Your Discrete Graph Diffusion Model is Secretly Memorizing

Adarsh Jamadandi, Nicolas Keriven
CNRS, IRISA, Rennes, France
adarsh.jamadandi@irisa.fr, nicolas.keriven@cnrs.fr

Abstract

Discrete graph diffusion models (GDMs) corrupt the graph structure by adding noise independently to nodes and edges under a Markovian noise model. In the unconditional setting without node features, an equivariant denoiser trained on a single graph is able to generate novel graphs and sample-level metrics such as edge overlap and WL kernel similarity find no evidence of memorization. In this work, we challenge this view and show graph diffusion models indeed exhibit memorization. Analyzing the reverse diffusion trajectories of GDMs trained on two-block stochastic block models (SBMs), we find a strong train-data bias: the model recovers a corrupted training graph almost perfectly compared to a held-out test graph. We term this *conditional memorization*, as it surfaces only when the reverse trajectory is conditioned on a corrupted training graph and is not apparent during inference. It coincides with a growing generalization gap, where the training loss improves as the loss on held-out graphs degrades, and it cannot be mitigated by early stopping. The onset of memorization always precedes the emergence of community structure in the generated graphs, thus every checkpoint earlier produces structureless graphs. This behavior holds across all node sizes we test.

1 Introduction

Diffusion models [32, 34, 16, 25] have recently emerged as a powerful class of generative models, with major improvements in image [37] and video generation [24]. Motivated by this success, diffusion-based methods have been extended to graph generation. The two most popular approaches are: embedding nodes into a continuous latent space and applying Gaussian diffusion [26, 18, 10], or operating directly in the graph space through a discrete corruption process on edges [3]. The latter approach has been shown to be more appropriate for generating realistic graph structures [15, 35], leading to a growing body of work on both graph diffusion and discrete flow matching [13, 28, 30].

An interesting empirical observation made in [9, 23] is that we can train graph diffusion models on a single graph and still generate novel and diverse graphs without memorizing the training data. This seems counter-intuitive compared to diffusion models applied to images [19, 5] which show dataset size as one of the factors driving generalization. In contrast, the study of memorization in graph diffusion models has received little attention; the question is instead studied under the guise of model expressivity. Recent works such as [21, 22] show that equivariant denoisers cannot copy their training graphs exactly, as this would require breaking the permutation symmetry of the input graphs, which equivariant models by design cannot do. A single training graph has $\Omega(n!)$ graphs in its isomorphism class [36], and an equivariant denoiser cannot reproduce a specific graph and instead produces an average over that class. Further, the sample-level metrics commonly used to assess whether the model has reproduced the training graph such as edge overlap [8, 7], which measures the intersection of exact edge pairs between the training and generated graph, and WL kernel similarity [31], which certifies that two graphs are not isomorphic, both assert no memorization.

Preprint.

In this work, we challenge this view and show graph diffusion models do exhibit a form of memorization, which we term *conditional memorization*: the training graph is recoverable from the reverse dynamics, but only when the trajectory is seeded with a corrupted version of that graph, and is otherwise undetectable during inference. Our contributions can be summarized as follows:

1. We study the learning dynamics of single-graph diffusion models on two-block SBMs at $n = \{64, 128, 256, 512\}$ and show the models exhibit *conditional memorization*, where the model restores a corrupted training graph almost perfectly against a held-out graph, a bias that is invisible in ordinary unconditional generation and surfaces only by probing the reverse diffusion trajectories.
2. We show the minimum held-out loss is a poor proxy for model selection, since memorization onset ($\tau_{\texttt{mem}}$) consistently precedes the emergence of community structure ($\tau_{\texttt{struct}}$) across all four sizes: the model cannot generate the right structure without first entering the memorization regime, so early stopping cannot separate the two.

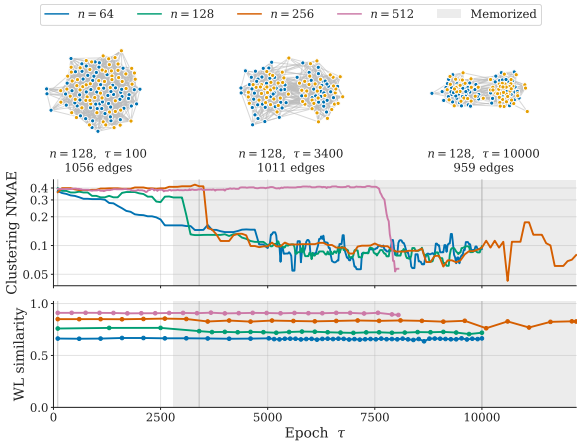


Figure 1: Sample-level metrics report a well-generalizing model. Clustering coefficient NMAE (lower is better) improves throughout training while WL kernel similarity remains flat, suggesting the samples acquire the correct community structure without reproducing the training graph.

