Cross-Domain Graph Data Scaling: A Showcase with Diffusion Models

Wenzhuo Tang[1], Haitao Mao[2]*, Danial Dervovic[3], Ivan Brugere[3], Saumitra Mishra[3], Yuying Xie[1], Jiliang Tang[1]
[1] Michigan State University [2] Amazon [3] J.P. Morgan AI Research
{tangwen2, xyy, tangjili}@msu.edu, maoht@amazon.com
{danial.dervovic, ivan.brugere, saumitra.mishra}@jpmchase.com

Abstract

Models for natural language and images benefit from data scaling behavior: the more data fed into the model, the better they perform. This 'better with more' phenomenon enables the effectiveness of large-scale pre-training on vast amounts of data. However, current graph pre-training methods struggle to scale up data due to heterogeneity across graphs. To achieve effective data scaling, we aim to develop a general model that is able to capture diverse data patterns of graphs and can be utilized to adaptively help the downstream tasks. To this end, we propose *UniAug*, a universal graph structure augmentor built on a diffusion model. We first pre-train a discrete diffusion model on thousands of graphs across domains to learn the graph structural patterns. In the downstream phase, we provide adaptive enhancement by conducting graph structure augmentation with the help of the pre-trained diffusion model via guided generation. By leveraging the pre-trained diffusion model for structure augmentation, we consistently achieve performance improvements across various downstream tasks in a plug-and-play manner. To the best of our knowledge, this study represents the first demonstration of a data-scaling graph structure augmentor on graphs across domains.

1 Introduction

The effectiveness of existing foundation models [1, 2, 3] heavily relies on the availability of substantial amounts of data, where the relationship manifests as a scaling behavior between model performance and data scale [4]. Consistent performance gain has been observed with the increasing scale of pre-training data in both Natural Language Processing [4, 5] and Computer Vision [6, 7] domains. This data scaling phenomenon facilitates the development of general models endowed with extensive knowledge and effective data pattern recognition capabilities. In downstream applications, these models are capable of adaptively achieving performance gains across tasks.

In the context of graphs, the availability of large-scale graph databases [8, 9, 10] enables possible data scaling across datasets and domains. Existing works have demonstrated graph data scaling following two limited settings: in-domain pre-training [11, 12] and task-specific selection for pre-training data [13]. During the pre-training process, each graph in the pre-training pool must be validated as in-domain or relevant to the downstream dataset. Given a specific domain or task, the crucial discriminative data patterns are likely confined to a fixed set [14], leaving other potential patterns in diverse graph data distribution as noisy input. In terms of structure, graphs from different domains are particularly composed of varied patterns [15], making it hard to transfer across domains. For example, considering the building blocks of the graphs, the motifs shared by the World Wide Web hyperlinks

*This work was conducted while H.M. was a Ph.D. student at Michigan State University and is independent of H.M.'s current position at Amazon.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

only partially align with those shared by genetic networks [15]. Therefore, closely aligning the characteristics of the pre-training graphs and the downstream data both in feature and structure is essential for facilitating positive transfer [13]. As a consequence, the necessity of such meticulous data filtering restricts these methods from scaling up graphs effectively, as they can only utilize a small part of the available data. Given the limitation of the graph pre-training methods, a pertinent question emerges: *How can we effectively leverage the increasing scale of graph data across domains?*

Rather than focusing solely on data patterns specific to particular domains, we aim to develop a model that has a comprehensive understanding of data patterns inherent across various types of graphs. In line with the principles of data scaling, we hypothesize that incorporating a broader range of training datasets can help the model build an effective and universal graph pattern library, avoiding an overemphasis on major data patterns specific to any single dataset [16]. To construct such a general-purpose model, we propose to utilize a diffusion model operating only on the structure as the backbone, for the following key reasons. (1) Unlike features, graph structures follow a uniform construction principle, namely, the connections between nodes. This allows for positive transfer across domains when the upstream and downstream data exhibit similar topological patterns [13]. In particular, while the graph representations of neurons and forward electronic circuits are derived from distinct domains, they still share common motifs [15]. (2) Current supervised and self-supervised methods tend to capture only specific patterns of graph data, with models designed for particular inductive biases [16, 14, 17]. For instance, graph convolutional networks (GCNs) excel in node-level representation learning by emphasizing homophily, whereas graph-level representation learning benefits from expressive GNNs capable of distinguishing complex graph structures. (3) We opt for a structure-only model due to the heterogeneous feature spaces across graphs, which often include missing features or mismatched semantics [18]. For instance, node features yield completely different interpretations in citation networks, where they represent keywords of documents, compared to molecular networks, where they denote properties of atoms. To this end, we pre-train a structure-only diffusion model on thousands of graphs, which serves as the upstream component of our framework.

In the downstream stage, we employ the pre-trained diffusion model as a **Uni**versal graph structure **Aug**mentor (**UniAug**) to enhance the dataset, where diffusion guidance [19, 20, 21] is employed to align the generated structure with the downstream requirements. Specifically, we generate synthetic structures with various guidance objectives, and the resulting graphs consist of *generated structures* and *original node features*. This data augmentation paradigm strategically circumvents feature heterogeneity and fully utilizes downstream inductive biases by applying carefully designed downstream models to the augmented graphs in a plug-and-play manner. Empirically, we apply *UniAug* to graphs from diverse domains and consistently observe performance improvement in node classification, link prediction, and graph property prediction. To the best of our knowledge, this study represents the first demonstration of a cross-domain data-scaling graph structure augmentor.

2 Preliminary and Related Work

Learning from unlabeled graphs. Graph self-supervised learning (SSL) methods provide examples of pre-training and fine-tuning paradigm [22, 23, 24, 25, 26]. However, these methods benefit from limited data scaling due to feature heterogeneity, structural pattern differences across domains, and varying downstream inductive biases. It is worth mentioning that DCT [27] presents a pre-training and then data augmentation pipeline on molecules. Despite its impressive performance improvement on graph-level tasks, DCT is bounded with molecules and thus the use cases are limited.

Graph data augmentation. There have been many published works exploring graph data augmentation (GDA) since the introduction of graph neural networks (GNNs), with a focus on node-level [28, 29, 30], link-level [31, 32], and graph-level [33, 34, 35, 36, 37]. These GDA methods have been generally designed for specific tasks or particular aspects of graph data. In addition, they are often tailored for a single dataset and struggle to transfer to unseen patterns, which limits their generalizability to a broader class of applications.

Diffusion models on graphs. Diffusion models [38, 39, 40] are latent variable models that learn data distribution by gradually adding noise into the data and then recovering the clean input. Existing diffusion models on graphs can be classified into two main categories depending on the type of noise injected, i.e. Gaussian or discrete. Previous works employed Gaussian diffusion models

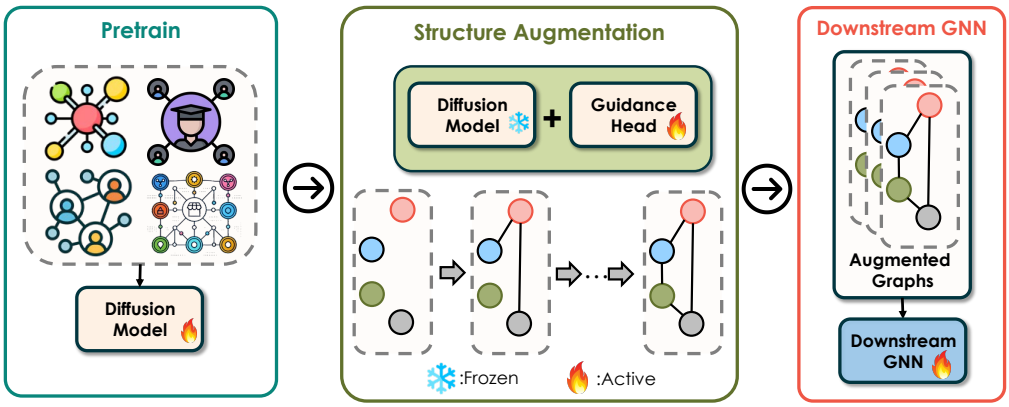

Figure 1: The pipeline of *UniAug*. We pre-train a diffusion model across domains and perform structure augmentation on the downstream graphs. The augmented graphs consist of *generated structures* and *original node features* and are then processed by a downstream GNN.

both on general graphs [41, 42] and molecules [43, 44]. However, adding Gaussian noise into the adjacency matrix will destroy the sparsity of the graph, which hinders the scalability of the diffusion models [45]. Recent works adapted discrete diffusion models to graphs with categorical transition kernels [46, 47, 48]. We denote the adjacency matrix of a graph as $\mathbf{A}^0 \in \{0,1\}^{n\times n}$ with n nodes. With details in Appendix A, we write the *forward process* to corrupt the adjacency matrix into a sequence of latent variables as Bernoulli distribution

$$\begin{aligned} q\left(\mathbf{A}^t \mid \mathbf{A}^{t-1}\right) &= \text{Bernoulli}\left(\mathbf{A}^t; \alpha^t \mathbf{A}^{t-1} + \left(1-\alpha^t\right)\pi\right), \\ q\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t, \mathbf{A}^0\right) &= \frac{q\left(\mathbf{A}^t \mid \mathbf{A}^{t-1}\right) q\left(\mathbf{A}^{t-1} \mid \mathbf{A}^0\right)}{q\left(\mathbf{A}^t \mid \mathbf{A}^0\right)}, \end{aligned} \tag{1}$$

where π is the converging non-zero probability, α^t is the noise scale, and $\bar{\alpha}^t = \prod_{i=1}^{t} \alpha^i$. Under predict-$\mathbf{A}^0$ parameterization, the *reverse process* denoise the adjacency matrix with a Markov chain

$$p_\theta\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t\right) \propto \sum_{\widetilde{\mathbf{A}}_0} q\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t, \widetilde{\mathbf{A}}_0\right) \tilde{p}_\theta\left(\widetilde{\mathbf{A}}_0 \mid \mathbf{A}^t\right), \tag{2}$$

where $\tilde{p}_\theta(\widetilde{\mathbf{A}}_0 \mid \mathbf{A}^t)$ represents the denoising network that predicts the original adjacency matrix from the noisy adjacency matrix. The parameters are estimated by optimizing the variational lower bound on the negative log-likelihood [49]

$$\begin{aligned} L_{\text{vb}} = &\sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{A}^t \mid \mathbf{A}^0)}\left[D_{\text{KL}}\left(q\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t, \mathbf{A}^0\right) \| \, p_\theta\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t\right)\right)\right] \\ &- \mathbb{E}_{q(\mathbf{A}^1 \mid \mathbf{A}^0)}\left[\log p_\theta\left(\mathbf{A}^0 \mid \mathbf{A}^1\right)\right] + \mathbb{E}_{q(\mathbf{A}^0)}\left[D_{\text{KL}}\left(q\left(\mathbf{A}^t \mid \mathbf{A}^0\right) \| \, p\left(\mathbf{A}^t\right)\right)\right]. \end{aligned} \tag{3}$$

3 Method

In this section, our goal is to build *UniAug* to understand the diverse structure patterns of graphs and perform data augmentation with a range of objectives. As illustrated in Fig.1, *UniAug* consists of two main components: a pre-trained diffusion model and the downstream adaptation through structure augmentation. We first collect thousands of graphs from varied domains with diverse patterns. To construct a general model free of downstream inductive biases, we train a self-conditioned discrete diffusion model on graph structures. In the downstream stage, we train an *MLP guidance head* on top of the diffusion model with objectives across different levels of granularity. We then augment the downstream dataset by generating synthetic structures through guided generation, where the augmented graph is composed of *generated structures* and *original node features*. Subsequently, we apply the augmented data to train a task-specific model for performing downstream tasks. Below, we elaborate on the data collection process, the architecture of the discrete diffusion model, and the guidance objectives.

