PROJECT REPORT: EFFICIENT GRAPH GENERATION WITH LATENT FLOW MATCHING

Ziqiao Wang & Haichen Huang
School of Computing
National University of Singapore
{E1374492, haichen}@u.nus.edu

1 INTRODUCTION

Recently, graph generation tasks have garnered significant attention across various areas, including molecular docking (Corso et al., 2023; Guan et al., 2023; Igashov et al., 2024) and Protein Modeling (Bose et al., 2024; Jing et al., 2024), where robust generative models are crucial for producing high-quality results. Among the most popular generative models today, Diffusion Generative Models (DGMs) (Sohl-Dickstein et al., 2015; Song et al., 2020; Ho et al., 2020) and generative Continuous Normalizing Flows (CNFs) with Conditional Flow Matching (CFM) methods (Lipman et al., 2023; Tong et al., 2024) have achieved remarkable progress.

Existing graph generation models based on DGMs are capable of learning complex distributions and handling challenging tasks; however, they often require dozens or even thousands of iterations during the inference process, leading to substantial computational costs. Additionally, DGMs for high dimensional data often possess complex model architectures and require long training process. Therefore, we aim to improve the efficiency of graph generation utilizing CNFs in latent spaces, which offer faster training and sampling while maintaining comparable quality(Liu et al., 2023; Dao et al., 2023; Tong et al., 2024).

In this project, we propose a demo of latent flow matching for graphs: we improved the Varational Autoencoder (VAE) (Kingma & Welling, 2022) provided by the class tutorial to encode graphs $p(G)$ into the latent space $q(z)$, and then use flow matching methods to model the transform between the noise distribution $N(0, I)$ and $q(z)$. In the sampling process, we start from a random noise and obtain the latent code $z \sim q(z)$ through solving the flow ODE, and then decode z with the pretrained VAE to generate a graph G. Compared with traditional DGMs or CNFs, modeling in the latent space can cut down computational cost and provide stabler sampling process (Rombach et al., 2022).

We also adapt high order ODE solvers (Liu et al., 2022; Lu et al., 2023) for the sampling process, reducing the number of function evaluations (NFEs) to 10$\sim$20, achieving an obvious acceleration compared with the DGMs provided in the tutorials.

2 BACKGROUND

2.1 VARIATIONAL GRAPH AUTOENCODER

The variational graph autoencoder (VGAE) (Kipf & Welling, 2016) embeds a given graph G into a latent space Z. Similar to VAE (Kingma & Welling, 2022), we simultaneously train an encoder and a decoder . For a given undirected, unweighted graph $G = (V, E)$ with $N = |V|$ as the number of nodes, we introduce the node feature vector $A \in \mathbb{N}_{\geq 0}^{N}$ and the edge feature matrix $W \in \mathbb{N}_{\geq 0}^{N\times N}$. We further introduce stochastic latent variables $Z \in \mathbb{R}^{dz}$ as a vector, where dz is the dimensionality of the latent space. We define the encoder and decoder as two graph neural networks, and we have:

$$q(z_i|A, W) = \mathcal{N}(z_i|\mu_i, \sigma_i^2) \tag{1}$$

Here $\mu = \text{encoder}_\mu(A, W)$ is the mean vector and $\log\sigma = \text{encoder}_\sigma(A, W)$ is the variance vector. We also have:

$$p(A|Z) = \text{decoder}_A(Z),\ p(W|Z) = \text{decoder}_W(Z) \tag{2}$$

Therefore, we optimize the variational lower bound $\mathcal{L}$ w.r.t. the variational parameters in the encoder and the decoder.

$$\mathcal{L} = \mathbb{E}_{q(Z|A,W)}\left[\log p(A, W|Z)\right] - KL\left[q(Z|A, W)||\, p(Z)\right] \tag{3}$$

