Generative Molecular Morphing for Flexible-Size Design via Unbalanced Optimal Transport

Malte Franke[1, 2*] **Stefan P. Schmid**[1, 2] **Žarko Ivković**[1, 2]

Kjell Jorner[1,2] **Andreas Krause**[1,2]

[1]ETH Zürich [2]NCCR Catalysis

Abstract

The success of generative molecular design hinges on a model's *steerability* toward high-reward samples. Because many molecular properties are intrinsically linked to molecular size, accurately capturing the joint distribution of properties and the number of atoms is essential. However, current diffusion and flow-based models fix the number of atoms, which ultimately limits their ability to navigate this complex relationship. To address this, we introduce **Morph**, a flexible-size generative model for conditional and unconditional 3D molecular design based on geometric graphs. By dynamically adapting size, **Morph** can seamlessly integrate existing structural priors, like scaffolds, and significantly enhances property steering. We show that **Morph** matches current fixed-size state-of-the-art models while offering the benefit of unparalleled sampling flexibility. We demonstrate out-of-distribution generation in regimes where previous models fail, paving the way for enhanced generative modeling for molecular design.

1 Introduction

Generative models can significantly accelerate scientific discovery by guiding chemical space exploration, ultimately reducing the number of expensive physical experiments Du et al. [2024]. However, the practical success of these models depends on i) their *expressivity* — whether they can model molecular systems and their interactions, and ii) on their *steerability* — how reliably they can be guided toward high-reward regions of molecular structure distributions. These properties are essential across different modes of discovery, whether the objective is to generate entirely new molecules in de novo design, or to explore the surrounding chemical neighborhood of an initial structure to optimize for specific properties.

Current diffusion and flow-based 3D generative models Vignac et al. [2023], Le et al. [2023], Dunn and Koes [2026], Irwin et al. [2025] face a significant algorithmic limitation: they cannot adapt the number of atoms during the generation process. Since many chemical properties depend on molecular size, this constraint artificially restricts the model's sample efficiency and limits both expressivity and steerability. To enable successful discoveries of molecules, an ideal model should learn the joint distribution of size and target properties. To this end, the model should be capable of dynamically adjusting the size (*i.e.*, the number of atoms) during generation.

In this work, we introduce **Morph**[2], a flexible-size generative model for geometric graphs, building on recent progress in flexible-size sequence generation, specifically Edit Flows Havasi et al. [2025],

*Corresponding author: malte.franke@inf.ethz.ch

[2]Code: https://github.com/maltefranke/morph-chem.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

and learning of general Markov chains via Generator Matching Holderrieth et al. [2025]. Our discrete-continuous flow-based model can dynamically change the number of nodes in a graph by performing multiple insertion and deletion operations, while substituting node and edge types and continuously moving atom positions to transform any prior geometric graph into a valid 3D molecular structure. Our main contributions are:

1. The extension of flexible-size sequence generation to 3D geometric graphs, enabling the seamless integration of structural priors.
2. A matching algorithm based on unbalanced optimal transport that aligns geometric graphs of different sizes to define the target probability paths for training.
3. Empirical demonstration that **Morph** matches state-of-the-art fixed-size models in unconditional generation, while unlocking enhanced property steerability and successful out-of-distribution (OOD) generation where previous models fail.

2 Methods

2.1 Notation

Let $\mathcal{A} = \{1, \ldots, Z_{\text{max}}\}$ be the set of atom types and $\mathcal{B} = \{0, \ldots, b_{\text{max}}\}$ be the set of bond types. We initially represent a molecule with n atoms as a geometric graph $\tilde{g} = (n, \mathcal{V}, \mathcal{E})$. The indexed tuple $\mathcal{V} = (p_i)_{i=1}^n$ contains nodes $p_i = (a_i, x_i)$ with $a_i \in \mathcal{A}$ and positions $x_i \in \mathbb{R}^3$. The edges are represented by an adjacency matrix $\mathcal{E} = [e_{ij}]_{i,j=1}^n \in \mathcal{S}_n(\mathcal{B})$, where $\mathcal{S}_n(\mathcal{B})$ denotes the space of symmetric $n \times n$ bond-type matrices. To enforce permutation and spatial symmetries, we quotient the state space by the group $SE(3) \times S_n$, where $SE(3)$ is the Special Euclidean group acting on x_i, and S_n is the symmetric group acting on the node indices. Because graphs with varying node counts occupy different dimensional spaces (in practice bounded by a finite maximum node count n_{max}), the overall space $\mathcal{G}$ is constructed as the disjoint union of these quotient spaces:

$$g \in \mathcal{G} = \biguplus_{n=1}^{n_{\text{max}}} \left(\mathcal{A}^n \times \mathbb{R}^{n\times 3} \times \mathcal{S}_n(\mathcal{B})\right) \;/\; (SE(3) \times S_n) \tag{1}$$

Having established the state space of geometric graphs $\mathcal{G}$[3], our objective is to construct a generative process that transforms any prior graph g_0 into a valid molecular structure g_1 with a potentially different number of nodes. We can appropriately formulate these transitions as jump-flow processes which mix continuous dynamics with discrete jumps Applebaum [2009].

2.2 Generative Dynamics: Flow Matching and CTMC

With fixed graph topology and dimension n_t, the continuous positions $x_t \in \mathbb{R}^{n_t \times 3}$ evolve under the probability flow ODE $\frac{dx_t}{dt} = v_t(g_t)$, governed by a velocity field $v_t : \mathcal{G} \to \mathbb{R}^{n_t \times 3}$, whose marginal density satisfies the continuity equation $\partial_t p_t(x) = -\nabla_x \cdot (v_t(g_t) p_t(x))$. The infinitesimal generator of these dynamics acts on test functions $f : \mathcal{G} \to \mathbb{R}$ by directional differentiation:

$$[\mathcal{L}_t^{\text{flow}} f](g) \;=\; v_t(g) \cdot \nabla_x f(g). \tag{2}$$

Concurrently, the graph can undergo discrete jumps via a Continuous-Time Markov Chain (CTMC) Ethier and Kurtz [2009]. Because these jumps — such as inserting a new atom — require proposing both discrete components (atom/bond types) and continuous 3D coordinates, we characterize transitions on $\mathcal{G}$ using a rate density $u_t(g' \mid g_t)$ defined with respect to a reference measure $\nu(dg')$. For a small time step h, the transition kernel from the current state g_t to a measurable set $A \subseteq \mathcal{G}$ is:

$$\mathbb{P}(g_{t+h} \in A \mid g_t) \;=\; \delta_{g_t}(A) \;+\; h \int_A u_t(g' \mid g_t)\, \nu(dg') \;+\; o(h), \tag{3}$$

where $\delta_{g_t}(A)$ denotes the Dirac measure centered at g_t, ensuring the state remains unchanged unless a jump occurs into A. The generator of this pure-jump process defined by u_t acts on test functions as:

$$[\mathcal{L}_t^{\text{jump}} f](g) \;=\; \int_{\mathcal{G}} \big(f(g') - f(g)\big)\, u_t(g' \mid g)\, \nu(dg'). \tag{4}$$

[3] Point clouds are trivially subsumed under this formulation by setting $b_{\text{max}} = 0$.

Conceptually, the integral over the trans-dimensional space $\mathcal{G}$ with respect to $\nu(dg')$ measures the total rate of transitioning from the current graph g to all possible modified graphs g'. Since flow and jumps act on g_t independently, the generator of the coupled process is their sum:

$$[\mathcal{L}_t f](g) \;=\; \underbrace{v_t(g)\cdot\nabla_x f(g)}_{\text{flow}} \;+\; \underbrace{\int_{\mathcal{G}} \big(f(g')-f(g)\big)\, u_t(g' \mid g)\, \nu(dg')}_{\text{jump}}. \tag{5}$$

The combined generator $\mathcal{L}_t$ governs the trans-dimensional process via the Kolmogorov forward equation $\partial_t p_t = \mathcal{L}_t^* p_t$. Thus, learning the dynamics requires fitting the drift v_t and jump rates u_t. However, these targets rely on the intractable marginal $p_t(g)$ and are therefore inaccessible directly from data.

2.3 Training via Conditional Generator Matching

To bypass this intractability, we utilize the Conditional Generator Matching (CGM) framework Holderrieth et al. [2025], which generalizes conditional flow matching to arbitrary Markov processes. CGM exploits the fact that the marginal generator $\mathcal{L}_t$ defined in Eq. 5 can be recovered as a conditional expectation of *conditional* generators defined along paths between paired endpoints $(g_0, g_1) \sim \pi$, where π is a coupling of p_{prior} and p_{data}:

$$[\mathcal{L}_t f](g) \;=\; \mathbb{E}_{(g_0,g_1)\sim\pi(\cdot,\cdot|g_t=g)}\big[\,[\mathcal{L}_t^{g_0,g_1} f](g)\,\big]. \tag{6}$$

Because $\mathcal{L}_t$ is linear in (v_t, u_t), Eq. 6 implies that a network (v_t^θ, u_t^θ) trained to regress the conditional velocity and rates under any Bregman divergence recovers the correct marginal generator at the optimum Holderrieth et al. [2025]. Learning $\mathcal{L}_t$ therefore reduces to a supervised problem over conditional paths that we construct in the following.

2.4 Unbalanced Optimal Transport for Geometric Graphs

Since g_0 and g_1 generally have different numbers of nodes (*i.e.*, $n_0 \neq n_1$), before any interpolation can be defined, we must establish a structural alignment. Such an assignment should specify for each source node whether it is *matched* to a target node (and thus survives, possibly with substituted type), or *deleted*; and which target nodes are *inserted*. However, finding an exact optimal transport plan between two fully connected graphs is expensive Riesen and Bunke [2009]. We bypass this combinatorial bottleneck by setting the cost of all edge edits to zero, which is justifiable for molecular geometries, where bond topology can be inferred given the 3D structure and atom valency. The resulting matching of discrete-continuous point clouds can be formulated with Unbalanced Optimal Transport (UOT) Séjourné et al. [2023].

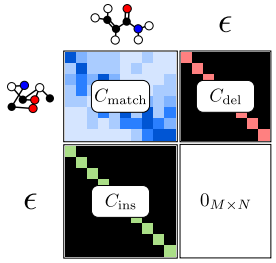
The UOT framework relaxes the strict mass-conservation constraints of OT, allowing us to match distributions of different masses by penalizing marginal deviations. In our case, the UOT objective seeks a transport plan $\mathbf{P} \in \mathbb{R}_{\geq 0}^{n_0 \times n_1}$ that minimizes

$$\min_{\mathbf{P}\in\mathbb{R}_{\geq 0}^{n_0\times n_1}} \left(\sum_{i=1}^{n_0}\sum_{j=1}^{n_1} \tilde{C}[i,j]P[i,j] + w_{\text{del}}||\mathbb{1}_{n_0} - \mathbf{P}\mathbb{1}_{n_1}||_1 + w_{\text{ins}}||\mathbb{1}_{n_1} - \mathbf{P}^T\mathbb{1}_{n_0}||_1 \right) \tag{7}$$

where $\tilde{C}[i,j]$ are the entries of the cost matrix $\tilde{\mathbf{C}} \in \mathbb{R}^{n_0\times n_1}$, and w_{del} and w_{ins} control the cost of mass deletion and insertion. To obtain a set of exact graph edits, we restrict the transport plan $\mathbf{P}$ to integer assignments. This discrete UOT problem is theoretically equivalent to formulating an assignment problem in an augmented space. By introducing auxiliary variables ϵ that lift the space to $(n_0+n_1)\times(n_0+n_1)$, the objective becomes finding the binary assignment matrix $\hat{\mathbf{P}}$ that minimizes the total cost

$$\min_{\hat{\mathbf{P}}\in\{0,1\}^{(n_0+n_1)\times(n_0+n_1)}} \sum_{i=1}^{n_0+n_1}\sum_{j=1}^{n_0+n_1} \mathbf{C}[i,j]\hat{\mathbf{P}}[i,j] \tag{8}$$

subject to the strict constraints that every row and column is assigned exactly once ($\sum_j \hat{\mathbf{P}}[i,j] = 1$ and $\sum_i \hat{\mathbf{P}}[i,j] = 1$). The augmented cost matrix $\mathbf{C} \in \mathbb{R}^{(n_0+n_1)\times(n_0+n_1)}$ is structured as



where:

$$C_{\text{match}}[i,j] = w_{\text{move}}\|x_{0,i} - x_{1,j}\|_2 + w_{\text{type}}\mathbb{1}(a_{0,i} \neq a_{1,j}),$$

$$C_{\text{del}}[i,l] = \begin{cases} w_{\text{del}} & \text{if } i + n_1 = l \\ \infty & \text{otherwise} \end{cases},$$

$$C_{\text{ins}}[k,j] = \begin{cases} w_{\text{ins}} & \text{if } k = j + n_0 \\ \infty & \text{otherwise} \end{cases},$$

for source nodes $i \in \{1,\dots,n_0\}$, target nodes $j \in \{1,\dots,n_1\}$, and slack indices $k \in \{n_0+1,\dots,n_0+n_1\}$ and $l \in \{n_1+1,\dots,n_1+n_0\}$. The optimization problem in Eq. 8 can be efficiently solved using the Hungarian algorithm for small- to moderate sized-graphs Kuhn [1955], Munkres [1957]. However, such matching does not yet take into account the translation and rotation symmetry of the SE(3) group. Therefore, we rotationally align matched positions by applying the Kabsch algorithm Kabsch [1976] , leading to a locally optimal transport plan Klein et al. [2023]. We use the resulting coupling to define interpolation paths between g_0 and g_1.

2.5 Graph Interpolation

We can define a conditional probability path evolving the three modalities independently as

$$p_t(g \mid g_0, g_1) \;=\; p_t(x \mid x_0, x_1) \cdot p_t(a \mid a_0, a_1) \cdot p_t(e \mid e_0, e_1). \tag{9}$$

Under this decomposition, the conditional generator splits into a continuous velocity for surviving node coordinates and discrete rates governing four atomic operations that span all transitions in $\mathcal{G}$[4]: node insertion and deletion, atom-type substitution, and edge-type substitution. Therefore, the optimal assignment matrix $\hat{\mathbf{P}}^*$ which aligns valid node sets $\mathcal{S}_0 = \{1,\dots,n_0\}$ from g_0 and $\mathcal{S}_1 = \{1,\dots,n_1\}$ from g_1, partitions the non-zero entries into three mutually exclusive edit sets. Each set dictates a distinct interpolation along the time variable $t \in [0,1]$. Given a scheduler κ_t (with boundary conditions $\kappa_0 = 0$ and $\kappa_1 = 1$) Gat et al. [2024], Campbell et al. [2024] we define:

Matched nodes $\mathcal{M} = \{(i,j) \mid \hat{\mathbf{P}}^*[i,j] = 1,\ i \in \mathcal{S}_0, j \in \mathcal{S}_1\}$ are present throughout the entire interpolation. Their continuous positions follow the standard Gaussian path Lipman et al. [2022], $x_t \sim \mathcal{N}((1-t)x_0 + tx_1, \sigma^2 I)$, while atom and edge type interpolation follows the scheduler κ_t^{sub}, yielding $a_t \sim \text{Cat}\left((1-\kappa_t^{\text{sub}})\frac{1}{|\mathcal{A}|} + \kappa_t^{\text{sub}}\delta_{a_1}\right)$ and $e_t \sim \text{Cat}\left((1-\kappa_t^{\text{sub}})\frac{1}{|\mathcal{B}|} + \kappa_t^{\text{sub}}\delta_{e_1}\right)$.

