally, through the findings on the exchangeable structure components over datasets, we show the potential of our proposed framework to be an unified pre-training framework for link prediction in the future.

Acknowledgements

Geri Skenderi is funded by the European Union through a Next Generation EU - MIUR PRIN PNRR 2022 grant. The views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Lei Tang and Huan Liu. Graph mining applications to social network analysis. *Managing and mining graph data*, pages 487–513, 2010. 1

[2] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2110–2119, 2018.

[3] Hao Wang, Tong Xu, Qi Liu, Defu Lian, Enhong Chen, Dongfang Du, Han Wu, and Wen Su. Mcne: An end-to-end framework for learning multiple conditional network representations of social network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1064–1072, 2019. 1

[4] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31, 2018. 1

[5] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4(3): 279–287, 2022.

[6] Hao Wang, Jiaxin Yang, and Jianrong Wang. Leverage large-scale biological networks to decipher the genetic basis of human diseases using machine learning. *Artificial Neural Networks*, pages 229–248, 2021.

[7] Hongzhi Wen, Jiayuan Ding, Wei Jin, Yiqi Wang, Yuying Xie, and Jiliang Tang. Graph neural networks for multimodal single-cell data integration. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 4153–4163, 2022. 1

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

[9] 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*, pages 112–121, 2022. 1

[10] Mohammad Al Hasan and Mohammed J Zaki. A survey of link prediction in social networks. *Social network data analytics*, pages 243–275, 2011. 1

[11] Aaron Scott Pope, Daniel R Tauritz, and Melissa Turcotte. Automated design of tailored link prediction heuristics for applications in enterprise network security. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1634–1642, 2019. 1

[12] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018. 1, 2, 6, 14

[13] 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. 1, 2

[14] Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. *arXiv preprint arXiv:2302.00890*, 2023. 1, 2

[15] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34:29476–29490, 2021. 1, 2

[16] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016. 1, 2, 6, 14

[17] Jinyin Chen, Xiang Lin, Chenyu Jia, Yuwei Li, Yangyang Wu, Haibin Zheng, and Yi Liu. Generative dynamic link prediction. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(12), 2019. 1

[18] Milad Abdollahzadeh, Touba Malekzadeh, Christopher TH Teo, Keshigeyan Chandrasegaran, Guimeng Liu, and Ngai-Man Cheung. A survey on generative modeling with limited data, few shots, and zero shot. *arXiv preprint arXiv:2307.14397*, 2023. 1

[19] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14, 2001. 1, 9

[20] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019. 3

[21] Huanran Chen, Yinpeng Dong, Zhengyi Wang, Xiao Yang, Chengqi Duan, Hang Su, and Jun Zhu. Robust classification via a single diffusion model. *arXiv preprint arXiv:2305.15241*, 2023. 1, 3

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 2, 5

[23] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 1, 2

[24] Roland S Zimmermann, Lukas Schott, Yang Song, Benjamin A Dunn, and David A Klindt. Score-based generative classifiers. *arXiv preprint arXiv:2110.00473*, 2021. 1, 4, 16

[25] Alexander C Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier. *arXiv preprint arXiv:2303.16203*, 2023. 1, 3

[26] Tomer Amit, Tal Shaharbany, Eliya Nachmani, and Lior Wolf. Segdiff: Image segmentation with diffusion probabilistic models. *arXiv preprint arXiv:2112.00390*, 2021. 2

[27] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018. 2

[28] Jie Bu, Kazi Sajeed Mehrab, and Anuj Karpatne. Let there be order: Rethinking ordering in autoregressive graph generation. *arXiv preprint arXiv:2305.15562*, 2023. 2

[29] Xingping Xian, Tao Wu, Xiaoke Ma, Shaojie Qiao, Yabin Shao, Chao Wang, Lin Yuan, and Yu Wu. Generative graph neural networks for link prediction. *arXiv preprint arXiv:2301.00169*, 2022. 2

[30] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems*, 34:9061–9073, 2021. 2, 4

[31] Haoteng Yin, Muhan Zhang, Yanbang Wang, Jianguo Wang, and Pan Li. Algorithm and system co-design for efficient subgraph-based graph representation learning. *arXiv preprint arXiv:2202.13538*, 2022. 2

[32] Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR, 2020. 2

[33] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 2

[34] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[35] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2

[36] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022. 2

[37] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 2

[39] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020. 2

[40] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023. 2

[41] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 2

[42] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023. 2

[43] Kevin Clark and Priyank Jaini. Text-to-image diffusion models are zero-shot classifiers. *arXiv preprint arXiv:2303.15233*, 2023. 3

[44] Soumik Mukhopadhyay, Matthew Gwilliam, Vatsal Agarwal, Namitha Padmanabhan, Archana Swaminathan, Srinidhi Hegde, Tianyi Zhou, and Abhinav Shrivastava. Diffusion models beat gans on image classification. *arXiv preprint arXiv:2307.08702*, 2023. 3