2 Sample-level metrics are blind to memorization

Data. We generate synthetic graphs with nodes $n = \{64, 128, 256, 512\}$ drawn from a symmetric two block stochastic block model (SBM) with block probabilities (p_{in}, p_{out}) chosen at each size to hold the Kesten–Stigum detectability [1] margin fixed at 7.0 and mean degree fixed at ≈ 16 (both well above the detectability threshold of 1), so that graph size is varied independently of task difficulty, details are present in Table 2. We choose this setting as it allows us to control the true generating process and it is easier to see the *structure* emerging during training.

Model setup. We use EDGE [9] as our backbone diffusion model as it is computationally efficient compared to models like DiGress [35] which scales quadratically in the number of nodes. Given an input graph G with adjacency matrix A, the forward corruption involves deleting the edges resulting in an empty graph as a terminal distribution. The reverse process predicts edges between active nodes, let $s_i^t = \mathbf{1}[d_i^{t-1} \neq d_i^t]$ be a latent variable marking whether node i's degree changed at step t. With $d^t = \deg(A^t)$, the forward process is given by

$$q\big(A^{1:T}, s^{1:T} \mid A^0\big) = \prod_{t=1}^{T} q\big(A^t \mid A^{t-1}\big)\, q\big(s^t \mid A^{t-1}, A^t\big), \tag{1}$$

Marginalizing Equation 1 over intermediate steps gives the absorbing marginal, in which each present edge survives to step t independently with probability $\bar{\alpha}_t$ giving rise to $q\big(A_{ij}^t = 1 \mid A_{ij}^0 = 1\big) = \bar{\alpha}_t$ and $q\big(A_{ij}^t = 1 \mid A_{ij}^0 = 0\big) = 0$ where $\bar{\alpha}_t$ follows a cosine schedule with $\bar{\alpha}_T = 0$, so that edges are

only ever removed and the terminal state A^T is the empty graph. The reverse process is given by

$$p_\theta\big(A^{0:T}, s^{1:T}\big) = p\big(A^T\big) \prod_{t=1}^{T} p_\theta\big(A^{t-1} \mid A^t, s^t\big)\, p_\theta\big(s^t \mid A^t\big), \tag{2}$$

where s^t in (1) is a deterministic function of A^{t-1}, A^t, and in (2) the model first predicts which nodes are active and then predicts edges only between them, copying every other pair forward unchanged. The posterior $q(s^t \mid d^t, d^0)$ has closed form in terms of the degree and depends on A^0 only through d^0.

Pre-mature generalization. To demonstrate that existing sample level metrics are blind to the memorization, we plot the normalized mean average error (NMAE) of clustering coefficient and the WL kernel similarity for graphs generated by the diffusion model for SBMs with $n = \{64, 128, 256, 512\}$ in Figure 1. We can observe that as the training progresses, the NMAE improves consistently (lower is better), indicating the generated graphs are generating graphs with the right community structure and the WL kernel similarity is flat across the training epochs implying, the generated graphs are novel and diverse. Seeing these results one would conclude rather pre-maturely that the model is indeed generalizing. However, as we show in the next section, the model is actually memorizing the training graph, this is indicated as a growing gray band in the plot, interestingly looking at the generated samples, we can observe that the community structure of the graph only appears well inside this memorization regime and before this onset, the generated samples are structureless blobs.

3 Conditional memorization in the reverse diffusion trajectories

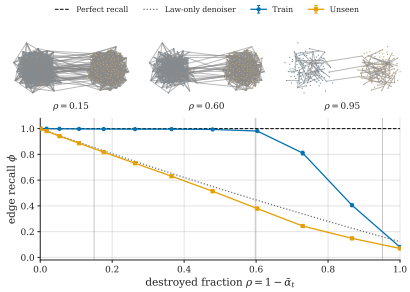


Figure 2: We measure the ability of the denoiser to recover the training graph vs. test graph and find the model exhibits a training-data bias where it can recover the training graph perfectly.

To show that the graph diffusion models indeed exhibit conditional memorization, we adopt the *U-turn* experimental setup proposed in [12]which involves analyzing the reverse diffusion trajectories seeded with the training data vs. the test data.

Reference law-only denoiser. We begin by defining a law-only denoiser that can serve as an unbiased reference. The reference denoiser knows the generating law and the corruption level, but does not have access to the graph itself. Concretely, it knows the block assignment z and the block probabilities $p_{z_i z_j}$. This gives us a baseline behavior we can expect from a model that is not memorizing a particular training graph. The details of which are present in Appendix C.

