SPARSE TRAINING OF DISCRETE DIFFUSION MODELS FOR GRAPH GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative models for graphs often encounter scalability challenges due to the inherent need to predict interactions for every node pair. Despite the sparsity often exhibited by real-world graphs, the unpredictable sparsity pattern of their adjacency matrices, stemming from their unordered nature, leads to quadratic computational complexity. In this work, we introduce SparseDiff, a denoising diffusion model for graph generation that can exploit sparsity during its training phase. At the core of SparseDiff is a message-passing neural network tailored to predict only a subset of edges during each forward pass. When combined with a sparsity-preserving noise model, this model can efficiently work with edge list representations of graphs, paving the way for scalability to much larger structures. During the sampling phase, SparseDiff iteratively populates the adjacency matrix from its prior state, ensuring the prediction of the full graph while controlling memory utilization. Experimental results show that SparseDiff simultaneously matches state-of-the-art generation performance on both small and large graphs, highlighting the versatility of our method. [1]

1 INTRODUCTION

Random graph models have been foundational in graph generation, with a rich legacy spanning several decades (Erdős et al., 1960; Aiello et al., 2000; Barabási, 2013). However, recent interest has gravitated towards learned graph models, primarily due to their enhanced ability to represent intricate data distributions. Traditional frameworks like generative adversarial networks (De Cao & Kipf, 2018) and variational autoencoders (Simonovsky & Komodakis, 2018) predominantly addressed graphs with a maximum of 9 nodes. This limitation was somewhat alleviated with the advent of denoising diffusion models (Niu et al., 2020; Jo et al., 2022; Vignac et al., 2023a), elevating capacity to roughly 100 nodes. However, these models are still not scaled for broader applications like transportation (Rong et al., 2023) or financial system anomaly detection (Li et al., 2023).

The primary bottleneck of many generative graph models is their computational complexity. While many natural graphs are sparse, the unordered nature of graphs makes it challenging to exploit this trait. Without a predetermined sparsity pattern, models frequently make exhaustive predictions for every node pair, constraining them to a ceiling of $\sim$200 nodes (Vignac et al., 2023a). Proposed methods to circumvent this issue include imposing a node ordering (Dai et al., 2020), assembling sub-graphs (Limnios et al., 2023), generating hierarchically (Karami, 2023; Jang et al., 2023), and conditioning the generation on a sampled degree distribution (Chen et al., 2023). These methods, designed for large graphs, implicitly make assumptions about the data distribution which sometimes reflects a poor ability to model very constrained graphs such as molecules (Chen et al., 2023; Kong et al., 2023).

To address these limitations, we propose SparseDiff, a generative model for graphs that exploits sparsity in its training phase by adopting edge list representations. Unlike other scalable models, SparseDiff leverages the intrinsic sparsity of training graphs without requiring additional assumptions about the data distribution. SparseDiff defines a sparse diffusion model that comprises three primary components: 1). A noise model designed to retain sparsity throughout the diffusion process; 2). A loss function computed on a set of random node pairs; 3) A sparse graph transformer rooted

[1] Our code is available at https://anonymous.4open.science/r/SparseDiff-B861.

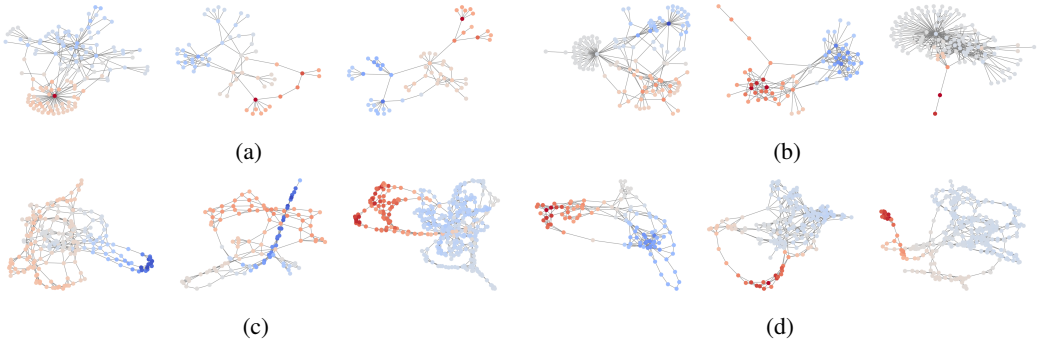

Figure 1: Samples from SparseDiff trained on large graphs. (a) Ego training set (50 to 399 nodes); (b) Generated Ego graphs; (c) Protein training set (100 to 500 nodes); (d) Generated Protein graphs.

in the message-passing framework. During the sampling process, our model iterates over all pairs of nodes and progressively builds the predicted graph.

Experiment demonstrates that, despite its simplicity, SparseDiff achieves generation performance comparable to scalable models on large graphs. Additionally, it attains results similar to state-of-the-art dense models on small molecular datasets, rendering our model suitable for graphs of varying sizes.

2 Related Work

2.1 Denoising diffusion models for graphs

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) have gained increasing popularity due to their impressive performance across generative tasks in computer vision (Dhariwal & Nichol, 2021; Ho et al., 2022; Poole et al., 2022), protein generation (Baek et al., 2021; Ingraham et al., 2022) or audio synthesis (Kong et al., 2020). Two core components define diffusion models. The first is a Markovian noise model, which iteratively corrupts a data point x to a noisy sample z^t until it conforms to a predefined prior distribution at the final step T. The second component is the denoising network, which is trained to revert the corrupted data to a less noisy state. This denoising network typically predicts the original data x or equivalently, the added noise ϵ.