3.1 Pre-training data collection

In light of the data scaling spirit, we expect our pre-training data to contain diverse data patterns with sufficient volume. As graphs from different domains exhibit different patterns [15], we wish to build a collection of graphs from numerous domains to enable a universal graph pattern library with pre-training. Within the publicly available graph databases, Network Repository [8] provides a comprehensive collection of graphs with varied scales from different domains, such as biological networks, chemical networks, social networks, and many more. Among the thousands of graphs in the Network Repository, some of them contain irregular patterns, including multiple levels of edges, extremely high density, et cetera. To ensure the quality of the graphs, we analyze the graph properties following Xu et al. [50] and filter out the outliers. In addition, we observe that the coverage of graphs in the Network Repository is incomplete according to the network entropy and scale-free exponent, as we observe a relatively scattered space in the middle of Fig. 2. To fill in the gap, we include a subset of the GitHub Star dataset [51] by random sampling 1000 graphs into our graph collection. The selected graphs are utilized to train a discrete diffusion model.

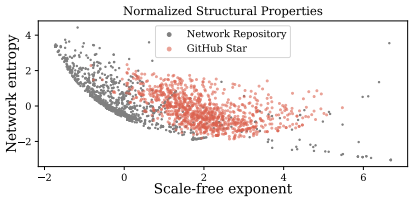

Figure 2: Normalized structural properties of Network Repository and Github Star. We enlarge the distribution coverage of our collection by combining both datasets.

3.2 Pre-training through diffusion model

Diffusion models have demonstrated the ability to facilitate transferability from a data augmentation perspective on the images [52, 53, 54]. Unlike the traditional hand-crafted data augmentation methods, diffusion models can produce more diverse patterns with high quality [52]. With the aid of diffusion guidance [19, 20], these methods can achieve domain customization tailored to specific semantic spaces [53, 54]. Despite the success of data augmentation through diffusion models on images, the non-Euclidean nature of graph structures poses challenges for data-centric learning on graphs. In addition, the fact that most graphs in the Network Repository are unlabeled exacerbates the challenges, as the absence of labeled data results in substantially lower generation quality for diffusion models [20, 55].

To address the aforementioned challenges, we propose to construct a self-conditioned discrete diffusion model on graph structures. Unlike Gaussian-based diffusion models, discrete diffusion models [56, 49, 57, 46] operate with discrete transition kernels between latent variables, as shown in Section 2. The key reason we opt for the discrete diffusion models lies in the sparse nature of graphs, where adding Gaussian noise into the adjacency matrix will result in a dense graph [45]. On the contrary, discrete diffusion models effectively preserve the sparse structure of graphs during the diffusion process, thus maintaining the efficiency of the models on graphs.

To accommodate for unlabeled graphs, we adopt a self-supervised labeling strategy as an auxiliary conditioning procedure [58, 59]. By leveraging the self-labeling technique, we are able to upscale the diffusion model to data with more diverse patterns [58]. The self-labeling technique requires two components: a feature extractor and a self-supervised annotator.

Feature extractor. We extract graph-level features by calculating graph properties, including the number of nodes, density, network entropy, average degree, degree variance, and scale-free exponent following [50]. The first two represent the scale of the graph corresponding to nodes and edges, and the rest indicate the amount of information contained within a graph [50]. We compute the properties of one graph and concatenate them to get a graph-level representation.

Self-supervised annotator. To assign labels to graphs in a self-supervised manner, we employ clustering algorithms on the graph-level representations. The number of clusters is determined jointly by the silhouette score [60] and the separation of the graphs. The candidates of the number of clusters are chosen to ensure different clusters are well separated. Among the candidates, we select the final number of clusters by maximizing the mean Silhouette Coefficient of all samples.

Table 1: Comparison between GDA methods, pre-training methods, and *UniAug*. By cross-domain transfer, we emphasize the ability of the method to train on vastly different domains and benefit all of them.

	GDA methods				Pre-training methods			*UniAug*
	GraphAug	CFLP	Half-Hop	FLAG	AttrMask	D-SLA	GraphMAE	
Effective on graph-level task	✓	–	–	✓	✓	✓	✓	✓
Effective on edge-level task	–	✓	–	✓	–	✓	–	✓
Effective on node-level task	–	–	✓	✓	–	–	✓	✓
In-domain transfer	–	–	–	–	✓	✓	✓	✓
Cross-domain transfer	–	–	–	–	–	–	–	✓

Next we detail the parameterization of the denoising model $\tilde{p}_\theta(\widetilde{\mathbf{A}}^0 \mid \mathbf{A}^t)$ with the self-assigned graph-level labels $\mathbf{k}$. The denoising model recovers the edges of the original adjacency matrix by predicting the connectivity of the upper triangle, which can be formulated as a link prediction problem [61, 62]. Following the link prediction setup, the denoising model is composed of a graph transformer (GT) [63] and an MLP link predictor. Denote the hidden dimension as d, we treat the node degrees as node features and utilize a linear mapping $f_d : \mathbb{R} \mapsto \mathbb{R}^d$ to match the dimension. Similarly, we utilize another linear mapping $f_t : \mathbb{R} \mapsto \mathbb{R}^d$ for timestep t and learnable embeddings $f_k : \{0, \ldots, K\} \mapsto \mathbb{R}^d$ for labels $\mathbf{k}$, where K is the number of clusters. The outputs are summed together and then fed into the GT. Mathematically, we have

$$\begin{aligned} \mathbf{h}^t &= \mathrm{GT}\left(f_d\left(\mathrm{degree}\left(\mathbf{A}^t\right)\right) + f_t(t) + f_k(\mathbf{k}), \mathbf{A}^t\right), \\ \tilde{p}_\theta(\widetilde{\mathbf{A}}^0_{ij} \mid \mathbf{A}^t; t, \mathbf{k}) &\coloneqq \tilde{p}_\theta(\widetilde{\mathbf{A}}^0_{ij} \mid \mathbf{h}^t) = \mathrm{MLP}\left([\mathbf{h}^t_i, \mathbf{h}^t_j]\right). \end{aligned} \tag{4}$$

With the above denoising network, our diffusion model is trained on the collected graphs by optimizing the variational lower bound in (3). After the pre-training process, we perform adaptive downstream enhancement through graph structure augmentation.

3.3 Downstream adaptation through data augmentation

The downstream phase of *UniAug* is to augment the graph topology through guided generation. This guidance process serves to provide downstream semantics for the diffusion model, thus bridging the gap between the pre-training distribution and the downstream datasets. Among the techniques for diffusion guidance, gradient-based methods [20, 21] offer versatile approaches by incorporating external conditions that are not present during training. For the discrete diffusion process, we opt for the gradient-based NOS method [21] due to its flexibility and efficiency. Specifically, we build an *MLP regression head* $g_\theta : \mathbb{R}^d \mapsto \mathbb{R}^r$ that takes the hidden representations $\mathbf{h}^t$ as the input and outputs the guidance objective of dimension r. Denote τ as the temperature, γ as the step-size, λ as the regularization strength, and ε drawn from $\mathcal{N}(0, I)$, we sample from $\tilde{p}'(\widetilde{\mathbf{A}}^0 \mid \mathbf{h}^t) \propto \tilde{p}_\theta(\widetilde{\mathbf{A}}^0 \mid \mathbf{h}^t) \exp\left(g_\theta\left(\mathbf{h}^t\right)\right)$ via Langevin dynamics

$$\mathbf{h}^{t,\prime} \leftarrow \mathbf{h}^{t,\prime} - \gamma \nabla_{\mathbf{h}^{t,\prime}} \left[\lambda \mathrm{KL}\left(\tilde{p}'\left(\widetilde{\mathbf{A}}^0 \mid \mathbf{h}^{t,\prime}\right) \| \, \tilde{p}'\left(\widetilde{\mathbf{A}}^0 \mid \mathbf{h}^t\right)\right) - g_\theta\left(\mathbf{h}^{t,\prime}\right)\right] + \sqrt{2\gamma\tau}\varepsilon. \tag{5}$$

One key question to answer is how to choose the proper guidance objectives. Our goal is to find numerical characteristics that can best describe the structural properties of a graph. This includes supervision signal and self-supervised information on the level of node, edge, and graph.

Node level. Node labels provide the supervision signal for node classification tasks. Beyond node labels, node degrees are a fundamental factor in the evolutionary process of a graph [64]. From the perspective of network analysis, centrality measures indicate the importance of nodes from various viewpoints [65]. Empirically, we observe that utilizing different node-level heuristics as guidance targets tends to yield similar outcomes. Therefore, we focus on node labels and node degrees.

Edge level. Edge-level heuristics can be broadly classified into two categories: local structural heuristics, such as Common Neighbor and Adamic Adar [66], and global structural heuristics, such as Katz [67] and SimRank [68]. Similar to node-level heuristics, empirical observations suggest that different edge-level heuristics tend to yield comparable guidance effects. In this work, we focus on the Common Neighbors (CN) heuristic due to its efficiency. Another edge-level guidance objective is to recover the adjacency matrix from the node representations in a link prediction way, similar to how we parameterize the denoising network. We anticipate that such link prediction objective helps to align the generated graph with the downstream data on the granularity of edges.

