DMol: A Highly Efficient and Chemical Motif-Preserving Molecule Generation Platform

Peizhi Niu[1], Yu-Hsiang Wang[1], Vishal Rana[1],
Chetan Rupakheti[2], Abhishek Pandey[2], Olgica Milenkovic[1]
[1]University of Illinois Urbana-Champaign
[2]AbbVie
{peizhin2, yw121, vishalr, milenkov}@illinois.edu
{chetan.rupakheti, abhishek.pandey}@abbvie.com

Abstract

We introduce a new graph diffusion model for small drug molecule generation which simultaneously offers a 10-fold reduction in the number of diffusion steps when compared to existing methods, preservation of small molecule graph motifs via motif compression, and an average 3% improvement in SMILES validity over the DiGress model across all real-world molecule benchmarking datasets. Furthermore, our approach outperforms the state-of-the-art DeFoG method with respect to motif-conservation by roughly 4%, as evidenced by high ChEMBL-likeness, QED and newly introduced shingles distance scores. The key ideas behind the approach are to use a combination of deterministic and random subgraph perturbations, so that the node and edge noise schedules are codependent; to modify the loss function of the training process in order to exploit the deterministic component of the schedule; and, to "compress" a collection of highly relevant carbon ring and other motif structures into supernodes in a way that allows for simple subsequent integration into the molecular scaffold[1].

1 Introduction

Graphs are fundamental data structures used to model a wide range of complex interactions, including molecules and drugs, biological pathways, and social and co-purchase networks Wu et al. [2021], Zhang et al. [2024], Yang et al. [2024], Waikhom and Patgiri [2021]. Hence, the ability to generate graphs that accurately capture domain-specific distributions is of great significance Tsai et al. [2023], You et al. [2018], Gamage et al. [2020]. While generative models for images and texts are being used with great success Dhariwal and Nichol [2021], Ho et al. [2022], Waikhom and Patgiri [2021], graph generation remains a challenging frontier due to the discrete nature of the models, the need for permutation invariance and other distinctive properties Velikonivtsev et al. [2024], Wang et al. [2023]. This is particularly the case when trying to generate graph samples that innovate molecular compounds Jin et al. [2018], Mitton et al. [2021], Vignac and Frossard [2021], Gómez-Bombarelli et al. [2018], Kwon et al. [2020], Wu et al. [2024], You et al. [2018], Liao et al. [2019], Segler et al. [2018], Popova et al. [2019], Madhawa et al. [2019], Zang and Wang [2020], in which case the generated samples can fail to obey necessary biochemical and physical constraints and preserve important motif compounds.

One way to address these challenges is to use diffusion models Sohl-Dickstein et al. [2015], Ho et al. [2020], Fan et al. [2023], Chen et al. [2024], Trippe et al. [2023], Haefeli et al. [2023], Niu et al. [2020b], Huang et al. [2022], Liu et al. [2024c] which rely on a gradual noise injection process that perturbs the input samples and then reverses the process using a learned denoising network.

[1]The link to the code can be found at: https://github.com/liekon/Discrete-Graph-Generation

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

In the context of small molecule drug generation, existing methods focus on *discrete diffusion* in order to avoid the loss of key structural properties that arise when embedding graph structures into continuous spaces Niu et al. [2020a], Jo et al. [2022b]. Discrete diffusion models incorporate "transition mechanisms" that explicitly consider the categorical attributes of nodes and edges, ensuring fine-grained structural changes during the noise injection process Liu et al. [2024b], Xu et al. [2024a]. They also allow for continuous and discrete denoising steps that diversify the outputs Huang et al. [2022], Vignac et al. [2023], Jo et al. [2022a]. However, they also typically require a very large number of diffusion steps, resulting in large running times and extremely slow graph generation Kong et al. [2023]. More importantly, they lead to structures that have low chemical validity even when measured through exceptionally coarse SMILES nomenclature constraints Hoogeboom et al. [2022], Liu et al. [2021]. Most learning methods consequently fail to produce molecules that can be chemically synthesized or that can properly dock on target proteins.

One well-known graph diffusion method, DiGress Vignac et al. [2023], relies on a cosine noise injection mechanism that controls the amount of perturbations during the different diffusion steps. In the process, every node and edge in the graph is allowed to be modified at each step. DiGress evaluates the generated drug sample quality through SMILES validity, novelty, and uniqueness all of which provide a highly limited measure of the utility of molecules. Also, the training and sampling time of the method is excessive. Furthermore, in its current form, and unlike some nondiffusion based methods Jin et al. [2018], Simonovsky and Komodakis [2018], Mitton et al. [2021], Vignac and Frossard [2021], Gómez-Bombarelli et al. [2018], Kwon et al. [2020], Wu et al. [2024], You et al. [2018], Liao et al. [2019], Segler et al. [2018], Popova et al. [2019], DiGress cannot preserve the structures and frequencies of important drug network motifs (including carbon rings, which are the most important building blocks of organic molecules).

We describe a new small molecule drug diffusion model, Diffusion Models for Molecular Motifs, **DMol**, which significantly outperforms DiGress across all benchmarking datasets in terms of running time, SMILES validity, and novelty scores and at the same time, allows for network motif preservation via compressed representations of subsets of motifs. At the same time, the DMol improves the more informative chemical validity scores when compared to the very recent state-of-the-art DeFoG Qin et al. [2025] method by roughly 4%, since the latter cannot guarantee preservation of motifs. Its main features are as follows:

1. DMol directly relates the **diffusion time-steps** with the **number** of nodes and edges that are allowed to undergo state changes in that step. This approach effectively increases the learning rate of the model as it learns increasingly larger submodels in a gradual manner. It also significantly reduces the time required for molecule generation, leading to an order of magnitude reduction in the number of steps required by diffusion models.

2. The **number** of nodes and edges that DMol perturbs at each step is **deterministic**, but the nodes themselves are selected at random. Furthermore, the perturbed edges are confined to the subgraph induced by the selected vertices, which couples the node and edge perturbations and leads to better preservation of subgraph structures.

3. DMol uses new penalty terms in the objective function that **penalize mismatches** in the deterministic counts of nodes and edges perturbed during the forward and backward steps. Such penalties lead to poor results when the number of perturbed nodes is random, as in DiGress and other methods, and motivates the use of a mixed deterministic/random noise schedule. This allows for further "forced motif" preservation and a more flexible control of the noise schedule/distribution, as evidenced by a straightforward theoretical analysis.

4. One of the most important features of DMol is **motif compression** which allows one to convert chemically important node motifs into supernodes that are diffused similarly to regular atom nodes. The chosen motifs (which are mostly carbon rings) have the property that they can always be reattached to the molecular scaffold, and the number of such molecules is bounded to retain computational savings guaranteed by the first three innovations. Due to their careful chemical selection, only one type of bond is possible during reconstruction, and "decoding" is extremely simple. Motif compression differs both from JT-VAEs Jin et al. [2019] and ring freezing proposed in DiGress (which failed to perform in a satisfactory manner), since in our case one is still allowed to **directly** replace one valid motif structure by a single atom or another valid motif structure (in addition, the choices of substructures used differ widely).

As a result, DMol improves the SMILES validity of DiGress by roughly 3% over all real-world molecular benchmarking datasets while keeping the number of diffusion steps at an order that is one magnitude smaller. At the same time, the significantly more meaningful chemical likeness scores (e.g., RDKit's Landrum et al. [2006] ChEMBL likeness, QED and shingle distances, which indicate how likely the generated molecule is to have desired chemical motifs and similar biological activity properties) of DMol are consistently better than those of DiGress. These scores for DMol are roughly 4% higher, on average, than those of the recent state-of-the-art flow matching method DeFoG Qin et al. [2025], since the latter does not tend to preserve chemical motifs.

As a final remark, it is important to point out that in the biochemistry literature, it has been long recognized that SMILES validity optimization may not be as practically relevant from the chemical synthesis/properties point of few; motif, and especially carbon ring structure preservation, is deemed much more important, but tangible evaluation metrics for true chemical validity are still lacking. See for example the recent work Skinnider [2024] that discusses pros and cons of SMILES validity.

2 Related Works

Viewing small molecules as graphs has inspired a myriad of generative models for drug design, including JT-VAE Jin et al. [2018], MoFlow Zang and Wang [2020], CDGS Huang et al. [2023], EDM Hoogeboom et al. [2022], etc Cornet et al. [2024], Shi et al. [2021], Xu et al. [2024a], Chen et al. [2023], Wu et al. [2022], Bao et al. [2023], You et al. [2018], Liao et al. [2019], Mercado et al. [2021], Segler et al. [2018], Kwon et al. [2020], Jensen [2019], Jo et al. [2024].

Furthermore, numerous surveys on the subject are readily available Tang et al. [2024], Du et al. [2022], Yang et al. [2024]. We partition and review prior works on generative models for drug discovery based on the methodology used.

Traditional Methods. Traditional approaches to graph-based molecular generation relied on the use of Variational Autoencoders (VAEs) as generative models Jin et al. [2018], Simonovsky and Komodakis [2018], Mitton et al. [2021], Vignac and Frossard [2021], Gómez-Bombarelli et al. [2018], Kwon et al. [2020], Wu et al. [2024]. VAEs comprise an encoder that maps the input graph into a latent space and a decoder that reconstructs the graph from the latent embedding. Many methods also involve autoregressive models, which generate molecular graphs by sequentially predicting the next element in the sequence based on previous outputs You et al. [2018], Liao et al. [2019], Segler et al. [2018], Popova et al. [2019]. On the other hand, flow-based methods leverage the concept of normalizing flows for graph generation Madhawa et al. [2019], Zang and Wang [2020], Luo et al. [2021b], Lippe and Gavves [2021], Shi* et al. [2020]. These techniques apply a series of invertible transformations to model complicated distributions based on simpler ones (such as a Gaussian). Additionally, Generative Adversarial Networks (GANs) approaches have also been used for molecule generation De Cao and Kipf [2018], Łukasz Maziarka et al. [2019], Martinkus et al. [2022a]. GANs include a generator, which learns to create realistic molecular graphs, and a discriminator, which learns to differentiate between real and generated samples.

Diffusion-Based Methods. Diffusion models for graph generation can be broadly categorized as follows. Denoising Diffusion Probabilistic Models (DDPM) utilize a Markov chain for the diffusion process Vignac et al. [2023], Xu et al. [2022], Liu et al. [2024b], Xu et al. [2024a], Liu et al. [2024a], Hoogeboom et al. [2022], Jo et al. [2024]. Score-Based Generative Models (SGM) leverage score matching techniques to model the data distribution, and generate graphs by reversing a diffusion process guided by the score of the data distribution Luo et al. [2021a], Chen et al. [2023], Wu et al. [2022]. Furthermore, by replacing discrete time steps with continuous time, one can use stochastic differential equations to model the diffusion and denoising processes Jo et al. [2022a], Lee et al. [2023], Huang et al. [2023], Bao et al. [2023]. Furthermore, studies have shown that different noise schedules may significantly affect molecule generation quality Shi et al. [2025], Nichol and Dhariwal [2021]. Comprehensive surveys on the subjects include Mengchun Zhang et al. [2023], Fan et al. [2023], Chen et al. [2024].

DiGress Vignac et al. [2023] uses a denoising diffusion probabilistic model within a discrete state space and is considered one of the practically effective models for molecular generation tasks. It represents graphs in the discrete space of node and edge attributes by assigning categorical labels to each node and edge. DiGress models the noise addition process as a Markov process, where each node and edge label evolves independently of the others. This assumption mirrors the approach in image-based diffusion models. Denoising is performed using a graph transformer network, trained by

minimizing the cross-entropy loss between the predicted probabilities for nodes and edges and the ground-truth graph. The transformer network reconstructs a denoised graph from a noisy input. To generate new graphs, a noisy graph is first sampled according to the model's limiting distribution, and the trained denoising transformer is then used to produce a graph with the desired properties. The DiGress model is discussed further in Section B.