U-turn probe for edge recall. Let A^0 be the edge states of a reference graph, with edge set $\mathcal{E}(A^0)$ and degree sequence $d^0 = \deg(A^0)$. We corrupt A^0 up to step t using the forward process: each edge is deleted independently with probability $\rho = 1 - \bar{\alpha}_t$, which we refer to as the destroyed fraction. We then run the reverse process from A^t back to $t = 0$ to obtain a reconstruction $\tilde{A}^0$. At each reverse step the model first predicts the active nodes s^t and then predicts edges only between active pairs, with d^0 supplied as an auxiliary feature. We measure edge recall as

$$\phi_{\texttt{edge-recall}} = \mathbb{E}\left[\frac{\left|\mathcal{E}(\tilde{A}^0) \cap \mathcal{E}(A^0)\right|}{|\mathcal{E}(A^0)|}\right], \tag{3}$$

estimated as the mean over R independent reverse trajectories. More details are given in Appendix C. Figure 2 plots $\phi_{\texttt{edge-recall}}$ against ρ for $n = 256$, where $\phi = 1$ denotes perfect recall, every corrupted edge restored to its original position. The denoiser exhibits a train-data bias, for instance, for $n = 256$ at $\rho = 0.60$ it restores 96.9% of the corrupted edges of the training graph, against 7.7% for a denoiser that has learned only the generating law, and recall on an unseen test graph stays close to that reference throughout. We find this behavior across node sizes although the edge recall rate varies for different node sizes.

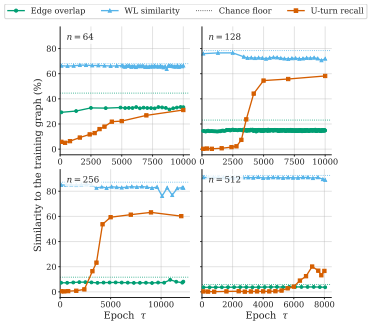


Figure 3: Analyzing the edge recall through a U-turn probe allows us to detect memorization.

We term this *conditional memorization*: the training instance is recoverable from the model's reverse dynamics, but only when those dynamics are initialized inside the region of noisy states the model was trained on, a region unconditional sampling does not enter. A similar observation has been noted in contemporaneous work [20], although our analysis suggests a different conclusion, that is, it is not that the models memorize during training and generalize in inference, it is that the sample-level metrics commonly used to assess generalization fail to capture this bias enabling a premature conclusion of the model's generalization abilities which can have unintended consequences for privacy-preserving deployment [6, 33].

Edge recall for memorization. In Figure 3, we plot edge overlap, WL kernel similarity and edge recall from the U-turn experiment across node sizes. Edge overlap and WL similarity stay flat, which would ordinarily indicate no memorization. This is an artifact of how sampling works: inference starts from an empty graph and never passes near the training graph, so any sample-level comparison is clean by construction. The U-turn probe queries the model where it was actually trained, and there edge recall climbs steadily, exposing a bias that sample-level metrics cannot see.

Community structure emerges only after memorization Garnier-Brun et al. [12] study a phenomenon of biased generalization where the test loss improves while the diffusion model continues to show unusually high ability to recover the training samples. In contrast, we observe that a graph diffusion model progressively overfits the denoising objective, resulting in a wide generalization gap. We measure the community structure of the generated samples by their average clustering coefficient c, normalized against two references: the training graph's own clustering c_{train}, and a floor $c_{\text{ER}}(\tau) = \bar{d}_{\text{gen}}(\tau)/(n-1)$, the clustering expected of an Erdős–Rényi graph of the same mean degree, giving $\mathcal{S}(\tau) = \frac{c_{\text{gen}}(\tau) - c_{\text{ER}}(\tau)}{c_{\text{train}} - c_{\text{ER}}(\tau)}$. A value of 0 means the samples are as unstructured as a random graph of their own density and 1 means they match the training graph.[1] We define $\tau_{\texttt{struct}}$ as the first epoch at which $\mathcal{S}(\tau)$ reaches 0.5 and stays there for two consecutive evaluations.

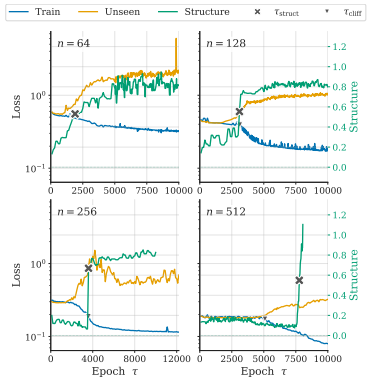


Figure 4: Train and test loss with the structure fraction $\mathcal{S}(\tau)$ across node sizes. Community structure emerges at $\tau_{\texttt{struct}}$ well after the onset of overfitting at $\tau_{\texttt{cliff}}$.

Figure 4 plots $\mathcal{L}_{\texttt{train}}$ and $\mathcal{L}_{\texttt{test}}$ alongside $\mathcal{S}(\tau)$, with $\tau_{\texttt{struct}}$ annotated: community structure appears well inside the regime in which the model has already begun overfitting the training graph. The memorized regime cannot be avoided by stopping early. Table 1 gives the onset of instance recall $\tau_{\texttt{mem}}$ and the emergence of community structure $\tau_{\texttt{struct}}$ at every size, and the ordering is the same in all the four cases: $\tau_{\texttt{mem}} < \tau_{\texttt{struct}}$. Every checkpoint that generates graphs with the right structure is already past the onset of instance recall, and every checkpoint before it produces structureless samples. This also makes the test loss a poor selection criterion, since its minimum returns a model that has not yet learned the community structure at all.

