Discriminator-Driven Diffusion Mechanisms for Molecular Graph Generation

Gerrit Großmann
Saarland Informatics Campus
DFKI
Germany
gerrit.grossmann@dfki.de

Abstract

Generating molecules that adhere to a specific distribution is a significant challenge in chemo-informatics. Diffusion models have emerged as a leading approach for data generation across various domains, including molecular graphs. However, their practicality is hampered by their computational demand and is largely dependent on the expressive capacity of the Graph Neural Network (GNN) backbone architecture. Taking inspiration from Generative Adversarial Networks (GANs), this study introduces an auxiliary component to diffusion models: a second discriminator neural network, which predicts the congruence of a generated sample with the data distribution. The discriminator navigates the diffusion process (e.g., by reinforcing molecular valency rules), thereby enhancing the sample quality. Preliminary results suggest an improvement in the correctness and novelty of the generated molecules, even when utilizing a fairly simple GNN backbone. Code is available at `github.com/XXX/MoleculeDiffusionGAN`.

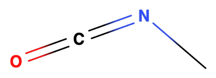
1 Introduction

In recent years, a new family of flow-based generative machine learning models has emerged. The fundamental concept involves transforming samples (e.g., a training set of molecules) through a *forward* process (i.e., *flow*), causing them to move around [1, 2]. Typically, they end in a normal distribution. The flow, which can be either stochastic (as in diffusion or score-based models) or deterministic (as in normalizing flows), and can be applied in either the continuous or discrete domain. Reversing the flow (i.e., following the *reverse* or *backward* process) provides the means to generate novel samples from the data distribution. A neural network can be utilized to guide the reverse process.

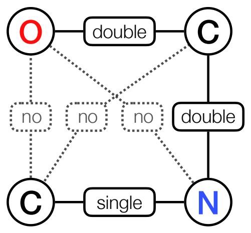
In *denoising* diffusion models, sample transformation (the forward process) occurs by incrementally adding Gaussian noise and shifting the samples toward the center, culminating in an standard normal distribution [3]. This approach has been demonstrated to be an effective strategy for synthesizing high-quality images. However, graph structures introduce additional complications. Specifically, it becomes significantly more challenging to navigate away from dead ends. Moreover, binary switches pose difficulties for the continuous conceptualization of diffusion.

We address this by introducing a secondary *discriminator* Graph Neural Network (GNN). Inspired by the GAN architecture [4], this GNN judges the congruence of a generated sample (specifically, the predicted endpoint) with the data distribution. During training, we remove the noise from a molecule with dual objectives: deceiving the discriminator and reconstructing the original training sample. We hypothesize that this architecture makes it easier to capture the complex constraints inherent in the data, for example, fulfilling molecular constraints such as valency rules.

Preprint. Under review.



(a)



(b)

$$\mathbf{X} = \begin{bmatrix} 1&0&0&1&0&0\\ 1&1&0&0&0&0\\ 1&0&0&0&1&0\\ 1&1&0&0&0&0\\ -1&0&0&1&0&0\\ -1&0&0&1&0&0\\ -1&0&1&0&0&0\\ -1&1&0&0&0&0\\ -1&1&0&0&0&0\\ -1&1&0&0&0&0 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\}\text{Atoms} \\ \left.\begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix}\right\}\text{Bonds} \end{matrix}$$

(c)

Figure 1: Schematic overview of our molecular representation. (a) Example molecule: *methyl isocyanate*. (b) The molecule is transformed into a graph, where both atoms and bonds are represented as nodes. (c) All information is captured in the feature matrix $\mathbf{X}$. The first column indicates the type of the node, while the remaining columns provide the element (C, H, O, N, F) or the bond type (no bond, single, double, triple, aromatic).

