
Mixed Continuous and Categorical 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. Diffusion models currently achieve state of the art performance for 3D molecule generation. In this work, we explore the use of flow matching, a recently proposed generative modeling framework that generalizes diffusion models, for the task of de novo molecule generation. Flow matching provides flexibility in model design; however, the framework is predicated on the assumption of continuously-valued data. 3D de novo molecule generation requires jointly sampling continuous and categorical variables such as atom position and atom type. We extend the flow matching framework to categorical data by constructing flows that are constrained to exist on a continuous representation of categorical data known as the probability simplex. We call this extension SimplexFlow. We explore the use of SimplexFlow for de novo molecule generation. However, we find that, in practice, a simpler approach that makes no accommodations for the categorical nature of the data yields equivalent or superior performance. As a result of these experiments, we present FlowMol, a flow matching model for 3D de novo generative model that achieves improved performance over prior flow matching methods, and we raise important questions about the design of prior distributions for achieving strong performance in flow matching models. 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 also 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]. State of the art performance in these tasks is presently achieved by applying diffusion models [13–15] to point cloud representations of molecular structures.

Flow matching, a recently proposed generative modeling framework [16–19], generalizes diffusion models. Under diffusion models, the transformation of prior samples to data is formulated as a reversal of a predefined forward process. The forward process is a Markov chain or differential equation that must converge to a tractable stationary distribution as $t \rightarrow \infty$; this requirement constrains the viable options for forward/reverse processes and prior distributions. In contrast, flow matching prescribes a method for directly learning a differential equation that maps samples from nearly arbitrary distributions. In doing so, flow matching permits valuable flexibility when designing models for specific applications. For example, Jing et al. [20] and Stärk et al. [21] make use of the fact that flow matching allows arbitrary prior distributions to design models whose priors are closer to realistic 3D molecular conformations than a Gaussian prior.

In this work we explore the application of flow matching to 3D de novo small molecule generation. We adapt the approach of state of the art diffusion models for this task [22–24] to the flow matching framework. This approach entails predicting atom positions, atom types (chemical elements), formal charges, and bond orders between all pairs

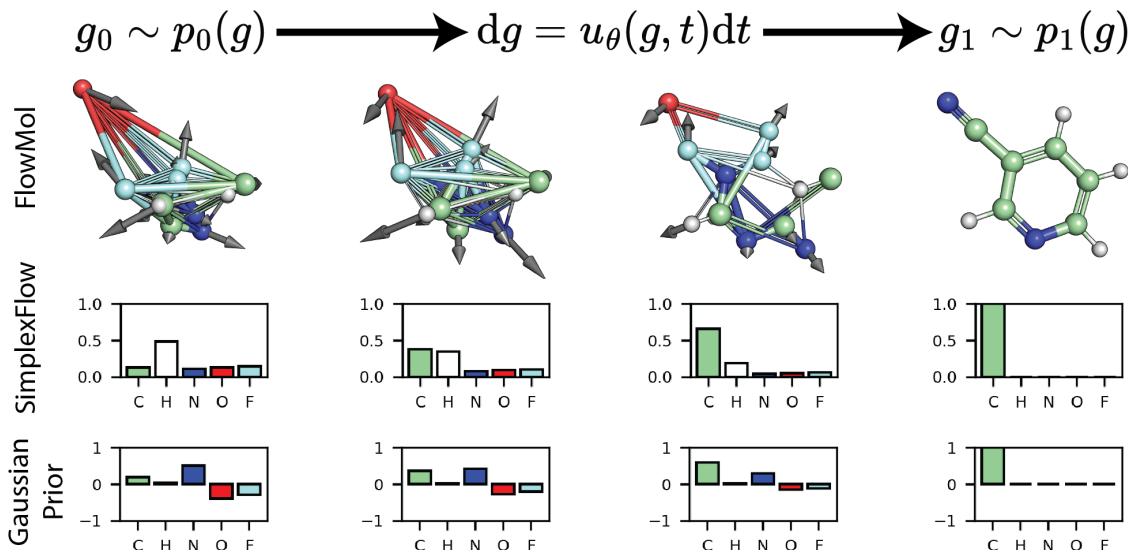


Figure 1: **Overview of FlowMol** *Top:* We adapt the flow matching framework for unconditional 3D molecule generation. An ordinary differential equation parameterized by a graph neural network transforms a prior distribution over atom positions, types, charges, and bond orders to the distribution of valid molecules. Black arrows show the instantaneous direction of the ODE on atom positions. *Middle:* Trajectory of the atom type vector for a single atom under SimplexFlow, a variant of flow matching developed for categorical variables. Atom type flows lie on the probability simplex. *Bottom:* Trajectory of an atom type vector starting from a Gaussian prior. This approach does not respect the categorical nature of the data; however, we find it yields superior performance to SimplexFlow.

of atoms. All of these variables are categorical with the exception of atom positions. Therefore, molecule generation requires sampling from a joint distribution of continuous and categorical variables.

Effectively adapting flow matching for this mixed continuous/categorical generative task may be non-trivial because the flow matching framework is predicated on the assumption of continuously valued data. In this work, we extend the flow matching framework to categorical data by constructing flows that are constrained to exist on a continuous representation of categorical data known as the probability simplex. We call this extension SimplexFlow. We present a model for de novo small-molecule generation that uses SimplexFlows to generate categorical features.

This work was motivated by the intuition that designing a generative process that respects the categorical nature of the data it operates on may yield improved performance; however, our empirical results contradict this intuition. We show that in practice, a simpler approach that makes no accommodations for the categorical nature of the data yields superior performance to a de novo model using SimplexFlow. Our final flow matching model for molecule generation, FlowMol, achieves improved performance over existing flow matching methods for molecule generation and is competitive with state of the art diffusion models while exhibiting a >10-fold reduction in inference time.

2 Background

2.1 Discrete Diffusion

The original formulation of diffusion models [13] was defined in terms of a Markov chain of random variables that converged to a tractable stationary distribution in the limit of an infinite number of steps in the Markov chain. This formulation made no assumptions about the sample space of the random variables modeled, allowing for natural extensions to discrete data [25–27].

A separate formulation of diffusion models as continuous-time stochastic differential equations (SDE)[15] became popular in the literature. The SDE formulation of diffusion models is dependent on the assumption of having continuously-valued data. Similar to our approach, there is a line of work developing SDE-based diffusion models that operate on continuous representations of discrete data. Several works developed diffusion models where diffusion trajectories were constrained to the simplex [28–30]. An alternative approach is to embed categorical features into a continuous latent space and train diffusion models on the embeddings [31].

2.2 De Novo Molecule Generation

Initial attempts at de novo molecule generation focused on generating either textual representations (SMILES strings) [32–34] or 2D molecular graphs [35–38]: molecular representations that exclude all information about 3D structure. Subsequent approaches were developed for 3D molecule generation using a variety of molecular representations and generative paradigms [39–43].