4 Conclusion

In this work we show that single-graph diffusion models exhibit what we term *conditional memorization*, where the denoiser is capable of recovering the training graph with perfect recall compared to a test graph. We demonstrate this phenomenon on synthetic graphs drawn from a SBM across different node sizes. Further, we show that the community structure in the graph only appears after the onset of memorization and thus choosing a model based on test loss will result in a sub-optimal denoiser that outputs blob-like graphs.

[1] the floor is recomputed at every evaluation from the samples' *own* mean degree, because clustering rises with density and a fixed floor would credit a denser sample with structure it does not have.

References

[1] Emmanuel Abbe. Community detection and stochastic block models: Recent developments. *Journal of Machine Learning Research*, 18(177):1–86, 2018. URL http://jmlr.org/papers/v18/16-480.html.

[2] Luca Ambrogioni. The statistical thermodynamics of generative diffusion models: Phase transitions, symmetry breaking and critical instability, 2024. URL https://arxiv.org/abs/2310.17467.

[3] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/958c530554f78bcd8e97125b70e6973d-Paper.pdf.

[4] Giulio Biroli, Tony Bonnaire, Valentin de Bortoli, and Marc Mézard. Dynamical regimes of diffusion models. *Nature Communications*, 15(1), November 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-54281-3. URL http://dx.doi.org/10.1038/s41467-024-54281-3.

[5] Tony Bonnaire, Raphaël Urfin, Giulio Biroli, and Marc Mezard. Why diffusion models don't memorize: The role of implicit dynamical regularization in training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=BSZqpqgqM0.

[6] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models, 2023. URL https://arxiv.org/abs/2301.13188.

[7] Sudhanshu Chanpuriya, Cameron Musco, Konstantinos Sotiropoulos, and Charalampos E. Tsourakakis. On the power of edge independent graph models. *CoRR*, abs/2111.00048, 2021. URL https://arxiv.org/abs/2111.00048.

[8] Sudhanshu Chanpuriya, Cameron N Musco, Konstantinos Sotiropoulos, and Charalampos Tsourakakis. On the role of edge dependency in graph generative models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 6325–6345. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chanpuriya24a.html.

[9] Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling, 2023. URL https://arxiv.org/abs/2305.04111.

[10] Iakovos Evdaimon, Giannis Nikolentzos, Christos Xypolopoulos, Ahmed Kammoun, Michail Chatzianastasis, Hadi Abdine, and Michalis Vazirgiannis. Neural graph generator: Feature-conditioned graph generation using latent diffusion models, 2024. URL https://arxiv.org/abs/2403.01535.

[11] Tyler Farghly, Patrick Rebeschini, George Deligiannidis, and Arnaud Doucet. Implicit regularisation in diffusion models: An algorithm-dependent generalisation analysis, 2025. URL https://arxiv.org/abs/2507.03756.

[12] Jerome Garnier-Brun, Luca Biggio, Davide Beltrame, Marc Mézard, and Luca Saglietti. Biased generalization in diffusion models, 2026. URL https://arxiv.org/abs/2603.03469.

[13] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37:133345–133385, 2024.

[14] Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models, 2025. URL https://arxiv.org/abs/2310.02664.

[15] Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces, 2023. URL https://arxiv.org/abs/2210.01549.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020. URL https://arxiv.org/abs/2006.11239.

[17] Keyue Jiang, Jiahao Cui, Xiaowen Dong, and Laura Toni. Bures-wasserstein flow matching for graph generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=5Bl5qf3fON.

[18] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10362–10383. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/jo22a.html.

[19] Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ANvmVS2Yr0.

[20] Tim Kaiser and Markus Kollmann. Diffusion models memorize in training – and generalize in inference, 2026. URL https://arxiv.org/abs/2603.13419.

[21] Najwa Laabid, Severi Rissanen, Markus Heinonen, Arno Solin, and Vikas Garg. Equivariant denoisers cannot copy graphs: Align your graph diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=onIro14tHv.

[22] Hannah Lawrence, Vasco Portilheiro, Yan Zhang, and Sékou-Oumar Kaba. Improving equivariant networks with probabilistic symmetry breaking, 2025. URL https://arxiv.org/abs/2503.21985.

[23] Mufei Li, Eleonora Kreacic, Vamsi K. Potluru, and Pan Li. Graphmaker: Can diffusion models generate large attributed graphs?, 2024. URL https://openreview.net/forum?id=ledQ1BCrwc.