Algorithm 1 Train epoch: GNN_{pred}

```
Require: $\text{GNN}_{\text{pred}}$, $\text{GNN}_{\text{disc}}$, $\gamma$, $\bar{\alpha}$
 1: for $\mathbf{X}_0$ in Training data do
 2:     $t \sim \text{Uniform}(\{1, \dots, T\})$
 3:     $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
 4:     $\mathbf{X}_t = \sqrt{\bar{\alpha}_t}\mathbf{X}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$
 5:     $\mathbf{X}_0^{\text{pred}} = \text{GNN}_{\text{pred}}(\mathbf{X}_t, t)$
 6:     $\mathcal{L}^{\text{pred}} = L_2(\mathbf{X}_0^{\text{pred}}, \mathbf{X}_0)$
 7:     if $\text{GNN}_{\text{disc}}$ is not None then
 8:         $\mathcal{L}^{\text{disc}} = L_2\big(1, \text{GNN}_{\text{disc}}(\mathbf{X}_0^{\text{pred}})\big)$
 9:     else
10:         $\mathcal{L}^{\text{disc}} = 0$
11:     end if
12:     $\mathcal{L} = (1 - \gamma)\mathcal{L}^{\text{pred}} + \gamma\mathcal{L}^{\text{disc}}$
13:     Optimize $\text{GNN}_{\text{pred}}$ w.r.t. $\mathcal{L}$.
14: end for
```

Algorithm 2 Train epoch: GNN_{disc}

```
1: for $\mathbf{X}_0$ in Training data do
2:     $\mathcal{L}^{\text{real}} = L_2(1, \text{GNN}_{\text{disc}}(\mathbf{X}_0))$
3:     $\mathbf{X}_0^{\text{fake}}$ = inference()
4:     $\mathcal{L}^{\text{fake}} = L_2(0, \text{GNN}_{\text{disc}}(\mathbf{X}_0^{\text{fake}}))$
5:     $\mathcal{L} = \mathcal{L}^{\text{real}} + \mathcal{L}^{\text{fake}}$
6:     Optimize $\text{GNN}_{\text{disc}}$ w.r.t. $\mathcal{L}$.
7: end for
```

Algorithm 3 Function inference()

```
Require: $\text{GNN}_{\text{pred}}$, $\sigma$, $\alpha$, $\bar{\alpha}$
1: $\mathbf{X}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
2: for $t = T \dots 1$ do
3:     $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
4:     $\epsilon_{\text{pred}} = \frac{(\text{GNN}_{\text{pred}}(\mathbf{X}_t, t) - \sqrt{\bar{\alpha}_t}\mathbf{X}_0)}{\sqrt{1 - \bar{\alpha}_t}}$
5:     $\mathbf{X}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{X}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_{\text{pred}}\right) + \sigma_t\mathbf{z}$
6: end for
7: return $\text{GNN}_{\text{pred}}(\mathbf{X}_1, 1)$
```

Figure 2: Schematic overview of the training procedure. Technical details are omitted for the sake of clarity. We alternately call **Algorithm 1** and **Algorithm 2** to train the model.

2 Method

Our method follows the most widely used and arguably simplest form of a generative diffusion model, introduced by Ho et al. in [3]. For the backbone of both the predictor (i.e., the denoiser) and the discriminator network, we utilize a straightforward GNN based on the PNA architecture [5].

Molecule Representation An n-node graph can be represented as a symmetric adjacency matrix $A \in \{0, 1\}^{n \times n}$, accompanied by a node-feature matrix $X \in \mathbb{R}^{n \times d}$. We represent each molecule as a graph, such that each atom and each covalent bond is encoded by a node. The feature matrix $\mathbf{X}$ encodes for each node whether it represents a bond and, if so, which type of bond (single, double, triple, aromatic, or no bond), or if it represents an atom and, if so, which type of element (we consider only four heavy elements). The adjacency matrix connects each atom-node with all other atoms nodes via a specific bond node indicating how/if the two atoms are connected (cf. Figure 1). This approach diverges from most other graph-based diffusion models (cf. Related Work), which often use a fully connected graph and additional edge features. We opted for our method due to its simplicity and because it is widely recognized that edge convolutions (i.e., message-passing aggregations of the bonds) yield good performance in prediction tasks. We omit hydrogen atoms and do not take chirality into account. Note that the graph structure $\mathbf{A}$ is the same for all molecules of the same size.

GNN Backbone We employ an off-the-shelf PNA architecture as our GNN backbone. The predictor/denoiser GNN takes the noisy graph as input (i.e., the node features that were transformed with the forward process) and the current timepoint (indicating the overall amount of added noise) and returns the denoised graph (i.e., the node features of the original graph). Since we represent the bonds as nodes, there is no need to alter the edges during the process. The discriminator network takes a molecular graph as input and outputs a probability, indicating how likely it is that this molecule belongs to the training data (i.e., is *real*) or was generated using the diffusion process (i.e., is *fake*).