Hoogetboom et al. [44] proposed the first diffusion model for 3D molecule generation, which yielded superior performance over previous approaches. Molecules are represented in Hoogetboom et al. [44] by attributed point clouds where each atom has a position in space and type. A continuous diffusion process is defined for both atom positions and types where the prior for both is a standard Gaussian distribution. A purported weakness of this approach is that atom connectivity is not predicted by the model and must be inferred in a post-processing step. Several concurrent works sought to address these issues by predicting bond order in addition to atom positions/types: Huang et al. [22], Vignac et al. [23], Peng et al. [24], Hua et al. [45]. These models report substantially improved performance over Hoogetboom et al. [44]. Three of these four concurrent works (Vignac et al. [23], Peng et al. [24], Hua et al. [45]) use discrete diffusion processes for categorical features and attribute (in part) their improved model performance to the use of discrete diffusion.; however, only Peng et al. [24] presents an ablation study isolating the effect of discrete diffusion. Moreover, Huang et al. [22] uses only continuous diffusion processes and reports superior performance. This suggests that while predicting graph connectivity provides performance benefits, the utility of discrete diffusion for molecule generation is less clear. Vignac et al. [23] and Huang et al. [22] fully specify the molecular structure by also predicting atom formal charges and the presence of hydrogen atoms; for this reason, these works are the most similar to the model presented here.

2.3 Flow-Matching for De Novo Molecule Generation

To our knowledge, Song et al. [46] is the only existing work that performs de novo molecule generation with flow matching. Molecules are represented as point clouds where each atom has a position in space and an atom type. The final molecule structure is inferred after the inference procedure. The prior distribution for atom type vectors is a standard Gaussian distribution, and so the generative process does not have any inductive biases to respect the discrete nature of the data. This work can be viewed as the flow matching analog of Hoogetboom et al. [44].

2.4 Flow Matching for Discrete Data

Concurrent work [47] developed a variant of flow matching on the simplex which we refer to as Dirichlet Flows. In Dirichlet Flows, conditional probability paths are only conditioned on x_1 and, as a result, do not permit arbitrary choices of the prior and must use a uniform distribution over the simplex. In contrast, our formulation permits the use of any prior distribution. Stark et al. [47] identify problems with the choice of commonly used conditional vector fields that limit performance on variables with a large number of categories. They propose an alternative choice of conditional probability paths that alleviate this issue.

There are also other works which develop flow matching variants for discrete data. Boll et al. [48] equip the simplex with the Fisher-Rao metric to form a Riemannian manifold, and apply Riemannian Flow Matching [49] to this manifold. Campbell et al. [50] develop a flow matching method for discrete data built on continuous-time Markov chains.

Importantly, none of the aforementioned works, which present methods for training flow matching models for categorical data, benchmark their model performance against simpler flow matching models that do not account for the categorical nature of their data.

2.5 Flow Matching

Flow matching [16–19] is a new generative modeling framework that generalizes diffusion models. Flow matching permits useful design flexibility in the choice of prior of and nature of the map between two distributions. Flow matching is also conceptually simpler than diffusion and permits substantially faster inference. We briefly describe the flow matching framework here.

An ordinary differential equation (ODE) that exists on $\mathbb{R}^d$ is defined by a smooth, time-dependent vector-field $u(x, t) : \mathbb{R}^d \times [0, 1] \rightarrow \mathbb{R}^d$.

$$\frac{dx}{dt} = u(x, t) \tag{1}$$

Note that we only consider this ODE on the time interval $[0, 1]$. For simplicity we will use $u_t(x)$ interchangeably with $u(x, t)$. Given a probability distribution over initial positions $x_0 \sim p_0(x)$, the ODE (1) induces time dependent probability distributions $p_t(x)$. The objective in flow matching is to approximate a vector field $u_t(x)$ that pushes a source distribution $p_0(x)$ to a desired target distribution $p_1(x)$. A neural network u_θ can be regressed to the vector field u_t by minimizing the Flow Matching loss.

$$\mathcal{L}_{FM} = \mathbb{E}_{x \sim p_t} \|u_\theta(x, t) - u_t(x)\|^2 \quad (2)$$

Computing $\mathcal{L}_{FM}$ requires access to u_t and p_t , quantities that are typically intractable. Flow matching provides a method for approximating $u_t(x)$ without having access to it. If we consider the probability path $p_t(x)$ to be a mixture of *conditional* probability paths $p_t(x|z)$:

$$p_t(x) = \int p_t(x|z)p(z)dz \quad (3)$$

and we know the form of the conditional vector fields $u_t(x|z)$ that produce $p_t(x|z)$, then the marginal vector field $u_t(x)$ can be defined as a mixture of conditional vector fields:

$$u_t(x) = \mathbb{E}_{p(z)} \frac{u_t(x|z)p_t(x|z)}{p_t(x)} \quad (4)$$

We still cannot compute $u_t(x)$ but the neural network u_θ that is the minimizer of $\mathcal{L}_{FM}$ is also the minimizer of the Conditional Flow Matching (CFM) loss defined in (5)

$$\mathcal{L}_{CFM} = \mathbb{E}_{p(z), p_t(x|z), t \sim \mathcal{U}(0,1)} \|u_\theta(x, t) - u_t(x|z)\|^2 \quad (5)$$

That is, regressing to conditional vector fields, in expectation, is equivalent to regressing to the marginal vector field. The remaining design choices for a flow matching model are the choice of conditioning variable z , conditional probability paths $p_t(x|z)$, and conditional vector fields $u_t(x|z)$.

3 Methods

3.1 Problem Setting

We represent a molecule with N atoms as a fully-connected graph. Each atom is a node in the graph. Every atom has a position in space $X = \{x_i\}_{i=1}^N \in \mathbb{R}^{N \times 3}$, an atom type (in this case the atomic element) $A = \{a_i\}_{i=1}^N \in \mathbb{R}^{N \times n_a}$, and a formal charge $C = \{c_i\}_{i=1}^N \in \mathbb{R}^{N \times n_c}$. Additionally, every pair of atoms has a bond order $E = \{e_{ij}\}_{i,j \in [N], i \neq j} \in \mathbb{R}^{(N^2-N) \times n_e}$. Where n_a, n_c, n_e are the number of possible atom types, charges, and bond orders; these are categorical variables represented by one-hot 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. Concretely, we choose the target distribution that is the distribution of valid 3D molecules $p_1(g) = p(g)$. Our choice of prior $p_0(g)$ is described in Section 3.5.

Our strategy for adapting flow matching to molecular structure is one that mimics prior work on applying diffusion and flow-based generative models to molecular structure. That is, we define conditional vector fields and conditional probability paths for each data modality and jointly regress one neural network for all data modalities. Our total loss is a weighted combination of CFM losses from (5):

$$\mathcal{L} = \eta_X \mathcal{L}_X + \eta_A \mathcal{L}_A + \eta_C \mathcal{L}_C + \eta_E \mathcal{L}_E \quad (6)$$

Where $(\eta_X, \eta_A, \eta_C, \eta_E)$ 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. [23]. Our specific choice of conditional vector fields and probability paths is described in Section 3.2. In practice, we use a variant of the CFM objective called the endpoint-parameterized objective that we present in Section 3.3. These choices are used in turn to design SimplexFlow, our method of performing flow matching for categorical variables, which is described in Section 3.4.