After the denoising network has been trained, it can be used to sample new objects. The noise z^T is firstly sampled from a prior distribution, the denoising network is then applied at each time step to predict the less noisy distribution defined by $p_\theta(z^{t-1}|z^t) = \int_x q(z^{t-1}|z^t, x)\, dp_\theta(x)$, from which the new data z^{t-1} is sampled. While this integral is in general difficult to evaluate, two prominent frameworks allow for its efficient computation: Gaussian diffusion (Ho et al., 2020) and discrete diffusion (Austin et al., 2021).

When tailored to graph generation, initial diffusion models employed Gaussian noise on the adjacency matrices (Niu et al., 2020; Jo et al., 2022). They utilized a graph attention network to regress the added noise ϵ. Given that $\epsilon = z^t - z$, regressing the noise is, up to an affine transformation, the same as regressing the clean graph, which is a discrete object. To keep the inherent discreteness, subsequent models (Vignac et al., 2023a; Haefeli et al., 2022) leveraged discrete diffusion and achieved top-tier results. However, such models make predictions for all pairs of nodes, which leads to a quadratic space complexity and thus restricts their scalability.

2.2 Scalable Graph Generation

Efforts to enhance the scalability of diffusion models for graph generation have mainly followed two paradigms: subgraph aggregation and hierarchical refinement.

Subgraph Aggregation This approach divides larger graphs into smaller subgraphs, which are subsequently combined. Notably, SnapButton (Yang et al., 2021) enhances autoregressive models

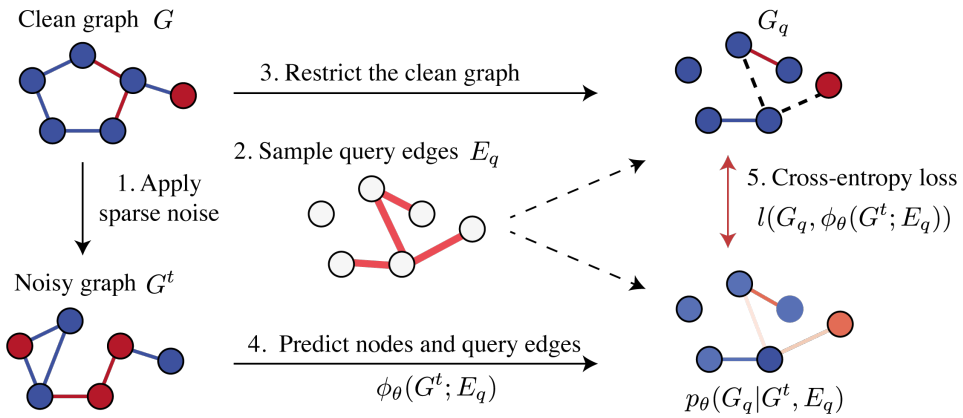

Figure 2: Overview of SparseDiff. To train a denoising neural network without considering all pairs of nodes, SparseDiff combines *i)* a noise model that preserves sparsity during diffusion; *ii)* a graph transformer ϕ_θ implemented within the message-passing framework; *iii)* a loss function computed on a subset $\boldsymbol{E}_q$ of all pairs of nodes. Together, these components allow for using edge lists and training diffusion models on significantly larger graphs than dense methods.

(Liu et al., 2018; Liao et al., 2019; Mercado et al., 2021) by merging subgraphs. Meanwhile, BiGG (Dai et al., 2020) deconstructs adjacency matrices using a binary tree data structure, gradually generating edges with an autoregressive model. One notable limitation of autoregressive models is the breaking of permutation equivariance due to node ordering dependency. To counter this, (Kong et al., 2023) proposed learning the node ordering – a task theoretically at least as hard as isomorphism testing. Separately, SaGess (Limnios et al., 2023) trains a dense DiGress model to generate subgraphs sampled from a large graph and then merges these subgraphs.

Hierarchical Refinement This approach initially generates a low-resolution graph, which undergoes successive refinements for enhanced detail (Yang et al., 2021; Karami, 2023). For instance, the HGGT model (Jang et al., 2023) employs a hierarchical K^2-tree representation. Specifically for molecular generation, fragment-based models (Jin et al., 2018; 2020; Maziarz et al., 2022) adeptly assemble compounds using pre-defined molecular fragments.

A unique approach outside these paradigms was presented by EDGE (Chen et al., 2023), who initially generate a node degree distribution $\boldsymbol{d}^0$ for the nodes, and gradually craft an adjacency matrix $\boldsymbol{A}$ conditioned on this distribution along the reverse process. Despite the universal feasibility of this factorization, the ease of learning the conditional distribution $p_\theta(\boldsymbol{A}|\boldsymbol{d}^0)$ remains uncertain, as there do not always exist undirected graphs that satisfy a given degree distribution.

Overall, scalable generation models typically either introduce a dependence on node orderings or rely heavily on the existence of a community structure in the graphs. In contrast, the SparseDiff model described in the next section aims at making no assumption besides sparsity, which results in very good performance across a wide range of graphs.

3 SparseDiff: Sparse Denoising Diffusion for Large Graph Generation

We introduce the Sparse Denoising Diffusion Model (SparseDiff), designed to bolster the scalability of discrete diffusion models for sparse graphs. Our model enables efficient training, extends the high performance of current discrete graph models to significantly larger graphs, and at the same time offers a user-friendly method for controlling GPU usage.