DisCo Xu et al. [2024b] uses discrete-state continuous-time diffusion, addressing limitations of discrete-time approaches like DiGress. The model formulates graph generation as a continuous-time Markov Chain (CTMC) that preserves the discrete nature of graph-structured data while enabling flexible sampling trade-offs. Unlike discrete-time models with fixed sampling steps, DisCo can adjust sampling steps after training without retraining the model. DisCo enables numerical approaches like τ-leaping for efficient simulation of the reverse process, allowing practitioners to dynamically balance generation quality and computational efficiency during inference. Despite its innovations, DisCo suffers from quadratic computational complexity with respect to the number of nodes, limiting its scalability for generating large graphs. Moreover, DisCo reduces the number of sampling steps at the cost of degraded generation quality.

Flow Matching-based Methods. Flow Matching (FM) has recently emerged as an alternative to diffusion models for generative tasks Lipman et al. [2023], Dao et al. [2023]. FM frameworks define continuous probability paths between a simple prior distribution and the target data distribution, offering comparable performance and efficiency compared to diffusion-based approaches. To address discrete state spaces, Discrete Flow Matching (DFM) formulations have been developed Gat et al. [2024], providing a streamlined approach with more flexible sampling procedures. DeFoG Qin et al. [2025] extends the DFM framework specifically for graph generation tasks. It employs a linear interpolation noising process and a CTMC-based denoising process while preserving the inherent symmetries of graphs. The key innovation in DeFoG is its disentanglement of training and sampling stages, which enables independent optimization of each component. Despite its innovation, DeFoG still requires dataset-specific hyperparameter tuning for optimal sampling strategies, making its adoption potentially challenging for new graph domains without extensive experimentation. Importantly, experiments to be described in this work reveal that DeFoG under-performs on key biochemical motif molecular property metrics such as QED.

3 DMol Diffusion

3.1 Review of Standard Graph Diffusion Models

The focal concept in diffusion models is the noise model q, designed to enable T forward diffusion steps. Graph diffusion relies on progressively introducing noisy perturbations to an undirected graph $\mathcal{G}^0$, encoded as $\mathbf{z}^0$, resulting in a sequence of increasingly noisy data points $(\mathbf{z}^1, \ldots, \mathbf{z}^T)$. The noise addition process is Markovian, which translates to $q(\mathbf{z}^1, \ldots, \mathbf{z}^T \mid \mathbf{z}^0) = \prod_{t=1}^{T} q(\mathbf{z}^t \mid \mathbf{z}^{t-1})$.

More precisely, the encoding of a graph involves categorical information about the nodes V and vertices E of the graph $\mathcal{G} = (V, E)$. The assumptions are that each node belongs to one of a classes and is represented as a one-hot vector in $\mathbb{R}^a$. The same is true for edges, which are assumed to belong to one of b classes (with one class indicating the absence of the edge). We follow the notation from Vignac et al. [2023], where $\mathbf{x}_i$ is used to denote the one-hot encoding of the class of node i. The one-hot encodings are arranged in a node matrix $\mathbf{X}$ of dimensions $n \times a$. Again, in a similar manner, edges are encoded into a tensor $\mathbf{E}$ of dimension $n \times n \times b$. The Markovian state space for each node and edge is the set of classes of the nodes and edges, respectively. With a slight abuse of notation, $\mathbf{x}_l$ stands for the class (state) of node l, while $\mathbf{e}_{ls}$ stands for the class (state) of edge ls.

Diffusion is performed separately on nodes and edges, while the noisy perturbations in the forward process are governed by a state transition matrix. For node-level noise we use transition matrices $(\mathbf{Q}_X^1, \ldots, \mathbf{Q}_X^T)$, where each entry $[\mathbf{Q}_X^t]_{ij}$ defines the probability of an arbitrary node transitioning from state i to state j at time step t. A similar model is used for edges, where the transition matrices $(\mathbf{Q}_E^1, \ldots, \mathbf{Q}_E^T)$ describe the probabilities $[\mathbf{Q}_E^t]_{ij}$ of an arbitrary edge transitioning from state i to state j. Formally, $[\mathbf{Q}_X^t]_{ij} = q(x^t = j \mid x^{t-1} = i)$ and $[\mathbf{Q}_E^t]_{ij} = q(e^t = j \mid e^{t-1} = i)$, where x and e correspond to a generic node and edge, while the superscript indicates the time stamp.

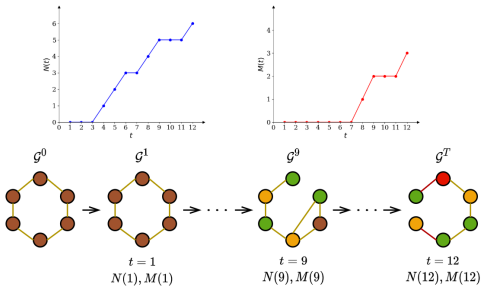
Consequently, the transition probability at time t is given by $q(\mathbf{z}^t \mid \mathbf{z}^{t-1}) = \mathbf{z}^{t-1}\mathbf{Q}^t$, where $\mathbf{z}$ stands for either $\mathbf{x}$ or $\mathbf{e}$ and the subscript of $\mathbf{Q}$ is set accordingly. Adding noise to a graph $\mathcal{G}^{t-1}$ at

time $t-1$ results in a new graph $\mathcal{G}^t = (\mathbf{X}^t, \mathbf{E}^t)$, and involves sampling each node and edge class from a categorical distribution succinctly described as $q(\mathcal{G}^t \mid \mathcal{G}^{t-1}) = \left(\mathbf{X}^{t-1}\mathbf{Q}_X^t, \mathbf{E}^{t-1}\mathbf{Q}_E^t\right)$ and $q(\mathcal{G}^t \mid \mathcal{G}^0) = \left(\mathbf{X}^0\overline{\mathbf{Q}}_X^t, \mathbf{E}^0\overline{\mathbf{Q}}_E^t\right)$, where $\overline{\mathbf{Q}}_X^t = \mathbf{Q}_X^1 \ldots \mathbf{Q}_X^t$ and $\overline{\mathbf{Q}}_E^t = \mathbf{Q}_E^1 \ldots \mathbf{Q}_E^t$. The state transition matrices are invertible, which ensures the reversibility of the noise model: the graphs at time t and time 0 can be transformed into each other using the state transition matrix. Reversibility guarantees consistency between the forward and backward processes and simplifies backward denoising.

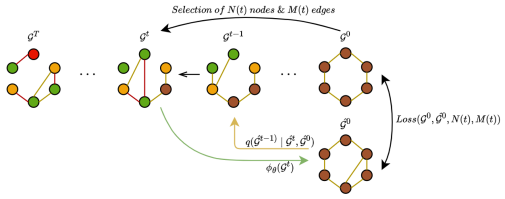
A common issue with most known diffusion models is the large number of steps needed for convergence Hang et al. [2023], Wang et al. [2024]. For large training sets the computational complexity may be overwhelming (see our Results section). Hence, the first important question is how to reduce the number of diffusion steps for graph generation, and at the same time, preserve the chemical utility of the model. These issues are addressed by DMol.

3.2 DMol Forward Noise Adding Strategy

The key idea behind our approach is as follows: at each forward diffusion step t, we randomly select $N(t)$ nodes and $M(t)$ edges from $\mathcal{G}^0$ contained in the complete subgraph induced by the selected nodes (note that the absence of an edge is represented by a special label so that we are effectively dealing with a complete graph), and change their classes according to fixed state transition matrices $\mathbf{Q}_X$ and $\mathbf{Q}_E$, respectively, to obtain the graph $\mathcal{G}^t$. Here, $N(t)$ and $M(t)$ are **deterministic functions of the time step index t.** Note that we effectively run the diffusion process on a determinist number of **randomly selected subsets of nodes and edges**, with the perturbed edges confined to the subgraph induced by the selected nodes. This effectively couples the node and edge diffusion processes.



(a) At each time step t, we randomly select a predetermined deterministic number of nodes and add noise only to the class labels of these nodes and a fixed number of edges in the subgraph *induced* by the nodes. Both selection numbers are controlled by the time index t. The top row depicts how the number of selected nodes $N(t)$ and edges $M(t)$ of the selected subgraph evolves with t (we set $n = 6, k = 2, r = 0.2$).



(b) DMol diffusion relies on randomly selecting $N(t)$ nodes and $M(t)$ edges from the subgraph induced by the $N(t)$ nodes at each time step t, and diffusing their labels to generate a noisy graph $\mathcal{G}^t$. The denoising network ϕ_θ learns to predict the denoised graph from $\mathcal{G}^t$. During inference, the denoised graph is combined with $q(\hat{\mathcal{G}}^{t-1}|\hat{\mathcal{G}}^t, \hat{\mathcal{G}}^0)$ to predict $\hat{\mathcal{G}}^{t-1}$.

Figure 1: (a) The forward process; (b) DMol illustration.

For $N(t)$ and $M(t)$, we adopt cosine schedules, $N(t) = \lfloor (1-\alpha)\, n \rfloor$ and $M(t) = \lfloor (1-\alpha)\, N(t)\,(N(t)-1)\, 0.5\, r \rfloor$, where $\alpha = \cos^2(0.5\pi(t/T + c)/(1+c))$, and n stands for the the number of nodes in the graph, r is a hyperparameter and c is a small positive constant. The total number of diffusion steps is fixed at $T = k\, n$, where k is a small hyperparameter (typically set to 1 or 2 so as to keep the number of diffusion steps as small as possible; details behind the selection of appropriate values of k can be found in Appendix G.3). The maximum number of diffusion steps is proportional to the number of nodes n (typically around 40 for small molecule drugs). Hence, DMol requires significantly fewer diffusion steps than other methods, such as Digress (which requires 500 or 1000 steps), and adapts itself to the size of the graph.

Additionally, our forward process constrains the class changes of edges to the subgraph induced by the nodes that underwent a state change. This ensures that the sets of altered nodes and edges are co-dependent. Still, once the selections are made, the actual state changes of nodes and edges are governed by independent processes. We next turn our attention to the choice of the transition probability matrices. The results of DiGress Vignac et al. [2023] have shown that making the

transition probability from class i to class j proportional to the marginal probability of class j within the training dataset results in more effective learning of the true data distribution compared to when using uniform transitions. We follow the same procedure and define the marginal probabilities of node classes in the training dataset as $\mathbf{m}_x \in \mathbb{R}^{1\times a}$ and those of edge classes as $\mathbf{m}_e \in \mathbb{R}^{1\times b}$, respectively. Hence,

$$\mathbf{Q}_X[i,j] = \begin{cases} \frac{\mathbf{m}_x[j]}{\sum_{\ell\neq i}\mathbf{m}_x[\ell]}, & \text{if } i \neq j, \\ 0, & \text{if } i = j. \end{cases} \quad \mathbf{Q}_E[i,j] = \begin{cases} \frac{\mathbf{m}_e[j]}{\sum_{\ell\neq i}\mathbf{m}_e[\ell]}, & \text{if } i \neq j, \\ 0, & \text{if } i = j. \end{cases}$$

During the forward process, we compute the distributions of all node and edge classes after state transitions. However, we only focus on the distributions corresponding to the selected $N(t)$ nodes and $M(t)$ edges, sampling from these distributions to update their states. We preserve the states of the remaining nodes from the previous step. Algorithm 1 and Figure 1b illustrate our forward process.

Algorithm 1 Forward Process

Input: $\mathcal{G}^0 = (\mathbf{X}^0, \mathbf{E}^0)$, state transition matrices $\mathbf{Q}_X \& \mathbf{Q}_E$, node&edge selection functions $N(\cdot) \& M(\cdot)$.
Process:
Sample $t \in \{1 \dots T\}$.
Sample $\tilde{\mathcal{G}}^t = (\tilde{\mathbf{X}}^t, \tilde{\mathbf{E}}^t)$ from $\mathbf{X}^0\mathbf{Q}_X \times \mathbf{E}^0\mathbf{Q}_E$.
Calculate $u = N(t)$ and $v = M(t)$.
Randomly select u nodes and v edges from the subgraph induced by the u nodes and generate "filter" matrices $\mathbf{K}_x$ and $\mathbf{K}_e$ which serve as indicators of the selections.
Set $\mathcal{G}^t = \mathcal{G}^0$ and replace $\mathbf{K}_x\mathbf{X}^t$ with $\mathbf{K}_x\tilde{\mathbf{X}}^t$, and $\mathbf{K}_e\mathbf{E}^t$ with $\mathbf{K}_e\tilde{\mathbf{E}}^t$.

