From Graph Diffusion to Graph Classification

Jia Jun Cheng Xian [1] **Sadegh Mahdavi** [1] **Renjie Liao** [1] **Oliver Schulte** [2]

Abstract

Generative models have achieved remarkable success in state-of-the-art image and text tasks. Recently, score-based diffusion models have extended their success beyond image generation, showing competitive performance with discriminative methods in image classification tasks (Zimmermann et al., 2021). However, their application to classification in the *graph* domain, which presents unique challenges such as complex topologies, remains underexplored. We show how graph diffusion models can be applied for graph classification. We find that to achieve competitive classification accuracy, score-based graph diffusion models should be trained with a novel training objective tailored for graph classification.

1. Introduction

Recent breakthroughs with generative models have enabled outstanding performance in challenging tasks in various modalities such as image generation (Saharia et al., 2022), speech generation (Le et al., 2023), and natural language processing (OpenAI et al., 2024). For text, early GPT models (Radford & Narasimhan, 2018; Radford et al., 2019) showed that generative training not only provides excellent text generation performance, but can be competitive to models discriminatively trained on the downstream task. Later improvements showed that generative training significantly outperforms all alternative approaches (Brown et al., 2020; OpenAI et al., 2024). On the image domain, Li et al. (2023) show that text-to-image diffusion models hold promise for *zero-shot classification on images* without any additional discriminative training. Furthermore, Zimmermann et al. (2021) train score-based diffusion models and show competitive performance with discriminative models in the CIFAR-10 data set. Collectively, these previous works show the promise of generative models for classification tasks.

While there has been extensive research on the applicability of generative models for classification on image and text domains, this question has remained unexplored in the *graph domain.* Therefore, it is natural to ask: *Do generative models bring the same classification competitiveness to graphs?* This work takes a step toward answering this question: we show that generative models are indeed strong baselines for graph classification. In particular, our contributions are as follows.

• We found that purely generatively trained graph diffusion models do not perform well as zero-shot classifiers using exact likelihood for inference. We therefore develop a new discriminative training objective, based on the generative ELBO likelihood approximation, that leads to strong classification performance as well as high-quality graph generation.

• Our base diffusion model is not permutation-invariant. We show that both training and inference can be improved by randomly sampling adjacency matrices from the isomorphism class of the training/test graph.

• We observe that inference time of classification using exact likelihood with score-based diffusion models is time-intensive at test time, which impedes model checkpoint selection. We propose model checkpoint selection using approximate inference rather than the exact model likelihood for better time efficiency.

2. Background

Graph Neural Network. Graph Neural Networks (GNNs) have emerged as effective architectures to process graph-structured data. A simple GNN layer operates by taking two primary inputs: a node feature matrix and an adjacency matrix A. The node feature matrix represents the attributes of each node in the graph, while the adjacency matrix A encodes the graph's structure by indicating which nodes are connected to each other. The core operation of a GNN layer involves updating the representation of each node. The new node representation is updated by aggregating the information from the neighbor of each node and its own node representation. This process mirrors the convolution oper-

[*]Equal contribution [1]Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada [2]Department of Computer Science, Simon Fraser University, Burnaby, Canada. Correspondence to: Jia Jun Cheng Xian <anthony@ece.ubc.ca>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria.

ation in the Convolutional Neural Networks (CNNs) used for image processing. The final node representations can be used for for downstream tasks such as graph classification.

Diffusion Models. Diffusion models operate through a dual process comprising forward and backward steps. The forward process involves introducing noise into the data, with the noise level denoted by $\sigma(t)$ at each time step t. Conversely, the backward process aims to denoise the data, transitioning from a noisy state to a clean one. When a diffusion stochastic process is appropriately defined and trained, it enables the sampling of a data point from pure noise using a reversed stochastic differential equation (Song et al., 2021). Subsequently, the diffusion model facilitates the simulation of the backward process for that data point, ultimately yielding a data point sampled from the clean data distribution. Moreover, Song et al. (2021) demonstrate that when provided with a data point from the ground truth data distribution, a diffusion model can accurately estimate its likelihood by solving an ordinary differential equation. This methodology has demonstrated significant success, particularly in the field of image processing.