3.2 Flow Matching with Temporally Non-Linear Interpolants

We choose the conditioning variable to be the initial and final states of a trajectory: $z = (g_0, g_1)$. We choose the conditional probability path to be a Dirac density placed on a “straight” line connecting these states $p_t(g|g_0, g_1) = \delta(g - (1 - \alpha_t)g_0 - \alpha_t g_1)$. This particular choice of conditional vector fields and probability paths gives us the freedom to choose any prior distribution $p_0(g)$ [17, 18, 36]. Our choice of $p_t(g|g_0, g_1)$ is equivalent to defining a deterministic interpolant:

$$g_t = (1 - \alpha_t)g_0 + \alpha_t g_1 \quad (7)$$

where $\alpha_t : [0, 1] \rightarrow [0, 1]$ is a function that takes t as input and returns a value between 0 and 1. The rate at which a molecule from the prior distribution g_0 is transformed into a valid molecule g_1 can be controlled by choice of α_t , which we name the “interpolant schedule.”¹ We define separate interpolant schedules for each data type comprising a molecule: $\alpha_t = (\alpha_t^X, \alpha_t^A, \alpha_t^C, \alpha_t^E)$. Taking inspiration from Vignac et al. [23], we define a cosine interpolant schedule:

$$\alpha_t = 1 - \cos^2\left(\frac{\pi}{2}t^\nu\right) \quad (8)$$

where different values of ν are set for atom positions, types, charges, and bond orders. The interpolant (7) gives rise to conditional vector fields of the form:

$$u(g_t|g_0, g_1) = \alpha'_t(g_1 - g_0) \quad (9)$$

Where α'_t is the time derivative of α_t .

3.3 Endpoint Parameterization

By solving (7) for g_0 and substituting this expression into (9) we obtain an alternate form of the conditional vector field.

$$u(g_t|g_0, g_1) = \frac{\alpha'_t}{1 - \alpha_t}(g_1 - g_t) \quad (10)$$

As described in Section 2.5, the typical flow matching procedure is to regress a neural network $u_\theta(g_t)$ directly to conditional vector fields by minimizing the CFM loss (5). Instead, we apply a reparameterization initially proposed by Jing et al. [20]:

$$u_\theta(g_t) = \frac{\alpha'_t}{1 - \alpha_t}(\hat{g}_1(g_t) - g_t) \quad (11)$$

By substituting (11) and (10) into (5), we obtain our endpoint-parameterized objective

$$\mathcal{L}_{EP} = \mathbb{E}_{t, g_t} \left[\frac{\alpha'_t}{1 - \alpha_t} \|\hat{g}_1(g_t) - g_1\| \right] \quad (12)$$

Therefore our objective becomes to train a neural network that predicts valid molecular structures given samples from a conditional probability path $\hat{g}_1(g_t)$. This is particularly advantageous when operating on categorical data, as placing a softmax layer on model outputs constrains the domain of model outputs to the simplex. Empirically, we find that the endpoint objective yields better performance than the vector field regression objective (5) for the task of molecule generation. Moreover, we leverage the theoretical guarantee that our predicted endpoint for categorical data lie on the simplex to ensure our flows lie on the simplex.

In practice, the interpolant-dependent loss weight $\frac{\alpha'_t}{1 - \alpha_t}$ produces unreasonably large values as $\alpha_t \rightarrow 1$. We replace this term with a time-dependent loss function inspired by Le et al. [51]: $w(t) = \min(\max(0.005, \frac{\alpha_t}{1 - \alpha_t}), 1.5)$. For categorical variables we use a cross entropy loss rather than the L2 norm shown in (12).

¹This is intended to be analogous to noise schedules for diffusion models.

3.4 SimplexFlow

To design flow matching for categorical data, our strategy is to define a continuous representation of categorical variables, and then construct a flow matching model where flows are constrained to this representation. We choose the d -dimensional probability simplex $\mathcal{S}^d$ as the continuous representation of a d -categorical variable.

$$\mathcal{S}^d = \{x \in \mathbb{R}^d | x_i > 0, \mathbb{1} \cdot x = 1\} \quad (13)$$

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

$$p_1(x) = \sum_{i=1}^d q_i \delta(x - e_i) \quad (14)$$

where e_i is the i^{th} vertex of the simplex and q_i is the probability of x belonging to the i^{th} category. If we choose a prior distribution $p_0(x)$ such that $\text{supp}(p_0) = \mathcal{S}^d$, then all conditional probability paths produced by the interpolant (7) will lie on the simplex. This is because the simplex is closed under linear interpolation (see Appendix A) and the conditional trajectories are obtained by linearly interpolating between two points on the simplex ($x_0, x_1 \in \mathcal{S}^d$).

Although choosing (p_0, p_1) with support on the simplex results in conditional trajectories on the simplex, training a flow under the vector field objective (5) provides no guarantee that trajectories produced by the learned vector field lie on the simplex. However, training a flow matching model under the endpoint parameterization (Section 3.3) enables us to guarantee by construction that generated flows lie on the simplex; proof of this is provided in Appendix B.

3.5 Priors

We define the prior distribution for a molecule as a composition of independent samples for each atom and pair of atoms. Our prior distributions take the form:

$$p_0(g) = p_0(x, a, c, e) = \prod_{i=1}^{N_{\text{atoms}}} p_0(x_i) p_0(a_i) p_0(c_i) \times \prod_{i,j < i}^{N_{\text{atoms}}} p_0(e_{ij}) \quad (15)$$

Our choice of conditional trajectory (7) permits the choice of any prior distribution. SimplexFlow places the constraint that the prior distribution for categorical variables have support bounded to the simplex.

We always set $p_0(x_i) = \mathcal{N}(x_i | 0, \mathbb{I})$; atom positions are independently sampled from a standard Gaussian distribution. We explore the use of several prior distributions for categorical variables a_i, c_i, e_{ij} . We experiment with three different categorical priors for SimplexFlow. The **uniform-simplex** prior is a uniform distribution over the simplex; the simplest choice for a categorical prior. This choice is analogous to the "Linear FM" model described in [47]. The **marginal-simplex** prior is designed to be "closer" to the data distribution by using marginal distributions observed in the training data. Specifically, we replace $p_0(a_i)p_0(c_i)$ and $p_0(e_{ij})$ in (15) with $p_1(a_i, c_i)$ and $p_1(e_{ij})$, respectively. Finally, for the **barycenter** prior, categorical variables are placed at the barycenter of the simplex; the point in the center of the simplex assigning equal probability to all categories. The intuition behind the barycenter prior is all categorical variables will be "undecided" at $t = 0$.

In practice, the model fails when the prior distributions for categorical variables only have density on a small, fixed number of points on the simplex; this is the case for the marginal-simplex and barycenter priors. We find that "blurring" the prior samples for categorical variables significantly improves performance. That is, Gaussian noise is added to the samples before they are projected back onto the simplex.

3.6 Optimal Transport Alignment

Previous work [17] has shown that aligning prior and target samples via optimal transport significantly improves the performance of flow matching by minimizing the extent to which conditional trajectories intersect. When performing flow matching on molecular structure, this consists of computing the optimal permutation of node ordering and the rigid-body alignment of atom positions [46, 52]. We apply the same alignment between target and prior positions at training time. This also ensures that prior positions $p_0(X)$ and target positions $p_1(X)$ effectively exist in the center of mass free subspace proposed in Hoogetboom et al. [44] that renders the target density $p_1(g)$ invariant to translations.