SparseDiff adopts the sparse graph representation, where a graph G composed of n nodes and m edges, is denoted as a triplet $(\boldsymbol{E}, \boldsymbol{X}, \boldsymbol{Y})$. Here, $\boldsymbol{E} \in \mathbb{N}^{2\times m}$ represents the edge list detailing indices of endpoints, while the node and edge attributes are encoded respectively with the one-hot format $\boldsymbol{X} \in \{0,1\}^{n\times a}$ and $\boldsymbol{Y} \in \{0,1\}^{m\times b}$. As illustrated in Fig. 2, our approach incorporates three core

Algorithm 1: Sparse Training at Step t with predefined edge fraction λ

1: Sample the sparse noisy graph $G^t = (\boldsymbol{E}^t, \boldsymbol{X}^t, \boldsymbol{Y}^t)$ with the sparse noise model (3.1);
2: Sample random query edges $\boldsymbol{E}_q$ of size λn^2 from all node pairs (3.2);
3: Construct the sparse computational graph G_c whose edge list $\boldsymbol{E}_c$ contains $\boldsymbol{E}^t \cup \boldsymbol{E}_q$ (3.3.1);
4: Feed G_c into the message-passing network ϕ_θ and predict $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}_q$ (3.3.2);
5: Perform sparse loss calculation on $(\boldsymbol{X}, \hat{\boldsymbol{X}})$ and $(\boldsymbol{Y}_q, \hat{\boldsymbol{Y}}_q)$ (3.2);

components to enable training in a space-efficient manner, and Algorithm 1 provides a step-by-step description of the training algorithm, referring to their respective explanations. Subsequent sections will elaborate on each of these three components.

3.1 Sparsity-preserving noise model

To begin with, a sparse graph diffusion model involves a noise model designed to maintain sparsity throughout the diffusion process. This implies that the number of edges of the noisy graph G^t must match that of G to sustain consistent complexity throughout the diffusion. At the same time, the computational complexity of applying noise to a graph should remain sub-quadratic as well.

Marginal Transition The first requirement necessitates the adoption of a discrete diffusion model. In this framework, instead of using the noisy distribution $q(G^t|G)$ itself as the noisy data, we sample G^t from $q(G^t|G)$ to keep its discrete nature. Precisely, the noisy distribution can be obtained through $q(G^t|G) = (\boldsymbol{X}\bar{\boldsymbol{Q}}^t_{\boldsymbol{X}}, \boldsymbol{Y}\bar{\boldsymbol{Q}}^t_{\boldsymbol{Y}})$, where $\bar{\boldsymbol{Q}}^t = \boldsymbol{Q}^1\boldsymbol{Q}^2 \dots \boldsymbol{Q}^t$ for $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively, and $\boldsymbol{Q}^t$ represents the Markov transition matrix from step $t-1$ to step t. While there exist different Markov transition matrices such as uniform transitions, absorbing transitions, and marginal transitions, only the last one is supported theoretically (Ingraham et al., 2022; Vignac et al., 2023a) and maintains the same level of sparsity (i.e. edge numbers) through diffusion. In the marginal transition model, the probability of transitioning to a state is proportional to the marginal probability of that state in the data. In the context of sparse graphs, this means that jumping to the state "no edge" will be very likely, as it is the dominant label in the data. Formally, if $\boldsymbol{p}_{\boldsymbol{X}}$ and $\boldsymbol{p}_{\boldsymbol{E}}$ are the marginal distribution of node and edge types and $\boldsymbol{p}'$ is the transpose of $\boldsymbol{p}$, and β^t controls the noise intensity at step t and $\alpha^t = 1 - \beta^t$, the marginal transition matrices for nodes and edges are defined by: $\boldsymbol{Q}^t_{\boldsymbol{X}} = \alpha^t \boldsymbol{I} + \beta^t \mathbf{1}_a \boldsymbol{p}_{\boldsymbol{X}}$ and $\boldsymbol{Q}^t_{\boldsymbol{Y}} = \alpha^t \boldsymbol{I} + \beta^t \mathbf{1}_b \boldsymbol{p}'_{\boldsymbol{Y}}$.

Theoretical Analysis regarding Sparsity We note that our choice of noise model does not guarantee that the noisy graph is always sparse. However, it is the case with high probability as stated by the following lemma, which is an application of Desolneux et al. (2008) (cf. Appendix B). This lemma shows that, in large and sparse graphs, the probability that the fraction of edges r in the noisy graph is higher than the actual existing edge ratio k decreases exponentially with the graph size. For instance, for k small and $r = 2k$, this probability can be written with $c_1 e^{-c_2 n^2 k}$ for two constants c_1 and c_2, which is considerably small when n is large.