Insertion nodes $\mathcal{I} = \{(\epsilon,j) \mid \hat{\mathbf{P}}^*[k,j] = 1,\ k \notin \mathcal{S}_0, j \in \mathcal{S}_1\}$ require sampling an independent event time $t_{\text{ins}} \sim \kappa^{\text{ins}}$ for each node. For $t \leq t_{\text{ins}}$, the node does not yet exist. If inserted ($t > t_{\text{ins}}$), its position is sampled from a Gaussian centered around the target position $x_t \sim \mathcal{N}(x_1, \sigma_t^2 I)$ with a shrinking variance $\sigma_t = \sigma(1-t) + \varepsilon$. The discrete features follow the insertion scheduler κ_t^{ins}, with $a_t \sim \text{Cat}\left((1-\kappa_t^{\text{ins}})\frac{1}{|\mathcal{A}|} + \kappa_t^{\text{ins}}\delta_{a_1}\right)$ and $e_t \sim \text{Cat}\left((1-\kappa_t^{\text{ins}})\frac{1}{|\mathcal{B}|} + \kappa_t^{\text{ins}}\delta_{e_1}\right)$.

Deletion nodes $\mathcal{D} = \{(i,\epsilon) \mid \hat{\mathbf{P}}^*[i,l] = 1,\ i \in \mathcal{S}_0, l \notin \mathcal{S}_1\}$ are assigned an independent, node-wise deletion event time $t_{\text{del}} \sim \kappa^{\text{del}}$. For $t < t_{\text{del}}$, we fix the node and its edges at their prior attributes, i.e., position $x_t = x_0$, atom type $a_t = a_0$, and edge types $e_t = e_0$. Else, if $t \geq t_{\text{del}}$, the node and all its connected edges are removed.

The algorithm is shown in Algorithm 2. Using the edit sets, the state transitions are trained via their conditional generator target using a Bregman divergence, yielding an overall objective of the form

$$\mathcal{J}(\theta) \;=\; \mathbb{E}_{t,\,(g_0,g_1)\sim\pi,\,g_t\sim p_t(\cdot|g_0,g_1)}\big[\,\textstyle\sum_k \mathcal{J}_k(\theta)\,\big]\,, \tag{10}$$

whose terms we describe in the following.

[4]Edge insertions and deletions are absorbed into edge substitution via the null bond type.

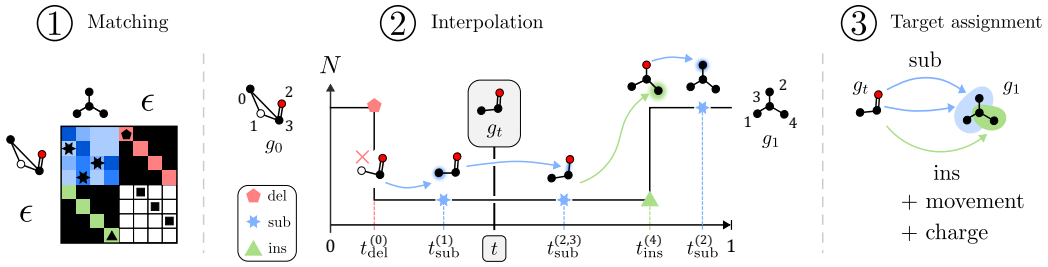


Figure 1: Interpolation between a randomly sampled graph ($t = 0$) and a valid molecule ($t = 1$). In the first step, we use the UOT matching to obtain edit sets determining whether nodes must be substituted, inserted or deleted. Given these sets, we interpolate to an intermediate graph g_t. Lastly, we assign the conditional generator matching targets.

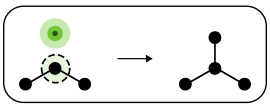
2.6 State Transitions

All discrete structural modifications to the graph must be fully resolved by the end of the generative process at $t = 1$. To achieve this, we model the transition rates of our CTMC using bounded hazard distributions. Similar to Nguyen et al. [2025], we choose to factorize the general transition rate λ for any discrete action as

$$\lambda = \Delta \cdot \frac{\dot{\kappa}_t}{1 - \kappa_t} \geq 0. \tag{11}$$

In this formulation, the explicit hazard rate $\frac{\dot{\kappa}_t}{1-\kappa_t}$ determines *when* the action happens. By construction, the schedule $\kappa_t \to 1$ as $t \nearrow 1$, causing the denominator $(1 - \kappa_t)$ to vanish. Consequently, the transition rate $\lambda \to \infty$ forces the Markov chain to resolve any remaining actions and ensures the termination of the generative process by $t = 1$. The variable Δ determines *how much* of an action happens (or if it happens at all) and is dynamically extracted from our UOT matching at time t. Since we impose the scheduler κ_t, the model only has to learn the target Δ. We now define the full set of objectives for nodes currently present in g_t.

Insertions For each insertion node ($j \in \mathcal{I}$) that has not yet been inserted ($t \leq t^{(j)}_{\text{ins}}$), we locate its nearest spatial neighbor to define $\mathcal{Q}_i = \left\{ q_j \mid j \in \mathcal{I},\ t \leq t^{(j)}_{\text{ins}},\ i = \arg\min_{k \in g_t} \|x_k - x^{\mathcal{I}}_j\|_2 \right\}$, the set of pending insertion nodes assigned to the existing node i at time t. The target insertion count is exactly the cardinality of this set, $\Delta^{\text{ins}}_i = |\mathcal{Q}_i|$. We then define the insertion action as



$$\text{ins}(g_t, p_i, q, \mathbf{e}_q) = \left(n_t + 1,\ \mathcal{V}_t \cup \{q\},\ \begin{pmatrix} \mathcal{E}_t & \mathbf{e}_q \\ \mathbf{e}_q^\top & 0 \end{pmatrix} \right) \tag{12}$$

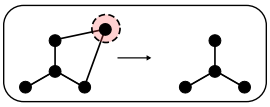
$$u^\theta_t(\text{ins}(g_t, p_i, q, \mathbf{e}_q) \mid g_t) = \lambda^{\text{ins}}_{t,i}(g_t) Q^{\text{ins}}_{t,i}(q \mid g_t) Q^{\text{ins}}_{t,i}(\mathbf{e}_q \mid q, g_t) \tag{13}$$

where $q = (a, x)$ is the new node and $\mathbf{e}_q$ are its edges to all existing nodes in g_t and to other inserted nodes. Importantly, the prediction of edges for inserted nodes depends on their sampled positions and types. We parameterize $Q^{\text{ins}}_{t,i}(q \mid g_t)$ with a unimodal head for positions and categorical atom types (Appendix A.3).

While in theory our CTMC formulation models single-node insertions in an infinitesimal time interval, in practice we batch the prediction of all pending nodes $\mathcal{Q} = \bigcup_{k=1}^{n_t} \mathcal{Q}_k$ during training. Therefore, we define an augmented edge target $\mathbf{e}^+_q$, which encompasses the edges connecting the newly inserted node q not only to the existing graph g_t, but also to all other concurrently inserted nodes in $\mathcal{Q}$. Finally, we can write the combined loss for insertions by summing over these assigned nodes:

$$\mathcal{J}_{\text{ins}}(\theta) = \frac{1}{n_t} \sum_{i=1}^{n_t} \left[\mathcal{J}_{\text{PoissonNLL}}(\Delta^{\theta,\text{ins}}_i, \Delta^{\text{ins}}_i) - \sum_{q \in \mathcal{Q}_i} \log Q^{\theta,\text{ins}}_{t,i}(q \mid g_t) + \sum_{q \in \mathcal{Q}_i} \mathcal{J}_{\text{CE}}(\hat{\mathbf{e}}^+_q, \mathbf{e}^+_q) \right]. \tag{14}$$

Deletions To reduce the number of nodes in a graph, we can define the deletion of node i as



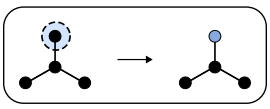
$$\mathrm{del}(g_t, p_i) = (n_t - 1,\ \mathcal{V}_t \setminus \{p_i\},\ \mathcal{E}_{t,-i}) \tag{15}$$

$$u_t^\theta(\mathrm{del}(g_t, p_i) \mid g_t) = \lambda_{t,i}^{\mathrm{del}}(g_t) \tag{16}$$

where i and all its edges are removed, leading to a reduced cardinality $n_t - 1$. For a node destined for deletion ($i \in \mathcal{D}$), its target is $\Delta_i^{\mathrm{del}} = 1$ if it has not yet reached its deletion time ($t < t_{\mathrm{del}}^{(i)}$). All other nodes have $\Delta_i^{\mathrm{del}} = 0$. Since a node can either be deleted or not, we formulate the deletion loss as a binary cross-entropy over all present nodes

$$\mathcal{J}_{\mathrm{del}}(\theta) = \frac{1}{n_t} \sum_{i=1}^{n_t} \mathcal{J}_{\mathrm{BCE}}(\Delta_i^{\theta,\mathrm{del}}, \Delta_i^{\mathrm{del}}). \tag{17}$$

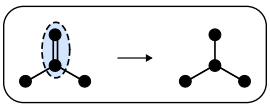
Substitutions Next, we define the substitution of node and edge types of the graph. For existing nodes, we set $\Delta_i^{\mathrm{sub,a}} = 1$ if the atom type of a node needs to be substituted (*i.e.*, $a_t^{(i)} \neq a_1^{(i)}$), and 0 otherwise. An analogous binary target $\Delta_{ij}^{\mathrm{sub,e}} \in \{0, 1\}$ is extracted for edges. We then denote the substitution of node i as



$$\mathrm{sub_a}(g_t, p_i, p_i') = (n_t,\ (\mathcal{V}_t \setminus \{p_i\}) \cup \{p_i'\},\ \mathcal{E}_t) \tag{18}$$

$$u_t^\theta(\mathrm{sub_a}(g_t, p_i, p_i') \mid g_t) = \lambda_{t,i}^{\mathrm{sub,a}}(g_t) Q_{t,i}^{\mathrm{sub,a}}(p_i' \mid g_t). \tag{19}$$

with $p_i' = (x_i, a_i')$. Correspondingly, the substitution of the bidirectional edge between i and j can be formulated as



$$\mathrm{sub_e}(g_t, e_{ij}, b) = (n_t,\ \mathcal{V}_t,\ \mathcal{E}') \tag{20}$$

$$u_t^\theta(\mathrm{sub_e}(g_t, e_{ij}, b) \mid g_t) = \lambda_{t,ij}^{\mathrm{sub,e}}(g_t) Q_{t,ij}^{\mathrm{sub,e}}(b \mid g_t) \tag{21}$$

where $\mathcal{E}_{ij}' = \mathcal{E}_{ji}' = b$ and $\mathcal{E}_{kl}' = (\mathcal{E}_t)_{kl}$ otherwise. The substitution loss is then

$$\mathcal{J}_{\mathrm{sub}}(\theta) = \frac{1}{n_t} \sum_{i=1}^{n_t} \Bigg[\underbrace{\mathcal{J}_{\mathrm{BCE}}(\Delta_i^{\theta,\mathrm{sub,a}}, \Delta_i^{\mathrm{sub,a}}) + \Delta_i^{\mathrm{sub,a}} \mathcal{J}_{\mathrm{CE}}(\hat{a}_{1,i}, a_{1,i})}_{\mathcal{J}_{\mathrm{sub,a}}} + \frac{1}{n_t} \sum_{j=1}^{n_t} \underbrace{\mathcal{J}_{\mathrm{BCE}}(\Delta_{ij}^{\theta,\mathrm{sub,e}}, \Delta_{ij}^{\mathrm{sub,e}}) + \Delta_{ij}^{\mathrm{sub,e}} \mathcal{J}_{\mathrm{CE}}(\hat{e}_{1,ij}, e_{1,ij})}_{\mathcal{J}_{\mathrm{sub,e}}} \Bigg] \tag{22}$$

Movement Finally, the nodes can change their positions. Using endpoint targets, we define the loss only on positions of persisting nodes as

$$\mathcal{J}_{\mathrm{move}}(\theta) = \frac{1}{n_t} \sum_{i=1}^{n_t} (1 - \Delta_i^{\mathrm{del}}) \mathcal{J}_{\mathrm{MSE}}(\hat{x}_{1,i}, x_{1,i}). \tag{23}$$

Although charge is not explicitly modeled as part of g_t, we treat its prediction as an auxiliary task. For newly inserted nodes, charge is modeled jointly with position and atom type by the insertion head, $Q_{t,i}^{\theta,\mathrm{ins}}$. For existing nodes, we include a separate cross-entropy loss $\mathcal{J}_{\mathrm{charge}}(\theta)$ evaluated at the final frame. Therefore, the final training objective is formulated as:

$$\mathcal{J}(\theta) = \mathbb{E}_{\substack{t\sim\mathcal{U}([0,1]),\\ (g_0,g_1)\sim\pi,\\ g_t\sim q(\cdot|g_0,g_1)}} \left[\mathcal{J}_{\mathrm{move}}(\theta) + \mathcal{J}_{\mathrm{sub,e}}(\theta) + \mathcal{J}_{\mathrm{sub,a}}(\theta) + \mathcal{J}_{\mathrm{del}}(\theta) + \mathcal{J}_{\mathrm{ins}}(\theta) + \mathcal{J}_{\mathrm{charge}}(\theta)\right] \tag{24}$$

Given a trained model, we numerically integrate the learned jump-flow process to transform a random g_0 into a molecule (see Alg. 3). For discrete updates, we exploit the exact integrability of the schedule hazards to compute finite-step transition probabilities, with network predictions held fixed within each step. We elaborate on additional model details in A.3.

3 Results

3.1 De novo design

We show results for 3D molecular design on GEOM-Drugs Axelrod and Gomez-Bombarelli [2022], targeting the unconditional design of small organic molecules. We evaluate models based on established metrics such as validity and stability (see A.5), energy, and PoseBusters Buttenschoen et al. [2024], and compare against current state-of-the-art models for fixed-size generation. **Morph** achieves 95.46% validity at 100 function evaluations (NFE), compared with 93.9% for SemlaFlow at the same NFE. Here, validity requires both connectivity and RDKit sanitization over all generation attempts. Joint success on all 12 PoseBusters checks is 88.30% on GEOM-Drugs and 93.53% on QM9, counting all attempts (Table 6). Additional ChEMBL3D results are reported in Appendix B.2.

Table 1: GEOM-Drugs de novo generation using SemlaFlow metrics (3 seeds; 10k molecules/seed). †: adapted from SemlaFlow Irwin et al. [2025]; *: reproduced from the FlowMol3 repository. The **Morph** row uses the explicit-H checkpoint after 100 completed epochs; stability uses the valence table and validity is connected sanitization.

