Prior-Informed Flow Matching for Graph Reconstruction

Anonymous authors
Paper under double-blind review

Abstract

We introduce *Prior-Informed Flow Matching (PIFM)*, a conditional flow model for graph reconstruction. Reconstructing graphs from partial observations remains a key challenge; classical embedding methods often lack global consistency, while modern generative models struggle to incorporate structural priors. PIFM bridges this gap by integrating embedding-based priors with continuous-time flow matching. Grounded in a permutation equivariant version of the distortion-perception theory, our method first uses a prior, such as GraphSAGE or node2vec, to form an informed initial estimate of the adjacency matrix based on local information. It then applies rectified flow matching to refine this estimate, transporting it toward the true distribution of clean graphs and learning a global coupling. Experiments on different datasets demonstrate that PIFM consistently enhances classical embeddings, outperforming them and state-of-the-art generative baselines in reconstruction accuracy.

1 Introduction

Graph generative models have seen remarkable progress in recent years, enabling the synthesis of realistic graph structures in domains such as drug design (Yang et al., 2024) and social networks (Grover et al., 2019). In particular, diffusion-based (Niu et al., 2020; Jo et al., 2022; Vignac et al., 2023) and flow-based (Qin et al., 2025; Eijkelboom et al., 2024) approaches have emerged as state-of-the-art. While these models excel at *unconditional* generation and property-controlled generation, their application to inverse problems, and in particular, the reconstruction of a graph from partial observations, remains a fundamental open problem.

Conditional graph reconstruction aims to recover the topology of a graph from a partially observed set of edges while ensuring that the reconstructed graph aligns with the global distribution of valid graphs. The most common formulation of this problem is link prediction, which typically assumes that the locations of the missing edges are known. However, existing link prediction methods generally model each missing connection independently, failing to enforce global consistency across the reconstructed graph. Classical embedding techniques, such as Node2Vec (Grover & Leskovec, 2016; Perozzi et al., 2014), and inductive architectures, like GraphSAGE (Hamilton et al., 2017), learn expressive local representations for this task but do not explicitly capture dependencies between edge predictions or preserve global structural properties. More recently, diffusion and flow-based generative models have been adapted for graph inpainting and posterior-guided sampling (Vignac et al., 2023; Trivedi et al., 2024; Sharma et al., 2024; Tenorio et al., 2025). While these approaches use powerful graph priors to generate globally plausible completions, they are primarily designed for constrained generation rather than faithful, observation-driven reconstruction. Consequently, a gap remains: existing methods either rely on local, decoupled edge predictions or generate structurally coherent samples that lack reconstruction fidelity.

In this work, we introduce **Prior-Informed Flow Matching (PIFM)**, a conditional flow model that treats a local link predictor as an informative source distribution and learns a global correction map to the clean graph distribution. Our approach combines the strengths of classical methods with modern generative frameworks; it uses learned structural graph priors (link predictors and latent graph models) while explicitly coupling edge predictions through a global, flow-based reconstruction framework. We formalize this design through the lens of the distortion-perception tradeoff (Blau & Michaeli, 2018), which motivates a two-stage estimator (Freirich et al., 2021; Ohayon et al., 2025): (i) Approximate the Minimum Mean Squared Error (MMSE) estimator using local information. For this, we employ baseline models such as GraphSAGE (Hamilton et al.,

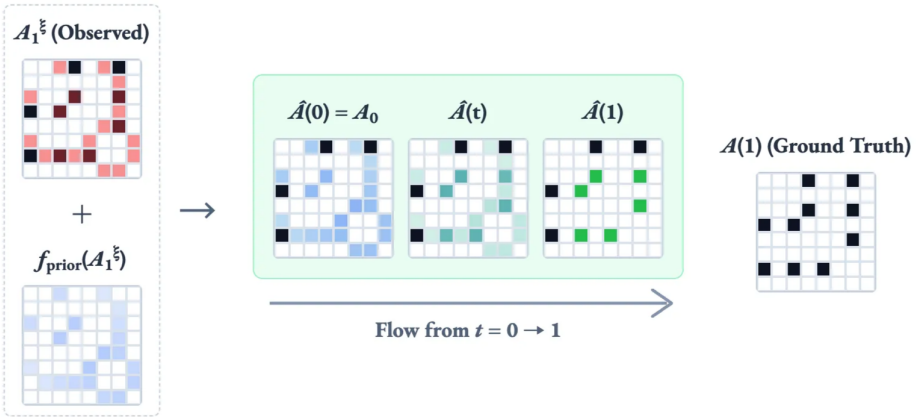


Figure 1: Overview of the **Prior-Informed Flow Matching (PIFM)** graph reconstruction framework. Starting from a partially observed adjacency matrix $\mathbf{A}^{\mathcal{O}} = \xi \odot \mathbf{A}$, where ξ denotes a mask, we form an initialization $\mathbf{A}_0$ by combining the observed entries with prior predictions $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})$ obtained with an element-wise predictor. In dark red we denote the true edges that are masked, while in light red those masked positions where there is no edge between nodes. A rectified flow then interpolates linearly from $\mathbf{A}_0$ to the ground-truth graph $\mathbf{A}_1 = \mathbf{A}$, learning global structural information from a coupling of all the edges. The intermediate states $\mathbf{A}_t$ improve on the prior-informed initialization, enabling recovery of the missing edges.

2017), Node2Vec (Grover & Leskovec, 2016), VGAE (Kipf & Welling, 2016), or NCNC (Wang et al., 2024); (*ii*) Learn a rectified flow (Liu et al., 2023; Albergo et al., 2023) that refines this initial estimate by explicitly modeling the dependencies among edge predictions.

We validate the advantages of PIFM through experiments on datasets with diverse structural characteristics, including both dense and sparse graphs. Our results demonstrate that PIFM effectively integrates local architectural priors with global flow-based modeling, offering a principled framework for graph reconstruction.

Our contributions are as follows:

- We formulate conditional graph reconstruction through the lens of the permutation-equivariant distortion–perception trade-off.
- We propose PIFM, a two-stage reconstruction framework that first constructs a prior-informed source distribution using embeddings from latent graph models, and then refines these predictions with a flow-matching model.
- We show that existing link prediction methods primarily rely on local edge-wise predictions and lack mechanisms to enforce global structural consistency. PIFM addresses this limitation by learning a generative model that integrates information across the full graph during reconstruction.
- We empirically validate our approach on link prediction and two blind reconstruction settings, termed *expansion* (recovering missing edges) and *denoising* (removing spurious edges), showing consistent improvements over predictors based solely on local information.

2 Related Work

Flow/Diffusion models on graphs. Diffusion and flow-based graph generative models have shown impressive performance in recent years. Early models, namely EDP-GNN (Niu et al., 2020) and GDSS (Jo et al., 2022), employ score-based *continuous* diffusion over a relaxation of the graph structure. However, given that graphs are inherently discrete, subsequent work has explored discrete diffusion processes (Austin et al., 2021). Models like DiGress (Vignac et al., 2023) demonstrated the effectiveness of this approach, which has been further advanced by discrete flow-based models like DeFoG (Qin et al., 2025) and variational approaches like CatFlow (Eijkelboom et al., 2024). A common point of these models is their reliance on a simple source distribution, such as Gaussian (continuous) or uniform (discrete) noise. While effective for

unconstrained generation, recent work on image-based inverse problems demonstrates the advantages of learning a data-dependent flow, using a prior-informed source distribution (Albergo et al., 2023; Delbracio & Milanfar, 2024; Ohayon et al., 2025). In the graph domain, concurrent work (Wijesinghe et al., 2026) also departs from a simple source distribution by employing structural graphette priors, though it targets unconditional graph generation rather than reconstruction from partial observations.

Graph topology inference via flow/diffusion-based solvers. Graph topology inference – the task of recovering hidden edges from a partially observed graph – is a long-standing inverse problem (Segarra et al., 2017; Dong et al., 2016). Several methods adapt diffusion for constrained graph generation, which is related to but distinct from topology inference. DiGress (Vignac et al., 2023) introduced an inpainting mechanism, inspired by Repaint (Lugmayr et al., 2022), to generate graph structures consistent with a partial observation. Similarly, in Trivedi et al. (2024) a similar mechanism is used for completing partially observed graphs. PRODIGY (Sharma et al., 2024) enforces hard constraints by projecting the graph estimate onto a feasible set at each sampling step. More recently, GGDiff (Tenorio et al., 2025) incorporates a guidance mechanism as a flexible alternative to inpainting. However, all these methods are designed for *constrained generation* (e.g., molecule generation with a given scaffold) rather than *recovering masked edges from a partially observed graph*. Hence, to the best of our knowledge, designing a diffusion-based model explicitly for graph topology inference remains an open problem.

Link prediction and Graph Autoencoders. Graph reconstruction from partial observations is closely related to link prediction, which aims to infer whether unobserved edges should exist in a partially observed graph (Newman, 2001; Adamic & Adar, 2003; Zhou et al., 2009). Classical approaches rely on topology based heuristics, while embedding based methods such as DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) learn node representations from random walks and score candidate edges from pairwise embedding features. Graph neural network (GNN) methods further improve link prediction by learning node representations through message passing. For example, GraphSAGE learns inductive neighborhood aggregation functions that can generalize across nodes or graphs (Hamilton et al., 2017), while variational graph autoencoders (VGAE) encode the observed graph into latent node embeddings and decode pairwise edge probabilities (Kipf & Welling, 2016). Other methods model pairwise or path based structure more directly, such as NBFNet, which formulates link prediction as neural path aggregation through a Bellman Ford style recurrence (Zhu et al., 2021).

3 Background

We represent an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the nodes and $\mathcal{E}$ the edges, by its binary symmetric adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. Table 1 summarizes the main notation used throughout the method.

Table 1: Notation used in PIFM.

Symbol	Meaning
$\mathbf{A}_1$	Clean ground-truth adjacency matrix
$\mathbf{A}^{\mathcal{O}}$	Observed partial adjacency matrix
ξ	Binary observation mask
$\hat{\mathbf{A}}^*$	Prior estimate or approximate MMSE estimate
$\mathbf{A}_0$	Prior-informed source sample for flow matching
$\mathbf{A}_t$	Intermediate state along the flow path
$\widehat{\mathbf{A}}$	Final reconstructed adjacency matrix

3.1 Continuous flow matching for graph generation

Flow matching (Albergo et al., 2023; Lipman et al., 2023) is a family of generative models that defines a continuous-time transport map from samples $\mathbf{A}_0$ drawn from a source distribution p_0 to samples $\mathbf{A}_1$ from a

target distribution p_1. It is governed by the ODE

$$d\mathbf{A}_t = v(\mathbf{A}_t, t)\, dt, \tag{1}$$

where $v(\cdot, t)$ is a velocity field and $\mathbf{A}_t$ denotes a forward process, also known as stochastic interpolant, for $t \in [0, 1]$. Typically, p_0 is a tractable distribution (e.g., a Gaussian distribution), while p_1 corresponds to the data distribution. To generate new samples, one must specify both $\mathbf{A}_t$ and v. A common choice for the forward process is $\mathbf{A}_t = \alpha_t \mathbf{A}_0 + \beta_t \mathbf{A}_1$, where α_t and β_t are differentiable functions such that $\alpha_0 = 1$, $\beta_0 = 0$ and $\alpha_1 = 0$, $\beta_1 = 1$. Differentiating this path gives a velocity $v(\mathbf{A}_t, t) = \dot{\alpha}_t \mathbf{A}_0 + \dot{\beta}_t \mathbf{A}_1$. Despite its closed-form, this expression depends explicitly on $\mathbf{A}_1$, making it impractical since the target is unknown at inference/sampling. To circumvent this, we instead consider $v(\mathbf{A}_t, t) = \mathbb{E}_{\mathbf{A}_0, \mathbf{A}_1}[\dot{\alpha}_t \mathbf{A}_0 + \dot{\beta}_t \mathbf{A}_1 \mid \mathbf{A}_t]$, the conditional expectation of the velocity given $\mathbf{A}_t$ (Albergo et al., 2023), which is then approximated with a neural network v_θ. The network is trained using a mean squared error loss:

$$\mathbb{E}_{t, \mathbf{A}_0, \mathbf{A}_1} \left[\left\| v_\theta(\mathbf{A}_t, t) - (\dot{\alpha}_t \mathbf{A}_0 + \dot{\beta}_t \mathbf{A}_1) \right\|_2^2 \right]. \tag{2}$$

In particular, this formulation does not require $\mathbf{A}_0$ and $\mathbf{A}_1$ to be independent; in fact, they might be sampled from a joint distribution, allowing for richer transport plans in cases where paired data is available. This has been exploited to solve inverse problems on images (Ohayon et al., 2025; Albergo et al., 2024; Delbracio & Milanfar, 2024), and is directly related to our proposed method, as described later.

Throughout this work, we consider the *rectified flow* case (Liu et al., 2023), where $\alpha_t = 1 - t$ and $\beta_t = t$. As shown in Tong et al. (2024), the velocity field associated with this linear path approximates the optimal transport vector field when the joint distribution $p(\mathbf{A}_0, \mathbf{A}_1)$ closely resembles the optimal coupling between the marginals $p(\mathbf{A}_0)$ and $p(\mathbf{A}_1)$. We defer to Appendix B.5.1 a more detailed background on generative models on graphs beyond continuous flow matching, including diffusion-based models, as well as related works.

3.2 Distortion–perception formulation for conditional graph reconstruction

Our goal is to reconstruct the ground-truth adjacency matrix $\mathbf{A}$ of a graph $\mathcal{G}$ from a partially observed version, denoted by $\mathbf{A}^{\mathcal{O}}$. Similarly to image restoration, there are two complementary criteria to assess the quality of reconstructed graphs: (i) the average closeness of the reconstruction to the ground truth, measured through a distortion metric, and (ii) the similarity in distribution between reconstructed and real graphs, namely, how well the reconstructed graph resembles a sample from the underlying graph distribution.

These two objectives are generally at odds and can be formalized through the perception–distortion trade-off introduced in Blau & Michaeli (2018):

$$D(P) = \min_{p(\hat{\mathbf{A}} \mid \mathbf{A}^{\mathcal{O}})} \left\{ \mathbb{E}_{p(\mathbf{A}, \hat{\mathbf{A}})} \left[\Delta(\mathbf{A}, \hat{\mathbf{A}}) \right] \;:\; d(p_{\mathbf{A}}, p_{\hat{\mathbf{A}}}) \leq P \right\}, \tag{3}$$

where $\hat{\mathbf{A}}$ denotes an estimator of $\mathbf{A}$ conditioned on the observation $\mathbf{A}^{\mathcal{O}}$, $\Delta(\cdot, \cdot)$ is a distortion metric (e.g., MSE), and $d(p_{\mathbf{A}}, p_{\hat{\mathbf{A}}})$ measures the divergence between the distribution of clean graphs $p_{\mathbf{A}}$ and the distribution of reconstructed graphs $p_{\hat{\mathbf{A}}}$.

The function in (3) has been extensively studied in the image domain, where different values of P yield estimators with different trade-offs between reconstruction fidelity (distortion) and perceptual realism. We adapt this framework to graph reconstruction by estimating the adjacency matrix $\mathbf{A}$ while accounting for the symmetry constraints induced by the permutation invariance of graph representations.

MSE distortion. We first consider the distortion metric $\Delta(\mathbf{A}, \hat{\mathbf{A}}) = \|\mathbf{A} - \hat{\mathbf{A}}\|_F^2$, corresponding to the mean squared error (MSE). This setting was studied in Freirich et al. (2021), particularly for the two extreme cases $P = \infty$ and $P = 0$. When $P = \infty$, the problem reduces to minimizing distortion without imposing perceptual constraints. The optimal estimator is the posterior mean

$$\hat{\mathbf{A}}^* = \mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}],$$

which minimizes the expected MSE. Although optimal in terms of distortion, this estimator may produce unrealistic graphs whose structural properties deviate from those of the underlying graph distribution.

At the other extreme, $P = 0$ enforces perfect perceptual consistency, meaning that the reconstructed graphs follow exactly the same distribution as the ground-truth graphs, i.e., $p_{\hat{\mathbf{A}}} = p_{\mathbf{A}}$. As shown in Freirich et al. (2021), the corresponding estimator can be characterized through the optimal transport problem

$$p^*_{\hat{\mathbf{A}},\hat{\mathbf{A}}^*} = \underset{p \in \Pi(p_{\mathbf{A}}, p_{\hat{\mathbf{A}}^*})}{\operatorname{argmin}} \mathbb{E}\left[\|\hat{\mathbf{A}} - \hat{\mathbf{A}}^*\|_F^2\right], \tag{4}$$

where $\Pi(p_{\mathbf{A}}, p_{\hat{\mathbf{A}}^*})$ denotes the set of couplings with marginals $p_{\mathbf{A}}$ and $p_{\hat{\mathbf{A}}^*}$.

For the special case of squared-error distortion with the Wasserstein-2 perception index, Freirich et al. (2021) show that the estimator at any point on the distortion–perception curve can be obtained by interpolating the two extremes, $\hat{\mathbf{A}}_p = (1 - \frac{P}{P_\infty})\hat{\mathbf{A}} + \frac{P}{P_\infty}\hat{\mathbf{A}}^*$, where $\hat{\mathbf{A}}$ is the perfect-perception ($P = 0$) estimator, $\hat{\mathbf{A}}^*$ is the MMSE estimator, and P_∞ is the perception index attained by $\hat{\mathbf{A}}^*$. Therefore, obtaining the estimator associated with $D(0)$ amounts to solving an optimal transport problem between the distribution of clean graphs $p_{\mathbf{A}}$ and the distribution of the MMSE estimator $p_{\hat{\mathbf{A}}^*}$. In practice, we approximate a $P = 0$ estimator by (i) computing the posterior mean $\hat{\mathbf{A}}^* = \mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]$, and (ii) learning a transport from $p_{\hat{\mathbf{A}}^*}$ to $p_{\mathbf{A}}$ with the rectified flow described below. The learned map is trained to push $p_{\hat{\mathbf{A}}^*}$ toward $p_{\mathbf{A}}$, i.e., to approach perfect perception ($P = 0$); this is only approximate, both because of finite model capacity and because a continuous, invertible flow cannot exactly match the discrete distribution of binary adjacency matrices. We likewise do not claim it recovers the exact optimal-transport coupling of (4). Intuitively, this procedure refines the coarse estimate $\hat{\mathbf{A}}^*$ into a graph that is consistent with the underlying data distribution.

We focus on the $P = 0$ case: while $P = \infty$ is MSE-optimal, it may produce graphs that violate structural constraints (e.g., chemical validity in molecule generation), whereas $P = 0$ guarantees distributional consistency.

Cross-entropy distortion. We also consider a cross-entropy-based distortion, which is particularly natural for graph reconstruction since adjacency matrices are discrete-valued objects. Specifically, letting $p^d_\theta(k \mid \mathbf{A}_t, t)$ denote the categorical distribution induced by the flow model for edge variable d, we define

$$\Delta(\mathbf{A}, \hat{\mathbf{A}}) = -\sum_{d=1}^{D}\sum_{k=1}^{K^d} \mathbb{I}[A^d = k] \log p^d_\theta(k \mid \mathbf{A}_t, t),$$

which corresponds to the cross-entropy between the ground-truth adjacency matrix and the predicted edge distributions, and where $D = \frac{N(N-1)}{2}$ denotes the number of edge variables. Unlike the MSE case, this formulation does not admit the same closed-form characterization of the perception–distortion trade-off. Nevertheless, as we demonstrate experimentally, it exhibits qualitatively similar behavior, yielding a comparable trade-off between reconstruction fidelity and perceptual consistency.