There are two basic types of diffusion models. **Score-based** models learn to approximate the gradient of the log-probability of the denoising process (Vincent, 2011). **Denoising Diffusion Models** directly learn to denoise the given input by predicting the clean image given a noised image. Training a score network with the **denoise loss** objective function is guaranteed to almost sure match the data-generating score. The denoise loss objective function is defined as follows:

$$E_{q(x),t\sim U[0,1],p_{\sigma(t)}(\tilde{x}|x)}||D_\theta(\tilde{x},t)-x||_F^2 \tag{1}$$

where $q(x)$ is the ground truth data distribution, $\tilde{x}$ is the noisy data, $p_{\sigma(t)}(\tilde{x}|x)$ is the data distribution at time step t under noise schedule $\sigma(t)$, the notation $D_\theta(\tilde{x},t)$ denotes the denoise network and $||.||_F$ is the Frobenius norm.

3. Related Work

Diffusion graph generative model. Motivated by the success of diffusion models on image generation, several studies have been conducted to extend these models to the graph domain (Jo et al. (2022); Vignac et al. (2023)). Vignac et al. (2023) propose an equivariant discrete denoising diffusion model to generate graphs. Yan et al. (2023) introduce SwinGNN, a non-equivariant score-based diffusion model that achieves state-of-the-art results on several benchmarks. In this work, we show how to adapt the SwinGNN model for classification tasks.

Diffusion generative classifier. One line of research studied how to adapt diffusion models for classification tasks in the image domain. Zimmermann et al. (2021) add the class label as a conditioning variable to score-based diffusion models and take advantage of the fact that the exact computation of likelihood is possible for score-based models to perform image classification tasks. Li et al. (2023) leverage trained text-to-image diffusion models (such as Stable Diffusion (Rombach et al., 2021)) to perform zero-shot classification. This method relies on estimating the class-conditional likelihood by computing an evidence lower bound (ELBO) on an image and its candidate labels. Our work is related to both works as we consider diffusion models for classification. It is different in 1) the modality we study (graph vs. image/text), 2) the loss function we use to train the diffusion model.

4. Training the Generative Model for Classification

Let $\mathcal{M} := \left\{(G^{(i)}, y^{(i)}) \,|\, 1 \le i \le m\right\}$ be a training dataset of size m, where the i-th example consists of an input graph $G^{(i)}$ and a discrete label $y^{(i)} \in \{1, 2, \ldots, C\}$. Our main goal is to build a generative classifier to classify graphs. To achieve this, we introduce a novel training objective and model checkpoint selection method, which we explain in this section.

4.1. Training Objective

Consider a graph-label pair $(G, y) \in \mathcal{M}$. Let A be the adjacency matrix of the observed graph G. Using Bayes' theorem, we can derive a graph class probability from a class-conditional graph probability:

$$p(y|A) = \frac{p(A|y)\,p(y)}{\sum_i p(A|y_i)\,p(y_i)} = [\mathrm{Softmax}(\mathbf{L})]_i \tag{2}$$

$$\mathbf{L}_j := \ln p(A|y_j) + \ln p(y_j) \tag{3}$$

Equation (3) follows from Equation (2), with Softmax denoting the softmax function. Uniform class priors $p(y_j)$ can be omitted from Equations (2) and (3).

We investigate three plausible training objectives for a generative classifier model.

$$\begin{aligned} &L_{\text{DEN}}(A, y, \theta) \\ &:= E_{t\sim U[0,1],\epsilon\sim\mathcal{N}(0,1)}[||D_\theta(A+\epsilon\sigma(t), y, t) - A||_F^2] \\ &\le \ln P_\theta(A|y) \end{aligned} \tag{4}$$

$$\begin{aligned} &L_{\text{CLF}}(A, y, \theta) := \\ &\quad -\ln\left([\mathrm{Softmax}\left(L_{\text{DEN}}\left(A, 1, \theta\right), \ldots, L_{\text{DEN}}\left(A, C, \theta\right)\right)]_y\right) \end{aligned} \tag{5}$$

$$L_{\text{SUM}}(A, y, \theta) := L_{\text{DEN}}(A, y, \theta) + L_{\text{CLF}}(A, y, \theta) \tag{6}$$

Equation (4) is a variational approximation to the class-conditional log-likelihood of A. Equation (5) uses the ap-

proximate class-conditional graph log-probability to derive an approximate graph class log-probability via Equation (3). Equation (6) combines the generative and discriminative losses so that the model generates realistic graphs while also supporting classification.