Algorithm 2 Sampling Process

Input: Denoising Network ϕ_θ, node&edge class distribution $\mathbf{m}_x^T = [p_i^{x,T}] \& \mathbf{m}_e^T = [p_k^{e,T}]$, state transition matrices $\mathbf{Q}_X \& \mathbf{Q}_E$.
Process:
Sample $\hat{\mathcal{G}}^T$ from $\mathbf{m}_x^T \times \mathbf{m}_e^T$.
for $t = T$ **to** 1 **do**
  $z \leftarrow f(\hat{\mathcal{G}}^t, t)$ {Structural and spectral features}.
  $\hat{p}^X, \hat{p}^E \leftarrow \phi_\theta(\hat{\mathcal{G}}^t, z, t, R_x, R_e)$ {Prediction}.
  Sample $\hat{\mathcal{G}}^0 = (\hat{\mathbf{X}}^0, \hat{\mathbf{E}}^0)$ from $\hat{p}^X \times \hat{p}^E$.
  Sample $\tilde{\mathcal{G}}^{t-1} = (\tilde{\mathbf{X}}^{t-1}, \tilde{\mathbf{E}}^{t-1})$ from $\hat{\mathbf{X}}^0\mathbf{Q}_X \times \hat{\mathbf{E}}^0\mathbf{Q}_E$.
  Calculate $u = N(t-1)$, $v = M(t-1)$.
  Randomly select u nodes and v edges from the subgraph induced by the u nodes and generate corresponding "filter" matrices $\mathbf{K}_x$ and $\mathbf{K}_e$.
  Set $\hat{\mathcal{G}}^{t-1} = \hat{\mathcal{G}}^0$ and replace $\mathbf{K}_x\hat{\mathbf{X}}^{t-1}$ with $\mathbf{K}_x\tilde{\mathbf{X}}^{t-1}$, $\mathbf{K}_e\hat{\mathbf{E}}^{t-1}$ with $\mathbf{K}_e\tilde{\mathbf{E}}^{t-1}$.
end for

3.3 Denoising/Sampling Process

During the denoising process, we first sample a random graph based on the marginal distributions of nodes and edges at $t = T$. This random graph is then used as the input to the denoising network ϕ_θ to predict a denoised graph. Based on the prediction, we use the noise model to obtain the noisy graph at $t = T - 1$. Subsequently, this noisy graph is fed into the denoising network, and the above process is repeated iteratively until a new graph is sampled.

To start the sampling process, we need to compute the marginal distributions of nodes and edges at $t = T$. At time $t = T$, a node class i at $t = T$ is a result of transitions from some other node classes at $t = 0$. As a result, the marginal probability of the i-th node class at time $t = T$ equals

$$p_i^{x,T} = p_i^x \sum_j \frac{p_j^x}{1 - p_j^x},$$

where p_i^x and p_j^x are the marginal probabilities of node classes i and j at time $t = 0$, respectively. For edges, we similarly have that the marginal probabilities of the i-th edge class at $t = T$ equals

$$p_i^{e,T} = (1 - r)\, p_i^e + r\, p_i^e \sum_l \frac{p_l^e}{1 - p_l^e},$$

where p_i^e and p_l^e are the marginal probabilities of edge classes i and l at $t = 0$.

After sampling the random graph, we use the denoising model to predict the clean graph and continue this process iteratively. At time step t, the input to the denoising model includes not only the noisy graph and the time t but also two additional functions: $R_x(t) = N(t)/n$ and

$R_e(t) = M(t)/(0.5\,n\,(n-1))$, which describe the proportion of nodes and edges in the noisy graph at time t that undergo state transitions relative to the total number of nodes and edges, respectively. They indicate to the denoising network that the input noisy graph and the predicted graph should differ by $N(t)$ nodes and $M(t)$ edges, thereby helping the network make more accurate predictions. The sampling steps are listed in Algorithm 2. Figure 8 in the Appendix is an example of the sampling process from a random graph to a denoised graph. Note that the posterior distribution used by the DMol sampling process, $q(\hat{\mathcal{G}}^{t-1}|\hat{\mathcal{G}}^t, \phi_\theta)$, factorizes as $q(\hat{\mathcal{G}}^{t-1}|\hat{\mathcal{G}}^0)\, q_\theta(\hat{\mathcal{G}}^0|\hat{\mathcal{G}}^t)$. This decomposition improves both computational efficiency and inference accuracy.

The loss we use to train the denoising network differs from that of DiGress in so far that it contains two additional penalties,

$$l(\hat{\mathbf{p}}^{\mathcal{G}}, \mathcal{G}) = \sum_{1\le i\le n} \text{CE}(\mathbf{x}_i, \hat{\mathbf{p}}_i^X) + \lambda_1 \sum_{1\le i,j\le n} \text{CE}(\mathbf{e}_{ij}, \hat{\mathbf{p}}_{ij}^E)$$
$$+ \lambda_2\, \text{MSE}\big(D(\text{argmax}(\hat{\mathbf{p}}^X); \text{argmax}(\mathbf{X})), N(t)\big) + \lambda_3\, \text{MSE}\big(D(\text{argmax}(\hat{\mathbf{p}}^E); \text{argmax}(\mathbf{E})), M(t)\big).$$

where $\hat{\mathbf{p}}^{\mathcal{G}} = (\hat{\mathbf{p}}^X, \hat{\mathbf{p}}^E)$ denotes the predicted probability distributions, while $\mathbf{x}_i$ and $\mathbf{e}_{ij}$ represent the one-hot encodings of nodes and edges, respectively. CE stands for the cross-entropy loss, while MSE refers to the mean squared error loss. The term $\text{argmax}(\hat{\mathbf{p}}^X)$ equals the class index of highest probability for each node, while $\text{argmax}(\mathbf{X})$ effectively converts the one-hot encoding of each node into its corresponding class index. $D(\cdot;\cdot)$ is equal to the number of nodes (edges) in the two arguments that have different classes. The first two terms are the standard node classification and edge classification losses. The third and fourth terms ensure that the differences in the number of nodes and edges of the predicted and the ground truth graphs, respectively, are as close as possible to the number of nodes and edges modified during the noise addition process. This is possible since $N(t), M(t)$ are deterministic - the added losses degrade performance otherwise.

We also briefly remark that it is easy to see that the DMol diffusion model and the modified loss function are permutation invariant. To see why the second claim is true, observe that our loss function comprises two components, a cross-entropy loss and an MSE loss. For the cross-entropy loss terms,

$$l_{CE}(\hat{\mathbf{p}}^{\mathcal{G}}, \mathcal{G}) = \sum_{1\le i\le n} \text{CE}(\mathbf{x}_i, \hat{\mathbf{p}}_i^X) + \lambda_1 \sum_{1\le i,j\le n} \text{CE}(\mathbf{e}_{ij}, \hat{\mathbf{p}}_{ij}^E),$$

Since the overall cross-entropy loss is obtained by summing up the individual cross-entropy values for each node and edge, and the sum operation is permutation invariant, then the cross-entropy loss is also invariant. We arrive at a similar conclusion for the MSE terms in $l_{MSE}(\hat{\mathbf{p}}^{\mathcal{G}}, \mathcal{G})$,

$$\lambda_2\, \text{MSE}\big(D(\text{argmax}(\hat{\mathbf{p}}^X); \text{argmax}(\mathbf{X})), N(t)\big) + \lambda_3\, \text{MSE}\big(D(\text{argmax}(\hat{\mathbf{p}}^E); \text{argmax}(\mathbf{E})), M(t)\big),$$

where invariance follows based on the definition of $D(\cdot;\cdot)$ and the fact that the loss is computed separately for nodes/edges with different indices and then summed up.

3.4 Sampling Efficiency of DMol

To explain the sampling efficiency of DMol, we analyze the minimum time step increment δt required to perturb exactly one node (i.e., the smallest discrete noise unit) during the diffusion process. We compare DMol with DiGress; however, it is important to note that the conclusion holds for all discrete diffusion models, including DiGress, DisCo, etc. The ratio of the required time step increments for DiGress and DMol satisfies

$$\frac{\delta t_{\text{DiGress}}}{\delta t_{\text{DMol}}} = \frac{1}{\sum_i p_i^x (1 - p_i^x)} > 1, \tag{1}$$

where p_i^x denotes the probability of node type i (for derivations, see Appendix D.2). The inequality demonstrates that DMol requires fewer steps to perturb a single node. Since the total number of diffusion steps in DMol is solely determined by the number of nodes, the efficient addition of noise to nodes directly contributes to the overall efficiency of DMol. For further discussion of likelihood computations, noise distribution evolution, and expressivity refer to the Appendices C D.

4 Motif Compression

To preserve motif compositions (e.g., carbon rings), we use a new motif compression feature, depicted in Figure 2. The process is designed to better preserve motifs in molecular graphs.

Table 1: Performance comparison on the QM9 dataset (see Appendix G.4 for more details).

MODEL	V↑	V.U.↑	V.U.N.↑
GRAPHNVP	83.5	18.4	-
GDSS	96.0	94.6	-
GRUM	**99.4**	85.1	22.2
DIGRESS	97.8	95.2	31.8
DISCO	98.2	95.6	57.5
DEFOG	97.9	95.9	62.8
DMOL(OURS)	98.3	**96.0**	**75.1**

Table 2: Performance comparison on the same version of MOSES. For fair comparisons, we ran the DisCo code without any modifications, and DeFoG with 50 diffusion steps to ensure similar computational complexity to our method (DeFoG also reports their results for 50 diffusion steps).

MODEL	V↑	U↑	N↑	FILTERS↑	FCD↓	SNN↑	SCAF↑
DIGRESS	84.8	**100**	94.5	97.2	1.18	0.55	14.6
DISCO	85.7	**100**	97.4	95.8	1.40	0.51	14.5
DEFOG	84.2	**100**	97.2	96.9	1.89	0.50	**14.8**
DMOL (OURS)	**87.8**	**100**	**100**	**97.8**	**1.12**	**0.58**	**14.8**

Specifically, we convert a small predefined number $(3 - 15)$ of most frequently occurring subgraphs/motifs (which may differ for different training sets) that can only form single bonds with the scaffolds through their available carbons into supernodes, introducing in the process new node classes to represent the compressed motifs. The compressed graph representations are directly fed into our DMol diffusion model. During the sampling process, the supernodes are converted back to their respective motifs and integrated into the graph scaffold in a chemically valid manner. Although superficially similar to part of the JT-VAE Jin et al. [2019] approach, motif compression is significantly different. The JT approach uses not just motifs but a large number (roughly several hundreds) of submolecular structures, and requires finding a minimum spanning tree during the encoding process; furthermore, VAEs do not directly generate graphs, and one has to perform complicated decoding. The main issue is that each compressed molecular substructure can form many different types of bonds with different atoms during the reconstruction process. The options are ranked according to a special scoring function and the overall process is computationally demanding. The motifs used in DMol for different datasets are available in the Appendix, Figures 3, 4, 5.

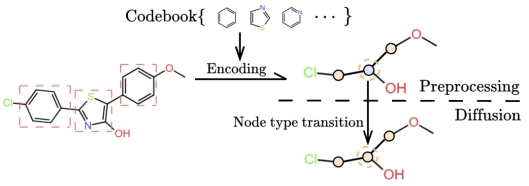


Figure 2: Motifs (i.e., substructures occurring with high frequencies or frequencies higher than predicted by random models) that also allow for unique scaffold integration are compressed into supernodes with their own labels. During diffusion, supernodes are either converted into other classes of supernodes or into atomic nodes, and vice versa. During sampling, the supernode is decoded back into its corresponding motif.

5 Experiments

We present next the results of running DMol on several benchmarking datasets. For experiments pertaining to other graph models (e.g., SBMs and planar graphs), the reader is referred to Appendix G.1.

Benchmarks. We compared DMol to several graph generation methods. The benchmarking models used in the experiments include GraphVAE Simonovsky and Komodakis [2018], GT-VAE Mitton et al. [2021], Set2GraphVAE Vignac and Frossard [2021], SPECTRE Martinkus et al. [2022b], GraphNVP Madhawa et al. [2019], GDSS Jo et al. [2022b], GruM Jo et al. [2024], DiGress Vignac et al. [2023], DisCo Xu et al. [2024b] and DeFoG Qin et al. [2025].