Lemma 3.1. *(High-probability bound on the sparsity of the noisy graph)*
Consider a graph with n nodes and m edges. If the edge ratio given by $m/(n(n-1)/2)$ is denoted as r, and the number of edges in the noisy graph G^t sampled from the marginal transition model is given by m_t. Then, for n sufficiently large and $k < 1/4$, for any $k < r < 1$, we have:

$$log(\mathrm{P}[\frac{2m_t}{n(n-1)} \geq r]) \sim -\frac{n(n-1)}{2}(rlog\frac{r}{k} + (1-k)log\frac{1-r}{1-k}) \tag{1}$$

Sparse Sampling of the Noise Model The second quadratic component arises from noise sampling. Although the marginal transition keeps the sparsity of G^t, to obtain the noisy graph, standard discrete diffusion models simply compute transition probabilities using a product $\boldsymbol{Y}\boldsymbol{Q}^t_{\boldsymbol{Y}} \in \mathbb{R}^{n \times n \times b}$ and sample from it. This multiplication based on the dense edge representation is however no longer compatible with our sparse edges encoded in $\boldsymbol{Y} \in \{0,1\}^{m \times b}$ and $\boldsymbol{E} \in \mathbb{N}^{2 \times m}$. To enable sparse sampling, we adopt a three-step approach to sample noisy graphs without using dense tensors.

1. We consider "existing edge" types by computing $\boldsymbol{Y}\boldsymbol{Q}^t_{\boldsymbol{Y}} \in \mathbb{R}^{m \times b}$ for edges in the edge list $\boldsymbol{E}$ of the clean graph G and sample their new label from this categorical distribution;

2. We consider the "no edge" type, and determine the number of new edges to add to the list $\boldsymbol{E}^t$. This number follows a binomial distribution $\mathcal{B}(\bar{m}_t, k)$, which takes $\bar{m}_t$ draws from all edges of "no edge" type with the probability k of turning into an "existing edge" state. Here, $\bar{m}_t = n(n-1)/2 - m_t$, and $k = 1 - \boldsymbol{Q}^t[0, 0]$.
3. We sample positions for these new edges uniformly from non-occupied positions in the adjacency matrix of G. As elaborated in Appendix A.2, an efficient and highly intricate algorithm has been devised for sampling with the sparse edge list $\boldsymbol{E}$ instead of the quadratic adjacency matrix $\boldsymbol{A}$, which ensures the sub-quadratic space complexity in computation.

3.2 Sparsity-Preserving Loss Function

In discrete denoising diffusion for graphs such as (Vignac et al., 2023a; Haefeli et al., 2022), a neural network is trained to predict the clean graph, i.e., the class of each node and each pair, which introduces another quadratic component to the model. To avoid this, one alternative method is to make predictions for a subset of edges instead of for all node pairs. For this purpose, SparseDiff introduces a parameter "edge fraction" λ which corresponds to a fraction of pairs that are sampled uniformly in each forward pass. Such sampled edges are called "query edges", and denoted by $\boldsymbol{E}_q$. In our implementation, λ was treated as a constant and chosen to balance GPU usage. The computational complexity of SparseDiff is up-bounded by $O(m + \lambda n^2)$, as the model needs to process both the noisy graph and the query edges. However, by choosing $\lambda = O(m/n^2)$, it could result in a $O(m)$ complexity as opposed to $O(n^2)$ for DiGress.

Precisely, if the constant c (set to 5 in experiments) is used to balance the importance of nodes and edges, the network is trained by minimizing the cross-entropy loss between the predicted distribution and the clean graph, which is simply a sum over nodes and query edges:

$$l(\hat{p}^G, G) = \sum_{1 \leq i \leq n} \text{cross-entropy}(x_i, \hat{p}_i^X) + \frac{c}{\lambda} \sum_{(i,j) \in E_q} \text{cross-entropy}(y_{ij}, \hat{p}_{ij}^Y),$$

3.3 Sparse Message-Passing Transformer

The final component of the Sparse Diffusion Model is a memory-efficient graph neural network. In previous diffusion models for graphs, the main complexity bottleneck lay in the need to encode features for all pairs of nodes, leading to a computation complexity that scaled as $O(l\, n^2\, d_e)$, where l is the number of layers and d_e the dimensionality of edge activations. To address this issue, it is necessary to avoid learning embeddings for all pairs of nodes. Fortunately, as our noisy graphs are sparse, edge list representations can be leveraged. These representations can be efficiently used within message-passing neural networks (MPNNs) architectures (Scarselli et al., 2008; Gilmer et al., 2017) through the use of specialized libraries such as Pytorch Geometric (Fey & Lenssen, 2019) or the Deep Graph Library (Wang, 2019).

3.3.1 Edge Embedding Module Design

The denoising network of SparseDiff has to deal with two simultaneous constraints. First, it needs to make predictions for the query edges $\boldsymbol{E}_q$. Second, in contrast to previous diffusion models, it cannot compute activations for all pairs of nodes. Although not possible within most message-passing architectures, the idea of predicting edge labels according to node features is however common in the context of *link prediction for knowledge graphs* (Zhang & Chen, 2018; Chamberlain et al., 2022; Boschin, 2023). We therefore first consider a link prediction approach to our problem.

A first approach: graph learning as a link prediction problem Instead of storing activations for pairs of edges, link prediction models typically only store representations for the nodes. In this framework, a graph neural network that learns embeddings for each node is coupled with an auxiliary module that predicts edges. In the simplest setting, this module simply computes the cosine similarity between node representations to predict the probability of being connected. While this approach is very memory efficient, we find that it has a slow convergence and poor overall performance in practice (cf. ablations in Appendix E.6). In particular, it fails to replicate the performance of dense denoising diffusion models, even on small graph datasets. This suggests that reconstructing the graph from node representations only, which is theoretically proved to be possible (Maehara & Rödl, 1990), might be hard to achieve in practice.