[45] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022. 3, 4, 14

[46] Wenqi Fan, Chengyi Liu, Yunqing Liu, Jiatong Li, Hang Li, Hui Liu, Jiliang Tang, and Qing Li. Generative diffusion models on graphs: Methods and applications. *arXiv preprint arXiv:2302.02591*, 2023. 3

[47] Xinyue Wang, Yilin Lyu, and Liping Jing. Deep generative model for robust imbalance classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14124–14133, 2020. 4

[48] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021. 4

[49] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 5, 6

[50] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 6

[51] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 6, 15

[52] Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 34:13683–13694, 2021. 6

[53] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. *Advances in Neural Information Processing Systems*, 36, 2024. 6

[54] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016. 6

[55] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. *arXiv preprint arXiv:2203.00199*, 2022. 6

[56] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning*, pages 695–704. PMLR, 2019. 8

[57] Sixiao Zhang, Hongxu Chen, Xiangguo Sun, Yicong Li, and Guandong Xu. Unsupervised graph poisoning attack via contrastive loss back-propagation. In *Proceedings of the ACM Web Conference 2022*, pages 1322–1330, 2022. 8

[58] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020. 9

A Appendix

A.1 Algorithm Pseudo Code

The entire process of SGDIFF is shown in Algorithm 1. We estimate the likelihood scores for the structure and features simultaneously from lines 2 to 9 and 10 to 16, respectively. The two components are then fused together on line 17. Lastly, the sample's final connection probability is estimated on line 18.

Algorithm 1: Sub-graph Based Diffusion Model (SGDIFF)

Input: Sub-graph $G = (\mathbf{A}, \mathbf{X})$, connection condition inputs $y_c \in \{0, 1\}$, structure diffusion model ϕ_θ, feature diffusion model ϵ_θ, fusion parameter set $\{\eta_1, \eta_2, \delta\}$, number of steps of structure diffusion model N_ϕ, number of steps of feature diffusion model N_ϵ.

1 Initialize $\text{StructureScore}[y_c] = \text{list}()$ and $\text{FeatureScore}[y_c] = \text{list}()$ for each y_c;
2 **for** *step* $t \leftarrow N_\phi$ **to** 1 **do**
3 prepare $\mathbf{X}'^{(0)}$ with labeling tricks on $\mathbf{A}^{(0)}$;
4 sample $G^{(t)}$ with $q(G^{(t)}|G^{(0)}) = (\mathbf{A}^{(0)}\bar{\mathbf{Q}}_A^t, \mathbf{X}'^{(0)}\bar{\mathbf{Q}}_X^t)$;
5 **for** *conditioning* $y_c \leftarrow 0$ **to** 1 **do**
6 $\text{StructureScore}[y_c].\text{append}($
7 $D_{KL}[q(G^{t-1}|G^t, G)||p_\theta(G^{t-1}|G^t, y_c))$
8 **end**
9 **end**
10 **for** *step* $t \leftarrow N_\epsilon$ **to** 1 **do**
11 sample $\epsilon \sim \mathcal{N}(0, I)$;
12 $\mathbf{X}^{(t)} = \sqrt{\bar{\alpha}_t}\mathbf{X}^{(0)} + \sqrt{1 - \bar{\alpha}_t}\epsilon$;
13 **for** *conditioning* $y_c \leftarrow 0$ **to** 1 **do**
14 $\text{FeatureScore}[y_c].\text{append}(||\epsilon - \epsilon_\theta(\mathbf{X}^{(t)}, \mathbf{A}^{(0)}, y_c)||^2)$
15 **end**
16 **end**
17 **calculate** $\log P(\mathbf{A}|y_c) = \text{mean}(\text{StructureScore}[y_c])$;
$\log P(\mathbf{X}|\mathbf{A}, y_c) = \text{mean}(\text{FeatureScore}[y_c])$;
$\log P(\mathbf{A}, \mathbf{X}|y_c) = \eta_1 \cdot \log P(\mathbf{X}|\mathbf{A}, y) + \eta_2 \cdot \log P(\mathbf{A}|y) + \delta$;
18 **return** $\arg\min_{y_c \in \{0,1\}}(\text{softmax}(\log P(\mathbf{A}, \mathbf{X}|y_c))$

A.2 Dataset Details

The details about each dataset are shown in Tabel 3. Following prior works [12, 16], we split the existing links in each graph into train/valid/test with the percentages 80%/5%/15%. For evaluation, we randomly sample the same amount of unconnected node pairs as the negative samples. The evaluation metrics used in our experiment are AUC, Average Precision(AP) and Hit@100.

Table 3: Detailed statistical information about each dataset.