Data. We tested the performance of different generative models on the QM9 Wu et al. [2018], MOSES Polykovskiy et al. [2020], and GUACAMOL Brown et al. [2019] datasets. QM9 is a small dataset: we used 100K molecules for training, 20K for validation, and 13K for testing. Both MOSES

and GUACAMOL contain millions of molecules, with the number of heavy atoms inside a single molecule bounded by 26 and 88, respectively. We used 85% of the molecules for training, 5% for validation, and 10% for testing.

Setup. For each molecule dataset, we choose a different (relatively small) number of motifs to be converted into supernodes. The selected motif structures can be found in the Appendix A. We set the hyperparameters of DMol in both setting to $k = 2$, $r = 0.2$, $\lambda_1 = 5$, $\lambda_2 = 1$, $\lambda_3 = 1$. The number of training epochs is set to 500. The experiments were conducted using NVIDIA H100 GPUs. The QM9 runs were executed on a single GPU, while the MOSES and GUACAMOL experiments were processed in parallel using 4 GPUs.

Evaluation Metrics. The metrics used in the experiments include validity, uniqueness, and novelty. Validity (V) measures the proportion of generated molecules that pass basic valency checks (e.g., correspond to valid SMILES files). Uniqueness (U) quantifies the proportion of generated molecules that are distinct. Novelty (N) represents the percentage of generated molecules that do not appear in the training dataset. Furthermore, V.U. stands for the product of validity and uniqueness scores, while V.U.N. stands for the product of validity, uniqueness, and novelty. Since MOSES and GUACAMOL are benchmarking datasets, they introduce their own set of metrics for reporting results. In addition to V, U, and N, MOSES incorporates a filter score, the Frechet ChemNet Distance (FCD) Preuer et al. [2018], SNN, and scaffold similarity scores, while GUACAMOL uses FCD and KL divergence. The definition of these metrics are in Appendix E. The metrics mentioned above still have certain limitations. Since these metrics assess the overall quality of the generation process, but do not provide insights into the performance of individual molecules. To address this limitation, we also use the ChEMBL Likeness Score (CLscore) Bühlmann and Reymond [2020], Quantitative Estimation of Drug-likeness (QED) Bickerton et al. [2012] score, and our newly introduced Shingle distance (SD) (which quantifies the divergence between two molecular motif (shingle) distributions, see the Appendix F). The CLscore is calculated by considering how many substructures in a molecule also occur in the drug-like dataset ChEMBL Gaulton et al. [2016]. QED evaluates drug-likeness by considering eight widely recognized molecular properties, such as molecular mass and polar surface area. More details are available in Appendix F. Another performance metrics used is the sample time (seconds) needed to generate a batch of 64 molecules.

Table 3: Classical validity, uniqueness and novelty performance comparison on GUACAMOL, for comparable generation complexity.

MODEL	V↑	U↑	N↑	KL DIV↑	FCD↑
DIGRESS	84.5	100	99.8	93.1	68.4
DISCO	86.0	100	99.8	92.8	59.7
DEFOG	**86.7**	**100**	99.5	92.5	58.1
DMOL (OURS)	**86.7**	**100**	**100**	**94.2**	**69.3**

Table 4: Biochemical performance comparison on MOSES and GUACAMOL. Here, CL=ChEMBL likeness score, SD=shingle distance, ANS=average number of diffusion steps, ST=sample time, and TT=training time. Note the significantly lower ANS, ST and TT times of DMol compared to DeFoG.

DATASET	MODEL	CL↑	QED↑	SD↑	ANS↓	ST(S)↓	TT(H)↓
MOSES	DIGRESS	4.4965	0.8024	0.647	500	68.24	136
	DISCO	4.1373	0.7518	0.685	500	65.36	132
	DEFOG	4.0502	0.7475	**0.693**	50	5.81	115
	DMOL (OURS)	**4.5033**	**0.8055**	0.683	**38**	**1.52**	**60**
GUACAMOL	DIGRESS	4.0412	0.5650	0.668	500	70.82	162
	DISCO	3.9093	0.5366	0.679	500	68.76	155
	DEFOG	3.8075	0.4927	**0.687**	50	7.62	132
	DMOL (OURS)	**4.2231**	**0.5786**	0.678	**48**	**1.84**	**88**

Results. The experimental results are presented in Tables 1, 2, 3 and 4. On QM9, DMol achieves the best performance compared to other baseline models. In contrast, GruM, DiGress, DisCo, DeFoG have low novelty scores, indicating that most of the generated molecules may be duplicates or small perturbation of those already present in the training set. On MOSES and GUACAMOL, DMol demonstrates superior performance compared to other methods with respect to all metrics.

Table 4 presents a comparison of diffusion models with respect to the CL/QED scores, average number of diffusion steps, sample time, shingle distance and training time. The results demonstrate that DMol consistently outperforms DiGress, DisCo, and DeFoG across multiple metrics. DMol produces samples with excellent chemical properties as evidenced by high CL scores and QED values. DMol also has larger SD than DiGress, indicating its ability to generate more novel molecules. Although DisCo and DeFoG achieve even higher SDs, as expected, due to lack of motif preservation, this comes at the cost of reduced chemical property preservation. Additionally, DMol exhibits excellent computational efficiency, reducing the average number of diffusion steps by an order of magnitude and significantly accelerating both sampling and training time. For a detailed ablation study, please see Appendix G.2.

6 Motif Distribution Analysis

To validate that motif compression does not introduce biases towards the selected compressed motifs, we conducted a comprehensive analysis comparing motif probabilities between training and generated molecular sets on the MOSES dataset. We examine the top 30 most frequent motifs, with the top 15 (IDs $0-14$) structures chosen for compression into supernodes, and the remaining 15 (IDs $15-29$) serving as a control group to assess potential distribution distortions.

Individual Motif Probabilities. For each motif, we compute the probability that a molecule contains that specific motif. The results (detailed in Appendix Table 10) demonstrate a close alignment between training and generated set distributions. The average supernode motif (IDs $0-14$) probability (IDs $0-14$) equals 0.1189 in the training set and 0.1143 in the generated set. For nonsupernode motifs (IDs 15-29), these average probabilities equal 0.0189 and 0.0160, respectively. This indicates that our approach does not create systematic bias between compressed and uncompressed substructures.

Motif Co-occurrence Patterns. We examined joint probabilities of frequent motif pairs to verify that our model maintains realistic structural relationships. For example, the co-occurrence probability of benzene (c1ccccc1) and pyridine (c1ccncc1) is 0.11 in the training set versus 0.10 in the generated set, and similar results hold for other pairs. This demonstrates that DMol closely preserves not only individual motif frequencies but also their co-occurrence patterns.

Node and Supernode Marginal Distributions. We compared the marginal distributions of all node and supernode classes of training and generated molecules (see Appendix Table 11). The distributions show strong alignment, confirming that DMol samples from the original data distribution without introducing biases. For instance, carbon atoms comprise 0.4897 of the training atom set, while for the generated set this value equals 0.4940; the most frequent supernode (c1ccccc1, benzene) appears with probabilities 0.0736 and 0.0750, respectively.

We attribute this strong preservation of substructure statistics to two key design choices: (1) using marginal distributions from the training set as priors for both supernode and atom sampling, and (2) enforcing that $N(t)$ nodes change at each timestep, ensuring adequate creation of generating nonmotif substructures. These results demonstrate that motif compression effectively preserves molecular motif distributions without introducing systematic bias.

7 Conditional Generation Based on Scaffolds and Docking Results

To avoid issues associated with direct conditional generation, we instead proposed a scaffold-informed pipeline. The key ideas are to cluster the training set molecules according to either their RDKit features, or embeddings generated by Mol2Vec or Grover Jaeger et al. [2018], Rong et al. [2020], and then run DMol on the samples in sufficiently large cluster groups separately (see Appendix G.6. For relevant docking results, please see Appendix I.

8 Conclusion

We present DMol, a highly computationally efficient motif-preserving graph diffusion for small molecule drug generation. DMol couples node and edge noise through a combination of deterministic and stochastic mechanisms, adapts the loss function and includes motif compression. On biochemically relevant performance metrics, such as the ChEMBL likeness and QED scores, it outperforms all other reported methods. **Limitations and Future Work:** Please refer to Appendix K and Appendix L.

Acknowledgments

We gratefully acknowledge the financial support and computational resources provided by Abbvie and the NSF grant CCF 24-02815, which were essential for conducting this research. We also extend our sincere thanks to Matthew S. Krafczyk from the National Center for Supercomputing Applications (NCSA) at the University of Illinois Urbana-Champaign for his valuable assistance in resolving technical simulation issues encountered during our experiments. Furthermore, we thank Jiwoong Jung and Ziheng Qi for useful discussion.

References

Fan Bao, Min Zhao, Zhongkai Hao, Peiyao Li, Chongxuan Li, and Jun Zhu. Equivariant energy-guided SDE for inverse molecular design. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=r0otLtOwYW.

G. Richard J. Bickerton, Gaia V. Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L. Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4 2:90–8, 2012. URL https://api.semanticscholar.org/CorpusID:205289650.

Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59 (3):1096–1108, March 2019. ISSN 1549-960X. doi: 10.1021/acs.jcim.8b00839. URL http://dx.doi.org/10.1021/acs.jcim.8b00839.

Sven Bühlmann and Jean-Louis Reymond. Chembl-likeness score and database gdbchembl. *Frontiers in Chemistry*, 8, 2020. ISSN 2296-2646. doi: 10.3389/fchem.2020.00046. URL https://www.frontiersin.org/journals/chemistry/articles/10.3389/fchem.2020.00046.

Hongyang Chen, Can Xu, Lingyu Zheng, Qiang Zhang, and Xuemin Lin. Diffusion-based graph generative methods, 2024. URL https://arxiv.org/abs/2401.15617.

Xiaohui Chen, Yukun Li, Aonan Zhang, and Li-Ping Liu. Nvdiff: Graph generation through the diffusion of node vectors, 2023. URL https://arxiv.org/abs/2211.10794.

François R J Cornet, Grigory Bartosh, Mikkel N. Schmidt, and Christian A. Naesseth. Equivariant neural diffusion for molecule generation. In *ICML 2024 AI for Science Workshop*, 2024. URL https://openreview.net/forum?id=3iih8PGAH7.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

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

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *CoRR*, abs/2105.05233, 2021. URL https://arxiv.org/abs/2105.05233.

Yuanqi Du, Tianfan Fu, Jimeng Sun, and Shengchao Liu. Molgensurvey: A systematic survey in machine learning models for molecule design, 2022. URL https://arxiv.org/abs/2203.14500.

Wenqi Fan, Chengyi Liu, Yunqing Liu, Jiatong Li, Hang Li, Hui Liu, Jiliang Tang, and Qing Li. Generative diffusion models on graphs: Methods and applications. *arXiv preprint arXiv:2302.02591*, 2023.

Anuththari Gamage, Eli Chien, Jianhao Peng, and Olgica Milenkovic. Multi-motifgan (mmgan): Motif-targeted graph generation and prediction. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4182–4186, 2020. doi: 10.1109/ICASSP40776.2020.9053451.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching, 2024. URL https://arxiv.org/abs/2407.15595.

Anna Gaulton, Anne Hersey, Michał Nowotka, A. Patrícia Bento, Jon Chambers, David Mendez, Prudence Mutowo, Francis Atkinson, Louisa J. Bellis, Elena Cibrián-Uhalte, Mark Davies, Nathan Dedman, Anneli Karlsson, María Paula Magariños, John P. Overington, George Papadatos, Ines Smit, and Andrew R. Leach. The chembl database in 2017. *Nucleic Acids Research*, 45(D1): D945–D954, 11 2016. ISSN 0305-1048. doi: 10.1093/nar/gkw1074. URL https://doi.org/10.1093/nar/gkw1074.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

D S Goodsell and A J Olson. Automated docking of substrates to proteins by simulated annealing. *Proteins*, 8(3):195–202, 1990.

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.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-snr weighting strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7441–7451, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models, 2022. URL https://arxiv.org/abs/2204.03458.

Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d, 2022. URL https://arxiv.org/abs/2203.17003.

Han Huang, Leilei Sun, Bowen Du, Yanjie Fu, and Weifeng Lv. Graphgdp: Generative diffusion processes for permutation invariant graph generation, 2022. URL https://arxiv.org/abs/2212.01842.

Han Huang, Leilei Sun, Bowen Du, and Weifeng Lv. Conditional diffusion based on discrete graph structures for molecular graph generation, 2023. URL https://arxiv.org/abs/2301.00427.

Sabrina Jaeger, Simone Fulle, and Samo Turk. Mol2vec: unsupervised machine learning approach with chemical intuition. *Journal of chemical information and modeling*, 58(1):27–35, 2018.

Jan H. Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chem. Sci.*, 10:3567–3572, 2019. doi: 10.1039/C8SC05372C. URL http://dx.doi.org/10.1039/C8SC05372C.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2323–2332. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/jin18a.html.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation, 2019. URL https://arxiv.org/abs/1802.04364.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. *arXiv:2202.02514*, 2022a. URL https://arxiv.org/abs/2202.02514.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*, pages 10362–10383. PMLR, 2022b.

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture, 2024. URL https://arxiv.org/abs/2302.03596.

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.

Youngchun Kwon, Dongseon Lee, Youn-Suk Choi, Kyoham Shin, and Seokho Kang. Compressed graph representation for scalable molecular graph generation. *Journal of Cheminformatics*, 12:1–8, 2020.

Greg Landrum et al. Rdkit: Open-source cheminformatics, 2006.

Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Phillip Lippe and Efstratios Gavves. Categorical normalizing flows via continuous transformations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-GLNZeVDuik.

Gang Liu, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Inverse molecular design with multi-conditional diffusion guidance. *CoRR*, abs/2401.13858, 2024a. URL https://doi.org/10.48550/arXiv.2401.13858.

Gang Liu, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Graph diffusion transformers for multi-conditional molecular generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=cfrDLD1wfO.

Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. GraphEBM: Molecular graph generation with energy-based models. In *Energy Based Models Workshop - ICLR 2021*, 2021. URL https://openreview.net/forum?id=Gc51PtL_zYw.

Yijing Liu, Chao Du, Tianyu Pang, Chongxuan Li, Wei Chen, and Min Lin. Graph diffusion policy optimization. *CoRR*, abs/2402.16302, 2024c. URL https://doi.org/10.48550/arXiv.2402.16302.

Shitong Luo, Chence Shi, Minkai Xu, and Jian Tang. Predicting molecular conformation via dynamic graph score matching. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021a. URL https://openreview.net/forum?id=hMY6nm9lld.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. 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 7192–7203. PMLR, 18–24 Jul 2021b. URL https://proceedings.mlr.press/v139/luo21a.html.

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs, 2019. URL https://arxiv.org/abs/1905.11600.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, pages 15159–15179. PMLR, 2022a.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators, 2022b. URL https://arxiv.org/abs/2204.01613.

Mengchun Zhang, Maryam Qamar, Taegoo Kang, Yuna Jung, Chenshuang Zhang, Sung-Ho Bae, and Chaoning Zhang. A survey on graph diffusion models: Generative ai in science for molecule, protein and material. 2023. doi: 10.13140/RG.2.2.26493.64480. URL https://rgdoi.net/10.13140/RG.2.2.26493.64480.

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*, 2(2):025023, 2021.

Joshua Mitton, Hans M Senn, Klaas Wynne, and Roderick Murray-Smith. A graph vae and graph transformer approach to generating molecular graphs. *arXiv preprint arXiv:2104.04345*, 2021.

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.

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*, pages 4474–4484. PMLR, 2020a.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling, 2020b. URL https://arxiv.org/abs/2003.00638.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alan Aspuru-Guzik, and Alex Zhavoronkov. Molecular sets (moses): A benchmarking platform for molecular generation models, 2020. URL https://arxiv.org/abs/1811.12823.

Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. Molecularrnn: Generating realistic molecular graphs with optimized properties, 2019. URL https://arxiv.org/abs/1905.13372.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Günter Klambauer. Fréchet chemnet distance: A metric for generative models for molecules in drug discovery, 2018. URL https://arxiv.org/abs/1803.09518.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation, 2025. URL https://arxiv.org/abs/2410.04263.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Grover: Self-supervised message passing transformer on large-scale molecular data. *arXiv preprint arXiv:2007.02835*, 2(3):17, 2020.

Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1): 120–131, 2018.

Chence Shi*, Minkai Xu*, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1esMkHYPr.

Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation, 2021. URL https://arxiv.org/abs/2105.03902.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025. URL https://arxiv.org/abs/2406.04329.

Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders, 2018. URL https://openreview.net/forum?id=SJlhPMWAW.

Michael A Skinnider. Invalid smiles are beneficial rather than detrimental to chemical language models. *Nature Machine Intelligence*, 6(4):437–448, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

Xiangru Tang, Howard Dai, Elizabeth Knight, Fang Wu, Yunyang Li, Tianxiao Li, and Mark Gerstein. A survey of generative ai for de novo drug design: New frontiers in molecule and protein generation, 2024. URL https://arxiv.org/abs/2402.08703.

Brian L. Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi S. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6TxBxqNME1Y.

Jui-Yi Tsai, Ya-Wen Teng, Ho Chiok Yew, De-Nian Yang, and Lydia Y. Chen. Cdgraph: Dual conditional social graph synthesizing via diffusion model, 2023. URL https://arxiv.org/abs/2311.01729.

Fedor Velikonivtsev, Mikhail Mironov, and Liudmila Prokhorenkova. Challenges of generating structurally diverse graphs, 2024. URL https://arxiv.org/abs/2409.18859.

Clément Vignac and Pascal Frossard. Top-n: Equivariant set and graph generation without exchangeability. *CoRR*, abs/2110.02096, 2021. URL https://arxiv.org/abs/2110.02096.

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*, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

Lilapati Waikhom and Ripon Patgiri. Graph neural networks: Methods, applications, and opportunities, 2021. URL https://arxiv.org/abs/2108.10733.

Kai Wang, Mingjia Shi, Yukun Zhou, Zekai Li, Zhihang Yuan, Yuzhang Shang, Xiaojiang Peng, Hanwang Zhang, and Yang You. A closer look at time steps is worthy of triple speed-up for diffusion model training. *arXiv preprint arXiv:2405.17403*, 2024.

Yuyang Wang, Zijie Li, and Amir Barati Farimani. *Graph Neural Networks for Molecules*, page 21–66. Springer International Publishing, 2023. ISBN 9783031371967. doi: 10.1007/978-3-031-37196-7_2. URL http://dx.doi.org/10.1007/978-3-031-37196-7_2.

Huaijin Wu, Xinyu Ye, and Junchi Yan. QVAE-mole: The quantum VAE with spherical latent variable learning for 3-d molecule generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=RqvesBxqDo.

Lemeng Wu, Chengyue Gong, Xingchao Liu, Mao Ye, and qiang liu. Diffusion-based molecule generation with informative prior bridges. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TJUNtiZiTKE.

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. *Chemical science*, 9(2):513–530, 2018.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PzcvxEMzvQC.

Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=YkSKZEhIYt.

Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation, 2024b. URL https://arxiv.org/abs/2405.11416.

Nianzu Yang, Huaijin Wu, Kaipeng Zeng, Yang Li, and Junchi Yan. Molecule generation for drug design: a graph learning perspective, 2024. URL https://arxiv.org/abs/2202.09212.

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*, pages 5708–5717. PMLR, 2018.

Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 617–626, 2020.

He Zhang, Bang Wu, Xingliang Yuan, Shirui Pan, Hanghang Tong, and Jian Pei. Trustworthy graph neural networks: Aspects, methods, and trends. *Proceedings of the IEEE*, 112(2):97–139, 2024. doi: 10.1109/JPROC.2024.3369017.

Łukasz Maziarka, Agnieszka Pocha, Jan Kaczmarczyk, and Michał Warchoł. Mol-cycleGAN - a generative model for molecular optimization, 2019. URL https://openreview.net/forum?id=BklKFo09YX.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims can be found in Abstract and Introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations can be found in Section K.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the full set of assumptions in Section 3.3 and the complete proof can be found in Section D.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the code to reproduce. The information and setting of experiments can be found in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code address can be found in Abstract.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The training and test details can be found in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Our experiments are extremely resource and time intensive, and due to the large scale of the training data, the experimental results exhibit virtually no fluctuation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The information on the computer resources can be found in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics and made sure that our research conform it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Broader impacts can be found in Section J.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all baseline methods, models and datasets. No additional licenses are required.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the code and the documentation alongside the code.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or nonstandard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in our work does not involve LLMs as any important, original, or nonstandard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or nonstandard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Motif Compression

Figure 3: The 3 selected motifs for QM9.

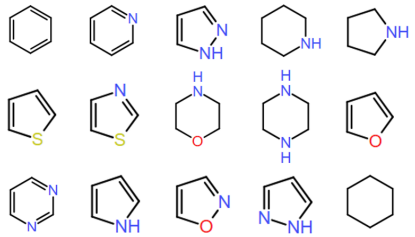


Figure 4: The 15 selected motifs for MOSES.

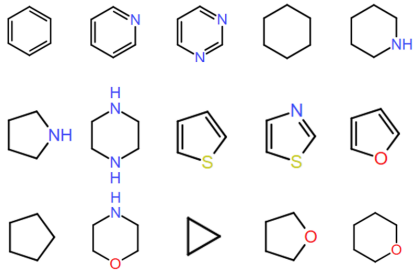


Figure 5: The 15 selected motifs for GUACAMOL.

Since each dataset has a different size of graph and complexity, we choose a different number of motifs, K to be converted to supernodes. We select 3 motifs for QM9 3, 15 motifs for MOSES 4, and 15 motifs for GUACAMOL 5.

B DiGress vs DMol

We now further compare DiGress and DMol. First, we observe that DiGress also constructs state transition matrices based on marginal distributions, with the key difference with respect to DMol being that its transition matrices vary over time as

$$\mathbf{Q}_X^t = \alpha^t \mathbf{I} + \beta^t \mathbf{1}_a \mathbf{m}_x, \quad \mathbf{Q}_E^t = \alpha^t \mathbf{I} + \beta^t \mathbf{1}_b \mathbf{m}_e,$$

where α^t and β^t are time-dependent scaling parameters, while $\mathbf{Q}_X^t$ and $\mathbf{Q}_E^t$ are the state transition matrices used to transition from $\mathcal{G}^{t-1}$ to $\mathcal{G}^t$. However, since the forward noise adding process is Markovian, DiGress directly utilizes $\overline{\mathbf{Q}}_X^t$ and $\overline{\mathbf{Q}}_E^t$ to achieve the transition from $\mathcal{G}^0$ to $\mathcal{G}^t$, where $\overline{\mathbf{Q}}_X^t = \mathbf{Q}_X^1 \dots \mathbf{Q}_X^t$ and $\overline{\mathbf{Q}}_E^t = \mathbf{Q}_E^1 \dots \mathbf{Q}_E^t$.

Since $(\mathbf{1}\mathbf{m})^2 = \mathbf{1}\mathbf{m}$, we have

$$\overline{\mathbf{Q}}_X^t = \overline{\alpha}^t \mathbf{I} + \overline{\beta}^t \mathbf{1}_a \mathbf{m}_x, \overline{\mathbf{Q}}_E^t = \overline{\alpha}^t \mathbf{I} + \overline{\beta}^t \mathbf{1}_b \mathbf{m}_e,$$

where $\overline{\alpha}^t = \prod_{\tau=1}^t \alpha^\tau$ and $\overline{\beta}^t = 1 - \overline{\alpha}^t$. Additionally, DiGress also adopts a cosine schedule for $\overline{\alpha}^t$: $\overline{\alpha}^t = \cos^2(0.5\pi(t/T + c)/(1 + c))$, where $c > 0$ is a constant.