Table 2: Mean and standard deviation of accuracy (%) with 10-fold cross-validation on graph classification. The best result is **bold**. The highlighted results indicate negative transfer for pre-training methods compared to GIN. The last column is the average rank.

	DD	Enzymes	Proteins	NCI1	IMDB-B	IMDB-M	Reddit-B	Reddit-12K	Collab	A.R.
GIN	75.81 ± 6.11	66.00 ± 7.52	73.32 ± 4.03	78.30 ± 3.20	71.10 ± 2.90	49.07 ± 2.81	90.85 ± 1.30	48.63 ± 1.62	74.54 ± 2.41	5.56
AttrMask	72.93 ± 3.09	23.66 ± 6.09	73.10 ± 3.90	77.67 ± 2.53	71.20 ± 2.40	48.00 ± 3.14	87.50 ± 3.31	48.00 ± 1.60	75.64 ± 1.52	8.00
CtxtPred	75.14 ± 2.67	21.67 ± 3.87	72.21 ± 4.60	78.99 ± 1.29	70.70 ± 1.55	48.20 ± 2.23	90.35 ± 2.31	47.62 ± 2.50	75.60 ± 1.49	7.67
EdgePred	75.64 ± 2.77	22.00 ± 3.32	71.22 ± 3.53	77.82 ± 2.95	70.20 ± 2.23	47.80 ± 2.42	90.80 ± 1.69	48.35 ± 1.44	74.64 ± 2.24	8.56
InfoMax	75.23 ± 3.43	22.50 ± 6.76	71.30 ± 5.18	76.94 ± 1.48	71.60 ± 2.06	46.70 ± 2.46	89.15 ± 2.84	48.98 ± 1.83	75.44 ± 1.12	8.00
JOAO	75.98 ± 2.86	22.17 ± 3.67	71.57 ± 5.31	76.87 ± 2.27	71.02 ± 1.81	48.85 ± 2.06	90.17 ± 2.13	49.01 ± 1.90	74.77 ± 1.71	7.11
D-SLA	74.66 ± 3.30	22.67 ± 4.21	71.97 ± 4.17	77.95 ± 2.11	71.92 ± 2.75	47.28 ± 1.88	89.77 ± 1.87	48.50 ± 1.33	75.99 ± 2.08	7.00
GraphMAE	76.07 ± 3.25	23.00 ± 3.64	70.45 ± 4.19	79.08 ± 2.72	71.50 ± 2.01	47.93 ± 3.03	86.10 ± 3.63	47.67 ± 1.16	74.84 ± 1.36	7.67
S-Mixup	73.12 ± 3.27	66.85 ± 7.04	74.61 ± 5.08	78.91 ± 1.61	69.61 ± 4.43	48.33 ± 5.36	88.65 ± 3.12	48.30 ± 2.50	75.89 ± 3.26	6.67
GraphAug	75.21 ± 2.63	68.14 ± 7.92	74.21 ± 3.70	79.53 ± 3.21	**74.00 ± 3.41**	48.11 ± 1.85	90.50 ± 3.17	49.00 ± 1.99	76.02 ± 2.67	3.67
FLAG	76.87 ± 7.21	68.35 ± 7.45	74.31 ± 4.21	79.03 ± 3.75	68.83 ± 4.67	47.21 ± 3.45	89.11 ± 2.40	47.48 ± 3.01	75.32 ± 3.13	7.00
UniAug	**78.13 ± 2.61**	**71.50 ± 5.85**	**75.47 ± 2.50**	**80.54 ± 1.77**	73.50 ± 2.48	**50.13 ± 2.05**	**92.28 ± 1.59**	**49.48 ± 0.71**	**77.00 ± 2.02**	1.11

Graph level. Graph labels offer the supervision signal for graph classes or regression targets. In addition, we incorporate graph-level properties [50] as quantitative measures to bridge the gap between the pre-training distribution and the downstream dataset. We empirically observe that graph label guidance offers significantly higher performance boosts compared to properties on graph-level tasks. Therefore, we focus on graph labels in our experiments.

We provide our choice of objectives for each task in Appendix B. We note that all the above objectives are natural choices inspired by heuristics and downstream tasks. There exist many other self-supervised objectives to be explored, such as community-level spectral change [69] and motif occurrence prediction [70]. We leave the study of objectives as one future work. With the diffusion guidance, we assemble the augmented graphs with *generated structures* and *original node features*. The augmented graphs are then fed into downstream-specific GNNs.

3.4 Comparison to existing methods

The data augmentation paradigm of *UniAug* allows us to disentangle the upstream and downstream. We construct a diffusion model as the upstream component to comprehend the structural patterns of graphs across various domains. In addition, we leverage downstream inductive biases with downstream-specific models in a plug-and-play manner. This allows *UniAug* to facilitate cross-domain transfer, offering a unified method that benefits graphs across different domains for various downstream tasks. On the contrary, existing GDA methods are typically designed for specific tasks and hard to transfer to unseen patterns. In the meantime, existing pre-training methods fail to transfer across domains due to heterogeneity in features and structures. This comparison highlights the success of *UniAug* as a data-scaling graph structure augmentor across domains. We summarize the comparison between methods in Table 1.

4 Experiment

In this section, we conduct experiments to validate the effectiveness of *UniAug*. We first pre-train our discrete diffusion model on thousands of graphs collected from diverse domains. For each downstream task, we train an *MLP guidance head* with corresponding objectives on top of the diffusion model. We then perform structure augmentation using *UniAug* and subsequently train a task-specific GNN on augmented data for prediction. Through the experiments, we aim to answer the following research questions:

- RQ1: Can *UniAug* benefit graphs from various domains across different downstream tasks?
- RQ2: What is the scaling behavior of *UniAug* corresponding to data scale and amount of compute?
- RQ3: Which components of *UniAug* are effective in preventing negative transfer?

4.1 Main results

To get a comprehensive understanding of *UniAug*, we evaluate it on 25 downstream datasets from 7 domains for graph property prediction, link prediction, and node classification. The statistics of the datasets can be found in Appendix C, and technical details of the experiments are in Appendix D.

Table 3: Mean and standard deviation across 10 runs on link prediction. Results are scaled $\times 100$. The last two methods are based on NCN, while the rest are GCN-based. The best result is **bold** for two backbones, respectively. The last column is the average rank of each GCN-based method.

		Cora MRR	Citeseer MRR	Pubmed MRR	Power Hits@10	Yeast Hits@10	Erdos Hits@10	Flickr Hits@10	A.R.
	GCN	30.26 ± 4.80	50.57 ± 7.91	16.38 ± 1.30	30.61 ± 4.07	24.71 ± 4.92	35.71 ± 2.65	8.10 ± 2.58	5.14
Self-supervised	MVGRL	29.13 ± 3.90	51.32 ± 4.12	15.21 ± 2.35	31.71 ± 3.78	23.74 ± 5.74	36.21 ± 2.81	8.42 ± 2.18	5.29
	GRACE	31.77 ± 4.31	49.13 ± 3.95	16.88 ± 1.74	28.21 ± 5.04	23.96 ± 4.31	33.90 ± 2.12	**9.87 ± 0.98**	5.14
	BGRL	33.59 ± 2.14	51.91 ± 5.01	16.93 ± 2.03	33.71 ± 3.21	25.91 ± 3.12	37.95 ± 1.73	8.52 ± 1.85	3.00
	GraphMAE	32.98 ± 5.01	52.71 ± 5.39	**18.83 ± 1.30**	32.81 ± 2.12	26.51 ± 2.92	35.63 ± 3.61	7.01 ± 3.86	3.43
GDA	CFLP	33.62 ± 6.44	**55.20 ± 4.16**	17.01 ± 2.75	16.02 ± 8.31	24.23 ± 5.23	28.74 ± 2.38	0.00 ± 0.00	4.57
	UniAug - GCN	**35.36 ± 7.88**	54.66 ± 4.55	17.28 ± 1.89	**34.36 ± 1.68**	**27.52 ± 4.80**	**39.67 ± 4.51**	9.46 ± 1.18	1.43
NCN-based	NCN	31.72 ± 4.48	58.03 ± 3.45	38.26 ± 2.56	27.36 ± 5.00	39.85 ± 5.07	36.81 ± 3.29	8.33 ± 0.92	–
	UniAug - NCN	**35.92 ± 7.85**	**61.69 ± 3.21**	**40.30 ± 2.53**	**30.20 ± 1.46**	**42.11 ± 5.74**	**39.26 ± 2.84**	**8.85 ± 0.90**	–

Table 4: Mean and standard deviation of accuracy (%) across 10 splits on node classification of heterophilic graphs. The best result is **bold**. The last column is the average rank of each method.

		Cornell	Wisconsin	Texas	Actor	Chameleon*	Squirrel*	A.R.
	GCN	59.41 ± 6.03	51.68 ± 4.34	63.78 ± 4.80	30.58 ± 1.29	40.94 ± 3.91	39.11 ± 1.74	4.50
Self-supervised	MVGRL	56.19 ± 2.42	50.64 ± 5.89	61.70 ± 3.94	31.37 ± 0.83	32.34 ± 2.11	35.32 ± 1.32	6.83
	GRACE	56.39 ± 2.11	53.83 ± 3.56	63.54 ± 2.57	28.14 ± 0.81	35.71 ± 1.95	33.65 ± 2.51	7.00
	BGRL	56.67 ± 2.13	59.80 ± 4.08	65.78 ± 2.66	29.80 ± 0.31	37.01 ± 2.89	34.77 ± 2.01	5.33
	GraphMAE	57.31 ± 2.11	58.27 ± 2.91	58.34 ± 3.57	28.97 ± 0.27	36.75 ± 1.78	39.13 ± 2.01	5.50
GDA	Half-Hop	62.46 ± 7.58	76.47 ± 2.61	72.35 ± 4.27	33.95 ± 0.68	38.59 ± 2.89	37.34 ± 2.18	3.17
	UniAug	68.11 ± 6.72	69.02 ± 4.96	73.51 ± 5.06	33.11 ± 1.57	**43.84 ± 3.39**	**41.90 ± 1.90**	2.00
	UniAug + Half-Hop	**72.43 ± 5.81**	**79.61 ± 5.56**	**77.03 ± 4.27**	**34.97 ± 0.55**	41.94 ± 2.77	38.79 ± 2.61	1.67

Baselines. We evaluate our model against three main groups of baselines. (1) Task-specific GNNs: For graph property prediction, we use GIN [17]; for link prediction, we use GCN [71] and NCN [72]; and for node classification, we use GCN [71]. (2) Graph pre-training methods: These include AttrMask, CtxtPred, EdgePred, and InfoMax [22], JOAO [25], D-SLA [24], and GraphMAE [23]. For each of these methods, we pre-train it on the same pre-training set as *UniAug*. While most of the pre-training graphs lack node features, we calculate the node degrees as the input. Each method consists of three pre-trained variants with different backbone GNNs, including GIN, GCN, and GAT. (3) Self-supervised methods: These include MVGRL [73], GRACE [74], BGRL [75] and GraphMAE [23]. For the self-supervised methods, we extract the node embeddings and feed them into downstream specific heads. (4) Graph data augmentation (GDA) methods: For graph property prediction, we include S-Mixup [34], GraphAug [35], FLAG [37], GREA [36], and DCT [27]; for link prediction, we include CFLP [31]; and for node classification on heterophilic graphs, we include Half-Hop [30]. The GDA methods are implemented based on chosen task-specific GNNs.