4 Method

PIFM can be viewed as a two-stage refinement framework for conditional graph reconstruction, motivated by the distortion–perception formulation introduced in Section 3.2. First, a structural prior estimates the conditional mean $\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]$ from the observed subgraph $\mathbf{A}^{\mathcal{O}}$ by predicting marginal probabilities for the masked edges. While this estimate minimizes distortion locally, it does not necessarily produce a globally consistent graph. To address this, PIFM uses the resulting adjacency estimate as the source state of a rectified flow that learns to refine it toward the true graph distribution, recovering higher-order structural dependencies not captured by edge-wise predictions alone. The flow is trained on paired samples $(\mathbf{A}_0, \mathbf{A}_1)$ constructed from the same underlying graph, where $\mathbf{A}_0$ combines observed edges with the prior predictions on the masked region. At inference time, only $\mathbf{A}^{\mathcal{O}}$ and ξ are available: the prior produces the initial estimate, and the flow integrates the learned velocity field to generate the final graph reconstruction.

4.1 Approximating the posterior mean

As discussed in Section 3.2, our goal is to approximate the conditional mean $\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]$ with a *permutation equivariant* estimator. Before moving to particular parameterizations of the conditional mean, we introduce two assumptions.

AS 1. *We assume each edge in $\mathbf{A} \in \{0,1\}^{n\times n}$ follows a Bernoulli distribution whose probabilities depend on latent node variables $\mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathcal{Z}$ such that:*

$$A_{ij} \;\sim\; \textit{Bernoulli}\big(f(\mathbf{z}_i, \mathbf{z}_j)\big), \qquad 1 \le i < j \le n. \tag{5}$$

The function f maps pairs of latent variables to edge probabilities, i.e., $f(\mathbf{z}_i, \mathbf{z}_j) = P(A_{ij}|\mathbf{z}_i, \mathbf{z}_j) = p_{ij}$.

AS 2. *We assume that the edges are conditionally independent given the latent structure, i.e., given the latent structure $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, we have:*

$$P(\mathbf{A} \mid Z) = \prod_{1\le i<j\le n} P(A_{ij} \mid \mathbf{z}_i, \mathbf{z}_j). \tag{6}$$

Under these two assumptions, and assuming access to the mapping $\mathbf{z}^{-1} : \mathbf{A} \to \mathcal{Z}$ that recovers the latent coordinates $Z = \mathbf{z}^{-1}(\mathbf{A}^{\mathcal{O}})$, the posterior mean can be computed element-wise: on observed entries it equals the known value $A^{\mathcal{O}}_{ij}$, while on masked entries $\big[\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]\big]_{ij} = P(A_{ij} = 1 \mid \mathbf{z}_i, \mathbf{z}_j) = f(\mathbf{z}_i, \mathbf{z}_j)$.

We adopt two different types of priors: (i) inductive methods, represented by *GraphSAGE* (Hamilton et al., 2017), a GNN-based estimator, and (ii) transductive ones, obtained from *node2vec* (Grover & Leskovec, 2016), which provides an instance-level learned probabilistic model.

Posterior mean using inductive methods (dataset-informed). We approximate the posterior mean using a dataset-informed, inductive approach based on *GraphSAGE* (Hamilton et al., 2017). We train the model on the partially observed graphs in the dataset to produce node embeddings $\{\mathbf{z}_i\}_{i=1}^N$. From these embeddings, we train a single logistic predictor on Hadamard edge features $(\mathbf{z}_i \odot \mathbf{z}_j)$ to estimate edge probabilities. The resulting conditional mean is parameterized as $\big[\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]\big]_{ij} \approx f_\phi(\mathbf{z}_i \odot \mathbf{z}_j)$.

Posterior mean using transductive methods (instance-specific). We use three per-graph priors that fit an embedding and a predictor specific to each graph instance from the partially observed graph $\mathbf{A}^{\mathcal{O}}$. *node2vec* (Grover & Leskovec, 2016) learns embeddings $\{\mathbf{z}_i\}_{i=1}^N$ from biased random walks; a per-graph logistic predictor on Hadamard edge features then gives $\big[\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]\big]_{ij} \approx f_\phi(\mathbf{z}_i \odot \mathbf{z}_j)$. *VGAE* (Kipf & Welling, 2016) uses a GCN encoder to produce a Gaussian posterior over per-node latents $q(\mathbf{z}_i \mid \mathbf{A}^{\mathcal{O}}) = \mathcal{N}(\boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2))$ and an inner-product decoder, yielding $\big[\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]\big]_{ij} \approx \sigma(\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j)$, trained by ELBO maximization with a standard-normal prior. *NCNC* (Wang et al., 2024) is a neural common-neighbor link predictor that combines a GCN encoder with a pooled sum over softly-completed common-neighbor representations, giving $\big[\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]\big]_{ij} \approx \sigma(\mathrm{NCNC}(i, j;\, \mathbf{A}^{\mathcal{O}}, \mathbf{X}))$.

4.2 Learning the flow model

We now learn a flow model that approximates the joint distribution (coupling) $p(\mathbf{A}, \hat{\mathbf{A}}^*)$ of the clean graph and its MMSE estimate. As explained in Section 3, we need to specify the forward path $\mathbf{A}_t$ and the velocity field v. Inspired by Ohayon et al. (2025), we incorporate prior information as the initialization of the forward path; with slight abuse of notation, we denote f_{prior} as the prediction of the full graph (i.e., $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) \approx \mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]$). Specifically, we compute the sample $\mathbf{A}_0$ from the source distribution as follows:

$$\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \left(f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) + \boldsymbol{\epsilon}_s\right), \tag{7}$$

where $\mathbf{A}$ is the ground-truth graph, ξ is the corresponding mask (taking value 1 for the observed pairs of nodes and 0 otherwise), $\mathbf{A}^{\mathcal{O}} = \xi \odot \mathbf{A}$ is the observed graph, and $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})$ is our approximate MMSE estimator for the masked edges. We also add a small amount of noise $\boldsymbol{\epsilon}_s \sim \mathcal{N}(0, \sigma_s^2)$ following Albergo et al. (2024). We define $\mathbf{A}_1 = \mathbf{A}$ for the target distribution.

Distortion instantiations. Given the source distribution, we can learn the flow by minimizing a certain distortion $\Delta(\hat{\mathbf{A}}_\theta, \mathbf{A})$. We consider two instances following the description in Section 3.2: (i) MSE loss, which boils down to the standard rectified flow-matching loss in (2); and (ii) CE loss, where the velocity field $v_\theta(\mathbf{A}_t, t)$ parameterizes a categorical distribution over edge values, similarly to (Eijkelboom et al., 2024; Dieleman et al., 2022) and recent approaches that adapt continuous models to discrete data (Lee et al., 2026; Roos et al., 2026).

Training. For the velocity field v, we use the architecture from Jo et al. (2022), a GNN-based network that yields a permutation-equivariant parameterization (see Appendix D for details). For each training graph $\mathbf{A}_1$, we sample a mask ξ and time $t \sim U[0, 1]$, construct $\mathbf{A}_0$ via Eq. (7), and minimize the flow-matching objective with the chosen distortion Δ over the resulting paired samples $(\mathbf{A}_0, \mathbf{A}_1)$. The full training procedure is described in Alg. 1 (lines 1–6).

Inference. Given a new observed graph $\mathbf{A}^{\mathcal{O}}$ and mask ξ, we initialize $\hat{\mathbf{A}}$ from the prior prediction and integrate the learned velocity field forward using K Euler steps (Alg. 1, lines 7–12). Setting $K = 1$ yields the lowest distortion and the highest AUC-ROC, while larger K improves perceptual quality as discussed in Section 5.3. The two distortion instantiations differ only in how the velocity field is obtained from the model. Under MSE, the network directly predicts the velocity field, which is integrated at each step. Under CE, the network predicts categorical distributions over edge values, and the corresponding velocity field is obtained from these probabilities before updating the observed entries (Alg. 1, line 9).

Permutation invariance. Combined with the posterior-mean parameterization of Section 4.1, the permutation-equivariant velocity field endows PIFM with two related symmetry guarantees. First, the reconstruction map is permutation-equivariant: relabeling the observation and mask $(\mathbf{A}^{\mathcal{O}}, \xi)$, with $\mathbf{A}^{\mathcal{O}} = \xi \odot \mathbf{A}$ the observed part of the graph, produces a correspondingly relabeled reconstruction $\tilde{\mathbf{A}}_1$ (the sampler output at $t = 1$). This holds for the full sampler of Alg. 1 (both the MSE and CE branches, for any number of steps K) and needs only equivariance and well-posedness of the dynamics; it is formalized in Theorem 1. Second, in the idealized continuous-time flow the conditional density of the reconstruction is itself invariant under simultaneous relabeling of $(\tilde{\mathbf{A}}_1, \mathbf{A}^{\mathcal{O}}, \xi)$; this stronger statement is formalized in Proposition 1. Since graphs are exchangeable, the model's output should not depend on node ordering, making these symmetries a desirable inductive bias.

Theorem 1 (Permutation equivariance of the reconstruction)**.** *Assume:*

(a) the prior $f_{prior} : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ and the velocity field $v_\theta : \mathbb{R}^{N\times N} \times [0, 1] \to \mathbb{R}^{N\times N}$ are permutation-equivariant;

(b) v_θ is locally Lipschitz in its first argument (e.g., a smooth neural network on a bounded domain), so the sampling dynamics are well-posed.

For an observed graph $\mathbf{A}^{\mathcal{O}}$ supported on the observed entries (i.e., $(1 - \xi) \odot \mathbf{A}^{\mathcal{O}} = 0$) and mask ξ, let $\mathrm{PIFM}(\mathbf{A}^{\mathcal{O}}, \xi)$ be the reconstruction returned by Alg. 1 under either the MSE or the CE branch, for any number of Euler steps $K \geq 1$. Then, for every permutation matrix $\mathbf{P}$, the reconstruction is permutation-equivariant in distribution,

$$\mathrm{PIFM}(\mathbf{P}^\top \mathbf{A}^{\mathcal{O}} \mathbf{P},\, \mathbf{P}^\top \xi \mathbf{P}) \overset{d}{=} \mathbf{P}^\top \mathrm{PIFM}(\mathbf{A}^{\mathcal{O}}, \xi)\, \mathbf{P}, \tag{8}$$

with equality holding pointwise either under the coupled noise draw $\mathbf{P}^\top \boldsymbol{\epsilon}_s \mathbf{P}$ or when $\sigma_s = 0$ (as in our CE experiments).

Theorem 1 is the symmetry that matters in practice: it certifies that PIFM's reconstruction does not depend on node ordering for the algorithm exactly as implemented (both distortions, any K, including the sigmoid, clipping, and re-imposition steps of the CE branch), and it is independent of the choice of distortion $\Delta(\cdot, \cdot)$. In the idealized continuous projected flow (with a C^1, symmetry-preserving velocity and $\sigma_s > 0$), this equivariance lifts to invariance of the reconstruction's conditional density on the masked edge coordinates; we state and prove this as Proposition 1 in Appendix C.1.

Algorithm 1 Training and Sampling

Training
Sample $\mathbf{A}_1 \sim p(\mathbf{A})$, a mask ξ, and time $t \sim U[0,1]$.
Train approximate MMSE estimator: $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})$
Compute
$\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \left(f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) + \boldsymbol{\epsilon}_s\right), \quad \boldsymbol{\epsilon}_s \sim \mathcal{N}(0, \sigma_s^2)$
Compute $\mathbf{A}_t = (1-t)\mathbf{A}_0 + t\mathbf{A}_1$.
Train flow model:
$\theta^* = \operatorname{argmin}_\theta \, \mathbb{E}_{\mathbf{A}_1, \mathbf{A}_0, \xi, t}\Big[\Delta\big(\hat{y}_\theta(\mathbf{A}_t, t),\ y(\mathbf{A}_0, \mathbf{A}_1)\big)\Big]$
Sampling (Reconstruction)
Initialize $\hat{\mathbf{A}} \leftarrow \xi \odot \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) + (1-\xi) \odot \boldsymbol{\epsilon}_s, \quad \boldsymbol{\epsilon}_s \sim \mathcal{N}(0, \sigma_{\text{samp}}^2)$.
for $i \leftarrow 0, \ldots, K-1$ **do**
  if MSE distortion **then**
    $\hat{\mathbf{A}} \leftarrow \hat{\mathbf{A}} + \frac{1}{K} v_{\theta^*}\left(\hat{\mathbf{A}}, \frac{i}{K}\right)$
  else
    // CE distortion
    $\mu \leftarrow \sigma\big(v_{\theta^*}(\hat{\mathbf{A}}, \frac{i}{K})\big)$
    $\hat{\mathbf{A}} \leftarrow \text{clip}\left(\hat{\mathbf{A}} + \frac{1}{K} \cdot \dfrac{\mu - \hat{\mathbf{A}}}{1 - i/K + \varepsilon},\ 0,\ 1\right)$
    $\hat{\mathbf{A}} \leftarrow \xi \odot \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \hat{\mathbf{A}}$
  end if
end for
$\hat{\mathbf{A}} \leftarrow \xi \odot \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \hat{\mathbf{A}}$ *// re-impose observed entries*
Return $\hat{\mathbf{A}}$

The proof is in Appendix C.1. Since the reconstruction task is exchangeable in node ordering (relabeling the input observation and mask should produce a correspondingly relabeled output), Theorem 1 guarantees that PIFM has this inductive bias built in.

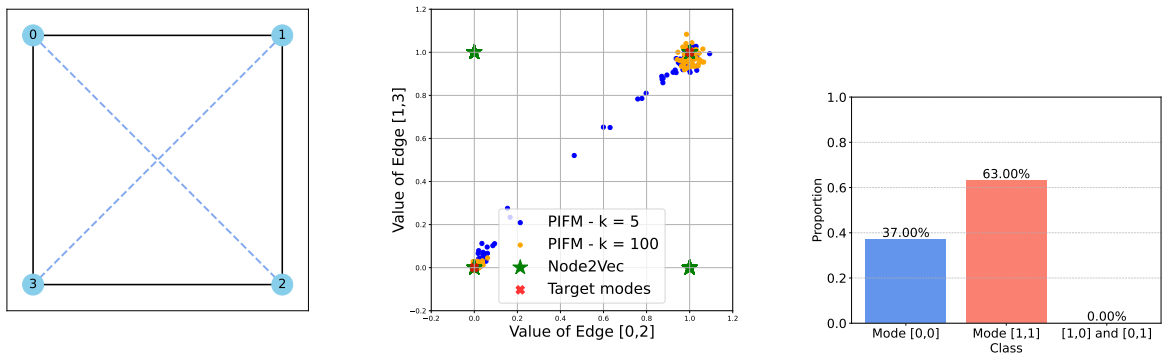
Final algorithm. In Alg. 1, we describe our training and sampling algorithms. In essence, PIFM is a general framework that learns a global graph structure to enhance simple, conditionally independent edge-wise priors.

To illustrate what we mean by *learning a global and dependent predictor*, we now describe a toy experiment. Consider a four-node graph $\mathcal{G}$ (see Fig. 2 (a)) where the goal is to predict the diagonal edges under a specific constraint: the only valid outcomes are either both edges are present or both are absent, i.e., $\mathcal{E} = \{[e_{02} = 1, e_{13} = 1], [e_{02} = 0, e_{13} = 0]\}$. Moreover, we assume that the probability of observing the first case is 0.6, while the second one is 0.4.

We first train an edge-wise prior using node2vec, which yields a probability of 0.6 for each diagonal edge. Crucially, because node2vec models each edge prediction independently, this prior is misspecified. A standard predictor based on this prior would always predict $[1, 1]$ if used as conditional mean or, if sampling were to be performed, could generate invalid predictions such as $[1, 0]$.

We then train a flow model using this node2vec prior to construct the initial state $\mathbf{A}_0$ as in (7). After training (see Appendix D for details), we generate 200 samples, illustrated in Fig. 2(b); the proportion of each mode is shown in Fig. 2(c). The results clearly demonstrate that the flow model (i) successfully leverages global information, learning a *probabilistic coupling* between the edges, to generate samples only from the two valid states, and (ii) learns the probability of each mode.

Computational cost. We remark that the flow model training time of PIFM is the **same** as flow model training time of PIFM with Gaussian prior, meaning that there is no additional cost when training the flow model; the only added expense is training the prior when one is not already available. Inference is roughly 0.04–0.05s per graph at $K = 1$ on an NVIDIA A100 (vs. 0.0002–0.002s for the priors alone) and scales linearly in K (e.g., ∼2.8s at K=100); full timings are in Appendix D.5 and D.6.



(a) (b) (c)

Figure 2: Toy experiment showcasing the advantage of PIFM (in this case, for link prediction). a) Graph $\mathcal{G}$ with four nodes, where the hidden edges are e_{02} and e_{13}. b) Generated samples by using node2vec and PIFM (our proposed method): clearly, our method learns a probabilistic coupling, rendering a model that generates only the two valid modes. c) Proportions of samples generated with PIFM from each mode; remarkably, the method also learns a good approximation of the probability of each mode.

Source distribution and prior model. PIFM is agnostic to the prior architecture: any model that produces edge-wise marginal probabilities on the masked region can serve as the source distribution. In practice, we recommend validating candidate priors independently before flow training and selecting the strongest available edge predictor for the target domain.

5 Experiments

5.1 Setup

We evaluate our method on IMDB-B, PROTEINS, and ENZYMES (Morris et al., 2020), three inductive graph datasets (training and testing on disjoint graphs), and on CORA (McCallum et al., 2000; Yang et al., 2016), which is transductive and large-scale, adding additional challenges for training the flow-matching module. Thus, we focus on families of graphs that are diverse to show that our model learns a general predictor. For the inductive datasets, each one is split into 85% train, 10% validation, and 5% test graphs, and we evaluate reconstruction quality under two masking levels (10% and 50% of edges, with masks generated uniformly at random). The implementation details are provided in Appendix D.

Evaluation metrics. Performance is measured exclusively on masked edges. We report both threshold-dependent classification metrics (FPR, FNR) and threshold-independent metrics (ROC-AUC, AP). Threshold-dependent metrics are computed by binarizing predictions at a fixed cutoff of 0.5, so entries with predicted probability ≥ 0.5 are treated as edges and those < 0.5 as non-edges. In addition, we use maximum mean discrepancy (MMD) (O'Bray et al., 2022) to compute the distance between the generated graphs and the ground truth, serving as a proxy for computing the perception quality. More details on these metrics are deferred to Appendix D.4.

Baselines. We compare PIFM against several baselines, including diffusion-based. Recall that PIFM is composed of a one-shot prediction used as prior followed by a flow model. We compare PIFM to the accuracy of the one-shot prediction (without the flow) and with a flow with a random starting point:

- **Node2Vec Prior (Grover & Leskovec, 2016)/GraphSAGE Prior (Hamilton et al., 2017)/NCNC Prior (Wang et al., 2024)**: one-shot predictions using the structural prior directly.
- **Flow with Gaussian prior**: flow model initialized from Gaussian $\mathcal{N}(0.5, 1)$ noise on masked entries.
- **DiGress + RePaint (Vignac et al., 2023)**: unconditional DiGress combined with RePaint-style resampling (Lugmayr et al., 2022).

- **GDSS + RePaint (Jo et al., 2022)**: unconditional GDSS combined with RePaint-style resampling (Lugmayr et al., 2022).
- **VGAE** (Kipf & Welling, 2016): a graph autoencoder that learns latent node embeddings from the observed graph and reconstructs edges using a probabilistic decoder.

Algorithmic details of the baselines are provided in Appendix A.

5.2 Link Prediction

Table 2: Graph reconstruction performance with **50% of edges masked**. We report results for PIFM trained with both the squared Frobenius distortion (MSE loss) and the cross-entropy distortion (CE loss). We report AUC, Average Precision (AP↑), False Positive Rate (FPR↓), and False Negative Rate (FNR↓), all in percent (%). The best result for each metric is in **blue** and the second best is green.