Diffusion Overview The forward process is composed of T time steps. In each step, a small amount of noise is added to the node-feature matrix $\mathbf{X}_0 = \mathbf{X}$. Following the standard convention for adding noise in each step and starting at $t = 0$, one can derive a closed form solution for the forward process to jump directly to any $t \in \{1, \dots, T\}$ (cf. Algorithm 2 and Figure 3a). Our GNN predicts the starting point $\mathbf{X}_0^{\text{pred}}$ from a given sample ($\mathbf{X}_t$) and time point t. This constitutes a minor technical divergence from [3], who predict the noise. We have found that our approach generally yields better results and simplifies the incorporation of the discriminator loss.

A key component of a diffusion model is the scheduler, which determines the amount of noise added at each time point. In this work, we employ a simple linear noise scheduler, increasing the noise added at each step linearly. For the closed-form solution to the forward diffusion process, the noise is denoted as a variable $\bar{\alpha}_t$, representing the aggregate noise at timepoint t (cf. Algorithm 2 and [3]). It is noteworthy that the first column of $\mathbf{X}$ (the bond-or-atom indicator) is not subject to the diffusion and remains unchanged.

Discriminator Initially, we train the model the traditional way (i.e., without the discriminator). Once the model achieves data generation capabilities, we train the discriminator network. This discriminator receives a molecule as input and outputs the probability of the molecule either belonging to the training data or being generated by the model. Next, the discriminator is incorporated into the training process, and the loss is adjusted. Now, the predicted starting point, $\mathbf{X}_0^{\text{pred}}$ (i.e., the denoised molecule), is evaluated based on its proximity to the ground truth molecule and its ability to deceive the discriminator. It is important to note that the discriminator output does not depend on the specific ground truth molecule in question ($\mathbf{X}_0$), but on how well it matches the data distribution of all training samples.

Utilizing the discriminator comes with a caveat. The predicted starting point will always contain a slight degree of noise and will never perfectly align with the one-hot encoded samples from the data distribution. This discrepancy makes the discriminator's task of distinguishing generated samples too simple. One possible solution is to binarize the predicted starting point, but this complicates the computation of gradients. To address this issue, the discriminator is equipped with a preprocessing step that introduces a tiny amount of noise to every input it receives.

Inference To generate new data, we start by selecting a random sample from the training data, ensuring a fixed number of atoms (following the data distribution), and replace all entries, expect the first column, with Gaussian noise to obtain $\mathbf{X}_T$. The number of bonds will still be variable during the diffusion due to the *no bond* option. Subsequently, we iteratively predict the starting point $\mathbf{X}_0^{\text{pred}}$ and reintroduce slightly reduced noise (cf. Algorithm 2). While reintroducing noise to the predicted $\mathbf{X}_0^{\text{pred}}$, we sample from a *bridge* distribution, connecting $\mathbf{X}_0^{\text{pred}}$ and the current sample $\mathbf{X}_t$. The classical approach for this involves incorporating an additional variance variable σ [3]. Given that we predict the original starting point, minor adjustments to Algorithm 2 from [3] are necessary for computing the bridge distribution.

3 Results

We trained two models on the QM9 dataset [6], one with and the other without a discriminator. Throughout the training process, we periodically generated 40,000 molecules and evaluated their uniqueness (fraction of molecules that were only generated once) as well as their validity and

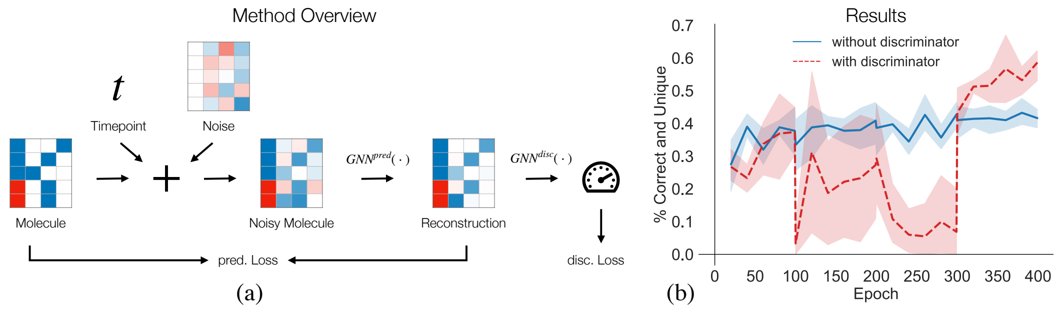


Figure 3: (a): Schematic overview of training procedure. (b): Quality of generated samples during training: The use of the discriminator starts at epoch 100, which accounts for the abrupt decline in quality evident in the red/dashed curve. Despite this initial setback, the quality notably rebounds and progressively improves. Conversely, the blue/straight curve demonstrates early stagnation in quality improvement.