2.2 Conditional Flow Matching

Flow models define a mapping between data samples $x_1 \sim p_1$ and prior samples $x_0 \sim p_0$ through an ordinary differential equation

$$\mathrm{d}x_t = v_\theta(x_t, t)\mathrm{d}t, \tag{4}$$

where $v_\theta(x_t, t)$ is a *time-dependent vector field* and θ are learnable parameters. With a chosen starting-point x_0 from the prior distribution p_0, the vector field generates a trajectory via the ODE and finally reach a data sample x_1 of the target distribution p_1. Conditional Flow Matching (CFM) (Lipman et al., 2023; Tong et al., 2024) is a kind of simulation-free training method for generative flows. The key of CFM is that the marginal vector field $u_t(x)$ that generates the marginal *probability path* $p_t(x)$ can be constructed by marginalizing over the conditional vector fields $u_t(x|z)$:

$$u_t(x) = \mathbb{E}_{q(z)}\left[\frac{u_t(x|z)p_t(x|z)}{p_t(x)}\right], \tag{5}$$

where $p_t(x|z)$ is the conditional probability path generated by $u_t(x|z)$. The observation enables us to design tractable $p_t(x|z)$ and $u_t(x|z)$, and regress v_θ via the CFM objective:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,p_t(x,z)}\|v_\theta(x, t) - u_t(x|z)\|^2. \tag{6}$$

3 Improved Graph VAEs

Due to the inherent nature of molecular bonds between atoms, we selected an anisotropic graph neural network as our autoencoder. Following Dwivedi & Bresson (2020), we compared the sparse transformer architecture and the dense transformer architecture for the molecule generation task. Contrary to the findings in Dwivedi & Bresson (2020), we observed that the dense transformer model performs significantly better than the sparse transformer model and runs faster after our optimization.

To achieve better performance, we adopted the architecture of Llama3 models (Dubey et al., 2024). Specifically, we removed the bias parameter from the linear modules in the multi-head attention mechanism and replaced the multilayer perceptrons with SwiGLU (Shazeer, 2020). Additionally, we incorporated parallel multi-head attention and batch norm (Ioffe, 2015) to improve efficiency.

3.1 Parallel Multi-head Attention

Instead of performing num_heads single-head attention operations sequentially in one MHA operation, we aggregate all single-head attention operations into a single parallel operation to accelerate model training. As described in 7, we combine all linear operations, and the chunk operation evenly splits the last dimension continuously.

$$Q_i = \text{Chunk}(W_Q h)_i,\ K_i = \text{Chunk}(W_K h)_i,\ V_i = \text{Chunk}(W_V h)_i \tag{7}$$

We then rearrange the storage format of the aggregated query, key, and value into $\hat{Q} \in \mathbb{R}^{B\times N\times V\times D}$, where B is the batch size, N is the number of heads, V is the number of nodes, and D is the model dimensionality. This allows us to calculate the attention scores in parallel. After the optimization, the dense model runs $5\times$ faster than the original.

$$\text{Atten score} = softmax\left(\frac{\sum \hat{E}\cdot\hat{Q}\cdot\hat{K}}{\sqrt{d_{head}}}\right) \tag{8}$$

3.2 Batch Norm

We replace the layer normalization (Ba, 2016) modules used for node hidden states in the transformer layers with batch normalization (Ioffe, 2015). Specifically, the node hidden states $H \in \mathbb{R}^{B\times V\times D}$ are rearranged into $\hat{H} \in \mathbb{R}^{(BV)\times D}$ and passed through the batch normalization module. After batch normalization, the hidden states $\hat{H}$ are transformed back into H. For edge hidden states E, we continue using RMS normalization(Zhang & Sennrich, 2019), as in the Llama3 models.

4 Latent Flow Matching

4.1 Optimal Transport Flow

In this project, we choose the optimal transport flow, namely rectified flow Lipman et al. (2023); Liu et al. (2023); Tong et al. (2024), as the generative model backbone:

$$x_t = tx_1 + (1-t)x_0, v_t = \frac{dx_t}{dt} = x_1 - x_0, \tag{9}$$

which constructs direct maps between data pairs. During the training process, we first encode the graph data $G(V, E)$ with the pretrained autoencoder Φ:

$$z = \Phi\left(G(V, E)\right). \tag{10}$$

And we then use OT-Flow to model the transform between the latent distribution $q(z)$ and noise distribution $\epsilon \sim N(0, I)$

$$z_t = tz + (1-t)\epsilon, t \in [0, 1], \tag{11}$$

with the objective

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t,p_t(z,\epsilon)} \|v_\theta\left(z_t, t, size\right) - (z - \epsilon)\|^2. \tag{12}$$

