Exploring Discrete Flow Matching for 3D De Novo Molecule Generation

Ian Dunn
Dept. of Computational & Systems Biology
University of Pittsburgh
Pittsburgh, PA 15260
ian.dunn@pitt.edu

David Ryan Koes
Dept. of Computational & Systems Biology
University of Pittsburgh
Pittsburgh, PA 15260
dkoes@pitt.edu

Abstract

Deep generative models that produce novel molecular structures have the potential to facilitate chemical discovery. Flow matching is a recently proposed generative modeling framework that has achieved impressive performance on a variety of tasks including those on biomolecular structures. The seminal flow matching framework was developed only for continuous data. However, *de novo* molecular design tasks require generating discrete data such as atomic elements or sequences of amino acid residues. Several discrete flow matching methods have been proposed recently to address this gap. In this work we benchmark the performance of existing discrete flow matching methods for 3D *de novo* small molecule generation and provide explanations of their differing behavior. As a result we present FlowMol-CTMC, an open-source model that achieves state of the art performance for 3D *de novo* design with fewer learnable parameters than existing methods. Additionally, we propose the use of metrics that capture molecule quality beyond local chemical valency constraints and towards higher-order structural motifs. These metrics show that even though basic constraints are satisfied, the models tend to produce unusual and potentially problematic functional groups outside of the training data distribution. Code and trained models for reproducing this work are available at https://github.com/dunni3/FlowMol.

1 Introduction

Deep generative models that can directly sample molecular structures with desired properties have the potential to accelerate chemical discovery by reducing or eliminating the need to engage in resource-intensive, screening-based discovery paradigms. Moreover, generative models may improve chemical discovery by enabling multi-objective design of chemical matter. In pursuit of this idea, there has been recent interest in developing generative models for the design of small-molecule therapeutics [1–8], proteins [9–11], and materials [12]. The most popular approach for these tasks has been to apply diffusion models [13–15] to point cloud representations of molecular structures.

Flow matching [16–19] is a generative modeling framework that generalizes diffusion models while being both simpler and providing more flexibility in model design. This flexibility has enabled flow matching to improve over diffusion in some cases, demonstrating impressive results [20–23]. However, the seminal flow matching formulation is only designed for continuous data, constraining the domain of problems that can be modeled under this framework. To address this gap, several recent works have proposed discrete flow matching (DFM) methods. Existing discrete flow matching methods fall into two general categories: applying continuous flow matching to a continuous embedding of discrete data [24–28] , or defining flows on discrete state spaces with Continuous Time Markov Chains (CTMC) [29, 30]. However, there has yet to be a controlled comparison of these distinct approaches for discrete generative modeling.

Machine Learning for Structural Biology Workshop, NeurIPS 2024.

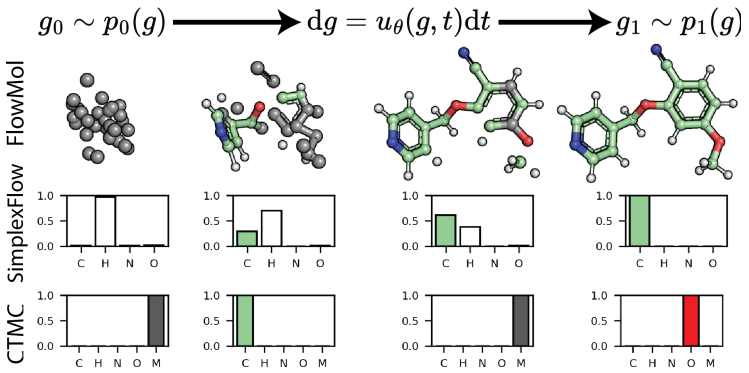

Figure 1: **Overview** *Top:* We adapt the flow matching framework for unconditional 3D molecule generation and explore the use of different discrete flow matching methods. This CTMC trajectory shows masked atoms in gray. *Middle:* Trajectory of the atom type vector for a single atom under SimplexFlow, a variant of continuous flow matching developed for categorical variables. Atom type flows lie on the probability simplex. *Bottom:* Trajectory of an atom type vector for a CTMC flow. Atom types jump between the mask state and real atom types.

Our contributions are as follows:

1. A direct comparison of discrete flow matching methods on *de novo* molecule generation, controlling for model architecture and training procedures.
2. FlowMol-CTMC, a flow matching model that achieves state of the art molecular validity while using fewer learnable parameters than baseline methods.
3. Novel metrics that capture molecule quality beyond local chemical valency constraints and towards higher-order structural motifs.

2 Background

Flow matching [16–19] prescribes a method to interpolate between two distributions q_{source}, q_{target} by modeling a set of time-dependent marginal distributions p_t having the property that $p_0 = q_{\text{source}}$ and $p_1 = q_{\text{target}}$. The source distribution is typically a simple prior and the target distribution a complex data distribution that cannot be sampled easily. Given access to a conditioning variable $z \quad p(z)$, the marginal process p_t can be described as an expectation over conditional probability paths:

$$p_t(x) = \mathsf{E}_{p(z)}\left[p_t(x|z)\right] \tag{1}$$

The conditioning variable is generally taken to be either the final value $z = x_1$ or pairs initial and final points $z = (x_0, x_1)$. Conditional probability paths are chosen such that they can be sampled in a simulation free manner for any t 2 $[0, 1]$

Continuous Flow Matching In the continuous case $p_t(x)$ can be sampled by numerical integration of a learned vector field $u_t(x) = \frac{dx}{dt}$. This vector field can be the direct output of a neural network or it can be parameterized as a function of an optimal denoiser $\hat{x}_1(x_t)$ that minimizes the tractable training objective (2). The latter approach has been found to be more effective for molecular structures [23, 24].

$$L \;/\; \mathsf{E}_{t,p(z),p_t(x_t|z)}\left[k\hat{x}_1(x_t) \quad x_1k\right] \tag{2}$$

Continuous Approaches to Discrete Flow Matching The simplest approach for flow matching on discrete data is to build continuous flows from a Gaussian prior to one-hot vectors [28, 31]; this approach makes no accommodation for the discrete nature of the data. Alternatively, one can define a continuous embedding of discrete data, and then perform continuous flow matching on the

embedding. This is the approach of SimplexFlow [24] and Dirichlet Flows [26] where the continuous representation of choice is the probability simplex. Additional works have proposed Riemannian flow matching on the probability simplex [25, 27]; these methods also fall into the category of continuous flow matching on continuous representations of discrete data, although we do not implement these methods in our work.

CTMC Campbell et al. [29] and Gat et al. [30] develop a flow matching method for sequences of discrete tokens $x = f x^i g_{i=1}^N$ where each sequence element $x^i \in f1, 2, \ldots, Dg$ "jumps" between D possible discrete states throughout the trajectory. Marginal trajectories $p_t(x)$ are obtained by iterative sampling of position-wise transition distributions $p^i(x^i_{t+dt} | x_t)$ that are categorical distributions parameterized by a learned sequence denoiser $p^\theta(x^i_1 | x_t)$ trained to minimize a standard cross-entropy loss:

$$L_{CE} = \mathbb{E}_{t,p(z),p_t(x_t|z)} \left[\quad \log p^\theta(x^i_1 | x_t) \right] \tag{3}$$

3 Methods

3.1 Problem Setting

We represent a 3D molecule with N atoms as a fully-connected graph. Each atom is a node in the graph. Every atom has a position in space $X = f x_i g_{i=1}^N \in \mathbb{R}^{N \times 3}$, an atom type (in this case the atomic element) $A = f a_i g_{i=1}^N$, and a formal charge $C = f c_i g_{i=1}^N$. Additionally, every pair of atoms has a bond order $E = f e_{ij} g i, j \in [N] | i \neq j g$. Atom types, charges, and bond orders are categorical variables. When treated with continuous FM they are represented as one-hot encoded vectors. For brevity, we denote a molecule by the symbol g, which can be thought of as a tuple of the constituent data types $g = (X, A, C, E)$.

