Laplacian-Guided Denoising Graph Diffusion for Graph Learning with an Adaptive Prior

Seoyoon Kim
LG AI Research
seoyoon.kim@lgresearch.ai

Hyemin Jung
LG AI Research
hm.jung@lgresearch.ai

Woohyung Lim
LG AI Research
w.lim@lgresearch.ai

Abstract

Graph representation learning often relies on manually engineered, task-specific inductive biases, which limit model flexibility and generalization across diverse tasks. While diffusion models have shown promising ability in capturing arbitrary distributions, they frequently lack a deep integration of graph structure. To address this, we propose the LapDiff, a novel diffusion-based framework that learns adaptive priors to dynamically align its inductive bias with the intrinsic characteristics of graph-structured data and their tasks. The novelty of LapDiff is its use of Laplacian smoothing as a structure-aware noise mechanism in the forward process, complemented by topological perturbations. This design enables the denoising network to effectively capture the underlying data-generating factors tied to a graph's unique structure and features. Specifically, the reverse denoising process implicitly learns which structural patterns are informative for a given graph, allowing the model to adapt its representations without fixed architectural assumptions. By capturing priors from a task and data, LapDiff mitigates the limitations of static biases and enhances task-agnostic generalization. Extensive experiments on large-scale OGB benchmarks demonstrate that LapDiff is universally effective for both link prediction and node classification, achieving state-of-the-art performance and offering a new perspective into graph representation learning.

1 Introduction

Large-scale network graphs are now foundational data structures across science and industry, from biological systems [1] and drug discovery [2] to recommender systems [3] and fraud detection [4]. The central challenge in this field is learning effective representations that capture the complex, non-Euclidean relationships inherent in such data. Graph Neural Networks (GNNs) have emerged as the dominant paradigm, achieving remarkable success by leveraging strong inductive biases.

However, this reliance on inductive biases is a double-edged sword. The vast majority of GNNs are built on a fixed assumption, such as homophily, the tendency of connected nodes to be similar. While effective in some scenarios, this rigid prior leads to significant performance degradation on graphs with more complex topologies or across different downstream tasks. To compensate, a fragmented landscape of specialized models has emerged, each with manually engineered biases tailored to specific tasks, such as subgraph-based features for link prediction (*e.g.,* SEAL [5]) or generalized graph heuristics (*e.g.,* Neo-GNNs [6]).

This approach merely trades one fixed assumption for another, inherently limiting generalization and requiring separate models for different tasks, which is impractical for real-world deployment [7]. The lack of a single, dominant architecture on comprehensive benchmarks like the Open Graph Benchmark (OGB) [8] highlights this fundamental limitation. For instance, while GCN [9] achieves high accuracy on the OGB-PPA node classification task, its performance plummets in OGB-Collab

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: New Perspectives in Advancing Graph Machine Learning.

link prediction, where heuristic-based methods like SEAL [5] excel. This disparity underscores that fixed, manually-specified priors are a bottleneck for creating truly universal graph representation learning methods.

This predicament raises a fundamental question: *Can we learn powerful graph representations without manually-designed fixed inductive bias?* We argue for a paradigm shift from static, hand-crafted biases to *adaptive* ones that emerge as data-driven priors, conditioned directly on the given task and intrinsic properties of the given graph.

To motivate our approach, we first draw intuition from the No-Free-Lunch (NFL) theorem [10]. The NFL theorem suggests that no single learner is universally optimal across all tasks and data distributions. The practical limitations of fixed-bias GNNs are consistent with this observation and often arise from a misalignment between manually-specified biases and a given objective or dataset. We thus use NFL theorem as intuition to align inductive bias with the problem family. Based on PAC-Bayes [11, 12], we formalize graph adaptive prior. PAC-Bayes theory provides a formal basis for using adaptive priors by directly connecting them to generalization of a model. A standard PAC-Bayes generalization bounds for a posterior over a hypothesis class and a prior. Let π_ϕ be a *graph prior* that depends only on unlabeled information of an input graph $\phi(G)$, and let ρ be an arbitrary posterior. With probability at least $1-\delta$ over a sample of size m, the inequality holds for priors Q as:

$$\mathbb{E}_{\theta\sim\rho}[R] \;\le\; \mathbb{E}_{\theta\sim\rho}[\hat{R}] + \sqrt{\frac{\mathrm{KL}(\rho\|\pi_\phi) + \ln(2\sqrt{m}/\delta)}{2(m-1)}}. \tag{1}$$

We instantiate π_ϕ as a prior with precision $\tau(L(G)+\epsilon I)$, which encodes *Laplacian smoothing*. Then, $\mathrm{KL}(\rho\|\pi_\phi)$ turns into a Dirichlet-energy penalty. This aligns with our Laplacian-MSE surrogate (Sec. 3.2), a second-order upper bound of the edge-wise Bernoulli KL shown in Appendix A, hence lowering both empirical error and the complexity term.

This bound gives the crucial insight that generalization error depends on an empirical error and on the Kullback-Leibler divergence $\mathrm{KL}(\rho||\pi_\phi)$. To minimize $\mathrm{KL}(\rho||\pi_\phi)$, a prior π_ϕ would be constructed from an input graph without labels and be depend on a given task so that a prior π_ϕ reflects the structure of a graph and the objective of interest. By doing so, a prior π_ϕ is better aligned with the posterior induced by a dataset.

This theoretical principle provides a foundation for our goal: to learn graph representation with an adaptive prior. We propose **LapDiff**, a novel diffusion-based framework that realizes parameterizing an adaptive prior $\pi_\phi(z|\mathcal{G},\mathcal{T})$ not as a static assumption, but as the learned outcome through diffusion models with a parameter prior, *i.e.,* $\pi_\phi(\theta|\mathcal{G},\mathcal{T})$. Its core innovation is the use of Laplacian smoothing as a novel structure-aware noise mechanism, a principled way to align this prior with the intrinsic geometry of graph data. Our contributions are summarized as follows:

- We propose a generative framework **LapDiff** that learns an adaptive, graph-conditioned prior, $\pi_\phi(z|G)$, to capture implicit structural and feature-level patterns without handcrafted biases.
- We introduce the novel use of **Laplacian smoothing** as a structure-aware noise mechanism within a diffusion model, providing a new perspective on encoding graph geometry into generative processes.
- We empirically demonstrate that LapDiff achieves state-of-the-art performance on diverse benchmarks for link prediction and node classification, validating the effectiveness of our adaptive, structure-aware learning approach.

2 Related work

Graph Representation Learning. GNNs like GCN [13] and GAT [14] have achieved strong performance in node classification by exploiting local aggregation schemes. However, these often encode fixed assumptions (e.g., homophily), limiting generalization to diverse graph types. While GRAND [15] models feature diffusion as a continuous process, structural signals remain underutilized in most node-centric models. In contrast, SEAL [5], Neo-GNNs [6], and NBFNet [16] introduce inductive biases tailored for link prediction via subgraph extraction or path-based heuristics. However, these methods are narrowly focused on specific tasks.

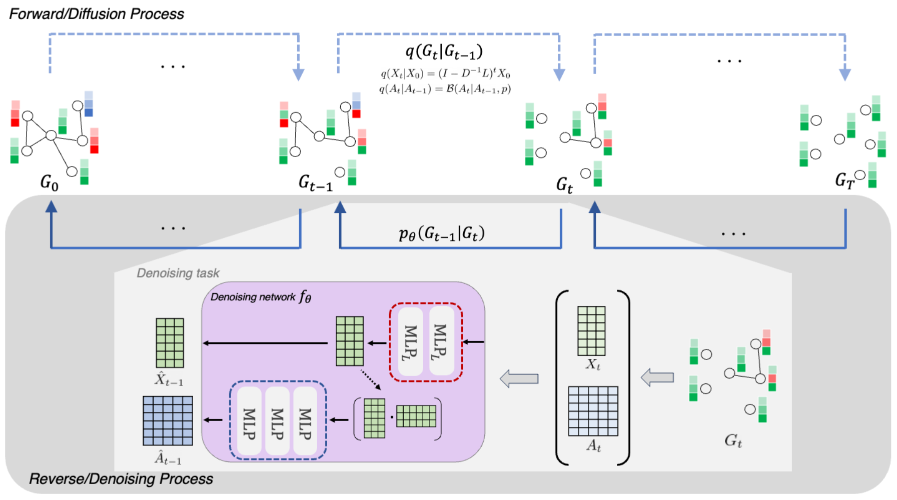

Figure 1: Graphical Model of LapDiff. LapDiff leverages Laplacian smoothing as the noise source of the forward process, inducing signal diffusion that reflects important aspects of graph structure.