In DMol, for a graph dataset with n nodes, the number of nodes undergoing state changes at time step t equals $N(t) = (1-\alpha)\, n$. At the same time step in Digress, the expected number of nodes undergoing state changes equals $\sum_i np_i^x(1-\alpha)(1-p_i^x)$, where p_i^x represents the marginal probability of node class i. Therefore, at the same time step, the number of nodes modified by DMol is $\frac{1}{\sum_i p_i^x(1-p_i^x)}$ times that expected from Digress (the derivations can be found in Appendix D.2). Moreover, among the nodes whose states are changed by DMol, the proportion of nodes with class i is p_i^x, whereas in DiGress, this proportion equals $p_i^x(1-p_i^x)$. The ratio of these fractions equals $1-p_i^x$.

For edges, a similar analysis can be applied, leading to the conclusion that at the last time step T, the ratio of the number of edges modified by our noise model and that modified by Digress equals $\frac{r}{\sum_j p_j^e(1-p_j^e)}$, where r is the hyperparameter mentioned in Section 3.2. At each time step, this ratio needs to be adjusted by a correction factor that accounts for the fact that our method only selects edges from the complete graph induced by the chosen nodes. This correction factor is the ratio of the number of edges in the induced subgraph and the total number of edges. Additionally, among the modified edges, the proportion of edges with class index j differs from the same in DiGress by a constant factor of $1-p_j^e$, where p_j^e equals the marginal probability of edge class j. As a result, DMol can be made to replicate the average "behavior" of DiGress by simply adjusting specific hyperparameters, whereas the opposite claim is not true. Comparisons of the noise evolution distribution, efficiency and expressivity of DMol and Digress are available in Appendix D.

The following modifications can be applied to DMol to obtain Digress:

1. Fix the maximum number of sampling steps to a constant length instead of varying based on the number of nodes.

2. Scale the number of nodes to be changed at each step by a factor of $\sum_i p_i^x(1-p_i^x)$.

3. Set the noising processes for edges and nodes to be independent, i.e., select edges from the entire graph instead of restricting to the subgraph formed by the selected nodes.

4. Scale the number of edges to be changed at each step by a factor of $\frac{\sum_j p_j^e(1-p_j^e)}{r}$.

5. When generating random scores for nodes and edges to determine the objects to be selected, multiply the random scores for nodes by a correction weight of $1-p_i^x$, and multiply the random scores for edges by $1-p_j^e$.

These modifications are straightforward to implement in practice and can be achieved by simply adjusting model hyperparameters. However, Digress cannot be easily converted into DMol.

C Likelihood Computation

DMol can be visualized through the model depicted in Figure 6, informed by the dataset distribution and where nodes are perturbed first, while edges are conditionally perturbed based on the node selection. This framework extends traditional hierarchical variational inference structures by incorporating a crucial dependency relationship: the selection of edges to perturb is conditioned on the selection of nodes at each timestep.

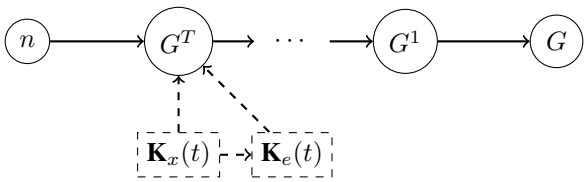


Figure 6: The conditional noise-addition model of DMol, where $\mathbf{K}_x(t)$ captures the random node selection while $\mathbf{K}_e(t)$ captures the random edge selection process; the dashed arrow indicating the dependency of edge selection on node selection.

For a molecular graph G, the model likelihood must account for this conditional relationship, according to:

$$\log p_\theta(G) = \log \sum_{n \in \mathcal{N}} p(n) \int p(G^T|n) p_\theta(G^{T-1}, \ldots, G^1|G^T) p_\theta(G|G^1) d(G^1, \ldots, G^T) \tag{2}$$

$$= \log p(n_G) + \log \int p(G^T|n_G) \prod_{t=2}^{T} p_\theta(G^{t-1}|G^t, \mathbf{K}_x(t), \mathbf{K}_e(t)) p_\theta(G|G^1) d(G^1, \ldots, G^T) \tag{3}$$

Note the key difference in the conditional probability $p_\theta(G^{t-1}|G_t, \mathbf{K}_x(t), \mathbf{K}_e(t))$, which explicitly accounts for the selection matrices $\mathbf{K}_x(t)$ and $\mathbf{K}_e(t)$ that designate which nodes and edges receive perturbations at time t.

Following established variational inference principles, we derive a tractable evidence lower bound (ELBO) adapted for this conditional structure:

$$\log p_\theta(G) \geq \log p(n_G) + \underbrace{D_{KL}[q(G^T|G) \| q_{\mathcal{X}}(n_G) \times q_{\mathcal{E}}(n_G)]}_{\text{Prior term}} \tag{4}$$

$$+ \underbrace{\sum_{t=2}^{T} \mathcal{L}_t}_{\text{Diffusion component}} \tag{5}$$

$$+ \underbrace{E_{q(G^1|G)}[\log p_\theta(G|G^1)]}_{\text{Reconstruction component}} \tag{6}$$

Where the diffusion component $\mathcal{L}_t$ now explicitly models the conditional relationship between node and edge noise:

$$\mathcal{L}_t(G) = E_{q(G^t|G)} \left[D_{KL} \left[q(G^{t-1}|G^t, G, \mathbf{K}_x(t-1), \mathbf{K}_e(t-1)) \| p_\theta(G^{t-1}|G^t, \mathbf{K}_x(t-1), \mathbf{K}_e(t-1)) \right] \right] \tag{7}$$

This can be further decomposed to highlight the conditional structure:

$$\mathcal{L}_t = \sum_{i \in K_x(t-1)} E_{q(x_i^t|x_i)} \left[D_{KL} \left[q(x_i^{t-1}|x_i^t, x_i) \| p_\theta(x_i^{t-1}|G^t) \right] \right] \tag{8}$$

$$+ \sum_{(i,j) \in K_e(t-1)} E_{q(e_{ij}^t|e_{ij})} \left[D_{KL} \left[q(e_{ij}^{t-1}|e_{ij}^t, e_{ij}) \| p_\theta(e_{ij}^{t-1}|G^t, \mathbf{K}_x(t-1)) \right] \right] \tag{9}$$

Note the critical difference in the edge term, where the predicted probability $p_\theta(e_{ij}^{t-1}|G^t, \mathbf{K}_x(t-1))$ is explicitly conditioned on the node selection $\mathbf{K}_x(t-1)$. This formulation captures the key aspect of DMol's noise process: edges are selected for perturbation only from within the subgraph induced by the selected nodes.

The conditional dependency structure provides two key advantages for DMol: 1. It preserves local structural coherence (e.g, motifs) by ensuring that edge modifications are correlated with node changes. 2. It significantly reduces the computational complexity by restricting edge perturbations to smaller subgraphs.

In practice, each ELBO component remains computationally tractable: we calculate $\log p(n_G)$ from the empirical node count distributions across the molecular corpus. The prior term involves categorical KL divergence calculations, while the diffusion component now incorporates the conditional selection of edges based on the node selection. The reconstruction component derives from generated probabilities of the original structure given its slightly perturbed version G^1.

D Theoretical Analysis of DMol

D.1 Noise Distribution Evolution

In DiGress, the node type distribution remains fixed at all forward time steps because the initial node type distribution is the marginal distribution $\mathbf{m}_X$. This is easy to see because $p_0 = \mathbf{m}_X$, so $p_1 = \overline{\alpha}^1 p_0 + \overline{\beta}^1 \mathbf{m}_X = \mathbf{m}_X$, and by induction, $p_t = \mathbf{m}_X$ for all t, where p_t is the node type distribution at step t. In DMol, the node type distribution changes over time. For a node with type i selected for perturbation, the probability of transitioning to type $j \neq i$ is $P(j|i_{\text{selected}}) = \frac{p_X(j)}{1-p_X(i)}$. This leads to a different recurrence relation for the probability distribution over time.

D.2 Efficiency

To compare the efficiency of DMol and DiGress, we compare the time step increment required to change exactly one node(minimal noise) under both settings. In DMol, the number of nodes changing at time t is $N(t) = (1 - \alpha^t) \cdot n$. To find δt_{DMol} such that $\delta N(t) = 1$, we need

$$N(t + \delta t) - N(t) = 1$$

The number of nodes that change at time $t + \delta t$ equals

$$N(t + \delta t) = (1 - \alpha(t + \delta t)) \cdot n$$

For DMol, we use $T = kn$ and the cosine schedule for α,

$$\alpha(t) = \cos^2\left(\frac{0.5\pi(t/T + s)}{1 + s}\right).$$

Taking the derivative with respect to t and using a small angle approximation for small δt, we obtain:

$$\frac{d\alpha(t)}{dt} \approx -\frac{\pi \sin\left(\frac{\pi(t/T+s)}{1+s}\right)\cos\left(\frac{\pi(t/T+s)}{1+s}\right)}{T(1+s)} = -\frac{\pi \sin\left(\frac{2\pi(t/T+s)}{1+s}\right)}{2T(1+s)}$$

For small δt, we can approximate:

$$\alpha(t + \delta t) - \alpha(t) \approx \frac{d\alpha(t)}{dt} \cdot \delta t$$

Therefore,

$$N(t + \delta t) - N(t) = (\alpha(t) - \alpha(t + \delta t)) \cdot n \approx -\frac{d\alpha(t)}{dt} \cdot \delta t \cdot n$$

Setting this equal to 1 and solving for δt, we can have the value of δt_{DMol}:

$$\delta t_{DMol} = \frac{1}{-\frac{d\alpha(t)}{dt} \cdot n} = \frac{2T(1+s)}{\pi n \sin\left(\frac{2\pi(t/T+s)}{1+s}\right)}$$

For DiGress, we use a similar approach to find $\delta t_{Digress}$. First, the expected number of nodes changed by DiGress at time t equals:

$$N_{Digress}(t) = n \cdot (1 - \alpha(t)) \cdot \sum_i p_i^x \cdot (1 - p_i^x)$$

Following a similar derivation as for DMol, we arrive at

$$\delta t_{Digress} = \frac{1}{-\frac{d\alpha(t)}{dt} \cdot n \cdot \sum_i p_i^x \cdot (1 - p_i^x)} = \frac{2T(1+s)}{\pi n \sin\left(\frac{2\pi(t/T+s)}{1+s}\right) \cdot \sum_i p_i^x \cdot (1 - p_i^x)}$$

Then, the ratio of the minimum time steps of interest equals

$$\frac{\delta t_{Digress}}{\delta t_{DMol}} = \frac{1}{\sum_i p_i^x \cdot (1 - p_i^x)}$$

Because $\sum_i p_i^x \cdot (1 - p_i^x) < 1$ for any probability distribution, we have that

$$\frac{\delta t_{Digress}}{\delta t_{DMol}} > 1$$

This proves that $\delta t_{Digress} > \delta t_{DMol}$, or, in words, that DMol requires a smaller time step increment to change exactly one node. As a result, the number of steps required by DMol to change a node will be smaller than that of required by Digress, which implies that DMol performs graph perturbations more efficiently.

D.3 Expressivity

Both DiGress and DMol start with the same underlying node and edge type distributions, but differ in how efficiently they explore the space of possible graphs. DMol tends to produce more diverse graphs within its reachable set because it makes larger localized changes. The Hamming distance between the initial graph $\mathcal{G}_0$ and the graph at time t, $\mathcal{G}_t$, is approximately $d_H(\mathcal{G}_0, \mathcal{G}_t) \sim \beta^t \cdot n$. This Hamming distance captures the difference between the number of node/edge types in the two graphs. For DiGress, the expected Hamming distance is $\mathbb{E}[d_H(\mathcal{G}_0, \mathcal{G}_t)] \sim \beta^t \cdot n \cdot \sum_{i \in \mathcal{C}_X} p_X(i) \cdot (1 - p_X(i))$. Since $\sum_{i \in \mathcal{C}_X} p_X(i) \cdot (1 - p_X(i)) < 1$, we conclude that DMol explores the graph space more efficiently per diffusion step.

E The Metrics used in MOSES and GUACAMOL

The metrics used in MOSES and GUACAMOL dataset are defined as follows: The filter score measures the proportion of molecules that pass the same filters used to construct the test set. The FCD score measures the similarity between molecules in the training and test sets using embeddings learned by a neural network. SNN quantifies the similarity of a molecule to its nearest neighbor based on the Tanimoto distance. Scaffold similarity compares the frequency distributions of so-called Bemis-Murcko scaffolds. KL divergence measures the differences in the distributions of various physicochemical descriptors.