	Model	Atom Stab ↑	Mol Stab ↑	Valid ↑	Unique ↑	Novel ↑	NFE
n_{atoms} fixed	MiDi†	99.8	91.6	77.8	**100.0**	**100.0**	500
	EQGAT-diff†	$99.8_{\pm 0.0}$	$93.4_{\pm 0.21}$	$94.6_{\pm 0.24}$	$\mathbf{100.0}_{\pm 0.0}$	$99.9_{\pm 0.07}$	500
	SemlaFlow†	$99.8_{\pm 0.0}$	$97.3_{\pm 0.08}$	$93.9_{\pm 0.19}$	$\mathbf{100.0}_{\pm 0.0}$	$99.6_{\pm 0.03}$	100
	FlowMol3*	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{99.95}_{\pm 0.02}$	$\mathbf{99.35}_{\pm 0.06}$	$99.54_{\pm 0.02}$	–	250
	Morph (Ours)	$99.98_{\pm 0.00}$	$95.92_{\pm 0.33}$	$95.46_{\pm 0.31}$	$\mathbf{100.00}_{\pm 0.00}$	$99.75_{\pm 0.07}$	100

Table 2: GEOM-Drugs molecular properties (mean$_{\pm \text{SD}}$ across molecules). Training: 248,147 molecules; **Morph**: 29,008 evaluated molecules pooled across three runs; baselines: 10k/model. Bold marks the mean closest to training.

Model	# heavy atoms	# rot. bonds	# HBD	# HBA	TPSA (Å^2)	# rings	# stereocentres
GEOM $_{\text{train}}$	$24.64_{\pm 5.28}$	$4.77_{\pm 2.19}$	$1.18_{\pm 0.89}$	$4.47_{\pm 1.67}$	$73.42_{\pm 27.45}$	$2.99_{\pm 0.99}$	$0.42_{\pm 0.92}$
SemlaFlow	$22.88_{\pm 5.03}$	$\mathbf{4.67}_{\pm 2.55}$	$\mathbf{1.29}_{\pm 1.01}$	$3.93_{\pm 1.58}$	$63.17_{\pm 26.92}$	$2.75_{\pm 1.05}$	$0.96_{\pm 1.36}$
FlowMol3	$22.09_{\pm 4.88}$	$4.89_{\pm 1.93}$	$0.93_{\pm 0.76}$	$3.78_{\pm 1.55}$	$60.43_{\pm 23.48}$	$2.39_{\pm 0.94}$	$\mathbf{0.36}_{\pm 0.75}$
Morph (Ours)	$\mathbf{23.64}_{\pm 5.21}$	$4.93_{\pm 2.33}$	$1.40_{\pm 1.05}$	$\mathbf{4.11}_{\pm 1.67}$	$\mathbf{68.67}_{\pm 29.39}$	$\mathbf{2.85}_{\pm 1.10}$	$0.98_{\pm 1.34}$

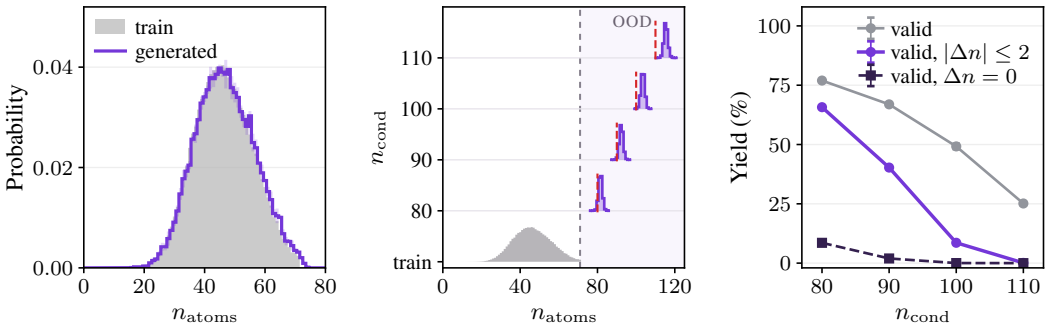
Furthermore, **Morph** most closely matches the training means for four of seven medicinally relevant descriptors: heavy-atom count, hydrogen-bond acceptors, rings, and topological polar surface area (Table 2). SemlaFlow is closest for rotatable bonds and hydrogen-bond donors, and FlowMol3 for stereocentres.

Together with the stability and validity results above, the distributional metrics highlight competitive sample quality in addition to **Morph**'s more expressive generation trajectories. Next, we demonstrate the steerability of our model on conditional design tasks.

Property steering To show the steerability of **Morph** in an illustrative setting, we generate structures of a desired size given an uninformative prior on the number of atoms $n_0 \sim \mathcal{U}(n_{\text{min}}, n_{\text{max}})$. We train a model conditioned on the target atom count on GEOM-Drugs Axelrod and Gomez-Bombarelli [2022], with training molecules capped at 72 atoms including hydrogens. After validating in-distribution steerability with empirical atom-count targets (Fig. 2a), we request $n_{\text{cond}} \in \{80, 90, 100, 110\}$, beyond the training cutoff.

Figure 2 shows that **Morph** recovers the training size distribution with 93.84% validity for empirical targets and can be steered to generate larger molecules under OOD conditioning. At $n_{\text{cond}} = 80$, validity is 76.96% and the atom-count MAE is 1.59. As targets increase, generated sizes increasingly overshoot them: at $n_{\text{cond}} = 110$, validity falls to 25.14% and MAE rises to 5.38 atoms. The joint

yield of valid molecules within two atoms of the target, and especially at the exact target, therefore decreases (Fig. 2c). Full results are in Table 11.



(a) Empirical size targets (b) OOD size targets (c) Valid molecule yield

Figure 2: Size conditioning on GEOM-Drugs (training cutoff: 72 atoms). Yields use all generation attempts; validity requires a connected, sanitizable molecule, and $\Delta n = n_{\text{atoms}} - n_{\text{cond}}$. We generate 10k attempts per seed for empirical targets and 5k per fixed target; bands and error bars show standard deviations across three seeds.

Towards out-of-distribution discovery To demonstrate **Morph**'s potential for OOD discovery, we generate molecules with a desired log partition coefficient (logP), which is a measure of lipophilicity with strong correlation to size. We use RDKit Landrum et al. [2026] to calculate logP values of all QM9 molecules and finetune a property-conditioned model.

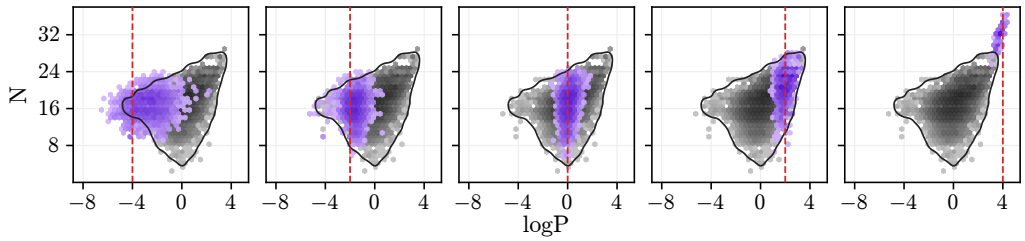


Figure 3: LogP-conditioned generation with **Morph** (10k samples). Gray contours show the training density, purple hexagons generated molecules, and dashed red lines the target logP. Despite uniform sampling of n_0, $p(n_1 \mid \text{logP})$ follows the corresponding training slice.

Figure 3 shows that our model can steer size depending on the property conditioning, automatically recovering the corresponding training data n-marginal distribution. When conditioned on values in low support or OOD regions (logP $= -4$ and 4, respectively), we observe that **Morph** can successfully generate valid molecules with sizes and logP values outside the training distribution. We validate multi-property steerability in B.9. Pooled logP evaluations on QM9 and GEOM-Drugs achieve MAEs of 0.304 and 0.538, respectively (Table 12) without externally supplying the target-dependent size distribution. Since scientific discovery requires generating candidates with properties unseen in the training data, **Morph**'s ability to generate flexibly-sized molecules paves the way for improved property-conditioned generative molecular design.

3.2 Scaffold decoration

Besides improved property steering, **Morph**'s flexible-size generation also enables the construction of molecules based on a given structural prior. Such a task is known as *scaffold decoration*, or lead optimization, and is common in drug discovery to optimize a structure while preserving the key molecular properties tied to its scaffold Fialková et al. [2022], Schneuing et al. [2024].

We demonstrate this task on GEOM-Drugs, starting from Bemis–Murcko scaffolds g_0 and extending them to complete molecules g_1 that contain the same motif (Fig. 4). To construct the training paths, we first identify the common scaffold between g_0 and g_1. Its atom and bond types remain fixed throughout the trajectory. We then apply UOT independently at each substituent site, building substituents through atom insertions, substitutions, and movement.

The test evaluation uses 512 scaffolds with eight completions each (4,096 generation attempts per seed, three seeds). **Morph** achieves $99.99 \pm 0.00\%$ atom stability, $97.90 \pm 0.16\%$ molecule stability, $97.99 \pm 0.20\%$ validity, $63.54 \pm 0.59\%$ pooled uniqueness, and $97.52 \pm 0.10\%$ novelty. All eight native PoseBusters checks pass for $94.10 \pm 0.19\%$ of all attempts (Table 9).

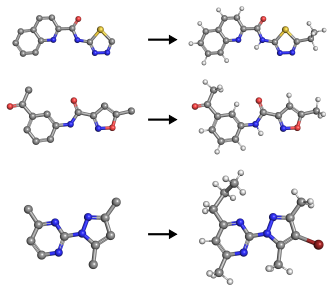


Figure 4: Scaffold priors (left) and **Morph** completions (right).

We additionally fine-tune the ChEMBL3D implicit-H **Morph** model on the in-pocket CrossDocked2020 multi-R benchmark Francoeur et al. [2020] from DiffDec Xie et al. [2024]. Given only an initial scaffold and protein pocket, **Morph** generates multiple substituents simultaneously without anchor-atom annotations or a prescribed final number of atoms. It produces 89.80% valid molecules, and 40.33% of its completions have higher predicted binding affinity than their reference molecules, yielding competitive performance against specialized in-pocket methods (Table 10).

4 Related Work

Flexible-size generative models Generative modeling of flexible-size data has been predominantly explored for discrete sequences. Autoregressive (AR) models naturally increase sequence length with strict ordering (*e.g.*, left-to-right), but generally prohibit retrospective editing. To alleviate this, several variable-length sequence models based on insertions and deletions have been developed Gu et al. [2019], Ruis et al. [2020], Johnson et al. [2021]. Recent advances in masked diffusion models enable any-order sequence modeling with token insertion and re-masking capabilities Kim et al. [2025], Patel et al. [2025]. Flexible flow-based generative models like Edit Flows Havasi et al. [2025] can model substitution, insertion, and deletion of tokens, even supporting multi-modal insertions Nguyen et al. [2025].

For graphs, AR methods sequentially add nodes and edges You et al. [2018], Liao et al. [2019], but inherit the same ordering limitations as sequence models. Diffusion models Vignac et al. [2022], Jo et al. [2022] generate graphs by evolving node and edge types simultaneously on a fixed-dimensional state space. To enable variable-size generation, they rely on a predetermined maximum size and utilize padding nodes or masking rather than actively scaling the dimensionality during generation. Only recently has this limitation been addressed by reformulating discrete graph diffusion to explicitly support either insertion or deletion of nodes Ninniri et al. [2025].

While frameworks for flexible-size generation exist in discrete domains, translating these transdimensional operations to discrete-continuous *geometric* graphs—where nodes possess continuous 3D coordinates and are subject to SE(3) symmetries—remains an open problem.

Flexible-size generative models for 3D molecular design Most diffusion and flow-based generative models for 3D molecular design decouple the determination of system size from the generation process itself. A common approach is to draw n from an empirical distribution, *i.e.*, $n_0 \sim p_{\text{data}}(n)$, and keep it fixed along the generation Hoogeboom et al. [2022], Vignac et al. [2023], Le et al. [2023], Irwin et al. [2025], Joshi et al. [2025], Zeng et al. [2026], Morehead et al. [2026], Schneuing et al. [2024], Xie et al. [2024], Vonessen et al. [2025], Reidenbach et al. [2026]. Other methods predict n_0 via a conditional task beforehand Igashov et al. [2024], Zhou et al. [2025], or make use of 'fake atoms' Dunn and Koes [2026]. Still, none of the listed methods go beyond the empirical distribution of number of atoms seen during training.

To overcome fixed-size limitations, several works have explored dynamic dimensionality during generation, though they face *directional* and hence geometric constraints. Transdimensional jump-diffusion Campbell et al. [2023] and AR models Cheng et al. [2025], Joshi et al. [2021], Daigavane et al. [2024], Rose et al. [2025] are restricted to *insertions*, *i.e.* they cannot transform

a graph g_0 into another graph g_1 if its number of nodes n_0 is larger than n_1. Lifting *directional* constraints there remains largely untackled, limiting the use of informative structural priors.

Other approaches increase flexibility by allowing both insertions and deletions. Branching Flows Billera et al. [2025] constructs a tree-based generative process via in-place duplication and coalescence of atoms using the CGM framework, but relies on the molecule's sequence representations to define training targets. On the other hand, by using different representations, fragment-based flow models Poletukhin et al. [2026] can alter the number of atoms during the trajectory by changing fragment types. Beyond graph-based representations, other paradigms bypass discrete nodes entirely by modeling molecules in continuous spaces, such as 3D voxels Pinheiro et al. [2024a,b], Faltings et al. [2025] or continuous fields Veljković et al. [2026].

5 Limitations

The jump process introduces separate objectives for insertions, deletions, and substitutions, making optimization and loss balancing more complex than in a fixed-dimensional flow. We defer systematic ablation ofthe interpolation schedules, matching procedure and -costs, or relative loss weights to future work. Although **Morph** samples quickly at low NFE, validity depends on the integration budget (Appendix B.3), resulting in a trade-off between speed and sample quality.

Furthermore, our extrapolation experiments remain simple. Atom-count steering on GEOM-Drugs shows declining validity and size accuracy beyond training support; logP steering is evaluated on QM9 and GEOM-Drugs. Reliability under arbitrary distribution shifts remains to be tested. Broader evaluations should cover chemically meaningful properties and more diverse molecular settings.

6 Conclusion

In this work, we introduced **Morph**, a flexible-size 3D generative model for geometric graphs which can simultaneously perform *multiple* insertions, deletions and substitutions. We show competitive performance to leading fixed-size generative models on de novo design tasks while matching the training distribution better and being significantly more expressive. **Morph** also enables flexible generation based on structural priors, as validated on a scaffold decoration task. By being able to generate OOD samples, **Morph** is a promising building block for accelerating discoveries in the chemical sciences and beyond.

Acknowledgments and Disclosure of Funding

This publication was created as part of NCCR Catalysis (grant numbers 180544 and 225147), a National Centre of Competence in Research funded by the Swiss National Science Foundation. This work was supported as part of the Swiss AI Initiative by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID a131 on Alps.

References

Yuanqi Du, Arian R Jamasb, Jeff Guo, Tianfan Fu, Charles Harris, Yingheng Wang, Chenru Duan, Pietro Liò, Philippe Schwaller, and Tom L Blundell. Machine learning-aided generative molecular design. *Nature Machine Intelligence*, 6(6):589–604, 2024.