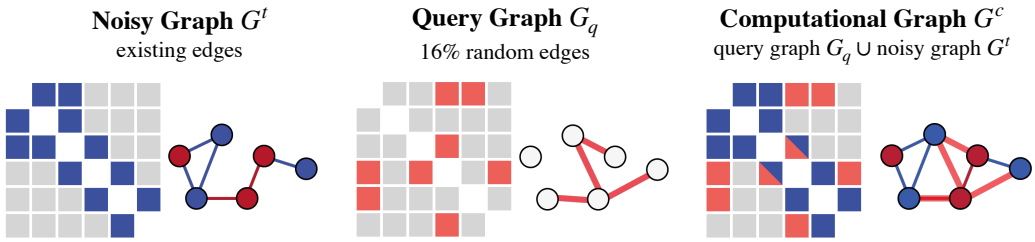

Figure 3: Definition of the noisy graph G^t, the query graph G_q, and the computational graph G_c, with an edge proportion $\lambda = 0.16$. The noisy graph G^t is the result of our sparsity-preserving noising process, the query graph G_q consists of a fraction λ of randomly chosen edges, and the computational graph G_c is the union of the noisy and query graphs. Self-loops are not included in the calculation.

Second approach: learning representations for edges Based on the previous findings, we consider an approach that stores activations for pairs of nodes. The list of pairs for which we store activations will define our *computational graph* G_c, i.e., the graph that is used as input to the message-passing architecture. This graph contains all nodes with their noisy features $\boldsymbol{X}^t$, as well as a list of edges denoted by $\boldsymbol{E}_c$. In order to bypass the need for an edge prediction module and obtain edge features directly, the computational graph should contain the list of query edges sampled previously, i.e., $\boldsymbol{E}_q \in \boldsymbol{E}_c$. Furthermore, this graph should contain all topological information about the noisy graph, which imposes $\boldsymbol{E}^t \in \boldsymbol{E}_c$. Under these two constraints, we define the computational graph as the union of the noisy and query edge lists. Since these two graphs are sparse, the computational graph used in our message-passing architecture is guaranteed to be sparse as well.

An additional advantage of employing a computational graph that encompasses not only G^t but also randomly sampled "no type" query edges is that it serves as a graph rewiring mechanism. Such edges that do not exist in the input graph G^t provide the message-passing network with shortcuts, which is known to help the propagation of information and alleviate over-squashing issues (Alon & Yahav, 2020; Topping et al., 2021; Di Giovanni et al., 2023).

3.3.2 Model Architecture

Our denoising network architecture builds upon the message-passing transformer architecture developed by Shi et al. (2020). These layers integrate the graph attention mechanism (Veličković et al., 2017) within a transformer architecture by adding normalization and feed-forward layers. In contrast to previous architectures used in denoising networks for graphs such as (Jo et al., 2022) or (Haefeli et al., 2022), the graph attention mechanism is based on edge list representations and is thus able to leverage the sparsity of graphs. We however incorporate several elements of (Vignac et al., 2023a) to improve performance. Similarly to their model, we internally manipulate graph-level features (such as the time information), as they are able to store information more compactly. Features for the nodes, edges, and graphs all depend on each other thanks to the use of PNA pooling layers (Corso et al., 2020) and FiLM layers (Perez et al., 2018) (cf. ablations in Appendix C).

Finally, we use a set of features as structural and positional encodings. These features, which include information about the graph Laplacian and cycle counts, are detailed in Appendix D. As highlighted in (Vignac et al., 2023a), these features can only be computed when the noisy graphs are sparse, which is an important benefit of discrete diffusion models. We note that not all these encodings can be computed in sub-quadratic time. However, in practice, we find that this is not an issue as these features are not back-propagated. For instance, on graphs with 500 nodes, computing these features is five times faster than the forward pass itself. Nevertheless, on even larger graphs, it might be beneficial to exclude these encodings for more efficient computation.

3.4 Iterative Sparse Sampling

Once the denoising network has been trained, it can be used to sample new graphs. Similar to other graph diffusion models, we first sample a node number n and keep it constant during diffusion. Then, from the prior distribution $G^T \sim \prod_{i=1}^{n} \mathrm{Cate}(\boldsymbol{p}_{\boldsymbol{X}}) \times \prod_{1 \leq i < j \leq n} \mathrm{Cate}(\boldsymbol{p}_{\boldsymbol{Y}})$, we sample a ran-

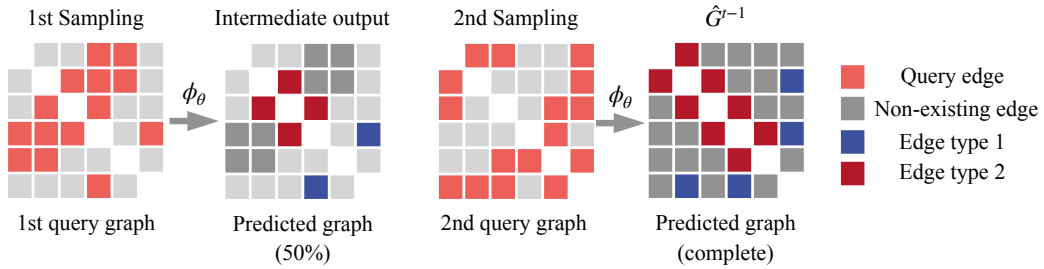

Figure 4: Visualization of the iterative sampling process, with a query edge proportion λ of 50%. In the figure, SparseDiff iterates twice to cover all node pairs, with each iteration involving the sampling of new edge types for 50% edges to populate the adjacency matrix.

dom graph based on the sparse sampling algorithm (c.f. 3.1), where $\boldsymbol{p_X}$ and $\boldsymbol{p_Y}$ are the marginal probabilities of each class in the data and $\mathrm{Cate}(\boldsymbol{p})$ denotes their corresponding categorical distribution. Note that, in the particular case of unattributed graphs, sampling from this prior distribution amounts to sampling an Erdos-Renyi graph.