DDPMs on Graph domain. DDPMs have recently been extended to graphs. Prior generative approaches [17–19] modeled node-edge distributions or joint graph likelihoods. Molecular graph generation works (e.g., [20, 21]) employed categorical and discrete formulations, while [22] proposed scalable structure perturbations. DDM citeyang2023directional introduced diffusion models for graph representation learning with anisotropic node feature noise, yet ignored graph structures. Recently, SGDiff [23] proposed a subgraph-based diffusion model tailored for link prediction, incorporating structural context into diffusion.

Graph Inverse Problems. Furthermore, our concept of an *adaptive prior* learned via denoising shares motivation with methods that learn explicit regularization. While these approaches typically learn a task-specific regularization term (e.g., a graph filter), LapDiff learns an entire generative process that models the data distribution conditioned on a graph. This allows the prior to be implicitly learned and adapted through the denoising objective itself, rather than being defined by a fixed parametric form. [24] proposed learning regularization for graph inverse problems, where a trainable graph regularizer is optimized jointly with a data-fidelity term. While both approaches involve graph structure and optimization, our work differs fundamentally: (1) We focus on representation learning rather than inverse problems (denoising, inpainting); (2) Our Laplacian smoothing acts as a fixed structural prior in the forward process, whereas [24] learns the regularizer itself; (3) We integrate topological diffusion (edge removal) alongside feature diffusion. Nevertheless, both works share the insight that structure-aware priors can improve over hand-designed alternatives, and future work could explore combining learnable Laplacian operators with our diffusion framework.

3 Method

Our model, **LapDiff**, implicitly learns the adaptive graph prior $p_\theta(z \mid G)$ posited in our theoretical motivation via a denoising diffusion process. LapDiff consists of two complementary processes: a **forward process** that incrementally injects structure-aware noise into the input graph $G_0 = (A_0, X_0)$, and a **reverse process** that learns to denoise the corrupted graph to reconstruct an informative latent representation. This section details both processes and the resulting objective function.

3.1 LapDiff Generative Process

Our generative process consists of two complementary forward processes applied to the input graph $G_0 = (A_0, X_0)$: first, a deterministic feature diffusion based on Laplacian smoothing, and secondly, a stochastic structural diffusion via edge perturbation. This corrupts feature information and topological information differently. A unified denoising network f_θ (detailed in Sec 3.2) is then trained to denoise both processes simultaneously, forcing it to learn the complex intervene between graph structure and node features. We define two Markov chains for feature and structural diffusion, $q(X_{1:T}|X_0)$ and $q(A_{1:T}|A_0)$, respectively.

3.1.1 Feature Diffusion via Laplacian Smoothing

To corrupt node features while retaining structural information, we define the forward process noise source using the graph Laplacian. Specifically, we use iterative Laplacian smoothing with the *symmetric* normalized Laplacian $\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, which acts as a low-pass filter on node features, causing them to dissipate and converge towards an over-smoothed state. The feature matrix X_t at timestep t is obtained as:

$$X_t = (\mathbf{I} - \alpha \mathbf{L}_{\text{sym}})X_{t-1} = (\mathbf{I} - \alpha \mathbf{L}_{\text{sym}})^t X_0, \tag{2}$$

where $\mathbf{L}$ is the graph Laplacian of the original graph G_0, $\mathbf{D}$ is its degree matrix, and α is a scaling factor. For simplicity, we set $\alpha = 1$. This process defines a deterministic Markov chain $q(X_{1:T}|X_0) = \prod_{t=1}^{T} q(X_t|X_{t-1})$ where the node feature information is progressively smoothed, also known as low-pass filtered, according to the original graph structure.

Theoretical Grounding of Laplacian Smoothing. Our choice of Laplacian smoothing as a noise source is theoretically grounded in spectral graph theory. Given the eigendecomposition of the *symmetric* Laplacian $\mathbf{L}_{\text{sym}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, the feature diffusion in Eq. (2) can be expressed in the spectral domain as:

$$X_t = \mathbf{U}(\mathbf{I} - \alpha\mathbf{\Lambda})^t\mathbf{U}^\top X_0. \tag{3}$$

The term $(\mathbf{I} - \alpha\mathbf{\Lambda})^t$ acts as a low-pass filter, attenuating high-frequency components of the feature signal (associated with large eigenvalues λ_i) more rapidly than low-frequency components (small λ_i). This selective attenuation preserves global structural patterns for longer, forcing the reverse process to learn representations that are tied to the graph's macro-structure, unlike isotropic Gaussian noise which degrades all signal components uniformly. In the spectral domain, the smoothed coefficients become

$$\hat{x}' = [(1 - \alpha\lambda_1)\hat{x}_1,\ (1 - \alpha\lambda_2)\hat{x}_2,\ \ldots,\ (1 - \alpha\lambda_N)\hat{x}_N]\,, \tag{4}$$

which indicates that components associated with larger λ_i are reduced more than those with smaller λ_i.

3.1.2 Structural Diffusion via Edge Perturbation

Concurrently of the feature diffusion process, we diffuse the graph topology using stochastic edge removal to ensure stochasticity in diffusion-based models. This process follows a stochastic Markov chain $q(A_{1:T}|A_0)$. At each step t, we sample a subgraph structure by randomly dropping edges from the previous state A_{t-1} with a removal probability $1 - p$. This process gradually sparsifies the graph, destroying its topological information. The per-edge transition is defined as:

$$A_t[i, j] \sim \text{Bernoulli}\big(p\, A_{t-1}[i, j]\big), \tag{5}$$

i.e., an existing edge is retained with probability p (non-edges remain absent). This defines the structural diffusion process $q(A_{1:T}|A_0) = \prod_{t=1}^{T} q(A_t|A_{t-1})$.

3.2 Reverse Process and Objective Function

The reverse process is parameterized by a denoising network $f_\theta(X_t, A_t, t)$, which predicts the state at $t - 1$. This network is implemented as a K-layer Graph Neural Network (GNN) encoder followed by two separate 3-layer MLP decoders. The GNN encoder takes both the smoothed features X_t and the perturbed adjacency matrix A_t as input, generating node representations $H_t = \text{GNN}(X_t, A_t)$. These representations capture the fused information from the corrupted structure and features. H_t is then fed into two decoders: A feature decoder $f_\theta^{(X)}(H_t, t)$ to predict X_{t-1} and a structural decoder $f_\theta^{(L)}(H_t, t)$ to predict the removed Laplacian components $(L_0 - L_{t-1})$.