Note that the classification objective L_{CLF} is a lower bound on the training set cross-entropy. We state this claim formally for binary 0/1 labels; the general case for C labels is similar. Let $A^{(j)}$ be the adjacency matrix of $G^{(j)}$. Then

$$\sum_{j=1}^{m} y^{(j)}\big(\ln P_\theta(y=1|A_j) + (1-y_j)\big(\ln(1-P_\theta(y=1|A_j))\big) \geq$$
$$\sum_{j=1}^{m} y_j \ln(\text{Softmax}(L_{\text{DEN}}(A,y_1,\theta),\ldots,L_{\text{DEN}}(A,y_k,\theta))_j) +$$
$$(1-y_j)\ln(1-\text{Softmax}(L_{\text{DEN}}(A,y_1,\theta),\ldots,L_{\text{DEN}}(A,y_k,\theta))_j)$$

4.2. Training-time Permutations

During training, we observed that our model tended to overfit rapidly, impairing its ability to generalize to the validation or test sets. Since the SwinGNN architecture is not a permutation equivariant architecture (*i.e.* , different permutations of the adjacency matrix give different outputs), one natural approach to prevent overfitting is to augment the dataset by considering random permutations of graphs. Therefore, we randomly permute the input adjacency matrices for each training batch so the batch is trained on a set of permuted adjacency matrices.

4.3. Model Checkpoint Selection

Our training strategy comprised model checkpoint selection: For each fold, we train the model for a fixed amount of computation time T, then select the best model generated through its accuracy on the validation set. However, evaluating the validation set accuracy with exact inference (see 5 below), requires solving ODEs and is time-consuming. For instance, evaluating the accuracy of a checkpoint on the validation set takes approximately 20 minutes, for one fold on the IMDB-B dataset. While it is desirable to evaluate many model checkpoints, this time cost allows us to utilize only a small number of checkpoints.

On the other hand, evaluating the variational approximation loss L_{CLF} of Equation (5) is fast, because approximate inference is much faster compared to solving ODE. For instance, evaluating the classification loss on the validation set takes approximately 0.5 second. This translates to more than 2000 times the speed up in the checkpoint selection speed compared to ODE solving. So utilizing approximate inference supports selecting from many checkpoints the one that performs the best on the validation set.

5. Classification Model

Let G be a graph to be classified with adjacency matrix A. We use the Bayes' theorem formula Equation (2) to derive class probabilities $P(y|A)$ from class-conditional probabilities $P(A|y)$. We investigate two basic methods for estimating $P(A|y)$.

Approximate Inference uses the variational approximation to the class-conditional graph log-likelihood, as in the L_{CLF} of Equation (5):

$$\ln P(A|y) \approx L_{\text{CLF}}(A,y,\theta) \tag{7}$$

Exact Inference One of the strengths of a score-based diffusion model is that exact likelihood computation is possible. In the image domain, Zimmermann et al. (2021) show how a trained class conditional score-based diffusion model with SDE can be used as zero-shot classifiers with impressive classification accuracy. Given a trained class-conditional SwinGNN D_θ, the exact likelihood computation based on ODE is as follows:

$$\begin{aligned}\log p(A|y) = \log p_T\left(A + \int_0^T \tilde{f}_\theta(A(t),t,y)\,dt\right) \\ + \int_0^T \nabla\cdot\tilde{f}_\theta(A(t),t,y)\,dt,\end{aligned} \tag{8}$$

with

$$\tilde{f}_\theta(A(t),t,y) = \frac{A(t) - D_\theta(A(t),t,y)}{t}, \tag{9}$$

where A is the adjacency matrix of the input graph, $A(t)$ is the adjacency matrix at time t of the stochastic process, y is the graph class label and $D_\theta(\tilde{A},y,t)$ is the output of the class-conditioned denoising network.

Test-time Permutations Recall that the SwinGNN model is not permutation-invariant. We can view the class probability of a *graph* as the expectation of the class probability of the adjacency matrices that represent it:
$p(G|y) \approx E_{A\in\Pi(G)}[p(A|y)]$ where $\Pi(G)$ is the isomorphism class of the graph's adjacency matrices. Our use of permutations during training can be viewed as approximating the training graph probability by sampling from the isomorphism sets of the training graphs.