When it comes to sampling, we start with random noise and solve the flow ODE iteratively from $t = 0$ to $t = 1$:

$$\mathrm{d}z_t = v_\theta(z_t, t, size)\mathrm{d}t, \tag{13}$$

and use the corresponding decoder to generate graph data:

$$G(V, E) = \Phi^{-1}\left(z\right). \tag{14}$$

4.2 Fast Sampling

Since both DGMs and CNFs are generative models based on Differential Equations, they often require dozens or even hundreds of iterations during the inference process, which is the sampling speed bottleneck of the graph generation project. Here we provide two fast sampling methods to accelerate generation.

4.2.1 Linear Multistep Methods

Liu et al. (2022) have demonstrated the validity of pseudo numerical methods on DGMs, which has inspired us to improve the sampling process of flow with linear multistep methods. Given the sampling ODE

$$\mathrm{d}z_t = v_\theta(z_t, t)\mathrm{d}t, \tag{15}$$

we can reuse previous approximation results to correct the sampling steps with different orders:

$$z_{t+\delta} = z_t + \delta f_t^k \tag{16}$$

$$f_{t_i}^{(1)} = f_{t_i} = v_\theta(z_{t_i}, t_i) \tag{17}$$

$$f_{t_i}^{(2)} = \frac{3}{2} f_{t_i} - \frac{1}{2} f_{t_{i-1}} \tag{18}$$

$$f_{t_i}^{(3)} = \frac{1}{12}\left(23 f_{t_i} - 16 f_{t_{i-1}} + 5 f_{t_{i-2}}\right) \tag{19}$$

$$f_{t_i}^{(4)} = \frac{1}{24}\left(55 f_{t_i} - 59 f_{t_{i-1}} + 37 f_{t_{i-2}} - 9 f_{t_{i-3}}\right). \tag{20}$$

4.2.2 Dpm-flow-solver

Lu et al. (2022; 2023) have designed a kind of exponential integrator accelerate the sampling process of DGMs. They simplify the exact solution from s to t of diffusion ODEs into

$$x_t = \frac{\alpha_t}{\alpha_s} x_s - \alpha_t \int_{\lambda_s}^{\lambda_t} e^{-\lambda} \hat{\epsilon}_\theta \left(\hat{x}_\lambda, \lambda \right) \mathrm{d}\lambda, \tag{21}$$

with the "change of variable" $\lambda_t = \log\left(\alpha_t / \sigma_t\right)$ and the parameterization $s_\theta(x,t) = -\frac{\epsilon_\theta(x,t)}{\sigma_t}$. Based on such formulation, the linear part is exactly computed and they approximate the exponentially weighted integral by the Taylor expansion of $\hat{\epsilon}_\theta$. Lu et al. (2023); Zheng et al. (2023) have also proposed single-step and multi-step solvers based on data prediction $X_\theta(x,t)$ leveraging the expRK and Adams–Bashforth techniques.

We can transform the velocity prediction of OT-Flow into noise or data prediction simply with

$$v_\theta(x_t,t) = \frac{\beta_t'}{\beta_t} x_t + \beta_t [\frac{\alpha_t}{\beta_t}]' X_\theta(x_t,t) = \frac{\alpha_t'}{\alpha_t} x_t + \alpha_t [\frac{\beta_t}{\alpha_t}]' \epsilon_\theta(x_t,t), \tag{22}$$

where $\alpha_t = t$ and $\beta_t = 1 - t$, and then reuse Dpm-solver to accelerate sampling.

5 Experiments

Dataset	Model	NFEs	Validity(%)	Uniqueness(%)	Novelty(%)	U&N(%)
QM9-1k(provided)	VAE(reported)	1	78.4	86.9	93.1	\
	GAN(reported)	1	66.5	68.8	100.0	\
	Diffusion(reported)	150	41.5	100.0	100.0	\
	VAE(provided)	1	73.4‘	**96.4**	**97.9**	69.5
	iVAE	1	**82.9**	92.4	96.8	**74.6**
	Flow+VAE	20+1	78.5	**96.5**	**96.1**	72.8
	Flow+iVAE	20+1	**86.0**	93.0	94.3	**76.6**
QM9-10k	VAE(provided)	1	74.58	**90.81**	**95.75**	**65.74**
	iVAE	1	**81.0**	85.4	92.6	64.6
	Flow+VAE	20+1	77.25	**90.04**	**95.22**	**67.54**
	Flow+iVAE	20+1	**81.8**	85.3	92.3	65.1

Table 1: Results on QM9.