Feature Denoising Objective. The feature denoising term aims to predict the features at the previous step, X_{t-1}, which can be simplified to a mean squared error loss:

$$\mathcal{L}_{\text{feat}}(t) = \left\| f_\theta^{(X)}(X_t, A_t, t) - X_{t-1} \right\|_F^2. \tag{6}$$

Structural Denoising Objective. Directly optimizing the KL divergence for discrete edges (i.e., $\mathcal{L}_{struct}$ in Eq. 294) is unstable. Instead, we optimize a stable, continuous surrogate objective.As theoretically derived in Appendix A, the per-edge Bernoulli KL divergence is upper-bounded by the squared error of the corresponding expected Laplacian matrices. This provides a well-motivated and stable optimization target. Therefore, we define the structural denoising loss $\mathcal{L}_{struct}$ as the MSE in the Laplacian space. Moreover, the Laplacian captures degree-coupled structural information more smoothly than binary adjacency matrices, leading to more stable gradients during training.

$$\mathcal{L}_{\text{struct}}(t) \approx \left\| f_\theta^{(L)}(X_t, A_t, t) - (\mathbf{L}_0 - \mathbf{L}_{t-1}) \right\|_F^2. \tag{7}$$

Here, the network predicts the change in the Laplacian, which corresponds to the structure that was removed between step 0 and $t-1$.

Final Loss. Combining the denoising terms for $t \in [2, T]$ and the explicit reconstruction losses for $t = 1$ (using MSE for features and binary cross-entropy for the adjacency matrix), the final training loss for LapDiff is:

$$\begin{aligned}\mathcal{L}_{\text{LapDiff}} := \sum_{t=2}^{T} \beta_t \left\| f_\theta^{(X)}(X_t, A_t, t) - X_{t-1} \right\|_F^2 \;+\; \gamma \sum_{t=2}^{T} \left\| f_\theta^{(L)}(X_t, A_t, t) - (\mathbf{L}_0 - \mathbf{L}_{t-1}) \right\|_F^2 \\ + \beta_0 \left\| f_\theta^{(X)}(X_1, A_1, 1) - X_0 \right\|_F^2 \;+\; \lambda\, \mathcal{L}_{\text{BCE}}\big(f_\theta^{(A)}(X_1, A_1, 1),\, A_0\big)\end{aligned} \tag{8}$$

where $\beta_0, \gamma, \beta_t, \lambda$ are weighting hyperparameters. The total loss is $\mathcal{L} = \mathcal{L}_{\text{LapDiff}} + \mathcal{L}_{\text{task}}$. The learned representations z from the denoising process capture the adaptive prior that is informative to solve a given downstream task. For link prediction, we apply MLPs to pairs of node representations, *i.e.,* $\hat{y}_{ij} = \text{MLP}(z_i^\top z_j)$. By learning to denoise graph-structured data through the diffusion-based framework, our LapDiff implicitly learns which structural patterns are highly informative to solve a task, making these representations effective across diverse tasks without task-specific handcrafted biases or architectures.

4 Experiments

Our goal is to demonstrate that LapDiff's adaptive prior mechanism leads to universally strong performance across diverse graph learning tasks. We then analyze the contribution of its core components. We evaluate LapDiff on seven benchmark datasets including Open Graph Benchmark (OGB), covering two diverse network graph learning tasks: link prediction and node classification. For evaluation, we follow the standard OGB protocols, using Hits@K and MRR for link prediction, and accuracy for node classification. For node classification, we adopt a challenging few-shot setting with a fixed number of labeled nodes $k \in \{1, 5, 10\}$ per class to test representation power under extreme label scarcity. We compare LapDiff against a comprehensive set of baselines, including heuristic methods, classic GNNs (GCN, GAT, GraphSAGE), and recent specialized architectures (SEAL, Neo-GNNs, C&S, GraphMAE2). Experimental details are provided in the Appendix. An anonymized implementation is available at `https://anonymous.4open.science/r/NPGMLworkshop2025CDF2`

4.1 Performance on Downstream tasks

Across all link prediction and node classification benchmarks, LapDiff consistently outperforms task-specific baselines, indicating it captures both structural and feature-level priors without hand-crafted biases. This robustness across diverse tasks and datasets confirms its ability to learn broadly generalizable graph representations, emphasizing LapDiff's utility for diverse graph-learning applications.

Link prediction. Table 1 reports the results of OGB link prediction benchmarks. Our LapDiff generally shows significantly improved performance than other baselines on OGB-Collab and OGB-DDI datasets and second-best performance on OGB-PPA and OGB-CItation2 with low discrepancy to the best models. This indicates our LapDiff is capable of capturing latent graph priors that are crucial in the context of link prediction tasks and data. Additionally, LapDiff demonstrates robust performance across various datasets. LapDiff has the ability to handle both underlying prior distribution and internal biases in graph structures and node features. For instance, LapDiff achieves the best performance on OGB-DDI, whereas SEAL, which is designed to generalize higher-order

Table 1: Link prediction performances on Open Graph Benchmark (OGB) datasets. OOM denotes 'out of memory'. <u>**Bold underline**</u> indicates the best performance and **bold** indicates the second best performance.

	Model	OGB-PPA	OGB-Collab	OGB-DDI	OGB-Citation2
Neighborhoood Heuristics	Common Neighbors	27.65 ± 0.00	50.06 ± 0.00	17.73 ± 0.00	76.20 ± 0.0
	Adamic Adar	32.45 ± 0.00	53.00 ± 0.00	18.61 ± 0.00	76.12 ± 0.0
	Resource Allocation	<u>**49.33**</u> ± 0.00	52.89 ± 0.00	6.23 ± 0.00	76.20 ± 0.0
Shallow Methods	Matrix Factorization	23.78 ± 1.82	34.87 ± 0.23	13.29 ± 2.32	50.48 ± 3.09
	MLP	0.99 ± 0.15	16.05 ± 0.48	N/A	25.13 ± 0.28
GRL Methods	GCN	15.37 ± 1.25	44.57 ± 0.64	40.87 ± 6.08	82.57 ± 0.26
	GAT	OOM	41.73 ± 1.01	32.57 ± 3.48	OOM
	SAGE	12.51 ± 2.02	47.86 ± 0.64	47.06 ± 5.21	80.18 ± 0.15
	JKNet	11.73 ± 1.98	47.52 ± 0.73	57.95 ± 7.69	OOM
	SEAL	47.18 ± 3.60	**54.27** ± 0.46	29.86 ± 4.37	<u>**86.72**</u> ± 0.31
	Neo-GNN	47.53 ± 0.63	53.95 ± 0.52	**60.02** ± 3.86	85.96 ± 0.94
	DDM	17.93 ± 1.91	49.56 ± 1.79	47.73 ± 3.10	83.51 ± 0.20
	LapDiff(ours)	**48.32** ± 0.68	<u>**54.33**</u> ± 0.35	<u>**60.56**</u> ± 2.32	**86.70** ± 0.27

Table 2: Node classification performance on OGB-Arxiv, OGB-Products, and PubMed dataset. OOM denotes 'out of memory'. **Bold** indicates the best performance.

Model	OGB-Arxiv			OGB-Products			PubMed		
Fixed k nodes	$k=1$	$k=5$	$k=10$	$k=1$	$k=5$	$k=10$	$k=1$	$k=5$	$k=10$
GCN	31.69 ± 2.74	52.97 ± 0.94	58.39 ± 0.50	38.93 ± 2.09	62.69 ± 1.27	66.23 ± 0.91	45.87 ± 2.44	60.56 ± 1.44	69.50 ± 0.68
GAT	25.60 ± 2.95	50.87 ± 1.78	57.23 ± 0.75	35.81 ± 2.42	60.72 ± 1.93	64.80 ± 1.21	43.57 ± 2.71	58.38 ± 2.06	68.40 ± 1.49
APPNP	29.36 ± 2.19	52.47 ± 1.26	56.42 ± 0.83	36.35 ± 2.20	63.01 ± 2.10	66.85 ± 0.84	43.04 ± 1.72	56.94 ± 1.90	69.99 ± 0.73
GCNII	30.94 ± 2.30	51.94 ± 1.18	57.65 ± 0.94	33.64 ± 2.32	61.43 ± 2.36	64.90 ± 1.39	43.29 ± 2.53	56.18 ± 1.84	70.60 ± 0.93
C&S	30.63 ± 1.88	51.73 ± 1.30	56.57 ± 1.43	40.47 ± 1.97	62.18 ± 1.57	67.53 ± 1.40	44.91 ± 1.24	57.44 ± 1.36	68.78 ± 1.07
CCA-SSG	29.21 ± 3.01	51.67 ± 2.31	57.40 ± 0.89	39.95 ± 2.67	63.10 ± 2.13	67.62 ± 1.56	47.56 ± 3.15	60.44 ± 2.60	69.24 ± 0.68
GraphMAE2	33.83 ± 3.23	54.67 ± 2.32	60.16 ± 0.75	41.65 ± 2.01	63.69 ± 2.30	68.08 ± 1.62	50.76 ± 5.10	64.29 ± 0.11	70.51 ± 0.11
DDM	36.08 ± 0.93	55.69 ± 1.41	60.71 ± 0.31	45.57 ± 1.28	64.90 ± 1.03	69.62 ± 0.15	50.95 ± 2.42	65.13 ± 0.10	70.12 ± 0.56
LapDiff(ours)	**39.04** ± 1.52	**57.83** ± 0.83	**61.23** ± 0.37	**49.10** ± 1.75	**67.47** ± 1.03	**70.69** ± 0.61	**53.82** ± 1.46	**66.93** ± 0.88	**72.77** ± 0.62