Mask Rate: 50% (0.5 Drop)												
	ENZYMES				PROTEINS				IMDB-B			
Method	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓
Baselines												
Node2Vec	19.14	55.22	46.37	44.29	23.51	53.83	51.37	44.36	54.20	52.22	48.41	47.51
GraphSAGE	22.79	57.77	40.02	52.16	27.71	53.99	32.16	66.86	75.74	75.54	18.18	44.86
VGAE	25.35	63.50	36.45	41.24	32.22	64.20	39.42	41.90	67.37	67.30	39.07	35.98
NCNC	28.22	61.11	95.54	1.11	36.28	62.22	92.47	0.99	80.49	74.85	74.65	**1.70**
DiGress + RePaint	17.34	55.22	77.95	11.62	23.65	55.45	71.46	17.65	56.47	58.89	73.00	10.27
GDSS + RePaint	16.43	49.65	69.45	30.46	22.33	51.42	66.44	32.23	53.39	51.20	69.35	29.22
Flow w/ Gaussian prior	17.43	51.84	98.49	1.07	26.40	55.55	93.21	5.25	78.72	79.76	41.56	14.62
Ours (MSE loss)												
PIFM (Node2Vec)	22.95	59.14	90.71	3.53	27.57	59.68	87.05	8.98	84.46	85.71	32.95	15.03
PIFM (GraphSAGE)	25.44	61.36	95.62	1.86	35.50	60.61	85.05	10.23	**93.13**	**93.84**	17.52	7.61
PIFM (VGAE)	27.47	65.69	98.23	**0.85**	35.42	67.43	82.37	**0.81**	89.39	87.10	19.43	14.98
PIFM (NCNC)	28.61	64.94	89.09	3.50	**37.30**	**68.56**	78.40	7.94	89.22	88.62	26.81	11.24
Ours (CE loss)												
PIFM (GraphSAGE)	25.23	60.27	42.41	45.47	29.75	59.13	32.58	57.18	89.80	90.90	**11.60**	22.10
PIFM (VGAE)	28.70	66.90	**22.80**	59.50	35.10	65.80	**5.90**	88.70	91.70	90.10	13.50	18.90
PIFM (NCNC)	**29.21**	**66.91**	55.96	20.70	37.12	68.12	44.03	27.39	88.81	87.34	19.62	17.78

Table 2 reports results for 50% masking; 10% results are deferred to Appendix E.1. Overall, PIFM improves the AUC-ROC of all base priors (node2vec, GraphSAGE, VGAE, and NCNC), highlighting the benefit of the flow module in capturing global graph structure beyond local edge-wise predictions. Informed priors consistently outperform the Gaussian-prior flow, validating the use of prior predictions as informative initializations. We further observe that different priors dominate different graph regimes: PIFM(NCNC) achieves the strongest AUC on the sparse bioinformatics datasets (ENZYMES and PROTEINS), while PIFM(GraphSAGE) remains strongest on the dense social-network dataset (IMDB-B). The standalone NCNC prior is itself competitive on these benchmarks, but the flow refinement still yields additional gains.

All PIFM results in this section use $K = 1$, which yields the lowest distortion and highest AUC-ROC (consistent with Section 3.2); perceptual quality improves with more integration steps as shown in Section 5.3 and Appendix E.5. Even with $K = 1$, PIFM outperforms the underlying priors despite their approximate MMSE optimality, indicating that the flow refinement captures global dependencies missed by the local predictors.

Notice we consider the two distortion instantiations introduced in Section 4.2. While MSE and CE achieve comparable AUC-ROC, they exhibit different thresholding behavior on sparse graphs. In particular, CE significantly reduces the false negative rate (e.g., $95.62 \rightarrow 42.41$ on ENZYMES with GraphSAGE; $98.23 \rightarrow 22.80$ with VGAE), at the expense of a higher false positive rate. This behavior stems from the loss itself: MSE regression toward sparse binary targets tends to collapse predictions toward zero, pushing many true edges below the decision threshold, whereas CE with positive-class weighting yields better calibrated probabilities. As a result, MSE is preferable when prioritizing ranking metrics such as AUC/AP, while CE provides more balanced binary reconstructions on sparse datasets. Additional ablations are provided in Appendix E.4.

Finally, these experiments illustrate that the importance of the prior strongly depends on the difficulty of the reconstruction problem. The gap between PIFM and the Gaussian-prior baseline increases substantially with the masking rate (much larger at 50% than at 10% on IMDB), consistent with the standard ill-posedness principle that priors become more important as observations become scarcer (Appendix E.9). More expressive priors also lead to consistently stronger reconstructions, further highlighting the complementary role between local predictive representations and global flow-based refinement.

Large-scale graph reconstruction. We additionally evaluate PIFM on Cora. Following Limnios et al. (2023), we use a subgraph-based training approach where each training example is constructed by sampling an edge-centered k-hop ego-subgraph (with a maximum node cap). Within the subgraph, we treat all other training edges as observed context and mask the centered edge as the supervision target. Then, PIFM is initialized using NCN (Wang et al., 2024) as structural prior and trained with the squared Frobenius distortion (MSE loss); experimental details are deferred to Appendix E.8. As shown in Table 3, PIFM outperforms the structural prior on held-out edges, albeit by a narrower margin than observed in the previous inductive datasets. This suggests that because the initial prior already achieves high performance on Cora, the marginal utility of the additional flow is somewhat diminished in this specific context.

Table 3: Link prediction on Cora (test edges split) with an NCN structural prior. We report AUC and Average Precision (AP) on the test edge split. Best results of each metrics are in **bold**. Refer to Table 12 for details about hyperparameters in each PIFM setting. PIFM uses MSE loss on Cora experiments.

Method	AUC↑	AP↑	FPR↓	FNR↓
NCN Wang et al. (2024)	93.78	**93.89**	23.50	**6.71**
PIFM (1-hop subgraphs)	**93.89**	93.71	**20.95**	7.69
PIFM (2-hop subgraphs)	93.82	93.75	22.04	6.98

5.3 Blind graph reconstruction

We focus on two blind versions of link prediction, namely expansion and denoising. In the expansion case, we only get to observe a subset of the edges (but no non-edges), and we need to determine which other entries correspond to existing edges. Conversely, for denoising, we observe a set of edges that includes some spurious ones (but no confirmed non-edges), and we need to determine which of the observed edges are spurious and should be removed. These cases are more challenging than link prediction since transductive priors like node2vec cannot be trained on the masked graphs (since we do not have positive and negative edges). We report expansion here; denoising is in Appendix E.3.

Expansion. The goal in expansion is to predict a set of hidden edges $\mathcal{E}_M$ given $\mathbf{A}^{\mathcal{O}}$, such that the edge set of the ground truth is $\mathcal{E} = \mathcal{E}_M \cup \mathcal{E}_O$. Therefore, defining $\mathbf{A}_1 = \mathbf{A}$, the initialization becomes $\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} + (1 - \mathbf{A}^{\mathcal{O}}) \odot (f_{prior}(\mathbf{A}^{\mathcal{O}}) + \boldsymbol{\epsilon}_s)$. Throughout the blind-reconstruction experiments (expansion here and denoising in Appendix E.3), we use PIFM with the MSE loss. The results for a drop rate of 50% are shown in Table 4. Among all baselines, PIFM (GraphSAGE) attains the top AUC on all datasets and the top AP on PROTEINS and IMDB-B (remaining a close second to the GraphSAGE prior on ENZYMES AP), improving over the diffusion baselines. Compared to a Gaussian start, the informed prior is crucial to improve AUC and AP, indicating effective global coupling beyond local scores. Overall, PIFM serves as a better reconstructor in this challenging case, with K providing a tunable perception–distortion trade-off (cf. Appendix E.6).

5.4 Distortion-perception trade-off.

While a single-step reconstruction ($K = 1$) yields the lowest distortion and the highest AUC-ROC, we assess if more steps improve perceptual quality. To quantify perceptual quality, we adopt the standard graph-generation protocol of measuring the maximum mean discrepancy (MMD) between graph statistics of generated and reference graphs. This protocol was used by GraphRNN (You et al., 2018), and it remains the

Table 4: Performance for the **expansion** task with **50% of edges masked (0.5 Drop)** (see Table 9 for definitions). PIFM uses MSE loss for expansion tasks.

Mask Rate: 50% (0.5 Drop)												
	ENZYMES				**PROTEINS**				**IMDB-B**			
Method	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓
Baselines												
GraphSAGE	**13.95**	57.54	**40.29**	52.74	18.91	53.91	**31.62**	67.18	67.18	74.92	**19.04**	45.20
DiGress + RePaint	2.41	54.25	84.03	7.54	4.52	56.87	81.10	13.00	22.93	56.37	81.05	**6.69**
GDSS + RePaint	9.21	49.63	69.45	30.80	14.66	51.03	66.44	32.06	39.43	50.68	69.35	30.00
Flow w/ Gaussian prior	9.45	50.40	90.64	9.35	14.71	50.31	82.32	17.28	49.46	62.28	71.41	13.64
Ours												
PIFM (GraphSAGE)	13.17	**60.09**	100.00	0.00	**21.70**	**62.34**	94.75	**4.54**	**83.49**	**87.28**	29.74	11.27

common perception metric for both diffusion-based (Jo et al., 2022) and flow-based (Qin et al., 2025) graph generators, which report MMD distance to measure how closely generated graphs match the data distribution. Sweeping this metric as a function of K therefore traces the perception axis of our distortion–perception trade-off.

We measure the MMD^2 score between the generated and ground-truth graph distributions on the ENZYMES dataset as a function of the number of steps, K. As shown in Fig. 3(a), the MMD^2 score decreases as K increases, signifying a closer match to the true data distribution and thus higher realism. We further validate this by comparing graph statistics (degree, triangles, clustering coefficients), which also show that a larger K more closely matches the ground-truth. From the Distortion-perception trade-off perspective, decreasing MMD^2 indicates that the refinement stage moves the reconstruction closer to the clean graph distribution, and hence toward better perceptual quality. Additional results and details are in Appendices E.6 and F.1.

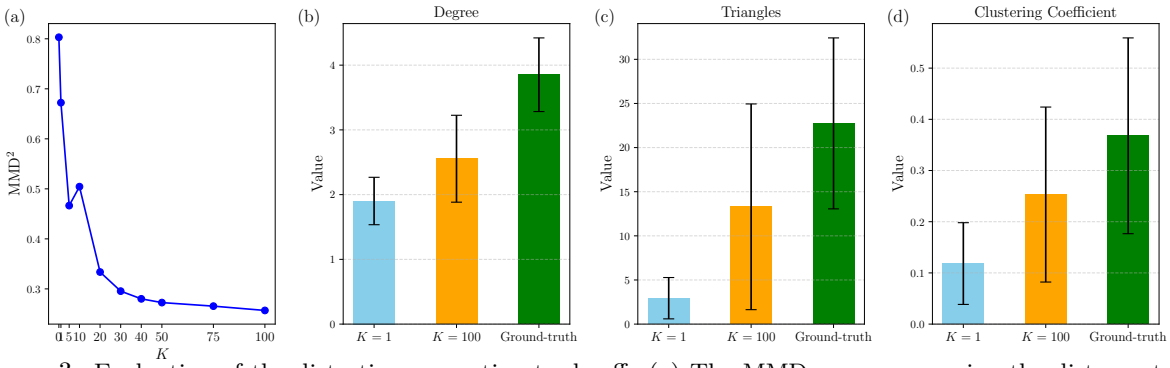


Figure 3: Evaluation of the distortion-perception trade-off. (a) The MMD score, measuring the distance to the true data distribution, decreases as K increases. (b-d) This result is corroborated by key graph statistics, where the average degree, number of triangles, and clustering coefficient for graphs generated with $K = 100$ more closely match the ground-truth distribution compared to those generated with $K = 1$. Error bars indicate the standard deviation over 300 samples (10 samples for each of the 30 test graphs).

6 Conclusions

In this paper, we introduced **Prior-Informed Flow Matching (PIFM)**, a flow-based estimator that learns global structural information by refining a local edge-prediction prior. PIFM is grounded in the distortion–perception framework: it transports a local posterior-mean estimate (obtained from GraphSAGE, node2vec, VGAE, or NCNC) toward the clean graph distribution, under either a mean squared error or cross-entropy distortion. Experiments on diverse benchmarks show that PIFM consistently outperforms both classical embedding methods and recent diffusion/flow-based baselines. The current formulation handles only homogeneous graphs; extending it to heterogeneous data via simplex-based (Eijkelboom et al., 2024) or discrete (Qin et al., 2025) flow formulations is another promising direction for future research.

References

Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Soc. Networks*, 25(3):211–230, 2003. doi: 10.1016/S0378-8733(03)00009-1.

Edo M Airoldi, Thiago B Costa, and Stanley H Chan. Stochastic blockmodel approximation of a graphon: Theory and consistent estimation. In *Advances in Neural Inf. Process. Syst. (NIPS)*, volume 26, 2013.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Michael Samuel Albergo, Mark Goldstein, Nicholas Matthew Boffi, Rajesh Ranganath, and Eric Vanden-Eijnden. Stochastic interpolants with data-dependent couplings. In *Intl. Conf. on Machine Learning (ICML)*, pp. 921–937. PMLR, 2024.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Inf. Process. Syst. (NIPS)*, volume 34, pp. 17981–17993, 2021.

Ali Azizpour, Nicolas Zilberstein, and Santiago Segarra. Scalable implicit graphon learning. In *Int. Conf. on Artif. Intell. and Stat.*, 2025.

Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 6228–6237, 2018.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *Intl. Conf. on Machine Learning (ICML)*, pp. 5453–5512. PMLR, 2024.

Stanley Chan and Edoardo Airoldi. A consistent histogram estimator for exchangeable graph models. In *Intl. Conf. on Machine Learning (ICML)*, pp. 208–216. PMLR, 2014.

Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z Kaplan, and Enrico Shippole. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. In *Intl. Conf. on Machine Learning (ICML)*, 2024.

Mauricio Delbracio and Peyman Milanfar. Inversion by direct iteration: An alternative to denoising diffusion for image restoration. *Trans. Mach. Learn. Res.*, 2024.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. Learning laplacian matrix in smooth graph signal representations. *IEEE Trans. Signal Process.*, 64(23):6160–6173, 2016.

Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *Advances in Neural Inf. Process. Syst. (NIPS)*, volume 37, pp. 11735–11764, 2024.

Dror Freirich, Tomer Michaeli, and Ron Meir. A theory of the distortion-perception tradeoff in wasserstein space. In *Advances in Neural Inf. Process. Syst. (NIPS)*, 2021.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining.*, pp. 855–864, 2016.

Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *Intl. Conf. on Machine Learning (ICML)*, pp. 2434–2444. PMLR, 2019.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Inf. Process. Syst. (NIPS)*, NIPS'17, pp. 1025–1035, 2017.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *Intl. Conf. on Machine Learning (ICML)*, pp. 10362–10383. PMLR, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Inf. Process. Syst. (NIPS)*, volume 35, pp. 26565–26577, 2022.

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

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising. *arXiv preprint arXiv:2602.16813*, 2026.

Stratis Limnios, Praveen Selvaraj, Mihai Cucuringu, Carsten Maple, Gesine Reinert, and Andrew Elliott. Sagess: Sampling graph denoising diffusion model for scalable graph generation. *arXiv preprint*, arXiv:2306.16827, 2023. doi: 10.48550/arXiv.2306.16827. URL `https://arxiv.org/abs/2306.16827`.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *Intl. Conf. Learn. Repr. (ICLR)*, 2023.

Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *Intl. Conf. Learn. Repr. (ICLR)*, 2023.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, 2022.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML Work. on Graph Rep. Learning*, 2020. URL `www.graphlearning.io`.

Mark E. J. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E*, 64(2): 025102, 2001. doi: 10.1103/PhysRevE.64.025102.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *Int. Conf. on Artif. Intell. and Stat.*, pp. 4474–4484. PMLR, 2020.

Leslie O'Bray, Max Horn, Bastian Rieck, and Karsten Borgwardt. Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions. In *Intl. Conf. Learn. Repr. (ICLR)*, 2022.

Guy Ohayon, Tomer Michaeli, and Michael Elad. Posterior-mean rectified flow: Towards minimum MSE photo-realistic image restoration. In *Intl. Conf. Learn. Repr. (ICLR)*, 2025.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining.*, pp. 701–710, 2014.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. In *Intl. Conf. on Machine Learning (ICML)*, 2025.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, Ismail Ilkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps. *arXiv preprint arXiv:2602.12233*, 2026.

Santiago Segarra, Antonio G. Marques, Gonzalo Mateos, and Alejandro Ribeiro. Network topology inference from spectral templates. *IEEE Trans. Signal and Info. Process. over Networks*, 3(3):467–483, 2017.

Kartik Sharma, Srijan Kumar, and Rakshit Trivedi. Diffuse, sample, project: plug-and-play controllable graph generation. In *Intl. Conf. on Machine Learning (ICML)*, 2024.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Inf. Process. Syst. (NIPS)*, volume 33, pp. 7462–7473, 2020.

Victor M Tenorio, Nicolas Zilberstein, Santiago Segarra, and Antonio G Marques. Graph guided diffusion: Unified guidance for conditional graph generation. *arXiv preprint arXiv:2505.19685*, 2025.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Trans. Mach. Learn. Res.*, 2024.

Puja Trivedi, Ryan A Rossi, David Arbour, Tong Yu, Franck Dernoncourt, Sungchul Kim, Nedim Lipka, Namyong Park, Nesreen K Ahmed, and Danai Koutra. Editing partially observable networks via graph diffusion models. In *Intl. Conf. on Machine Learning (ICML)*, 2024.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *Intl. Conf. Learn. Repr. (ICLR)*, 2023.

Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *Intl. Conf. Learn. Repr. (ICLR)*, 2024.

Asiri Wijesinghe, Sevvandi Kandanaarachchi, Daniel M. Steinberg, and Cheng Soon Ong. Flowette: Flow matching with graphette priors for graph generation. *arXiv preprint arXiv:2602.23566*, 2026.

Xinyue Xia, Gal Mishne, and Yusu Wang. Implicit graphon neural representation. In *Int. Conf. on Artif. Intell. and Stat.*, pp. 10619–10634. PMLR, 2023.

Hongteng Xu, Dixin Luo, Lawrence Carin, and Hongyuan Zha. Learning graphons via structured gromov-wasserstein barycenters. In *Proc. AAAI Conf. Artif. Intell.*, pp. 10505–10513, 2021.

Nianzu Yang, Huaijin Wu, Kaipeng Zeng, Yang Li, Siyuan Bao, and Junchi Yan. Molecule generation for drug design: a graph learning perspective. *Fundam. Res.*, 2024.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *Intl. Conf. on Machine Learning (ICML)*, pp. 40–48. PMLR, 2016.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning (ICML)*, pp. 5708–5717, 2018.

Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information, 2009. URL `https://arxiv.org/abs/0901.0553`. Journal version: *Eur. Phys. J. B* 71:623–630 (2009), doi:10.1140/epjb/e2009-00335-8.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: a general graph neural network framework for link prediction. In *Advances in Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Appendix

A Algorithm

In this section, we describe the algorithms that we use as baselines. Each method serves distinct purposes: the priors alone test whether the flow model provides meaningful improvement beyond the one-shot estimates given by the priors, uniform + flow evaluates whether the prior methods used (i.e. GraphSAGE/Node2Vec/VGAE/NCNC) are good structural priors for effective denoising, and DiGress + RePaint compares our model to standard modified unconditional generation models.

A.1 Uniform + flow Baseline

This baseline ablates the structural prior by initializing the flow from a state where unknown entries are filled with uniform noise. The model then learns to denoise from this less-informed starting point.