After the graph G^T has been sampled, the denoising network can be applied recursively. However, the full graph cannot be predicted at once, as this would first require quadratic memory, and would also create a distribution shift: as the message-passing network has been trained on sparse computational graphs G_c^t, dense graphs should not be used at inference time. We therefore use an iterative procedure, illustrated in Fig. 4, to cover all node pairs of G^{t-1}. We first consider all $n(n-1)/2$ indices representing pairs of nodes and randomly permute them. We cut the resulting array into equal-sized chunks that represent the query edge list $\boldsymbol{E}_q$ at each iteration. We then iterate over these blocks, adding new edges into the edge list $\boldsymbol{E}^{t-1}$, while keeping the noisy graph G^t fixed. This procedure results in $\lceil \frac{1}{\lambda} \rceil$ calls to the denoising diffusion model at each diffusion step.

We note that our approach introduces higher time complexity during sampling. However, as discussed before, it is challenging to avoid quadratic predictions without assuming specific characteristics of the data distribution. Moreover, while the high space complexity makes training with traditional diffusion models almost impossible, increased sampling time does not emerge as the primary bottleneck for practice usage.

4 Experiments

We conduct experiments to present the capability of SparseDiff across a wide range of graphs. SparseDiff matches state-of-the-art performance on datasets of small molecules (QM9, Moses), while being simultaneously very competitive on datasets of larger graphs (Planar, SBM, Protein, Ego). We compare the performance of SparseDiff to GraphNVP (Madhawa et al., 2019), DiGress (Vignac et al., 2023a), Spectre (Martinkus et al., 2022), GraphRNN (You et al., 2018), GG-GAN (Krawczuk et al., 2017), JDSS (Jo et al., 2022), as well as several scalable models: HiGen (Karami, 2023), EDGE (Chen et al., 2023), BiGG (Dai et al., 2020) and HGGT (Jang et al., 2023), and GraphARM (Kong et al., 2023). Considering that certain datasets are conventionally assessed based on a limited number of generated samples, the variance in results for each sampling can be important. To promote a more convincing comparison, we present the average and standard deviation across 5 runs for each metric. If a result falls within the bounds of the standard deviation of our results, we also consider them comparable for fair assessment.

4.1 Molecule generation

Since our method considers all node pairs as dense models when $\lambda = 1$, it should match their performance on datasets of small graphs. We verify this capability on the QM9, and Moses molecular datasets used in DiGress (Vignac et al., 2023a). The QM9 dataset (Wu et al., 2018) that contains molecules with up to 9 heavy atoms can either be treated with implicit or explicit hydrogens. The Moses benchmark (Polykovskiy et al., 2020), based on ZINC Clean Leads, contains drug-sized molecules and features many tools to assess the model performance. Since QM9 contains charged

Table 1: Molecule generation on QM9 with implicit hydrogens (mean and std over 5 samplings). For a fair comparison, DiGress was modified to handle formal charges and benchmarked. While there is no major benefit to using sparsity on small graph, SparseDiff is very competitive, while other scalable models have a poor FCD metric, indicating that they do not correctly model the data.

Class	Method	Valid (%) ↑	Unique (%) ↑	Connected (%) ↑	FCD ↓
Dense	SPECTRE	87.3	35.7	-	-
	GraphNVP	83.1	99.2	-	-
	GDSS	95.7	98.5	-	-
	DiGress	99.2	95.9	99.5	0.15
	DiGress + charges	99.3±.0	95.9±.2	99.4±.2	0.15±.01
Sparse	GraphARM	90.3	-	-	1.22
	EDGE	99.1	**100**	-	0.46
	HGGT	99.2	95.7	-	0.40
	SparseDiff(ours)	**99.6**±.04	99.7±.01	**99.7**±.02	**0.11**±.01

Table 2: Unconditional generation on the Stochastic Block Model (SBM) and Planar datasets. A SBM graph is valid if it passes a statistical test for the stochastic block model, while a planar graph is valid if it is planar and connected. Results are presented in the form of ratios: $\mathrm{MMD}(\text{generated}, \text{test})^2 / \mathrm{MMD}(\text{train}, \text{test})^2$. VUN: valid, unique & novel graphs.

Dataset	Stochastic block model				Planar			
Model	Deg.↓	Clust.↓	Orbit↓	V.U.N.↑	Deg. ↓	Clust. ↓	Orbit↓	V.U.N.↑
GraphRNN	6.9	1.7	3.1	5%	24.5	9.0	2508	0%
GRAN	14.1	1.7	2.1	25%	3.5	1.4	1.8	0%
GG-GAN	4.4	2.1	2.3	25%	–	–	–	–
SPECTRE	1.9	1.6	1.6	53%	2.5	2.5	2.4	25%
DiGress	**1.6**	**1.5**	1.7	**74%**	**1.4**	**1.2**	**1.7**	75%
HiGen	2.4	**1.5**	**1.4**	–	–	–	–	–
SparseDiff	**2.0**±1.6	**1.5**±.0	**1.4**±.1	56%±8.5	3.6±1.7	**1.4**±.4	3.4±1.2	**88%**±7

atoms, we incorporate formal charges as an additional discrete node feature that is learned during diffusion, similarly to (Vignac et al., 2023b). For a fair comparison, we also apply this improvement to DiGress.