There is no closed-form expression or analytical technique for sampling the distribution of realistic molecules $p(g)$. We seek to train a flow matching model to sample this distribution. We define the conditional probability path for a molecule to factorize into conditional probability paths over each modality. That is, conditional probability paths are defined independently for each modality.

$$p_t(g | g_0, g_1) = p_t(X | X_0, X_1) p_t(A | A_0, A_1) p_t(C | C_0, C_1) p_t(E | E_0, E_1) \tag{4}$$

We train one neural network to approximate $p(g_1 | g_t)$ by jointly minimizing reconstruction losses for all data modalities. Our total loss is a weighted combination of FM losses on each modality:

$$L = \eta_X L_X + \eta_A L_A + \eta_C L_C + \eta_E L_E \tag{5}$$

The raw output of the neural network is an estimate of the denoised molecule that will be obtained at the end of the trajectory. We denote the neural network outputs as $\hat{g}_1(g_t)$, and refer to this model as the "denoiser".

We train multiple variants of a *de novo* molecule flow matching model using the same model architecture, datasets, and training procedure. Model variants are distinguished only by the discrete flow matching method used to generate atom types, atomic charges, and bond orders.

DFM Variants We train models with four different DFM methods. The "Continuous" model uses a fully continuous approach; making no accommodations for the discrete data. Categorical variables are generated via flows from a Gaussian prior to one-hot vectors; this is effectively the formulation of categorical flow matching presented in Eijkelboom et al. [28]. We implement continuous methods constrained to a continuous embedding of discrete data; these are Dirichlet Flows [26] and SimplexFlow [24]. Finally we implement CTMC Flows [29, 30], which are defined on discrete state spaces. Detailed descriptions of each flow matching method are presented in Appendix A.

3.2 Model Architecture

We use the neural network architecture proposed by FlowMol [24]. Molecules are treated as fully-connected graphs. The model is designed to accept a sample g_t and predict the final molecule g_1.

FlowMol is comprised of stacks of sequential Molecule Update Blocks that update node features, node positions, and edge features. A single Molecule Update Block is composed of a message-passing graph convolution followed by node-wise and edge-wise updates. Geometric Vector Perceptrons (GVPs) [32] are used to learn and update vector features. The model architecture is explained in detail in Appendix B.

3.3 Datasets and Model Evaluation

We train models on GEOM-Drugs [33] using explicit hydrogens. GEOM-Drugs contains approximately 300k larger, drug-like molecules with multiple conformers for each molecule. We use the same dataset splits as Vignac et al. [34]. We also trained models on QM9 [35, 36] but only present these results in the Appendix E because we consider it to be an overly simple benchmark of model performance.

Validity We report standard metrics on the validity of generated molecular topology: percent molecules stable and percent molecules valid. An atom is defined as "stable" if it has valid valency. Atomic valency is the sum of bond orders for an atom. A molecule is counted as stable if all of its constituent atoms are stable. A molecule is considered "valid" if it can be sanitized by RDKit [37] using default sanitization settings.

Energy Metrics regarding molecular topology provide no indication of quality of the molecular geometries produced by a model. Therefore, we also compute the Jensen-Shannon divergence of the distribution of potential energies for molecules in the training data and molecules sampled from models. Potential energies are obtained from the Merck Molecular Mechanics Force-Field implemented in RDKit [37].

Functional Group Validity Basic chemical validity is a necessary but insufficient condition for designing molecules for a particular application such as therapeutic use. Small-molecule drug discovery scientists have compiled sets of functional groups known to be unstable, toxic, or produce erroneous assay results. Taking inspiration from Walters [38] we measure and report the presence of these problematic functional groups, which we call "structural alerts," as a metric of molecule quality[1]. We specifically use the well-known Dundee [39] and Glaxo Wellcome [40] structural alerts. Additionally, we count all of the unique ring systems observed in a batch of molecules. We then record how many times each unique ring system is observed in ChEMBL [41], a database of 2.4M bio-active compounds. We report the average rate at which ring systems occur that are never observed in ChEMBL. Ring system and structural alert counting are implemented using the useful_rdkit_utils repository [42].

Molecule quality metrics are reported for samples of $10,000$ molecules, repeated 5 times. Inference is run on FlowMol using Euler integration with 100 evenly-spaced time steps for QM9 and 250 time steps for GEOM. All results are reported with 95% confidence intervals. For all samplings, the number of atoms in each molecule is sampled according to the training data distribution.

4 Experiments

DFM ablations The results of ablations in DFM type are shown in Table 1. The best performing method, by a large margin, is CTMC, producing an increase in molecular stability of 26 percentage points over the Continuous approach to discrete FM. Also notable in these results is that structural alerts and OOD rings are over-represented in generated molecules.

Explaining the Performance Gap In the case of continuous flows on discrete data, a single atom's type vector moves towards a vertex of the simplex, e.g., a one-hot vector. The vertex that the atom type moves towards is the predicted atom type from the denoising model $\hat{g}_1(g_t)$ and the atom only gets close to the predicted vertex as $t \; ! \; 1$. As a result, there is a time lag between when the decoder $\hat{g}_1(g_t)$ makes an atom type assignment and when that assignment is reflected in g_t. We can quantify

[1]Determining whether a functional group is problematic in the context of a drug discovery campaign can, in some cases, be subjective. However, measuring the presence of functional groups can still quantify similarity to training data at higher-order levels of organization than chemical valency.

Table 1: Discrete Flow Type Ablations on GEOM-Drugs

Categorical Flow Type	Mols Stable (%) (↑)	Mols Valid (%) (↑)	JS(E) (#)	Structural Alert Rate (per mol) (#)	OOD Ring (per mol) (#)
(Training Data)	100	100	0	0.69	0.04
Dirichlet	20.4 0.3	15.6 0.3	0.45 0.01	3.09 0.03	0.64 0.01
SimplexFlow	64.6 0.3	42.5 0.6	0.33 0.01	2.01 0.04	0.32 0.01
Continuous	69.5 0.4	51.3 0.5	0.32 0.00	2.39 0.03	0.47 0.01
CTMC	**96.2** 0.1	**91.5** 0.3	**0.14** 0.00	**1.23** 0.01	**0.29** 0.00

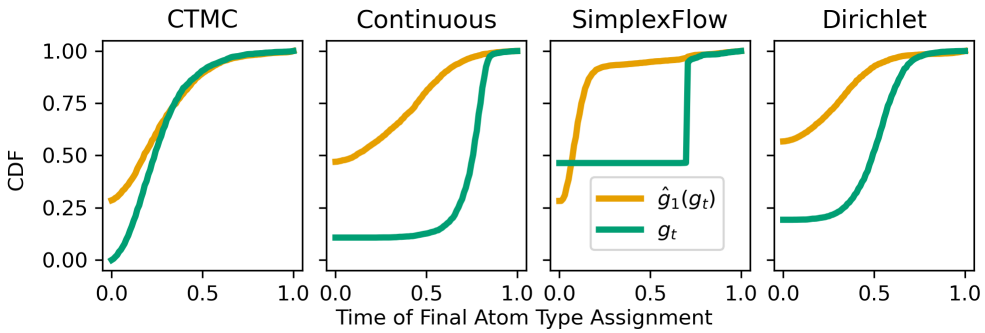

Figure 2: **Atom Type Assignment Times:** Cumulative Density Functions (CDFs) of the time at which an atom is assigned its final atom type, for each DFM method tested. Green lines show the time of final atom type assignments in g_t. Gold lines show the times when the final atom type is assigned in $\hat{g}_1(g_t)$ (the predicted final molecule given the current molecule at time t).