Graph property prediction. We employ graph label guidance for *UniAug* throughout the graph-level tasks by training a 2-layer MLP as the guidance head on the graph labels in the training set. In the augmentation stage, we generate multiple graphs per training sample, and the generated graphs are then fed into the baseline GIN. We present the results of molecule regression in Table 5 and graph classification in Table 2. Three key observations emerge from the analysis: (1) Existing pre-training methods show negative transfer compared to GIN. Some special cases are the Enzymes and molecule regression datasets, where all pre-training methods fail to yield satisfactory results. In these datasets, the features are one of the driving components for graph property prediction, while the pre-training methods fail to encode such information due to

Table 5: Mean and standard deviation of MAE ↓ across 10 runs on molecule regression. The last column is the average rank of each method. Among the methods, all pre-training methods discard atom and bond features due to dimension mismatch and we include the best-performing method JOAO into comparison; GIN and *UniAug* remove the bond features; others incorporate both.

	ogbg-Lipo	ogbg-ESOL	ogbg-FreeSolv	A.R.
GINE*	0.545 ± 0.019	0.766 ± 0.016	1.639 ± 0.146	5.00
GIN	0.543 ± 0.021	0.729 ± 0.018	1.613 ± 0.155	3.67
JOAO	0.859 ± 0.007	1.458 ± 0.040	3.292 ± 0.117	7.00
FLAG*	0.528 ± 0.012	0.755 ± 0.039	1.565 ± 0.098	3.00
GREA*	0.586 ± 0.036	0.805 ± 0.135	1.829 ± 0.368	6.00
DCT*	0.516 ± 0.071	0.717 ± 0.020	1.339 ± 0.075	1.33
UniAug	0.528 ± 0.006	0.677 ± 0.026	1.448 ± 0.049	1.67

*Results are taken from DCT [27].

incompatibility with the feature dimension. This reveals one critical drawback of the pre-training methods: their inability to handle feature heterogeneity. (2) GDA methods yield inconsistent results across different datasets. While these methods enhance performance in some datasets, they cause performance declines in others. This variability is directly reflected in the average rank, where some of them even fall behind the GIN. (3) Unlike the pre-training methods and GDA methods, *UniAug* shows consistent performance improvements against GIN with a large margin. In the molecule regression tasks, *UniAug* effectively compensates for the absence of bond features and achieves performance comparable to DCT, which is a data augmentation method pre-trained on in-domain molecule graphs. We also adapt the *semi-supervised* [76] and *self-supervised* [77] setting for the baselines for a comprehensive benchmark in Appendix D.1 Table 13, where we observe that UniAug presents consistently satisfactory performance according to the average rank, matching or outperforming the best baseline. These findings affirm that the pre-training and structure augmentation paradigm of *UniAug* effectively benefits the downstream datasets at the graph level.

Link prediction. We choose three guidance objectives for *UniAug*, including node degree, CN, and link prediction objective, as described in Section 3.3. For each objective, we train an MLP to provide guidance information. We then augment the graph structure by generating a synthetic graph and preserving the original training edges, ensuring that the augmented graph does not remove any existing edges. The augmented graph is then fed into a GCN for link prediction. We summarize the results in Table 3, where we observe that (1) GDA method CFLP leads to performance drops on the datasets without features and also suffers from high computation complexity during preprocessing. (2) *UniAug* enhances performance across all tested datasets. In addition, we employ *UniAug* to NCN [72], one of the state-of-the-art methods for link prediction. The results demonstrate consistent performance boosts from *UniAug* when we apply NCN as the backbone. The structure augmentation paradigm of *UniAug* allows plug-and-play applications to any downstream-specific models, showcasing its adaptability and effectiveness. Additional results, including the performance of pre-training baselines and the effects of three guidance objectives, can be found in Appendix D.2.

Node classification. To demonstrate the effectiveness of *UniAug* in node-level tasks, we transform the node classification into subgraph classification. Specifically, we extract the aggregation tree of each node, i.e., 2-hop subgraph for a 2-layer GCN, and label the subgraph with the center node. We then adopt a strategy similar to graph classification and train a 2-layer classifier as a guidance head. Inspired by the success of structure augmentation on heterophilic graphs [78, 30], we evaluate *UniAug* on 6 heterophilic datasets. We observe phenomena similar to those seen in graph- and link-level tasks in Table 4, with results of pre-training baselines in Appendix D.3. One thing to mention is the combination of *UniAug* and Half-Hop. Half-Hop offers performance improvements in four out of six datasets via data augmentation, and combining it with *UniAug* yields even higher results. This highlights the flexibility of *UniAug* and opens up possibilities for further exploration of its use cases. Given the impressive results of *UniAug* on heterophilic graphs, we anticipate it will also help to balance the performance disparities among nodes with different homophily ratios on homophilic graphs [16]. We split the nodes into five groups according to their homophily ratios and calculate the standard deviation (SD) across groups. As shown in Table 6, *UniAug* matches the performance of vanilla GCN and also reduces the performance discrepancies corresponding to SD.

Table 6: Results of node classification on homophily graphs. Results are scaled $\times 100$.

		Cora	Citeseer	Pubmed
ACC ↑	GCN	81.75 ± 0.73	70.71 ± 0.76	79.53 ± 0.25
	UniAug	81.78 ± 0.60	71.17 ± 0.58	79.54 ± 0.35
SD ↓	GCN	24.51 ± 1.06	22.57 ± 0.80	27.02 ± 0.56
	UniAug	**23.45 ± 0.90**	**19.90 ± 0.81**	**26.50 ± 0.55**

4.2 Scaling behavior of *UniAug*

In light of the neural scaling law [4, 5, 6, 7, 79], we expect *UniAug* to benefit from an increased coverage of data and more compute budget. In this subsection, we investigate the scaling behavior of *UniAug* in terms of data scale and amount of compute for pre-training.

Data coverage During the data collection process, we prepare three versions of the training data with increasing magnitude and growing coverage on the graph distribution. We first sample 10 graphs per category from the Network Repository [8] to build a SMALL collection. Next, we gather all the graphs from the Network Repository and filter out large-scale graphs and outliers for a FULL

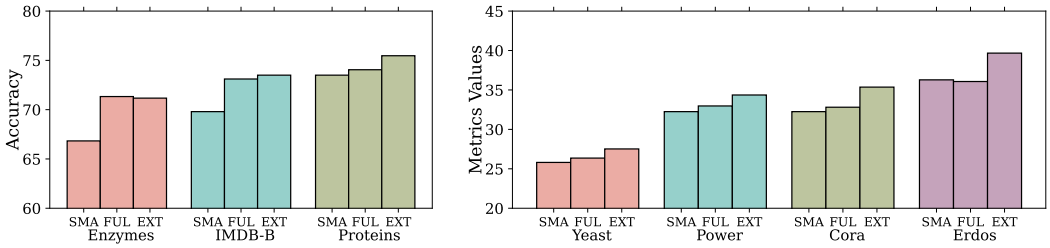

Figure 3: Effects of pre-training data scale on graph classification (left) and link prediction (right). The groups SMA, FUL, and EXT represent SMALL, FULL, and EXTRA data collection.

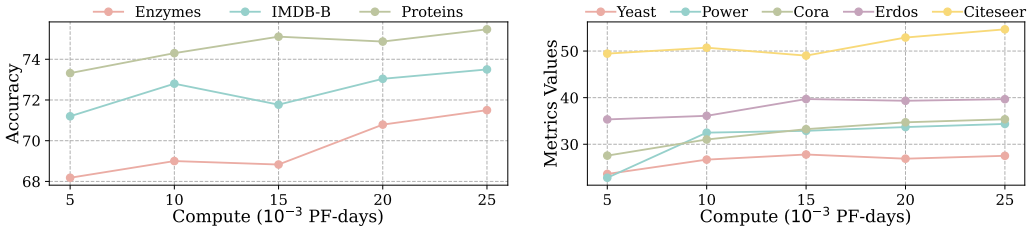

Figure 4: Effects of pre-training amount of compute on graph classification (left) and link prediction (right), where one PF-days $= 10^{15} \times 24 \times 3600 = 8.64 \times 10^{19}$ floating point operations.

collection. In addition, we add a 1000 graphs subset of the GitHub Star dataset from TUDataset [80] to enlarge the coverage of diverse patterns and form an EXTRA collection. We pre-train three versions of *UniAug* respectively on the three collections and evaluate them on graph classification and link prediction. As shown in Fig. 3, we observe a clear trend of increase in performance as we enlarge the coverage of pre-training data. This paves the way to scale up *UniAug* to even more pre-training datasets with an expanding distribution of graphs. Additional results on scaling effects with respect to the proportion of the full data can be found in Appendix D.4.

Amount of compute We sought to understand how effectively our diffusion model can learn data patterns as we continue to train it. To this end, we checkpointed *UniAug* every 2,000 epochs (5×10^{-3} PF-days) while training on the EXTRA collection, and then applied it to graph classification and link prediction tasks. The results are illustrated in Fig. 4. We observe that downstream performance generally improves with prolonged training, while the trend slows down for some datasets when we reach 8,000 epochs. We take the checkpoint at the 10,000th epoch for evaluations. Given the scaling behavior observed, we anticipate *UniAug* to become even more effective with additional resources.