For the QM9 dataset, we assess performance by checking the proportion of connected graphs, the molecular validity of the largest connected component (measured by the success of RDKit sanitization), and the uniqueness of over 10,000 molecules. Additionally, we use the Frechet ChemNet Distance (FCD) (Preuer et al., 2018) which measures the similarity between sets of molecules using a pretrained neural network.

In Table 1, we observe that SparseDiff overall achieves the best performance on QM9 with implicit hydrogens except on uniqueness. In particular, it clearly outperforms other scalable methods on the FCD metric, showing that such methods are not well suited to small and very structured graphs. Results for QM9 with explicit hydrogens and the MOSES dataset are presented in Tables 6 (Appendix E.3), and Table 7 (Appendix E.4). We find that SparseDiff compares similarly to the DiGress model. This result confirms further our performance on small and not highly sparse datasets.

4.2 Large graph generation

We also evaluate our model on datasets of graphs with increasing size. First, we test our model's ability to generate graphs without edge crossings through a dataset of planar graphs (with 64 nodes per graph). Then, we consider a dataset drawn from the Stochastic Block Model (SBM) (Martinkus et al., 2022) with 2 to 5 communities. Its graphs contain up to 200 nodes, which is the largest size used in dense diffusion models such as DiGress (Vignac et al., 2023a). Finally, we use the Ego (Sen et al., 2008) and Protein (Dobson & Doig, 2003) datasets that feature graphs with up to 500 nodes. Ego is sourced from the CiteSeer (Giles et al., 1998) dataset and captures citation relationships,

Table 3: Unconditional generation on graphs with up to 500 nodes. On such graphs, dense models such as DiGress clearly fail, whereas SparseDiff presents competitive performance on most metrics. Results are presented in the form of ratios: $\mathrm{MMD}(\text{generated}, \text{test})^2 / \mathrm{MMD}(\text{train}, \text{test})^2$.

Dataset	Class	Model	Degree ↓	Clustering ↓	Orbit ↓	Spectre ↓	RBF ↓
Protein	Dense	GRAN	6.7	7.1	40.6	5.7	–
		DiGress	18.4±2.3	14.8±2.1	16.0±5.5	5.9±1.1	5.2
	Sparse	DRuM	6.3	9.7	10.8	3.3	–
		BiGG	**0.3**	3.7	**7.1**	5.0	–
		HiGen	4.0	6.4	7.3	2.8	–
		SparseDiff	10.3±1.1	**3.4**±.2	15.1±2.0	**1.6**±.07	**3.4**±.7
Ego	Dense	DiGress	354	0.9	100	–	5.3
	Sparse	EDGE	290	17.3	4.3	–	4.0
		HiGen	236	**0.3**	3.2	–	**3.7**
		SparseDiff	**9.5**±3.5	5.4±.2	**2.5**±.1	**3.6**±1.1	**3.9**±1.0

Table 4: Convergence comparison after being trained for different time with Ego dataset.

Training time	2 days				4 days			
Metrics	Deg.↓	Orbit↓	Clust.↓	Spec.↓	Deg.↓	Orbit↓	Clust.↓	Spec.↓
DiGress	0.042	0.185	0.208	0.013	0.033	0.144	0.216	0.011
SparseDiff	**0.004**	**0.053**	**0.069**	**0.007**	**0.002**	**0.036**	**0.059**	**0.004**

while Protein represents amino acids connected when they are within 6 Angstroms of each other. Statistics for these datasets can be found in Appendix E.2.

For evaluation, we first include MMD metrics, which are commonly used in graph generation tasks. As MMD metrics usually produce small values that are challenging to compare directly, we report metrics divided by $\mathrm{MMD}(\text{training}, \text{test})^2$, and provide our raw results in Appendix E.5. We also use the RBF MMD metric defined in Thompson et al. (2022) to measure the diversity and fidelity of generated graphs using a randomly parametrized GNN. Besides, we especially report the validity of generated graphs for the SBM dataset, which is the fraction of graphs that pass a statistical test for the stochastic block model. For the Planar dataset, validity corresponds to the fraction of graphs that are planar and connected.

Results are presented in Tables 2 and 4. Although dense models demonstrate excellent performance on mid-sized graphs like SBM and planar graphs, they struggle with larger graphs. This is due to the necessity of using a small batch size (e.g., 2 on a 32GB GPU) for such graphs, resulting in slow training and poor convergence. In contrast, SparseDiff not only matches previous dense and sparse models on mid-sized datasets but also remains competitive with scalable models across various metrics on large datasets.

Efficiency Analysis To showcase the empirical efficiency improvement during training of SparseDiff over its limit case DiGress, Table 4 illustrates the significantly faster convergence of SparseDiff on the Ego dataset. More precisely, SparseDiff achieved superior results compared to a DiGress model trained for 4 days, even after only 2 days of training.