structural heuristics shows relatively poor performance. This result illustrates that existing GNNs often show degradation under conditions that are distinct from their intrinsic objectives. Thus, it highlights that our model effectively learn latent representations with hidden, complex aspects that are captured by latent priors in a graph naturally regarding certain objectives.

Node classification. We conduct experiments on semi-supervised node classification benchmark datasets to validate the effectiveness of LapDiff on learning node embeddings. We constrained the training index by the fixed k nodes per label. The number k is set to $1, 5,$ and 10. Table 2 shows the performance of a semi-supervised node classification task that is extremely limited to label scarcity. LapDiff outperforms other baselines on all datasets and settings. According to the results, LapDiff effectively captures the latent distribution of nodes, even under very constrained conditions.

4.2 Analysis on Components

We empirically validate the efficacy of each component in LapDiff through ablation experiments in Table 3. First, we demonstrate the capability of LapDiff to capture universal and comprehensive representation by training without downstream task loss. It is remarkable that it still shows a relatively high performance than LapDiff without feature process or structure process. Also, it is interesting that it still outperforms conventional GNNs including GCN, GAT, SAGE, and JKNet. This result implies that LapDiff is capable of learning critical latent representations within a graph that are significant by aligning hidden graph priors.

Then, we evaluate LapDiff without the feature diffusion process and structural diffusion process based on the average performance on link prediction datasets. LapDiff without feature diffusion process and LapDiff without structural diffusion process both show degraded performance on OGB-Collab and OGB-PPA.

4.3 Analysis on Noise Source

Table 4 presents an analysis of the efficacy of different noise sources to capture underlying latent priors dynamically aligning to the internal structures of task and dataset. The noise sources compared to Laplacian smoothing are random Gaussian noise from DDPM [25] and Anisotropic Gaussian

Table 3: Ablation study analyzing the efficacy of each component of LapDiff.

Dataset	OGB-Collab	OGB-PPA
LapDiff	54.33 ± 0.37	48.87 ± 0.64
LapDiff (w/o downstream loss)	51.78 ± 0.40	44.92 ± 1.52
LapDiff (w/o feature process)	45.13 ± 2.33	39.77 ± 1.13
LapDiff (w/o structure process)	45.47 ± 3.02	26.42 ± 2.30

Table 4: Analysis on the efficacy of noise source. As other baselines only consider node features, the structure process in LapDiff is excluded from evaluation.

Dataset Metric	OGB-Collab Hits@50	OGB-Arxiv ($k = 5$) Accuracy
LapDiff (Laplacian smoothing)	**52.30** $\pm$ 0.97	**57.02** $\pm$ 0.79
DDPM (Random noise)	47.43 ± 1.58	53.38 ± 1.02
DDM (Anisotropic noise)	49.56 ± 1.79	55.69 ± 1.41

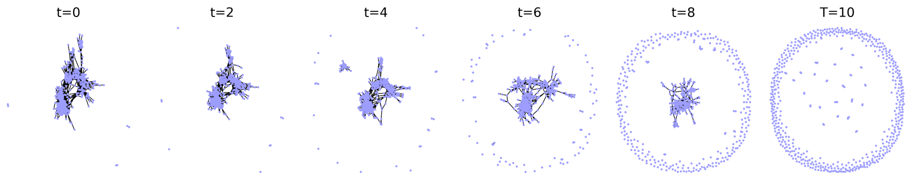

Figure 2: Visualization of graph reconstruction on PubMed at every even number time step. To improve the visibility of the figure, we sampled the subgraph of PubMed with large connectivity.

noise in DDM [26]. The effectiveness of each noise source is evaluated for link prediction and semi-supervised node classification tasks on OGB-Collab and OGB-Arxiv datasets, respectively. On both datasets, Laplacian smoothing outperforms the other two noise sources. It achieves a score of 52.30 on the OGB-Collab dataset and 57.02 on the OGB-Arxiv dataset. Compared to random Gaussian noise and anisotropic Gaussian noise, Laplacian smoothing shows a clear advantage. For instance, it exceeds random Gaussian Noise by nearly 5 percentage points and exceeds anisotropic Gaussian noise by over 3 percentage points on OGB-Collab dataset. These results strongly support our central hypothesis: the Laplacian smoothing, by selectively preserving low-frequency structural signals (as analyzed in Sec 3.1.1), provides a far more informative gradient for the denoising network than isotropic noise, which destroys all signal components uniformly. These results indicate that utilizing Laplacian smoothing as a noise source is not only effective but also consistent, offering a significant improvement over other random noise. It provides strong empirical evidence of the efficacy of Laplacian smoothing as a noise source in a diffusion model for graph representation learning tasks, specifically for large graph data.

4.4 Underlying Distribution of Graph

The goal of LapDiff is to learn universal and generalizable representations that are aligned with the underlying distribution of large graph data. To validate the capability to learn priors of network graphs and implicitly aligned with inherent priors, Fig 2 provides the visualization of the reconstruction of a graph on PubMed dataset at each even number of states. As shown in Fig 2, the trajectory of reconstruction is directed toward the input state, eventually, showing a similar structure at $t = 2$ compared to the input graph $t = 0$.

5 Conclusion

We introduced LapDiff, a framework that instantiates an adaptive inductive bias. We operationalize this by training a denoising network to invert two complementary processes: a deterministic feature smoothing via the graph Laplacian and a stochastic structural perturbation. LapDiff learns a graph-specific prior instead of relying on hand-crafted assumptions. Across diverse benchmarks, it delivers strong performance and provides evidence toward more universal graph learners. LapDiff shows the largest improvements on datasets where fixed biases fail. On simpler tasks like small citation networks, the performance gain is smaller because even fixed-bias GNNs work well. LapDiff requires more computation than standard GNNs due to the iterative denoising process (see Appendix D.1 for complexity analysis). For extremely large graphs, the computation may become a bottleneck, thus, future work could explore mini-batch Laplacian approximations or localized smoothing operators. Also, future work includes directed Laplacians, schedule learning, standardized diagnostics, exploring richer structure-aware noise processes and theoretically analyzing the properties of the learned graph priors. Our work encourages further research on adaptive inductive bias universal representation in graph machine learning.

References

[1] Vladimir Gligorijevic and et al. Structure-based protein function prediction using graph convolutional networks. *Nature Communications*, 12(1):3168, 2021.

[2] Javier Jiménez-Luna, Francesca Grisoni, and Gisbert Schneider. Drug discovery with graph neural networks: A survey. *Molecular Informatics*, 2021.

[3] Xiangnan He and et al. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference*, 2020.