We have performed experiments on QM9 dataset with 1k and 10k training data. For provided VAEs, we increase the layer number of both encoder and decoder from 4 to 6 (19 million). We use DiT-S/8 (32.52 million) (Peebles & Xie, 2023) for flow matching. Our improved VAE has a 5x faster training speed than provided ones, with fewer parameters (13.71 million) but comparable performance, and achieves higher validity rates and better generation diversity. Combining flow matching and autoencoders, we can get better performance than VAE and faster training than diffusion models.

5.1 U&N score

$$\text{U\&N score} = \frac{\text{\# of unique and novel molecules}}{\text{\# of samples}} \tag{23}$$

In the evaluation, we selected various metrics to assess the quality of the molecules generated by the tested models. At the start, we randomly sampled 1,000 molecules from each model. We then calculated the percentage of valid molecules and the percentage of unique molecules for each model. The validity percentage indicates whether the model has the knowledge to generate correct molecular structures. To evaluate the model's ability to generate novel molecules, we also measured the percentage of novel molecules not included in the training set. Additionally, we designed the U&N score to compare the trained models. The U&N score, defined as the percentage of unique and novel molecules 23, provides a direct basis for model comparison.

Dataset	Method	NFEs	Order	Validity(%)	Uniqueness(%)	Novelty(%)	U&N(%)
QM9-10k	Linear Multistep Method	10	1	**82.15**	85.18	91.75	65.61
			2	80.87	86.11	**92.45**	65.00
			3	82.1	**86.26**	92.1	**65.98**
	Dpm-flow-solver		2	**84.91**	16.45	**92.64**	17.93

Table 2: Results of different samplers

5.2 Fast sampling

With Linear Multistep Method, we can generate high quality samples with merely 10 20 NFEs. Compared with diffusion models provided, our generation architecture is more efficient and stable. Here we use 2-order and 3-order samplers because the higher order samplers may lead to instability Wizadwongsa et al. (2023). Dpm-flow-solver has better validity but poor uniqueness, we assume that this is because the variance of $q(z)$ does not strictly equals 1 and samples of Dpm-flow-solver tend to be around the average.

6 Conclusion

Motivated by the success of Stable Diffusion series, we propose latent flow matching for efficient graph generation. With the combination of autoencoders and flow matching, we have achieved better performance than using only VAE or DGMs. By incorporating high order samplers, we can generate high quality samples with a pretty faster speed. We hope that we can extend this demo into a more detailed project in the future and provide a proper solution to graph generation tasks.

Reproducibility. Code and pretrained checkpoints available at https://github.com/ziqiao-w/latent_graph_diffusion.

References

Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Avishek Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael Bronstein, and Alexander Tong. Se(3)-stochastic flow matching for protein backbone generation, 2024. URL https://arxiv.org/abs/2310.02391.

Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking, 2023. URL https://arxiv.org/abs/2210.01776.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space, 2023. URL https://arxiv.org/abs/2307.08698.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Jiaqi Guan, Wesley Wei Qian, Xingang Peng, Yufeng Su, Jian Peng, and Jianzhu Ma. 3d equivariant diffusion for target-aware molecule generation and affinity prediction. In *International Conference on Learning Representations*, 2023.

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

Ilia Igashov, Hannes Stärk, Clément Vignac, Arne Schneuing, Victor Garcia Satorras, Pascal Frossard, Max Welling, Michael Bronstein, and Bruno Correia. Equivariant 3d-conditional diffusion model for molecular linker design. *Nature Machine Intelligence*, pp. 1–11, 2024.

Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Bowen Jing, Bonnie Berger, and Tommi Jaakkola. Alphafold meets flow matching for generating protein ensembles, 2024. URL https://arxiv.org/abs/2402.04845.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

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.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds, 2022. URL https://arxiv.org/abs/2202.09778.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022. URL https://arxiv.org/abs/2206.00927.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models, 2023. URL https://arxiv.org/abs/2211.01095.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

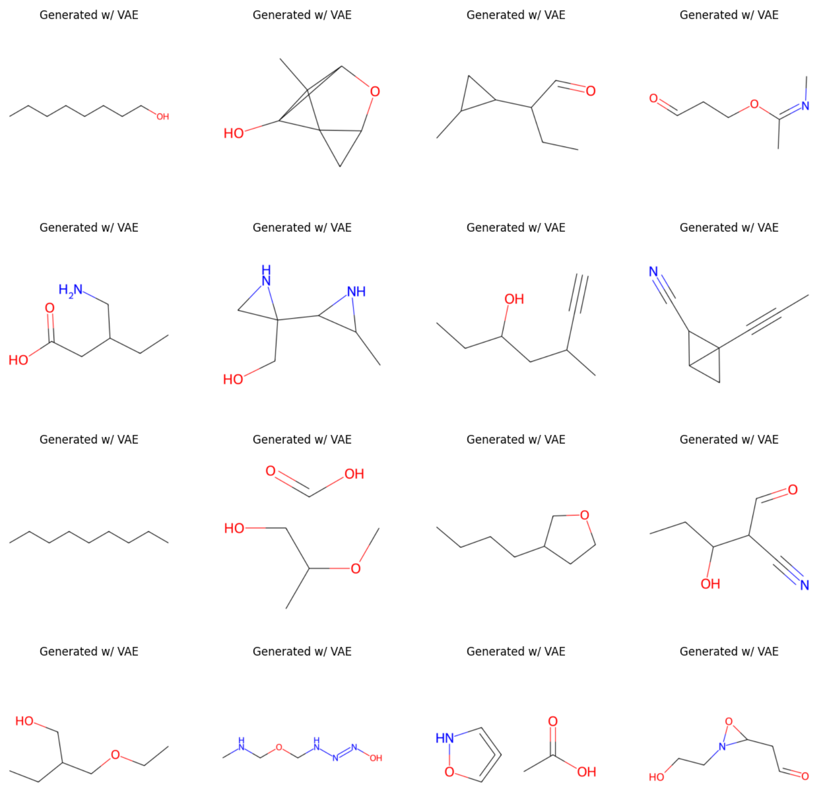
Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW. Expert Certification.

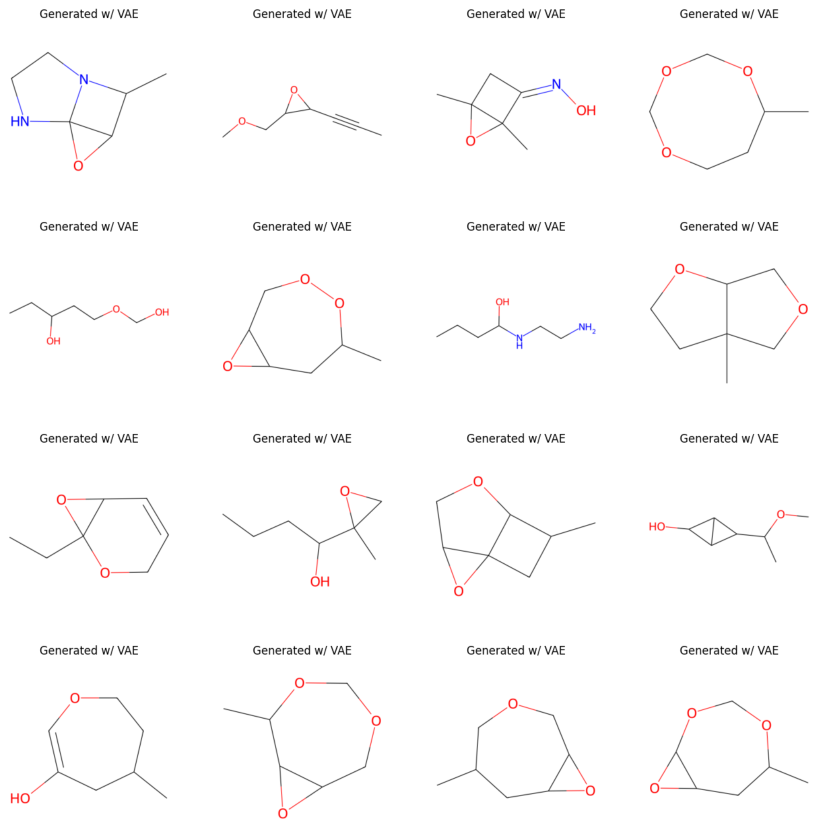
Suttisak Wizadwongsa, Worameth Chinchuthakun, Pramook Khungurn, Amit Raj, and Supasorn Suwajanakorn. Diffusion sampling with momentum for mitigating divergence artifacts, 2023. URL https://arxiv.org/abs/2307.11118.