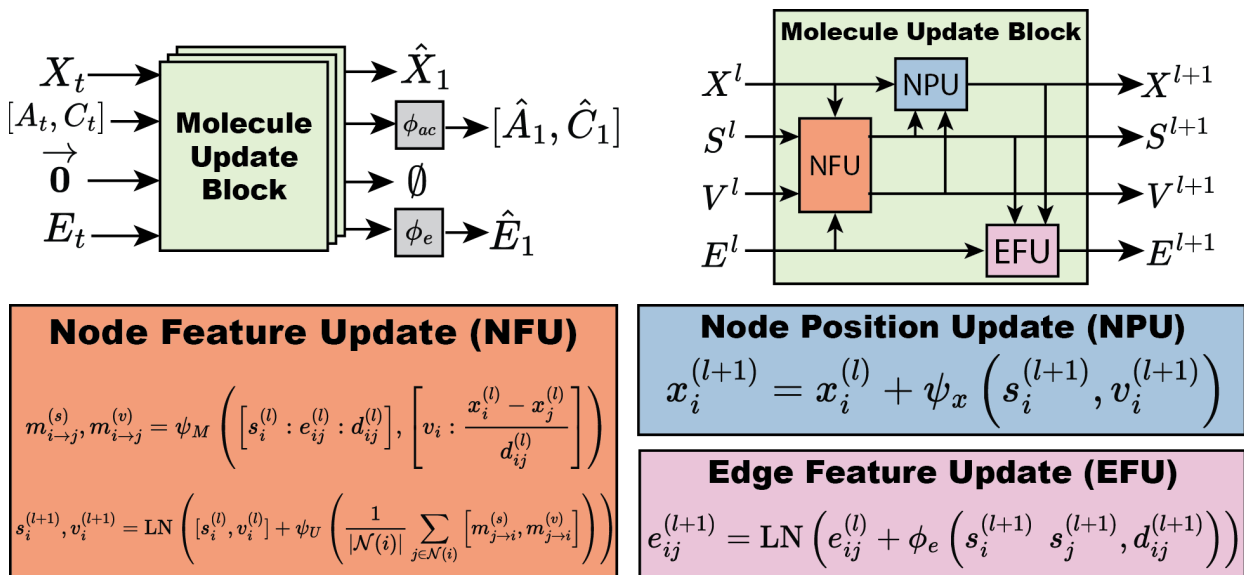


Figure 2: **FlowMol Architecture** *Top left*: An input molecular graph g_t is transformed into a predicted final molecular graph g_1 by being passed through multiple molecule update blocks. *Top right*: A molecule update block uses NFU, NPU, and EFU sub-components to update all molecular features. *Bottom*: Update equations for graph features. ϕ and ψ is used to denote MLPs and GVPs, respectively.

3.7 Model Architecture

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 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) [53] 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 2 and explained in detail in Appendix D.

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, $\hat{e}_1^{ij} = \text{MLP}(e_{ij} + e_{ji})$.

GVPs, as they were originally designed, predict vector quantities that are E(3)-equivariant. We introduce a variant of GVP that is made SE(3)-equivariant by the addition of cross product operations. The cross product is equivariant to rotations and translations of input vectors but not reflections. As a result, the learned density $p_1(g)$ is invariant to rotations and translations but not reflections. In other words, FlowMol is sensitive to chirality. Empirically we find that the addition of cross product operations to GVP improves performance. Schneuing et al. [3] proposed the addition of a cross product operation to the EGNN architecture [54]; we adopt this idea for GVP. We refer the reader to Appendix F of Schneuing et al. [3] for a detailed discussion of the equivariance of cross products. Our cross product variant of GVP is described in Appendix D.1.

4 Experiments

4.1 Datasets

We train on QM9 [55, 56] and GEOM-Drugs [57] using explicit hydrogens. QM9 contains 124k small molecules, each with one 3D conformation. GEOM-Drugs contains approximately 300k larger, drug-like molecules with multiple conformers for each molecule. Molecules in QM9 have an average of 18 atoms a max of 29 while those in GEOM-Drugs have an average of 44 atoms and a max of 181. We use the same dataset splits as Vignac et al. [23].

We chose to use explicit hydrogens because it is a more difficult learning task. By predicting explicit hydrogens in combination with atom types, bond orders, and formal charges, there is a 1-to-1 mapping from model outputs to molecules. If any one of these components were removed from the generative task, one model output could plausibly be interpreted as multiple molecular structures, and so it is “easier” for the model output to be interpreted as “correct” or “valid.” We view the task of predicting graph topology and structure with explicit hydrogens and formal charges as the most rigorous evaluation of the capabilities of generative models to fit the distribution of valid molecular structures.

4.2 Model Evaluation

We report three metrics measuring the validity of generated molecular topology: percent atoms stable, percent molecules stable, and percent molecules valid. An atom is defined as “stable” if it has valid valency. Atomic valency is defined as the sum of bond orders that an atom is participating atom. Aromatic bonds are assigned a bond order of 1.5. A valid valency is defined as any valency that is observed in the training data for atoms of a given element and formal charge. 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 [58] using default sanitization settings.

Metrics regarding the validity of molecular topology fail to capture a model’s ability to reproduce reasonable molecular geometries. Therefore, we also compute the Jensen-Shannon divergence of the distribution of potential energies for molecules in the training data and molecules sampled from trained models. Potential energies are obtained from the Merck Molecular Mechanics Force-Field implemented in rdkit [58]. Force-field energy cannot be obtained for molecules that cannot be sanitized by rdkit, and so the reported Jensen-Shannon divergences are for valid molecules only.

Molecule quality metrics are reported for samples of 10,000 molecules, repeated 5 times. We report inference time for FlowMol and baseline models. We measure inference time as the time required to generate one batch of 100 molecules on the same NVIDIA GeForce RTX 2060 GPU. This inference procedure is also repeated five times. Inference is run on FlowMol using Euler integration with 100 evenly-spaced timesteps. All results are reported with 95% confidence intervals. For all samplings, the number of atom in each molecule is sampled from the distribution of atoms in molecules from the training data.

4.3 Model Ablations

We train multiple versions of our model to evaluate the effects of several aforementioned design choices. To observe the effect of endpoint reparameterization (sec 3.3), we train equivalent models with both the vector-field objective (5) and the endpoint objective (12). We train models using SimplexFlow with all three categorical priors proposed in Section 3.5 which have support on the simplex. To determine whether SimplexFlow improves performance, we also train models where the prior distribution for categorical features is a standard Gaussian distribution. In this setting, the generated flows are not constrained to the simplex, and it can be said that the flows do not “respect” the categorical nature of the data. This is similar to the atom type flows in Song et al. [46] and atom type diffusion in Hooeboom et al. [44].

All of the mentioned model ablations are tested on the QM9 dataset and the results are presented in Section 5.1. A subset of these ablations were also performed on the GEOM dataset. GEOM ablations are available in Appendix E. None of the effects observed in GEOM ablations contradict those seen for QM9 ablations. For metrics reported in ablations, results are averaged over two identical models trained with different random seeds