At test time, we propose to utilize a similar permutation trick, and predict the class label based on several permutations of the graph. The predictions from different permutations are combined via majority vote to output the final solution. The majority vote avoids sensitivity to outlier permutations that might produce extreme probabilities. The exact working mechanism of permutation sampling is shown in Algorithm 1.

Algorithm 1 Classification Inference with Sampling

Input: Adjacency matrix A, number of permutations P
Output: Predicted label $\hat{y}$
Initialize ClassifiedList $\leftarrow$ []
for $i = 1$ **to** P **do**
  $A_{\text{i}} \leftarrow \text{randomPermute}(A)$ {Randomly permutes the graph}
  ClassifiedList $\leftarrow$ ClassifiedList + [Classify(A_i))] {Classifies the permuted graph using Eq. (**??**)}
end for
return majorityVote(ClassifiedList) {Returns the most frequent classification}

We apply permutation sampling with both approximate and exact inference.

6. Experiments

Datasets. We consider two datasets (1) a synthetic K-regular graph dataset (*i.e.* , in each graph all nodes have the same degree K). Graphs fall into a category of 4-regular and 6-regular, and the task is binary graph classification into the two classes. (2) IMDB-B (Yanardag & Vishwanathan, 2015) dataset, which consists of ego-graphs of IMDB actors/actresses. In each graph, two nodes are connected if their corresponding actors/actresses have occurred in a movie. The task is binary classification of graphs into two Genres of Action and Romance. For IMDB-B, we use the same split as Errica et al. (2019) to ensure a fair comparison with the GNN baselines.

Baselines. For IMDB-BINARY, we follow the same strategy as Errica et al. (2019), and include the discriminative baselines therein.

Our three training objectives (**??**) and two inference methods (Section 5) define a space of 6 possible designs. One of these designs is a zero-shot classifier baseline that has been previously applied to images. The combination of using L_{DEN} as a training objective with exact inference for classification is analogous to the approach of (Zimmermann et al., 2021) for image classification.

6.1. Results

Generative classifier is competitive with discriminative baselines. Table 2 shows the comparison of our method with various baselines. As shown in the Table, For IMDB-BINARY without node feature, our method achieves better average compared to the best discriminative model (GraphSAGE), while showing higher variance. When we train with node feature, our method shows both better average and lower variance then previous SOTA (GIN). This competitive performance shows promise in generative models for classification tasks.

More inference-time permutations improve test accuracy. Table 1 demonstrates how increasing the number of permutations increase accuracy on IMDB-BINARY dataset, where we achieve from 2% to more than 10% gain across the board by increasing the number of permutations from 1 to 5.

Approximate Inference is good when we have L_{DEN} in training objective. Table 1 also shows that approximate inference generally perform better within the two columns that has L_{DEN}, i.e. L_{DEN} and L_{SUM}, this may suggest that when our model still has L_{DEN} or still weighs a certain degree of sampling quality or denosing quality, using fast approximating inference will be a better choice.

7. Conclusion

Our study demonstrates the significant potential of generative models for graph classification tasks, thereby expanding the applications of generative approaches beyond the text and image domains explored so far. We presented a novel training objective that preserves generative capabilities while enhancing classification performance. Our proposed inference technique, involving graph random-permutation majority voting, was shown to improve classification accuracy. Furthermore, we addressed the challenge of utilizing exact inference for score-based diffusion models during training by utilizing a variational likelihood approximation, which allows more efficient and more powerful model selection. We hope that our study encourages further research into generative models for graph classification and inspires new methodologies that take advantage of the unique strengths of generative classifiers in different domains.

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Table 1. Ablation results on the total of 6 design space of our training and inference combination on K-Regular, IMDB-BINARY and IMDB-BINARY with features datasets. EXACT and APPROX refer to exact inference and approximate inference, respectively. Generally, larger number of permutations lead to higher accuracy, and L_{CLF} outperforms other strategies.