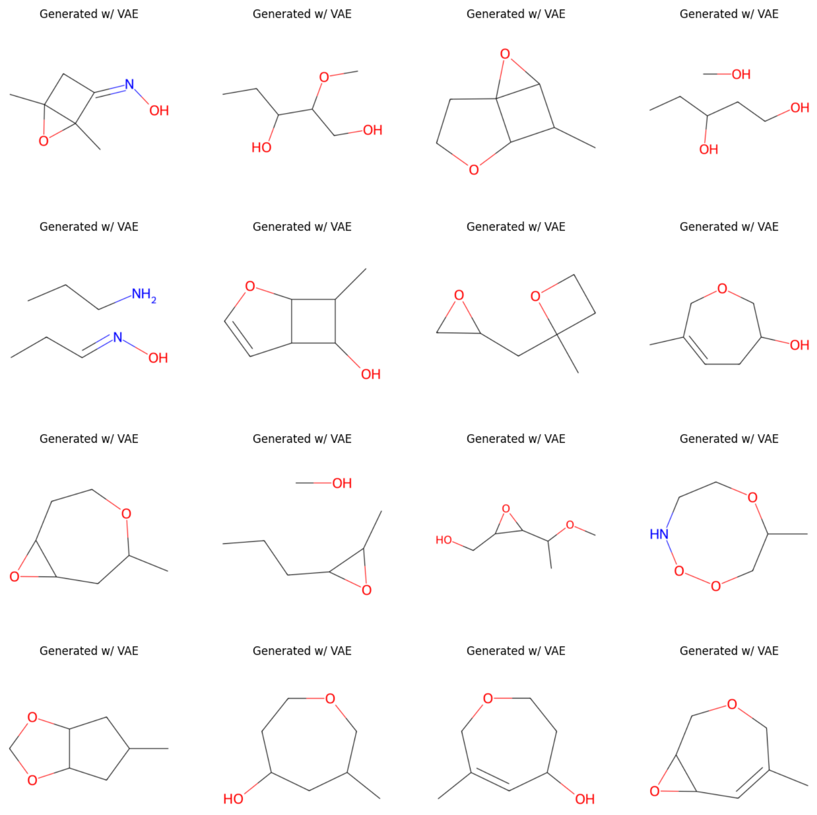
Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