5 Conclusion

In this study, we introduce SparseDiff, a scalable discrete denoising diffusion model for graph generation. SparseDiff provides high controllability over GPU usage and permits the use of edge list representations by predicting only a subset of edges at once. Experimental results demonstrate that SparseDiff exhibits high performance across all graph sizes, whereas other scalable methods tend to perform poorly on small, structured graphs. SparseDiff enhances the capabilities of discrete diffusion models to process larger datasets, thereby broadening its applicability, including tasks such as generating large biological molecules and community graphs, among others.

REFERENCES

William Aiello, Fan Chung, and Linyuan Lu. A random graph model for massive graphs. In *ACM Symposium on Theory of computing*, 2000. 1

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020. 6

Jacob Austin, Daniel Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, 2021. 2

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021. 2

Albert-László Barabási. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987), 2013. 1

Armand Boschin. *Machine learning techniques for automatic knowledge graph completion*. PhD thesis, Institut polytechnique de Paris, 2023. 5

Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. *arXiv preprint arXiv:2209.15486*, 2022. 5

Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. *arXiv preprint arXiv:2305.04111*, 2023. 1, 3, 7

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In *Advances in neural information processing systems*, volume 33, 2020. 15

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 2020. 6

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *International Conference on Machine Learning*. PMLR, 2020. 1, 3, 7

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018. 1

Agnés Desolneux, Lionel Moisan, and Jean-Michel Morel. Estimating the binomial tail. *From Gestalt Theory to Image Analysis: A Probabilistic Approach*, pp. 47–63, 2008. 4, 15

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, 2021. 2

Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M. Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. 6

Paul D. Dobson and Andrew J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 2003. 8

Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 1960. 1

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019. 5

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*. Association for Computing Machinery, 1998. 8

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 2017. 5

Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *arXiv preprint arXiv:2210.01549*, 2022. 2, 5, 6

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020. 2

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 2

John Ingraham, Max Baranov, Zak Costello, Vincent Frappier, Ahmed Ismail, Shan Tie, Wujie Wang, Vincent Xue, Fritz Obermeyer, Andrew Beam, et al. Illuminating protein space with a programmable generative model. *bioRxiv*, 2022. 2, 4

Yunhui Jang, Dongwoo Kim, and Sungsoo Ahn. Hierarchical graph generation with k2-trees. *arXiv preprint arXiv:2305.19125*, 2023. 1, 3, 7

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*. PMLR, 2018. 3

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International Conference on Machine Learning*. PMLR, 2020. 3

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. *arXiv preprint arXiv:2202.02514*, 2022. 1, 2, 6, 7

Mahdi Karami. Higen: Hierarchical graph generative networks. *arXiv preprint arXiv:2305.19337*, 2023. 1, 3, 7

Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B. Aditya Prakash, and Chao Zhang. Autoregressive diffusion model for graph generation, 2023. URL https://openreview.net/forum?id=98J48HZXxd5. 1, 3, 7

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020. 2

Igor Krawczuk, Pedro Abranches, Andreas Loukas, and Volkan Cevher. Gg-gan: A geometric graph generative adversarial network. *arXiv preprint arXiv:1711.0826*, 2017. 7

Xujia Li, Yuan Li, Xueying Mo, Hebing Xiao, Yanyan Shen, and Lei Chen. Diga: Guided diffusion model for graph recovery in anti-money laundering. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023. 1

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. In *NeurIPS*, 2019. 3

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. 1, 3

Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31, 2018. 3

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019. 7

Hiroshi Maehara and Vojtech Rödl. On the dimension to represent a graph by a unit distance graph. In *Graphs and Combinatorics*. Springer, 1990. 5

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. *arXiv preprint arXiv:2204.01613*, 2022. 7, 8, 16

Krzysztof Maziarz, Henry Richard Jackson-Flux, Pashmina Cameron, Finton Sirockin, Nadine Schneider, Nikolaus Stiefl, Marwin Segler, and Marc Brockschmidt. Learning to extend molecular scaffolds with structural motifs. In *International Conference on Learning Representations (ICLR)*, 2022. 3

Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design. *Machine Learning: Science and Technology*, 2021. 3

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020. 1, 2

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 6

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. In *Frontiers in pharmacology*. Frontiers Media SA, 2020. 7

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 2

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Gunter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741, 2018. 8

Can Rong, Jingtao Ding, Zhicheng Liu, and Yong Li. City-wide origin-destination matrix generation via graph denoising diffusion. *arXiv preprint arXiv:2306.04873*, 2023. 1

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. In *IEEE transactions on neural networks*. IEEE, 2008. 5

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. In *The AI Magazine*, 2008. 8

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020. 6

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*. Springer, 2018. 1

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015. 2

Rylee Thompson, Boris Knyazev, Elahe Ghalebi, Jungtaek Kim, and Graham W Taylor. On evaluation metrics for graph generative models. *arXiv preprint arXiv:2201.09871*, 2022. 9, 17

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021. 6

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 6

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023a. 1, 2, 4, 5, 6, 7, 8

Clément Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. Midi: Mixed graph and 3d denoising diffusion for molecule generation. In *ECML/PKDD*, 2023b. 8, 16

Minjie Yu Wang. Deep graph library: Towards efficient and scalable deep learning on graphs. In *ICLR workshop on representation learning on graphs and manifolds*, 2019. 5

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. In *Chem. Sci.* The Royal Society of Chemistry, 2018. 7

Shuai Yang, Xipeng Shen, and Seung-Hwan Lim. Revisit the scalability of deep auto-regressive models for graph generation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021. 2, 3

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*. PMLR, 2018. 7

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in neural information processing systems*, volume 31, 2018. 5