DATASET	INFERENCE (PERMUTATIONS)	L_{DEN}	L_{CLF}	L_{SUM}
K-REGULAR	EXACT (P=1)	83.7.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	EXACT (P=3)	88.7.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	EXACT (P=5)	92.7.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	APPROX	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
IMDB-BINARY	EXACT (P=1)	55.0 ± 4.29	70.0 ± 4.24	63.4 ± 3.32
	EXACT (P=3)	56.3 ± 4.36	69.7 ± 5.40	66.6 ± 5.12
	EXACT (P=5)	64.2 ± 3.31	**71.4 ± 5.24**	66.7± 3.80
	APPROX	66.3 ± 3.61	70.8 ± 5.10	68.0 ± 4.92
IMDB-BINARY (FEATURES)	EXACT (P=1)	59.4 ± 5.26	68.8 ± 3.43	66.2 ± 3.43
	EXACT (P=3)	63.4 ± 4.36	71.9 ± 2.70	67.7 ± 2.80
	EXACT (P=5)	65.9 ± 3.27	71.1 ± 3.18	70.0 ± 4.36
	APPROX	**72.1 ± 2.21**	66.3 ± 3.61	70.2 ± 4.26

Table 2. Comparison of our generative method with various discriminative GNN baselines on the IMDB-BINARY dataset. Numbers for other methods are recorded from Errica et al. (2019).

Method	IMDB-BINARY Accuracy	IMDBB-BINARY (features) Accuracy
DGCNN (Wang et al., 2019)	53.3 ± 5.0	69.2 ± 3.0
DiffPool (Ying et al., 2018)	68.3 ± 6.1	68.4 ± 3.3
ECC (Simonovsky & Komodakis, 2017)	67.8 ± 4.8	67.7 ± 2.8
GIN (Xu et al., 2019)	66.8 ± 3.9	71.2 ± 3.9
GraphSAGE (Hamilton et al., 2017)	69.9 ± 4.6	68.8 ± 4.5
Ours	**71.4 ± 5.2**	**72.1 ± 2.2**

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification. *ArXiv*, abs/1912.09893, 2019. URL https://api.semanticscholar.org/CorpusID:209439835.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Jo, J., Lee, S., and Hwang, S. J. Score-based generative modeling of graphs via the system of stochastic differential equations. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 10362–10383. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/jo22a.html.

Le, M., Vyas, A., Shi, B., Karrer, B., Sari, L., Moritz, R., Williamson, M., Manohar, V., Adi, Y., Mahadeokar, J., and Hsu, W.-N. Voicebox: Text-guided multilingual universal speech generation at scale. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=gzCS252hCO.

Li, A. C., Prabhudesai, M., Duggal, S., Brown, E., and Pathak, D. Your diffusion model is secretly a zero-shot classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2206–2217, October 2023.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning,

S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. Gpt-4 technical report, 2024.

Radford, A. and Narasimhan, K. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models, 2021.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. Photorealistic text-to-image diffusion models with deep language understanding. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 36479–36494. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ec795aeadae0b7d230fa35cbaf04c041-Paper-Conference.pdf.

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. doi: 10.1162/NECO_a_00142.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yan, Q., Liang, Z., Song, Y., Liao, R., and Wang, L. Swingnn: Rethinking permutation invariance in diffusion models for graph generation, 2023.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 1365–1374, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783417. URL https://doi.org/10.1145/2783258.2783417.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/e77dbaf6759253c7c6d0efc5690369c7-Paper.pdf.

Zimmermann, R. S., Schott, L., Song, Y., Dunn, B. A., and Klindt, D. A. Score-based generative classifiers. *CoRR*, abs/2110.00473, 2021.

Table 3. mean and standard deviation of 10 SwinGNN model trained using the 10-fold splits

DEGREE	ORBIT	CLUSTER	AVERAGE	SPECTRAL
0.02935 ± 0.00613	0.10185 ± 0.02141	0.22099 ± 0.07809	0.10438 ± 0.02717	0.05833 ± 0.01694

A. Appendix

A.1. Experiment Detail

For the backbone model of our experiments, we use the exact same architecture as SwinGNN (Yan et al., 2023), except for the class conditioning, we add a embedding layer for each label. The class label embedding would sum up with the noise embedding before passing to each layer.

For K-regular dataset, we generate five hundred 4-regular graphs and 6-regular graphs, we use a split ratio of 1/1/8 for test/validation/train set for this synthetic dataset, and a training of 3 hours lead to us a perfect classification accuracy.