Algorithm 2 Uniform + flow Training and Sampling

Training
Sample $\mathbf{A}_1$, a mask ξ, and time $t \sim U[0,1]$.
Define initial state with a uniform fill plus a small Gaussian perturbation on the masked region: $\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \mathcal{U}(0,1)^{N\times N} + (1-\xi) \odot \boldsymbol{\epsilon}_{\text{train}}$, where $\boldsymbol{\epsilon}_{\text{train}} \sim \mathcal{N}(0, \sigma^2_{\text{train}})$.
Define interpolant $\mathbf{A}_t = (1-t)\mathbf{A}_0 + t\mathbf{A}_1$.
Solve $\theta^* = \text{argmin}_\theta \, \mathbb{E}_{\mathbf{A}_1,\xi,t} \| \, v_\theta(\mathbf{A}_t, t) - (\mathbf{A}_1 - \mathbf{A}_0) \, \|_F^2$.
Sampling (Reconstruction)
Given observed graph $\mathbf{A}^{\mathcal{O}}$, define the initial state with masked noise: $\hat{\mathbf{A}} \leftarrow \xi \odot \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \mathcal{U}(0,1)^{N\times N} + (1-\xi) \odot \boldsymbol{\epsilon}_{\text{samp}}$, where $\boldsymbol{\epsilon}_{\text{samp}} \sim \mathcal{N}(0, \sigma^2_{\text{samp}})$.
for $i \leftarrow 0$ **to** $K-1$ **do**
  $\hat{\mathbf{A}} \leftarrow \hat{\mathbf{A}} + \frac{1}{K} \, v_{\theta^*}\big(\hat{\mathbf{A}}, \frac{i}{K}\big)$
end for
$\hat{\mathbf{A}} \leftarrow \xi \odot \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \hat{\mathbf{A}}$ *// re-impose observed entries*
Return $\hat{\mathbf{A}}$

A.2 DiGress + RePaint Baseline

Training (Unconditional) The model p_θ is trained unconditionally on complete graphs $\mathbf{A}_0 \sim p_{\text{data}}$ to reverse a discrete forward noising process q. The forward process is a fixed Markov chain $q(\mathbf{A}_t|\mathbf{A}_{t-1})$ that corrupts the graph over T steps. The training objective is to learn the denoising distribution $p_\theta(\mathbf{A}_0|\mathbf{A}_t)$, modeled as a categorical prediction task for each node and edge.

Algorithm 3 DiGress Unconditional Training

Sample $t \sim \mathcal{U}\{1,\ldots,T\}$ and $\mathbf{A}_t \sim q(\mathbf{A}_t \mid \mathbf{A}_0)$. */* Forward process */*
Predict $\mathbf{A}_0$ from $(\mathbf{A}_t, t)$ with $p_\theta(\cdot, t)$. */* Denoising objective */*
Minimize expected cross-entropy:

$$\theta^* = \arg\min_\theta \; \mathbb{E}_{\mathbf{A}_0 \sim p_{\text{data}},\, t\sim\mathcal{U}\{1,\ldots,T\}} \big[\mathcal{L}_{\text{CE}}(\mathbf{A}_0,\; p_\theta(\mathbf{A}_t, t))\big].$$

Sampling (Conditional Reconstruction via RePaint) At inference, given an observed graph $\mathbf{A}^{\mathcal{O}} = \xi \odot \mathbf{A}_0$, the unconditionally trained model p_{θ^*} generates the missing entries. This is achieved by iteratively re-imposing the known (unmasked) information during the reverse diffusion process (Lugmayr et al., 2022).

Algorithm 5 Node2Vec Prior: Training and Inference

Inputs: Full adjacency $\mathbf{A}_1$, mask ξ ($\xi_{ij} = 1$ if observed), Node2Vec hyperparams (dim d, walk length L, walks/node R, window w, p, q), negatives/positive ratio k.
Outputs: Probabilities $\hat{P}$ on masked entries, i.e., $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})$.
Training (per graph)
Construct observed graph $\mathbf{A}^{\mathcal{O}} \leftarrow \xi \odot \mathbf{A}_1$.
Train Node2Vec on $\mathbf{A}^{\mathcal{O}}$ to obtain embeddings $\{\mathbf{z}_i\}_{i=1}^{N} \in \mathbb{R}^d$.
Build labeled edge set on observed pairs ($i < j$):
$\mathcal{P}^+ = \{(i,j) : \xi_{ij} = 1,\ A_{ij} = 1\}$, $\mathcal{P}^- \sim k$-to-1 balanced samples from $\{(i,j) : \xi_{ij} = 1,\ A_{ij} = 0\}$.
Features: $\mathbf{x}_{ij} \leftarrow \mathbf{z}_i \odot \mathbf{z}_j$ (Hadamard); Labels: $y_{ij} \in \{0, 1\}$.
Fit logistic classifier $g_\phi(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$ (L2; class-balanced).
Inference (per graph)
For each masked pair (i,j) with $\xi_{ij} = 0$, compute $\mathbf{x}_{ij} \leftarrow \mathbf{z}_i \odot \mathbf{z}_j$.
Predict $\hat{P}_{ij} \leftarrow g_\phi(\mathbf{x}_{ij})$ and set $\hat{P}_{ji} \leftarrow \hat{P}_{ij}$.
Return $\hat{P}$ as $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})$ (used in Eq. (7)).

Algorithm 4 DiGress + RePaint Sampling

Input: Observed graph $\mathbf{A}^{\mathcal{O}}$, mask ξ, trained model p_{θ^*}, steps T.
Output: Reconstructed graph $\hat{\mathbf{A}}_0$.
Initialize $\hat{\mathbf{A}}_T \sim p_{\text{prior}}(\cdot)$, where p_{prior} is a random graph distribution.
for $t = T, T-1, \ldots, 1$ **do**
  // Predict clean graph from current state
  $\tilde{\mathbf{A}}_0 = p_{\theta^*}(\hat{\mathbf{A}}_t, t)$.

  // Impose known data by noising it to the next step
  $\mathbf{A}_{t-1}^{\text{known}} \sim q(\mathbf{A}_{t-1} | \mathbf{A}^{\mathcal{O}})$.

  // Sample the unknown region by noising the prediction to the next step
  $\mathbf{A}_{t-1}^{\text{unknown}} \sim q(\mathbf{A}_{t-1} | \tilde{\mathbf{A}}_0)$.

  // Combine known and unknown parts for the next state
  $\hat{\mathbf{A}}_{t-1} = \xi \odot \mathbf{A}_{t-1}^{\text{known}} + (1 - \xi) \odot \mathbf{A}_{t-1}^{\text{unknown}}$.
end for
Return $\hat{\mathbf{A}}_0$

A.3 Node2Vec Prior (Per-Graph Classifier)

This baseline learns a *per-graph* edge-probability model from the observed subgraph. We (i) fit node2vec embeddings on the observed topology and (ii) train a logistic classifier on Hadamard edge features to produce probabilities on the masked pairs.

Notes. (i) We train embeddings *only* on $\mathbf{A}^{\mathcal{O}}$ to avoid leakage. (ii) The Hadamard feature works well and is symmetric; concatenation can be used but breaks symmetry unless sorted. (iii) Thresholding at 0.5 yields hard reconstructions; we use scores $\hat{P}$ directly in PIFM.

B Extended background

B.1 Graphons and graphon estimation

A *graphon*, defined as a symmetric function $\mathcal{W} : [0,1]^2 \to [0,1]$, serves as a generative model for a family of graphs:

$$\begin{aligned} z_i &\sim \text{Uniform}[0,1], \quad i = 1, \ldots, n, \\ A_{ij} &\sim \text{Bernoulli}(\mathcal{W}(z_i, z_j)), \quad 1 \le i < j \le n. \end{aligned} \tag{9}$$

Graphons provide a functional representation of exchangeable random graphs where the conditional edge probability is $[\mathbb{E}[\mathbf{A} \mid \mathbf{z}]]_{ij} = \mathcal{W}(z_i, z_j)$. This offers a natural, permutation-equivariant framework for estimating the posterior mean, though it requires access to the inverse mapping $z_i = [\mathbf{z}^{-1}(\mathbf{A}^{\mathcal{O}})]_i$. Since $\mathcal{W}$ is unknown, we estimate it using Scalable Implicit Graphon Learning (SIGL) (Azizpour et al., 2025), which combines a graph neural network (GNN) encoder with an implicit neural representation (INR). SIGL operates in three steps: (1) a GNN-based sorting step to estimate latent node positions $\mathbf{z}$; (2) a histogram approximation of the sorted adjacency matrices; and (3) learning a graphon parameterization f_ϕ by minimizing its error against the histograms. A key feature of SIGL is its ability to recover the inverse mapping $\mathbf{z}^{-1}$, making it uniquely suitable for our model (Xia et al., 2023).

The generative process in (9) can also be viewed in reverse: given a collection of graphs (represented by their adjacency matrices) $\mathcal{D} = \{\mathbf{A}_t\}_{t=1}^M$ that are sampled from an *unknown* graphon $\mathcal{W}$, estimate $\mathcal{W}$. Several methods have been proposed for this task (Chan & Airoldi, 2014; Airoldi et al., 2013; Xu et al., 2021; Xia et al., 2023; Azizpour et al., 2025). We focus on SIGL (Azizpour et al., 2025), a resolution-free method that, in addition to estimating the graphon, also *infers the latent variables* $\boldsymbol{\eta}$. This method parameterizes the graphon using an implicit neural representation (INR) (Sitzmann et al., 2020), a neural architecture defined as $f_\phi(x, y) : [0, 1]^2 \to [0, 1]$ where the inputs are coordinates from $[0, 1]^2$ and the output approximates the graphon value $\mathcal{W}$ at a particular position. In a nutshell, SIGL works in three steps: (1) a sorting step using a GNN $g_{\phi'}(\mathbf{A})$ that estimates the latent node positions or representations $\boldsymbol{\eta}$; (2) a histogram approximation of the sorted adjacency matrices; and (3) learning the parameters ϕ by minimizing the mean squared error between $f_\phi(x, y)$ and the histograms (obtained in step 2).

B.2 Node2vec

node2vec (Grover & Leskovec, 2016) is a scalable model for learning continuous node representations in graphs. This method is *transductive*, meaning that it generates an embedding for each graph. It extends the Skip-gram model from natural language processing to networks by sampling sequences of nodes through biased random walks. Node2vec introduces two hyperparameters (p, q) that interpolate between breadth-first and depth-first exploration. This flexibility allows embeddings to capture both *homophily* (nodes in the same community) and *structural equivalence* (nodes with similar roles, e.g., hubs), which frequently coexist in real-world graphs.

The embeddings are learned via stochastic gradient descent with negative sampling to maximize the likelihood of preserving sampled neighborhoods. Once learned, node embeddings can be combined through simple binary operators (e.g., Hadamard product) to form edge features, enabling applications such as link prediction. Empirically, node2vec has been shown to outperform prior unsupervised embedding methods across tasks like classification and link recovery, while remaining computationally efficient and scalable to large graphs (Grover & Leskovec, 2016).

B.3 GraphSAGE

GraphSAGE (Hamilton et al., 2017) is an *inductive* technique for link prediction based on graph neural networks (GNN) framework designed to generate embeddings for nodes in large, evolving graphs. It consists of two steps: for a target node, it first samples a fixed-size neighborhood of adjacent nodes, and then it aggregates feature information from these sampled neighbors. By learning aggregation functions (such as a mean, pool, or LSTM aggregator) rather than embeddings for every single node, GraphSAGE can efficiently generate predictions for nodes that were not part of the training set, making it highly scalable and effective for real-world applications like social networks and recommendation systems.

B.4 NCN and NCNC

Neural Common Neighbor (NCN) (Wang et al., 2024) is a link predictor that combines a message-passing encoder with a structural feature: node features $\{\mathbf{h}_i\}_{i=1}^N$ are produced by a single MPNN pass on the graph, and the score for an edge (i, j) is a learned function of the Hadamard term $\mathbf{h}_i \odot \mathbf{h}_j$ and a pooled sum over common-neighbor representations $\sum_{u \in N(i) \cap N(j)} \mathbf{h}_u$. Because the MPNN runs only once per graph rather

than once per target link, NCN is significantly more scalable than methods that apply MPNN to a subgraph for each candidate edge, while remaining expressive enough to distinguish links by the identity of their common neighbors.

NCN with Completion (NCNC) extends NCN to address graph incompleteness: when the observed graph is partial, true common neighbors may be unobserved, distorting the structural feature on which NCN relies. NCNC softly completes the common-neighbor set by extending the pooled sum to all candidate intermediaries $u \in N(i) \cup N(j)$ and weighting each by a learned link-existence probability (1 if u is already an observed common neighbor; the predicted probability of the unobserved endpoint-to-u edge if exactly one endpoint connects to u; and 0 otherwise). In our work we use the depth-1 NCNC predictor (a single round of soft completion before re-applying NCN), corresponding to the `IncompleteCN1Predictor` module of the official implementation.

B.5 Graph Diffusion Models

Diffusion models are generative frameworks composed of two processes: a **forward process** that systematically adds noise to data until it becomes pure noise, and a **reverse process** that learns to reverse this, generating new data by starting from noise and progressively denoising it. While these models exist for both discrete (Vignac et al., 2023) and continuous domains (Jo et al., 2022), we describe the continuous case which is the most related to our method. Here, a graph $\mathbf{G}_0$ is defined by its node features $\mathbf{X}_0 \in \mathbb{R}^{N \times F}$ and its weighted adjacency matrix $\mathbf{A}_0 \in \mathbb{R}^{N \times N}$. Following the GDSS framework, the forward process is described by a stochastic differential equation (SDE) that gradually perturbs the graph data over a time interval $t \in [0, T]$:

$$\mathrm{d}\mathbf{G}_t = -\frac{1}{2}\beta(t)\mathbf{G}_t \,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{W}_t$$

In this equation, $\mathbf{W}_t$ represents standard Brownian motion (i.e., noise), and $\beta(t)$ is a noise schedule that typically increases over time. This process is designed so that by the final time T, the noised graph $\mathbf{G}_T$ is indistinguishable from a standard Gaussian.

The generative reverse process is defined by another SDE that traces the path from noise back to data. This process relies on the **score function**, $\nabla_{\mathbf{G}_t} \log p(\mathbf{G}_t)$, which is the gradient of the log-density of the noisy data at time t. Since the true score function is unknown, it must be approximated. This is done using a neural network, or **score network**, which is trained to predict the score. For graphs, separate networks are often used for the adjacency matrix and node features: $\boldsymbol{\epsilon}_{\boldsymbol{\theta}_A}(\mathbf{A}_t, t)$ and $\boldsymbol{\epsilon}_{\boldsymbol{\theta}_X}(\mathbf{X}_t, t)$. These networks are trained by minimizing the denoising score-matching loss.

Once trained, these score networks can be plugged into the reverse SDE. New graphs are then generated by solving this SDE numerically using standard samplers like DDPM or DDIM.

B.5.1 Diffusion-based inverse problems solver for graphs

We now expand on diffusion-based solvers for graph inverse problems. Given a condition $\mathcal{C}$ and a reward function $r(\mathbf{G}_0)$ that quantifies how close the sample $\mathbf{G}_0$ is to meeting $\mathcal{C}$, the objective is to generate graphs G_0 that maximize the reward function. From a Bayesian perspective, this problem boils down to sampling from the posterior $p(\mathbf{G}_0|\mathcal{C}) \propto p(\mathcal{C}|\mathbf{G}_0)p(\mathbf{G}_0)$ where $p(\mathcal{C}|\mathbf{G}_0) \propto \exp(r(\mathbf{G}_0))$ is a likelihood term and $p(\mathbf{G}_0)$ is a prior given by the pre-trained diffusion model. We now describe previous works for both differentiable and non-differentiable reward functions.

Guidance with Differentiable Reward Functions. Several approaches have been developed to guide generative models when the objective can be expressed as a **differentiable reward function**, particularly for inverse problems in imaging. These methods typically leverage the differentiability of the reward – often a likelihood tied to a noisy measurement – to calculate a *conditional score* using Bayes' rule:

$$\nabla_{\mathbf{G}_t} \log p(\mathbf{G}_t|\mathcal{C}) = \nabla_{\mathbf{G}_t} \log p(\mathcal{C}|\mathbf{G}_t) + \nabla_{\mathbf{G}_t} \log p(\mathbf{G}_t)$$

In this formulation, the diffusion model naturally serves as the prior ($p(\mathbf{G}_t)$), while the likelihood term ($p(\mathcal{C}|\mathbf{G}_t)$) provides the guidance. However, directly computing the score of the likelihood term is intractable because it requires integrating over all possible clean data: $p(\mathcal{C}|\mathbf{G}_t) = \int p(\mathcal{C}|\mathbf{G}_0)p(\mathbf{G}_0|\mathbf{G}_t)d\mathbf{G}_0$.

To overcome this, a common technique is to approximate the posterior distribution $p(\mathbf{G}_0|\mathbf{G}_t)$ with a Gaussian centered at the MMSE denoiser. This denoised estimate can be calculated efficiently using **Tweedie's formula**:

$$\mathbb{E}[\mathbf{G}_0|\mathbf{G}_t] = \frac{1}{\alpha_t}\left(\mathbf{G}_t + \sigma_t^2 \nabla_{\mathbf{G}_t} \log p(\mathbf{G}_t, t)\right)$$

While this framework is established for images, its application to graph-based inverse problems is less explored. This is primarily because most interesting properties and constraints in graphs are **not differentiable**. Some graph-specific methods, like DiGress (Vignac et al., 2023), implement guidance by training an auxiliary model, similar to classifier-free guidance, which introduces additional complexity.

Guidance with Non-Differentiable Reward Functions. For the more common scenario of non-differentiable constraints in graph generation, alternative strategies have emerged. The **PRODIGY** method, for instance, operates by repeatedly applying a two-step process at each denoising step: generation followed by projection.

First, it uses the unconditional diffusion model to produce a candidate sample $\hat{\mathbf{G}}_{t-1}$. Second, it projects this candidate onto the set of valid solutions using a projection operator: $\Pi_{\mathcal{C}}(\hat{\mathbf{G}}_{t-1}) = \arg\min_{\mathbf{Z}\in\mathcal{C}} \|\mathbf{Z} - \hat{\mathbf{G}}_{t-1}\|_2^2$. Since applying the full projection at every step can destabilize the generation process, PRODIGY uses a partial update to balance constraint satisfaction with the learned diffusion trajectory:

$$\mathbf{G}_{t-1} \leftarrow (1-\gamma_t)\hat{\mathbf{G}}_{t-1} + \gamma_t \Pi_{\mathcal{C}}(\hat{\mathbf{G}}_{t-1})$$

This approach has two main limitations. First, it is only practical for simple constraints where the projection operator $\Pi_{\mathcal{C}}(\cdot)$ has an efficient, closed-form solution. Second, it applies the projection directly to the noisy intermediate sample $\mathbf{G}_{t-1}$, whereas the constraint $\mathcal{C}$ is defined on the clean data $\mathbf{G}_0$, creating a domain mismatch. Recently, in Tenorio et al. (2025), the authors use zeroth-order optimization to build a guidance term, improving over PRODIGY in challenging tasks.

B.5.2 Flow-based inverse solvers

More recently, two flow-based generative models for graphs have been proposed. Catflow, introduced in Eijkelboom et al. (2024), formulates flow matching as a variational inference problem, allowing to build a model for categorical data. The key difference between Catflow and traditional flow matching is that in the former, the objective is to approximate the posterior probability path, which is a distribution over possible end points of a trajectory. Compared to discrete diffusion, this formulation defines a path in the probability simplex, building a continuous path. This formulation boils down to a cross-entropy loss. Another recent work is DeFoG, introduced in Qin et al. (2025). This method is inspired by discrete flow matching (Campbell et al., 2024), where a discrete probability path is used. Similarly, the loss is a cross-entropy.

C Proofs

C.1 Proof of Theorem 1 and Proposition 1

Proposition 1 (Permutation invariance of the reconstruction density)**.** *In addition to (a)–(b), assume:*

(b′) v_θ is continuously differentiable in its first argument (a sharpening of (b) needed for the change-of-variables Jacobian);

(c) f_{prior} and $v_\theta(\cdot, t)$ map symmetric, zero-diagonal matrices to symmetric, zero-diagonal matrices, so the dynamics stay in the space of undirected graphs;

(d) the source noise is non-degenerate, $\sigma_s > 0$.