this phenomenon by measuring the time at which an atom acquires its final state in g_t and $\hat{g}_1(g_t)$; we refer to this as the atom type assignment time. The distributions of assignment time in Figure 2 show that continuous varieties of DFM exhibit substantial assignment time lagging. For example, at $t = 0.5$ the Continuous DFM denoiser ($\hat{g}_1(g_t)$) has decided on the final atom type for 78% of atoms while only 15% of atoms in g_t occupy their final atom type. State changes take a significant amount of time to occur when using continuous representations of discrete data. A similar pattern is seen in the assignment time distributions for SimplexFlow and Dirichlet flows as well. In contrast, because CTMC flows truly treat the data as discrete, atom types can often jump to the state predicted by the denoiser without delay; correspondingly, there is a smaller assignment time gap shown in Figure 2. CTMC flows are inherently better at making use of prior predictions and navigating the discrete spaces they model. Note that we use atom type only as an example of this pathology; this phenomenon occurs for any categorical data modeled with continuous flows. Additional details on this analysis are provided in Appendix C.

Comparison to Baselines We compare FlowMol-CTMC to three diffusion baselines that are considered SOTA for this task: MiDi [34], JODO [43], and EQGAT-Diff [44]. Another relevant baseline is Irwin et al. [45]; however, as no code or trained models are available independent evaluation is impossible and reported metrics are not directly comparable.

Performance of FlowMol-CTMC relative to baselines is shown in Table 2. FlowMol-CTMC achieves SOTA performance on chemical valency metrics despite having fewer parameters. Interestingly, FlowMol-CTMC under performs baselines on functional group based metrics. All models express structural alerts at significantly higher rates than in the data.

5 Conclusions

We demonstrate that CTMC flows [29, 30] are the most effective discrete flow matching method evaluated; changing only the DFM formulation has dramatic effects on the quality of generated samples. We also present a compelling mechanistic explanation for the performance gap between CTMCs and continuous approaches to discrete flow matching. Our analysis reinforces the idea that,

Table 2: Comparison of FlowMol to baseline models on GEOM-Drugs

Model	Mols Stable (%) (″)	Mols Valid (%) (″)	JS(E) (#)	Structural Alerts (per mol) (#)	OOD Rings (per mol) (#)	Parameters (10^6) (#)
(Training Data)	100	100	0	0.69	0.04	-
MiDi	85.1 0.9	71.6 0.9	0.23 0.00	1.01 0.01	0.33 0.00	24.1
JODO	90.7 0.5	76.5 0.8	0.17 0.01	**0.84 0.02**	**0.21 0.00**	5.7
EQGAT-Diff	93.4 0.2	86.1 0.3	**0.11 0.00**	1.06 0.01	0.27 0.00	12.3
FlowMol-CTMC	**96.2 0.1**	**91.6 0.1**	0.14 0.00	1.23 0.01	0.28 0.00	**4.3**

generally, relaxing discrete data to continuous spaces introduces undesirable pathologies. As a result of this analysis, we present FlowMol-CTMC, a 3D de novo molecule generative model achieving state of the art molecular validity with fewer learnable parameters than comparable methods.

Finally, we introduce novel metrics for the quality of de novo designed molecules by measuring the presence of problematic functional groups and unusual ring systems. Our comparison to previous state of the art models shows that a model can have improved validity while producing more problematic functional groups; making clear that future work in this field should move beyond validity-based metrics and towards higher-order notions of molecular quality.

6 Acknowledgments

This work is funded through R35GM140753 from the National Institute of General Medical Sciences. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute of General Medical Sciences or the National Institutes of Health.

References

[1] Lei Huang, Tingyang Xu, Yang Yu, Peilin Zhao, Xingjian Chen, Jing Han, Zhi Xie, Hailong Li, Wenge Zhong, Ka-Chun Wong, and Hengtong Zhang. A dual diffusion model enables 3D molecule generation and lead optimization based on target pockets. *Nature Communications*, 15 (1):2657, March 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-46569-1. URL https://www.nature.com/articles/s41467-024-46569-1. Publisher: Nature Publishing Group.

[2] 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, March 2023. URL http://arxiv.org/abs/2303.03543. arXiv:2303.03543 [cs, q-bio].

[3] Arne Schneuing, Yuanqi Du, Charles Harris, Arian Jamasb, Ilia Igashov, Weitao Du, Tom Blundell, Pietro Lió, Carla Gomes, Max Welling, Michael Bronstein, and Bruno Correia. Structure-based Drug Design with Equivariant Diffusion Models, June 2023. URL http://arxiv.org/abs/2210.13695. arXiv:2210.13695 [cs, q-bio].

[4] Xingang Peng, Shitong Luo, Jiaqi Guan, Qi Xie, Jian Peng, and Jianzhu Ma. Pocket2Mol: Efficient Molecular Sampling Based on 3D Protein Pockets, May 2022. URL http://arxiv.org/abs/2205.07249. arXiv:2205.07249 [cs, q-bio].

[5] Meng Liu, Youzhi Luo, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. Generating 3D Molecules for Target Protein Binding, May 2022. URL http://arxiv.org/abs/2204.09410. arXiv:2204.09410 [cs, q-bio].

[6] Jos Torge, Charles Harris, Simon V. Mathis, and Pietro Lio. DiffHopp: A Graph Diffusion Model for Novel Drug Design via Scaffold Hopping, August 2023. URL http://arxiv.org/abs/2308.07416. arXiv:2308.07416 [q-bio].

[7] 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*, pages 1–11, April 2024. ISSN 2522-5839. doi: 10.1038/s42256-024-00815-9. URL https://www.nature.com/articles/s42256-024-00815-9. Publisher: Nature Publishing Group.

[8] Ian Dunn and David Koes. Accelerating Inference in Molecular Diffusion Models with Latent Representations of Protein Structure. October 2023. URL https://openreview.net/forum?id=Z4ia7s2tpV.

[9] Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, August 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06415-8. URL https://www.nature.com/articles/s41586-023-06415-8. Publisher: Nature Publishing Group.

[10] Nathaniel R. Bennett, Joseph L. Watson, Robert J. Ragotte, Andrew J. Borst, Déjenaé L. See, Connor Weidle, Riti Biswas, Ellen L. Shrock, Philip J. Y. Leung, Buwei Huang, Inna Goreshnik, Russell Ault, Kenneth D. Carr, Benedikt Singer, Cameron Criswell, Dionne Vafeados, Mariana Garcia Sanchez, Ho Min Kim, Susana Vázquez Torres, Sidney Chan, and David Baker. Atomically accurate de novo design of single-domain antibodies, March 2024. URL https://www.biorxiv.org/content/10.1101/2024.03.14.585103v1. Pages: 2024.03.14.585103 Section: New Results.