Data	Domain	Node Number	Edge Number	Average Node Degree	Node Feature / Label
Cora	Citation	2,708	10,556	3.89	✔
Citeseer	Citation	3,327	9,228	2.77	✔
Pubmed	Citation	19,717	88,651	4.49	✔
Router	Transporation	5,022	12,516	2.49	✘
USAir	Transporation	332	4,252	12.81	✘
NS	Collaboration	1,589	5,484	3.45	✘

A.3 Implement Details

The implementation and hyper-parameter settings of the two baseline models follow prior works [12, 16]. The implementation of our structure diffusion model follows the prior work [45] and the feature diffusion model is implemented with a multi-layer GCNs. During the enclosing graph generation

Table 4: Hyper-parameter setting of structure and feature diffusion models.

Name	Symbol	Cora	Citeseer	Pubmed	Router	USAir	NS
Sturcture Diffusion							
Hop number of subgraph enclosing the link	k	1	1	1	1	1	1
Maximum node number for each hop's sampling	ns	-1	20	20	10	40	5
Attention hidden neuron number of each layer for node representation	ha_x	256					
Attention hidden neuron number of each layer for edge representation	ha_e	64					
Attention hidden neuron number of each layer for global condition representation	ha_y	64					
MLP hidden neuron number of each layer for node representation	hm_x	256					
Attention hidden neuron number of each layer for edge representation	hm_e	128					
Attention hidden neuron number of each layer for global condition representation	hm_y	128					
Head number of attention	head	8					
Number of transformer layer	l_t	2					
Number of diffusion steps	ds_t	20	20	10	5	10	20
Feature Diffusion							
Hidden neruon of GCNs	h_g	256	256	64	N/A		
Number of GCN layers	l_g	2	2	2			
Number of diffusion steps	ds_g	100	100	50			

process, we incorporate the neighbor sampling trick [51] to avoid the graph size becoming extremely large when it encounters some hub nodes. To add DRNL into the structure diffusion process, we treat extracted structure labels as categorical variables and use the sum of node and feature cross-entropy loss to train the structure denoising model. We perform grid search over the hyper parameters of our score and feature diffusion models. The best parameter of each components for each dataset is shown in Table. 4. As the prior $p(y = 1)$ and $p(y = 0)$ are constant numbers during the ELBO calculation, we choose to set $p(y) = 0.5$ in our experiment. In parameter tuning process, we try to use the natural distribution of y for the prior $p(y)$. However, as we found that there is no significant difference, we keep using $p(y) = 0.5$ for all our experiments for simplicity.

A.4 Case Studies on Cross-data Transferability

To explore the explanations to the inconsistent cases among discriminative and generative over different datasets in the cross-data transferability experiment. To begin with, we choose to use the comparison between SEAL and SGDIFF over the 6 datasets as our study's target since they are the top-2 ranked methods listed in Table. 1 and both of them leverage the sub-graph learning idea. Then, we select one simple yet effective structure feature: common neighbor (CN), and calculate the its KL-divergence, D_{KL} between the feature distributions in **Source** ($\mathcal{S}$) and **Target** ($\mathcal{T}$) datasets as an index to represent the difference between the graph structures of source and target datasets. Specifically, we first calculate the CN for all positive edges ($+$) and negative edges ($-$) in the train split of $\mathcal{S}$ and the test split of $\mathcal{T}$. Then, we calculate $D_{\text{KL}}(q^{+}_{\mathcal{S}}(\text{CN})\|p^{+}_{\mathcal{T}}(\text{CN}))$ and $D_{\text{KL}}(q^{-}_{\mathcal{S}}(\text{CN})\|p^{-}_{\mathcal{T}}(\text{CN}))$, respectively. At last, we use the average value of the two KL-divergence as the index for the Source-Target transfer learning pair. To align with the presenting formats in Table 1, we aggregate the pair-wise results over different source and target datasets independently. Overall, our case study result is shown in Table. 5.

From the table, we can clearly observe that the performance gap between SEAL and SGDIFF is negatively related to the D_{KL}. This fact suggests that when the graph structure feature distribution between source and target is large (D_{KL} is large), the SGDIFF will be outperformed by SEAL. On the contrary, the SGDIFF outperforms SEAL when the source and target datasets share structure features.

Table 5: Case studies over D_{KL} and performance difference between SEAL and SGDIFF.