Clement Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. Midi: Mixed graph and 3d denoising diffusion for molecule generation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 560–576. Springer, 2023.

Tuan Le, Julian Cremer, Frank Noe, Djork-Arné Clevert, and Kristof Schütt. Navigating the design space of equivariant diffusion-based generative models for de novo 3d molecule generation. *arXiv preprint arXiv:2309.17296*, 2023.

Ian Dunn and David R Koes. Flowmol3: flow matching for 3d de novo small-molecule generation. *Digital Discovery*, 5(5):2052–2066, 2026.

Ross Irwin, Alessandro Tibo, Jon Paul Janet, and Simon Olsson. SemlaFlow – efficient 3d molecular generation with latent attention and equivariant flow matching. *arXiv preprint arXiv:2406.07266*, 2025.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky T. Q. Chen. Edit flows: Flow matching with edit operations. *arXiv preprint arXiv:2506.09018*, 2025.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky T. Q. Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes. *arXiv preprint arXiv:2410.20587*, 2025.

David Applebaum. *Lévy processes and stochastic calculus*. Cambridge university press, 2009.

Stewart N Ethier and Thomas G Kurtz. *Markov processes: characterization and convergence*. John Wiley & Sons, 2009.

Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision computing*, 27(7):950–959, 2009.

Thibault Séjourné, Gabriel Peyré, and François-Xavier Vialard. Unbalanced optimal transport, from theory to numerics. *Handbook of Numerical Analysis*, 24:407–471, 2023.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Foundations of Crystallography*, 32(5):922–923, 1976.

Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching. *Advances in Neural Information Processing Systems*, 36:59886–59910, 2023.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *arXiv preprint arXiv:2407.15595*, 2024.

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. *arXiv preprint arXiv:2402.04997*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

John Nguyen, Marton Havasi, Tariq Berrada, Luke Zettlemoyer, and Ricky T. Q. Chen. OneFlow: Concurrent mixed-modal and interleaved generation with edit flows. *arXiv preprint arXiv:2510.03506*, 2025.

Simon Axelrod and Rafael Gomez-Bombarelli. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific data*, 9(1):185, 2022.

Martin Buttenschoen, Garrett M. Morris, and Charlotte M. Deane. PoseBusters: AI-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical Science*, 15:3130–3139, 2024. doi: 10.1039/D3SC04185A. URL `http://dx.doi.org/10.1039/D3SC04185A`.

Greg Landrum et al. Rdkit: Open-source cheminformatics, 2026.

Vendy Fialková, Jiaxi Zhao, Kostas Papadopoulos, Ola Engkvist, Esben Jannik Bjerrum, Thierry Kogej, and Atanas Patronov. Libinvent: Reaction-based generative scaffold decoration for in silico library design. *Journal of Chemical Information and Modeling*, 62(9):2046–2063, 2022. doi: 10.1021/acs.jcim.1c00469. URL `https://doi.org/10.1021/acs.jcim.1c00469`. PMID: 34460269.

Arne Schneuing, Charles Harris, Yuanqi Du, Kieran Didi, Arian Jamasb, Ilia Igashov, Weitao Du, Carla Gomes, Tom L Blundell, Pietro Lio, Max Welling, Michael Bronstein, and Bruno Correia. Structure-based drug design with equivariant diffusion models. *Nature Computational Science*, 4 (12):899–909, 2024.

Paul G Francoeur, Tomohide Masuda, Jocelyn Sunseri, Andrew Jia, Richard B Iovanisci, Ian Snyder, and David R Koes. 3d convolutional neural networks and a crossdocked dataset for structure-based drug design. *Journal of chemical information and modeling*, 60(9):4200, 2020.

Junjie Xie, Sheng Chen, Jinping Lei, and Yuedong Yang. Diffdec: Structure-aware scaffold decoration with an end-to-end diffusion model. *Journal of Chemical Information and Modeling*, 64(7):2554–2564, 2024. doi: 10.1021/acs.jcim.3c01466. URL `https://doi.org/10.1021/acs.jcim.3c01466`. PMID: 38267393.

Jiatao Gu, Changhan Wang, and Jake Zhao. Levenshtein transformer. *arXiv preprint arXiv:1905.11006*, 2019.

Laura Ruis, Mitchell Stern, Julia Proskurnia, and William Chan. Insertion-deletion transformer. *arXiv preprint arXiv:2001.05540*, 2020.

Daniel D. Johnson, Jacob Austin, Rianne van den Berg, and Daniel Tarlow. Beyond in-place corruption: Insertion and deletion in denoising probabilistic models. *arXiv preprint arXiv:2107.07675*, 2021.

Jaeyeon Kim, Lee Cheuk-Kit, Carles Domingo-Enrich, Yilun Du, Sham Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Albergo. Any-order flexible length masked diffusion. *arXiv preprint arXiv:2509.01025*, 2025.

Dhruvesh Patel, Aishwarya Sahoo, Avinash Amballa, Tahira Naseem, Tim GJ Rudner, and Andrew McCallum. Insertion language models: Sequence generation with arbitrary-position insertions. *arXiv preprint arXiv:2505.05755*, 2025.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

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

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

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

Matteo Ninniri, Marco Podda, and Davide Bacciu. Graph diffusion that can insert and delete. *arXiv preprint arXiv:2506.15725*, 2025.

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

Chaitanya K. Joshi, Xiang Fu, Yi-Lun Liao, Vahe Gharakhanyan, Benjamin Kurt Miller, Anuroop Sriram, and Zachary W. Ulissi. All-atom diffusion transformers: Unified generative modelling of molecules and materials. *arXiv preprint arXiv:2503.03965*, 2025.

Cheng Zeng, Jirui Jin, Connor Ambrose, George Karypis, Mark Transtrum, Ellad B. Tadmor, Richard G. Hennig, Adrian Roitberg, Stefano Martiniani, and Mingjie Liu. PropMolFlow: property-guided molecule generation with geometry-complete flow matching. *Nature Computational Science*, pages 1–10, January 2026. ISSN 2662-8457. doi: 10.1038/s43588-025-00946-y. URL `https://www.nature.com/articles/s43588-025-00946-y`.

Alex Morehead, Miruna Cretu, Antonia Panescu, Rishabh Anand, Maurice Weiler, Tynan Perez, Samuel Blau, Steven Farrell, Wahid Bhimji, Anubhav Jain, Hrushikesh Sahasrabuddhe, Pietro Lio, Tommi Jaakkola, Rafael Gomez-Bombarelli, Rex Ying, N. Benjamin Erichson, and Michael W. Mahoney. Zatom-1: A multimodal flow foundation model for 3d molecules and materials. *arXiv preprint arXiv:2602.22251*, 2026.

Carlos Vonessen, Charles Harris, Miruna Cretu, and Pietro Liò. TABASCO: A fast, simplified model for molecular generation with improved physical quality. *arXiv preprint arXiv:2507.00899*, 2025.

Danny Reidenbach, Filipp Nikitin, Olexandr Isayev, and Saee Gopal Paliwal. Applications of modular co-design for de novo 3d molecule generation. *Digital Discovery*, 5(2):754–768, 2026.

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*, 6(4):417–427, 2024.

Jingyuan Zhou, Hao Qian, Shikui Tu, and Lei Xu. Prior-guided flow matching for target-aware molecule design with learnable atom number. *arXiv preprint arXiv: 2509.01486*, 2025.

Andrew Campbell, William Harvey, Christian Weilbach, Valentin De Bortoli, Tom Rainforth, and Arnaud Doucet. Trans-dimensional generative modeling via jump diffusion models. *arXiv preprint arXiv:2305.16261*, 2023.

Austin H. Cheng, Chong Sun, and Alán Aspuru-Guzik. Scalable autoregressive 3d molecule generation. *arXiv preprint arXiv:2505.13791*, 2025.

Rajendra P. Joshi, Niklas W. A. Gebauer, Mridula Bontha, Mercedeh Khazaieli, Rhema M. James, James B. Brown, and Neeraj Kumar. 3d-scaffold: A deep learning framework to generate 3d coordinates of drug-like molecules with desired scaffolds. *The Journal of Physical Chemistry B*, 125(44):12166–12176, 2021. doi: 10.1021/acs.jpcb.1c06437. URL `https://doi.org/10.1021/acs.jpcb.1c06437`. PMID: 34662142.

Ameya Daigavane, Song Eun Kim, Mario Geiger, and Tess Smidt. Symphony: Symmetry-equivariant point-centered spherical harmonics for 3d molecule generation. In *International Conference on Learning Representations*, volume 2024, pages 33975–34002, 2024.

Daniel Rose, Roxane Axel Jacob, Johannes Kirchmair, and Thierry Langer. Neat: Neighborhood-guided, efficient, autoregressive set transformer for 3d molecular generation. *arXiv preprint arXiv:2512.05844*, 2025.

Lukas Billera, Hedwig Nora Nordlinder, Jack Collier Ryder, Anton Oresten, Aron Stålmarck, Theodor Mosetti Björk, and Ben Murrell. Branching flows: Discrete, continuous, and manifold flow matching with splits and deletions. *arXiv preprint arXiv:2511.09465*, 2025.

Roman Poletukhin, Marcel Kollovieh, Eike Eberhard, and Stephan Günnemann. 3d molecule generation from rigid motifs via SE(3) flows. *arXiv preprint arXiv:2601.16955*, 2026.

Pedro O. Pinheiro, Arian Jamasb, Omar Mahmood, Vishnu Sresht, and Saeed Saremi. Structure-based drug design by denoising voxel grids. *arXiv preprint arXiv:2405.03961*, 2024a.

Pedro O. Pinheiro, Joshua Rackers, Joseph Kleinhenz, Michael Maser, Omar Mahmood, Andrew Martin Watkins, Stephen Ra, Vishnu Sresht, and Saeed Saremi. 3d molecule generation by denoising voxel grids. *arXiv preprint arXiv:2306.07473*, 2024b.

Felix Faltings, Hannes Stark, Regina Barzilay, and Tommi Jaakkola. ProxelGen: Generating proteins as 3d densities. *arXiv preprint arXiv:2506.19820*, 2025.

Tin Hadži Veljković, Erik Bekkers, Michael Tiemann, and Jan-Willem van de Meent. CORDS: Continuous representations of discrete structures. *arXiv preprint arXiv:2601.21583*, 2026.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

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, 2012.

Filipp Nikitin, Dylan M. Anstine, Roman Zubatyuk, Saee Gopal Paliwal, and Olexandr Isayev. Scalable low-energy molecular conformer generation with quantum mechanical accuracy. `https://github.com/isayevlab/LoQI`, 2025. Processed ChEMBL3D release: `https://doi.org/10.1184/R1/31441570`.

A Technical Appendices and Supplementary Material

A.1 Data details

QM9 The QM9 dataset Ramakrishnan et al. [2014] contains around 134k small organic molecules with up to 9 heavy atoms of types { `H, C, F, N, O` }. It is a subset of the enumerated GDB-17 dataset Ruddigkeit et al. [2012], filtered with heuristic for chemical plausibility. Structures were generated at the B3LYP/6-31G(2df,p) level of theory. There have been several inconsistencies in earlier versions of QM9, including molecules with non-zero charges and invalid bond orders. Therefore, we choose to use the corrected QM9 dataset from Zeng et al. [2026], and follow their train, validation and test split.

GEOM-Drugs The GEOM-Drugs dataset consists of more than 240k molecules relevant for computer-aided drug design Axelrod and Gomez-Bombarelli [2022]. Each molecule is represented by multiple conformers, leading to over 5.5M 3D structures. It contains atom types { `B, Bi, Br, C, Cl, F, H, I, N, O, P, S, Si` }. Similar to SemlaFlow Irwin et al. [2025], to increase training efficiency, we decide to remove all molecules with more than 72 atoms which constitute about 1% of the training data, and follow their train-val-test split.

A.2 Choices for schedules

When designing our schedules, we considered different stages across the generation time. In the first stage the model should focus on generating the rough molecule geometry and topology, whereas for late t the structure is finalized and relaxed to its energy minimum. To realize this, we make use of Smoothstep functions of the type

$$\kappa_t(\alpha) = 3t^{2\alpha} - 2t^{3\alpha} \tag{25}$$

which fulfill $\kappa_{t=0}(\alpha) = 0$ and $\kappa_{t=1}(\alpha) = 1$. We choose early-peaking schedules with $\alpha = 0.8$ for insertions and deletions. Substitutions are chosen to (likely) resolve after insertions ($\alpha = 1.5$), such that erroneous insertions could be corrected in the remaining time-interval.

A.3 Model details

We use a Diffusion Transformer (DiT) backbone similar to Joshi et al. [2025] with about 41M parameters for all experiments. The DiT-based model learns approximate equivariance via rotation augmentations. We therefore supply one random rotation for each molecule per batch. Figure 5 summarizes the backbone and output heads.

Embeddings and backbone Atom and bond types use learned categorical embeddings. We encode time with sinusoidal features and the current atom count with low-frequency Fourier features concatenated with a normalized scalar count. The latter retains a continuous count input beyond the training range. A learned projection of centered, normalized coordinates is added to the atom embeddings. The backbone uses eight transformer blocks of width 512 with 16 attention heads, query/key normalization, and SwiGLU feed-forward layers. Time and count embeddings condition the blocks through adaLN-Zero. Attention biases encode pairwise distances, bond types, and atom-type pairs. The output node and pair features have dimension 256.

State and edit heads Small neural readouts predict atom types, auxiliary charges, bond types, and edit decisions. The insertion-count head uses softplus; deletion and substitution logits are mapped to probabilities with a sigmoid. A residual coordinate head predicts clean positions and removes

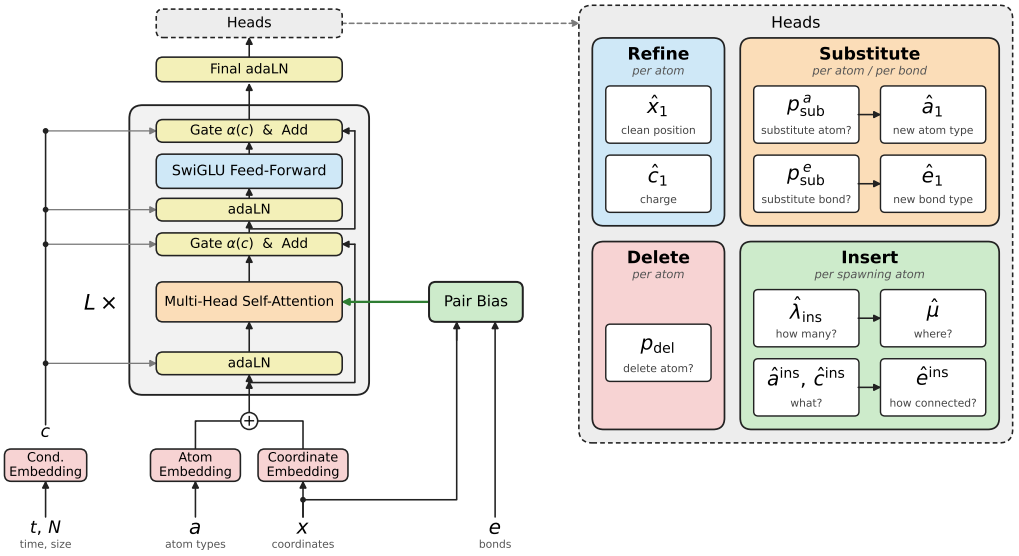


Figure 5: Architecture of **Morph**. Left: the DiT backbone combines atom and coordinate embeddings with time- and atom-count conditioning and pairwise attention biases. Right: readout heads predict clean coordinates and charges, substitutions, deletions, and unimodal insertions, including their count, position, atom type, charge, and bonds.

their graph-wise center of mass. Existing-bond features combine the sum and absolute difference of projected endpoint features with bond and distance embeddings, giving symmetric bond predictions.