[4] Yingtong Dou and et al. Enhancing graph neural network-based fraud detection with sequential and time information. In *Proceedings of the 26th ACM SIGKDD Conference (KDD)*, 2020.

[5] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pages 5165–5175, 2018.

[6] Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J. Kim. Neo-GNNs: Neighborhood overlap-aware graph neural networks for link prediction. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=Ic9vRN3VpZ`.

[7] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[8] 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. *arXiv preprint arXiv:2005.00687*, 2020.

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

[10] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[11] Amiran Ambroladze, Emilio Parrado-Hernández, and John Shawe-Taylor. Tighter pac-bayes bounds. *Advances in neural information processing systems*, 19, 2006.

[12] Omar Rivasplata, Ilja Kuzborskij, Csaba Szepesvári, and John Shawe-Taylor. Pac-bayes analysis beyond the usual bounds. *Advances in Neural Information Processing Systems*, 33:16833–16845, 2020.

[13] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations*, 2018.

[15] Benjamin Paul Chamberlain, James Rowbottom, Maria Goronova, Stefan Webb, Emanuele Rossi, and Michael M Bronstein. Grand: Graph neural diffusion. *Proceedings of the 38th International Conference on Machine Learning, (ICML) 2021, 18-24 July 2021, Virtual Event*, 2021.

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

[17] Jiaqi Ma, Weijing Tang, Ji Zhu, and Qiaozhu Mei. A flexible generative framework for graph-based semi-supervised learning. *Advances in neural information processing systems*, 32, 2019.

[18] Pantelis Elinas, Edwin V Bonilla, and Louis Tiao. Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. *Advances in Neural Information Processing Systems*, 33:18648–18660, 2020.

[19] 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.

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

[21] Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *arXiv preprint arXiv:2210.01549*, 2022.

[22] Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. *arXiv preprint arXiv:2305.04111*, 2023.

[23] Hang Li, Wei Jin, Geri Skenderi, Harry Shomer, Wenzhuo Tang, Wenqi Fan, and Jiliang Tang. Sub-graph based diffusion model for link prediction, 2024. URL `https://arxiv.org/abs/2409.08487`.

[24] Moshe Eliasof, Md Shahriar Rahim Siddiqui, Carola-Bibiane Schönlieb, and Eldad Haber. Learning regularization for graph inverse problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 16471–16479, 2025.

[25] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[26] Run Yang, Yuling Yang, Fan Zhou, and Qiang Sun. Directional diffusion models for graph representation learning, 2023.

[27] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 2017.

[28] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 5453–5462. PMLR, 10–15 Jul 2018.

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

[30] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, page 556–559. Association for Computing Machinery, 2003. doi: 10.1145/956863.956972.

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

[32] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71:623–630, 2009.

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

[34] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[35] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2019.

[36] Zhewei Wei Ming Chen, Bolin Ding Zengfeng Huang, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[37] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993*, 2020.

[38] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and S Yu Philip. From canonical correlation analysis to self-supervised graph neural networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[39] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, and Jie Tang Evgeny Kharlamov. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM Web Conference 2023 (WWW'23)*, 2023.

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

A Derivation of Loss function of LapDiff

This section provides a derivation of the variational lower bound (ELBO) and the loss function of our proposed model, LapDiff.

A.1 Derivation of Evidence Lower Bound (ELBO)

Let G_0 represent a given observed graph data consisting of X and A, denoting node feature matrix and adjacency matrix, respectively. Taking the log-likelihood $\log p(G_0) = \log p(X_0, A_0)$, we obtain

$$\begin{aligned}\log p_\theta(X_0, A_0) &= \log\int p_\theta(X_{0:T}, A_{0:T})\, dX_{1:T} dA_{1:T} \\ &\geq \int q(X_{1:T}, A_{1:T}|X_0, A_0) \log\frac{p_\theta(X_{0:T}, A_{0:T})}{q(X_{1:T}, A_{1:T}|X_0, A_0)}\, dG_{1:T} \\ &\qquad \text{(by variational posterior and Jensen's inequality)} \\ &= \mathbb{E}_{q(X_{1:T},A_{1:T}|X_0,A_0)}\left[\log\frac{p_\theta(X_{0:T}, A_{0:T})}{q(X_{1:T}, A_{1:T}|X_0, A_0)}\right]\end{aligned} \tag{9}$$

For simplifying the notation, $dG_{1:T} = dX_{1:T}dA_{1:T}$ and $\mathbb{E}_q = \mathbb{E}_{q(X_{1:T},A_{1:T}|X_0,A_0)}$. The variational lower bound (ELBO) is obtained as follows:

$$\text{ELBO:} = \mathbb{E}_q\left[\log\frac{p_\theta(X_{0:T}, A_{0:T})}{q(X_{1:T}, A_{1:T}|X_0, A_0)}\right] \tag{10}$$

$$= \mathbb{E}_q\left[\log\frac{p_\theta(X_T, A_T)\prod_{t=1}^{T} p_\theta(X_{t-1}|X_t, A_t)\cdot p_\theta(A_{t-1}|X_t, A_t)}{\prod_{t=1}^{T} q(X_t|X_{t-1})\cdot q(A_t|A_{t-1})}\right] \tag{11}$$

$$= \mathbb{E}_q\left[\log p_\theta(X_T, A_T) + \sum_{t=1}^{T}\log\frac{p_\theta(X_{t-1}|X_t, A_t)\cdot p_\theta(A_{t-1}|X_t, A_t)}{q(X_t|X_{t-1})\cdot q(A_t|A_{t-1})}\right] \tag{12}$$

Then, we could separate $t = 1$ and $t \geq 2$ because $t = 1$ indicates the reconstruction to the given data $G_0 = (X_0, A_0)$. Next, we flip latent variables in the variational posterior q. We reparametrize $q(X_t|X_{t-1}),\ q(A_t|A_{t-1})$ to $q(X_{t-1}|X_t, X_0),\ q(A_{t-1}|A_t, A_0)$, respectively. Specifically, Markov chain properties formalize $q(X_t|X_{t-1}) = q(X_{t-1}|X_t, X_0)\frac{q(X_t|X_0)}{q(X_{t-1}|X_0)}$ and this holds in A as well. The second term can be rewritten as

$$\sum_{t=2}^{T} \log \frac{p_\theta(X_{t-1}|X_t, A_t) \cdot p_\theta(A_{t-1}|X_t, A_t)}{q(X_{t-1}|X_t, X_0) \cdot q(A_{t-1}|A_t, A_0)} + \sum_{t=2}^{T} \log \frac{q(X_{t-1}|X_0) \cdot q(A_{t-1}|A_0)}{q(X_t|X_0) \cdot q(A_t|A_0)} \tag{13}$$

$$= \sum_{t=2}^{T} \left[\log \frac{p_\theta(X_{t-1}|X_t, A_t) \cdot p_\theta(A_{t-1}|X_t, A_t)}{q(X_{t-1}|X_t, X_0) \cdot q(A_{t-1}|A_t, A_0)} + \log \frac{q(X_1|X_0)}{\cancel{q(X_2|X_0)}} \cdot \frac{\cancel{q(X_2|X_0)}}{\cancel{q(X_3|X_0)}} \cdots \frac{\cancel{q(X_{T-2}|X_0)}}{\cancel{q(X_{T-1}|X_0)}} \cdot \frac{\cancel{q(X_{T-1}|X_0)}}{q(X_T|X_0)} + \log \frac{q(A_1|A_0)}{\cancel{q(A_2|A_0)}} \cdot \frac{\cancel{q(A_2|A_0)}}{\cancel{q(A_3|A_0)}} \cdots \frac{\cancel{q(A_{T-2}|A_0)}}{\cancel{q(A_{T-1}|A_0)}} \cdot \frac{\cancel{q(A_{T-1}|A_0)}}{q(A_T|A_0)} \right]. \tag{14}$$