The mask $\xi \in \{0,1\}^{N\times N}$ is symmetric with zero diagonal (it selects unordered edges), which is what lets $w_\theta = (1-\xi)\odot v_\theta$ preserve the symmetric, zero-diagonal subspace. We identify an undirected graph with its edge vector over the unordered pairs $\{i,j\}$, $i<j$; the masked edge coordinates are the pairs with $\xi_{ij}=0$, of which there are m. Define the projected *velocity field*

$$w_\theta(\mathbf{A}, t; \xi) = (1-\xi)\odot v_\theta(\mathbf{A}, t),$$

which vanishes on observed coordinates by construction and is the idealized object the trained MSE field approximates. Define

$$\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} + (1-\xi)\odot\big(f_{prior}(\mathbf{A}^{\mathcal{O}}) + \boldsymbol{\epsilon}_s\big), \quad \boldsymbol{\epsilon}_s \sim \mathcal{N}(0, \sigma_s^2 I_m) \text{ on masked edges,}$$
$$\mathbf{A}_t \text{ solves the projected flow } \dot{\mathbf{A}}_t = w_\theta(\mathbf{A}_t, t; \xi),\ \mathbf{A}_t|_{t=0} = \mathbf{A}_0 \text{ (assumed to exist on } [0,1]), \qquad \tilde{\mathbf{A}}_1 = \mathbf{A}_{t=1}.$$

Then the conditional density of $\tilde{\mathbf{A}}_1$ on the masked edge coordinates, given by the CNF change-of-variables formula

$$\log p(\tilde{\mathbf{A}}_1 \mid \mathbf{A}^{\mathcal{O}}, \xi) = \log p(\mathbf{A}_0 \mid \mathbf{A}^{\mathcal{O}}, \xi) - \int_0^1 \mathrm{tr}_m\left(\frac{\partial w_\theta(\mathbf{A}_t, t; \xi)}{\partial \mathbf{A}_t}\right) dt, \tag{10}$$

with tr_m the trace restricted to masked edge coordinates, is invariant under simultaneous relabeling: for every permutation matrix $\mathbf{P}$,

$$\log p(\mathbf{P}^\top \tilde{\mathbf{A}}_1 \mathbf{P} \mid \mathbf{P}^\top \mathbf{A}^{\mathcal{O}} \mathbf{P}, \mathbf{P}^\top \xi \mathbf{P}) = \log p(\tilde{\mathbf{A}}_1 \mid \mathbf{A}^{\mathcal{O}}, \xi). \tag{11}$$

Proof. Throughout, write $\mathbf{A}'^{\mathcal{O}} = \mathbf{P}^\top \mathbf{A}^{\mathcal{O}} \mathbf{P}$, $\xi' = \mathbf{P}^\top \xi \mathbf{P}$, and let

$$\mu(\mathbf{X}, \eta) \ = \ \mathbf{X} + (1-\eta)\odot f_{\text{prior}}(\mathbf{X})$$

denote the deterministic center of the source state. For the density statement we work in undirected-edge coordinates: a symmetric, zero-diagonal matrix $\mathbf{X}$ is identified with its edge vector $\mathbf{e}(\mathbf{X}) \in \mathbb{R}^{N(N-1)/2}$ over unordered pairs $\{i, j\}$, simultaneous relabeling $\mathbf{X} \mapsto \mathbf{P}^\top \mathbf{X} \mathbf{P}$ acts on $\mathbf{e}(\mathbf{X})$ by an orthogonal permutation matrix $R_{\mathbf{P}}$ that maps masked edges to masked edges, $S_\xi \in \{0,1\}^{m \times N(N-1)/2}$ extracts the m masked edge coordinates, and $\mathbf{M}_\xi = S_\xi^\top S_\xi$ is the corresponding diagonal projector. Assumption (c) makes w_θ tangent to this subspace, so the projected flow remains in edge coordinates. We prove both results together: Steps (i) and (iii), together with the equivariance of the discrete sampler, establish the equivariance of the reconstruction (Theorem 1); Steps (ii) and (iv), combined with the change-of-variables conclusion, establish the density invariance (Proposition 1). The four steps are (i) equivariance of μ; (ii) invariance of the conditional source density under simultaneous relabeling; (iii) equivariance of the ODE flow map; (iv) pointwise invariance of the masked-trace integrand.

Step (i): equivariance of the deterministic center μ. Using equivariance of f_{prior} (assumption (a)) together with the identity $(\mathbf{P}^\top X \mathbf{P}) \odot (\mathbf{P}^\top Y \mathbf{P}) = \mathbf{P}^\top (X \odot Y) \mathbf{P}$:

$$\begin{aligned}
\mu(\mathbf{A}'^{\mathcal{O}}, \xi') &= \mathbf{A}'^{\mathcal{O}} + (1-\xi')\odot f_{\text{prior}}(\mathbf{A}'^{\mathcal{O}}) \\
&= \mathbf{P}^\top \mathbf{A}^{\mathcal{O}} \mathbf{P} + (1 - \mathbf{P}^\top \xi \mathbf{P}) \odot f_{\text{prior}}(\mathbf{P}^\top \mathbf{A}^{\mathcal{O}} \mathbf{P}) \\
&= \mathbf{P}^\top \mathbf{A}^{\mathcal{O}} \mathbf{P} + \mathbf{P}^\top (1-\xi)\mathbf{P} \odot \mathbf{P}^\top f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) \mathbf{P} \\
&= \mathbf{P}^\top \big[\mathbf{A}^{\mathcal{O}} + (1-\xi)\odot f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})\big] \mathbf{P} \\
&= \mathbf{P}^\top \mu(\mathbf{A}^{\mathcal{O}}, \xi) \mathbf{P}.
\end{aligned}$$

Step (ii): invariance of the conditional source density. In the masked edge coordinates the source is $\mathcal{N}\big(\mathbf{e}_\xi(\mu(\mathbf{A}^{\mathcal{O}}, \xi)),\ \sigma_s^2 I_m\big)$ on $\mathbb{R}^m$, where $\mathbf{e}_\xi(\mathbf{X}) = S_\xi\, \mathbf{e}(\mathbf{X}) \in \mathbb{R}^m$ extracts the masked edges (each edge counted once), so that

$$p(\mathbf{A}_0 \mid \mathbf{A}^{\mathcal{O}}, \xi) \ \propto\ \exp\Big(-\tfrac{1}{2\sigma_s^2}\big\|\mathbf{e}_\xi\big(\mathbf{A}_0 - \mu(\mathbf{A}^{\mathcal{O}}, \xi)\big)\big\|_2^2\Big).$$

Evaluating at $\mathbf{P}^\top \mathbf{A}_0 \mathbf{P}$ under the permuted inputs $(\mathbf{A}'^{\mathcal{O}}, \xi')$ and using Step (i): since relabeling maps masked edges to masked edges, $\mathbf{e}_{\xi'}(\mathbf{P}^\top \mathbf{X} \mathbf{P})$ is a permutation of the m entries of $\mathbf{e}_\xi(\mathbf{X})$, so its ℓ_2 norm is unchanged,

$$\big\|\mathbf{e}_{\xi'}\big(\mathbf{P}^\top \mathbf{A}_0 \mathbf{P} - \mu(\mathbf{A}'^{\mathcal{O}}, \xi')\big)\big\|_2^2 = \big\|\mathbf{e}_\xi\big(\mathbf{A}_0 - \mu(\mathbf{A}^{\mathcal{O}}, \xi)\big)\big\|_2^2.$$

Hence

$$p(\mathbf{P}^\top \mathbf{A}_0 \mathbf{P} \mid \mathbf{A}'^{\mathcal{O}}, \xi') \ = \ p(\mathbf{A}_0 \mid \mathbf{A}^{\mathcal{O}}, \xi).$$

Step (iii): equivariance of the ODE flow map. Let g_ξ be a velocity field that is locally Lipschitz in $\mathbf{A}$ and satisfies the equivariance identity $\mathbf{P}^\top g_\xi(\mathbf{A}, t)\mathbf{P} = g_{\xi'}(\mathbf{P}^\top \mathbf{A}\mathbf{P}, t)$. This covers $g_\xi = v_\theta$ (the implemented continuous sampler, which does not depend on ξ, so $g_{\xi'} = v_\theta$) and $g_\xi = w_\theta(\cdot, \cdot; \xi)$ (the projected flow of Proposition 1, for which the identity follows from equivariance of v_θ and $1 - \xi' = \mathbf{P}^\top(1 - \xi)\mathbf{P}$). Let $\Phi_t^\xi(\mathbf{A}_0)$ solve $\dot{\mathbf{A}}_t = g_\xi(\mathbf{A}_t, t)$ from $\mathbf{A}_0$, and set $\tilde{\mathbf{A}}_t := \mathbf{P}^\top \Phi_t^\xi(\mathbf{A}_0)\mathbf{P}$. Differentiating in t and using the equivariance identity,

$$\dot{\tilde{\mathbf{A}}}_t \;=\; \mathbf{P}^\top g_\xi(\Phi_t^\xi(\mathbf{A}_0), t)\mathbf{P} \;=\; g_{\xi'}(\mathbf{P}^\top \Phi_t^\xi(\mathbf{A}_0)\mathbf{P}, t) \;=\; g_{\xi'}(\tilde{\mathbf{A}}_t, t),$$

with initial condition $\tilde{\mathbf{A}}_0 = \mathbf{P}^\top \mathbf{A}_0 \mathbf{P}$. By Picard-Lindelöf (assumption (b)), the ξ'-trajectory starting at $\mathbf{P}^\top \mathbf{A}_0 \mathbf{P}$ is exactly $\tilde{\mathbf{A}}_t$, so

$$\Phi_t^{\xi'}(\mathbf{P}^\top \mathbf{A}_0 \mathbf{P}) \;=\; \mathbf{P}^\top \Phi_t^\xi(\mathbf{A}_0)\mathbf{P} \qquad \text{for all } t \in [0, 1].$$

In particular, the inverse flow satisfies $(\Phi_1^{\xi'})^{-1}(\mathbf{P}^\top \tilde{\mathbf{A}}_1 \mathbf{P}) = \mathbf{P}^\top \mathbf{A}_0 \mathbf{P}$.

Equivariance of the reconstruction (Theorem 1). Step (i) gives equivariance of the deterministic center μ; since $\boldsymbol{\epsilon}_s$ is isotropic Gaussian, $\mathbf{P}^\top \boldsymbol{\epsilon}_s \mathbf{P} \overset{d}{=} \boldsymbol{\epsilon}_s$, so the source state obeys $\mathbf{A}_0(\mathbf{A}'^{\mathcal{O}}, \xi') \overset{d}{=} \mathbf{P}^\top \mathbf{A}_0(\mathbf{A}^{\mathcal{O}}, \xi)\mathbf{P}$ (pointwise under the coupled draw $\mathbf{P}^\top \boldsymbol{\epsilon}_s \mathbf{P}$, or when $\sigma_s = 0$). Step (iii) gives equivariance of the continuous flow map. The remaining operations of the discrete sampler in Alg. 1 act entrywise with shared parameters (the Euler increment, the sigmoid $\sigma(\cdot)$, the clip to $[0, 1]$, and the re-imposition $\xi \odot \mathbf{A}^{\mathcal{O}} + (1 - \xi) \odot \hat{\mathbf{A}}$ of the CE branch), so each commutes with the relabeling $X \mapsto \mathbf{P}^\top X \mathbf{P}$. Composing an equivariant source with the equivariant K-step sampler yields $\mathrm{PIFM}(\mathbf{A}'^{\mathcal{O}}, \xi') \overset{d}{=} \mathbf{P}^\top \mathrm{PIFM}(\mathbf{A}^{\mathcal{O}}, \xi)\mathbf{P}$, proving Theorem 1. This uses only assumptions (a)–(b) and holds for both distortions and every K.

Step (iv): pointwise invariance of the masked-trace integrand. Work in the edge coordinates of the preamble, on which simultaneous relabeling acts by the orthogonal permutation $R_{\mathbf{P}}$ (so $\mathbf{e}(\mathbf{A}'_t) = R_{\mathbf{P}}\, \mathbf{e}(\mathbf{A}_t)$). Let $J_\xi(\mathbf{A}_t)$ be the edge-coordinate Jacobian of the projected field $w_\theta(\cdot, t; \xi)$. Equivariance of w_θ and chain-rule differentiation give

$$J_{\xi'}(\mathbf{A}'_t) \;=\; R_{\mathbf{P}}\, J_\xi(\mathbf{A}_t)\, R_{\mathbf{P}}^\top.$$

Under simultaneous relabeling the masked-edge set maps to itself, so $\mathbf{M}_{\xi'} = R_{\mathbf{P}}\, \mathbf{M}_\xi\, R_{\mathbf{P}}^\top$. By cyclic trace invariance and $R_{\mathbf{P}}^\top R_{\mathbf{P}} = \mathbf{I}$,

$$\mathrm{tr}_m\Big(\tfrac{\partial w_\theta(\mathbf{A}'_t, t; \xi')}{\partial \mathbf{A}'_t}\Big) = \mathrm{tr}(\mathbf{M}_{\xi'} J_{\xi'}(\mathbf{A}'_t)) = \mathrm{tr}(R_{\mathbf{P}}\, \mathbf{M}_\xi\, J_\xi(\mathbf{A}_t)\, R_{\mathbf{P}}^\top) = \mathrm{tr}(\mathbf{M}_\xi J_\xi(\mathbf{A}_t)) = \mathrm{tr}_m\Big(\tfrac{\partial w_\theta(\mathbf{A}_t, t; \xi)}{\partial \mathbf{A}_t}\Big).$$

The integrand is pointwise invariant, hence so is its integral over t.

Conclusion (density invariance, Proposition 1). Applying the CNF formula at the permuted endpoint $\mathbf{P}^\top \tilde{\mathbf{A}}_1 \mathbf{P}$, and chaining Steps (ii)–(iv):

$$\begin{aligned}
\log p(\mathbf{P}^\top \tilde{\mathbf{A}}_1 \mathbf{P} \mid \mathbf{A}'^{\mathcal{O}}, \xi') &= \log p\big((\Phi_1^{\xi'})^{-1}(\mathbf{P}^\top \tilde{\mathbf{A}}_1 \mathbf{P}) \,\big|\, \mathbf{A}'^{\mathcal{O}}, \xi'\big) - \int_0^1 \mathrm{tr}_m\Big(\tfrac{\partial w_\theta(\mathbf{A}'_t, t; \xi')}{\partial \mathbf{A}'_t}\Big)\, dt \\
&\overset{\text{(iii)}}{=} \log p(\mathbf{P}^\top \mathbf{A}_0 \mathbf{P} \mid \mathbf{A}'^{\mathcal{O}}, \xi') - \int_0^1 \mathrm{tr}_m\Big(\tfrac{\partial w_\theta(\mathbf{A}'_t, t; \xi')}{\partial \mathbf{A}'_t}\Big)\, dt \\
&\overset{\text{(ii)}}{=} \log p(\mathbf{A}_0 \mid \mathbf{A}^{\mathcal{O}}, \xi) - \int_0^1 \mathrm{tr}_m\Big(\tfrac{\partial w_\theta(\mathbf{A}'_t, t; \xi')}{\partial \mathbf{A}'_t}\Big)\, dt \\
&\overset{\text{(iv)}}{=} \log p(\mathbf{A}_0 \mid \mathbf{A}^{\mathcal{O}}, \xi) - \int_0^1 \mathrm{tr}_m\Big(\tfrac{\partial w_\theta(\mathbf{A}_t, t; \xi)}{\partial \mathbf{A}_t}\Big)\, dt \\
&= \log p(\tilde{\mathbf{A}}_1 \mid \mathbf{A}^{\mathcal{O}}, \xi).
\end{aligned}$$

□

Remark (which priors satisfy assumption (a)). SIGL, GraphSAGE, VGAE, and NCNC are permutation-equivariant by construction: each is built from equivariant GNN layers, and the edge-prediction head reduces to a symmetric function of node embeddings (Hadamard product, inner product, or common-neighbor pooling). When inference is stochastic (VGAE latent sampling, GraphSAGE neighborhood sampling), this equivariance holds in distribution, under the coupled randomness obtained by relabeling the sampling seeds together with the nodes; it holds pointwise only when inference is fully deterministic (e.g., full-neighborhood aggregation with a deterministic readout and no neighborhood sampling), since a deterministic final readout alone does not remove randomness from the encoder. Node2vec is not strictly equivariant: it produces stochastic embeddings via SGD, and the PCA-based canonicalization we use to fix the embedding ordering relies on conventions (sign-fixing, tie-breaking under repeated eigenvalues) that are not in general permutation-invariant. For Node2vec, Theorem 1 and Proposition 1 should therefore be read as approximate guarantees that hold when the canonicalization is itself approximately invariant; the strictly equivariant priors give the unconditional guarantee.

D Experimental details

D.1 Details about the architecture

Our model adopts a modified version of the adjacency score network architecture introduced in GDSS (Jo et al., 2022). The network is a permutation-equivariant graph neural network designed to approximate the scores $\nabla_{\mathbf{A}_t} \log p_t(X_t, \mathbf{A}_t)$ and $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t, \mathbf{A}_t)$ at each diffusion step; in this paper, we use only score w.r.t. $\mathbf{A}_t$. Concretely, the architecture consists of stacked message-passing layers followed by a multi-layer perceptron. Each layer propagates node and edge information through adjacency-based aggregation, ensuring equivariance under node relabeling. Time information t is incorporated by scaling intermediate activations with the variance of the forward diffusion process, following the practice in continuous-time score models. Residual connections and normalization layers are used to stabilize training. The final output is an $N \times N$ tensor matching the adjacency dimension. This design provides the required permutation-equivariance and expressive power while remaining computationally tractable for mid-sized benchmark graphs.

The modification that incorporates is a module to build an embedding for the variable t and a FiLM style modulation to incorporate noise conditioning. In particular, we incorporate the following modules:

- A positional encoding based on a sinusoidal embedding following Karras et al. (2022)
- An MLP layer with SiLU activation per attention layer
- A modulation at each attention layer, where we scale the hidden features by an adaptive RMS norm operation (Crowson et al., 2024)

D.2 Details about the datasets

In Table 5 we report the statistics of the datasets used in the main text.

Table 5: Statistics of the datasets used for evaluation.

Dataset	# Graphs	Avg. Nodes	Avg. Edges	# Classes	Domain
ENZYMES	600	32.63	62.14	6	Bioinformatics
PROTEINS	1,113	39.06	72.82	2	Bioinformatics
IMDB-B	1,000	19.77	96.53	2	Social Network

D.3 Hyperparameters

D.3.1 Flow-based baselines

We report the hyperparameters of our model, both the architecture and training. All three baselines use the same rectified-flow architecture and optimizer family; the only substantive differences are the prior settings.

PIFM - Link Prediction, 10% masked	
`batch_size`	64 (IMDB-B & ENZYMES), 32 (PROTEINS)
`optimizer`	Adam
`learning_rate`	2e-4
`dropout`	0.2
`hidden_dim`	32
`num_layers`	5
`num_linears`	2
`channels`	{c_init: 2, c_hid: 8, c_final: 4}
`train/val/test_noise_std`	0.05 (IMDB-B & PROTEINS), 0.1 (ENZYMES)
`ode_steps` (Euler, K)	1 to 100
`prior`	GraphSAGE (default hyperparameters)

PIFM - Link Prediction, 50% masked	
`batch_size`	64 (IMDB-B & ENZYMES), 32 (PROTEINS)
`optimizer`	Adam
`learning_rate`	2e-4
`dropout`	0.2
`hidden_dim`	32
`num_layers`	5
`num_linears`	2
`channels`	{c_init: 2, c_hid: 8, c_final: 4}
`train/val/test_noise_std`	0.05 (IMDB-B & PROTEINS), 0.1 (ENZYMES)
`ode_steps` (Euler, K)	1 to 100
`prior`	GraphSAGE (default hyperparameters)

PIFM (VGAE) - Link Prediction, 50% masked	
`batch_size`	64 (IMDB-B & ENZYMES), 32 (PROTEINS)
`optimizer`	Adam
`learning_rate`	2e-4
`dropout`	0.2
`hidden_dim`	32
`num_layers`	5
`num_linears`	2
`channels`	{c_init: 2, c_hid: 8, c_final: 4}
`train/val/test_noise_std`	0.01 (IMDB-B), 0.00 (PROTEINS, ENZYMES)
`ode_steps` (Euler, K)	1 to 100
`prior`	VGAE (epochs: 200, lr: 0.01, hidden: 32/16)

PIFM (NCNC) - Link Prediction, 50% masked	
`batch_size`	64 (IMDB-B & ENZYMES), 32 (PROTEINS)
`optimizer`	Adam
`learning_rate`	2e-4
`dropout`	0.2
`hidden_dim`	32
`num_layers`	5
`num_linears`	2
`channels`	{c_init: 2, c_hid: 8, c_final: 4}
`train/val/test_noise_std`	0.05 (IMDB-B), 0.01 (ENZYMES, PROTEINS)
`ode_steps` (Euler, K)	1 to 100
`prior`	NCNC (default hyperparameters)

Flow w/ Gaussian Prior	**Value**
`epochs`	1000
`batch_size`	64 (default), 32 (PROTEINS)
`optimizer`	Adam
`learning_rate`	2e-4
`dropout`	0.2
`hidden_dim`	32
`num_layers`	5
`num_linears`	2
`channels`	{c_init: 2, c_hid: 8, c_final: 4}
`train_noise_std` (masked t=0)	0.00
`val_noise_std` (masked t=0)	0.00
`ode_steps` (Euler, K)	1 to 100
`prior`	None (masked entries initialized from $\mathcal{N}(0.5, 1)$)

D.4 Metrics Calculation

We evaluate performance only on the set of masked (unknown) edges in the upper triangle of the adjacency matrix. For each test graph, all metrics are computed on these entries and then averaged across graphs.