[11] John B. Ingraham, Max Baranov, Zak Costello, Karl W. Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M. Lord, Christopher Ng-Thow-Hing, Erik R. Van Vlack, Shan Tie, Vincent Xue, Sarah C. Cowles, Alan Leung, João V. Rodrigues, Claudio L. Morales-Perez, Alex M. Ayoub, Robin Green, Katherine Puentes, Frank Oplinger, Nishant V. Panwar, Fritz Obermeyer, Adam R. Root, Andrew L. Beam, Frank J. Poelwijk, and Gevorg Grigoryan. Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078, November 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06728-8. URL https://www.nature.com/articles/s41586-023-06728-8. Publisher: Nature Publishing Group.

[12] Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Sasha Shysheya, Jonathan Crabbé, Lixin Sun, Jake Smith, Bichlien Nguyen, Hannes Schulz, Sarah Lewis, Chin-Wei Huang, Ziheng Lu, Yichi Zhou, Han Yang, Hongxia Hao, Jielan Li, Ryota Tomioka, and Tian Xie. MatterGen: a generative model for inorganic materials design, January 2024. URL http://arxiv.org/abs/2312.03687. arXiv:2312.03687 [cond-mat].

[13] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, November 2015. URL http://arxiv.org/abs/1503.03585. arXiv:1503.03585 [cond-mat, q-bio, stat].

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, December 2020. URL http://arxiv.org/abs/2006.11239. arXiv:2006.11239 [cs, stat].

[15] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations, February 2021. URL http://arxiv.org/abs/2011.13456. arXiv:2011.13456 [cs, stat].

[16] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling, February 2023. URL http://arxiv.org/abs/2210.02747. arXiv:2210.02747 [cs, stat].

[17] Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport, July 2023. URL http://arxiv.org/abs/2302.00482. arXiv:2302.00482 [cs].

[18] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic Interpolants: A Unifying Framework for Flows and Diffusions, November 2023. URL http://arxiv.org/abs/2303.08797. arXiv:2303.08797 [cond-mat].

[19] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow, September 2022. URL http://arxiv.org/abs/2209.03003. arXiv:2209.03003 [cs].

[20] Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring Flow and Diffusion-based Generative Models with Scalable Interpolant Transformers, January 2024. URL http://arxiv.org/abs/2401.08740. arXiv:2401.08740 [cs].

[21] 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, April 2024. URL http://arxiv.org/abs/2310.02391. arXiv:2310.02391 [cs].

[22] Jason Yim, Andrew Campbell, Andrew Y. K. Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Regina Barzilay, Tommi Jaakkola, and Frank Noé. Fast protein backbone generation with SE(3) flow matching, October 2023. URL http://arxiv.org/abs/2310.05297. arXiv:2310.05297 [q-bio].

[23] Guillaume Huguet, James Vuckovic, Kilian Fatras, Eric Thibodeau-Laufer, Pablo Lemos, Riashat Islam, Cheng-Hao Liu, Jarrid Rector-Brooks, Tara Akhound-Sadegh, Michael Bronstein, Alexander Tong, and Avishek Joey Bose. Sequence-Augmented SE(3)-Flow Matching For Conditional Protein Backbone Generation, May 2024. URL http://arxiv.org/abs/2405.20313. arXiv:2405.20313 [cs, q-bio].

[24] Ian Dunn and David Ryan Koes. Mixed Continuous and Categorical Flow Matching for 3D De Novo Molecule Generation, April 2024. URL http://arxiv.org/abs/2404.19739. arXiv:2404.19739 [cs, q-bio].

[25] Oscar Davis, Samuel Kessler, Mircea Petrache, İsmail İlkan Ceylan, Michael Bronstein, and Avishek Joey Bose. Fisher Flow Matching for Generative Modeling over Discrete Data, May 2024. URL http://arxiv.org/abs/2405.14664. arXiv:2405.14664 [cs].

[26] Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet Flow Matching with Applications to DNA Sequence Design, February 2024. URL http://arxiv.org/abs/2402.05841. arXiv:2402.05841 [cs, q-bio].

[27] Chaoran Cheng, Jiahan Li, Jian Peng, and Ge Liu. Categorical Flow Matching on Statistical Manifolds, May 2024. URL http://arxiv.org/abs/2405.16441. arXiv:2405.16441 [cs, stat].

[28] Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent. Variational Flow Matching for Graph Generation, June 2024. URL http://arxiv.org/abs/2406.04843. arXiv:2406.04843 [cs, stat].

[29] Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design, June 2024. URL http://arxiv.org/abs/2402.04997. arXiv:2402.04997 [cs, q-bio, stat].

[30] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete Flow Matching, July 2024. URL http://arxiv.org/abs/2407.15595. arXiv:2407.15595 [cs].

[31] Yuxuan Song, Jingjing Gong, Minkai Xu, Ziyao Cao, Yanyan Lan, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. Equivariant Flow Matching with Hybrid Probability Transport, December 2023. URL http://arxiv.org/abs/2312.07168. arXiv:2312.07168 [cs].

[32] Bowen Jing, Stephan Eismann, Pratham N. Soni, and Ron O. Dror. Equivariant Graph Neural Networks for 3D Macromolecular Structure, July 2021. URL http://arxiv.org/abs/2106.03843. arXiv:2106.03843 [cs, q-bio].

[33] Simon Axelrod and Rafael Gómez-Bombarelli. GEOM, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):185, April 2022. ISSN 2052-4463. doi: 10.1038/s41597-022-01288-4. URL https://www.nature.com/articles/s41597-022-01288-4. Publisher: Nature Publishing Group.

[34] Clement Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. MiDi: Mixed Graph and 3D Denoising Diffusion for Molecule Generation, June 2023. URL http://arxiv.org/abs/2302.09048. arXiv:2302.09048 [cs].

[35] Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, November 2012. ISSN 1549-9596. doi: 10.1021/ci300415d. URL https://doi.org/10.1021/ci300415d. Publisher: American Chemical Society.

[36] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1): 140022, August 2014. ISSN 2052-4463. doi: 10.1038/sdata.2014.22. URL https://www.nature.com/articles/sdata201422. Publisher: Nature Publishing Group.

[37] RDKit. URL http://www.rdkit.org/.

[38] Patrick W. Walters. Generative Molecular Design Isn't As Easy As People Make It Look, May 2024. URL https://practicalcheminformatics.blogspot.com/2024/05/generative-molecular-design-isnt-as.html.

[39] Ruth Brenk, Alessandro Schipani, Daniel James, Agata Krasowski, Ian Hugh Gilbert, Julie Frearson, and Paul Graham Wyatt. Lessons Learnt from Assembling Screening Libraries for Drug Discovery for Neglected Diseases. *ChemMedChem*, 3(3):435–444, 2008. ISSN 1860-7187. doi: 10.1002/cmdc.200700139. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cmdc.200700139. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cmdc.200700139.

[40] Mike Hann, Brian Hudson, Xiao Lewell, Rob Lifely, Luke Miller, and Nigel Ramsden. Strategic Pooling of Compounds for High-Throughput Screening. *Journal of Chemical Information and Computer Sciences*, 39(5):897–902, September 1999. ISSN 0095-2338. doi: 10.1021/ci990423o. URL https://doi.org/10.1021/ci990423o. Publisher: American Chemical Society.