4.4 Comparison to Dirichlet Flows

We compare SimplexFlow to concurrent work that developed Dirichlet Flows [47] for flow matching on the simplex. Briefly, for a d -categorical variable x represented as a point on the simplex, the conditional probability path is

$$p_t(x|x_1 = e_i) = \text{Dir}(x|\gamma = 1 + e_i\omega) \quad (16)$$

Table 1: FlowMol Ablations on QM9 with explicit hydrogens

Flow Type	Categorical Prior	Atoms Stable (%) ($\uparrow$)	Mols Stable (%) ($\uparrow$)	Mols Valid (%) ($\uparrow$)	JS(E) ($\downarrow$)
Dirichlet endpoint	uniform-simplex	98.4 $\pm$ 0.0	80.0 $\pm$ 0.3	85.5 $\pm$ 0.3	0.15 $\pm$ 0.01
	uniform-simplex	98.9 $\pm$ 0.1	84.2 $\pm$ 0.9	88.9 $\pm$ 0.6	0.11 $\pm$ 0.01
	marginal-simplex	99.5 $\pm$ 0.1	91.9 $\pm$ 0.7	96.1 $\pm$ 0.2	0.06 $\pm$ 0.00
	barycenter	99.5 $\pm$ 0.0	91.4 $\pm$ 0.5	93.6 $\pm$ 0.5	0.05$\pm$0.00
	Gaussian	99.7$\pm$0.0	96.0$\pm$0.1	96.9$\pm$0.1	0.09 $\pm$ 0.01
vector-field	marginal-simplex	98.6 $\pm$ 0.0	79.4 $\pm$ 0.3	86.2 $\pm$ 0.3	0.07 $\pm$ 0.00
	Gaussian	99.5 $\pm$ 0.0	93.6 $\pm$ 0.7	94.7 $\pm$ 0.7	0.08 $\pm$ 0.01

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 $\delta(x - e_i)$ in the limit $\omega \rightarrow \infty$. In order to incorporate Dirichlet flows into our model, we define the relation $\omega_t = \omega_{\max}\alpha_t + 1$, where α_t is defined by (8). Dirichlet flow matching necessitates the use of a uniform prior over the simplex for categorical variables and so we do not experiment with other simplex priors described in Section 3.5.

4.5 Baselines

We compare FlowMol to three baselines: MiDi [23], JODO [22], and EquiFM [46]. MiDi and JODO perform the same generation task: predicting atom positions, atom types, formal charges, and bond orders. The key difference from FlowMol is that MiDi and JODO are diffusion models. EquiFM as described in Section 2.3 is a flow matching model for de novo molecule generation; however, the model does not predict bond orders or atomic charges. We do not report the performance of EquiFM on the GEOM dataset because the authors have not released a model checkpoint.

5 Results

5.1 Model Ablations

Results of model ablation experiments on the QM9 dataset are shown in Table 1. Most notably, models that use SimplexFlow for categorical variables (those with categorical priors constrained to the simplex) consistently underperform models with Gaussian categorical priors. The best performing SimplexFlow model (endpoint parameterization, marginal-simplex prior) achieves 96.1% valid molecules while an equivalent model using a Gaussian prior achieves 96.9% valid molecules.

Models trained under the endpoint objective achieve superior performance to otherwise identical models trained under the vector-field objective. For example, Table 1 shows that a model trained with the a marginal-simplex categorical prior obtains 79% stable molecules under the vector-field objective and 92% stable molecules under the endpoint objective. This is effect is also observed with models using a Gaussian categorical prior but to a lesser extent.

We find that models using Dirichlet conditional probability paths [47] yields approximately equivalent performance to the conditional probability path (7) with a uniform-simplex categorical prior. Among models satisfying the constraints of SimplexFlow (sec. 3.4), the uniform-simplex prior yielded the worst performance. The marginal-simplex and barycenter priors yield approximately equivalent performance. Although the models using marginal-simplex and barycenter priors produce relatively fewer valid molecules, the molecules generated by these models exhibit the lowest Jensen-Shannon divergence to the energy distribution of the training data.

5.2 Comparison with Baselines

FlowMol achieves superior performance to EquiFM [46] on QM9; for example, it produces 3% more valid molecules while having equivalent divergence to the training data energy distribution. FlowMol approaches the performance of diffusion baselines (JODO, MiDi) on QM9 but does not perform as well on the GEOM-Drugs dataset. The fact that fewer generated molecules are valid on the GEOM-Drugs dataset cannot be attributed solely to the difference in molecule sizes between the two datasets, because FlowMol’s atom-level stability is also worse for GEOM-Drugs than QM9 (99.0% on GEOM vs 99.7% on QM9). Despite the fact that MiDi and FlowMol achieve equivalent atom-level stability (99.0%), MiDi produces significantly more topologically correct molecules. For example, FlowMol achieves 68% stable molecules while MiDi achieves 85%.

Table 2: Comparison of FlowMol to baseline models on the QM9 and GEOM-Drugs datasets

Model	Dataset	Atoms Stable (%) ($\uparrow$)	Mols Stable (%) ($\uparrow$)	Mols Valid (%) ($\uparrow$)	JS(E) ($\downarrow$)	Inference Time (s) ($\downarrow$)
JODO [22]	QM9	99.9 $\pm$ 0.0	98.7 $\pm$ 0.2	98.9 $\pm$ 0.2	0.12 $\pm$ 0.01	116 $\pm$ 2
MiDi [23]		99.8 $\pm$ 0.0	97.5 $\pm$ 0.1	98.0 $\pm$ 0.2	0.05 $\pm$ 0.00	89 $\pm$ 7
EquiFM [46]		99.4 $\pm$ 0.0	93.2 $\pm$ 0.3	94.4 $\pm$ 0.2	0.08 $\pm$ 0.00	25 $\pm$ 3
FlowMol (ours)		99.7 $\pm$ 0.0	96.2 $\pm$ 0.1	97.3 $\pm$ 0.1	0.08 $\pm$ 0.00	6 $\pm$ 0
JODO [22]	GEOM-Drugs	99.8 $\pm$ 0.0	90.7 $\pm$ 0.5	76.5 $\pm$ 0.8	0.17 $\pm$ 0.01	235 $\pm$ 16
MiDi [23]		99.0 $\pm$ 0.2	85.1 $\pm$ 0.9	71.6 $\pm$ 0.9	0.23 $\pm$ 0.00	754 $\pm$ 119
FlowMol (ours)		99.0 $\pm$ 0.0	67.5 $\pm$ 0.2	51.2 $\pm$ 0.3	0.33 $\pm$ 0.01	22 $\pm$ 1

FlowMol exhibits substantially faster inferences times than all baseline models. This difference is primarily due to the fewer number of integration steps needed by FlowMol. We find empirically that sample quality does not improve when using more than 100 integration steps. JODO, MiDi, and EquiFM use 1000 integration steps by default. The need for fewer integration steps than diffusion models is a recognized advantage of flow matching models over diffusion [17, 19].

6 Discussion

FlowMol improves upon the existing state of the art flow matching method for molecule generation; however, it still does not outperform diffusion models trained for the same task. A key difference between FlowMol and the diffusion baselines presented here is that the conditional trajectories are deterministic in FlowMol and stochastic in diffusion models. Prior works have presented theoretical [18] and empirical [21] evidence that stochastic conditional trajectories yield improved model performance.

Our results raise interesting questions about the design of prior distributions for flow matching models. Our intuition was that a stronger prior that is “closer” to the data distribution would yield more faithful recapitulation of the target distribution. The results of our model ablations suggest this intuition is incorrect. The next natural questions are: why is a Gaussian prior the most performant of those tested here? and what are the qualities of a prior that best enable recapitulation of the target distribution? A possible explanation for our results is a dependence on the “volume” of the prior. Empirically when the prior for categorical features has support on a small number of unique values, the model fails to produce any valid molecules. Adding a “blur” as described in Section 3.5 dramatically improves model performance. Correspondingly, priors constrained to the simplex reliably yield poorer performance than Gaussian priors; these observations could all be explained through the perspective of the prior’s capacity for serving as one domain of a homeomorphism to a more complex distribution.