Unimodal insertion head Each spawning node i, with hidden features h_i and position x_i, predicts one clean insertion position and categorical atom-type and charge probabilities:

$$\mu_i = x_i + f_x(h_i), \qquad A_i = \mathrm{softmax}(f_a(h_i)), \qquad C_i = \mathrm{softmax}(f_c(h_i)). \tag{26}$$

The readouts f_x, f_a, f_c use separate normalized, gated adapters and linear output projections. At inference, the clean insertion position is μ_i, while atom type and charge are sampled from A_i and C_i. The prescribed interpolation noise supplies the positional spread.

Inserted bonds For a new atom q spawned by i and an existing atom j, an MLP predicts bond logits from

$$m_{qj} = \big[h_i \parallel h_j \parallel a_j \parallel a_q \parallel c_q \parallel \mathrm{RBF}(d_{qj}) \parallel \mathrm{RBF}(d_{qi})\big], \tag{27}$$

where atom types and charges are one-hot encoded, d_{qj} and d_{qi} are distances from the clean inserted position, and $\parallel$ denotes concatenation. For bonds between two inserted atoms, a shared encoder combines each atom's spawning-node features, predicted attributes, and distance to its spawn. A second MLP uses the sum and absolute difference of these endpoint encodings together with their interatomic distance, ensuring symmetric logits. Bonds are predicted from clean inserted attributes; positions, atom types, and bonds are then noised to the next integration time using the interpolation schedules. Charge remains an auxiliary clean prediction.

Practical deviations from the full objective The implementation uses a simplified insertion-content objective. The Poisson count loss still supervises the full pending count $\Delta_i^{\mathrm{ins}} = |\mathcal{Q}_i|$, but the unimodal head is trained only on the nearest pending atom assigned to each spawning node with a nonempty $\mathcal{Q}_i$. Its position uses mean squared error over the three spatial coordinates, and its atom type and charge use cross-entropy. This replaces the likelihood sum over all assigned atoms in the full insertion-loss expression. Bond supervision still covers all pending inserted atoms, including bonds to existing atoms and between inserted atoms, using their clean target attributes. Components are averaged over their supervised targets within each molecule.

For practical inference, we cap insertions at one atom per spawning node per integration step. We combine the predicted pending count with the integrated schedule probability, clip the result to $[0, 1]$,

and draw a Bernoulli event. This cap applies to sampling; the training count target includes all pending insertions. Further atoms can be inserted in subsequent steps as the model is evaluated on the updated graph.

A.4 Training details

We optimize hidden two-dimensional weight matrices with Muon and use auxiliary AdamW for the remaining parameters, including embeddings and prediction heads. Tables 3 and 4 report the settings shared by the released unconditional QM9, GEOM-Drugs, and ChEMBL3D models. Both optimizers use linear warmup followed by cosine decay. We train in BF16 mixed precision with $t \sim \mathcal{U}(0, 1)$ and evaluate the exponential moving average (EMA) of the model parameters.

Table 3: Optimizer and scheduler settings. The schedule horizon includes warmup.

Parameter	Value
Muon peak learning rate	10^{-2}
Muon momentum (Nesterov)	0.95
Newton–Schulz iterations	5
Muon ε	10^{-7}
AdamW peak learning rate	3×10^{-4}
AdamW (β_1, β_2)	$(0.9, 0.95)$
AdamW ε	10^{-10}
Weight decay (both)	0
Global gradient-norm clip	1.0
LR warmup steps	2,000
LR schedule horizon (steps)	200,000
Minimum / peak learning rate	0.05
EMA decay	0.999
EMA warmup steps	5,000

Table 4: Components of Eq. (24), with references to their definitions.

Objective	Component	Weight
$\mathcal{J}_{\text{move}}$ (23)	MSE	2.00
$\mathcal{J}_{\text{charge}}$ (24)	CE	0.50
$\mathcal{J}_{\text{sub,a}}$ (22)	Decision (BCE)	0.20
	Type (CE)	0.03
$\mathcal{J}_{\text{sub,e}}$ (22)	Decision (BCE)	0.25
	Type (CE)	0.50
$\mathcal{J}_{\text{del}}$ (17)	BCE	0.40
	Count (Poisson NLL)	0.15
$\mathcal{J}_{\text{ins}}$ (14)	Content (approx.)	0.01
	Bonds to existing	0.40
	Bonds to inserted	0.03

The implemented objective sums component losses averaged over supervised targets per molecule and then over molecules with targets. Component weights are fixed, categorical class weights are uniform, and no additional time weighting is applied. The insertion-content weight multiplies the sum of position MSE and atom-type and charge cross-entropies. Appendix A.3 describes the practical deviations from the full objective.

The released unconditional models were trained on two NVIDIA RTX 4090 GPUs. QM9 uses 1,024 molecules per GPU and two gradient-accumulation steps, giving an effective batch size of 4,096. GEOM-Drugs and both ChEMBL3D variants use 320 molecules per GPU without accumulation, giving an effective batch size of 640. The reported checkpoints correspond to 1,500 completed epochs for QM9, 100 for GEOM-Drugs, and 200 for both ChEMBL3D variants. Checkpoint configurations are distributed with the public code and model release.

The fixed-target property-conditioned QM9 models were trained on four NVIDIA RTX 4090 GPUs with an effective batch size of 1,024 for 500 epochs (<12h total runtime).

A.5 Metrics

We use the following metric definitions, with stability evaluated according to SemlaFlow Irwin et al. [2025]:

- **Atom stability** is the fraction of atoms satisfying the element- and formal-charge-dependent valence table, including implicit hydrogens.
- **Molecule stability** is the fraction of generation attempts in which every atom satisfies this table; failed molecule construction counts as unstable.
- **Validity** is the fraction of all generation attempts that produce a connected graph passing RDKit sanitization (`connected-sanitization-validity`), with no additional radical or net-charge filter.

- **Uniqueness** is the number of distinct canonical molecular identities divided by the number of SMILES-convertible outputs. Updated summaries use hydrogen-stripped identities.

- **Novelty** is the fraction of SMILES-convertible outputs whose canonical identity is absent from the training reference, with duplicates retained in the denominator.

For all reported PoseBusters evaluations, pass rates divide the number of passing structures by the total number of generation attempts. Failed molecule construction and unavailable checks count as failures, and joint success requires every enabled check to pass. The unconditional, logP, and atom-count evaluations use all 12 checks, including internal energy and flatness, whereas the native scaffold protocol uses eight checks and excludes energy and flatness. Unless stated otherwise, uncertainties are sample standard deviations across three seeds, including uncertainties on mean properties. The multi-property experiment instead reports within-run property spreads.

A.6 Interpolation algorithm

Algorithm 1 Discrete Interpolation

function INTERPOLATEDISCRETE(y_0, y_1, κ)
  Sample $u \sim \mathcal{U}(0, 1)$
  return y_0 if $u > \kappa$ else y_1
end function

Algorithm 2 Molecular Graph Interpolation (Flow Matching Training)

Require: Prior sample $g_0 \sim \pi$ with nodes $\mathcal{S}_0$, Target graph $g_1 \sim p_{\text{data}}$ with nodes $\mathcal{S}_1$, Time $t \in [0, 1)$
Ensure: Interpolated state g_t, Active targets g_1^{filt}, Pending insertions $\mathcal{Q}$

// 1. Optimal Transport Alignment
Compute optimal assignment matrix $\hat{\mathbf{P}}^*$ aligning $\mathcal{S}_0$ and $\mathcal{S}_1$ ▷ Align valid node sets
Define node partitions based on non-zero entries of $\hat{\mathbf{P}}^*$:
  $\mathcal{M} \leftarrow \{i \mid \hat{\mathbf{P}}^*[i, j] = 1, i \in \mathcal{S}_0, j \in \mathcal{S}_1\}$ ▷ Matched nodes
  $\mathcal{D} \leftarrow \{i \mid \hat{\mathbf{P}}^*[i, l] = 1, i \in \mathcal{S}_0, l \notin \mathcal{S}_1\}$ ▷ Deletion nodes
  $\mathcal{I} \leftarrow \{j \mid \hat{\mathbf{P}}^*[k, j] = 1, k \notin \mathcal{S}_0, j \in \mathcal{S}_1\}$ ▷ Insertion nodes
// 2. Evaluate Schedules and Sample Event Times
Compute schedulers $\kappa_t^{\text{sub}}, \kappa_t^{\text{del}}, \kappa_t^{\text{ins}}$
Sample independent deletion event times $t_{\text{del},i} \sim \kappa^{\text{del}}$ for $i \in \mathcal{D}$
Sample independent insertion event times $t_{\text{ins},j} \sim \kappa^{\text{ins}}$ for $j \in \mathcal{I}$
// 3. Determine Active Nodes at Time t
$\mathcal{V}_t \leftarrow \mathcal{M}$
$\mathcal{V}_t \leftarrow \mathcal{V}_t \cup \{i \in \mathcal{D} \mid t < t_{\text{del},i}\}$ ▷ Not yet deleted
$\mathcal{V}_t \leftarrow \mathcal{V}_t \cup \{j \in \mathcal{I} \mid t > t_{\text{ins},j}\}$ ▷ Already inserted
Let $n_t \leftarrow |\mathcal{V}_t|$
// 4. Interpolate Graph Features
for each active node $k \in \mathcal{V}_t$ **do**
  if $k \in \mathcal{D}$ **then**
    $x_{t,k} \leftarrow x_{0,k}$ ▷ Fix positions at prior
    $a_{t,k} \leftarrow a_{0,k}$ ▷ Fix atom types at prior
  else if $k \in \mathcal{I}$ **then**
    $\sigma_t \leftarrow \sigma(1 - t) + \varepsilon$
    Sample $x_{t,k} \sim \mathcal{N}(x_{1,k}, \sigma_t^2 I)$ ▷ Gaussian centered on target
    $a_{t,k} \leftarrow \text{InterpolateDiscrete}(a_{0,k}, a_{1,k}, \kappa_t^{\text{ins}})$
  else ▷ $k \in \mathcal{M}$
    Sample $x_{t,k} \sim \mathcal{N}((1 - t)x_{0,k} + tx_{1,k}, \sigma^2 I)$ ▷ Standard Gaussian path
    $a_{t,k} \leftarrow \text{InterpolateDiscrete}(a_{0,k}, a_{1,k}, \kappa_t^{\text{sub}})$
  end if
end for
for each valid pair (u, v) in $\mathcal{V}_t$ **do**
  if $u \in \mathcal{M} \wedge v \in \mathcal{M}$ **then**
    $e_{t,uv} \leftarrow \text{InterpolateDiscrete}(e_{0,uv}, e_{1,uv}, \kappa_t^{\text{sub}})$
  else if $u \in \mathcal{I} \vee v \in \mathcal{I}$ **then**
    $e_{t,uv} \leftarrow \text{InterpolateDiscrete}(e_{0,uv}, e_{1,uv}, \kappa_t^{\text{ins}})$
  else ▷ Involves deletion nodes
    $e_{t,uv} \leftarrow e_{0,uv}$ ▷ Fix edges at prior attributes
  end if
end for
Symmetrize $\mathcal{E}_t$ so $e_{t,vu} = e_{t,uv}$
// 5. Construct Targets and Local Spawn Assignments
$g_t \leftarrow (n_t, \mathcal{V}_t, \mathcal{E}_t)$
Filter $\tilde{g}_1$ to only contain target features for active nodes $\mathcal{V}_t$, yielding g_1^{filt}
Initialize pending insertion sets $\mathcal{Q}_k \leftarrow \emptyset$ for all $k \in \mathcal{V}_t$
for each pending insertion node $j \in \mathcal{I} \setminus \mathcal{V}_t$ **do**
  $k^* \leftarrow \arg\min_{k \in \mathcal{V}_t} \|x_{t,k} - x_{1,j}\|_2$ ▷ Find nearest active neighbor
  $\mathcal{Q}_{k^*} \leftarrow \mathcal{Q}_{k^*} \cup \{j\}$
end for
Center x_t and active targets x_1^{filt} to zero center-of-mass
return $g_t, g_1^{\text{filt}}, \{\mathcal{Q}_k\}_{k=1}^{n_t}$

A.7 Sampling

Algorithm 3 Jump-flow integration for molecular generation

Require: Hazard schedules $\kappa_t^{\mathrm{del}}, \kappa_t^{\mathrm{sub}}, \kappa_t^{\mathrm{ins}}$; time grid $0 = t_0 < t_1 < \cdots < t_S = 1$ with $\Delta t_s = t_{s+1} - t_s$; prior π; trained network u_t^θ

Sample node count $n_0 \sim \mathcal{U}(\mathcal{N}_{\mathrm{train}})$ ▷ Uniform draw over train distribution of atom counts
Sample prior graph $g_0 = (n_0, \mathcal{V}_0, \mathcal{E}_0) \sim \pi(g_0 \mid n_0)$ ▷ $\mathcal{V}_0 = (p_i)_{i=1}^{n_0}$ and $\mathcal{E}_0 = [e_{ij}]_{i,j=1}^{n_0}$
for $s = 0, \ldots, S-1$ **do**
  $t \leftarrow t_s, \quad \Delta t \leftarrow \Delta t_s$
  $H_{t,\Delta t}^k \leftarrow \displaystyle\int_t^{t+\Delta t} \frac{\dot\kappa_\tau^k}{1-\kappa_\tau^k}\, d\tau = \log \frac{1-\kappa_t^k}{1-\kappa_{t+\Delta t}^k}$ for $k \in \{\mathrm{del}, \mathrm{sub}, \mathrm{ins}\}$
  $\rho_{t,\Delta t}^k \leftarrow 1 - \exp(-H_{t,\Delta t}^k) = \frac{\kappa_{t+\Delta t}^k - \kappa_t^k}{1-\kappa_t^k}$ ▷ Conditional schedule probability; equals 1 at $t + \Delta t = 1$
  $\big((\hat{x}_{i,1}, \hat{a}_{i,1}, \hat{c}_{i,1})_{i=1}^{n_t}, [\hat{e}_{ij,1}]_{i,j=1}^{n_t}, \Delta^{\mathrm{sub,a}}, \Delta^{\mathrm{sub,e}}, \Delta^{\mathrm{del}}, \Delta^{\mathrm{ins}}, Q_t^{\mathrm{ins}}\big) \leftarrow u_t^\theta(g_t)$

  // 1. Continuous updates
  for each node index $i \in \{1, \ldots, n_t\}$ **do**
    $x_i \leftarrow x_i + \frac{\hat{x}_{i,1} - x_i}{1-t}\, \Delta t$ ▷ Position Euler step
    $c_i \leftarrow \hat{c}_{i,1}$ ▷ Auxiliary charge prediction takes final-frame target
  end for