F ChEMB Likeness, QED Scores, and Shingle Distance

The ChEMBL-Likeness Score (CLscore) Bühlmann and Reymond [2020] is a metric used to assess the drug-likeness of a molecule. It is determined by analyzing how many of the molecule's substructures are present in the ChEMBL datasetGaulton et al. [2016], a database of bioactive molecules with drug-like properties. Essentially, the CLscore quantifies the likelihood of a molecule being drug-like based on its substructure similarity to known compounds in ChEMBL.

Quantitative Estimation of Drug-likeness (QED) Bickerton et al. [2012] is a computational metric used in drug discovery to assess how "drug-like" a compound is. It combines eight physicochemical properties, including molecular mass (M_r), octanol–water partition coefficient (ALOGP), number of hydrogen bond donors (HBDs), number of hydrogen bond acceptors (HBAs), molecular polar surface area (PSA), number of rotatable bonds (ROTBs), number of aromatic rings (AROMs), and number of structural alerts (ALERTS), into a single score ranging from 0 (least drug-like) to 1 (most drug-like). The score is measured through defined desirability functions of the eight properties. QED provides a balanced evaluation by weighting these properties based on their distributions in known drugs.

The shingles distance (SD) is a metric designed to quantify the structural divergence between generated molecules and a reference dataset. It is computed by comparing the frequencies of important, predefined molecular drug substructures (motifs), known in RdKit as shingles, according to the ChEMBL dataset Gaulton et al. [2016]. The same set of shingles is used in the computation of the ChEMBL-Likeness Score (CLscore) Bühlmann and Reymond [2020]. SD measures the cosine distance between the vectors of shingle occurrence counts for two molecular distributions. By emphasizing differences in the presence of key drug-relevant motifs, SD offers a more nuanced assessment of both validity and novelty.

G Supplementary Experiments

G.1 General Graph Generation

We conducted experiments using the benchmark proposed by Martinkus et al. [2022a], which comprises two datasets: SBM and Planar. Each dataset consists of 200 graphs. DMol's ability to accurately model various properties of these graphs was assessed using metrics such as the degree distribution (Deg.), clustering coefficient distribution (Clus.), and orbit count distribution (Orb., the number of occurrences of substructures with 4 nodes), measured by the relative squared Maximum Mean Discrepancy (MMD).

Table 5: MMD Performance Comparison on SBM and Planar Graphs.

DATASET	MODEL	DEG ↓	CLUS ↓	ORB ↓	VALID ↑
SBM	GRAPHRNN	6.7	1.6	3.2	5.2
	GRAN	14.3	1.7	2.0	25.5
	SPECTRE	1.9	1.7	1.5	100.0
	DIGRESS	1.7	1.6	1.7	66.7
	DMOL(OURS)	1.6	1.5	1.5	66.1
PLANAR	GRAPHRNN	24.6	8.8	2534.0	0.0
	GRAN	3.6	1.2	1.8	98.2
	SPECTRE	2.6	2.5	2.5	100.0
	DIGRESS	1.6	1.5	1.6	84.5
	DMOL(OURS)	1.8	1.6	1.4	83.9

Table 6: Ablation Study

METHOD	V ↑	U ↑	N ↑
ONLY CE LOSS	79.6	100	100
BOTH CE & MSE LOSS	85.4	100	100
SELECTING EDGES FROM THE WHOLE GRAPH	55.6	100	100
SELECTING EDGES FROM THE INDUCED SUBGRAPH	85.4	100	100

MMD measures the discrepancy between two sets of distributions. The relative squared MMD is defined as follows:

$$\frac{\text{MMD}^2\left(\mathcal{G}_{gen} \| \mathcal{G}_{test}\right)}{\text{MMD}^2\left(\mathcal{G}_{train} \| \mathcal{G}_{test}\right)},$$

where $\mathcal{G}_{gen}$, $\mathcal{G}_{train}$, and $\mathcal{G}_{test}$ are the sets of generated graphs, training graphs, and test graphs.

In our experiments, we split the datasets into training, validation, and test sets with proportions of 64%, 16%, and 20%, respectively. The chosen hyperparameters were $k = 2$ and $r = 0.01$. The experimental results are presented in Table 5. These results demonstrate that DMol performs comparably to DiGress in generating general graphs.

G.2 Ablation Study

The ablation study highlights the performance improvements of DMol compared to DiGress due to the use of a new loss function that penalizes discrepancies in the counts of nodes and edges of training molecular graphs and sampled graphs; and performance improvements due to the use of special subgraph selection procedures. The results are summarized in Table 6. The model we used is DMol, and the dataset is GUACAMOL. In Table 6, "ONLY CE LOSS" stands for using only the cross-entropy loss (the loss used by DiGress), while "BOTH CE & MSE LOSS" refers to using both cross-entropy loss and mean squared error loss (the loss used by DMol). "SELECTING EDGES FROM THE WHOLE GRAPH" means selecting edges from the entire graph, whereas "SELECTING EDGES FROM THE INDUCED SUBGRAPH" means selecting edges only from the induced subgraph formed by the selected nodes. The results show that incorporating the MSE loss can improve the validity of the generated molecules. This is because the MSE loss penalizes discrepancies between the number of altered nodes and edges and the noise added during the forward process, resulting in more accurate predictions by the denoising model. Additionally, selecting edges from the induced subgraph significantly enhances the validity of the generated molecular graphs. This is achieved by making the sets of altered nodes and edges codependent. However, if edges are selected from the whole graph, the reduced number of diffusion steps significantly increases the learning difficulty for the denoising model, leading to lower validity in the generated molecular graphs.

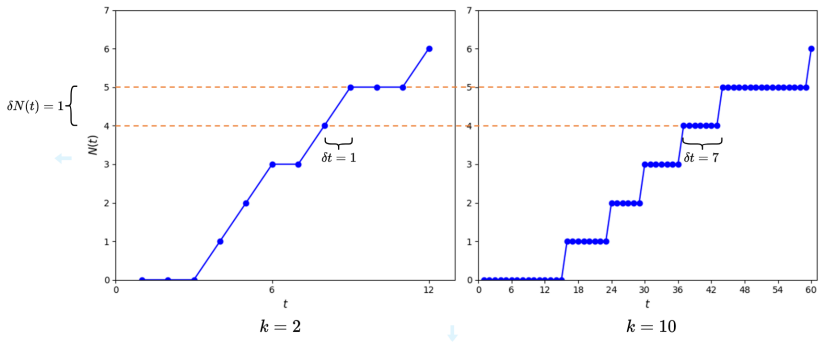


Figure 7: Choosing a larger value of k results in a less step-like (flatter) $N(t)$ function. As shown in the two figures, using $k = 10$ leads to a more skewed distribution of δt toward $N(t)$. This means that during sampling, the model is more likely to select t during the early diffusion stages, which makes it harder for the neural network model to learn how to denoise.

Table 7: DMol with different values of k on MOSES dataset

k	VALIDITY ↑	UNIQUENESS ↑	NOVELTY ↑	SAMPLE TIME ↓
1	85.1	100	100	2.53
2	**85.6**	100	100	5.28
3	84.9	100	100	7.81
4	84.7	100	100	10.37
5	84.4	100	100	12.69
10	83.4	100	100	24.76

G.3 Selection of the parameter k

The hyperparameter k determines the total number of diffusion steps since we set $T = kn$, where n is the number of nodes. In practice, k is typically set to 1 or 2 to reduce the number of diffusion steps and enhance performance (as explained in Figure 7). The latter observation can be intuitively explained as follows: the number of nodes modified at each time step increases over time, so that by the final time step, all nodes are changed. As k increases, the x-axis of the diffusion process is effectively stretched by a factor of k, while the y-axis remains unchanged (Figure 7). Consequently, the number of diffusion steps required to modify a single node increases. This leads to a higher frequency of generated samples during the early stages of diffusion, when fewer nodes are altered; furthermore, fewer samples are generated during the "middle stages." However, these additional early-stage samples introduce redundancy during the process of training the denoising network, ultimately degrading performance. Furthermore, these samples are generated at the cost of samples with a larger number of node changes, further degrading the performance.

In addition, to empirically evaluate the impact of k on the performance of the diffusion model, we conducted experiments for different values of k. As shown in Table 7, the sample time is approximately proportional to k. Moreover, setting $k = 2$ yields the highest validity.

G.4 Complete Experimental Results for the QM9 Dataset

For complete performance comparison on the QM9 dataset, please refer to Table 8.

G.5 Forward Process Schedule

To examine the impact of the cosine schedule on different molecular scaffolds and topologies, we conducted an ablation study comparing different values of the hyperparameter c in the cosine schedule $\alpha = \cos^2(0.5\pi(t/T+c)/(1+c))$, as well as comparing cosine versus linear schedules on the MOSES dataset. The results are presented in Table 9.

Table 8: Performance comparison on the QM9 dataset.

MODEL	V↑	V.U.↑	V.U.N.↑
GRAPHVAE	56.1	42.8	26.5
GT-VAE	74.8	16.5	15.6
SET2GRAPHVAE	60.0	56.8	-
SPECTRE	87.5	31.4	29.0
GRAPHNVP	83.5	18.4	-
GDSS	96.0	94.6	-
GRUM	**99.4**	85.1	22.2
DIGRESS	97.8	95.2	31.8
DISCO	98.2	95.6	57.5
DEFOG	97.9	95.9	62.8
DMOL(OURS)	98.3	**96.0**	**75.1**

Table 9: Ablation study for forward process schedules on the MOSES dataset. The cosine schedule with $c = 0.008$ achieves optimal performance across all metrics.

Method	Validity↑	Uniqueness↑	Novelty↑	Filters↑	FCD↓	SNN↑	Scaf↑	ChEMBL↑	QED↑
cos ($c = 0.004$)	86.9	100	100	97.4	1.13	0.57	14.2	4.5022	0.8022
cos ($c = 0.006$)	87.5	100	100	97.7	1.12	0.57	14.6	4.5020	0.8037
cos ($c = 0.008$)	**87.8**	**100**	**100**	**97.8**	**1.12**	**0.58**	**14.8**	**4.5033**	**0.8055**
cos ($c = 0.01$)	87.3	100	100	97.0	1.19	0.55	14.3	4.5018	0.8042
Linear	85.2	99.8	99.9	96.1	1.19	0.56	14.2	4.5011	0.8016

The experimental results demonstrate that varying the value of c has a relatively small impact on crucial performance metrics, with an optimal performance achieved at $c = 0.008$, which is the setting used in our experiments. This finding aligns with the findings by DiGress, which also adopts $c = 0.008$ as the default value.

When comparing cosine and linear schedules, we see that the cosine schedule clearly outperforms the linear schedule. This advantage stems from the fact that the cosine schedule adds less noise per timestep during both the initial and final phases of the diffusion process, compared to intermediate stages. This distribution better aligns with the learning characteristics of diffusion models. Specifically, maintaining relatively small noise levels during the early and final diffusion phases facilitates the model's learning of fine-grained data features and ensures "smoother" convergence. In contrast, the uniform noise distribution in linear schedules may prove either overly aggressive or overly conservative at certain stages, leading to compromised training efficiency and performance.

G.6 Scaffold-Constrained and Conditional Generation

To avoid issues associated with direct conditional generation, we instead proposed a scaffold-informed pipeline. The key ideas are to cluster the training set molecules according to either their RDKit features, or embeddings generated by Mol2Vec or Grover Jaeger et al. [2018], Rong et al. [2020], and then run DMol on the samples in sufficiently large cluster groups separately. More details are provided below.

RDKit-Based Scaffold Classification: Molecules are categorized into four scaffold classes following RDKit recommendations: aromatic monocyclic, aromatic monocyclic heterocycle, fused bicyclic, and aromatic heterocyclic.

GNN, Transformer and Other Embedding Classification Methods: One can use Mol2Vec or Grover to perform molecular graph embeddings, and follow up with Kmeans++ clustering.

The detailed analysis of the pros and cons of this approach compared to classical conditional generation will be discussed in more detail in a companion paper.