Metrics Used in Tables We report the following four metrics in the main results:

- **Area Under the ROC Curve (AUC).** Computed on the raw predicted scores (when available). AUC measures the probability that a randomly chosen true edge receives a higher predicted score than a randomly chosen non-edge. Larger AUC indicates stronger ranking performance.
- **Average Precision (AP).** Computed from the precision–recall curve induced by ranking the predictions. AP summarizes how well the model recovers true edges across all possible thresholds, with higher values indicating better precision–recall trade-offs.
- **False Positive Rate (FPR).** After thresholding predictions at 0.5, the FPR is defined as

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}},$$

- **False Negative Rate (FNR).** After thresholding predictions at 0.5, the FNR is defined as

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}},$$

- **MMD.** A kernel-based method that measures the difference between two probability distributions by embedding them in a feature space and finding the maximum difference between their mean embeddings.

AUC and AP are threshold-independent metrics (computed directly on the provided scores), while FPR and FNR are threshold-dependent error rates (obtained after binarizing at 0.5). All values reported in the tables are averaged over test graphs and expressed in percentage.

D.5 Training cost

All models are trained on NVIDIA A100-SXM4-80GB GPUs. Once the prior is fixed, the flow model used in PIFM has essentially the same training loop and computational cost as the flow model initialized with a Gaussian prior. Thus, the main additional cost of PIFM comes from fitting the prior itself, not from the subsequent flow-matching stage. This observation is consistent with our empirical training runs, where we did not observe noticeable differences in optimization behavior between Gaussian-prior and prior-informed flow variants across edge-drop settings. Table 6 reports training time of the flow model for 100 epochs on each dataset, with PROTEINS taking the most time while the other 2 datasets remain similar training costs.

Table 6: Training time of the flow model for 100 epochs on each dataset under 50% mask rate.

Dataset	**Training Time (s)**
ENZYMES	206.4
PROTEINS	646.7
IMDB-B	243.7

D.6 Inference cost

Since PIFM reconstructs graphs by integrating a learned velocity field over K Euler steps, its runtime grows approximately linearly with the number of Euler steps integrated. We observe a cost of about 0.03 seconds per step per graph, so increasing K improves refinement quality at the expense of proportionally higher latency.

Table 7 reports the per-graph inference time of PIFM as a function of the number of Euler steps. The results show near-linear scaling, from approximately 0.03 seconds in the lowest-step setting to about 2.8 seconds when using 100 steps.

Table 7: Per-graph inference time of PIFM as a function of the number of Euler steps.

Setting	**Inference Time per Graph (s)**
$K = 1$	0.03
$K = 10$	0.28
$K = 50$	1.40
$K = 100$	2.80

To contextualize these costs, Table 8 compares PIFM against the standalone prior predictors on each dataset. As expected, the one-shot priors are faster because they do not require iterative refinement. However, PIFM with $K = 1$ remains relatively inexpensive while retaining smallest reconstruction error, with per-graph inference times below 0.06 seconds on all three benchmarks. This makes the single-step setting attractive in regimes where distortion is the main objective, while larger values of K are better suited to settings where perceptual refinement is more important.

Table 8: Per-graph inference time comparison between standalone priors and PIFM.

Method	**PROTEINS (s)**	**ENZYMES (s)**	**IMDB-B (s)**
Node2Vec Prior	0.0016	0.0020	0.0010
GraphSAGE Prior	0.0003	0.0003	0.0002
PIFM ($K = 1$)	0.0425	0.0532	0.0440
PIFM ($K = 100$)	2.8207	2.8419	2.8660

E Additional results

E.1 Link prediction with 10 % of edges masked

Table 9: Graph reconstruction performance with **10% of edges masked (0.1 Drop)** (see Table 2 for definitions).

Mask Rate: 10% (0.1 Drop)												
	ENZYMES				PROTEINS				IMDB-B			
Method	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓
Baselines												
Node2Vec	24.62	59.60	51.39	37.96	33.24	64.40	48.56	35.37	65.00	56.36	50.27	41.68
GraphSAGE	41.28	73.70	**13.49**	60.59	46.36	74.58	**11.00**	63.50	83.55	83.26	16.42	36.89
DiGress + RePaint	33.39	67.86	58.92	5.19	40.34	72.39	47.82	6.00	59.25	58.63	76.44	7.68
GDSS + RePaint	18.35	47.04	74.31	32.19	26.96	51.39	63.07	32.09	57.89	46.11	69.75	36.17
Flow w/ Gaussian prior	40.09	72.44	71.03	5.87	57.86	80.83	65.09	3.07	98.89	98.37	2.26	**2.54**
Ours												
PIFM (Node2Vec)	41.67	76.86	72.09	5.11	**58.25**	**81.74**	59.37	6.34	97.60	97.28	**1.37**	3.77
PIFM (GraphSAGE)	**47.21**	**80.25**	72.85	**2.40**	54.79	81.02	55.73	5.40	**99.37**	**98.79**	1.81	3.37

E.2 PIFM with a graphon-based prior (SIGL)

The priors used in the main text (GraphSAGE, node2vec, VGAE, NCNC) are all embedding-based link predictors. To test whether PIFM is agnostic to the *family* of structural prior, we additionally evaluate a graphon-based prior: Scalable Implicit Graphon Learning (SIGL) (Azizpour et al., 2025), which fits a graphon to the training graphs and predicts masked edges from the learned latent node coordinates (background in Appendix B.1). Unlike the main-text priors, SIGL is a generative model of an exchangeable graph family rather than a discriminative edge predictor, making it a useful test on PIFM's prior-agnosticism.

Table 10: PIFM with a graphon-based prior (SIGL) on link prediction, at 10% and 50% edge drop. AP, AUC, FNR, FPR in % (see Table 2 for definitions). **Bold** marks the better of the two rows per ranking metric (AP, AUC) within each block.

	ENZYMES				PROTEINS				IMDB-B			
Method	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓
Mask rate: 10% (0.1 Drop)												
SIGL prior	18.17	48.04	69.33	25.43	26.77	48.91	100.00	0.00	58.91	50.61	88.44	16.33
PIFM (SIGL)	**26.93**	**59.48**	71.33	11.33	**42.21**	**60.76**	60.75	7.48	**85.60**	**83.21**	16.37	18.41
Mask rate: 50% (0.5 Drop)												
SIGL prior	16.88	**49.30**	72.01	27.85	22.90	52.55	100.00	0.00	50.05	45.41	87.68	18.80
PIFM (SIGL)	**17.08**	49.15	86.06	12.28	**28.38**	**59.58**	61.20	20.38	**59.83**	**58.11**	38.90	36.76

E.3 Denoising

Denoising is the complement of expansion, meaning that we seek to remove a set of spurious edges $\mathcal{E}_S$ from $A^{\mathcal{O}}$, such that the edge set of the ground truth is $\mathcal{E} = \mathcal{E}_O \setminus \mathcal{E}_S$. Hence, the initialization becomes $\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} \odot (f_{prior}(\mathbf{A}^{\mathcal{O}}) + \boldsymbol{\epsilon}_s)$. As in the expansion task, PIFM here is trained with the squared Frobenius distortion (MSE loss). We corrupt the observed graph by flipping 20% of the upper-triangle zero-entries into spurious edges (denoted 0.2 Flip); the results are shown in Table 11. Similarly to expansion, PIFM (GraphSAGE) attains the best AUC/AP on all datasets, again surpassing the GraphSAGE prior and remaining baselines. It *strongly* reduces false positives from the given prior initialization, while FNR is low on dense IMDB-B (2.67) and higher on sparser sets. Overall, PIFM removes spurious edges more reliably while also improving other metrics.

Table 11: Performance for the **denoising** task with **20% of upper-triangle 0-entries flipped (0.2 Flip)**. We report AUC, Average Precision (AP↑), False Positive Rate (FPR↓), and False Negative Rate (FNR↓), all in percent (%). The best result for each metric is in **bold blue** and the second best is green. The metrics are restricted on the upper-triangle 1-region of $A^{\mathcal{O}}$, and compared against $\mathbf{A}_1$ on that region.

Flip Rate: 20% (0.2 Flip)												
	ENZYMES				PROTEINS				IMDB-B			
Method	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓
Baselines												
GraphSAGE	68.19	73.89	**16.14**	61.72	73.79	76.70	**12.68**	60.47	92.54	77.29	16.77	53.37
DiGress + RePaint	41.98	49.38	87.91	**13.33**	49.36	51.20	78.44	**18.19**	80.54	51.59	73.41	23.49
GDSS + RePaint	44.59	50.86	69.18	29.70	49.72	49.43	68.39	32.64	82.36	53.23	69.05	26.00
Flow w/ Gaussian prior	49.90	54.56	52.30	38.93	57.18	58.70	62.84	24.32	96.75	94.63	3.80	12.66
Ours												
PIFM (GraphSAGE)	**69.40**	**77.17**	45.66	18.14	**77.43**	**81.78**	32.41	20.91	**98.46**	**96.52**	**2.67**	**12.10**

E.4 Loss-calibration analysis on sparse graphs

The MSE and CE instantiations of PIFM (Table 2) show a pronounced calibration difference on the sparser datasets, PROTEINS and ENZYMES. Here we analyze its cause.

Under the MSE distortion, PIFM is trained as a continuous regression toward the conditional mean. On a sparse graph, non-edges vastly outnumber edges, so the loss is dominated by the negative class and the fitted edge scores concentrate near zero (the conditional-mean target of the rare positive class is itself small). True edges are then treated as rare events and underweighted, and at the fixed decision threshold of 0.5 almost all of them are missed, inflating the false-negative rate (FNR). This is a thresholded-calibration effect rather than a ranking failure: the threshold-independent metrics (AP, AUC) remain informative even where FNR is high.

The cross-entropy (CE) instantiation addresses this directly. Inspired by CatFlow (Eijkelboom et al., 2024), which establishes cross-entropy as an effective distortion for flow matching on graphs, the CE distortion is an element-wise binary cross-entropy on the upper triangle of the masked region. In contrast to CatFlow's unweighted categorical objective, we additionally apply a per-batch positive-class weight $w = \min(n_{\text{neg}}/n_{\text{pos}}, 50)$ to counter the class imbalance specific to our conditional, sparse-graph setting. We emphasize that the positive weighting, not the switch from MSE to cross-entropy alone, is what restores calibration: an *unweighted* cross-entropy on these datasets would face the same negative-dominated gradient and leave the fitted probabilities below the 0.5 threshold, so the thresholded predictions again collapse to no edges, as with MSE. The weighting rebalances the per-edge contributions so that the rare positive class is no longer ignored, sharply reducing FNR (Table 2) at a substantial cost in FPR (which can rise from near zero to 45–89% on the sparsest datasets), while AP/AUC stay comparable. The clamp at 50 caps the weight on the very sparsest graphs, where an unbounded ratio would destabilize training.

These findings suggest that there are two natural directions for handling sparse graphs more effectively. One option is to adopt a BCE-style objective, which improves threshold calibration for binary edges but departs from the classical closed-form distortion-perception formulation. Another option is to retain the continuous distortion-perception objective while replacing the Gaussian perturbation with a noise model better suited to sparse binary data. We leave this second direction for future work.

E.5 Ablation

Our method has two main hyperparameters:

1. σ_s, which is used for computing $\boldsymbol{\epsilon}_s \sim \mathcal{N}(0, \sigma_s^2)$ in $\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \left(f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) + \boldsymbol{\epsilon}_s\right)$
2. K, which is the total number of steps in the Euler approximation

We ablate these two hyperparameters.

Performance as a function of σ_s. We run an ablation of the performance of PIFM with GraphSAGE as a function of σ_s. We focus on ENZYMES and IMDB, and we evaluate the ROC for the best value of K for each noise level.

The ablation is illustrated in Fig. 4. First, we observe that the gains of using PIFM are higher for a smaller drop rate, as expected; in particular, we observe that PIFM with σ_s jumps from ≈ 0.73 for $\sigma_s = 0$ to ≈ 0.81 for $\sigma_s = 0.1$. Second, for both configurations, performance peaks not at zero noise, but at a small noise level of $\sigma_s = 0.1$. This suggests that a slight injection of noise benefits model generalization. Beyond this optimal point, increasing the noise level leads to a steady decline in performance, meaning that the effect of the prior decreases, as expected.

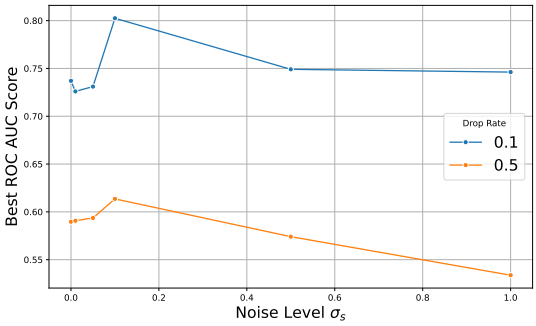


Figure 4: ROC as a function of the noise σ_s in $p(\mathbf{A}_0)$. The impact of noise level σ_s on model performance, measured by the best ROC AUC score. Results are shown for two different drop rates: 0.1 (blue) and 0.5 (orange). A small amount of noise improves performance for both configurations, after which increasing noise leads to performance degradation.

Performance as a function of K. To determine the optimal number of processing steps, K, we evaluated model performance while varying this parameter from 1 to 100. Figures 5 and 6 shows the results for a fixed drop rate of 0.1 and 0.5 respectively, across five different noise levels.

A key observation is that peak performance, in terms of AUC-ROC, is achieved within a very small number of steps, typically for $K < 10$. In particular, the introduction of a moderate noise level allows the model to achieve its highest overall score (≈ 0.80 ROC AUC) in a single step ($K = 1$). However, this advantage diminishes as the number of steps increases. The model without noise ($\sigma_s = 0.0$) provides the most stable and consistently high performance for larger K. Conversely, a high noise level ($\sigma_s = 1$) consistently degrades performance regardless of the number of steps. This analysis suggests a trade-off: while noise can provide a significant boost for models with very few steps, a no-noise configuration is more robust for models with a larger number of steps.

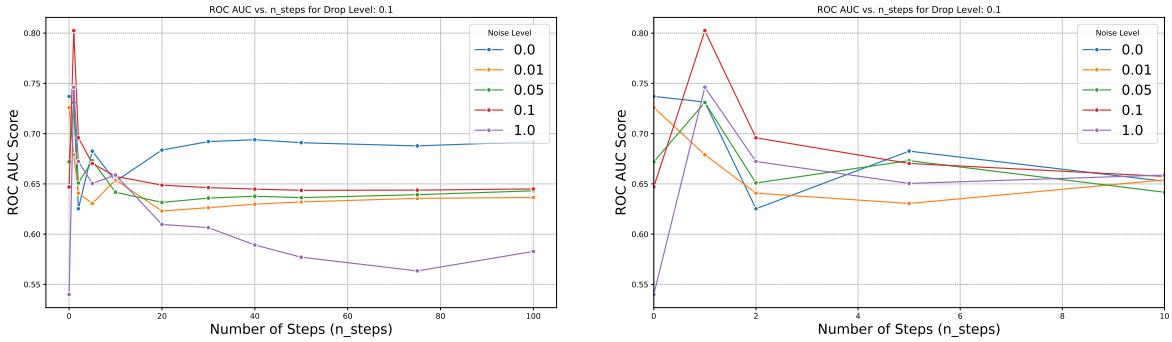


Figure 5: An analysis of the ROC AUC score as a function of the number of processing steps (K) for a drop rate of 10%. This experiment was conducted with a fixed drop rate of 0.1, while varying the noise level, σ_s. The results show that the optimal number of steps is small, typically under 10.

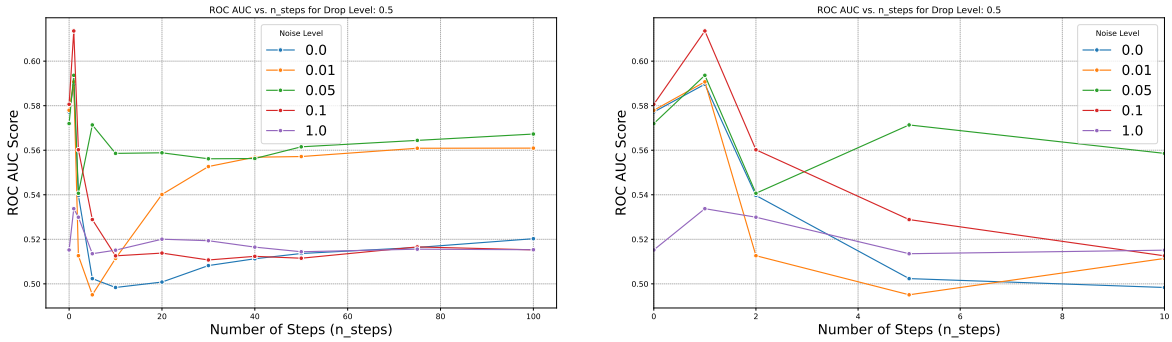


Figure 6: An analysis of the ROC AUC score as a function of the number of processing steps (K) for a drop rate of 50%. This experiment was conducted with a fixed drop rate of 0.5, while varying the noise level, σ_s. The results show that the optimal number of steps is small, typically under 10.

Threshold levels. We investigated the impact of alternative thresholding levels (0.3 and 0.7, beyond the standard 0.5), with results for ENZYMES and PROTEINS in Figs. 7 and 8, each reporting the ranking metrics and the binarized error rates as a function of the number of steps K. Since AP and AUC are computed from the raw scores, they are threshold-independent: the ROC-AUC-vs-K curves are the same across the three thresholds (left panels), so thresholding does not change ranking quality. The threshold instead governs the binarized error rates (center and right panels): lowering it reduces the FNR at the cost of a higher FPR (e.g., on ENZYMES at K=1 the FNR drops from 91.9 to 82.0 as the FPR rises from 4.5 to 9.9 when lowering from $\tau = 0.7$ to $\tau = 0.3$). This effect is appreciable only for small K (and for the prior at K=0), where the model's outputs lie nearer the decision boundary; for larger K the outputs are pushed towards the binary extremes (0 or 1) and the three thresholds nearly coincide. Crucially, across steps the FNR and FPR merely trade off, so an alternative threshold yields no significant improvement in overall performance; the principled rebalancing of these error rates is instead provided by the cross-entropy distortion (Sec. 4.2).