Another explanation for the superiority of Gaussian priors may involve the shape of conditional trajectories induced by the prior. Conditional trajectories are more likely to intersect when constrained to a smaller space, such as the simplex. This explanation is also supported by the observation that the marginal-simplex and barycenter priors yield substantially improved performance over uniform-simplex priors. Tong et al. [17] suggest that sampling conditional pairs (g_0, g_1) from an optimal transport (OT) alignment $\pi(g_0, g_1)$ improves performance precisely because the marginal vector field yields straighter lines with fewer intersections. In this work, an OT plan is computed but only for atomic positions. Perhaps computing an OT alignment over the product space of all the data modalities represented here could alleviate this issue.

7 Conclusions

FlowMol is the first generative model to jointly sample the topological and geometric structure of small molecules. FlowMol improves upon existing flow matching models for molecule generation and achieves competitive performance with diffusion-based models while exhibiting inference speeds an order of magnitude faster. We present a method for flow matching on categorical variables, SimplexFlow, and demonstrate that constraining flows to a smaller space does not yield performance benefits. We think this result raises interesting and relevant questions about the design of flow matching for mixed continuous/categorical generative tasks and provide potential hypotheses to begin exploring in future work.

8 Acknowledgements

We thank Rishal Aggarwal, Gabriella Gerlach, and Daniel Peñahererra for useful feedback and discussions.

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 Goreschnik, 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, Jarrod 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] Bowen Jing, Bonnie Berger, and Tommi Jaakkola. AlphaFold Meets Flow Matching for Generating Protein Ensembles, February 2024. URL <http://arxiv.org/abs/2402.04845>. arXiv:2402.04845 [cs, q-bio].
 - [21] Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Harmonic Self-Conditioned Flow Matching for Multi-Ligand Docking and Binding Site Design, March 2024. URL <http://arxiv.org/abs/2310.05764>. arXiv:2310.05764 [cs].
 - [22] 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].
 - [23] 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].
 - [24] Xingang Peng, Jiaqi Guan, Qiang Liu, and Jianzhu Ma. MolDiff: Addressing the Atom-Bond Inconsistency Problem in 3D Molecule Diffusion Generation, May 2023. URL <http://arxiv.org/abs/2305.07508>. arXiv:2305.07508 [cs, q-bio].
 - [25] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions, October 2021. URL <http://arxiv.org/abs/2102.05379>. arXiv:2102.05379 [cs, stat].
 - [26] Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A Continuous Time Framework for Discrete Denoising Models, October 2022. URL <http://arxiv.org/abs/2205.14987>. arXiv:2205.14987 [cs, stat].
 - [27] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces, February 2023. URL <http://arxiv.org/abs/2107.03006>. arXiv:2107.03006 [cs].

- [28] Pierre H. Richemond, Sander Dieleman, and Arnaud Doucet. Categorical SDEs with Simplex Diffusion, October 2022. URL <http://arxiv.org/abs/2210.14784>. arXiv:2210.14784 [cs].
- [29] Griffin Floto, Thorsteinn Jonsson, Mihai Nica, Scott Sanner, and Eric Zhengyu Zhu. Diffusion on the Probability Simplex, September 2023. URL <http://arxiv.org/abs/2309.02530>. arXiv:2309.02530 [cs, stat].
- [30] Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet Diffusion Score Model for Biological Sequence Generation, June 2023. URL <http://arxiv.org/abs/2305.10699>. arXiv:2305.10699 [cs, q-bio].
- [31] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data, December 2022. URL <http://arxiv.org/abs/2211.15089>. arXiv:2211.15089 [cs].
- [32] Francesca Grisoni, Michael Moret, Robin Lingwood, and Gisbert Schneider. Bidirectional Molecule Generation with Recurrent Neural Networks. *Journal of Chemical Information and Modeling*, 60(3):1175–1183, March 2020. ISSN 1549-9596. doi: 10.1021/acs.jcim.9b00943. URL <https://doi.org/10.1021/acs.jcim.9b00943>. Publisher: American Chemical Society.
- [33] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*, 4(2):268–276, February 2018. ISSN 2374-7943. doi: 10.1021/acscentsci.7b00572. URL <https://doi.org/10.1021/acscentsci.7b00572>. Publisher: American Chemical Society.
- [34] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-Directed Variational Autoencoder for Structured Data, February 2018. URL <http://arxiv.org/abs/1802.08786>. arXiv:1802.08786 [cs].
- [35] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction Tree Variational Autoencoder for Molecular Graph Generation, March 2019. URL <http://arxiv.org/abs/1802.04364>. arXiv:1802.04364 [cs, stat].
- [36] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. Constrained Graph Variational Autoencoders for Molecule Design, March 2019. URL <http://arxiv.org/abs/1805.09076>. arXiv:1805.09076 [cs, stat].
- [37] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation, February 2020. URL <http://arxiv.org/abs/2001.09382>. arXiv:2001.09382 [cs, stat].
- [38] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, February 2019. URL <http://arxiv.org/abs/1806.02473>. arXiv:1806.02473 [cs, stat].
- [39] Matthew Ragoza, Tomohide Masuda, and David Ryan Koes. Learning a Continuous Representation of 3D Molecular Structures with Deep Generative Models, November 2020. URL <http://arxiv.org/abs/2010.08687>. arXiv:2010.08687 [cs, q-bio].
- [40] Matthew Ragoza, Tomohide Masuda, and David Ryan Koes. Generating 3D molecules conditional on receptor binding sites with deep generative models. *Chemical Science*, 13(9):2701–2713, March 2022. ISSN 2041-6539. doi: 10.1039/D1SC05976A. URL <https://pubs.rsc.org/en/content/articlelanding/2022/sc/d1sc05976a>. Publisher: The Royal Society of Chemistry.
- [41] Niklas W. A. Gebauer, Michael Gastegger, and Kristof T. Schütt. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules, January 2020. URL <http://arxiv.org/abs/1906.00957>. arXiv:1906.00957 [physics, stat].
- [42] Youzhi Luo and Shuiwang Ji. An Autoregressive Flow Model for 3D Molecular Geometry Generation from Scratch. October 2021. URL <https://openreview.net/forum?id=C03Ajc-NS5W>.
- [43] Victor Garcia Satorras, Emiel Hoogetboom, Fabian B. Fuchs, Ingmar Posner, and Max Welling. E(n) Equivariant Normalizing Flows, January 2022. URL <http://arxiv.org/abs/2105.09016>. arXiv:2105.09016 [physics, stat].