[41] Barbara Zdrazil, Eloy Felix, Fiona Hunter, Emma J Manners, James Blackshaw, Sybilla Corbett, Marleen de Veij, Harris Ioannidis, David Mendez Lopez, Juan F Mosquera, Maria Paula Magarinos, Nicolas Bosc, Ricardo Arcila, Tevfik Kizilören, Anna Gaulton, A Patrícia Bento, Melissa F Adasme, Peter Monecke, Gregory A Landrum, and Andrew R Leach. The ChEMBL Database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods. *Nucleic Acids Research*, 52(D1):D1180–D1192, January 2024. ISSN 0305-1048. doi: 10.1093/nar/gkad1004. URL https://doi.org/10.1093/nar/gkad1004.

[42] Patrick Walters. PatWalters/useful_rdkit_utils, September 2024. URL https://github.com/PatWalters/useful_rdkit_utils. original-date: 2021-12-31T00:24:33Z.

[43] Han Huang, Leilei Sun, Bowen Du, and Weifeng Lv. Learning Joint 2D & 3D Diffusion Models for Complete Molecule Generation, June 2023. URL http://arxiv.org/abs/2305.12347. arXiv:2305.12347 [cs, q-bio].

[44] Tuan Le, Julian Cremer, Frank Noé, Djork-Arné Clevert, and Kristof Schütt. Navigating the Design Space of Equivariant Diffusion-Based Generative Models for De Novo 3D Molecule Generation, November 2023. URL http://arxiv.org/abs/2309.17296. arXiv:2309.17296 [cs].

[45] Ross Irwin, Alessandro Tibo, Jon Paul Janet, and Simon Olsson. Efficient 3D Molecular Generation with Flow Matching and Scale Optimal Transport, June 2024. URL http://arxiv.org/abs/2406.07266. arXiv:2406.07266 [cs].

[46] Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching, November 2023. URL http://arxiv.org/abs/2306.15030. arXiv:2306.15030 [physics, stat].

[47] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks, August 2020. URL http://arxiv.org/abs/1909.01315. arXiv:1909.01315 [cs, stat].

A Model Formulation

A.1 Atomic Coordinate Flows

The conditioning variable for atomic coordinate flows is paired initial and final atomic positions $z = (X_0, X_1)$. The distribution of the conditioning variable, often referred to as the coupling distribution [17, 30] is the equivariant optimal transport coupling $(X_0, X_1) \quad \pi\ (X_0, X_1)$ [31, 46]. This means that X_0 and X_1 are independently sampled from the prior and data distribution, respectively, before being aligned via rigid body rotation, translation, and a distance minimizing permutation of node assignments. The conditional probability path for positions is a Dirac density placed on a straight line connecting the terminal states.

$$p_t(X|X_0, X_1) = \delta(X \quad (1 \quad \kappa_t)X_0 \quad \kappa_t X_1) \tag{6}$$

This is equivalent to a deterministic interpolant [18]:

$$X_t = (1 \quad \kappa_t)X_0 + \kappa_t X_1 \tag{7}$$

Where $\kappa_t : [0, 1] \ ! \ [0, 1]$ is the "interpolant schedule": a function taking t as input and returning a number between 0 and 1. The interpolant schedule controls the rate at which atomic coordinates transition from the prior to the target distribution. We design the conditional paths of each modality to have an interpolant schedule, and these interpolant schedules need not be the same for each modality. Interpolant schedule choices are discussed in Appendix A.5.

The prior distribution for atomic coordinates are i.i.d. samples from a standard Gaussian $p_0(X) = \prod_{i=1}^{N} \mathcal{N}(x_0^i|0, \mathrm{I})$.

The conditional probability path (7) is produced by the conditional vector field (8) [17, 24, 30].

$$u_t(X|X_0, X_1) = \frac{\dot{\kappa}_t}{1 \quad \kappa_t}(X_1 \quad X_t) \tag{8}$$

Because of this, it can be shown that the marginal distribution $p_t(X)$ can be sampled by numerical integration of the vector field (9) [24]

$$u(X_t) = \frac{\dot{\kappa}_t}{1 \quad \kappa_t}\left(\hat{X}_1(X_t) \quad X_t\right) \tag{9}$$

Where $\dot{\kappa}_t$ is the time derivative of the interpolant schedule. $\hat{X}_1(X_t)$ is a neural network trained under the denoising loss (10).

$$\mathcal{L} = \mathbb{E}_{t, p_t(X_t|X_0, X_1), \pi^*(X_0, X_1)}\left[\left\|\hat{X}_1(X_t) \quad X_t\right\|\right] \tag{10}$$

In practice $\hat{X}_1$ predictions are obtained as an output head of a neural network that takes in a partially formed molecule g_t and simultaneously predicts $(\hat{X}_1, \hat{A}_1, \hat{C}_1, \hat{E}_1) = \hat{g}_1(g_t)$.

A.2 Continuous Flows on Discrete Data

For the "Continuous" model variant shown in Table 1, categorical variables are treated as fully continuous. We describe here the formulation using atom types as an example but the equations apply equivalently to atom charges and bond orders.

The conditioning variable for fully continuous flows is pairs of initial and terminal states $z = (A_0, A_1)$. The coupling distribution (A_0, A_1) $\pi(A_0, A_1)$ is the independent coupling $\pi(A_0, A_1) = p_0(A_0)p_1(A_1)$; samples are drawn independently from the prior and data distribution.

Each atom type is considered a vector of real numbers a_t^i 2 R^{n_a} where n_a is the number of atom types supported. The prior distribution for atom types are IID samples from a standard Gaussian: $p(A_0) = \prod_{i=1}^{N} p(a_0^i) = \prod_{i=1}^{N} \mathsf{N}(a_0^i j 0, \mathsf{I})$. The data distribution are one-hot vectors indicating the type of each atom.

The conditional probability path is the same used for atomic coordinate flows (8). The learned vector field for sampling $p_t(A_t)$ is of the same form as well (9). However, we use a softmax activation layer guaranteeing predicted endpoints are on the simplex. Correspondingly the denoising loss is a cross-entropy loss rather than an MSE loss.

A.3 Continuous Flows on a Continuous Embedding of Discrete Data

To design flow matching for categorical data, the strategy of SimplexFlow [24] and Dirichlet Flows [26] is to define a continuous representation of categorical variables, and then construct a flow matching model where flows are constrained to this representation. The continuous representation chosen for a d-categorical variable is the d-dimensional probability simplex S^d:

$$\mathsf{S}^d = \left\{ x \; 2 \; \mathsf{R}^d j x_i \quad 0, \mathbb{1} \quad x = 1 \right\} \tag{11}$$

A d-categorical variable x_1 2 $\mathsf{f}1, 2, \ldots, d\mathsf{g}$ can be converted to a point on S^d via one-hot encoding. Correspondingly, the categorical distribution $p_1(x) = \mathsf{C}(q)$ can be converted to a distribution on S^d:

$$p_1(x) = \sum_{j=1}^{d} q_i \delta(x \quad e_j) \tag{12}$$

where e_j is the j^{th} vertex of the simplex and q_j is the probability of x belonging to the j^{th} category.

A.3.1 SimplexFlow

SimplexFlow [24] uses the same conditional probability path (6) and learned vector field (9) as for atomic coordinate flows. The training objective for categorical feature denoising is the cross entropy loss as described in Section A.3.