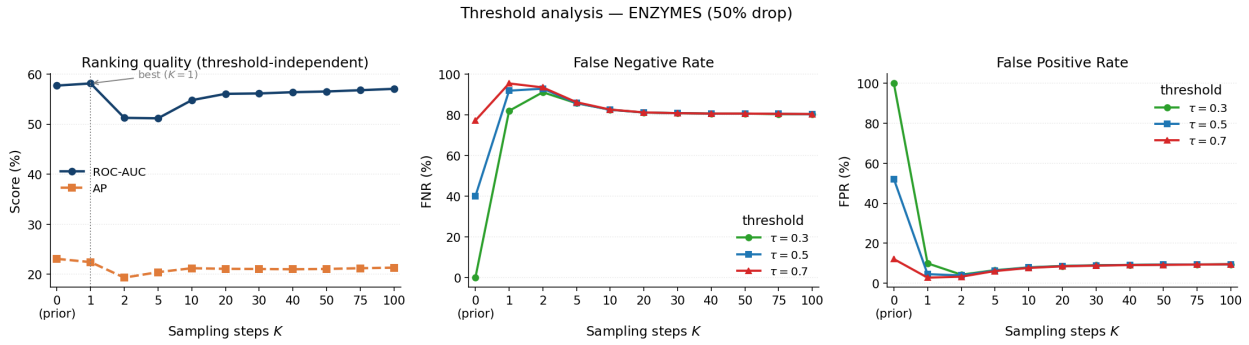


Figure 7: Threshold analysis on ENZYMES (link prediction, 50% drop) versus the number of sampling steps K. **Left:** ROC-AUC and AP are computed on the raw scores and are therefore *identical* for all binarization thresholds $\tau \in \{0.3, 0.5, 0.7\}$ (threshold-independent), peaking at K=1. **Center/Right:** the false-negative (FNR) and false-positive (FPR) rates at $\tau = 0.3, 0.5, 0.7$; the threshold shifts only these binarized error rates, and only appreciably at small K, where the outputs lie near the decision boundary. "K=0" is the prior initialization.

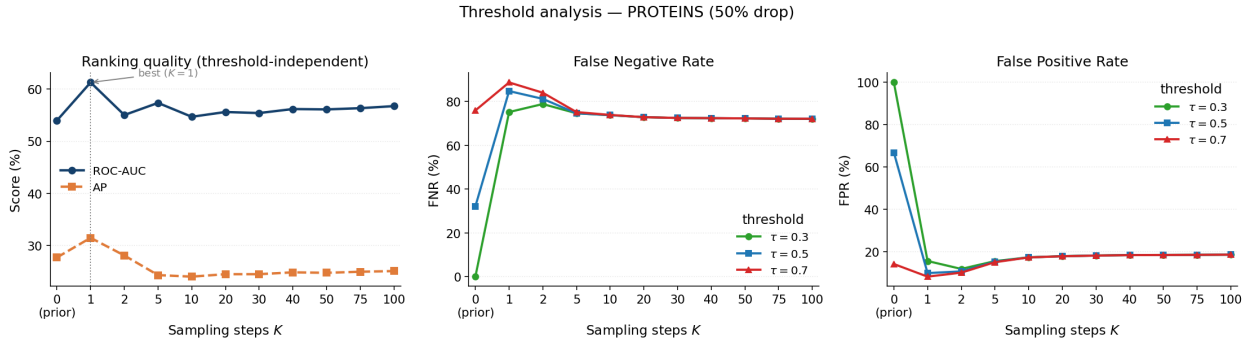


Figure 8: Threshold analysis on PROTEINS (link prediction, 50% drop) versus the number of sampling steps K, with panels as in Fig. 7. ROC-AUC and AP (left) coincide across thresholds, while lowering the threshold trades a lower FNR (center) for a higher FPR (right).

E.6 Distortion-perception trade-off.

Here we expand on the distortion-perception trade-off by computing the MMD as a function of the number of steps The results are shown in Figures 9 and 10. Again, both figures show the MMD² distance dropping sharply over the first few steps (most noticeably for $0 < \sigma_s \leq 0.1$), rather than decreasing monotonically in K: for some noise levels it plateaus or rebounds after its early minimum. In other words, if we aim for a high-quality perceptual reconstruction, we should consider $\sigma_s = 0.01$ or 0.05. However, if we are aiming for high reconstruction quality in terms of AUC-ROC, we should use $\sigma_s = 0.1$ (see Fig.4). In other words, the choice of the K is heavily dependent on the downstream task.

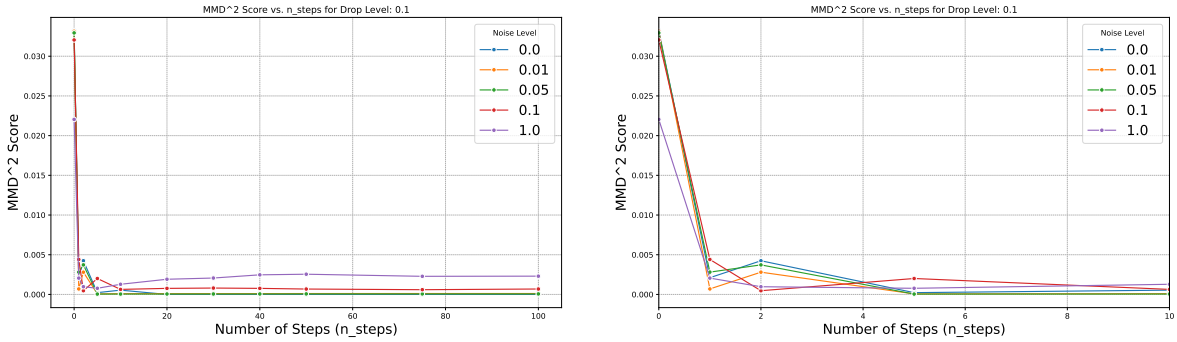


Figure 9: Analysis of the perception component of the distortion-perception trade-off. The plot shows the MMD² score (where lower is better) versus the number of steps, K, for a fixed drop rate of 0.1. Each line represents a different noise level σ_s.

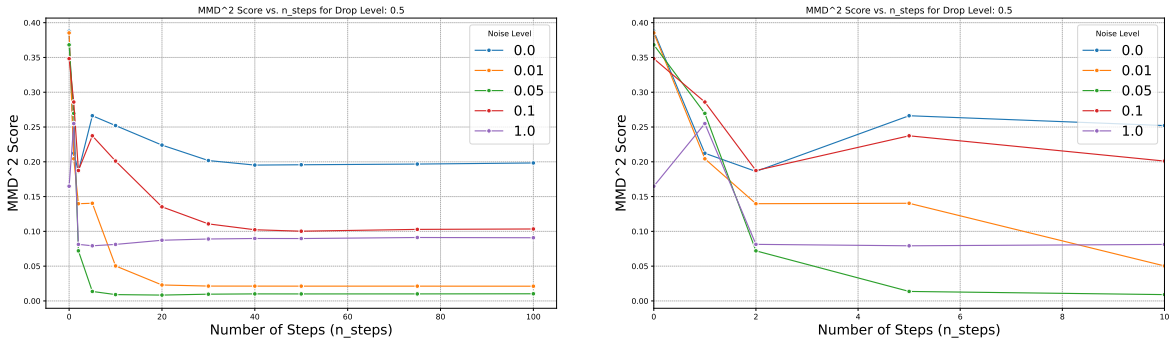


Figure 10: Analysis of the perception component of the distortion-perception trade-off. The plot shows the MMD² score (where lower is better) versus the number of steps, K, for a fixed drop rate of 0.5. Each line represents a different noise level σ_s.

E.7 Evaluation using additional metrics

This section incorporates additional metrics to showcase the observed performance trade-off. The results for IMDB are in Figs. 11 and 12, for PROTEINS in Figs. 13 and 14, and for ENZYMES in Figs. 15a and 15b. While an increased number of steps yields an improvement in generating an estimated graph with statistics that more closely align with the ground-truth distribution, the reconstruction performance (measured in terms of AUC) declines relative to the initial step. Critically, the trend is found to be highly contingent on the underlying dataset's sparsity. For the dense case (IMDB), the AUC exhibits a consistent monotonic decrease after the optimal initial guess, independent of drop rates. In contrast, the sparser PROTEINS and ENZYMES datasets demonstrate an intermediate improvement as the number of steps increases, though their overall AUC still trails that achieved at the initial step ($K = 1$).

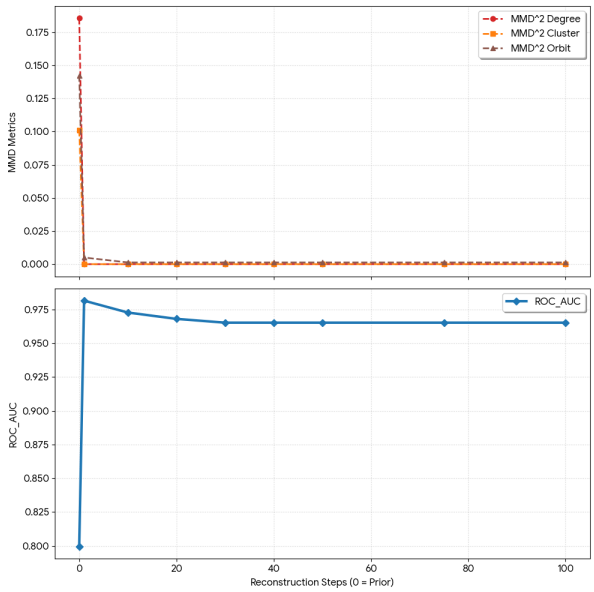


Figure 11: IMDB dataset, expansion task, 10% drop rate

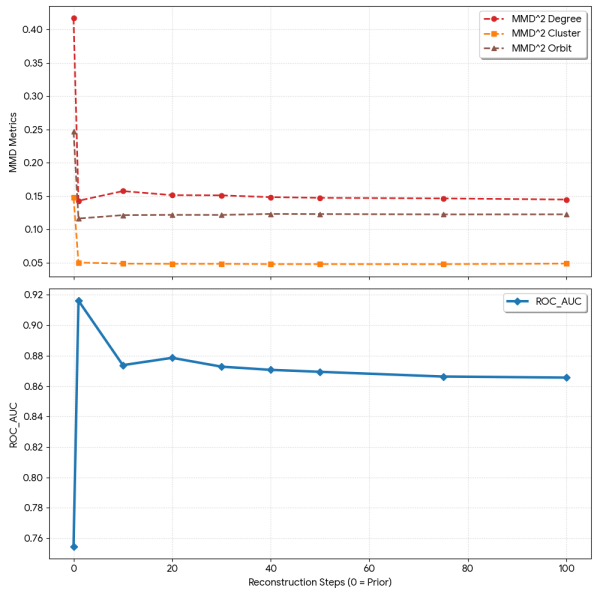


Figure 12: IMDB dataset, expansion task, 50% drop rate

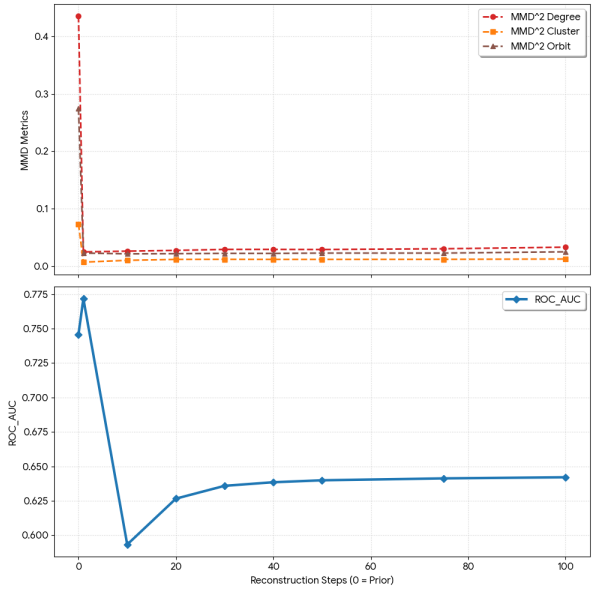


Figure 13: PROTEINS dataset, expansion, 10% drop rate

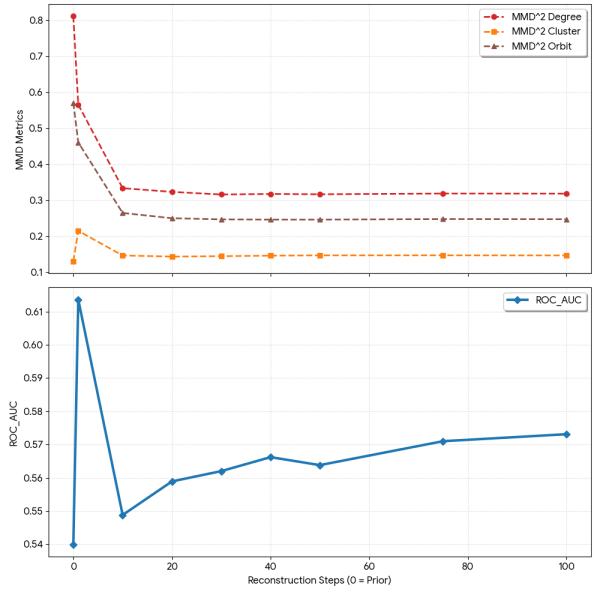
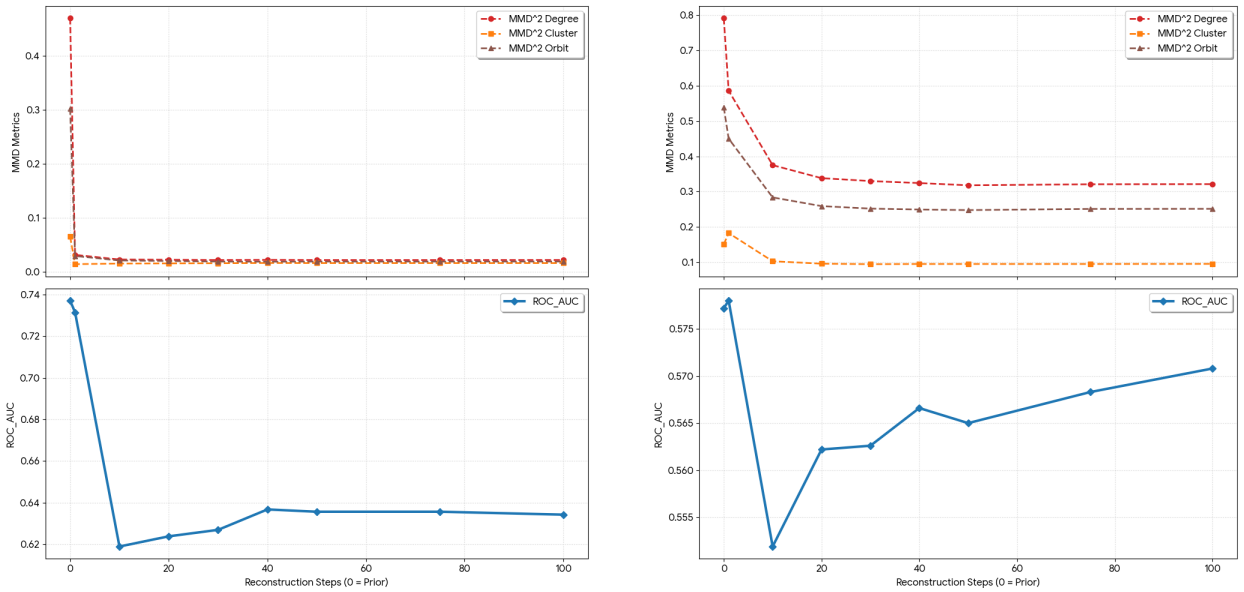


Figure 14: PROTEINS dataset, expansion, 50% drop rate



(a) 10% drop rate

(b) 50% drop rate

Figure 15: ENZYMES dataset, expansion task.

E.8 PIFM on large-scale graphs

In this section, we train PIFM on large-scale graphs. In particular, we focus on CORA Yang et al. (2016).

Building the dataset. To enable scalable diffusion training on Cora while maintaining full-graph link prediction capabilities, we introduce a subgraph-based variant of PIFM following Limnios et al. (2023). We instantiate this via an edge-centered sampling scheme, where each subgraph represents a k-hop ego-network (capped at a maximum node count) centered around a seed edge from the training split. To ensure reproducibility, we sample a fixed set of seed edges (both positive and negative) and corresponding subgraphs which remain constant throughout each run.

Following the 50/15/35 (train/validation/test) edge split, we evaluate link prediction on held-out edges using the protocol established in the main paper. Furthermore, we utilize purely structural node features. By default, we concatenate: (i) Laplacian positional encodings derived from the smallest k generalized eigenvectors of $Lv = \lambda Dv$ computed on the training adjacency; and (ii) a 2-dimensional local context vector comprising both raw and normalized node degrees.

Training. To initialize PIFM, we first pre-train the NCN structural prior on the full Cora graph using only the training edge split (50% of edges). These learned embeddings are subsequently used to initialize PIFM for inference on the test split (35% of edges).

During PIFM training, each training edge seeds a unique edge-centered subgraph. Within each subgraph, we define the observed context as all other training edges, and a hidden region comprising all remaining node pairs (including non-edges and edges outside the context). The seed edge is explicitly masked from the context and marked as the only supervision target. Consequently, each training example tasks the model with reconstructing a single missing edge within its local neighborhood.

We construct the initial state matrix $\mathbf{A}_0$ by combining the observed context with structural prior predictions in the hidden region. The flow model is trained by flow-matching under the squared Frobenius distortion (MSE loss), with the loss computed only on the supervised entries. While the flow ODE updates all entries within the hidden region during inference, the gradient is computed exclusively from the seed edge. This

adapts the global training procedure of Algorithm 1 to a localized, subgraph-based regime. Noise is still added to the seeded edge during training, but no noise is used during inference

Inference. For inference, every held-out test edge seeds a k-hop subgraph. We define the observed context using the training edges, remove the centered test edge from the mask, and apply diffusion to the resulting hidden region. We evaluate stitched scores for held-out edges. To resolve potential overlaps, we aggregate predictions via logit averaging across all subgraphs where a specific edge is present, producing a single probability matrix over all node pairs. We then compute metrics on the held-out positive and negative edges. Since PIFM and the structural prior baseline are evaluated on the exact same set of pairs, the results isolate the specific benefits of the diffusion process.

Results. Results in Table 12 show that PIFM is learning useful global structural representations even on very large graphs by exploiting the subgraph sampling approach.

Table 12: Link prediction on Cora (test edges split) with an NCN structural prior. We report AUC, Average Precision (AP), False Positive Rate (FPR), False Negative Rate (FNR). We report PIFM's performance on Cora for different k-hop neighborhoods during subgraph sampling, and some notable hyperparameters we tuned. We include also the average node counts for each subgraph for each case. Best values for each metrics is in **bold**. All the models are all evaluated on the same 50/15/35 edge split, and the results are using the best saved checkpoints based on AUC for each combination of the hyperparameters.

Method	AUC↑	AP↑	FPR↓	FNR↓
NCN Wang et al. (2024)	93.78	**93.89**	23.50	**6.71**
1hop ($\sim$ 8.9 nodes)				
`lr=2e-5, bs=128, train_noise_std=0.02, hidden_dim=32, c_init=1, c_hid=8, c_final=8`	**93.89**	93.71	**20.95**	7.69
2hop ($\sim$ 25.8 nodes)				
`lr=2e-5, bs=128, train_noise_std=0.01, hidden_dim=32`	93.82	93.75	22.04	6.98

E.9 Transferability

We test whether a PIFM model trained on one dataset still works on another. We take a PIFM checkpoint trained on a *source* dataset and run it on a different *target* dataset at the same drop rate; Table 13 reports these transfer results.

Since the flow always starts from the prior, we split the two parts. Table 14 shows the GraphSAGE prior on its own (no flow): this is just the prior computed on the target graphs, and it does not depend on the source checkpoint. Table 13 then shows what the transferred flow adds on top of that prior. The gap between the two tables is the effect of the flow.

Table 13: **PIFM transfer.** A PIFM flow trained on the *source* dataset, run on a different *target* dataset at the same drop rate, starting from the target's prior (Table 14). For each run we report the step with the highest AUC (K in parentheses) and the last step (K=100). Metrics are Average Precision (AP), AUC, False Negative Rate (FNR), and False Positive Rate (FPR), all in percent.