  // 2. Node Deletions and Substitutions (Competing Events)
  for each node index $i \in \{1, \ldots, n_t\}$ **do**
    $q_i^{\mathrm{sub,a}} \leftarrow \Delta_i^{\mathrm{sub,a}} \cdot \rho_{t,\Delta t}^{\mathrm{sub}}$
    $q_i^{\mathrm{del}} \leftarrow \Delta_i^{\mathrm{del}} \cdot \rho_{t,\Delta t}^{\mathrm{del}}$
    $q_i^{\mathrm{total}} \leftarrow q_i^{\mathrm{sub,a}} + q_i^{\mathrm{del}}$
    Sample $u_1, u_2 \sim \mathcal{U}(0,1)$
    if $u_1 < q_i^{\mathrm{total}}$ **then**
      if $u_2 < q_i^{\mathrm{del}} / q_i^{\mathrm{total}}$ **then**
        Remove p_i and its incident edges from g_t ▷ Node deletion
      else
        $a_i \leftarrow \hat{a}_{i,1}$ ▷ Atom-type substitution
      end if
    end if
  end for

  // 3. Edge Substitutions
  for each valid pair (i, j) with $i < j$ in surviving graph **do**
    Sample $b_{ij}^{\mathrm{sub,e}} \sim \mathrm{Bernoulli}(\Delta_{ij}^{\mathrm{sub,e}} \cdot \rho_{t,\Delta t}^{\mathrm{sub}})$
    if $b_{ij}^{\mathrm{sub,e}} = 1$ **then**
      $e_{ij} \leftarrow \hat{e}_{ij,1}$
    end if
  end for
  Symmetrize $\mathcal{E}_t$ so $e_{ji} = e_{ij}$

  // 4. Node Insertions
  for each node index i present at start of step **do**
    Sample $b_i^{\mathrm{ins}} \sim \mathrm{Bernoulli}(\min\{1, \Delta_i^{\mathrm{ins}} \cdot \rho_{t,\Delta t}^{\mathrm{ins}}\})$ ▷ At most one insertion per spawning node
    if $b_i^{\mathrm{ins}} = 1$ **then**
      Sample $(x_q, a_q, c_q) \sim Q_{t,i}^{\mathrm{ins}}(q \mid g_t)$ ▷ Unimodal insertion head
      Sample edges $\mathbf{e}_q \sim Q_{t,i}^{\mathrm{ins}}(\mathbf{e}_q \mid q, g_t)$
      Add new node $p_q = (a_q, x_q)$, charge c_q, and edges $\mathbf{e}_q$ to g_t
    end if
  end for
  Update $n_{t+\Delta t} \leftarrow |\mathcal{V}_{t+\Delta t}|$
end for
return g_1

B Further results

B.1 De novo design

Table 5: QM9 de novo generation. Baselines are adapted from SemlaFlow Irwin et al. [2025]. The **Morph** row uses the explicit-H checkpoint after 1,500 completed epochs, evaluated with 10,000 attempts per seed and three seeds. Stability uses the valence table; validity is connected sanitization.

Model		Atom Stab ↑	Mol Stab ↑	Valid ↑	Unique ↑	NFE
n_{atoms} fixed	FlowMol	99.7	96.2	97.3	–	100
	MiDi	99.8	97.5	97.9	97.6	500
	EQGAT-diff	$99.9_{\pm 0.0}$	$98.7_{\pm 0.18}$	$99.0_{\pm 0.16}$	$\mathbf{100.0}_{\pm 0.0}$	500
	SemlaFlow	$99.9_{\pm 0.0}$	$\mathbf{99.7}_{\pm 0.03}$	$\mathbf{99.4}_{\pm 0.03}$	$95.4_{\pm 0.0}$	100
	Morph (Ours)	$\mathbf{100.00}_{\pm 0.00}$	$99.11_{\pm 0.06}$	$\underline{99.14}_{\pm 0.08}$	$94.91_{\pm 0.11}$	100

Table 6: Unconditional PoseBusters results. Every value is a percentage of all 10,000 generation attempts per seed, reported as mean$_{\pm \text{SD}}$ across 3 seeds. Failed construction and unavailable checks count as failures. QM9 and GEOM-Drugs use explicit-H checkpoints after 1,500 and 100 completed epochs. Both ChEMBL3D models use checkpoints after 200 epochs. All 12 checks are enabled; the implicit-H ChEMBL3D scores refer to heavy-atom structures.

Metric	QM9	GEOM-Drugs	ChEMBL3D explicit H	ChEMBL3D implicit H
Molecule loaded	$99.16_{\pm 0.08}$	$96.69_{\pm 0.22}$	$96.83_{\pm 0.13}$	$99.05_{\pm 0.18}$
Sanitization	$99.16_{\pm 0.08}$	$96.69_{\pm 0.22}$	$96.83_{\pm 0.13}$	$99.05_{\pm 0.18}$
All atoms connected	$99.14_{\pm 0.08}$	$95.46_{\pm 0.31}$	$95.80_{\pm 0.17}$	$98.87_{\pm 0.18}$
Bond lengths	$99.15_{\pm 0.08}$	$96.04_{\pm 0.25}$	$96.33_{\pm 0.18}$	$97.33_{\pm 0.06}$
Bond angles	$99.15_{\pm 0.09}$	$96.41_{\pm 0.22}$	$96.57_{\pm 0.15}$	$98.58_{\pm 0.17}$
Internal steric clash	$99.01_{\pm 0.11}$	$92.53_{\pm 0.10}$	$95.95_{\pm 0.12}$	$98.31_{\pm 0.24}$
No radicals	$99.01_{\pm 0.05}$	$94.10_{\pm 0.31}$	$94.44_{\pm 0.11}$	$99.05_{\pm 0.18}$
InChI convertible	$99.15_{\pm 0.08}$	$96.67_{\pm 0.24}$	$96.81_{\pm 0.13}$	$99.05_{\pm 0.18}$
Internal energy	$94.21_{\pm 0.17}$	$96.38_{\pm 0.25}$	$96.55_{\pm 0.17}$	$98.78_{\pm 0.15}$
Aromatic ring flatness	$99.16_{\pm 0.08}$	$96.69_{\pm 0.23}$	$96.82_{\pm 0.12}$	$99.04_{\pm 0.18}$
Double-bond flatness	$98.82_{\pm 0.12}$	$96.25_{\pm 0.34}$	$95.88_{\pm 0.14}$	$98.59_{\pm 0.22}$
Non-aromatic ring non-flatness	$98.89_{\pm 0.10}$	$96.16_{\pm 0.15}$	$96.40_{\pm 0.11}$	$98.55_{\pm 0.21}$
All 12 checks	$93.53_{\pm 0.19}$	$88.30_{\pm 0.24}$	$91.46_{\pm 0.24}$	$95.27_{\pm 0.16}$

B.2 Additional de novo experiment on ChEMBL3D

We additionally evaluate unconditional generation on ChEMBL3D using the processed LoQI release Nikitin et al. [2025], retaining its train/validation/test partitions. The released training subset is restricted to molecules with at most 100 atoms before removing hydrogens. We evaluate both explicit-H graphs and heavy-atom graphs with implicit hydrogens.

For each hydrogen representation, we select the checkpoint with the highest mean connected-sanitization validity among the three supplied checkpoints. Both selected models have completed 200 training epochs. This selection criterion follows the paper's definition of validity. Each seed generates 10,000 structures. Joint PoseBusters success uses all attempts and requires all 12 enabled checks; individual pass rates are reported in Table 6.

Table 7: Unconditional ChEMBL3D generation with explicit-H and implicit-H models after 200 completed epochs. Rates are percentages, except diversity, and are reported as mean$_{\pm \mathrm{SD}}$ across three seeds. Validity and PoseBusters use all generation attempts. Uniqueness and novelty use SMILES-convertible outputs.

Metric	Explicit H	Implicit H
Atom stability	$99.98_{\pm 0.00}$	$99.99_{\pm 0.00}$
Molecule stability	$96.14_{\pm 0.18}$	$98.85_{\pm 0.21}$
Connected-sanitization validity	$95.80_{\pm 0.17}$	$98.87_{\pm 0.18}$
All 12 PoseBusters checks	$91.46_{\pm 0.24}$	$95.27_{\pm 0.16}$
Uniqueness	$99.99_{\pm 0.02}$	$100.00_{\pm 0.01}$
Novelty	$99.69_{\pm 0.05}$	$99.64_{\pm 0.10}$
Diversity	$0.8953_{\pm 0.0002}$	$0.9140_{\pm 0.0002}$

B.3 Sampling efficiency

This separate GEOM-Drugs timing experiment was run on one NVIDIA RTX 4090 primarily to compare sampling speed under shared hardware. Validity values therefore need not exactly match the aggregated results in Table 1. At the same 100 NFE, **Morph** generates 27.6 molecules/s with 95.6% validity, compared with 9.2 molecules/s and 93.9% validity for SemlaFlow. Thus, **Morph** provides three times the throughput at this operating point. Varying its NFE exposes a controllable speed–quality trade-off, from 202.5 molecules/s at 10 NFE to 97.0% validity at 200 NFE. FlowMol3 attains the highest validity, but uses 250 NFE and samples 2.7 molecules/s.

Table 8: Sampling efficiency on one NVIDIA RTX 4090. The bold **Morph** row (100 NFE) is the setting used throughout the main text.

Model	NFE	Throughput (molecules/s)	Validity (%)
SemlaFlow	100	9.2	93.9
FlowMol3	250	2.7	99.3
	10	202.5	88.1
	25	83.0	94.0
	50	46.1	95.7
Morph (Ours)	**100**	**27.6**	**95.6**
	200	15.3	97.0
	500	6.6	96.3
	1000	3.4	96.6

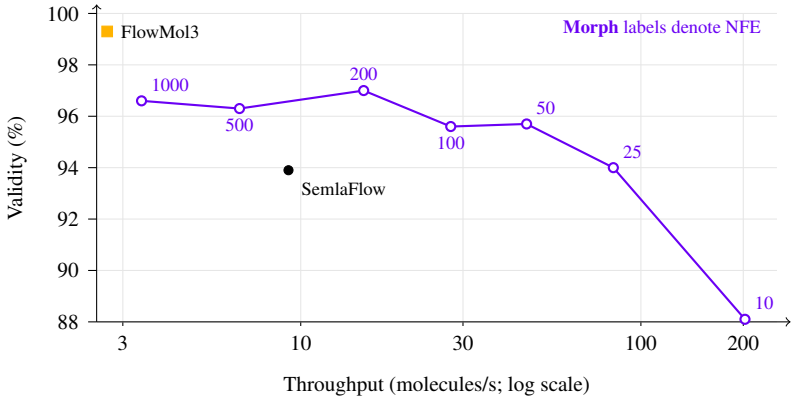


Figure 6: Validity–throughput trade-off on one NVIDIA RTX 4090. The line connects **Morph** operating points; numerical labels give NFE.

B.4 Scaffold decoration on GEOM-Drugs

Table 9: GEOM-Drugs scaffold decoration with the rank-16 LoRA model after 25 completed fine-tuning epochs. Each seed generates eight completions for each of 512 test scaffolds (4,096 attempts). Rates are percentages of all attempts, reported as mean$_{\pm \mathrm{SD}}$ across three seeds; failed construction and unavailable checks count as failures. The native eight-check protocol excludes energy and flatness and uses no inherited-clash exemptions.

Metric	Pass rate (%, ↑)
Molecule loaded	$98.13_{\pm 0.15}$
Sanitization	$98.13_{\pm 0.15}$
All atoms connected	$97.99_{\pm 0.20}$
Bond lengths	$97.29_{\pm 0.22}$
Bond angles	$97.79_{\pm 0.12}$
Internal steric clash	$95.64_{\pm 0.22}$
No radicals	$97.44_{\pm 0.17}$
InChI convertible	$98.13_{\pm 0.15}$
All eight checks	$94.10_{\pm 0.19}$

B.5 In-pocket scaffold decoration

For in-pocket scaffold decoration, we initialize **Morph** from the pretrained ChEMBL3D implicit-H model and fine-tune it on the CrossDocked2020 multi-R task used by DiffDec Xie et al. [2024]. Given an initial scaffold in a protein pocket, the model generates multiple synthetically relevant substituents simultaneously, without information about anchor atoms or the final number of atoms. We take the baseline results directly from DiffDec. Because DiffDec uses QVina whereas we use AutoDock Vina 1.2.7, we report the reference score from each evaluation so that generated molecules can be compared with their corresponding reference molecules.

Table 10: In-pocket scaffold decoration on the CrossDocked2020 multi-R task. Baseline results, including QVina scores, are reported from DiffDec Xie et al. [2024]; **Morph** and its reference molecules are evaluated with AutoDock Vina 1.2.7. Docking scores should therefore be compared with the corresponding reference row rather than directly across the two blocks. High affinity is the percentage of generated molecules with a lower docking score than their corresponding reference molecule. The best comparable generated-model result in each column is shown in bold.

Method	Valid. (%, ↑)	Unique (%, ↑)	Recovery (%, ↑)	Similarity (↑)	Docking score (kcal/mol, ↓)	High affinity (%, ↑)
Scaffolds	–	–	–	–	−7.77	–
DiffDec reference molecules	–	–	–	–	−8.46	–
LibINVENT	8.62	6.93	26.85	0.70	−7.83	4.57
Pocket2Mol	60.90	50.84	1.34	0.68	−8.04	31.32
FLAG	88.18	64.73	0.00	0.62	−7.73	38.62
DiffDec (w/o anchors)	89.84	**71.75**	29.54	0.73	−8.15	34.67
DiffDec	**97.13**	60.12	**45.34**	**0.80**	−8.19	**40.50**
Morph reference molecules	–	–	–	–	−8.42	–
Morph (Ours)	89.80	66.43	37.83	0.77	−8.09	40.33

Using the standard **Morph** formulation, without anchor annotations or the final number of atoms, we obtain competitive results on in-pocket scaffold decoration. In particular, 89.80% of generated molecules are valid and 40.33% have higher predicted binding affinity than their corresponding reference molecule. Qualitative examples are shown in Fig. 7.

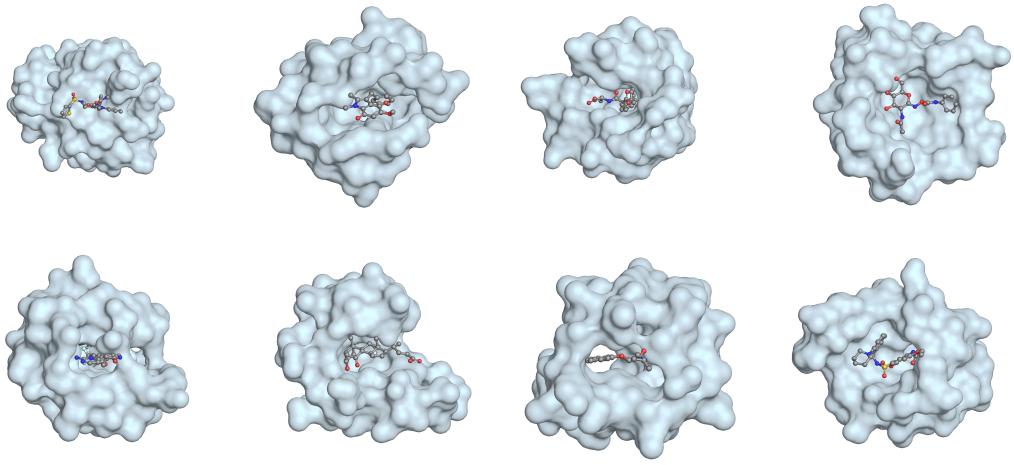


Figure 7: Example in-pocket scaffold decorations generated by **Morph** on the CrossDocked2020 multi-R task. Protein pockets are rendered as surfaces and generated molecules as sticks.