The prior distribution used is the "marginal-simplex" prior. That is, atom types are first sampled as one-hot vectors according to their marginal distribution in the training data. We then apply a blurring procedure by Gaussian noise addition and projection back on to the simplex. The marginal-simplex prior is visualized in Figure 3

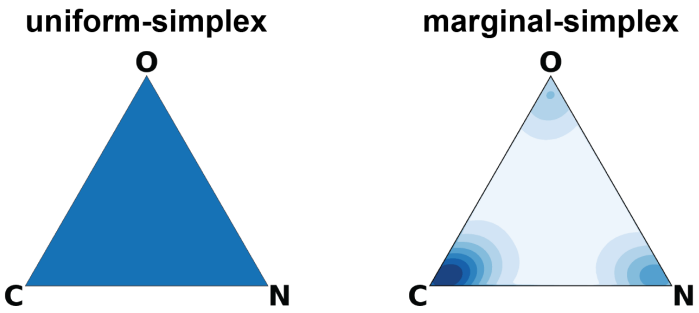

Figure 3: Prior distributions used with support on the probability simplex

A.3.2 Dirichlet Flows

For Dirichlet Flows [26] the conditional probability path for a single atom a_t^i is

$$p_t(a^i | a_1^i = e^j) = \text{Dir}(a^i | \gamma = 1 + e^j \omega) \tag{13}$$

Where Dir is a Dirichlet distribution parameterized by γ and ω represents time. The Dirichlet conditional flow must start at $\omega = 1$ and only converges to $\delta(a^i \quad e^j)$ in the limit $\omega \; ! \; 1$. In order to incorporate Dirichlet flows into our model, we define the relation $\omega_t = \omega_{\max} \kappa_t + 1$, where κ_t is the interpolant schedule. We set $\omega_{\max} = 10$. Dirichlet flow matching necessitates the use of a uniform prior over the simplex which is visualized in Figure 3.

The training objective is a cross-entropy loss on category logits. We refer the reader to [26] for a description of the generating conditional vector field. The marginal vector field is computed as an expectation over conditional vector fields with respect to $p^\theta(a_1^i | a_t^i)$.

A.4 CTMC Flows

When we model discrete data with CTMC flows [29, 30], we introduce an additional category, the "mask" token. Our molecule atom types are considered as a sequence of discrete tokens $A = (a^1, a^2, \ldots, a^N)$ where N is the number of atoms in the molecule and $a^i \; 2 \; f1, 2, \ldots, n_a, Mg$ where n_a is the number of supported atom types and M is the mask token. The prior distribution is Kronecker delta placed on a sequence of mask states, i.e., $p_0(A) = \prod_{i=1}^N \delta_M(a^i)$. Conditional probability paths factorize over each atom in the molecule:

$$p_t(A | A_0, A_1) = \prod_{i=1}^N p_t^i(a^i | A_0, A_1) \tag{14}$$

Where the per-atom conditional probability path takes the form:

$$p_t^i(a^i | A_0, A_1) = \kappa_t \delta_{A_1}(a^i) + (1 \quad \kappa_t) \delta_M(a^i) \tag{15}$$

Where M is the mask token. In other words, at time t, atom a^i has probability κ_t of being in the masked state and probability $1 \quad \kappa_t$ of being in its final state a_1^i. Note that we also use an abbreviated notation here where $\delta_{A_1}(a^i) = \delta_{a_1^i}(a^i)$.

Atom type trajectories are sampled by iterative sampling of transition distributions that factorize over atoms:

$$p(A_{t+ \quad t} | A_t) = \prod_{i=1}^N p^i(a_{t+ \quad t}^i | A_t) \tag{16}$$

The per-atom transition distributions are categorical distributions with logits:

$$p^i(a_{t+ \quad t}^i = j | A_t) = \delta_{a_t^i}(j) + u^i(j, A_t) \Delta t \tag{17}$$

Where $u^i(j, A_t) \; 2 \; \mathsf{R}$ is the instantaneous flow of probability towards atom type j for atom i. Campbell et al. [29] refers to this quantity as a rate matrix and Gat et al. [30] calls it the probability velocity.

The primary difference between the CTMC formulations in Campbell et al. [29] and Gat et al. [30] are the form of the probability velocity used to generate trajectories. For our application, we find the probability velocity proposed by Campbell et al. [29] to give better results; this is contradictory to the conclusions of Gat et al. [30].

Using the proposed conditional rate matrix of Campbell et al. [29], our conditional probability path (15) induces the following marginal probability velocity for $j \; 6= \; a_t^i$:

$$u^i(j, A_t) = \frac{\dot{\kappa}_t + \eta \kappa_t}{1 \quad \kappa_t} p_{1|t}^\theta(a_1^i = j^i | A_t) \delta_M(a_t^i) + \eta(1 \quad \delta_M(a_t^i)) \delta_M(j) \tag{18}$$

Where η is the stochasticity parameter that, when increased, increases both the rate at which tokens are unmasked and remasked. To ensure that (17) defines a valid categorical distribution we set $u^i(a_t^i, A_t) = \sum_{j \neq a_i^t} u^i(j, A_t)$.

Plugging $u^i(a_t^i, A_t)$ into the transition distribution (17) yields sampling dynamics that can be described very simply. If $a_i^t = M$, the probability of unmasking is $\Delta t \frac{\kappa_t + \eta \kappa_t}{1 - \kappa_t}$. If we do unmask, the unmasked state is selected according to our learned model $p_{1jt}^\theta(a_1^i j A_t)$. If we are currently in an unmasked state, the probability of switching to a masked state is $\eta \Delta t$.

We find low-temperature sampling, where the prediction logits are re-normalized with temperature τ:

$$p_{1jt}^\theta(a_1^i j A_t) = \text{softmax}\left(\tau^{-1} \log p_{1jt}^\theta(a_1^i j A_t)\right) \tag{19}$$

to be critical for model performance. In practice we use $\tau = 0.05$ and $\eta = 30$.

A.5 Interpolant Schedules

When using CTMC flows for categorical data, we find a linear interpolant $\kappa_t = t$ for all modalities gives strong performance. For all other flows, we find that a cosine interpolant works better:

$$\kappa_t = 1 - \cos^2\left(\frac{\pi}{2} t^\nu\right) \tag{20}$$

where separate values of ν are used for each modality. We use the same values as described in [24]. Using separate interpolants for each modality encourages the model to determine positions before atom types, before bond orders, and so on. The reason this specifically benefits continuous FM approaches is likely due to the pathologies outlined in Section 4.

B Architecture

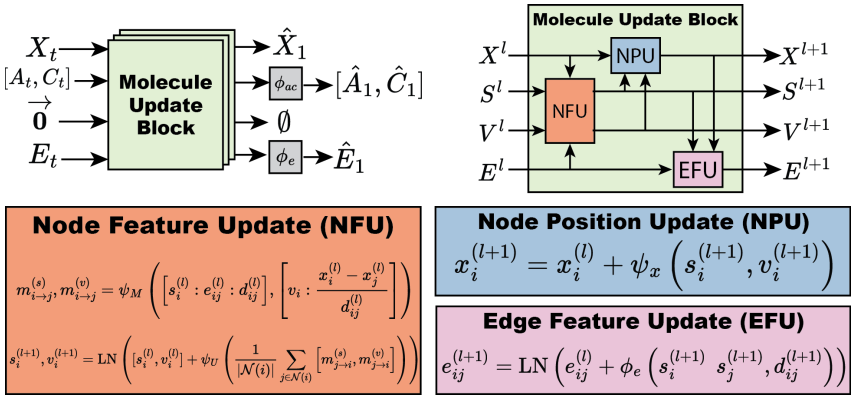

Figure 4: **FlowMol Architecture** *Top left*: An input molecular graph g_t is transformed into a predicted final molecular graph g_1 by being passed through multiple molecule update blocks. *Top right:* A molecule update block uses NFU, NPU, and EFU sub-components to update all molecular features. *Bottom:* Update equations for graph features. ϕ and ψ denote MLPs and GVPs, respectively.