Data	Source						Target					
	Cora	Citeseer	Pubmed	Router	NS	USAir	Cora	Citeseer	Pubmed	Router	NS	USAir
D_{KL}	4.43	2.31	2.55	4.05	2.95	2.24	0.86	0.87	0.69	0.84	3.13	14.15
AUC												
SEAL	89.09	84.55	88.84	87.98	86.55	75.03	84.88	83.14	80.02	78.45	94.86	90.69
SGDiff	85.94	90.49	92.07	87.99	87.98	83.80	86.93	86.23	90.78	88.62	91.62	84.09
Difference	-3.15	5.94	3.23	0.01	1.43	8.77	2.05	3.09	10.76	10.17	-3.24	-6.60
AP												
SEAL	90.54	87.06	91.51	89.02	88.16	78.76	87.50	86.89	82.25	81.32	95.92	91.16
SGDiff	87.79	91.36	92.65	86.54	88.66	85.17	87.93	86.98	90.90	89.62	91.56	85.19
Difference	-2.75	4.30	1.14	-2.48	0.50	6.41	0.43	0.09	8.65	8.30	-4.36	-5.97
Hit@100												
SEAL	78.42	74.46	82.31	77.71	73.46	59.26	77.16	76.14	39.09	62.92	94.96	95.35
SGDiff	74.99	83.37	86.02	76.82	75.01	74.51	78.95	80.08	50.43	78.67	93.84	88.75
Difference	-3.43	8.91	3.71	-0.89	1.55	15.25	1.79	3.94	11.34	15.75	-1.12	-6.60

Table 6: Area under the curve (AUC) of different models in cross-data transferability experiment.

Target	Source	GCN	GAT	SAGE	NeoGNN	VGAE	SEAL	SGDiff
Cora	Cora	90.49 ± 0.59	89.85 ± 0.97	90.28 ± 0.84	92.01 ± 0.61	88.98 ± 1.09	91.74 ± 0.91	90.21 ± 2.21
	Citeseer	71.99 ± 3.08	80.35 ± 1.47	77.29 ± 1.62	84.05 ± 1.72	67.16 ± 3.87	88.11 ± 1.73	90.09 ± 0.81
	Pubmed	75.14 ± 4.31	77.12 ± 0.75	79.08 ± 1.29	84.49 ± 2.72	67.68 ± 3.80	88.36 ± 0.52	90.73 ± 2.03
	Router	72.70 ± 1.42	72.18 ± 1.19	69.71 ± 1.48	82.44 ± 1.51	60.69 ± 3.17	84.42 ± 1.65	86.17 ± 2.31
	NS	50.42 ± 4.60	66.20 ± 1.11	64.61 ± 1.41	75.60 ± 1.45	59.07 ± 1.58	84.08 ± 2.39	84.65 ± 1.91
	USAir	62.36 ± 2.93	62.85 ± 1.88	59.01 ± 1.26	77.40 ± 2.77	59.87 ± 1.35	72.57 ± 3.80	79.74 ± 5.07
Citeseer	Citeseer	89.64 ± 1.11	88.90 ± 1.62	89.35 ± 1.28	90.60 ± 1.01	88.17 ± 0.80	89.37 ± 0.99	89.36 ± 2.16
	Cora	80.12 ± 1.87	79.55 ± 1.45	79.43 ± 2.64	82.82 ± 1.43	67.93 ± 2.58	89.14 ± 1.04	87.81 ± 2.39
	Pubmed	75.94 ± 4.77	78.04 ± 1.92	79.18 ± 1.92	81.11 ± 3.02	68.34 ± 2.73	80.78 ± 1.31	91.64 ± 1.77
	Router	62.68 ± 1.59	69.02 ± 3.04	66.06 ± 1.94	75.84 ± 2.27	58.79 ± 2.73	82.45 ± 2.04	87.49 ± 2.96
	NS	57.36 ± 3.88	66.37 ± 2.27	64.79 ± 2.34	67.45 ± 2.76	57.74 ± 2.20	86.53 ± 1.16	82.93 ± 2.09
	USAir	60.98 ± 2.27	63.24 ± 2.74	59.24 ± 1.91	70.45 ± 3.38	56.62 ± 1.16	70.58 ± 3.15	78.13 ± 5.75
Pubmed	Pubmed	96.01 ± 0.30	93.07 ± 0.37	96.17 ± 0.20	96.50 ± 0.32	95.37 ± 0.19	97.36 ± 0.18	95.97 ± 0.75