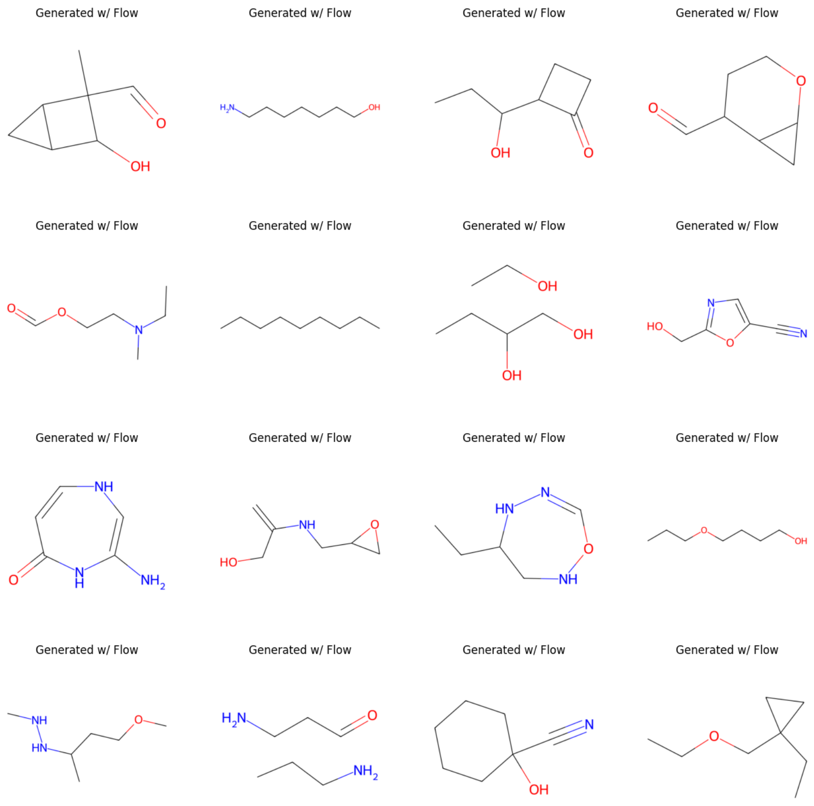
Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics, 2023. URL https://arxiv.org/abs/2310.13268.

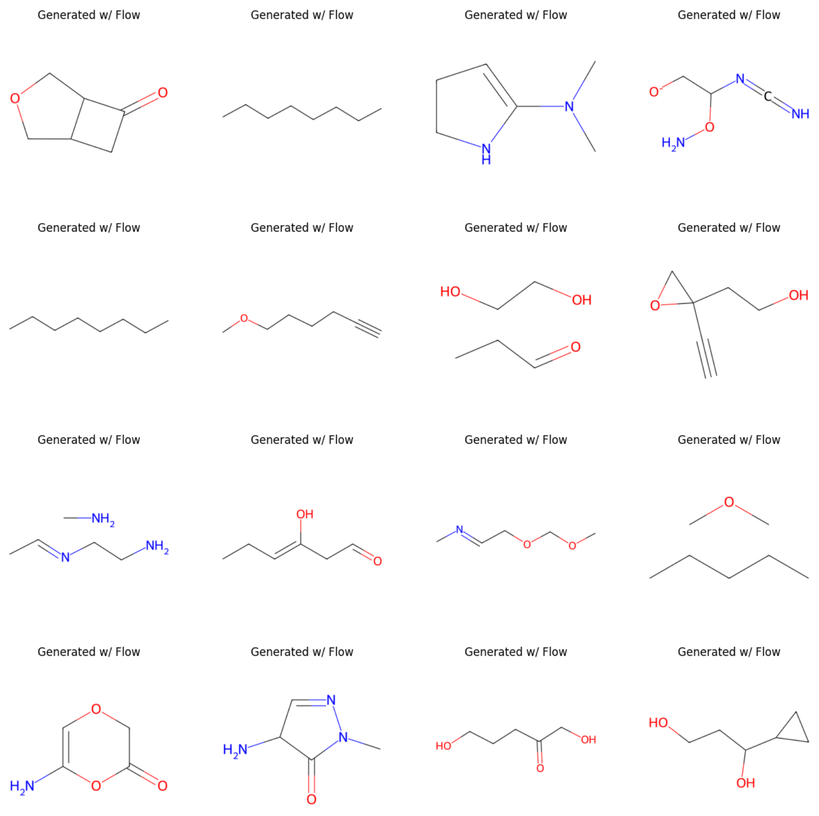
A MORE SAMPLES




Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE


Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE
Generated w/ VAE


Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow
Generated w/ Flow


Generated w/ Flow O O
Generated w/ Flow
Generated w/ Flow N H N
Generated w/ Flow O N C NH H2N O
Generated w/ Flow
Generated w/ Flow O
Generated w/ Flow HO OH O
Generated w/ Flow O OH
Generated w/ Flow NH2 N NH2
Generated w/ Flow OH O
Generated w/ Flow N O O
Generated w/ Flow O
Generated w/ Flow O H2N O O
Generated w/ Flow N H2N N O
Generated w/ Flow HO O OH
Generated w/ Flow HO OH