Overview Molecules are treated as fully-connected graphs. The model is designed to accept a sample g_t and predict the final destination molecule g_1. Within the neural network, molecular features are grouped into node positions, node scalar features, node vector features, and edge features. Node positions are identical to the atom positions discussed in Section 3.1. Node scalar features are a concatenation of atom type and atom charge. Node vector features are geometric vectors (vectors with rotation order 1) that are relative to the node position. Node vector features are initialized to zero vectors. Molecular features are iteratively updated by passing g_t through several Molecule Update

Blocks. A Molecule Update Block uses Geometric Vector Perceptrons (GVPs) [32] to handle vector features. Molecule Update Blocks are composed of three components: a node feature update (NFU), node position update (NPU) and edge feature update (EFU). The NFU uses a message-passing graph convolution to update node features. The NPU and EFU blocks are node and edge-wise operations, respectively. Following several molecule update blocks, predictions of the final categorical features $(\hat{A}_1, \hat{C}_1, \hat{E}_1)$ are generated by passing node and edge features through shallow node-wise and edge-wise multi layer perceptrons (MLPs). For models using endpoint parameterization, these MLPs include softmax activations. The model architecture is visualized in Figure 4.

FlowMol is implemented using PyTorch and the Deep Graph Library (DGL) [47].

Each node is endowed with a position in space $x_i \in \mathbb{R}^3$, scalar features $s_i \in \mathbb{R}^d$, and vector features $v_i \in \mathbb{R}^{c \times 3}$. Scalar features are initialized at the network input by concatenating atom type and charge vectors: $s_i^{(0)} = [a_i : c_i]$. Vector features are initialized to zeros $v_i^{(0)} = \mathbf{0}$. Each edge is endowed with scalar edge features that, at the input to the network, are the bond order at time t. We enforce that the bond order on both edges for a pair of atoms is identical: $e_{ij}^{(0)} = e_{ji}^{(0)}$.

In practice, graphs are directed. For every pair of atoms i, j there exists edges in both directions: $i \rightarrow j$ and $j \rightarrow i$. When predicting the final bond orders $\hat{E}_1$ for an edge, we ensure that one prediction is made per pair of atoms and that this prediction is invariant to permutations of the atom indexing. This is accomplished by making our prediction from the sum of the learned bond features. That is, $\hat{e}_1^{ij} = MLP(e_{ij} + e_{ji})$.

Molecule Update Block We define a Molecule Update Block that will update all graph features x_i, s_i, v_i, e_{ij}. Each molecule update block is comprised of 3-sub blocks: a node feature update block, a node position update block, and an edge feature update block. The input molecule graph is passed through L Molecule Update Blocks. Vector features are operated on by geometric vector perceptions (GVPs). A detailed description of our implementation of GVP is provided in Section B.1.

Node Feature Update Block The node feature update block will perform a graph convolution to update node scalar and vector features s_i, v_i. The message generating and node-update functions for this graph convolution are each chains of GVPs. GVPs accept and return a tuple of scalar and vector features. Therefore, scalar and vector messages $m_{i \rightarrow j}^{(s)}$ and $m_{i \rightarrow j}^{(v)}$ are generated by a single function ψ_M which is two GVPs chained together.

$$m_{i \rightarrow j}^{(s)}, m_{i \rightarrow j}^{(v)} = \psi_M \left(\left[s_i^{(l)} : e_{ij}^{(l)} : d_{ij}^{(l)} \right], \left[v_i : \frac{x_i^{(l)} \quad x_j^{(l)}}{d_{ij}^{(l)}} \right] \right) \tag{21}$$

Where : denotes concatenation, and d_{ij} is the distance between nodes i and j at molecule update block l. In practice, we replace all instances of d_{ij} with a radial basis embedding of that distance before passing through GVPs or MLPs. Message aggregation and node features updates are performed as described in [32], with the exception that we do not use a dropout layer:

$$s_i^{(l+1)}, v_i^{(l+1)} = \text{LayerNorm} \left([s_i^{(l)}, v_i^{(l)}] + \psi_U \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \left[m_{j \rightarrow i}^{(s)}, m_{j \rightarrow i}^{(v)} \right] \right) \right) \tag{22}$$

The node update function ψ_U is a chain of three GVPs.

Node Position Update Block The purpose of this block is to update node positions x_i. Node positions are updated as follows:

$$x_i^{(l+1)} = x_i^{(l)} + \psi_x \left(s_i^{(l+1)}, v_i^{(l+1)} \right) \tag{23}$$

Where P is a chain of 3 GVPs in which the final GVP emits 1 vector and 0 scalar features. Moreover for the final GVP, the vector-gating activation function (σ_g in Algorithm 1), which is typically a sigmoid function, is replaced with the identity.

Algorithm 1 Geometric Vector Perceptron with Cross Product

Input: Scalar and vector features: (s, v) 2 R^f R^ν 3
Output: Scalar and vector features: (s^0, v^0) 2 R^j R^μ 3
Hyperparameter: Number of hidden vector features n_h 2 Z^+
Hyperparameter: Number of cross product features n_{cp} 2 Z^+
v_h $W_h v$ 2 R^{n_h} 3
v_{cp} $W_{cp} v$ 2 $\mathsf{R}^{2n_{cp}}$ 3
v_{cp} $v_{cp}[: n_{cp}]$ $v_{cp}[n_{cp} :]$ 2 $\mathsf{R}^{n_{cp}}$ 3 // cross product
v_{h+cp} $\mathrm{Concat}(v_h, v_{cp})$ 2 $\mathsf{R}^{(n_h+n_{cp})}$ 3 // concatenation along rows
v_μ $W_\mu v_{h+cp}$ 2 R^μ 3
s_{h+cp} $\mathsf{k}v_{h+cp}\mathsf{k}$ 2 $\mathsf{R}^{n_h+n_{cp}}$
s_{f+h+cp} $\mathrm{Concat}(s, s_{h+cp})$,
s_j $W_j s_{f+h+cp} + b_j$ 2 R^j
s^0 $\sigma(s_j)$ 2 R^j
v^0 $\sigma_g\left(W_g[\sigma^+(s_m)] + b_g\right)$ v_μ (row-wise) 2 R^μ 3
return (s^0, v^0)

Edge Feature Update Block Edge features are updated by the following equation:

$$e_{ij}^{(l+1)} = \mathrm{LayerNorm}\left(e_{ij}^{(l)} + \phi_e\left(s_i^{(l+1)}, s_j^{(l+1)}, d_{ij}^{(l+1)}\right)\right) \tag{24}$$

Where ϕ_e is a shallow MLP that accepts as input the node scalar features of nodes participating in the edge as well as the distance between the nodes from the positions compute in the NPU block.

B.1 GVP with Cross Product

A geometric vector perception (GVP) can be thought of as a single-layer neural network that applies linear and point-wise non-linear transformation to its inputs. The difference between a GVP and a conventional feed-forward neural network is that GVPs operate on two distinct data types: scalars and vectors. GVPs also allow these data types to exchange information while preserving equivariance of the output vectors. The original GVP only applied linear transformations to the vector features and as a result produces output vectors that are E(3)-equivariant.