Then, the ELBO is derived as follows:

$$\text{ELBO} = \mathbb{E}_q \left[\log p_\theta(X_T, A_T) + \sum_{t=2}^{T} \log \frac{p_\theta(X_{t-1}|X_t, A_t) \cdot p_\theta(A_{t-1}|X_t, A_t)}{q(X_{t-1}|X_t, X_0) \cdot q(A_{t-1}|A_t, A_0)} + \log \frac{\cancel{q(X_1|X_0)} \cdot \cancel{q(A_1|A_0)}}{q(X_T|X_0) \cdot q(A_T|A_0)} + \log \frac{p_\theta(X_0|X_1, A_1) \cdot p_\theta(A_0|X_1, A_1)}{\cancel{q(X_1|X_0)} \cdot \cancel{q(A_1|A_0)}} \right] \tag{15}$$

$$= \mathbb{E}_q \left[\log \overset{\text{constant}}{\frac{p_\theta(X_T, A_T)}{q(X_T|X_0) \cdot q(A_T|A_0)}} + \sum_{t=2}^{T} \log \frac{p_\theta(X_{t-1}|X_t, A_t) \cdot p_\theta(A_{t-1}|X_t, A_t)}{q(X_{t-1}|X_t, X_0) \cdot q(A_{t-1}|A_t, A_0)} + \log p_\theta(X_0|X_1, A_1) \cdot p_\theta(A_0|X_1, A_1) \right] \tag{16}$$

The first term is not trainable. Since the degree of noise injection in the forward process is fixed and not optimized during training, it can be treated as a constant. The second term is equivalent to the definition of KL divergence, thus the final form of the ELBO of the model is

$$\text{ELBO} = \mathbb{E}_q \Big[\log p_\theta(X_0|X_1, A_1) + \log p_\theta(A_0|X_1, A_1) - \sum_{t \geq 2} \text{KL}\left[q(X_{t-1}|X_t, X_0) \| p_\theta(X_{t-1}|X_t, A_t) \right] - \sum_{t \geq 2} \text{KL}\left[q(A_{t-1}|A_t, A_0) \| p_\theta(A_{t-1}|X_t, A_t) \right] \Big] \tag{17}$$

A.2 Variational Posterior

In our proposed model, the feature-level and structural diffusion processes are designed to be Markov chains, meaning the noisy data at each timestep t depends only on the state at $t - 1$. In the context of the forward diffusion processes implies:

$$q(X_t|X_0, X_1, \dots, X_{t-1}, A_0) = q(X_t|X_{t-1}, \overset{\text{Fixed}}{\cancel{A_0}}) \tag{18}$$
$$q(A_t|A_0, A_1, \dots, A_{t-1}) = q(A_t|A_{t-1}), \tag{19}$$

where the noise source, Laplacian smoothing, is fixed with A_0. By the definition of Markov chains, we can remove A_0, which serves the role of a fixed noise schedule. Also, recall that we defined noise in the feature-level diffusion process as

$$q(X_t|X_{t-1}) := (I - D^{-1}L)^t X_0 = (I - D^{-1}L)X_{t-1}$$
$$q(X_{1:T}|X_0) = \prod_{t=1}^{T} q(X_t|X_0) = \prod_{t=1}^{T} q(X_t|X_{t-1}). \tag{20}$$

Start from the joint conditional distribution of all X_t and A_t, given X_0, A_0:

$$\begin{aligned} q(X_{1:T},&A_{1:T}|X_0, A_0) \\ &= q(X_1, X_2, \dots, X_T, A_1, A_2, \dots, A_T|X_0, A_0). \end{aligned} \tag{21}$$

By the chain rule, Eq (21) equals

$$\begin{aligned} q(X_1|X_0)q(A_1|A_0) \cdot q(X_2|X_0, X_1)q(A_2|A_0, A_1) \\ \dots q(X_T|X_0, \dots, X_{T-1})q(A_T|A_0, \dots, A_{T-1}). \end{aligned} \tag{22}$$

By applying the Markov chain property that each X_t, A_t only depends on X_{t-1}, A_{t-1}:

$$q(X_{1:T}, A_{1:T}|X_0, A_0) = \prod_{t=1}^{T} q(X_t|X_{t-1}) \cdot q(A_t|A_{t-1}) \tag{23}$$

A.3 Approximation to Laplacian Difference Prediction

For one undirected edges (i, j), let the one-edge KL be

$$p_{ij} = q\big(A_{t-1,ij} = 1 \mid A_t, A_0\big),\ \hat{p}_{ij} = p_\theta\big(A_{t-1,ij} = 1 \mid G_t\big). \tag{24}$$

The Bernoulli Kullback–Leibler term that appears in the ELBO can be rewritten as

$$D_{KL}\big(p_{ij} \,\|\, \hat{p}_{ij}\big) = p_{ij}\ \log\frac{p_{ij}}{\hat{p}_{ij}} + (1 - p_{ij})\log\frac{1 - p_{ij}}{1 - \hat{p}_{ij}}. \tag{25}$$

Fix the *target* probability $p_{ij} \in (0, 1)$ and leverage Taylor expansion, $\varepsilon := \hat{p}_{ij} - p_{ij}$. Because the first derivative vanishes at $\varepsilon = 0$ and the second derivative equals $\frac{1}{p_{ij}(1-p_{ij})}$,

$$D_{KL}\big(p_{ij} \,\|\, p_{ij} + \varepsilon\big) = \frac{\varepsilon^2}{2\, p_{ij}(1 - p_{ij})} + \mathcal{O}(\varepsilon^3) \tag{26}$$

whenever $|\varepsilon| < \min\{p_{ij}, 1 - p_{ij}\}$. Thus to second order the KL is proportional to the squared error $\big(p_{ij} - \hat{p}_{ij}\big)^2$.

For a random adjacency matrix A with edge–existence probabilities q the expected combinatorial Laplacian is

$$\bar{L}(q) = \mathrm{diag}\Big(\sum_k q_{1k}, \dots\Big)\ -\ q. \tag{27}$$

For off-diagonal entries $(i \neq j)$: $\bar{L}(q)_{ij} = -\, q_{ij}$. Therefore, we can obtain

$$p_{ij} - \hat{p}_{ij} = -(\,\bar{L}(p)_{ij} - \bar{L}(\hat{p})_{ij}\,). \tag{28}$$

Then all edges can be summed over:

$$\begin{aligned} \sum_{i<j} D_{KL}\big(p_{ij} \,\|\, \hat{p}_{ij}\big) \overset{(2)}{=} \frac{1}{2}\sum_{i<j} \frac{\big(\bar{L}(p)_{ij} - \bar{L}(\hat{p})_{ij}\big)^2}{p_{ij}(1 - p_{ij})} \\ + \mathcal{O}\big(\|\bar{L}(p) - \bar{L}(\hat{p})\|_F^3\big). \end{aligned} \tag{29}$$

Assume every posterior edge probability is clipped away from $0, 1$: $p_{ij} \in [\varepsilon, 1-\varepsilon]$ for some fixed $0 < \varepsilon < \frac{1}{2}$. Then

$$\begin{aligned} p_{ij}(1-p_{ij}) \;&\geq\; \varepsilon(1-\varepsilon) \\ \Longrightarrow D_{KL}(p_{ij} \,\|\, \hat{p}_{ij}) \;&\leq\; c_\varepsilon \left(\bar{L}(p)_{ij} - \bar{L}(\hat{p})_{ij}\right)^2, \end{aligned} \tag{30}$$

with $c_\varepsilon = \frac{1}{2\varepsilon(1-\varepsilon)}$. Using the Frobenius norm of a symmetric matrix and (non-overlapping) edges:

$$\mathcal{L}_{\text{edge\_KL}} := \sum_{i<j} D_{KL}(p_{ij} \,\|\, \hat{p}_{ij}) \;\leq\; c_\varepsilon \left\|\bar{L}(p) - \bar{L}(\hat{p})\right\|_F^2 \tag{31}$$

Hence, minimizing the Laplacian MSE upper–bounds the exact edge-wise KL.