4.3 Preventing negative transfer

In the previous parts of the experiments, we showcase the positive transfer of *UniAug* across different tasks. We now investigate which aspects of the design prevent negative transfer. *UniAug* consists of two main components: a pre-trained diffusion model and the structure augmentation through guided generation. In the pre-training process, we inject self-supervised graph labels into the diffusion model and we wonder about the performance of its unconditioned counterpart. Regarding the augmentation process, we examine the impact

Table 7: Demonstration of negative transfer on graph classification (up) and link prediction (down).

	Enzymes	Proteins	IMDB-B	IMDB-M
GIN	66.00 ± 7.52	73.32 ± 4.03	71.10 ± 2.90	49.07 ± 2.81
UniAug	71.50 ± 5.85	75.47 ± 2.50	73.50 ± 2.48	50.13 ± 2.05
w/o self-cond	71.11 ± 7.50	73.31 ± 4.63	71.50 ± 2.27	49.00 ± 2.74
w/o guidance	62.17 ± 3.93	71.15 ± 4.56	53.80 ± 3.29	35.33 ± 3.17
w/ cross-guide	51.50 ± 7.64	72.46 ± 4.35	71.10 ± 2.38	49.20 ± 2.59

	Cora MRR	Citeseer MRR	Power Hits@10	Yeast Hits@10	Erdos Hits@10
GCN	30.26 ± 4.80	50.57 ± 7.91	30.61 ± 4.07	24.71 ± 4.92	35.71 ± 2.65
UniAug	35.36 ± 7.88	54.66 ± 4.55	34.36 ± 1.68	27.52 ± 4.80	39.67 ± 4.51
w/o self-cond	27.97 ± 16.11	37.65 ± 6.00	28.95 ± 7.73	23.54 ± 8.28	34.33 ± 6.18
w/o guidance	29.60 ± 6.06	51.41 ± 7.10	25.57 ± 6.04	25.26 ± 6.06	37.11 ± 4.16
w/ cross-guide	32.37 ± 4.20	50.59 ± 5.67	32.99 ± 2.54	26.76 ± 3.88	36.30 ± 3.67

of diffusion guidance by exploring outcomes when the guidance is either removed or applied using another dataset from a different domain (cross-guide). We summarize the results in Table 7 for graph classification and link prediction. All modifications investigated lead to performance declines in both tasks. We observe that removing guidance results in significant negative transfers for graph classification, while the effects of self-conditioning are more pronounced for link prediction. We conclude that both the self-conditioning strategy and diffusion guidance are crucial in preventing negative transfer, underscoring their importance in the design of *UniAug*.

5 Conclusion and Discussion

In this work, we propose a graph structure augmentor *UniAug* to leverage the increasing scale of graph data. We collect thousands of graphs from various domains and pre-train a self-conditioned discrete diffusion model on them. In the downstream stage, we augment the graphs by preserving the original node features and generating synthetic structures. We apply *UniAug* to node-, link-, and graph-level tasks and achieve consistent performance gain. We have successfully developed a showcase that benefits from cross-domain graph data scaling using diffusion models.

One limitation of the current analysis is the absence of an investigation into the effects of model parameters due to limited resources. Given the scaling behavior of *UniAug* in terms of data scale and amount of compute, we anticipate that a large-scale model will provide significant performance improvements. One future direction is to investigate the adaptation of fast sampling methods to the discrete diffusion models on graphs. This will lead to lower time complexity and enable broader application scenarios.

6 Impact Statements

In this work, we build a universal graph structure augmentor that benefits from data scaling across domains. Given the consistent performance improvements for different tasks, we expect this work to contribute significantly towards the goal of building a graph foundation model. In the meantime, we showcase the power of the deep generative models on graphs by introducing new application scenarios. We anticipate such success will contribute to the community of generative models and graph learning.

It is important to mention that the model backbones of our method and baselines heavily rely on neighboring node information as an inductive bias. However, this characteristic can result in biased predictions, especially when patterns in neighborhood majorities dominate, leading to potential ethical issues in model predictions.

7 Acknowledgment

W.T. and J.T. are supported by the National Science Foundation (NSF) under grant numbers CNS2321416, IIS2212032, IIS2212144, IIS 2504089, DUE2234015, CNS2246050, DRL2405483 and IOS2035472, the Michigan Department of Agriculture and Rural Development, US Dept of Commerce, Amazon Faculty Award, Meta, NVIDIA, Microsoft and SNAP. W.T. and Y.X. are supported by the National Institutes of Health (NIH) under grant numbers U01DE033330, R01HL166508, R01DE026728 and the NSF under grant numbers OISE2434687 and IOS2107215.

8 Disclaimer

References

[1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[2] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[3] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[4] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[5] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[6] Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. Exploring the limits of large scale pre-training. In *International Conference on Learning Representations*, 2022.

[7] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022.

[8] Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[9] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[10] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[11] Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. Mole-BERT: Rethinking pre-training graph neural networks for molecules. In *The Eleventh International Conference on Learning Representations*, 2023.

[12] Zhiyuan Liu, Yaorui Shi, An Zhang, Enzhi Zhang, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. Rethinking tokenizer and decoder in masked graph modeling for molecules. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] Yuxuan Cao, Jiarong Xu, Carl Yang, Jiaan Wang, Yunchao Zhang, Chunping Wang, Lei Chen, and Yang Yang. When to pre-train graph neural networks? an answer from data generation perspective! *arXiv preprint arXiv:2303.16458*, 2023.

[14] Haitao Mao, Juanhui Li, Harry Shomer, Bingheng Li, Wenqi Fan, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Revisiting link prediction: a data perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[15] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[16] Haitao Mao, Zhikai Chen, Wei Jin, Haoyu Han, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Demystifying structural disparity in graph neural networks: Can one size fit all? *Advances in Neural Information Processing Systems*, 36, 2024.

[17] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[18] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Michael Galkin, and Jiliang Tang. Graph foundation models. *arXiv preprint arXiv:2402.02216*, 2024.

[19] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[20] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[21] Nate Gruver, Samuel Stanton, Nathan Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew G Wilson. Protein design with guided discrete diffusion. *Advances in Neural Information Processing Systems*, 36, 2024.

[22] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[23] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2022.

[24] Dongki Kim, Jinheon Baek, and Sung Ju Hwang. Graph self-supervised learning with accurate discrepancy learning. *Advances in Neural Information Processing Systems*, 35:14085–14098, 2022.

[25] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *International Conference on Machine Learning*, pages 12121–12132. PMLR, 2021.

[26] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. Self-supervised graph-level representation learning with local and global structure. In *International Conference on Machine Learning*, pages 11548–11558. PMLR, 2021.

[27] Gang Liu, Eric Inae, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Data-centric learning from unlabeled graphs with diffusion model. *Advances in neural information processing systems*, 36, 2024.

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

[29] Songtao Liu, Rex Ying, Hanze Dong, Lanqing Li, Tingyang Xu, Yu Rong, Peilin Zhao, Junzhou Huang, and Dinghao Wu. Local augmentation for graph neural networks. In *International conference on machine learning*, pages 14054–14072. PMLR, 2022.

[30] Mehdi Azabou, Venkataramana Ganesh, Shantanu Thakoor, Chi-Heng Lin, Lakshmi Sathidevi, Ran Liu, Michal Valko, Petar Veličković, and Eva L Dyer. Half-hop: A graph upsampling approach for slowing down message passing. In *International Conference on Machine Learning*, pages 1341–1360. PMLR, 2023.

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

[32] Trung-Kien Nguyen and Yuan Fang. Diffusion-based negative sampling on graphs for link prediction. *arXiv preprint arXiv:2403.17259*, 2024.

[33] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. G-mixup: Graph data augmentation for graph classification. In *International Conference on Machine Learning*, pages 8230–8248. PMLR, 2022.

[34] Hongyi Ling, Zhimeng Jiang, Meng Liu, Shuiwang Ji, and Na Zou. Graph mixup with soft alignments. In *International Conference on Machine Learning*, pages 21335–21349. PMLR, 2023.

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

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

[37] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. Robust optimization as data augmentation for large-scale graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 60–69, 2022.

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

[39] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[40] 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.

[41] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020.

[42] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, pages 10362–10383. PMLR, 2022.

[43] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. In *International conference on machine learning*, pages 9558–9568. PMLR, 2021.

[44] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022.

[45] Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

[46] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023.

[47] Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *International Conference on Machine Learning*, pages 4585–4610. PMLR, 2023.

[48] Jialin Chen, Shirley Wu, Abhijit Gupta, and Zhitao Ying. D4explainer: In-distribution explanations of graph neural network via discrete denoising diffusion. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[49] 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.

[50] Jiarong Xu, Renhong Huang, Xin Jiang, Yuxuan Cao, Carl Yang, Chunping Wang, and Yang Yang. Better with less: A data-active perspective on pre-training graph neural networks. *Advances in Neural Information Processing Systems*, 36:56946–56978, 2023.

[51] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. Karate club: an api oriented open-source python framework for unsupervised learning on graphs. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 3125–3132, 2020.

[52] Brandon Trabucco, Kyle Doherty, Max A Gurinas, and Ruslan Salakhutdinov. Effective data augmentation with diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

[53] Zebin You, Yong Zhong, Fan Bao, Jiacheng Sun, Chongxuan Li, and Jun Zhu. Diffusion models and semi-supervised learners benefit mutually with few labels. *Advances in Neural Information Processing Systems*, 36, 2024.

[54] Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and XIAOJUAN QI. Is synthetic data from generative models ready for image recognition? In *The Eleventh International Conference on Learning Representations*, 2023.

[55] Fan Bao, Chongxuan Li, Jiacheng Sun, and Jun Zhu. Why are conditional generative models better than unconditional ones? *arXiv preprint arXiv:2212.00362*, 2022.

[56] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

[57] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

[58] Shanghua Gao, Zhong-Yu Li, Ming-Hsuan Yang, Ming-Ming Cheng, Junwei Han, and Philip Torr. Large-scale unsupervised semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[59] Vincent Tao Hu, David W Zhang, Yuki M Asano, Gertjan J Burghouts, and Cees GM Snoek. Self-guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18413–18422, 2023.

[60] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

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

[62] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020.

[63] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

[64] Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Controllability of complex networks. *nature*, 473(7346):167–173, 2011.

[65] Stephen P Borgatti. Centrality and network flow. *Social networks*, 27(1):55–71, 2005.

[66] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[67] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[68] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543, 2002.

[69] Shiyin Tan, Dongyuan Li, Renhe Jiang, Ying Zhang, and Manabu Okumura. Community-invariant graph contrastive learning. *arXiv preprint arXiv:2405.01350*, 2024.