	Cora	89.54 ± 1.96	83.13 ± 0.95	86.70 ± 1.26	87.24 ± 1.49	78.10 ± 2.94	87.34 ± 2.19	90.74 ± 2.17
	Citeseer	78.05 ± 5.57	82.29 ± 2.45	85.50 ± 1.04	83.25 ± 1.38	77.78 ± 3.73	79.54 ± 3.27	94.47 ± 1.48
	Router	75.93 ± 0.58	68.19 ± 1.21	78.76 ± 0.54	83.61 ± 6.86	57.84 ± 4.50	87.06 ± 2.36	85.59 ± 3.48
	NS	33.15 ± 4.98	58.91 ± 0.79	66.91 ± 0.93	67.22 ± 0.58	52.27 ± 0.65	78.06 ± 7.26	88.76 ± 3.73
	USAir	57.39 ± 3.76	58.34 ± 1.37	63.21 ± 0.88	77.04 ± 2.09	53.71 ± 0.92	50.77 ± 7.42	89.13 ± 2.39
Router	Router	84.05 ± 1.03	64.33 ± 2.32	75.23 ± 0.97	70.09 ± 4.99	63.39 ± 2.35	95.90 ± 0.27	94.76 ± 0.69
	Cora	60.09 ± 3.61	49.57 ± 1.94	65.65 ± 1.52	45.48 ± 3.37	53.21 ± 1.61	77.99 ± 3.03	88.80 ± 2.24
	Citeseer	38.67 ± 4.50	45.66 ± 1.63	64.93 ± 1.12	37.46 ± 2.21	50.32 ± 0.96	66.34 ± 6.74	91.90 ± 1.68
	Pubmed	70.05 ± 1.16	57.13 ± 1.04	69.89 ± 1.08	69.96 ± 1.24	48.15 ± 1.40	84.44 ± 1.04	94.20 ± 0.91
	NS	22.42 ± 3.13	42.00 ± 1.62	58.89 ± 0.95	35.12 ± 1.91	52.34 ± 2.18	81.99 ± 4.12	82.95 ± 8.46
	USAir	59.23 ± 4.96	43.91 ± 2.18	58.11 ± 2.08	62.11 ± 5.49	56.81 ± 2.66	64.03 ± 7.81	79.09 ± 5.71
NS	NS	89.79 ± 1.98	90.97 ± 1.45	91.75 ± 1.09	88.87 ± 1.47	93.32 ± 0.90	98.28 ± 0.35	97.47 ± 0.57
	Cora	86.88 ± 1.33	90.26 ± 1.18	87.28 ± 1.65	90.75 ± 1.62	77.60 ± 1.60	97.59 ± 0.42	92.16 ± 4.30
	Citeseer	87.58 ± 1.16	88.28 ± 0.84	86.38 ± 1.76	89.51 ± 1.64	78.74 ± 2.77	97.07 ± 0.57	95.46 ± 2.66
	Pubmed	90.52 ± 1.24	90.99 ± 1.58	90.16 ± 1.37	91.20 ± 1.48	84.41 ± 1.62	92.62 ± 1.15	95.02 ± 1.16
	Router	76.78 ± 3.02	90.18 ± 1.42	84.92 ± 2.44	92.44 ± 1.11	76.58 ± 5.54	89.01 ± 2.84	89.15 ± 2.57
	USAir	78.54 ± 1.42	85.21 ± 1.61	74.37 ± 1.68	88.10 ± 2.67	73.66 ± 2.07	94.59 ± 1.77	80.43 ± 5.64
USAir	USAir	93.75 ± 1.64	95.34 ± 1.10	94.91 ± 1.07	94.15 ± 1.50	90.84 ± 1.21	97.62 ± 0.55	96.25 ± 1.58
	Cora	86.38 ± 2.19	83.48 ± 2.43	85.11 ± 2.37	83.25 ± 13.72	65.53 ± 5.33	90.76 ± 2.43	65.90 ± 24.32
	Citeseer	86.34 ± 2.46	81.83 ± 2.95	84.21 ± 2.40	62.04 ± 13.81	79.33 ± 4.57	86.86 ± 4.01	81.65 ± 11.75
	Pubmed	92.90 ± 1.39	87.43 ± 2.88	90.53 ± 1.49	92.68 ± 1.78	88.81 ± 1.67	89.49 ± 1.18	84.84 ± 12.23
	Router	77.06 ± 3.07	86.80 ± 1.30	83.11 ± 1.64	82.13 ± 17.75	70.04 ± 12.63	89.04 ± 2.05	84.79 ± 3.49
	NS	51.92 ± 9.13	81.07 ± 1.77	76.82 ± 3.07	48.71 ± 6.89	59.31 ± 2.54	90.34 ± 1.96	91.11 ± 2.95