[24] Andrew Melnik, Michal Ljubljanac, Cong Lu, Qi Yan, Weiming Ren, and Helge Ritter. Video diffusion models: A survey, 2024. URL https://arxiv.org/abs/2405.03150.

[25] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/nichol21a.html.

[26] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 4474–4484. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/niu20a.html.

[27] Nagham Osman, Keyue Jiang, Davide Buffelli, Xiaowen Dong, and Laura Toni. Lgdc: Latent graph diffusion via spectrum-preserving coarsening, 2025. URL https://arxiv.org/abs/2512.01190.

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

[29] Yiming QIN, Clement Vignac, and Pascal Frossard. Sparsediff: Sparse discrete diffusion for scalable graph generation. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=kuJ3lpxnVC.

[30] Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps, 2026. URL https://arxiv.org/abs/2602.12233.

[31] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77): 2539–2561, 2011. URL http://jmlr.org/papers/v12/shervashidze11a.html.

[32] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[33] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models, 2022. URL https://arxiv.org/abs/2212.03860.

[34] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020. URL https://arxiv.org/abs/1907.05600.

[35] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation, 2023. URL https://arxiv.org/abs/2209.14734.

[36] Qi Yan, Zhengyang Liang, Yang Song, Renjie Liao, and Lele Wang. Swingnn: Rethinking permutation invariance in diffusion models for graph generation, 2024. URL https://arxiv.org/abs/2307.01646.

[37] Tianyi Zhang, Zheng Wang, Jing Huang, Mohiuddin Muhammad Tasnim, and Wei Shi. A survey of diffusion based image generation models: Issues and their solutions, 2023. URL https://arxiv.org/abs/2308.13142.

A Limitations

Our experiments use a single two-block SBM, and it remains to be seen whether the same picture holds for other graph families, where the structure that emerges is less cleanly summarized by a single statistic. The dependence on node size is also only partially probed: our four sizes suggest a trend, but we do not characterize how $\tau_{\texttt{cliff}}$ and $\tau_{\texttt{struct}}$ scale with node size. Finally, we train on a single graph throughout. How the bias measured by the U-turn probe behaves as the training set grows beyond one graph is left for future work.

B Related work

Generalization vs. Memorization. Understanding why diffusion models [16, 3] generalize is interesting both theoretically and from a privacy perspective [33, 6]. Many recent works have explored a variety of avenues to explain the generalization vs memorization dichotomy in diffusion models. Kadkhodaie et al. [19] attribute the generalization ability of the diffusion models to the inductive bias while other works explore implicit regularization and network capacity [5, 4, 2, 11, 14] as possible arguments. Garnier-Brun et al. [12] show that diffusion models exhibit a biased generalization, where even though the test loss improves, the model shows unusually high affinity to generate some of the training samples. Similarly, Kaiser and Kollmann [20] show that models that generalize well at the sample level overfit the denoising objective resulting in a generalization gap between the train samples and test samples.

Memorization in graph diffusion models. Graph diffusion models have received comparatively little attention along these lines. Existing works on discrete graph diffusion and flow matching [35, 29, 9, 27, 17] target generation quality and efficient scaling to large graphs. The notion of memorization is ill-defined for GDMs, since checking whether a model has memorized a graph amounts to solving the graph isomorphism problem as each training graph has $\Omega(n!)$ graphs in its isomorphism class. This is further worsened by the message-passing GNN architectures used as denoisers, which are permutation equivariant by construction and so invariant to node ordering. Works such as [21, 36, 22] show graph diffusion models cannot exactly copy their training graphs. In the single-graph training paradigm such as EDGE [9], edge overlap [8, 7] is used to assess whether generated graphs are highly similar to the reference training graph. Inspired by the notion of biased generalization in image diffusion models [12, 20], we study the learning dynamics of single-graph diffusion models, and find that they exhibit the same signature — a widening train–test generalization gap and a biased reverse process that leads to a behavior we call *conditional memorization* where the model has the ability to recover the training graph exactly compared to a test graph. Further, our analysis reveals that the community structure only appears after the onset of this memorization which early stopping cannot mitigate.

C Details on U-turn experiment

We adopt the experimental setup of [12] to demonstrate train-data bias. Given a fixed checkpoint and a graph G on n nodes with edge set $E(G)$, for a chosen timestep t we apply the model's own forward kernel to G's clean state G_0, obtaining a corrupted state G_t, and then run the model's reverse process from G_t down to G_0, generating a reconstruction $\tilde{G}$. We measure edge recall

$$\phi_t(G) \;=\; \frac{|E(\tilde{G}) \cap E(G)|}{|E(G)|},$$

We report t on the interpretable axis $\rho = 1 - \bar{\alpha}_t$, the expected fraction of G's edges removed by step t. All runs use $T = 256$ diffusion steps with a cosine schedule. We evaluate at 12 timesteps evenly spaced over the full range, $t \in \{5, 28, 51, 74, 96, 119, 142, 165, 188, 210, 233, 255\}$, covering ρ from 0.002 to 1.