connectivity using RDKit. The results are reported in Figure 3b. Further technical details about the architecture and hyperparameters can be found in the repository. We only re-trained the dicriminator every 100 epochs. The steep jumps indicate that there is still a lot of room to adjust the hyperparamters.

4 Related Work

A plethora of papers have employed machine learning techniques for molecule generation, creating a body of literature that is too extensive to fully review here. Generally, approaches can be categorized into those that predict text-based molecule representations, such as utilizing transformers and SMILES descriptions [7], and those that predict the molecular graph, potentially including 3D coordinates [8]. A range of techniques has been deployed in this domain, including reinforcement learning [9], autoencoders [10], GANs [11], and diffusion models. Generating structured data, such as graphs and molecules, using diffusion models presents notable challenges. For a comprehensive overview, we direct the reader to [12]. State-of-the-art methods in this sphere include DiGress [13] and TIDE [14], while for molecular point cloud generation, the diffusion model of [8] and GeoLDM [15] are notable.

The intersection of GANs and diffusion models appears to be a relatively underexplored territory in the literature. Some studies, such as [16], utilize the final output of diffusion processes and subject it to evaluation by a discriminator. As explored in [17], the output of a diffusion model can serve to shield a discriminator NN from adversarial attacks. Additionally, the work by [18] employs a diffusion process to generate noise, which is subsequently utilized in a GAN, thereby enhancing GAN training. They construct a discriminator conditioned on the timepoint, enabling it to accept images across all noise levels as input. Notably, [19] employ a discriminator, which is trained just once, to guide the inference process.

5 Conclusions and Future Work

We find that we can answer our hypothesis, that the guidance from a discriminator enhances the training of a molecular diffusion mechanism, affirmatively. Although our approach has not yet attained the level of current state-of-the-art results, it offers many options for further improvements and can be easily trained on a standard desktop computer. Furthermore, the underlying paradigm, being model-agnostic, can be applied to a wide range of diffusion and flow-based architectures. It would also be interesting to see how well this method performs on other data with many inherent constraints (such as solving Sudoku puzzles).

The immediate next step involves a more thorough evaluation, especially testing the impact of the discriminator weight, γ. Moreover, numerous other questions remain unexplored. A straightforward extension involves integrating the discriminator into the inference process and manipulating the predicted endpoint, similar to universal guidance [20]. Another viable approach would be to train a discriminator also for intermediate noise steps, using it directly as a guidance network as it is done in classifier-based conditional sampling.

References

[1] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 2022.

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

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[5] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

[6] 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.

[7] Viraj Bagal, Rishal Aggarwal, PK Vinod, and U Deva Priyakumar. Liggpt: Molecular generation using a transformer-decoder model. 2021.

[8] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pages 8867–8887. PMLR, 2022.

[9] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

[10] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[11] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[12] 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. *arXiv preprint arXiv:2304.01565*, 2023.

[13] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

[14] Maysam Behmanesh, Maximilian Krahn, and Maks Ovsjanikov. Tide: time derivative diffusion for deep learning on graphs. In *International Conference on Machine Learning*, pages 2015–2030. PMLR, 2023.

[15] Minkai Xu, Alexander S Powers, Ron O Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pages 38592–38610. PMLR, 2023.

[16] Nate Gruver, Samuel Stanton, Nathan C Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew Gordon Wilson. Protein design with guided discrete diffusion. *arXiv preprint arXiv:2305.20009*, 2023.

[17] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Animashree Anandkumar. Diffusion models for adversarial purification. In *International Conference on Machine Learning*, pages 16805–16827. PMLR, 2022.

[18] Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-gan: Training gans with diffusion. *arXiv preprint arXiv:2206.02262*, 2022.

[19] Dongjun Kim, Yeongmin Kim, Wanmo Kang, and Il-Chul Moon. Refining generative process with discriminator guidance in score-based diffusion models. *arXiv preprint arXiv:2211.17091*, 2022.

[20] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 843–852, 2023.