Source checkpoint	Target dataset	Step	AP↑	AUC↑	FNR↓	FPR↓
PROTEINS (10%)	ENZYMES (10%)	best (K=1)	43.87	76.96	71.96	4.24
		final (K=100)	37.83	59.43	64.45	4.87
IMDB-B (10%)	PROTEINS (10%)	best (K=1)	46.79	75.53	47.07	11.62
		final (K=100)	43.12	60.24	43.65	51.66
ENZYMES (10%)	PROTEINS (10%)	best (K=1)	49.63	76.60	59.02	6.63
		final (K=100)	42.36	70.64	53.12	8.20
PROTEINS (50%)	ENZYMES (50%)	best (K=1)	25.68	61.88	95.37	1.70
		final (K=100)	21.05	58.68	80.82	8.67
IMDB-B (50%)	PROTEINS (50%)	best (K=75)	24.88	59.02	61.02	24.32
		final (K=100)	24.42	57.22	60.80	24.37
ENZYMES (50%)	PROTEINS (50%)	best (K=1)	26.81	55.80	86.13	9.97
		final (K=100)	24.54	54.26	76.68	16.23

Table 14: **Prior only (no flow).** The GraphSAGE prior computed on the target graphs — the starting point for every PIFM run in Table 13. It depends only on the target, not on the source, so it is the same for every source→target pair with the same target. Metrics are AP, AUC, FNR, and FPR, all in percent.

Target dataset	AP↑	AUC↑	FNR↓	FPR↓
ENZYMES (10%)	41.28	73.70	13.49	60.59
PROTEINS (10%)	46.36	74.58	11.00	63.50
ENZYMES (50%)	23.08	57.72	40.02	52.16
PROTEINS (50%)	27.71	53.99	32.16	66.86

From the table we can see that the model degrades the predictions a lot at lower (10%) drop rates, making the results incomparable when comparing to results in Table 9. It can be seen that with more steps integrated, at the final step $t = 100$ the metrics are much worse than the best result over all sampling steps or the structural prior. Although at higher (50%) drop rates the best results are more comparable to the best results in Table 2, this is due to the graphs being very corrupted under 50% drop rate and hence the observed graphs are much more degraded than they used to be.

F Visual results

F.1 Examples of reconstructed graphs

We show here a few samples for the expansion case. We plot the samples from ENZYMES, using a subset of the dataset used in Section 5.3.

Binary comparison. In this case, we first compare the thresholded versions (with 0.5) of the mean matrices. We compute this for 3 graphs in the test set (24, 16 and 21). The plots are in Figs. 16, 17 and 18

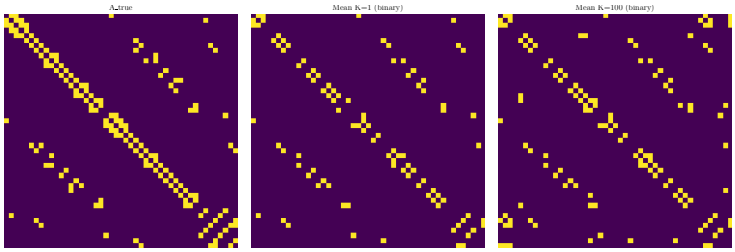


Figure 16: Graph reconstruction for sample 24, thresholded with 0.5

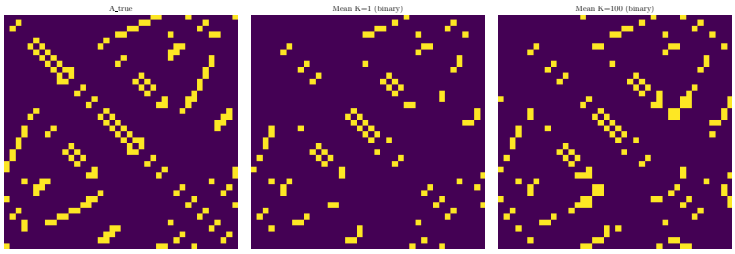


Figure 17: Graph reconstruction for sample 16, thresholded with 0.5

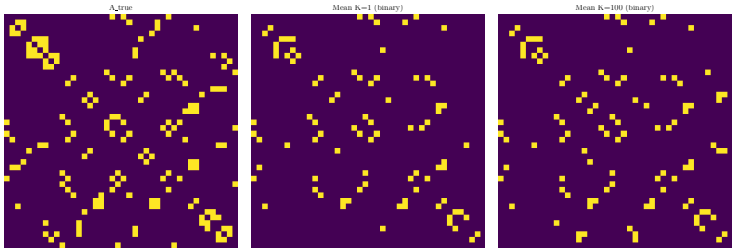


Figure 18: Graph reconstruction for sample 21, thresholded with 0.5

Raw comparison - Mean. In this case, we compare the raw versions of the mean matrices. We compute this for 3 graphs in the test set (24, 16 and 21). The plots are in Figs. 19, 20 and 21. Notice that the mean reconstructions for $K = 100$ have values that are between 0 and 1; this can be explained by looking at individual samples (see below), which are more diverse, and therefore, they have non-overlapping set of existing edges.

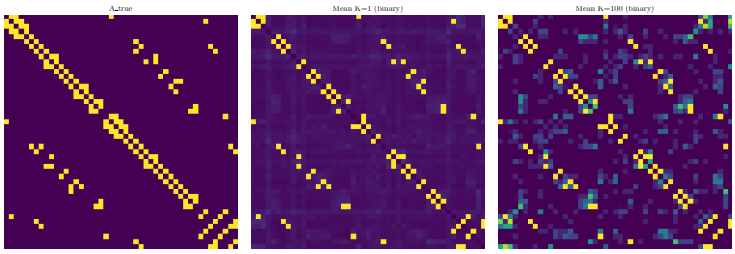


Figure 19: Graph reconstruction for sample 24, mean raw

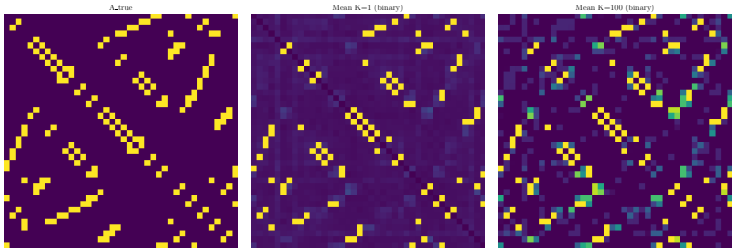


Figure 20: Graph reconstruction for sample 16, mean raw

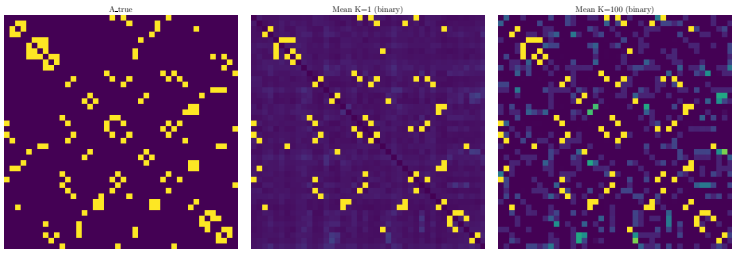


Figure 21: Graph reconstruction for sample 21, mean raw

Raw comparison - Median. In this case, we compare the raw versions of the median matrices. We compute this for 3 graphs in the test set (24, 16 and 21). The plots are in Figs. 22, 23 and 24.

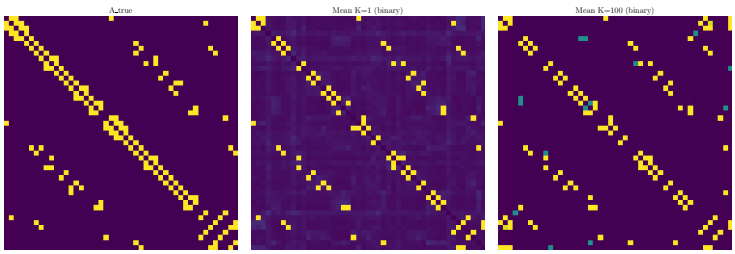


Figure 22: Graph reconstruction for sample 24, median raw

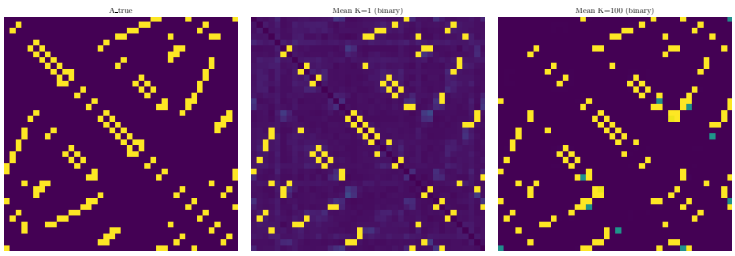


Figure 23: Graph reconstruction for sample 16, median raw

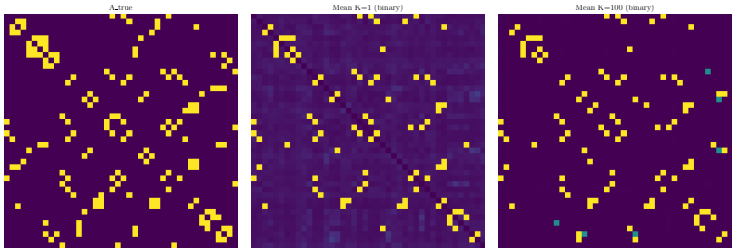


Figure 24: Graph reconstruction for sample 21, median raw

Individual samples for each graph. Lastly, we show the raw versions of different realizations (individual samples) for each graph. We compute this for 3 graphs in the test set (24, 16 and 21). Interestingly, the samples for $K = 100$ are more diverse (similar to the case of images in Ohayon et al. (2025)); this diversity explains why the raw mean in Figs. 19, 20 and 21 have values that are not exactly 0 or 1 (which indicates that the samples disagree on these entries rather than sharing a single structure).

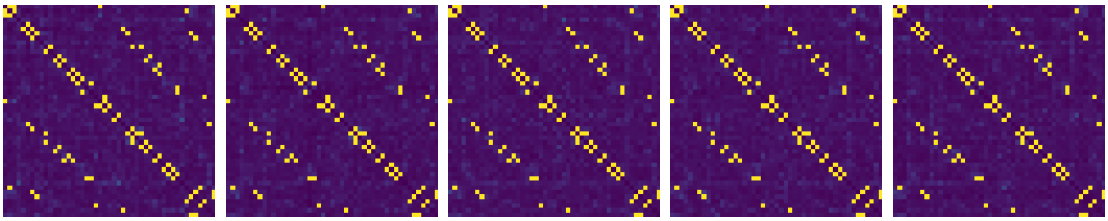


Figure 25: Individual samples for $K = 1$ and for sample 24

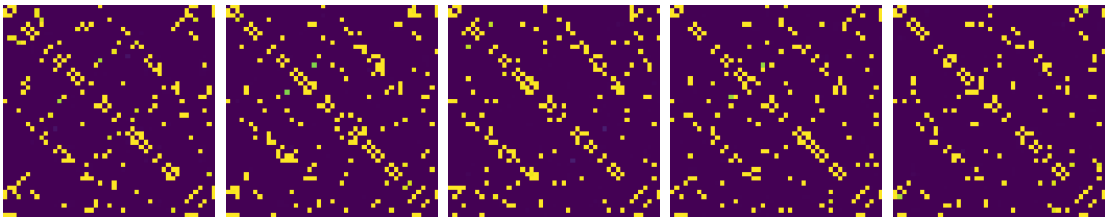


Figure 26: Individual samples for $K = 100$ and for sample 24

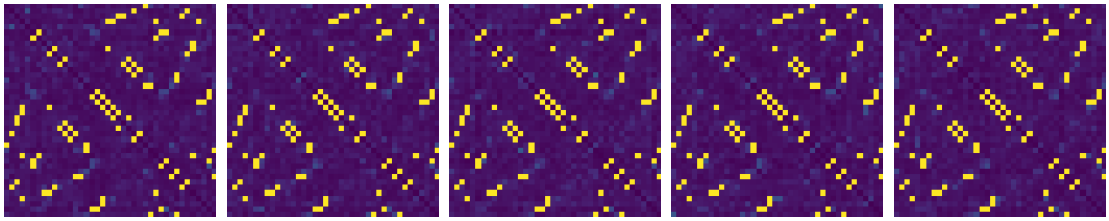


Figure 27: Individual samples for $K = 1$ and for sample 16

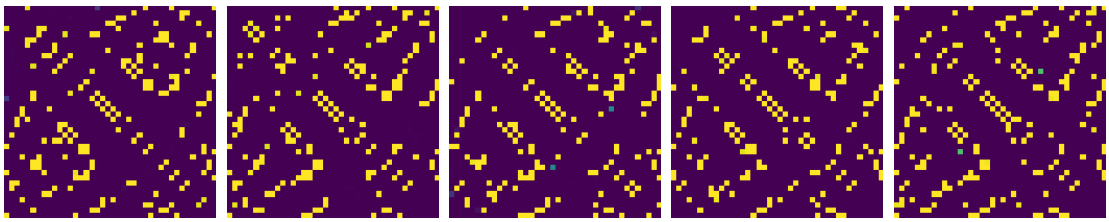


Figure 28: Individual samples for $K = 100$ and for sample 16

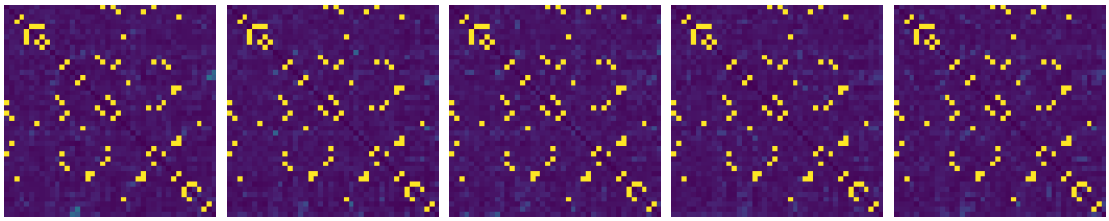


Figure 29: Individual samples for $K = 1$ and for sample 21

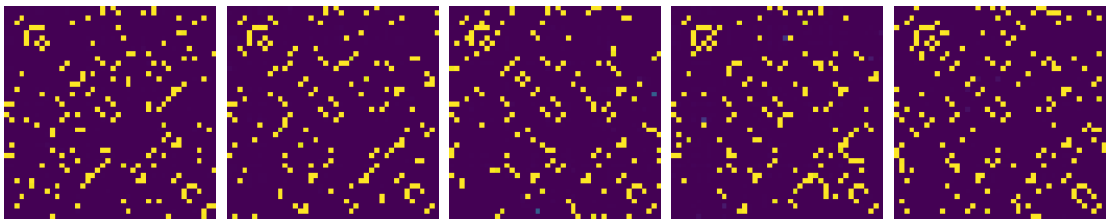


Figure 30: Individual samples for $K = 100$ and for sample 21

F.2 Intermediate adjacency matrices

In Figs. 31- 33, we show visualizations of the diffusion trajectory of the link-prediction sampler by "snapshotting" the predicted adjacency matrix at steps 0 to 100 in 10-step-increments of a sample path with 100 total steps. Each panel shows the raw adjacency values, zeroed on the diagonal, rendered with the colormap with black associated to 0 and yellow to 1. The bottom-rightmost panel is the ground-truth adjacency for comparison.

To see the sampling process, progress from left-to-right and top-to-bottom shows how the sampler denoises toward the final reconstruction (steps=100). From these images we could see the smooth transitions along the full reconstruction trajectory of PIFM.

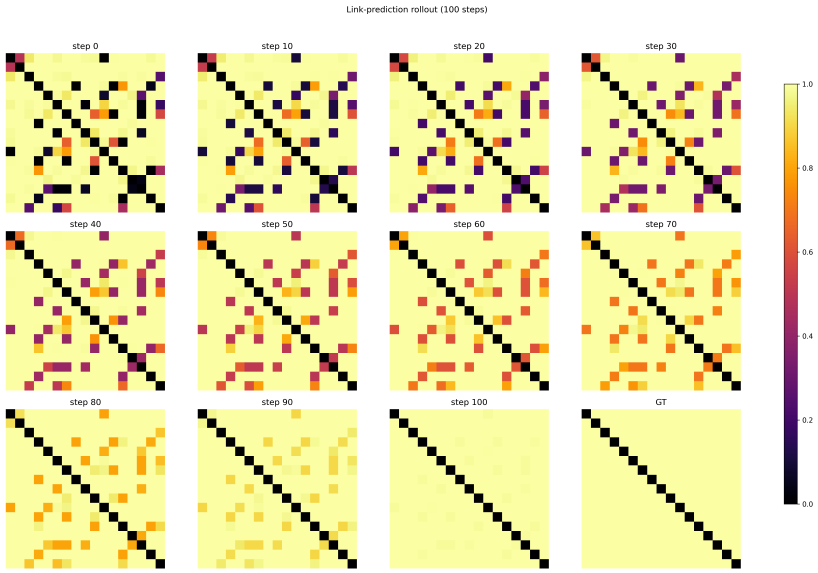


Figure 31: Visualization of IMDB 50% drop rate reconstruction. (Graph 1)

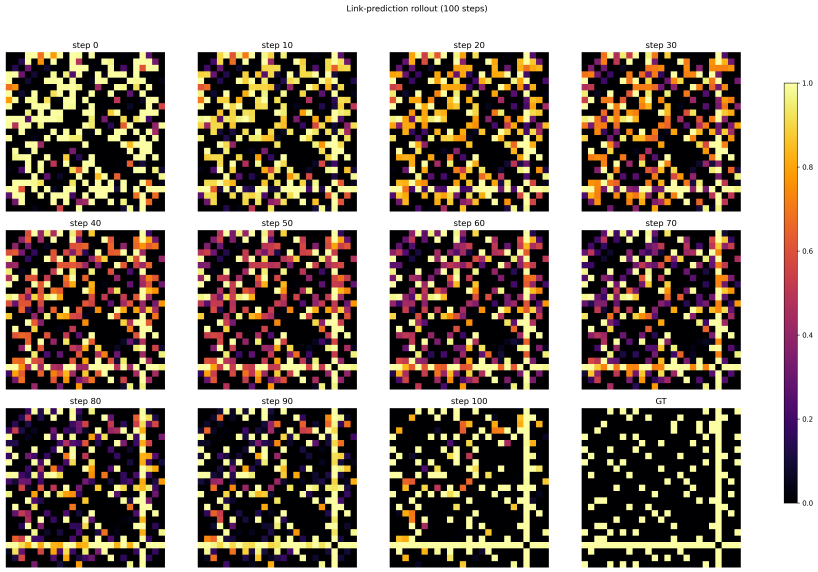


Figure 32: Visualization of IMDB 50% drop rate reconstruction. (Graph 2)

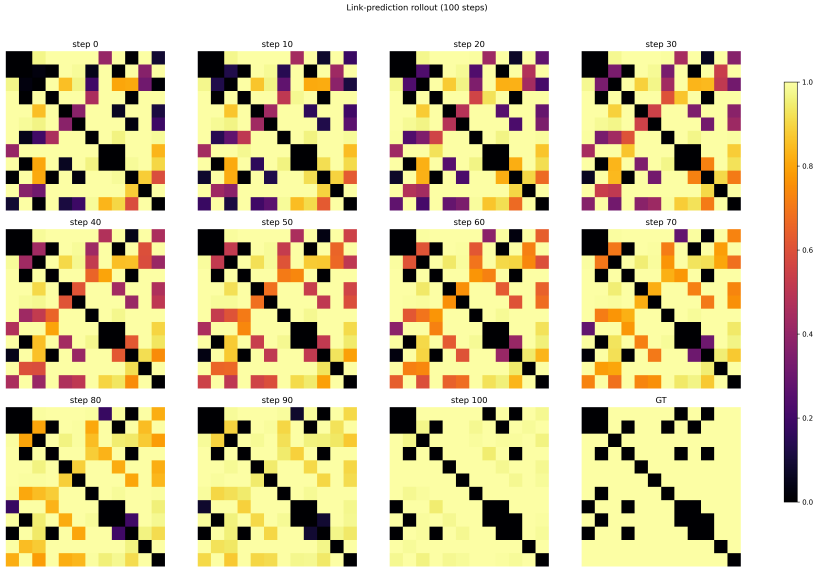


Figure 33: Visualization of IMDB 50% drop rate reconstruction. (Graph 3)