B.6 Atom-count conditioning on GEOM-Drugs

The empirical-target evaluation uses 10,000 attempts per seed, and each fixed target uses 5,000 attempts per seed. All runs use three seeds. The generated atom-count mean, MAE, and exact-match rate include all attempts, rather than only valid outputs. The fixed targets exceed the 72-atom training cutoff, therefore operating in the OOD regime. Validity and exact size targeting deteriorate as the requested size increases.

Table 11: GEOM-Drugs atom-count conditioning. Validity uses connected sanitization; PB is joint success on all 12 PoseBusters checks. Both rates and exact matches are percentages of all attempts. Values are mean$_{\pm\mathrm{SD}}$ across three seeds. Uniqueness is 100% for every setting; novelty is $99.90_{\pm 0.03}\%$ for empirical targets and 100% for fixed targets.

Target n		Valid (%)	PB (%)	Mean n	MAE (atoms)	Exact (%)
Empirical ($n < 72$)		$93.84_{\pm 0.09}$	$85.55_{\pm 0.45}$	$46.87_{\pm 0.06}$	$0.790_{\pm 0.003}$	$37.21_{\pm 0.10}$
OOD	80	$76.96_{\pm 0.09}$	$61.60_{\pm 0.51}$	$81.56_{\pm 0.01}$	$1.590_{\pm 0.006}$	$10.65_{\pm 0.26}$
	90	$66.93_{\pm 0.60}$	$45.53_{\pm 0.93}$	$92.29_{\pm 0.01}$	$2.294_{\pm 0.014}$	$2.50_{\pm 0.19}$
	100	$49.20_{\pm 1.15}$	$25.72_{\pm 0.44}$	$103.47_{\pm 0.01}$	$3.465_{\pm 0.009}$	$0.05_{\pm 0.03}$
	110	$25.14_{\pm 0.73}$	$7.71_{\pm 0.20}$	$115.38_{\pm 0.02}$	$5.378_{\pm 0.021}$	$0.00_{\pm 0.00}$

B.7 Pooled logP conditioning

The pooled evaluations use 10,000 generation attempts per dataset and seed on QM9 and GEOM-Drugs. They are separate from the fixed-target QM9 experiments below. Validity and PoseBusters success use all attempts. LogP MAE and R^2 use outputs with a finite computed logP; the scored fraction makes this population explicit. The target sets have mean (standard deviation) 0.213 (0.889) on QM9 and 2.811 (1.329) on GEOM-Drugs.

Table 12: Pooled logP-conditioned generation. Validity is connected sanitization, PB denotes success on all 12 PoseBusters checks, and scored is the fraction with a finite computed logP. These three fractions are percentages of all generation attempts. Values are mean$_{\pm\mathrm{SD}}$ across three seeds.

Dataset	Valid (%)	PB (%)	Scored (%)	LogP MAE	R^2
QM9	$98.69_{\pm 0.05}$	$92.83_{\pm 0.18}$	$98.73_{\pm 0.05}$	$0.304_{\pm 0.002}$	$0.815_{\pm 0.003}$
GEOM-Drugs	$94.49_{\pm 0.05}$	$87.14_{\pm 0.21}$	$95.96_{\pm 0.14}$	$0.538_{\pm 0.001}$	$0.728_{\pm 0.002}$

Table 13: PoseBusters for the conditional runs (10,000 attempts per seed). Every value is a percentage of all generation attempts, including failures, reported as mean$_{\pm \mathrm{SD}}$ across three seeds. GEOM-n uses empirical atom-count targets. All three columns use the same 12-check protocol.

Metric	QM9 logP	GEOM logP	GEOM-n
Molecule loaded	$98.73_{\pm 0.05}$	$95.96_{\pm 0.14}$	$95.51_{\pm 0.13}$
Sanitization	$98.73_{\pm 0.05}$	$95.96_{\pm 0.14}$	$95.51_{\pm 0.13}$
All atoms connected	$98.69_{\pm 0.05}$	$94.49_{\pm 0.05}$	$93.84_{\pm 0.09}$
Bond lengths	$98.69_{\pm 0.04}$	$95.16_{\pm 0.13}$	$94.73_{\pm 0.23}$
Bond angles	$98.72_{\pm 0.06}$	$95.70_{\pm 0.20}$	$95.18_{\pm 0.15}$
Internal steric clash	$98.62_{\pm 0.04}$	$91.90_{\pm 0.14}$	$91.00_{\pm 0.12}$
No radicals	$98.23_{\pm 0.06}$	$93.05_{\pm 0.13}$	$91.86_{\pm 0.35}$
InChI convertible	$98.73_{\pm 0.05}$	$95.94_{\pm 0.14}$	$95.46_{\pm 0.12}$
Internal energy	$93.66_{\pm 0.24}$	$95.62_{\pm 0.19}$	$95.16_{\pm 0.08}$
Aromatic ring flatness	$98.73_{\pm 0.05}$	$95.96_{\pm 0.14}$	$95.50_{\pm 0.14}$
Double-bond flatness	$98.48_{\pm 0.05}$	$95.57_{\pm 0.14}$	$94.98_{\pm 0.08}$
Non-aromatic ring non-flatness	$98.57_{\pm 0.03}$	$95.56_{\pm 0.19}$	$94.94_{\pm 0.13}$
All 12 checks	$92.83_{\pm 0.18}$	$87.14_{\pm 0.21}$	$85.55_{\pm 0.45}$

B.8 Fixed-target logP conditioning on QM9

Table 14: Fixed-target logP conditioning on QM9 (10,000 generation attempts per target, one run per target). Rates are percentages. Validity requires connected sanitization; PB is joint success on all 12 PoseBusters checks over all attempts. Stability uses the valence table. N_{logP} counts outputs with finite logP; the mean$_{\pm \mathrm{SD}}$ and MAE use all such outputs. These runs are distinct from the pooled evaluation in Table 12; Table 16 reports the same **Morph** runs under stricter molecule filtering.

logP	Valid	PB	Uniq.	Nov.	Atom stab.	Mol. stab.	N_{logP}	μ_{logP}	$\mathrm{MAE}_{\mathrm{logP}}$
-4	76.82	62.17	65.81	81.55	99.63	72.91	7,730	$-2.833_{\pm 0.976}$	1.271
-2	96.20	88.94	68.63	51.02	99.97	95.82	9,633	$-1.654_{\pm 0.468}$	0.465
0	98.95	92.50	90.87	23.50	99.99	98.88	9,899	$0.077_{\pm 0.371}$	0.305
2	97.72	92.02	62.97	11.06	99.99	97.66	9,785	$2.039_{\pm 0.383}$	0.313
4	27.55	26.93	17.39	98.80	99.55	27.96	3,157	$3.751_{\pm 0.151}$	0.261

Table 15: PoseBusters for fixed-target QM9 logP conditioning, using the same 10,000 generation attempts per target as Table 14. Each value is a percentage of all attempts from one run per target; failed construction and unavailable checks count as failures. Joint success requires all 12 checks to pass.

Metric	$\mathrm{logP}_{\mathrm{cond}}$				
	-4	-2	0	2	4
Molecule loaded	77.30	96.33	98.99	97.85	31.57
Sanitization	77.30	96.33	98.99	97.85	31.57
All atoms connected	76.82	96.20	98.95	97.72	27.55
Bond lengths	76.95	96.25	98.89	97.78	30.76
Bond angles	77.14	96.27	98.95	97.83	31.38
Internal steric clash	76.13	95.65	98.86	97.79	30.61
No radicals	65.13	94.42	98.42	97.43	27.30
InChI convertible	76.10	96.28	98.99	97.85	31.57
Internal energy	75.03	91.67	93.57	92.90	31.40
Aromatic ring flatness	77.30	96.31	98.99	97.85	31.57
Double-bond flatness	77.20	96.24	98.59	97.20	31.57
Non-aromatic ring non-flatness	76.88	96.16	98.73	97.67	31.57
All 12 checks	62.17	88.94	92.50	92.02	26.93

Pooling the 36,845 sanitized, connected, neutral, radical-free **Morph** outputs with finite logP across targets gives a global MAE/R^2 of 0.468/0.916. SemlaFlow reports 0.607/0.909 using $p(n)$ and 0.489/0.943 using the oracle $p(n \mid \text{logP})$ (Table 16). The different molecule filters limit direct comparison of these scores.

Table 16: Target-wise logP-steering performance using 10,000 generation attempts per target for **Morph** (one seed) and 5,000 per target for SemlaFlow. **Morph** statistics use sanitized, connected, neutral, radical-free molecules; the existing SemlaFlow results use sanitized molecules without the same additional filters. SemlaFlow selects its fixed atom count from either $p(n)$ or the oracle $p(n \mid \text{logP})$ estimated on the training set. QM9 slices use $|\text{logP} - \text{logP}_{\text{cond}}| \leq 1$; atom counts include hydrogens. Lowest errors and means closest to their target or reference are bolded using unrounded values. W_1 at target $+4$ is omitted.

Metric	Model	$\text{logP}_{\text{cond}}$				
		-4	-2	0	2	4
Mean n_{atoms}	QM9 slice	16.5	15.9	17.1	20.0	26.1
Mean logP	QM9 slice	-3.62	-1.39	0.07	1.57	3.16
logP MAE ↓	**Morph**	1.20	0.46	0.30	0.31	**0.27**
	SemlaFlow, $p(n)$	0.98	0.48	0.29	0.28	1.15
	SemlaFlow, oracle	**0.82**	**0.41**	**0.26**	**0.21**	0.82
Mean logP	**Morph**	-2.91	-1.66	0.08	**2.04**	**3.74**
	SemlaFlow, $p(n)$	-3.13	-1.78	0.05	1.84	2.85
	SemlaFlow, oracle	$\mathbf{-3.29}$	$\mathbf{-1.84}$	**0.02**	1.93	3.18
Mean n_{atoms}	**Morph**	**16.8**	**16.4**	17.1	21.2	31.8
	SemlaFlow, $p(n)$	18.0	18.0	18.0	18.1	18.0
	SemlaFlow, oracle	17.0	17.3	**17.1**	**21.0**	**25.3**
$W_1(n_{\text{atoms}})$ ↓	**Morph**	0.6	**0.5**	0.4	1.2	–
	SemlaFlow, $p(n)$	1.6	2.1	0.9	1.9	–
	SemlaFlow, oracle	**0.5**	1.4	**0.3**	**0.9**	–

B.9 Multi-property conditioning QED-logP

We perform additional experiments on multi-property conditioning to evaluate **Morph**'s ability to generate molecules satisfying joint property requirements. We combine QED, which measures drug-likeness, with logP, using the target grid $\text{logP}_{\text{cond}} \in \{-2, 0, 2\}$ and $\text{QED}_{\text{cond}} \in \{0.3, 0.5, 0.7\}$, with 10,000 generation attempts per pair. Figure 8 and Table 17 report the results. The $(-2, 0.7)$ setting yields 11.74% validity and 5.28% PoseBusters success, but no output meets both targeting tolerances.

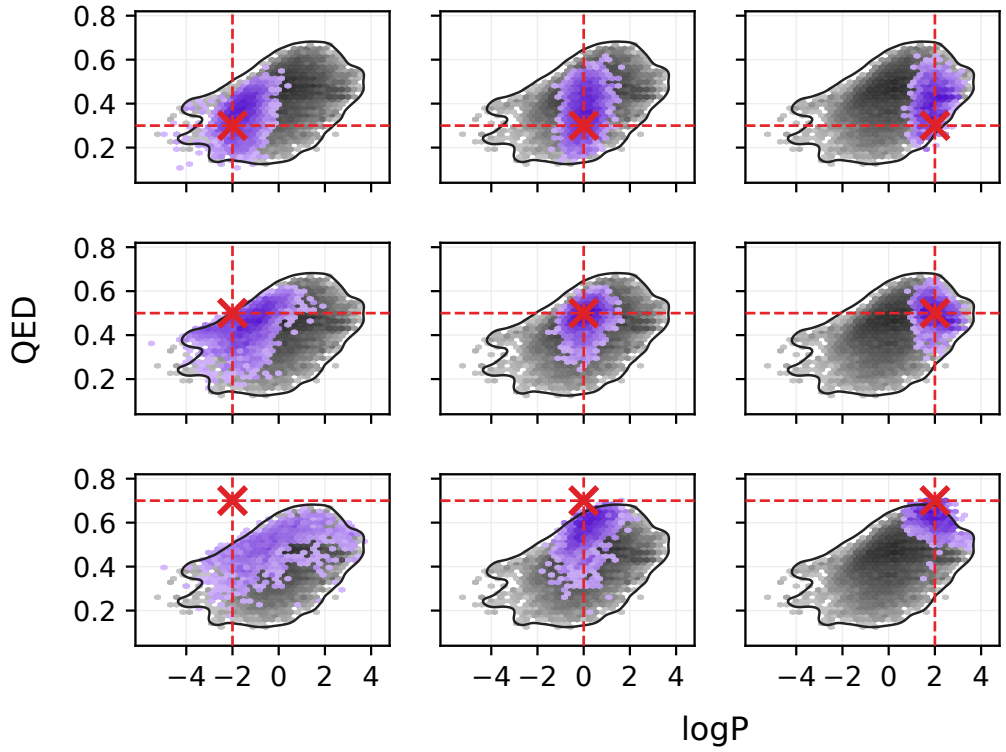


Figure 8: Joint QED–logP conditioning on QM9, with 10,000 generation attempts per target pair. Generated property distributions are shown in purple against the training distribution in gray. Red crosses and dashed lines mark the targets: logP values $-2, 0, 2$ across columns and QED values $0.3, 0.5, 0.7$ down rows.

Table 17: Joint QED–logP conditioning. Rates are percentages. Valid is connected sanitization; PB requires all 12 PoseBusters checks. Scored denotes outputs with finite logP and QED values. Valid, PB, Scored, and Joint use all attempts. Uniqueness and novelty use SMILES-convertible outputs; stability uses the valence table. Means (μ), SDs, and MAEs use scored outputs within each run. Joint requires both $|\text{logP} - \text{logP}_{\text{cond}}| \leq 0.5$ and $|\text{QED} - \text{QED}_{\text{cond}}| \leq 0.05$.