At training time we see a *single* Laplacian $L_{t-1} = D(A_{t-1}) - A_{t-1}$ drawn from the posterior $q(A_{t-1} \mid A_t, A_0)$. Its expectation is $\bar{L}(p)$ and its entry-wise variance is $\mathcal{O}(\beta_t)$, the forward step size. For small β_t we can drop that noise and use

$$\bar{L}(p) \;\approx\; L_0 - L_{t-1},$$

because the forward diffusion from A_{t-1} to A_t removes each edge with probability β_t, so the **expected** difference to the clean graph is $L_0 - L_{t-1}$.

Step 6: Define the practical surrogate loss
Let $f_\theta(X_t, A_t)$ be the GNN output that tries to recover $(L_0 - L_{t-1})$. Combining (5) and (6):

$$\mathcal{L}_{\text{Lap}} := \;\lambda \left\|f_\theta(X_t, A_t) - (L_0 - L_{t-1})\right\|_2^2, \quad \lambda \geq c_\varepsilon.$$

Because it is an **upper bound** of the true KL term in the ELBO, driving $\mathcal{L}_{\text{Lap}}$ to zero also drives the Bernoulli KL towards zero. Empirically the continuous, degree-coupled signal contained in the Laplacian makes gradients less noisy and optimisation easier than edge-wise cross-entropy.

Therefore the Laplacian-MSE loss you observe to work well is mathematically a second-order surrogate—and an upper bound—for the exact ELBO term that compares adjacency posteriors. Minimising it is guaranteed to minimise, up to a bounded factor, the information-theoretic quantity we truly care about.

A.4 Training of LapDiff with denoising network

The training procedure of LapDiff is described as Algorithm 1.

Algorithm 1 Training LapDiff

```
Input: Large graph G=(X,A)
1: for $k = 1$ to $T$ do
2:     $X_{k-1} \leftarrow (I - D^{-1}L)^{k-1}X$
3:     $X_k \leftarrow (I - D^{-1}L)^{k}X$
4:     $A_k \sim \mathcal{B}(A_{k-1}; p_{k-1})$
5:     $\theta_{k-1} \leftarrow \theta_k - \eta\nabla_\theta[\mathcal{L}_{\text{feat}}(f_{\theta_k}(X_k, A_k), X_{k-1}) + \mathcal{L}_{\text{struct}}(f_{\theta_k}(X_k, A_k), A_{k-1})]$
6: end for
7: $\theta \leftarrow \eta\nabla_\theta\mathcal{L}_{\text{task}}(f_{\theta_0}(X, A), Y_{\text{task}})$
```

B Hyperparameter Sensitivity Analysis

We analyze LapDiff to demonstrate how hyperparameters affect the performance of LapDiff. We conduct experiments with 4 hyperparameters in LapDiff loss function, $\mathcal{L}_{\text{LapDiff}}$. β_t, β_1, γ and λ is weighting hyperparameters for $\mathcal{L}_{\text{feat}}$, $\mathcal{L}_{\text{feat-recon}}$, $\mathcal{L}_{\text{Lap}}$, and $\mathcal{L}_{\text{recon}}$, respectively. We measure Hits@50 by changing one hyperparameter while the rest of the hyperparameters are fixed to the best value. The result (Figure. 3) demonstrates that LapDiff is fairly robust to hyperparameters that weight the components of LapDiff loss $\mathcal{L}_{\text{LapDiff}}$. Accordingly, hyperparameters affect the performance of LapDiff slightly. Consequently, we can conclude that the performances of LapDiff are fairly consistent under various hyperparameter sets.

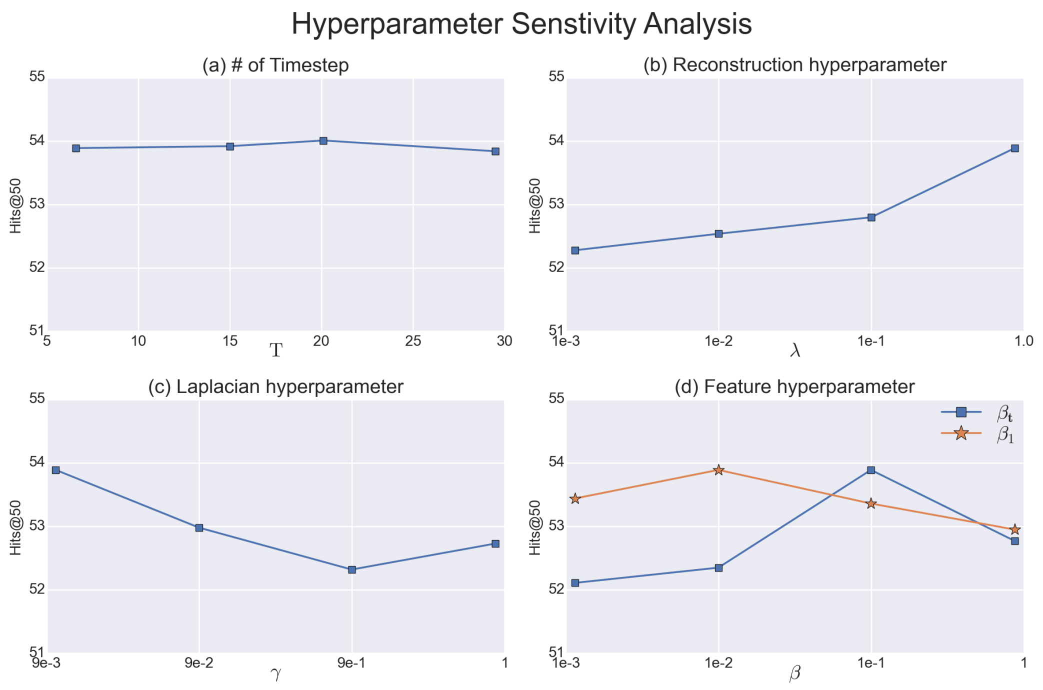

Figure 3: Visualization of hyperparameter sensitivity analysis on OGB-Collab.

C Experimental Setting Details

Datasets. To validate our models, we utilize Open Graph Benchmark (OGB) dataset for link prediction tasks and node classification tasks [8]. We use four OGB link property datasets for link prediction tasks: OGB-PPA, OGB-Collab, OGB-DDI, and OGB-Citation2. OGB-PPA is an undirected and unweighted graph representing protein association. Nodes are proteins from different specifies and edges mean biological associations. Each node feature is a one-hot vector indicating the species to which the protein belongs. OGB-Collab is an undirected graph, which represents a collaboration network where edges denote collaborations between authors. OGB-DDI is an undirected, unweighted graph that contains drug-drug interactions, with edges indicating interactions such as combined effects. Note that this dataset lacks node features. OGB-Citation2 is a citation network graph with direction. Each node in the graph corresponds to a paper, and a directed edge indicates that one paper cites another. Both OGB-Citation2 and OGB-Collab include node features obtained from embedding models. For node classification tasks, we use three benchmark datasets: OGB-Arxiv, OGB-Products, and PubMed.

Evaluation. According to the evaluation protocol of OGB, we evaluate our model with Hits@K metric and Mean reciprocal rank (MRR) in link prediction. Hits@K is based on ranking positive test edges against randomly sampled negative edges. The ranking performance is measured by the ratio of positive test edges ranked at or above the K-th position. In OGB-PPA, the K-th position is set to 100, while for OGB-Collab and OGB-DDI, it is set to 50 and 20, respectively. The evaluation metric for OGB-Citation2 is MRR. It calculates the reciprocal rank of the true edges within the pool of negative candidates for each source node and then averages these values across all source nodes. To further demonstrate the ability to learn compendious underlying structures in node classification, we constrain a fixed k-nodes setting by vastly reducing the number of nodes per label in train sets. Under this setting, accuracy measures the performance on OGB-Arxiv, OGB-Products, and PubMed.