[70] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in neural information processing systems*, 33:12559–12571, 2020.

[71] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[72] Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *The Twelfth International Conference on Learning Representations*, 2024.

[73] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pages 4116–4126. PMLR, 2020.

[74] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

[75] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*, 2022.

[76] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. L2-gcn: Layer-wise and learned efficient training of graph convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2127–2135, 2020.

[77] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020.

[78] Wendong Bi, Lun Du, Qiang Fu, Yanlin Wang, Shi Han, and Dongmei Zhang. Make heterophily graphs better fit gnn: A graph rewiring approach. *arXiv preprint arXiv:2209.08264*, 2022.

[79] Jingzhe Liu, Haitao Mao, Zhikai Chen, Tong Zhao, Neil Shah, and Jiliang Tang. Neural scaling laws on graphs. *arXiv preprint arXiv:2402.02054*, 2024.

[80] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[81] Ze Wang, Jiang Wang, Zicheng Liu, and Qiang Qiu. Binary latent diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22576–22585, 2023.

[82] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[83] Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1):D431–D433, 2004.

[84] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14:347–375, 2008.

[85] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

[86] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[87] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[88] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, et al. Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic acids research*, 31(9):2443–2450, 2003.

[89] Vladimir Batagelj and Andrej Mrvar. Pajek datasets. http://vlado.fmf.uni-lj.si/pub/networks/data/, 2006.

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

[91] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[92] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816, 2009.

[93] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023.

[94] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[95] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020.

[96] Zhikai Chen, Haitao Mao, Jingzhe Liu, Yu Song, Bingheng Li, Wei Jin, Bahare Fatemi, Anton Tsitsulin, Bryan Perozzi, Hui Liu, et al. Text-space graph foundation models: Comprehensive benchmarks and new insights. *arXiv preprint arXiv:2406.10727*, 2024.

[97] Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4443–4454, 2024.

[98] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, pages 417–428, 2023.

[99] Quang Truong and Peter Chin. Weisfeiler and lehman go paths: Learning topological features via path complexes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15382–15391, 2024.

[100] 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.

[101] WANG Botao, Jia Li, Yang Liu, Jiashun Cheng, Yu Rong, Wenjia Wang, and Fugee Tsung. Deep insights into noisy pseudo labeling on graph data. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[102] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[103] 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.

[104] Jingyu Chen, Runlin Lei, and Zhewei Wei. PolyGCL: GRAPH CONTRASTIVE LEARNING via learnable spectral polynomial filters. In *The Twelfth International Conference on Learning Representations*, 2024.

A Derivation of Diffusion Process

In the following, we will formulate the existing discrete diffusion models into binary diffusion on the adjacency matrix. We denote the adjacency matrix of a graph as $\mathbf{A}^0 \in \{0,1\}^{n\times n}$ with n nodes. Following D3PM [49], we corrupt the adjacency matrix into a sequence of latent variables $\mathbf{A}^{1:T} = \mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^T$ by independently injecting noise into each element with a Markov process

$$q\left(\mathbf{A}^t \mid \mathbf{A}^{t-1}\right) = \prod_{i,j:i<j} \operatorname{Cat}\left(\mathbf{A}_{ij}^t; \mathbf{p} = \mathbf{A}_{ij}^{t-1}\mathbf{Q}^t\right), \tag{6}$$

where $\mathbf{Q}^t \in [0,1]^{2\times 2}$ is the transition probability of timestep t. The above Markov process is called *forward process*. Existing works provide different designs for the transition matrix $\mathbf{Q}^t$, including

$$\begin{aligned}
\text{Uniform [48]}: &\begin{pmatrix} 1-\beta^t & \beta^t \\ \beta^t & 1-\beta^t \end{pmatrix}; \\
\text{Absorbing [47]}: &\begin{pmatrix} 1 & 0 \\ \beta^t & 1-\beta^t \end{pmatrix}; \\
\text{Predefined [46]}: &\begin{pmatrix} 1-\beta^t\cdot\pi & \beta^t\cdot p \\ (1-\pi)\beta^t & 1-(1-\pi)\beta^t \end{pmatrix},
\end{aligned} \tag{7}$$

where π is the converging non-zero probability and β^t is the noise scale. All three transition matrices can be written as binary diffusion with Bernoulli distribution

$$\begin{aligned}
q\left(\mathbf{A}^t \mid \mathbf{A}^{t-1}\right) &= \text{Bernoulli}\left(\mathbf{A}^t; \alpha^t\mathbf{A}^{t-1} + \left(1-\alpha^t\right)\pi\right), \\
q\left(\mathbf{A}^t \mid \mathbf{A}^{0}\right) &= \text{Bernoulli}\left(\mathbf{A}^t; \bar{\alpha}^t\mathbf{A}^{0} + \left(1-\bar{\alpha}^t\right)\pi\right), \\
q\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t, \mathbf{A}^0\right) &= \frac{q\left(\mathbf{A}^t \mid \mathbf{A}^{t-1}\right) q\left(\mathbf{A}^{t-1} \mid \mathbf{A}^0\right)}{q\left(\mathbf{A}^t \mid \mathbf{A}^0\right)},
\end{aligned} \tag{8}$$

where $\alpha^t = 1-\beta^t$ and $\bar{\alpha}^t = \prod_{i=1}^t \alpha^i$. The prior $\mathbf{A}^T$ is determined by π with $p\left(\mathbf{A}_{ij}^T\right) = \text{Bernoulli}(\pi)$, i.e., the existence of each edge follows a Bernoulli distribution with probability π. The main difference of the *forward process* among the existing works is the choice of π, where $\pi = 0$ for EDGE [47], $\pi = 0.5$ for D4Explainer [48], and a pre-computed average density π for DiGress [46].

In our early experiments, we observe that the absorbing kernel $\pi = 0$ surpasses the other two in terms of efficiency and effectiveness for graph generation. The *forward process* with non-zero π will add non-existing edges, which brings in additional computations. When sampling from prior, non-zero π will introduce additional uncertainty because we will first sample every edge from $\text{Bernoulli}(\pi)$. Therefore, we choose the absorbing prior $\pi = 0$ in this work and leave the exploration of other transition kernels as a future work.

We note that in our implementation, we choose the number of timesteps T as 128 according to our early experiments and some existing works [81, 47]. We leave the study of the effects of diffusion timesteps on downstream tasks as a future work.

B Guidance Objective for Downstream Tasks

We mention various guidance objectives in Section 3.3 with different granularity. Here, we specify the objectives we use for each downstream task. Our empirical results suggest that supervision signals will lead to better performance. Thus, we use node labels for node classification and graph labels for graph property prediction in Section 4. Regarding link prediction, we anticipate that both node-level and edge-level objectives may help the downstream adaptation. Therefore, we choose three objectives including node degree, CN heuristic, and link prediction objective.

C Datasets

The license of the datasets use in this work is in Table 8.

Graph property prediction datasets include DD and Proteins [82], Enzymes [83], NCI1 [84], IMDB-Binary, IMDB-Multi, Reddit-Binary, and Reddit-Multi-12K [85], ogbg-Lipo, ogbg-ESOL and ogbg-FreeSolv [9]. The statistics are summarized in 9.

Table 8: List of datasets and corresponding License

Dataset	License
Network Repository	CC BY-SA
Github Star	CC BY 4.0
Cora	NLM license
Citeseer	NLM license
Pubmed	NLM license
WebKB	MIT license
Wikipedia Network	MIT license
Actor	MIT license
Power	BSD License
Yeast	BSD License
Erdos	BSD License
Amazon Photo	MIT license
Flickr	MIT license
DD	CC BY 4.0
Enzymes	CC BY 4.0
Proteins	CC BY 4.0
NCI1	CC BY 4.0
IMDB	CC BY 4.0
Reddit	CC BY 4.0

Table 9: Statistics of graph property prediction datasets.

Domain	Dataset	Task type	# Graphs	# Tasks	# Nodes	# Edges
Biology	DD	Classification	1,178	2	284	716
	Enzymes	Classification	600	6	33	64
	Proteins	Classification	1,113	2	40	73
Academic	Collab	Classification	5,000	3	74	2458
Social	IMDB-B	Classification	1,000	2	20	97
	IMDB-M	Classification	1,500	3	13	66
	Reddit-5k	Classification	4,999	5	509	595
	Reddit-12k	Classification	11,929	11	391	1305
Chemical	NCI	Classification	4,110	2	30	32
	ogbg-Lipo	Regression	4200	1	27	59
	ogbg-ESOL	Regression	1128	1	13	27
	ogbg-FreeSolv	Regression	642	1	9	17

Link prediction datasets include Cora, Citeseer, and Pubmed [86], Power [87], Yeast [88], Erdos [89], Amazon Photo [90], and Flickr [10]. The statistics are summarized in 10.

Table 10: Statistics of link prediction datasets.

	Cora	Citeseer	Pubmed	Power	YST	ERD	Flickr
Domain	Citation			Transport	Biology	Academic	Social
#Nodes	2,708	3,327	18,717	4,941	2,284	6,927	334,863
#Edges	5,278	4,676	44,327	6,594	6,646	11,850	899,756
Mean Degree	3.9	2.81	4.74	2.67	5.82	3.42	5.69

Node classification datasets include Cora, Citeseer, and Pubmed [86], WebKB (Texas, Cornell, and Wisconsin) [91], Wikipedia Network (Chameleon and Squirrel) [91], and Actor [92]. The first three are homophilic graphs, and the others are heterophilic. The statistics are summarized in 11.

D Experiment

In this section, we introduce the implementation details and additional results for the experiments. Throughout all the experiments, we train all the methods with Adam optimizer on an A100 GPU. We train the guidance head of *UniAug* with cross-entropy loss for class labels and mean squared error loss for all other objectives. For multi-class objectives, we apply the label smoothing [94] technique

Table 11: Statistics of node classification datasets.