- [44] Emiel Hooeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant Diffusion for Molecule Generation in 3D, June 2022. URL <http://arxiv.org/abs/2203.17003>. arXiv:2203.17003 [cs, q-bio, stat].
- [45] Chenqing Hua, Sitao Luan, Minkai Xu, Rex Ying, Jie Fu, Stefano Ermon, and Doina Precup. MUDiff: Unified Diffusion for Complete Molecule Generation, February 2024. URL <http://arxiv.org/abs/2304.14621>. arXiv:2304.14621 [cs, q-bio].
- [46] 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].
- [47] 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].
- [48] Bastian Boll, Daniel Gonzalez-Alvarado, and Christoph Schnörr. Generative Modeling of Discrete Joint Distributions by E-Geodesic Flow Matching on Assignment Manifolds, February 2024. URL <http://arxiv.org/abs/2402.07846>. arXiv:2402.07846 [cs, stat].
- [49] Ricky T. Q. Chen and Yaron Lipman. Flow Matching on General Geometries, February 2024. URL <http://arxiv.org/abs/2302.03660>. arXiv:2302.03660 [cs, stat].
- [50] 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, February 2024. URL <http://arxiv.org/abs/2402.04997>. arXiv:2402.04997 [cs, q-bio, stat].
- [51] 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].
- [52] Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching, November 2023. URL <http://arxiv.org/abs/2306.15030>. arXiv:2306.15030 [physics, stat].
- [53] 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].
- [54] Victor Garcia Satorras, Emiel Hooeboom, and Max Welling. E(n) Equivariant Graph Neural Networks, February 2022. URL <http://arxiv.org/abs/2102.09844>. arXiv:2102.09844 [cs, stat].
- [55] 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.
- [56] 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.
- [57] 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.
- [58] RDKit. URL <http://www.rdkit.org/>.
- [59] 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 Proof that the simplex is closed under linear interpolation

Assume we have access to two vectors on the simplex $a, b \in \mathcal{S}^d$. We define another point c by linear interpolation between a and b :

$$c := ta + (1 - t)b \quad \text{where } t \in [0, 1] \quad (17)$$

Therefore, each entry of c can be written as $c_i = ta_i + (1 - t)b_i$. Given that $a_i, b_i \geq 0$ by definition, c_i must also be positive. We can also write the sum of the entries of c as:

$$\sum_{i=1}^d c_i = \sum_{i=1}^d ta_i + (1 - t)b_i \quad (18)$$

$$= t \sum_{i=1}^d a_i + (1 - t) \sum_{i=1}^d b_i \quad (19)$$

$$= t + (1 - t) = 1 \quad (20)$$

We have shown that $c_i \geq 0$ and $\sum_{i=1}^d c_i = 1$. Therefore $c \in \mathcal{S}^d$ and we can conclude that $\mathcal{S}^d$ is closed under linear interpolation.

B Proof that flows are on the simplex

We will define a flow matching model for a categorical variable y that can take one of d discrete values. The target distribution for y , $p_1(y)$ is a mixture of point masses on the vertices of $\mathcal{S}^d$. We add the constraint that $p_0(y)$ only has support on $\mathcal{S}^d$.

We choose conditional trajectories of the form (7) and a cosine interpolant schedule of the form (8). We train a neural network to minimize the endpoint objective (12), and as a result we have access to a model which predicts the endpoint of our trajectory given a current position: $\hat{y}_1(y_t)$.

To generate samples from $p_1(y)$, we first sample $y_0 \sim p_0(y)$, and integrate the ODE

$$\frac{dy}{dt} = u_\theta(y) = \frac{\alpha'_t}{1 - \alpha_t}(\hat{y}_1 - y_t) \quad (21)$$

Integration via Euler's method would produce trajectories according to:

$$y_s = y_t + u_\theta(y_t)(s - t) \quad (22)$$

Where $s = t + \Delta t$. In order to prove that all trajectories produced by our vector field lie on the simplex, it would be sufficient to prove that $y_s \in \mathcal{S}^d$ in the limit as $\Delta t \rightarrow 0$.

Substituting (21) into (22) yields the following update rule for integration by Euler's method:

$$y_s = \frac{\alpha'_t(s - t)}{1 - \alpha_t} \hat{y}_1 + \frac{1 - \alpha_t - \alpha'_t(s - t)}{1 - \alpha_t} y_t \quad (23)$$

Relying on the property that the simplex is closed under linear interpolation (Appendix A), our strategy is to prove that integrating trajectories via (23) results in recursive applications of linear interpolation between two points on the simplex.

More formally, the right hand side of (23) would be linear interpolation between two points on the simplex if the following conditions were satisfied:

1. $\frac{\alpha'_t(s - t)}{1 - \alpha_t} + \frac{1 - \alpha_t - \alpha'_t(s - t)}{1 - \alpha_t} = 1$
2. $\frac{\alpha'_t(s - t)}{1 - \alpha_t} \in [0, 1]$

3. $\hat{y}_1(y_t), y_t \in \mathcal{S}^d$

The first condition is obviously true.

The second condition can be written as two inequalities $0 \leq \frac{\alpha'_t(s-t)}{1-\alpha_t} \leq 1$. The first inequality reduces to $\alpha'_t \geq 0$; the interpolant must be monotonically increasing. The second inequality can be seen as an upper bound on the step size that can be used during integration:

$$s - t \leq \frac{1 - \alpha_t}{\alpha'_t} \quad (24)$$

For well-behaved interpolant schedules and many reasonable choices of interpolant, this inequality is satisfied in the limit as $s - t \rightarrow 0$.

Regarding the third condition: by definition, $\hat{y}_1 \in \mathcal{S}^d$; this is practically enforced by placing softmax activations on the output of the neural network. If $y_0 \in \mathcal{S}^d$, a condition which is guaranteed by our choice of prior $p_0(y)$, then the first application of the update rule (23) would satisfy all three conditions and as a result $y_{0+\Delta t} \in \mathcal{S}^d$. By induction, every subsequent application of the update rule (23) would yield an integration step that is linear interpolation between two points on the simplex.

As a result, all trajectories generated by the ODE will lie on the simplex in the limit of a infinitely small integration step. And, in practice, infinitely small integration steps are not actually necessary to yield trajectories on the simplex. More on this in Appendix C.

C Interpolation Schedules and Integration Step Sizes

The relative rates at which molecular features are generated are determined by setting values of the parameter ν in the cosine interpolant schedule (8). For the QM9 dataset we set $\nu = (\nu_X, \nu_A, \nu_C, \nu_E) = (1, 2, 2, 1.5)$. For the GEOM-Drugs dataset this is set to $(1, 2, 2, 2)$. These are the same values used for the cosine noise schedule in Vignac et al. [23]. The interpolant schedules used for the QM9 dataset are plotted in Figure 3.

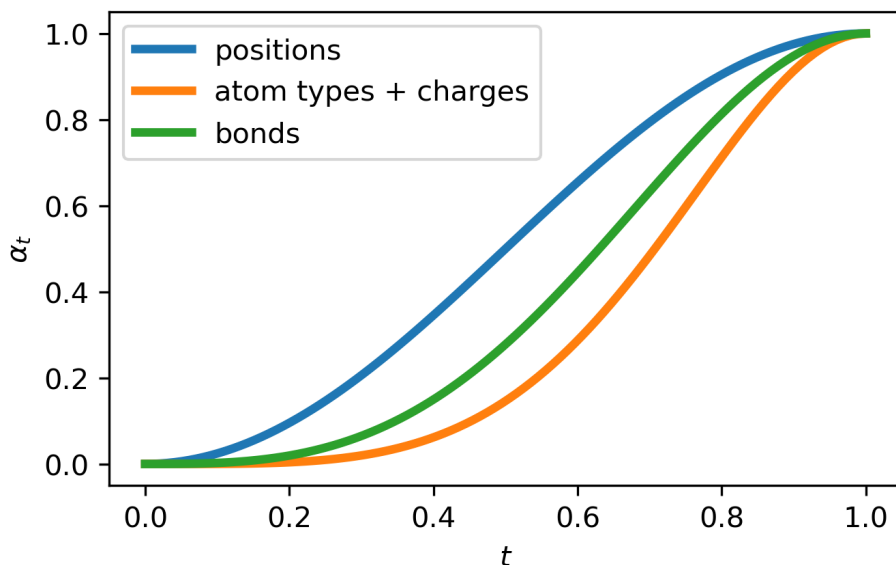


Figure 3: Interpolant Schedules for the QM9 Dataset

In Appendix B we derive an upper bound on the integration step size that can be used that guarantees SimplexFlow trajectories will remain on the simplex (24). In Figure 4 we plot this maximum step size as a function of t for cosine interpolation schedules (8). For the results presented in this paper we sample molecules by performing Euler integration with 100 evenly-spaced integration steps. This corresponds to a constant step size of 10^{-2} . According to Figure 4, this step size ensures trajectories will remain on the simplex until approximately $t = 0.98$.