We think such observation may be caused by the difference in discriminative and generative training methods. As generative training is not naturally designed for discriminative tasks, it fails to face the large distribution shift between source and target datasets [24].

A.5 Cross-dataset Performance Details

We explore the transferability of models by setting each of the six graph as the training dataset and testing the trained model on the other five and itself under the zero-shot scenario. Specicially, detailed performances of seven baseline models and SGDIFF on AUC, AP and Hit@100 are presented in Table. 6, Table. 7 and Table. 8. For each metric, we run experiment for 10 times and the mean value and standard deviation are reported in the format of mean ± std%.

Table 7: Average precision (AP) of different models in cross-data transferability experiment.

Target	Source	GCN	GAT	SAGE	NeoGNN	VGAE	SEAL	SGDiff
Cora	Cora	92.16 ± 0.40	91.14 ± 0.69	91.43 ± 0.87	93.39 ± 0.50	90.81 ± 0.83	92.85 ± 0.64	90.00 ± 2.41
	Citeseer	76.79 ± 2.49	82.48 ± 1.22	78.37 ± 1.70	88.17 ± 1.05	67.47 ± 3.82	90.21 ± 1.40	91.08 ± 1.11
	Pubmed	77.79 ± 3.86	80.26 ± 0.99	81.15 ± 1.20	88.79 ± 1.56	67.43 ± 4.72	90.86 ± 0.47	91.92 ± 1.86
	Router	74.05 ± 1.45	77.65 ± 0.87	70.89 ± 1.27	86.74 ± 1.75	61.39 ± 4.15	86.90 ± 1.30	85.85 ± 3.17
	NS	58.42 ± 4.08	71.68 ± 1.19	65.44 ± 0.92	82.44 ± 1.63	58.96 ± 1.71	86.00 ± 1.47	86.15 ± 1.85
	USAir	62.80 ± 2.40	66.42 ± 2.42	57.48 ± 1.38	78.76 ± 3.95	59.32 ± 1.27	78.18 ± 3.58	82.55 ± 3.98
Citeseer	Citeseer	91.54 ± 0.94	90.93 ± 1.36	91.23 ± 1.12	92.41 ± 0.99	90.19 ± 0.85	91.62 ± 0.91	90.87 ± 1.93
	Cora	80.59 ± 2.04	81.67 ± 1.96	81.80 ± 2.97	87.44 ± 0.99	67.36 ± 2.22	91.28 ± 0.70	88.95 ± 2.02
	Pubmed	77.94 ± 4.75	83.54 ± 1.72	82.63 ± 1.52	86.62 ± 2.05	67.55 ± 3.15	86.52 ± 1.24	92.46 ± 1.48
	Router	64.64 ± 1.14	76.56 ± 2.10	68.65 ± 1.76	81.66 ± 3.05	58.43 ± 2.36	85.03 ± 1.79	86.66 ± 4.02
	NS	63.72 ± 3.10	74.11 ± 1.92	66.03 ± 2.82	78.70 ± 1.89	57.15 ± 2.19	89.43 ± 0.97	85.18 ± 2.17
	USAir	59.55 ± 1.94	69.57 ± 2.31	57.79 ± 1.56	72.32 ± 3.94	55.39 ± 1.27	77.47 ± 3.06	77.77 ± 6.36
Pubmed	Pubmed	96.28 ± 0.30	93.73 ± 0.29	96.29 ± 0.24	97.04 ± 0.25	96.08 ± 0.20	97.38 ± 0.17	95.91 ± 1.14