At each (checkpoint, t) we run R independent U-turns of the training graph, differing only in the forward and reverse randomness, and report the mean and the standard error across them. We repeat the identical procedure on fresh graphs drawn from the same stochastic block model never seen in training.

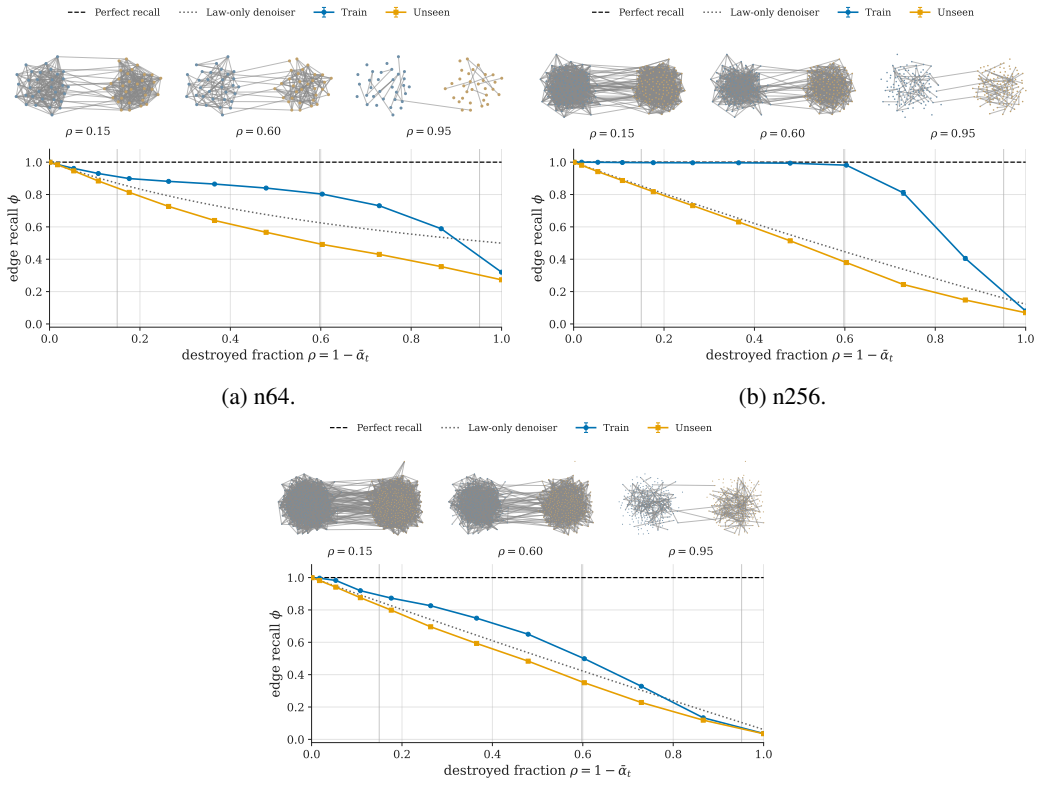
Reference law-only denoiser. First, we begin by defining a law-only denoiser that can serve as an unbiased reference. The reference denoiser knows the generating law and the corruption level, but does not have access to the graph itself. Concretely, it knows the block assignment z and the block probabilities $p_{z_i z_j}$. The posterior factorizes as $p(A^0 \mid A^t, z) = \prod_{i<j} p(A^0_{ij} \mid A^t_{ij}, z)$. The forward process only deletes edges, so an observed present pair must have been an edge, $p(A^0_{ij} = 1 \mid A^t_{ij} = 1) = 1$, while for an absent pair Bayes' rule gives

$$\beta(p, \rho) \;=\; p\big(A^0_{ij} = 1 \mid A^t_{ij} = 0\big) \;=\; \frac{\rho\, p}{\rho\, p + (1 - p)}, \tag{4}$$

In the reverse process the edges are sampled independently with probability $\beta(p, \rho)$, thus the law-only denoiser is capable of restoring the correct *number* of edges in expectation, but cannot recall the exact pairs. Averaging over true edges, a given edge either survives corruption or is deleted and predicted, giving

$$\phi_{\text{law}}(\rho) \;=\; \underbrace{(1 - \rho)}_{\text{survived}} + \underbrace{\rho\, \beta(p, \rho)}_{\text{restored}} \;=\; (1 - \rho) + \frac{p\, \rho^2}{p\, \rho + 1 - p} \tag{5}$$

Checkpoint admissibility. Late in training at $n = 512$ the sampler develops a density instability in which a growing fraction of reverse trajectories emit far too many edges; recall is inflated one-sidedly under this failure, since a denser reconstruction intersects the target more often. We therefore discard any checkpoint whose reconstructions exceed $1.15\times$ the training graph's edge density on *either* arm, evaluated at the ρ of peak separation. This removes 5 of 25 checkpoints at $n = 512$ (epochs 8399–9999, densities 1.75–$2.26\times$) and none at any other size. All reported $n = 512$ quantities are therefore truncated at epoch 8050. In Figures 5, we show the edge recall from the U-turn experiment for different node sizes.