	Cora	Citeseer	Pubmed	Cornell	Wisconsin	Texas	Chameleon*	Squirrel*	Actor
Domain	Citation			Web					Social
#Nodes	2,708	3,327	19,717	183	251	183	890	2,223	7,600
#Edges	5,278	4,676	44,324	295	499	309	8,854	46,998	33,544
#Classes	7	6	3	5	5	5	5	5	5

*Chameleon and Squirrel are filtered to remove duplicated nodes [93].

following NOS [21]. Denote y as the one-hot label and C as the number of classes, we have

$$\overline{\mathbf{y}}_t = \bar{\alpha}_t * \mathbf{y} + (1 - \bar{\alpha}_t) / C * \mathbf{1}. \tag{9}$$

D.1 Graph property prediction

For graph classification, we follow [95] for the setting with 10-fold cross-validation. We utilize a 5-layer GIN with latent dimensions of 64 throughout the datasets. For molecule regression, we implement a 5-layer GIN with a virtual node, and the latent dimensions are 300. We have mainly four hyperparameters for *UniAug*: step-size γ and regularization strength λ in (5), number of repeats per training graph, and whether augment validation and test graphs with the trained guidance head. For each training graph, we repeatedly generate structures and plug in the original node features for multi-repeat augmentation. We perform the update in (5) for 5 times per each sampling step. The hyperparameters are tuned from the choices in Table 12.

Table 12: Hyperparameter choices for graph property prediction.

λ	0.01
γ	[0.1, 0.5, 1.0]
# repeats	[1, 5, 10, 32, 64]
Aug val and test	[True, False]

In Section 4.1, we aim to benchmark the capability of cross-domain pre-training of different methods on the same set of pre-training graphs. While the pre-training graphs contain vastly different features, we have to align the feature space to allow pre-training for the baseline methods. There are two ways to tackle the feature heterogeneity issues in the existing literature. One line of them utilizes LLMs to align text-space graphs [96], which is not applicable to broader classes of graphs. Other works, like GCOPE [97], perform dimension reduction to align the feature dimension of different graphs. We emphasize that dimension reduction methods fail to deal with extreme cases like missing features. This phenomenon is pretty common in real life, as a large proportion of the graphs in the Network Repository do not have corresponding features. Therefore, we simply use the node degrees as the features in Section 4.1.

We understand that removing the node features may result in a performance drop for the baseline methods. Note that most of the baselines follow the pre-training paradigm of [22] with domain-specific model designs for chemistry and biology datasets, and thus cannot be directly applied to the chosen graph classification datasets. Therefore, we adapt the **semi-supervised** [76] and **self-supervised** [77] setting for the baselines for a comprehensive benchmark. The semi-supervised setting involves pre-training with all data of that specific dataset and finetuning the training set of each split. Meanwhile, baselines of the self-supervised setting pre-train on the whole dataset and then classify the learned graph embeddings with a downstream SVM classifier. The results are summarized in Table 13, where the best and second-best results are highlighted in **bold** and *italic*, respectively. We observe that UniAug presents consistently satisfactory performance according to the average rank, matching or outperforming the best baseline.

To demonstrate the effectiveness of *UniAug* in scenarios with limited labeled data, we perform 5-shot graph classification following [98]. The results are summarized in Table 14 These results show that *UniAug* achieves significant performance improvements over the self-supervised baselines, underscoring its robustness and adaptability in few-shot settings.

In addition, to showcase the flexibility of *UniAug* on the downstream backbone, we pick one of the SOTA method PIN [99] for graph classification and evaluate *UniAug* on the basis of it. The results are summarized in Table 15, where we see *UniAug* offers constant improvements over PIN.

Table 13: Mean and standard deviation of accuracy (%) with 10-fold cross-validation on graph classification. The best and second-best results are highlighted in **bold** and *italic*. The last column is the average rank.

		DD	Proteins	NCI1	IMDB-B	IMDB-M	Reddit-B	Collab	A.R.
Semi-supervised	CtxtPred	74.66±0.51	70.23±0.63	73.00±0.30	–	–	88.66±0.95	73.69±0.37	6.80
	InfoMax	75.78±0.34	72.27±0.40	74.86±0.26	–	–	88.66±0.95	73.76±0.29	5.60
	GraphCL	*76.17±1.37*	74.17±0.34	74.63±0.25	–	–	89.11±0.19	74.23±0.21	4.60
	JOAO	75.81±0.73	73.31±0.48	74.86±0.39	–	–	88.79±0.65	75.53±0.18	4.60
Self-supervised	InfoGraph	–	74.44±0.31	76.20±1.06	73.03±0.87	49.69±0.53	82.50±1.42	70.65±1.13	5.17
	GraphCL	–	74.39±0.45	77.87±0.41	71.14±0.44	48.58±0.67	*89.53±0.84*	71.36±1.15	4.50
	JOAO	–	74.55±0.41	78.07±0.47	70.21±3.08	49.20±0.77	85.29±1.35	69.50±0.36	5.17
	GraphMAE	–	*75.30±0.39*	*80.40±0.30*	**75.52±0.66**	**51.63±0.52**	88.01±0.19	**80.32±0.46**	2.17
	UniAug	**78.13±2.61**	**75.47±2.50**	**80.54±1.77**	*73.50±2.48*	*50.13±2.05*	**92.28±1.59**	*77.00±2.02*	1.43

Table 14: Accuracy of 5-shot graph classification.

	Proteins	Enzymes
GIN	58.17 ± 8.58	20.34 ± 5.01
InfoGraph	54.12 ± 8.20	20.90 ± 3.32
GraphCL	56.38 ± 7.24	28.11 ± 4.00
JOAO	57.21 ± 6.91	35.31 ± 3.79
GraphMAE	60.03 ± 5.35	33.91 ± 6.58
UniAug	**66.85 ± 4.71**	**48.37 ± 4.77**

Table 15: Accuracy of graph classification with PIN.

	Proteins	NCI1	IMDB-B
PIN	78.8 ± 4.4	85.1 ± 1.5	76.6 ± 2.9
UniAug - PIN	**80.2 ± 2.8**	**86.5 ± 1.4**	**77.9 ± 1.8**

D.2 Link prediction

For link prediction, we follow the model designs and evaluation protocols of [100]. For results based on GCN and NCN, we use a GCN encoder to produce node embeddings and perform link prediction with a prediction head. The prediction head of GCN is a 3-layer MLP. The number of layers and the latent dimension of the GCN encoder are taken from [100]. We have mainly three hyperparameters for *UniAug*: step-size γ and regularization strength λ, and the number of updates in (5) per each sampling step. In addition, inspired by the pseudo labeling strategy [101], we provide an option threshold q for the sampling process of the diffusion model. Specifically, we only keep the edges with the probability of existence higher than q for each sampling step. After the sampling process, we recover the training edges of the original graph structure. The hyperparameters are tuned from the choices in Table 16. One thing to mention is that we handle the large graphs by graph partitioning with METIS [102]. Specifically, we augment the partitions of a large graph and then assemble the partitions back into a single graph. The edges between different partitions are recovered after the assembling process.

Table 16: Hyperparameter choices for link prediction.

λ	[0.01, 1, 100]
γ	[0.1, 1.0, 10.0]
q	[None, 0.9, 0.99, 0.999]
# updates	[5, 10, 20]

In addition to the results shown in Table 3, we have the pre-training baselines as mentioned in Section 4.1. The results are summarized in Table 17. We observe that existing pre-training methods provide negative transfer, especially on datasets with node features. More explanation on removing the node features can be found in Appendix D.1.

As mentioned in Section 4.1, we choose three guidance objectives for link prediction with different granularity. The effects of different objectives can be found in Table 18. We observe that the outcomes of different objectives differ across datasets and there is no consistently winning strategy.

Table 17: Mean and standard deviation across 10 runs on link prediction. Results are scaled $\times 100$. The last two methods are based on NCN, while the rest are GCN-based. The best result is **bold** for two backbones, respectively. The highlighted results indicate negative transfer for pre-training methods compared to GCN. The last column is the average rank of each GCN-based method.

	Cora MRR	Citeseer MRR	Pubmed MRR	Power Hits@10	Yeast Hits@10	Erdos Hits@10	Flickr Hits@10	A.R.
GCN	30.26 ± 4.80	50.57 ± 7.91	16.38 ± 1.30	30.61 ± 4.07	24.71 ± 4.92	35.71 ± 2.65	8.10 ± 2.58	4.14
AttrMask	13.43 ± 1.93	20.23 ± 1.29	16.39 ± 3.62	29.92 ± 2.61	25.10 ± 4.77	30.85 ± 3.13	8.77 ± 1.65	6.43
CtxtPred	15.68 ± 2.91	22.31 ± 1.31	13.10 ± 3.70	29.30 ± 3.55	22.96 ± 4.28	34.82 ± 2.55	3.61 ± 1.01	7.86
EdgePred	15.31 ± 3.54	22.91 ± 1.87	17.85 ± 4.45	29.54 ± 3.78	25.78 ± 4.51	34.65 ± 3.84	6.86 ± 3.24	5.43
InfoMax	16.35 ± 2.57	22.90 ± 1.30	15.91 ± 2.71	29.29 ± 4.72	26.33 ± 4.12	35.82 ± 4.12	3.23 ± 0.38	6.00
JOAO	17.21 ± 3.66	23.10 ± 1.41	15.33 ± 3.70	28.98 ± 4.01	26.47 ± 4.65	33.77 ± 3.05	6.01 ± 1.57	6.00
D-SLA	15.55 ± 3.12	23.05 ± 1.54	16.10 ± 3.96	29.37 ± 2.88	26.15 ± 3.32	36.02 ± 4.58	6.70 ± 2.03	5.29
GraphMAE	15.94 ± 1.73	20.35 ± 1.52	13.80 ± 1.36	27.69 ± 1.99	26.51 ± 2.92	35.63 ± 3.61	8.41 ± 2.44	6.14
CFLP	33.62 ± 6.44	**55.20 ± 4.16**	17.01 ± 2.75	16.02 ± 8.31	24.23 ± 5.23	28.74 ± 2.38	OOM	6.43
UniAug-GCN	**35.36 ± 7.88**	54.66 ± 4.55	**17.28 ± 1.89**	**34.36 ± 1.68**	**27.52 ± 4.80**	**39.67 ± 4.51**	**9.46 ± 1.18**	1.29
NCN	31.72 ± 4.48	58.03 ± 3.45	38.26 ± 2.56	27.36 ± 5.00	39.85 ± 5.07	36.81 ± 3.29	8.33 ± 0.92	–
UniAug-NCN	**35.92 ± 7.85**	**61.69 ± 3.21**	**40.30 ± 2.53**	**30.20 ± 1.46**	**42.11 ± 5.74**	**39.26 ± 2.84**	**8.85 ± 0.90**	–

Table 18: Effects of different guidance objectives.