	Cora	90.84 ± 1.38	83.57 ± 1.26	86.82 ± 1.66	90.96 ± 0.95	78.87 ± 3.33	88.59 ± 1.64	90.77 ± 2.18
	Citeseer	83.65 ± 4.32	82.88 ± 3.50	85.39 ± 1.40	88.27 ± 0.91	78.57 ± 4.34	83.58 ± 2.56	94.71 ± 1.48
	Router	75.35 ± 0.67	74.48 ± 1.34	76.68 ± 0.77	87.68 ± 5.16	58.19 ± 6.46	88.05 ± 2.17	83.14 ± 4.23
	NS	41.07 ± 3.42	63.73 ± 1.33	63.31 ± 1.19	76.37 ± 2.95	50.45 ± 0.56	79.39 ± 4.81	89.59 ± 2.35
	USAir	59.05 ± 2.32	61.54 ± 2.04	58.71 ± 0.90	79.47 ± 1.63	54.09 ± 0.98	56.52 ± 5.29	91.25 ± 1.56
Router	Router	85.60 ± 1.01	72.16 ± 2.20	78.39 ± 1.60	77.53 ± 3.67	67.73 ± 1.85	95.78 ± 0.29	94.60 ± 1.00
	Cora	65.25 ± 3.35	56.96 ± 2.33	65.97 ± 1.52	61.65 ± 2.47	55.76 ± 1.28	81.42 ± 2.26	89.70 ± 1.49
	Citeseer	46.31 ± 3.28	55.85 ± 1.55	64.93 ± 1.57	53.99 ± 3.46	53.11 ± 0.89	72.38 ± 5.33	92.01 ± 1.86
	Pubmed	75.82 ± 1.15	64.60 ± 1.26	71.79 ± 1.18	77.18 ± 1.11	52.72 ± 1.78	88.63 ± 0.82	93.95 ± 0.90
	NS	40.44 ± 2.00	50.49 ± 2.10	57.38 ± 1.15	54.00 ± 4.46	54.25 ± 1.82	82.96 ± 3.28	83.05 ± 6.31
	USAir	63.68 ± 3.67	50.28 ± 1.61	56.08 ± 1.77	68.92 ± 5.06	59.78 ± 2.23	66.72 ± 6.39	84.40 ± 4.73
NS	NS	93.27 ± 1.72	93.85 ± 0.74	94.00 ± 0.88	93.77 ± 0.83	94.74 ± 0.74	98.53 ± 0.29	97.76 ± 0.60
	Cora	87.92 ± 1.93	93.85 ± 0.81	89.58 ± 1.35	94.59 ± 0.94	74.49 ± 1.68	98.00 ± 0.39	94.55 ± 2.70
	Citeseer	91.02 ± 0.93	92.32 ± 0.59	88.77 ± 1.62	94.07 ± 0.92	75.98 ± 3.08	97.65 ± 0.37	95.11 ± 3.32
	Pubmed	93.61 ± 0.81	94.08 ± 0.89	92.32 ± 0.96	94.57 ± 0.96	82.18 ± 2.09	94.85 ± 1.09	95.79 ± 1.27
	Router	74.83 ± 3.48	93.66 ± 0.89	87.33 ± 1.95	94.99 ± 0.66	73.91 ± 5.63	90.36 ± 2.40	87.93 ± 3.75
	USAir	74.33 ± 1.56	89.54 ± 1.25	72.80 ± 1.82	89.42 ± 3.48	68.82 ± 2.75	96.15 ± 1.18	78.21 ± 7.62
USAir	USAir	95.24 ± 1.12	95.89 ± 0.82	94.55 ± 1.04	95.84 ± 1.09	93.00 ± 0.98	97.51 ± 0.86	96.84 ± 1.39
	Cora	85.73 ± 2.55	81.46 ± 4.03	83.24 ± 3.65	84.95 ± 12.86	59.94 ± 6.20	91.07 ± 3.55	72.79 ± 17.80
	Citeseer	86.89 ± 2.64	79.64 ± 3.26	80.82 ± 4.13	65.22 ± 13.01	79.95 ± 5.95	86.93 ± 5.12	84.36 ± 8.04
	Pubmed	94.22 ± 1.35	85.22 ± 4.18	90.03 ± 1.57	94.08 ± 1.81	91.27 ± 1.40	90.83 ± 1.37	85.84 ± 8.85
	Router	74.80 ± 3.13	86.09 ± 2.05	80.22 ± 2.01	83.47 ± 15.26	69.21 ± 15.81	87.97 ± 2.52	81.08 ± 4.30
	NS	55.78 ± 6.96	80.01 ± 1.71	72.94 ± 4.36	50.48 ± 4.39	53.57 ± 2.24	92.67 ± 2.46	90.24 ± 4.21