(a) n64.

(b) n256.

(c) n512.

Figure 5: Edge recall across node sizes.

Early stopping cannot mitigate memorization. Memorization begins before the model produces structured samples, at every size. τ_{mem} is the first checkpoint at which the peak recall $\max_\rho \Delta\phi_{\texttt{edge-recall}}$ exceeds a common absolute floor of 0.02; τ_{struct} is the first checkpoint at which the structure fraction of *unconditionally* generated samples crosses the E-R floor; τ_{gen} minimises the per-node-pair loss on held-out graphs from the same law. At $n = 64$ the recall advantage is already above the floor at the earliest available checkpoint, so τ_{mem} is bounded rather than measured. At $n = 512$ the held-out loss has no interior minimum within the scored range. The ordering $\tau_{\mathrm{gen}} < \tau_{\mathrm{mem}} < \tau_{\mathrm{struct}}$ holds wherever all three are defined: early stopping at the held-out optimum yields a model that has not yet learnt the block structure, and every checkpoint that has learnt it is already past the onset of instance recall. This is shown in Table 1.

Table 1: Memorization onset happens before the emergence of community structure.

	$n = 64$	$n = 128$	$n = 256$	$n = 512$
τ_{gen} (argmin held-out loss)	500	1300	700	—
τ_{mem} (recall onset)	≤ 99	2799	3199	5599
τ_{struct} (structure in samples)	1900	3100	3600	7775
$\tau_{\mathrm{struct}} - \tau_{\mathrm{mem}}$	$\geq +1801$	$+301$	$+401$	$+2176$

D Experimental setup

D.1 Data generation

In Table 2 we show the hyperparameters used to generate our SBM training graphs. We choose the simplest two-block setting as this allows us to control the underlying generation process and it also provides the advantage of having an interpretation of what emergence of structure means.

Table 2: SBM parameters for data generation.

n	p_{in}	p_{out}	Mean degree	Edges (intra/inter)
64	0.4995	0.0161	17.00	544 (519/25)
128	0.2458	0.0081	16.19	1036 (1000/36)
256	0.1219	0.0040	16.43	2103 (2045/58)
512	0.0607	0.0020	16.12	4126 (4014/112)

D.2 Computing metrics.

In Figure 3 we plot the chance floor for edge overlap and WL kernel similarity metrics, this allows us to get a null behavior to compare against since neither marginal statistic is zero for an unmemorized model, so each is reported against its own measured floor.

Edge overlap. For labelled graphs G, H on a common vertex set we write $\mathrm{ov}(G, H) = |E(G) \cap E(H)|$, reported as a percentage of $|E(G_{\mathrm{train}})|$. The floor is the overlap that two *unrelated* graphs from the same law already share: we draw $K = 16$ independent graphs $F_1, \ldots, F_K$ from the same stochastic block model as G_{train} (disjoint seeds) and take the mean over all $\binom{K}{2} = 120$ pairs,

$$\mathrm{ov}_{\mathrm{chance}} \;=\; \binom{K}{2}^{-1} \sum_{i<j} \mathrm{ov}(F_i, F_j).$$

This quantity is model-independent, so it is computed once and reused at every checkpoint. At each checkpoint we draw 32 samples and report $\overline{\mathrm{ov}}(G_{\mathrm{gen}}, G_{\mathrm{train}})$ against it.

WL kernel similarity. We use the normalized WL subtree kernel with $h = 3$ refinement rounds. Because the model has no node features, the initial color of every vertex is its degree; colors are

refined by $c^{(r+1)}(v) = \big(c^{(r)}(v),\ \{\{c^{(r)}(u) : u \in N(v)\}\}\big)$, and $\mathrm{WL}(G, H)$ is the cosine similarity of the concatenated color histograms over rounds $0, \ldots, h$, so that $\mathrm{WL} = 1$ exactly for a relabeled copy. The floor is a *degree-preserving* null: we generate 8 rewirings $R_1, \ldots, R_8$ of G_{train} by double-edge swaps ($10\,|E|$ attempted swaps, each accepted only if it creates no self-loop or multi-edge), which preserve every vertex degree exactly while destroying the block structure. Two quantities are reported,

$$\mathrm{WL}_{\mathrm{floor}} = \tfrac{1}{8}\sum_{k} \mathrm{WL}(G_{\mathrm{train}}, R_k), \qquad \mathrm{WL}_{\mathrm{excess}} = \overline{\mathrm{WL}}(G_{\mathrm{gen}}, G_{\mathrm{train}}) - \overline{\mathrm{WL}}(G_{\mathrm{gen}}, R_\cdot),$$

the second being the part of the similarity that is specific to the training *instance* rather than to its degree sequence.