For IMDBB, We use the same split as (Errica et al., 2019), where we also use the 10-fold CV for model selection and evaluation, however, as training generative model is more expensive and especially in our case, which we randomly permuted the training set resulting the cost of training the model more expansive, we slightly modify some of the setting, which would only make our performance sub-optimal in trade of time and resource. We did not do hyper-parameter tuning for each fold, instead, we only experiment a few hyper-parameter setting on one fold, and then use the better set of hyper-parameter for all the folds. In Erricas' work, in each fold, they randomly do a 90/10 split three times and train three times and perform early stopping to select their model to evaluate on a test set for each fold. Although in our case, it is expansive to train three times, so instead, we only do this once, which would bring us at the risk of higher variance of our result.

During inference time to calculate accuracy, we did not let the ODE solver solve the whole process from clean data to data with the maximum noisy level $\sigma(t)$ of $t = 80$. Instead, we test t=4 is good enough, there is no need for solver to solve for highly noisy data as they are less accurate and more time consuming. All experiments of this work use this setting for likelihood computation,

For approximate inference, we choose to sample 300 times noise data from each clean data, classify 300 times and choose to take majority vote of the 300 classification as our prediction. Future research could also investigate the effect of the number of sampling.

A.2. Generation

Our model is not just a good classifier, it is also a good generative model, we provide the sampling here; these sampling comes from the checkpoint that gives us the best test set accuracy. See Figure 1

We also provide the mean and standard deviation of the MMD score across 10-fold for an experiment, see Table 3

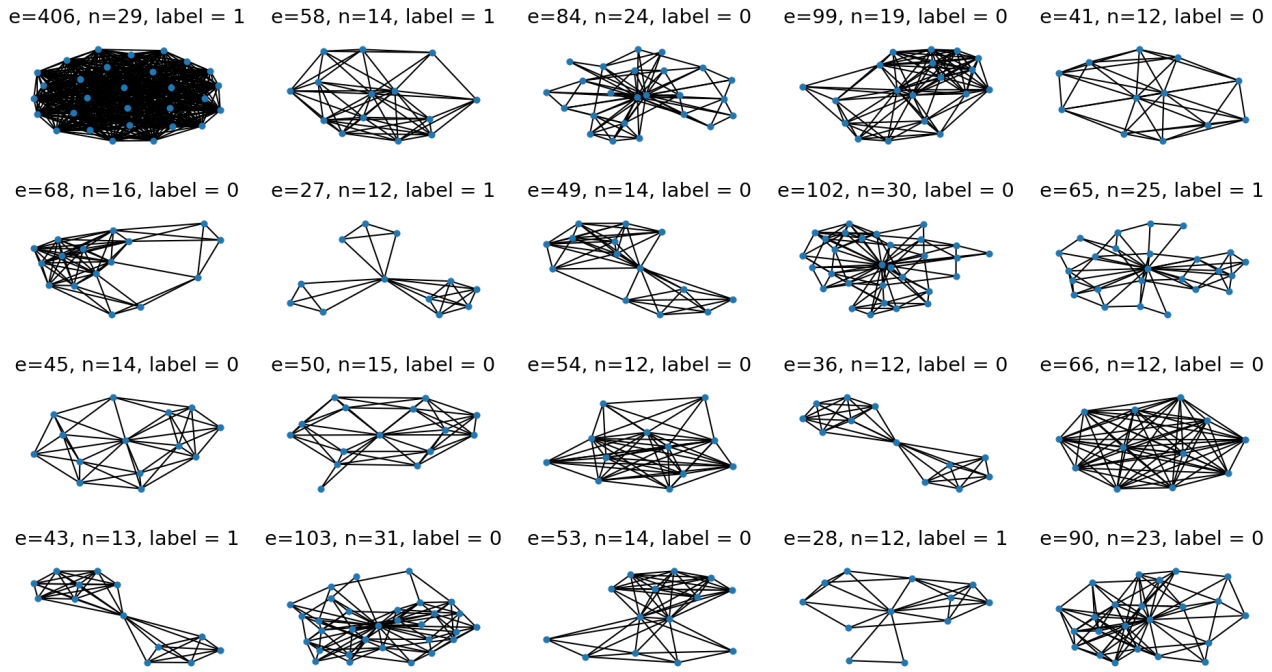

Figure 1. Sampling from out trained SwinGNN model trained with L_{SUM}, it captures some structure of the IMDB-BINARY dataset.