Table 8: Hit@100 of different models in cross-data transferability experiment.

Target	Source	GCN	GAT	SAGE	NeoGNN	VGAE	SEAL	SGDiff
Cora	Cora	85.97 ± 1.06	84.94 ± 1.78	84.43 ± 1.56	87.36 ± 2.02	81.87 ± 1.80	87.56 ± 2.24	83.68 ± 4.48
	Citeseer	56.57 ± 3.22	66.23 ± 2.66	58.34 ± 2.97	76.13 ± 2.60	41.64 ± 6.27	81.86 ± 2.75	84.49 ± 1.64
	Pubmed	59.31 ± 6.58	61.78 ± 1.86	64.82 ± 2.69	76.59 ± 3.75	41.70 ± 6.66	82.26 ± 2.34	85.76 ± 3.76
	Router	50.81 ± 3.66	56.53 ± 1.67	48.61 ± 3.53	72.88 ± 2.98	33.03 ± 6.65	74.72 ± 3.72	76.48 ± 5.75
	NS	28.19 ± 5.61	44.88 ± 3.04	39.54 ± 3.08	69.92 ± 2.06	28.56 ± 2.64	76.31 ± 5.02	72.69 ± 3.30
	USAir	35.85 ± 3.76	39.08 ± 3.82	29.42 ± 2.50	66.90 ± 6.91	31.18 ± 3.48	60.26 ± 5.85	70.58 ± 8.60
Citeseer	Citeseer	85.39 ± 1.93	83.20 ± 2.47	84.44 ± 1.54	85.30 ± 1.10	81.87 ± 1.23	85.99 ± 2.44	84.34 ± 4.09
	Cora	68.89 ± 3.02	69.90 ± 2.67	67.68 ± 5.43	74.10 ± 2.25	44.70 ± 4.69	86.88 ± 2.01	82.55 ± 5.19
	Pubmed	63.97 ± 7.51	68.11 ± 3.33	69.66 ± 3.13	72.32 ± 3.89	46.42 ± 6.43	72.99 ± 2.30	88.44 ± 3.00
	Router	40.31 ± 3.01	56.82 ± 3.78	49.03 ± 3.13	65.08 ± 3.69	32.06 ± 3.95	71.84 ± 5.42	83.08 ± 4.04
	NS	39.98 ± 5.08	53.45 ± 3.61	44.70 ± 4.47	59.42 ± 2.78	28.27 ± 3.24	81.10 ± 2.22	70.49 ± 5.02
	USAir	35.19 ± 3.45	47.15 ± 3.52	34.15 ± 3.05	59.89 ± 7.71	26.99 ± 3.24	58.01 ± 8.41	71.59 ± 8.65
Pubmed	Pubmed	73.75 ± 3.23	64.00 ± 2.08	73.53 ± 1.59	78.90 ± 1.85	74.22 ± 1.48	75.21 ± 1.43	68.77 ± 8.79
	Cora	50.94 ± 4.53	28.90 ± 3.76	35.51 ± 4.32	60.68 ± 5.67	22.74 ± 5.04	44.20 ± 5.27	44.38 ± 10.69
	Citeseer	44.58 ± 7.02	29.59 ± 8.35	30.35 ± 4.80	52.23 ± 9.10	22.81 ± 6.28	44.32 ± 5.29	65.81 ± 8.92
	Router	17.70 ± 1.14	25.68 ± 3.13	13.59 ± 2.02	51.12 ± 11.81	5.61 ± 3.48	42.64 ± 5.97	24.02 ± 10.25
	NS	1.36 ± 0.71	11.25 ± 1.90	3.91 ± 0.83	28.64 ± 20.89	2.50 ± 0.37	22.92 ± 10.61	45.86 ± 6.43
	USAir	2.95 ± 3.62	8.07 ± 2.40	2.10 ± 0.30	18.53 ± 9.65	4.86 ± 0.53	5.26 ± 2.38	53.75 ± 8.30
Router	Router	68.28 ± 2.66	44.53 ± 3.30	57.77 ± 1.76	55.55 ± 5.69	36.55 ± 3.89	93.11 ± 1.78	89.47 ± 3.41
	Cora	38.95 ± 4.66	25.20 ± 2.73	40.67 ± 2.80	34.71 ± 3.29	23.93 ± 2.36	58.70 ± 7.77	78.41 ± 4.29
	Citeseer	14.82 ± 3.68	23.09 ± 1.84	39.26 ± 2.31	29.73 ± 2.30	20.71 ± 1.68	44.98 ± 10.47	82.71 ± 3.27
	Pubmed	54.16 ± 2.18	32.35 ± 1.77	49.49 ± 2.29	54.23 ± 2.04	23.11 ± 2.84	77.06 ± 2.17	88.19 ± 5.71
	NS	4.76 ± 2.30	19.70 ± 1.90	26.84 ± 2.49	28.02 ± 1.83	21.49 ± 2.62	65.66 ± 9.07	65.79 ± 14.49
	USAir	37.04 ± 6.24	19.30 ± 1.76	24.70 ± 2.98	48.32 ± 5.97	29.72 ± 3.13	37.99 ± 10.19	67.46 ± 11.50
NS	NS	88.31 ± 2.16	89.02 ± 2.03	89.95 ± 1.63	87.57 ± 1.83	92.77 ± 1.21	98.92 ± 0.69	98.54 ± 1.28
	Cora	88.62 ± 1.83	88.79 ± 1.58	87.42 ± 2.12	89.53 ± 2.35	79.89 ± 3.75	98.39 ± 0.80	91.66 ± 5.77
	Citeseer	87.37 ± 0.91	86.32 ± 0.84	87.14 ± 2.87	88.56 ± 1.96	81.06 ± 5.13	97.30 ± 1.17	96.77 ± 2.46
	Pubmed	89.06 ± 1.54	88.90 ± 1.74	87.80 ± 2.05	89.27 ± 1.88	87.99 ± 1.65	93.23 ± 1.68	96.26 ± 1.39
	Router	75.13 ± 4.47	88.57 ± 1.82	85.18 ± 2.65	92.07 ± 2.40	77.03 ± 9.37	87.68 ± 6.58	94.52 ± 3.24
	USAir	83.05 ± 3.09	84.78 ± 1.98	73.44 ± 3.57	91.69 ± 1.95	74.14 ± 4.42	94.24 ± 2.74	85.28 ± 6.62
USAir	USAir	95.77 ± 1.63	96.95 ± 1.20	97.92 ± 1.16	96.05 ± 1.44	92.90 ± 1.54	99.78 ± 0.39	98.38 ± 0.62
	Cora	94.21 ± 1.77	91.36 ± 1.93	92.62 ± 2.75	86.18 ± 17.49	71.32 ± 7.43	94.77 ± 1.45	69.26 ± 26.48
	Citeseer	91.48 ± 2.84	90.62 ± 2.56	92.57 ± 2.93	58.84 ± 17.22	84.17 ± 4.74	92.30 ± 3.24	86.12 ± 11.80
	Pubmed	95.00 ± 1.66	93.97 ± 1.21	94.98 ± 1.93	95.32 ± 1.62	91.91 ± 1.69	93.13 ± 2.32	88.68 ± 15.24
	Router	84.39 ± 4.53	92.91 ± 1.74	90.54 ± 2.14	84.01 ± 24.45	72.60 ± 15.83	96.26 ± 1.49	93.37 ± 3.52
	NS	49.36 ± 12.58	89.01 ± 2.74	86.01 ± 3.94	39.81 ± 7.77	63.12 ± 2.84	95.86 ± 1.98	96.70 ± 1.74