D.3 Training details

We use the experimental setup of EDGE [9] as this is more computationally efficient compared to models like DiGress [35] which scale quadratically with node size. Across all node sizes we train the models to 10K epochs. We train on a single graph. At each step we sample a timestep t, which sets the corruption level, apply the forward noising process at that level, and train the model to denoise the result back to the clean graph. Since there is only one graph, the same graph is presented many times during training; what changes from step to step is the timestep drawn and the particular corruption realized. Held-out loss is measured on fresh graphs drawn from the same block model, which the model never sees during training.

Table 3: Training hyperparameters for the single-graph experiments. All settings are shared across node sizes except where noted.

Parameter	Value
Diffusion	
Diffusion steps T	256
Noise schedule	Cosine
Forward process	Absorbing (edge deletion)
Architecture	
Backbone	Degree-guided transformer GNN
Hidden dimension	128
Attention heads (per layer)	8, 8, 8, 8, 1
Node feature	Degree
Max degree	26 / 27 / 27 / 30
Optimisation	
Optimiser	Adam
Learning rate	1×10^{-4}
Gradient clipping	1.0
Warmup	None
Batch size	32
Steps per epoch	256
Epochs	10,000 (12,200 for $n = 256$)
Seed	0

Loss windows. In Figures 6 we track the test loss on held-out test graphs to demonstrate the growing generalization gap. For this we generate 8 fresh graphs from the same SBM law as used for generating the training graph.

Structure emergence. For Figure 4 at each evaluation epoch (25) we generate 64 graphs and the clustering coefficient and mean degree is computed and averaged. After computing $\mathcal{S}(\tau)$ we use a rolling median with window 3 to smooth out the values and $\tau_{\texttt{struct}}$ is defined as the first epoch where this smoothed value reaches 0.5 and stays there for 2 consecutive epochs.

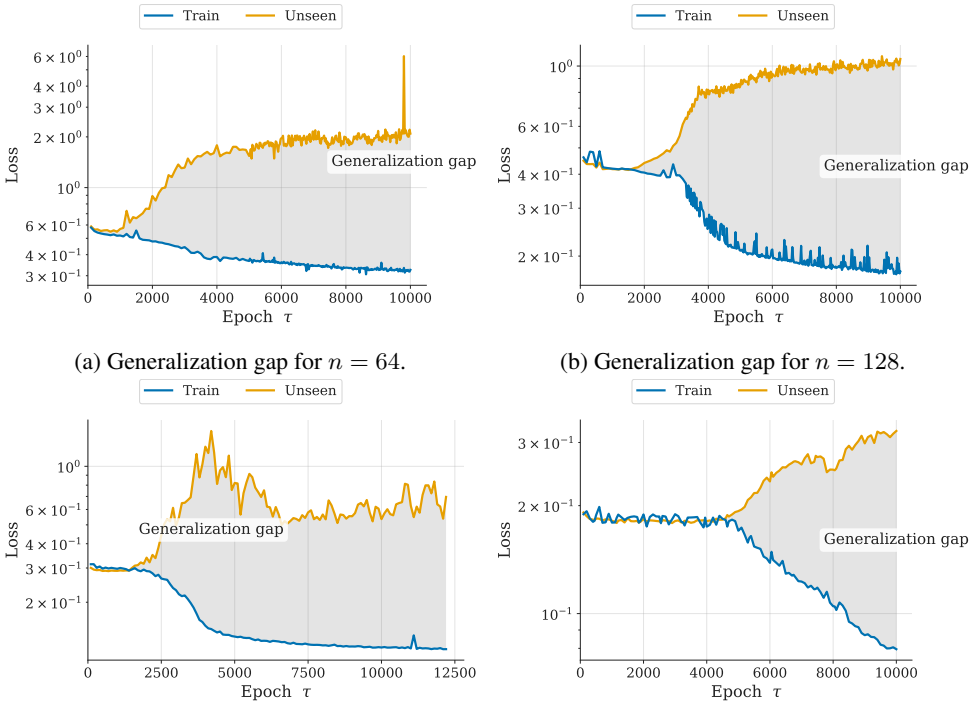
Edge recall. For the U-turn probe we average over R independent reverse trajectories per checkpoint. The details are provided in the table below, where we use smaller R sizes for increasing node size.

Table 4: U-turn experiment R draws.

n	R	Checkpoints
64	50	12
128	40	12
256	30	12
512	20	12

E Loss windows

We plot the generalization gap across node sizes.



(a) Generalization gap for $n = 64$.

(b) Generalization gap for $n = 128$.

(c) Generalization gap for $n = 256$.

(d) Generalization gap for $n = 512$.

Figure 6: Generalization gap across node sizes.