We introduce a modification to the GVP as its presented in Jing et al. [32]; specifically we perform a cross product operation on the input vectors. The motivation for this is that the cross product is *not* equivariant to reflections. As a result, the version of GVP we present here is SE(3) equivariant. The benefit of being SE(3) equivariant rather than E(3) equivariant is that the model becomes sensitive to chiral centers in molecules, since reflecting the molecule will produce different model outputs. The operations for our cross product enhanced GVP are described in Algorithm 1.

C Atom Type Assignment Time Analysis

For a sampled molecule, we obtain two trajectories: the state of the molecule at every time step g_t and the output of the denoiser at every time step $\hat{g}_1(g_t)$. For both of the these trajectories, we record the time of the final type assignment for each atom; this is the time at which an atom occupies its final state and remains in that final state for the remainder of the trajectory. For clarity, Figure 5 shows atom type trajectories for single atoms and shows where the time of final assignment, t_{assign}, occurs in each trajectory.

For all DFM variants other than CTMCs, atom type vectors are not necessarily one-hot vectors during a trajectory. For the analyses presented in Figure 2 and Figure 5, atom type vectors are assigned to discrete states by finding the one-hot vector that is closest, as determined by Euclidean distance, to the current atom type vector.

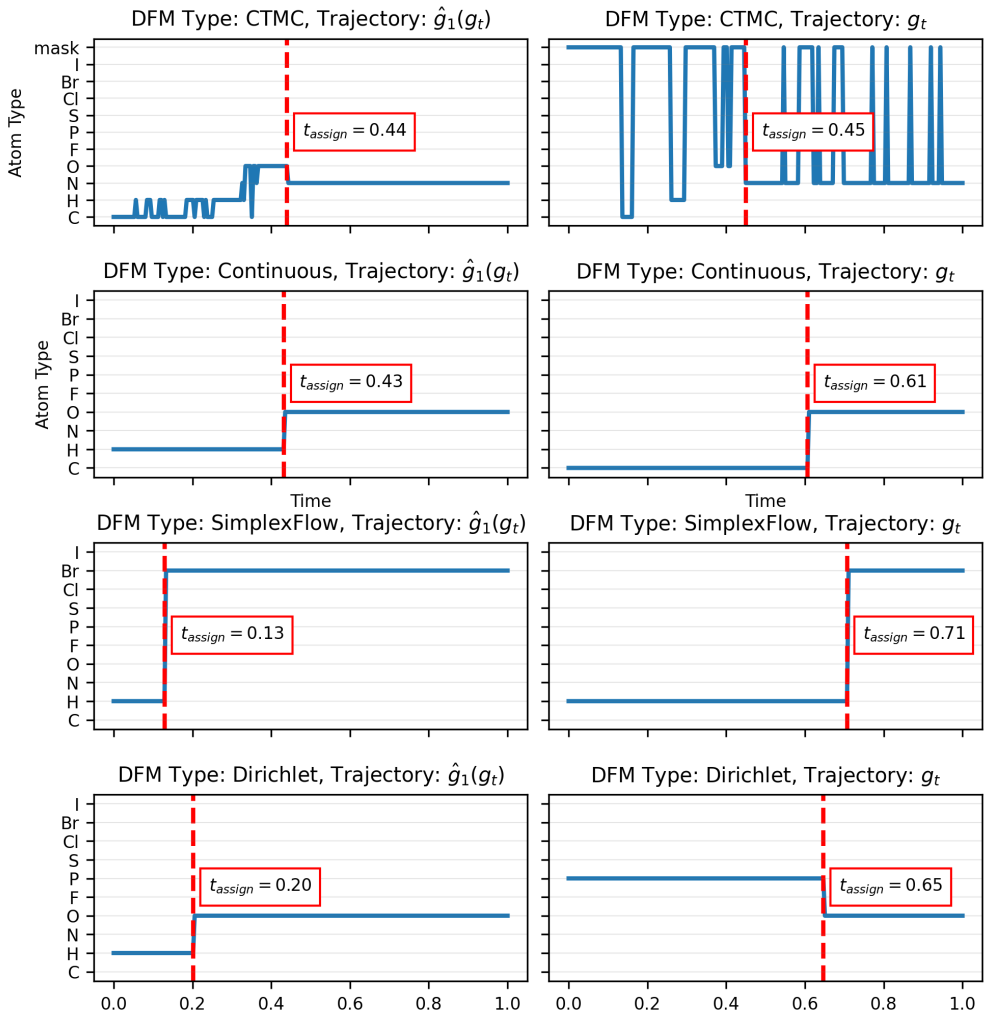

Figure 5: Atom type trajectories and associated times of final type assignment. Each plot contains the atom type trajectory for a single atom. Subplot columns correspond to the trajectory type: either a $\hat{g}_1(g_t)$ or g_t trajectory. Subplot rows correspond to the DFM variant used by the model. Within a row, the $\hat{g}_1(g_t)$ and g_t trajectories displayed are for the same atom. The time of final atom type assignment is overlaid in red on each trajectory.

When computing atom type assignment times for CTMC flows in g_t, masked states are not counted as state changes contributing to t_{assign}; an atom is only considered to switch states when it enters an unmasked state that is different from its previous masked state.

To generate Figure 2, we sampled 100 molecules from each of the four DFM variants of FlowMol. The number of atoms in each molecule was sampled from the atom count distribution from the training data and kept the same across models.

D Model Details

QM9 models are trained with 8 Molecule Update Blocks while GEOM models are trained with 5. Atoms contain 256 hidden scalar features and 16 hidden vector features. Edges contain 128 hidden features. QM9 models are trained for 1000 epochs and GEOM models are trained for 20 epochs. QM9 models are trained on a single L40 GPU with a batch size of 128. GEOM models are trained on 4xL40 GPUs with a per-GPU batch size of 16. QM9 models train in about 3 days while GEOM

models take 4-5 days. All model hyperparameters are visible in the config files provided in our github repository.

$(\eta_X, \eta_A, \eta_C, \eta_E)$ are scalars weighting the relative contribution of each loss term. We set these values to $(3, 0.4, 1, 2)$ as was done in Vignac et al. [34].

E QM9 results

The QM9 dataset [35, 36] contains 124k small molecules, each with one 3D conformation. Molecules in QM9 have an average of 18 atoms and a max of 29. Results for DFM ablations on QM9 are shown in Table 3. FlowMol-CTMC performance relative to baselines on QM9 is shown in Table 4

Table 3: Discrete Flow Type Ablations on QM9

Discrete Flow Type	**Mols Stable (%) (″)**	**Mols Valid (%) (″)**	**JS(E) (#)**
Dirichlet	85.0 1.0	87.4 0.5	0.12 0.01
SimplexFlow	87.9 0.3	92.3 0.2	**0.07** **0.00**
Continuous	96.8 0.4	96.2 0.3	0.10 0.01
CTMC	**99.3** **0.1**	**99.3** **0.1**	0.09 0.00

Table 4: Comparison of FlowMol to baseline models on QM9

Model	**Mols Stable (%) (″)**	**Mols Valid (%) (″)**	**JS(E) (#)**
MiDi	97.5 0.1	98.0 0.2	**0.05** **0.00**
JODO	98.7 0.2	98.9 0.2	0.12 0.01
EQGAT-Diff	98.8 0.1	99.0 0.0	0.08 0.00
FlowMol-CTMC	**99.3** **0.1**	**99.3** **0.1**	0.10 0.00