Targets		Generative quality (%)						Scored	logP		QED		Joint
logP	QED	Valid	PB	Uniq.	Nov.	Atom stab.	Mol. stab.	(%)	μ	MAE	μ	MAE	(%)
-2.0	0.30	96.99	89.71	63.94	46.14	99.97	96.57	97.05	$-1.66_{\pm 0.46}$	0.453	$0.36_{\pm 0.06}$	0.070	28.04
-2.0	0.50	87.82	81.75	66.73	74.58	99.80	85.09	88.26	$-1.32_{\pm 0.76}$	0.869	$0.45_{\pm 0.06}$	0.059	8.91
-2.0	0.70	11.74	5.28	94.36	99.20	96.16	7.87	17.56	$-0.42_{\pm 1.35}$	1.684	$0.49_{\pm 0.08}$	0.210	0.00
0.0	0.30	98.85	91.87	77.86	22.06	99.99	98.77	98.90	$0.07_{\pm 0.38}$	0.307	$0.37_{\pm 0.07}$	0.082	29.15
0.0	0.50	98.95	94.43	84.74	24.51	99.99	98.86	98.99	$0.08_{\pm 0.39}$	0.324	$0.49_{\pm 0.04}$	0.030	66.35
0.0	0.70	88.72	87.28	39.18	68.36	99.94	88.68	89.63	$0.23_{\pm 0.46}$	0.408	$0.59_{\pm 0.05}$	0.109	2.03
2.0	0.30	97.02	93.28	46.41	18.85	99.99	96.91	97.12	$1.98_{\pm 0.38}$	0.304	$0.40_{\pm 0.06}$	0.099	19.81
2.0	0.50	96.39	92.71	55.76	11.59	99.99	96.28	96.54	$2.04_{\pm 0.41}$	0.336	$0.50_{\pm 0.04}$	0.030	63.61
2.0	0.70	88.72	87.89	25.99	44.22	99.93	88.81	89.87	$1.94_{\pm 0.39}$	0.310	$0.64_{\pm 0.04}$	0.064	31.67

C Interpolation examples

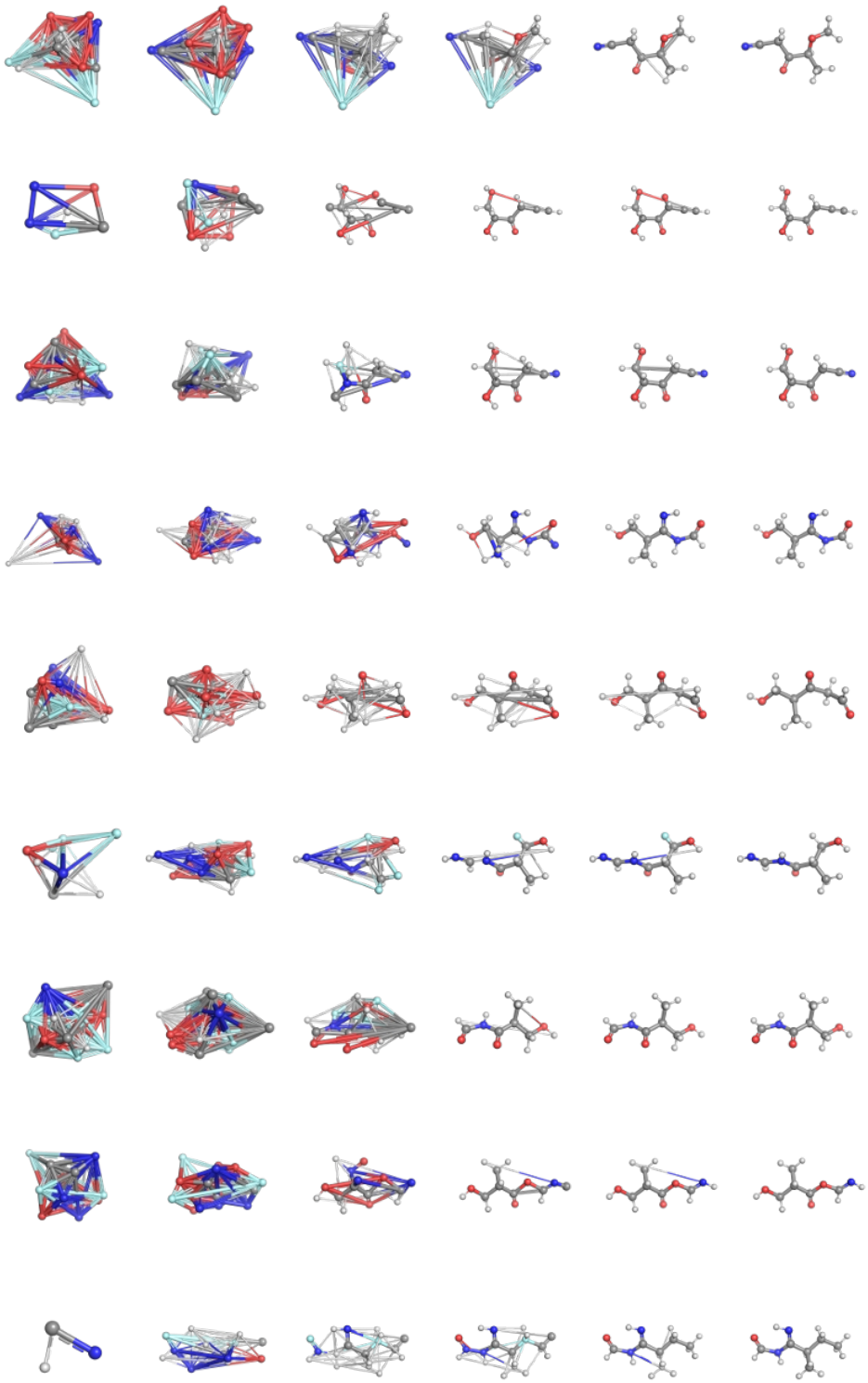


Figure 9: Randomly selected QM9 interpolation trajectories.

D Sampled Molecules

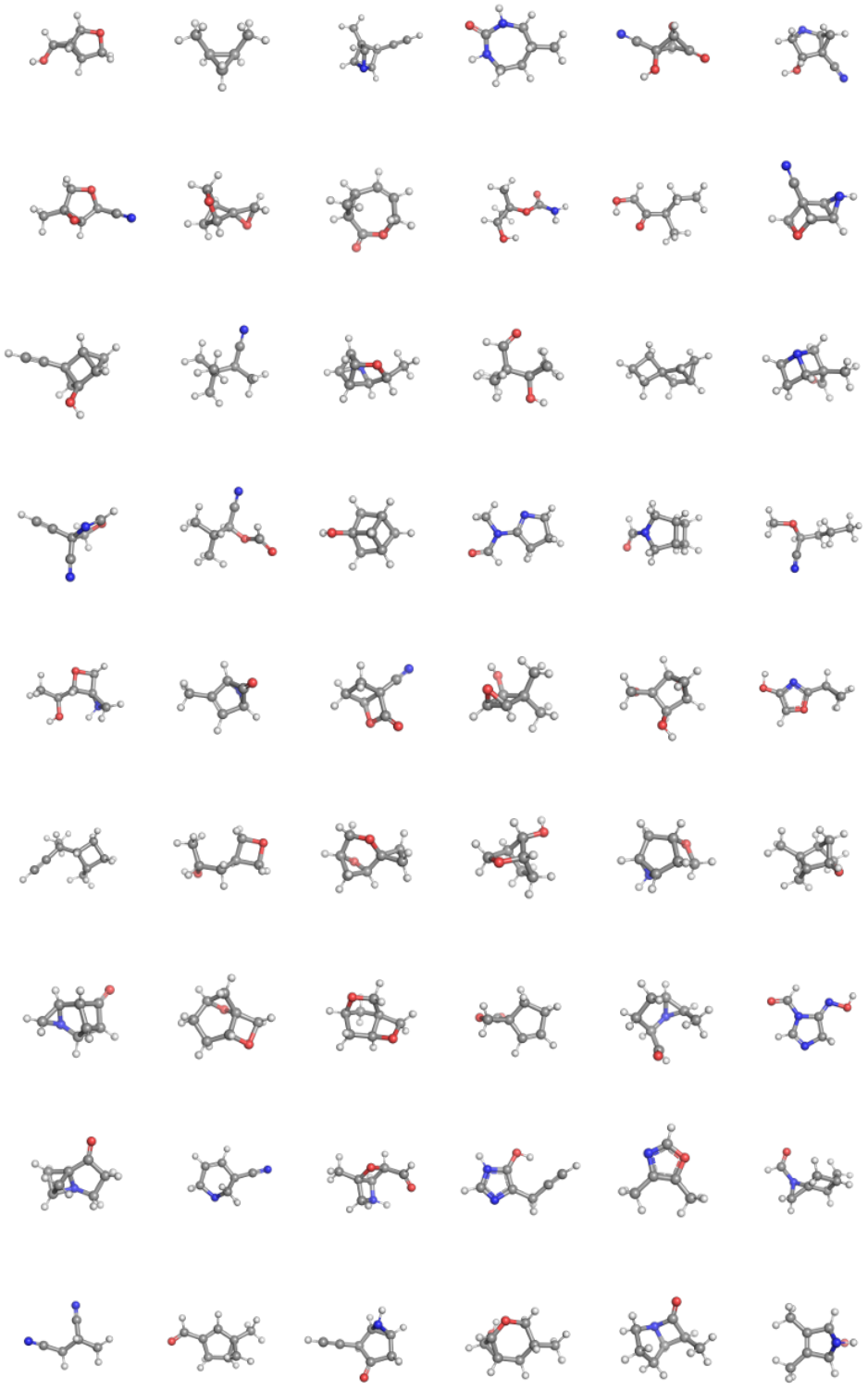


Figure 10: Sampled valid molecules using **Morph** trained on QM9.

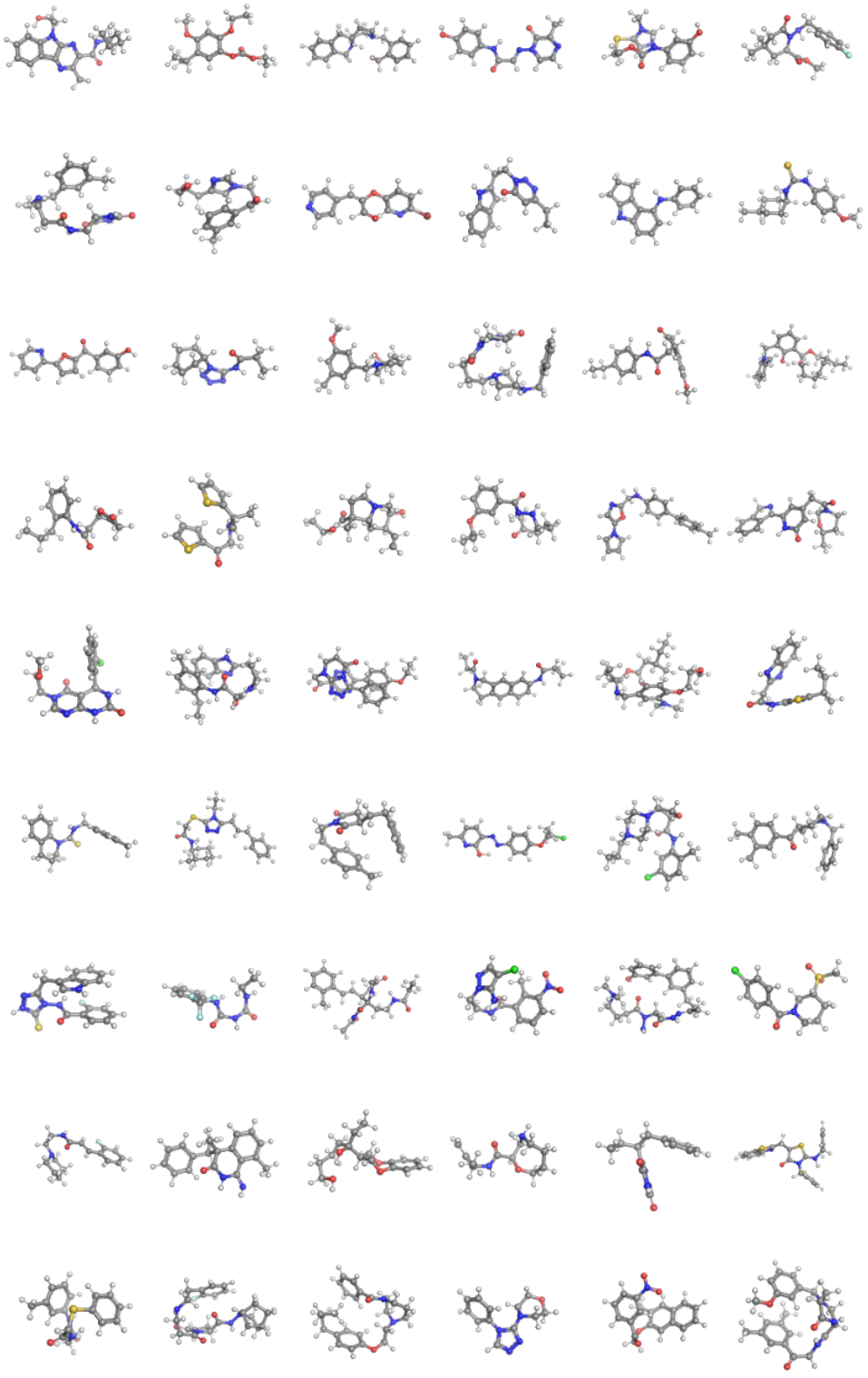


Figure 11: Sampled valid molecules using **Morph** trained on GEOM-Drugs.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

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

   Answer: [Yes]

   Justification: Yes, we tackle all problems mentioned in our abstract and introduction, as can be seen in our results sections 3.1 and 3.2.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We highlight several limitations of our work in Section 5, including higher training cost, more complex loss weight tuning and limitations wrt. the systems modeled. We also show specific samples where our model failed in Sec. D.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.

- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: Our main contributions are not of theoretical nature.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide implementation details and training setups in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [No]

   Justification: All data used in this paper is publicly available. While we do not publish a code repository with this paper, we provide detailed algorithms and descriptions (*e.g.* 2 and 3) that ensure reproducibility of the paper. Full publication of code and model weights is planned after acceptance.

   Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: All training details and validation metrics are provided in the Appendix, *e.g.* Sections A.2, A.3 and A.4, as well as the algorithms 2 and 3.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: We report model errors and plot error bars whenever possible. In some cases, where the main purpose is analysis based on the whole distribution of outputs (*e.g.* Fig. 3 and, by extension Secs. B.8 and B.9), we choose to omit multiple seeds.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

Answer: [Yes]

Justification: We specify both the hardware as well as estimated run times in A.4.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Yes, our paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: We do not explicitly discuss societal impacts within the main text. However, we acknowledge the inherent dual-use nature of generative chemistry models. While a primary positive downstream impact of advancing this methodology is the potential to accelerate the discovery of 'beneficial molecules'—such as novel therapeutics or sustainable materials—the same generative capabilities could theoretically be repurposed by malicious actors to design harmful substances or chemical toxins.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

Justification: Our paper poses no such risk.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All models and datasets used are properly referenced and credited.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

Answer: [N/A]

Justification: We do not publish any assets with this submission.

Guidelines:

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

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

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

Answer: [N/A]

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

Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [No]

Justification: The core method development in our contribution does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.