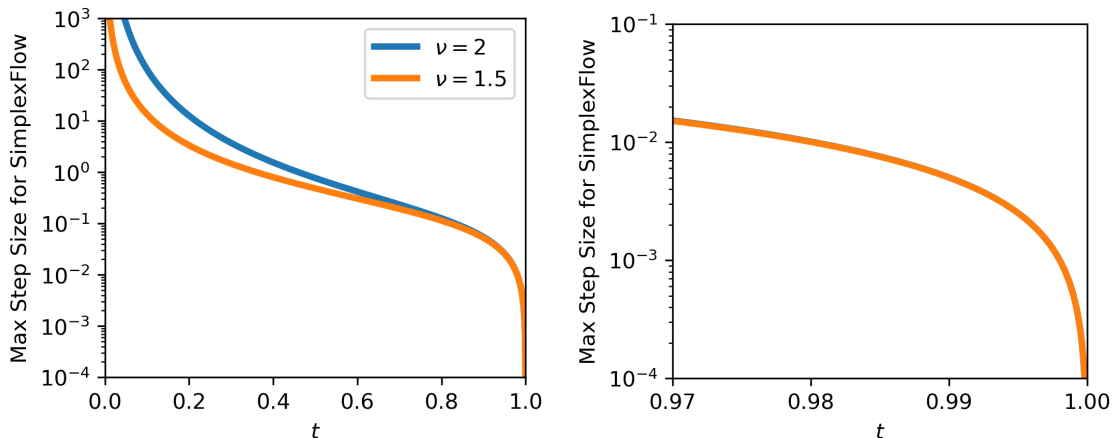


Figure 4: **Left:** maximum integration step size to remain on the simplex for a cosine interpolant. **Right:** zoomed in view of the asymptotic decline of the maximum step size as $t \rightarrow 1$

D Model Architecture

FlowMol is implemented using PyTorch and the Deep Graph Library (DGL) [59].

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 $v_i^{(0)} = 0$. 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: $e_{ij}^{(0)} = e_{ji}^{(0)}$.

Molecule Update Block We define a Molecule Update Block which 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 D.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.

$$m_{i \rightarrow j}^{(s)}, m_{i \rightarrow j}^{(v)} = \psi_M \left(\left[s_i^{(l)} : e_{ij}^{(l)} : d_{ij}^{(l)} \right], \left[v_i : \frac{x_i^{(l)} - x_j^{(l)}}{d_{ij}^{(l)}} \right] \right) \quad (25)$$

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 [53]:

$$s_i^{(l+1)}, v_i^{(l+1)} = \text{LayerNorm} \left(\left[s_i^{(l)}, v_i^{(l)} \right] + \psi_U \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \left[m_{j \rightarrow i}^{(s)}, m_{j \rightarrow i}^{(v)} \right] \right) \right) \quad (26)$$

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:

$$x_i^{(l+1)} = x_i^{(l)} + \psi_x \left(s_i^{(l+1)}, v_i^{(l+1)} \right) \quad (27)$$

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.

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

$$e_{ij}^{(l+1)} = \text{LayerNorm} \left(e_{ij}^{(l)} + \phi_e \left(s_i^{(l+1)}, s_j^{(l+1)}, d_{ij}^{(l+1)} \right) \right) \quad (28)$$

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.

D.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 GVP and a conventional feed-forward neural network is that GVPs operate on two distinct data types: scalars and vectors. GVP also allows 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. [53]; 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 operations for our cross product enhanced GVP are described in Algorithm 1.

Algorithm 1 Geometric Vector Perceptron with Cross Product

Input: Scalar and vector features: $(s, v) \in \mathbb{R}^f \times \mathbb{R}^{\nu \times 3}$
Output: Scalar and vector features: $(s', v') \in \mathbb{R}^j \times \mathbb{R}^{\mu \times 3}$
Hyperparameter: Number of hidden vector features $n_h \in \mathbb{Z}^+$
Hyperparameter: Number of cross product features $n_{cp} \in \mathbb{Z}^+$

```

 $v_h \leftarrow W_h v \in \mathbb{R}^{n_h \times 3}$ 
 $v_{cp} \leftarrow W_{cp} v \in \mathbb{R}^{2n_{cp} \times 3}$ 
 $v_{cp} \leftarrow v_{cp}[:, n_{cp}] \times v_{cp}[n_{cp} :] \in \mathbb{R}^{n_{cp} \times 3}$  // cross product
 $v_{h+cp} \leftarrow \text{Concat}(v_h, v_{cp}) \in \mathbb{R}^{(n_h+n_{cp}) \times 3}$  // concatenation along rows
 $v_\mu \leftarrow W_\mu v_{h+cp} \in \mathbb{R}^{\mu \times 3}$ 
 $s_{h+cp} \leftarrow \|v_{h+cp}\| \in \mathbb{R}^{n_h+n_{cp}}$ 
 $s_{f+h+cp} \leftarrow \text{Concat}(s, s_{h+cp})$ 
 $s_j \leftarrow W_j s_{f+h+cp} + b_j \in \mathbb{R}^j$ 
 $s' \leftarrow \sigma(s_j) \in \mathbb{R}^j$ 
 $v' \leftarrow \sigma_g(W_g[\sigma^+(s_m)] + b_g) \odot v_\mu \text{ (row-wise)} \in \mathbb{R}^{\mu \times 3}$ 
return  $(s', v')$ 

```

E Model Ablations on GEOM Dataset

Table 3: FlowMol ablations on GEOM-Drugs with explicit hydrogens

Flow Type	Categorical Prior	Atoms Stable (%) ($\uparrow$)	Mols Stable (%) ($\uparrow$)	Mols Valid (%) ($\uparrow$)	JS(E) ($\downarrow$)
Dirichlet endpoint	uniform-simplex	94.6 $\pm$ 0.2	18.7 $\pm$ 0.2	14.0 $\pm$ 1.4	0.46 $\pm$ 0.01
	marginal-simplex	97.7 $\pm$ 0.0	36.3 $\pm$ 0.1	28.3 $\pm$ 0.3	0.36 $\pm$ 0.01
	barycenter	97.6 $\pm$ 0.0	35.2 $\pm$ 0.4	27.6 $\pm$ 0.5	0.30 $\pm$ 0.01
	Gaussian	98.9 $\pm$ 0.0	66.3 $\pm$ 1.2	49.9 $\pm$ 1.0	0.34 $\pm$ 0.01

F Training Details and Hyperparameter Choices

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 64. GEOM models are trained on 4xL40 GPUs with a per-GPU batch size of 16. QM9 models train in about 3-4 days while GEOM models take 4-5 days. All model hyperparameters are visible in the config files provided in our github repository.