Baselines. For baselines on link prediction, we include prevalent GNN-based models: GCN [13], GAT [14], GraphSAGE [27], JKNet [28], Variational Graph Autoencoder [29], SEAL [5], Neo-GNNs [6], and DDM. Note that SEAL extract enclosing subgraph to utilize in link prediction. Additionally, three link prediction heuristics [30–32], Matrix factorization [33], and Multi-layer perceptron [34] are included in baselines. Baseline models for semi-supervised node classification include GCN, GAT, APPNP [35], GCNII [36], and Correct&Smooth (C&S) [37]. We also compare LapDiff with self-supervised graph learning methods, CCA-SSG [38], and generative method GraphMAE2 [39].

Implementation Details. We implemented link prediction heuristics, such as Common Neighbor(CN), Adamic Adar(AA), and Resource Allocation(RA), based on the paper [30–32]. For GCN, GraphSAGE, GAT, JKNet, APPNP, GCNII, and MLP we used the implementation in PyTorch Geometric [40], and for SEAL and C&S, we used the implementation from the official repository. We trained LapDiff with a 2-layer LapDiff encoder with latent Laplacian parameters and 3-layer MLP decoder for OGB-Collab, OGB-DDI, OGB-PPA, and OGB-Citation2. For OGB-Arxiv, OGB-Products, and PubMed, we used 3-layer LapDiff encoder and 3-layer MLP decoder. For the link prediction task, we shared the last layer of the decoder as a predictor, and for the node classification task, we utilized 1 layer MLP as a classifier. Also, we set the diffusion state to 10 for OGB-Collab, OGB-DDI, and OGB-PPA, and 3 for OGB-Citation2 due to the dataset's memory issue. For a fair comparison, we reported performances of all baselines and LapDiff as the mean and the standard deviation obtained from 10 independent runs with fixed random seed $0, \ldots, 9$. To simulate a real-world scenario, we did not use validation edges as input in OGB-Collab. The experiments are conducted on A100(40GB) and A40(48GB).

D Computational Complexity

Notation

- $N = |V|$: number of nodes ; $E = |\mathcal{E}|$: number of edges.
- d : input feature dimension ; h : hidden dimension of the denoiser f_θ.
- T : number of diffusion steps ($T \ll N$ in practice).

Diffusion Process Each Laplacian-smoothing step multiplies X_{t-1} by $(I - D^{-1}L)$. For a sparse adjacency ($E = O(N)$ in large graphs), this costs

$$\mathcal{O}\big(E\,d\big)\ \text{per step} \quad \Longrightarrow \quad \mathcal{O}\big(T\,E\,d\big)\ \text{overall.} \tag{32}$$

Sampling edges (edge removal) is $\mathcal{O}(E)$ per step; overall $\mathcal{O}(T\,E)$. No matrix materialization beyond the original sparse A is required. **Total (forward).**

$$\mathcal{O}\big(T\,E\,(d+1)\big) \;\approx\; \mathcal{O}(T\,E\,d)$$

Reverse (Denoising) Process The denoiser f_θ is a 2-layer MLP encoder + 3-layer MLP decoder. Given sparse adjacency, each call costs $\mathcal{O}(E\,h + N\,h^2)$. Invoked once per diffusion step, the reverse chain costs

$$\mathcal{O}\big(T\,(E\,h + N\,h^2)\big).$$

D.1 Overall Time Complexity

Combining forward and reverse,

$$\mathcal{O}\Big(T\,\big[E\,(d+h) + N\,h^2\big]\Big)$$

With typical settings ($h \approx d$, $E \gg N$, $T \leq 10$), the leading term is $T\,E\,d$, comparable to a *single* pass of a conventional L-layer GNN when $L \approx T$.

D.2 Memory Complexity

- **Parameters.** Two MLPs of width h: $\mathcal{O}(h^2)$, independent of T.
- **Activations.** We store X_t and (sparse) A_t for the current step only, so in-memory activations scale as:

$$\mathcal{O}\big(N\,d + E\big) + \mathcal{O}(N\,h) \;=\; \mathcal{O}\big(N(d+h) + E\big).$$

- **Comparison to L-layer GNN.** A standard L-layer message-passing GNN stores L intermediate node embeddings, yielding $\mathcal{O}(L\,N\,h)$ memory. LapDiff keeps a single embedding per step and can recompute forward activations (checkpointing), requiring at most $\mathcal{O}(N\,h)$—often smaller than a deep GNN when $L > T$.

Scalability. With T chosen ≤ 10, LapDiff's runtime is on par with—or lower than—deep GNNs that rely on $L \geq 10$ layers. The memory footprint remains modest due to sparse storage and step-wise recomputation.

E License of the assets

Our source code is based on PyTorch which was released under Berkeley Software Distribution (BSD) License. We implement GNN-based baselines using PyTorch Geometric, a deep learning framework licensed under MIT. Additionally, we implement SEAL [1] and GraphMAE [2] from the official GitHub repository under MIT License. Both BSD license and MIT license can be used or redistributed under stipulated conditions. Moreover, we conduct experiments on four benchmark datasets from Open Graph Benchmark (OGB). OGB is released under MIT License. We visualize significant results by using Matplotlib where the license is based on Python Software Foundation (PSF) license.

F Broader Impact

LapDiff aims to capture latent factors for graph representation learning. It provides a strong foundation for future models that aim to understand the rich information of relational data in graphs. Our model would not only be capable of discerning the latent structures within graph data but also adept at applying this knowledge across a broad spectrum of applications. With its Laplacian smoothing noise which is structure-aware, LapDiff contributes to the ongoing discourse on data privacy. By generating representations that respect the underlying structure of data without compromising individual privacy, LapDiff aligns with the ethical use of data in AI. Also, our model's versatility suggests broad applicability beyond traditional domains, offering potential breakthroughs in any field that benefits from understanding complex networks, including neuroscience, epidemiology, and environmental studies. However, LapDiff needs to be used carefully for graph representation learning tasks such as link prediction or node classification in social networks where privacy and anonymity are important.

[1] `https://github.com/facebookresearch/SEAL_OGB`

[2] `https://github.com/THUDM/GraphMAE2`

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 abstract and introduction explicitly state the development of the LapDiff with Laplacian-based diffusion, and the claims match the theoretical derivations and experiments reported (see Section 1 and 2)

   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: Limitations are mentioned regarding scalability and assumptions in the diffusion process. (See Section 5)

   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: [Yes]

Justification: The derivations of the ELBO and loss function are presented in detail in the Appendix. Assumptions about Markov properties and Laplacian smoothing are clearly stated.

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: Experimental settings, datasets (OGB), and baselines are described in the main text and Appendix. Hyperparameters and complexity analysis are also provided in Appendix. We provide codes as Supplementary.

   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 paper states that implementations are based on PyTorch and PyTorch Geometric with clear licenses (Appendix), but the code is released only to reviewers at submission time.

   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: Training procedures, loss components, and hyperparameter sensitivity analyses are described in Appendix. Optimizer details are also outlined in Algorithm 1.

   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: The results are provided with mean and standard deviation under 10 fixed 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: Computational and memory complexity analyses, GPU are detailed in 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 datasets (OGB) and libraries used are all properly licensed. No ethical violations are apparent in the methodology.

   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: In Appendix, we discussed privacy implications and broad applicability. Negative societal impacts such as misuse are only briefly mentioned, but at least acknowledged.

    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 model is not released as a pretrained asset with high misuse risk. Only standard datasets and code bases are used.

    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: Appendix E provides details about BSD/MIT/PSF licenses of all used assets and repositories.

    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: [NA]

    Justification: No new datasets or assets are introduced.

    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 human subjects or crowdsourcing.

    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 research does not involve 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: LLMs were not used as part of the methodology.

    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.