	Cora MRR	Citeseer MRR	Pubmed MRR	Power Hits@10	Yeast Hits@10	Erdos Hits@10	Flickr Hits@10
Link guide	30.45 ± 2.90	**54.66 ± 4.55**	16.97 ± 0.92	33.41 ± 2.95	25.80 ± 4.10	36.79 ± 1.98	**9.46 ± 1.18**
Degree guide	32.73 ± 6.71	51.13 ± 5.51	16.37 ± 0.58	32.88 ± 2.02	**27.52 ± 4.80**	**39.67 ± 4.51**	9.11 ± 0.88
CN guide	**35.36 ± 7.88**	50.86 ± 5.73	**17.28 ± 1.89**	**34.36 ± 1.68**	26.67 ± 4.02	36.18 ± 4.32	9.28 ± 1.18

D.3 Node classification

For node classification on heterophilic graphs, we use the fixed splits from Geom-GCN [91] for Cornell, Wisconsin, Texas, and Actor. For Chameleon and Squirrel, we remove duplicated nodes following [93] and take their fixed splits. Regarding node classification on homophilic graphs, we employ the semi-supervised setting [103]. The GCN backbone is implemented as a 2-layer classifier. Similar to graph property prediction, we have mainly four hyperparameters for *UniAug*: step-size γ and regularization strength λ in (5), number of repeats per training graph, and whether augment validation and test graphs with the trained guidance head. The hyperparameters are tuned from the choices in Table 19.

Table 19: Hyperparameter choices for node classification.

λ	0.01
γ	[0.1, 0.5, 1.0]
# repeats	[1, 5, 10]
Aug val and test	[True, False]

Table 20: Mean and standard deviation of accuracy (%) across 10 splits on node classification of heterophilic graphs. The best result is **bold**. The highlighted results indicate negative transfer for pre-training methods compared to GCN. The last column is the average rank of each method.

	Cornell	Wisconsin	Texas	Actor	Chameleon*	Squirrel*	A.R.
GCN	59.41 ± 6.03	51.68 ± 4.34	63.78 ± 4.80	30.58 ± 1.29	40.94 ± 3.91	39.11 ± 1.74	3.83
AttrMask	44.86 ± 5.43	53.73 ± 4.31	60.54 ± 5.82	25.31 ± 1.03	35.81 ± 2.88	30.63 ± 1.68	5.83
CtxtPred	40.81 ± 7.78	36.67 ± 17.23	58.92 ± 4.32	23.97 ± 2.63	24.36 ± 4.13	26.26 ± 7.50	9.50
EdgePred	42.70 ± 5.51	48.04 ± 6.63	59.37 ± 5.11	22.99 ± 6.22	21.02 ± 5.06	27.94 ± 8.41	8.83
InfoMax	39.19 ± 12.75	39.80 ± 16.38	58.87 ± 4.06	23.30 ± 4.37	22.59 ± 4.91	27.52 ± 9.09	10.17
JOAO	40.13 ± 8.60	44.70 ± 7.45	57.06 ± 3.43	24.17 ± 5.02	25.81 ± 3.79	31.72 ± 7.03	8.33
D-SLA	41.05 ± 6.88	42.13 ± 9.58	59.93 ± 4.29	23.74 ± 4.06	26.49 ± 4.27	28.50 ± 6.90	8.00
GraphMAE	47.05 ± 4.37	57.06 ± 4.59	63.70 ± 5.51	24.69 ± 0.68	37.18 ± 3.08	31.94 ± 1.65	5.00
Half-Hop	62.46 ± 7.58	76.47 ± 2.61	72.35 ± 4.27	33.95 ± 0.68	38.59 ± 2.89	37.34 ± 2.18	3.00
UniAug	68.11 ± 6.72	69.02 ± 4.96	73.51 ± 5.06	33.11 ± 1.57	**43.84 ± 3.39**	**41.90 ± 1.90**	2.00
UniAug + Half-Hop	**72.43 ± 5.81**	**79.61 ± 5.56**	**77.03 ± 4.27**	**34.97 ± 0.55**	41.94 ± 2.77	38.79 ± 2.61	1.50

*Chameleon and Squirrel are filtered to remove duplicated nodes [93].

In addition to the results shown in Table 4, we have the pre-training baselines as mentioned in Section 4.1. The results are summarized in Table 17, with similar scenarios as graph- and link-level

tasks. The existing pre-training methods provide negative transfer when pre-trained on the same data collection as *UniAug*. More explanation on removing the node features can be found in Appendix D.1.

To showcase the flexibility of *UniAug* on the downstream backbone, we pick one of the SOTA methods PolyGCL [104] for node classification on heterophilic datasets and evaluate *UniAug* on the basis of it. The results are summarized in Table 21, where we see *UniAug* produces improvements over PolyGCL in three out of four datasets.

Table 21: Accuracy of node classification with PolyGCL.

	Cornell	Wisconsin	Texas	Actor
PolyGCL	82.62 ± 3.11	85.50 ± 1.88	**88.03 ± 1.80**	41.15 ± 0.88
UniAug - PolyGCL	**84.31 ± 2.88**	**88.35 ± 2.58**	86.70 ± 2.77	**43.01 ± 1.27**

D.4 Investigation on scaling

In Section 4.2, we investigate the scaling behavior of *UniAug* regarding data scale and pre-training time. We omit some of the results for a better visualization. Here we present the numerical results in Table 22 and Table 23.

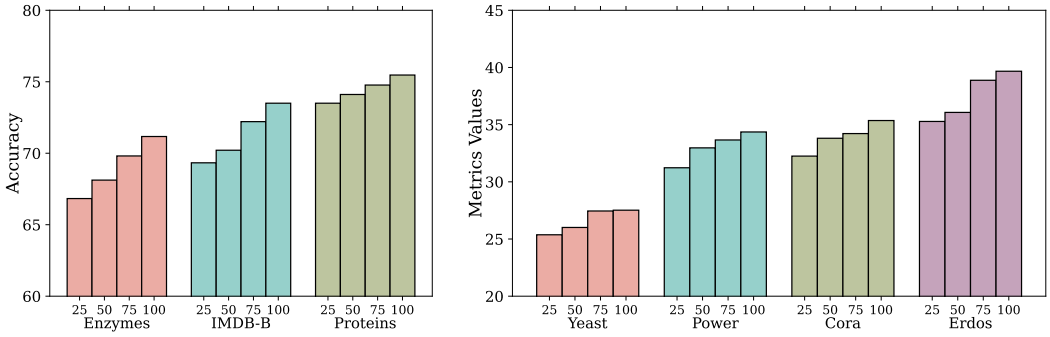

Figure 5: Effects of pre-training data scale (ratio) on graph classification (left) and link prediction (right).

We recognize that these three sets vary in both scale and diversity. To analyze the scaling effect of *UniAug* based solely on data quantity, we clustered the pre-training set into 10 clusters based on graph-level representations (Section 3.2) and performed stratified sampling within these clusters. From this, we created three subsets containing 25%, 50%, and 75% of the total graphs, and pre-trained *UniAug* on each subset. The results, summarized in Fig.5, show a clear trend of performance improvement as the size of the pre-training set increases. Combined with experiments on the SMALL, FULL, and EXTRA sets, these findings suggest that *UniAug* benefits from both increasing the scale and enhancing the diversity of the pre-training data.

Table 22: Effects of pre-training data scale on graph classification (up) and link prediction (down).

	Enzymes	Proteins	IMDB-B	IMDB-M
GIN	66.00 ± 7.52	73.32 ± 4.03	71.10 ± 2.90	49.07 ± 2.81
UniAug- SMALL	66.83 ± 7.38	73.50 ± 5.61	69.80 ± 2.70	48.93 ± 3.20
UniAug- FULL	**71.33 ± 6.51**	74.05 ± 4.82	73.11 ± 2.35	49.67 ± 2.41
UniAug- EXTRA	71.17 ± 7.10	**75.47 ± 2.50**	**73.50 ± 2.48**	**50.13 ± 2.05**

	Cora MRR	Citeseer MRR	Power Hits@10	Yeast Hits@10	Erdos Hits@10
GCN	30.26 ± 4.80	50.57 ± 7.91	30.61 ± 4.07	24.71 ± 4.92	35.71 ± 2.65
UniAug- SMALL	32.25 ± 8.71	47.91 ± 3.87	32.25 ± 3.72	25.81 ± 4.89	36.28 ± 3.56
UniAug- FULL	32.81 ± 7.44	48.32 ± 6.00	32.97 ± 3.75	26.36 ± 4.62	36.07 ± 4.20
UniAug- EXTRA	**35.36 ± 7.88**	**54.66 ± 4.55**	**34.36 ± 1.68**	**27.52 ± 4.80**	**39.67 ± 4.51**

Table 23: Effects of pre-training amount of compute on graph classification (up) and link prediction (down).

10^{-3} PF-days	Enzymes	Proteins	IMDB-B	IMDB-M
5	68.18 ± 6.21	73.32 ± 3.63	71.20 ± 2.90	48.28 ± 2.75
10	69.00 ± 5.10	74.30 ± 5.33	72.80 ± 3.85	48.60 ± 2.23
15	68.83 ± 5.88	75.11 ± 3.18	71.77 ± 2.38	48.60 ± 2.48
20	70.79 ± 5.73	74.87 ± 5.30	73.04 ± 2.82	49.47 ± 2.20
25	**71.50 ± 5.85**	**75.47 ± 2.50**	**73.50 ± 2.48**	**50.13 ± 2.05**

10^{-3} PF-days	Cora MRR	Citeseer MRR	Power Hits@10	Yeast Hits@10	Erdos Hits@10
5	27.56 ± 4.36	49.45 ± 9.20	22.81 ± 9.47	23.62 ± 9.77	35.33 ± 3.16
10	31.02 ± 6.53	50.72 ± 6.22	32.49 ± 2.52	26.70 ± 4.85	36.10 ± 4.66
15	33.24 ± 7.97	49.02 ± 5.92	32.88 ± 3.31	**27.80 ± 4.55**	**39.70 ± 3.67**
20	34.71 ± 9.08	52.90 ± 3.84	33.69 ± 3.23	26.90 ± 3.93	39.33 ± 3.16
25	**35.36 ± 7.88**	**54.66 ± 4.55**	**34.36 ± 1.68**	27.52 ± 4.80	39.67 ± 4.51

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: It's in the Conclusion and Discussion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.

- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experimental details can be found in the main text and the appendix. We also provide the code in the supplemental material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The data are publicly available and the code is in the supplemental material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please see the experiment section and the corresponding appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report the standard deviation across runs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See the impact statement.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See appendix.

    Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: See method and experiment sections.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.