G.7 Motif Probability Distribution Analysis

Table 10 presents the complete per-motif probabilities for the top 30 most frequent motifs in the MOSES dataset. Table 11 shows the marginal distributions of node and supernode categories.

Table 10: Comparison of motif probabilities between training and generated sets on the MOSES dataset. IDs $0-14$ correspond to supernodes, while IDs $15-29$ are nonsupernode motifs.

ID	Motif (SMILES)	Prob on Training Set	Prob on Generated Set
	Supernode Motifs (IDs $0-14$*)*		
0	c1ccccc1	0.7753	0.7861
1	c1ccncc1	0.1829	0.1630
2	c1cnnc1	0.1019	0.0912
3	C1CCNCC1	0.0852	0.0783
4	C1CCNC1	0.0798	0.0662
5	c1cscc1	0.0763	0.0740
6	c1ccsn1	0.0715	0.0686
7	C1COCCN1	0.0625	0.0651
8	C1CNCCN1	0.0620	0.0681
9	c1ccoc1	0.0573	0.0429
10	c1cncnc1	0.0559	0.0512
11	c1cncn1	0.0464	0.0416
12	c1ncon1	0.0459	0.0458
13	c1ncnn1	0.0453	0.0415
14	C1CCCCC1	0.0351	0.0301
	Nonsupernode Motifs (IDs 15-29)		
15	C1CC1	0.0298	0.0242
16	c1cnoc1	0.0289	0.0238
17	C1CCCC1	0.0246	0.0281
18	C1CCOC1	0.0231	0.0241
19	c1ccnc1	0.0214	0.0191
20	c1cCNCC1	0.0202	0.0169
21	c1ccnnc1	0.0191	0.0123
22	c1nnnn1	0.0177	0.0161
23	c1nncs1	0.0164	0.0143
24	c1cnccn1	0.0154	0.0127
25	c1cnnn1	0.0151	0.0104
26	c1cocn1	0.0146	0.0088
27	c1nnco1	0.0135	0.0090
28	c1cCCCC1	0.0123	0.0098
29	c1cNCC1	0.0114	0.0102
	Average Probabilities		
Top 15 (Supernodes) mean		0.1189	0.1143
Last 15 (Nonsupernodes) mean		0.0189	0.0160

H Examples of Generated Molecules.

Figure 8 illustrates the sampling process.

Figure 9, 10, 11 shows examples of generated molecules.

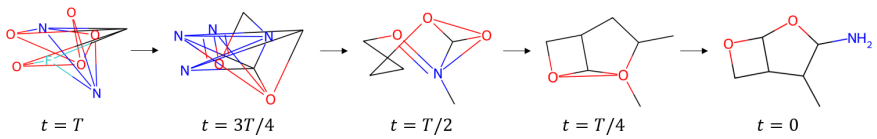


Figure 8: Sampling process.

Table 11: Marginal distributions of node and supernode categories in the MOSES dataset. Supernodes are treated as new node types alongside individual atoms.

Node / Supernode Type	Training Set	Generated Set
Individual Atoms		
C	0.4897	0.4940
N	0.1511	0.1506
S	0.0170	0.0167
O	0.1654	0.1644
F	0.0246	0.0243
Cl	0.0093	0.0091
Br	0.0025	0.0026
Supernode Motifs		
c1ccccc1	0.0736	0.0750
c1ccncc1	0.0132	0.0130
c1cnnc1	0.0081	0.0077
C1CCNCC1	0.0068	0.0070
C1CCNC1	0.0049	0.0045
c1cscc1	0.0055	0.0052
c1ccsn1	0.0056	0.0050
C1COCCN1	0.0035	0.0036
C1CNCCN1	0.0036	0.0031
c1ccoc1	0.0041	0.0039
c1cncnc1	0.0044	0.0040
c1cncn1	0.0010	0.0008
c1ncon1	0.0027	0.0025
c1ncnn1	0.0024	0.0020
C1CCCCC1	0.0010	0.0008

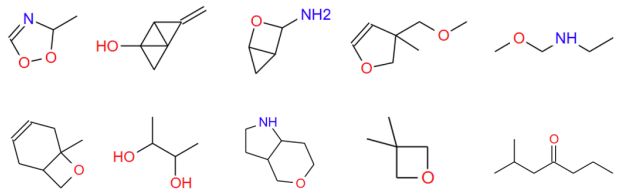


Figure 9: Molecular graphs sampled from DMol trained on the QM9 dataset.

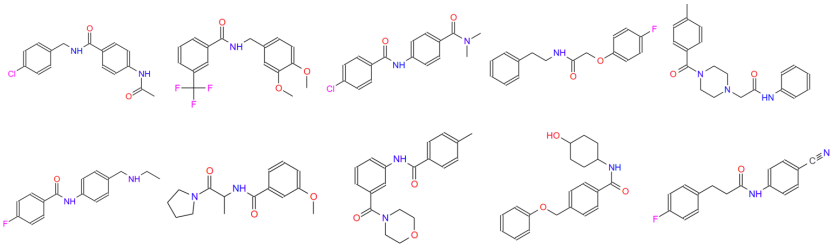


Figure 10: Molecular graphs sampled from DMol trained on the MOSES dataset.

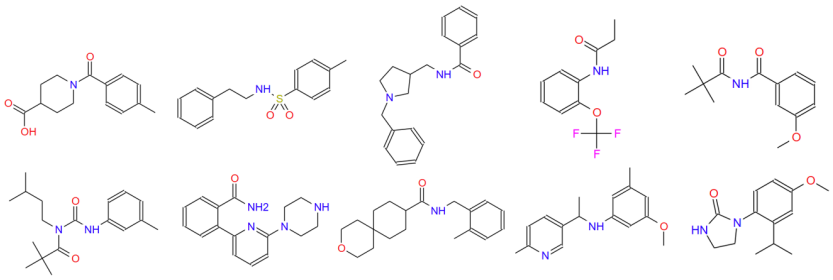
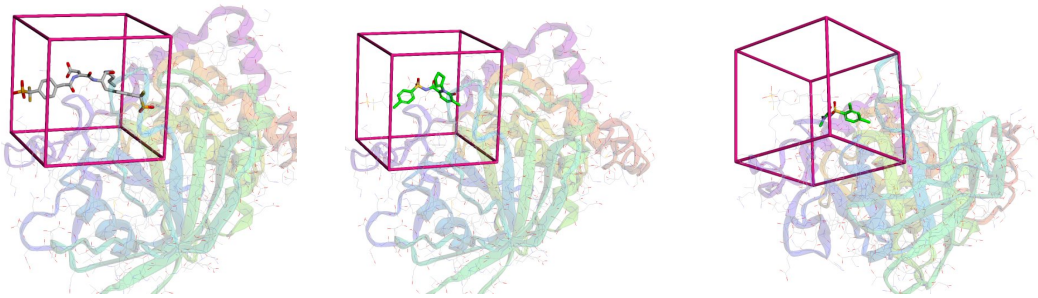


Figure 11: Molecular graphs sampled from DMol trained on the GUACAMOL dataset.

I Sample Docking Results

We used AutoDock Vina version 1.2.0 Goodsell and Olson [1990] to visualize the docking of two of the top-scoring generated molecules (with respect to the ChEMBL likeness) that also exhibit strong biding affinities on the 1PXH and 4MQS proteins when compared to reference drug ligants (the reason behind the selection of these proteins is confidential information). The 4MQS complex represents the active human M2 muscarinic acetylcholine receptor bound to the antagonist iperoxo, while 1PXH represents the protein tyrosine phosphatase 1B with bidentate inhibitor compound 2. The free energy results are shown in Figures 12 and 13, and they indicate that the generated molecules can significantly outperform or underperform existing reference drugs.

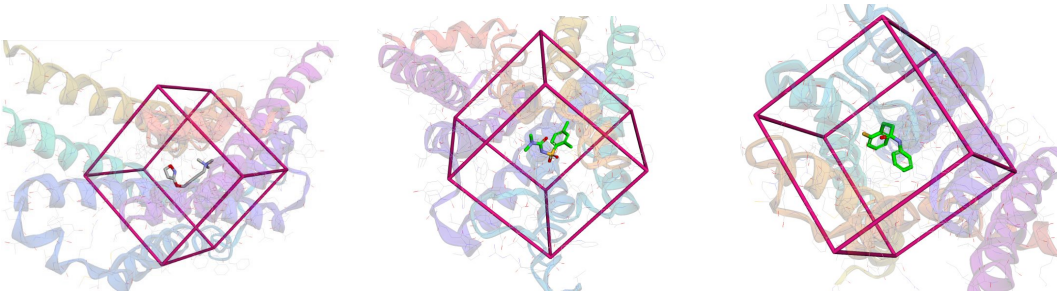


(a) Reference: Free energy = -8.7 kcal/mol

(b) Generated drug 1: Free energy = -6.4 kcal/mol

(c) Generated drug 2: Free energy = -6.0 kcal/mol

Figure 12: The reference molecule (a) and two of our generated molecules (b) and (c) docked at the protein receptor 1PXH, and the corresponding free energies. The generated molecules significantly underperform with respect to the reference due to their smaller sizes/masses. As a result, molecular weight constraints have to be considered as part of the design process.



(a) Reference: Free energy = -6.4 kcal/mol

(b) Generated drug 1: Free energy = -6.6 kcal/mol

(c) Generated drug 2: Free energy = -9.1 kcal/mol

Figure 13: The reference molecule (a) and two of our generated molecules (b) and (c) docked at the protein receptor 4MQS, and the corresponding free energies. The generated molecules significantly outperform with respect to the reference, and the sizes/masses of the drug molecules are comparable. This confirms the importance of focusing on the right molecular weight/size.

J Broader Impact

This work presents DMol, an efficient molecular graph generation framework with a strong potential for practical applications in accelerating drug discovery, designing novel materials, and advancing computational chemistry. By substantially reducing computational requirements while maintaining or improving molecular quality metrics, our approach could democratize access to molecular design tools, enabling researchers with limited computational resources to engage in this field. The efficiency gains may also enable exploration of larger chemical spaces, potentially leading to discoveries that address pressing challenges in healthcare, energy storage, and environmental remediation. However, we acknowledge that molecular generation technologies could potentially be misused for designing harmful substances if deployed without appropriate safeguards. Additionally, as with many other AI systems, there is risk of amplifying biases present in training data, which could limit the diversity of generated molecules or reproduce historical biases in pharmaceutical development. We encourage future work to address these concerns through development of responsible use protocols, integration of toxicity and safety prediction tools, and careful curation of training datasets to ensure equitable representation across chemical domains. The machine learning community should collaborate with domain experts in chemistry, biology, and ethics to ensure that advances in molecular generation technologies maximize societal benefit while minimizing potential harms.

K Limitations

Despite the many documented advantages of DMol in terms of efficiency and molecular quality, several limitations remain. First, our approach still struggles with generating certain complex motif structures and stereochemistries, which are crucial for therapeutic applications. Second, while we achieve high validity scores, we do not explicitly enforce chemical constraints during generation, occasionally producing molecules that are formally valid but synthetically impossible to bring into existence. Third, our evaluation focuses primarily on small drug-like molecules; the performance on larger biomolecules, polymers, or metal-organic frameworks remains unexplored. Finally, the model's adaptability to conditional generation tasks (e.g., optimizing for specific target properties) requires additional architectural modifications that may affect the established efficiency gains. Future work should address these limitations to further bridge the gap between computational generation and practical chemical synthesis. Most importantly, this and all other related works should put more effort in identifying more comprehensive evaluation metrics for the generated molecules, since an overwhelming number of created samples cannot be synthesized or do not dock on any known protein.

L Future Work

Future work on DMol will focus on extending the model to conditional molecule generation, where specific chemical properties can be targeted through controlled diffusion processes. We also plan to address the remaining stereochemistry challenges by incorporating 3D structural information into the diffusion process. Additionally, we aim to develop more sophisticated motif compression strategies that can handle a wider variety of molecular scaffolds while preserving chemical feasibility. Exploring the application of DMol to larger biomolecular structures and integrating it with downstream property prediction models presents promising research directions. Finally, we intend to investigate dynamic scheduling of diffusion steps based on molecular complexity to further optimize computational efficiency.