NEAT: Neighborhood-Guided, Efficient, Autoregressive Set Transformer for 3D Molecular Generation

Daniel Rose[1,2,3*] **Roxane Axel Jacob**[1,2,3*] **Johannes Kirchmair**[1,2] **Thierry Langer**[1,2]
[1]Department of Pharmaceutical Sciences, Faculty of Life Sciences, University of Vienna
[2]Christian Doppler Laboratory for Molecular Informatics in the Biosciences,
Department of Pharmaceutical Sciences, University of Vienna
[3]Vienna Doctoral School of Pharmaceutical, Nutritional and Sport Sciences, University of Vienna

Abstract

Transformer-based autoregressive models offer an efficient alternative to diffusion- and flow-based approaches for generating 3D molecules. One challenge remains: standard transformer architectures require a sequential ordering of tokens, which is not inherently defined for the atoms in a molecule. Prior works have addressed this by using canonical atom orderings. However, these approaches are not permutation invariant w.r.t. atoms and bias next-token prediction towards ordering conventions. We overcome this limitation by introducing a novel neighborhood-guided training strategy. Our model, NEAT (Neighborhood-Guided, Efficient, Autoregressive Set Transformer) treats molecular graphs as sets of atoms and learns an order-agnostic distribution over admissible tokens at the graph boundary, thereby ensuring atom-level permutation invariance. NEAT achieves state-of-the-art generation quality on the QM9 and GEOM-Drugs datasets, enables seamless completion of molecular prefixes, and offers a significant speed advantage over existing baselines.

1 Introduction

Generative modeling offers a promising alternative to virtual screening by enabling the construction of new molecules based on user-defined constraints. Molecular generation can be approached through the generation of (1D) SMILES strings [16, 25], (2D) molecular graphs [5, 47], (3D) atomic point clouds [6, 8, 15, 18, 34, 45, 46, 58, 62], or a combination thereof [19, 38, 54]. While modeling 3D structures is challenging, the additional spatial information is useful for many downstream applications. In this work, we focus on *unconditional* 3D generation, which is a necessary prerequisite to condition the generation process towards compounds with specific biological functions later on [37, 43].

State-of-the-art 3D molecular generation models leverage diffusion and flow matching techniques [9, 11, 18, 49, 50, 58, 62]. At each integration step, these methods rely on quadratically scaling message-passing updates to capture atom interdependencies. Furthermore, most of these models are not inherently adaptive to molecule size, as the number of atoms is typically fixed by the initial nodes sampled from the noise distribution. Trans-dimensional jump-diffusion [4] enables atom insertions during denoising, but managing jump points in the trajectory is challenging. The use of virtual atoms [9, 44] allows dynamic masking of atoms during denoising, yet the maximum number of atoms in the generated molecule remains constrained by the initialization.

Autoregressive models offer a flexible and efficient alternative. They naturally support variable atom counts and require only one encoder call per generated token [29]. For 3D autoregressive molecular generation, there currently exist three lines of work: (1) using graph neural networks (GNNs) in

*Equal contribution, correspondence to: {first.middle.last}@univie.ac.at

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

combination with focus atoms [8, 15, 34]; (2) using standard large language models (LLMs) for generating XYZ, CIF or PDB files [13] or SE(3)-invariant 1D tokens [14, 30, 33]; (3) combining a causal transformer with a diffusion loss to generate continuous tokens (atomic coordinates) [6, 28]. Although current GNN-based methods ensure permutation invariance w.r.t. atom ordering, they generate molecules with low chemical validity. Conversely, sequence-based methods perform well in terms of validity but are not permutation-invariant w.r.t. atom ordering.

In this work, we present the first autoregressive transformer model for 3D molecular generation that respects the atom permutation-invariance of molecular graphs. Working without canonical atom orderings introduces two challenges. First, it precludes the use of sequence embeddings, which are an essential component of standard autoregressive models. Second, it changes the autoregressive training set-up from supervision with a single token per step to supervision with a set of valid tokens: without a canonical atom order, all atoms in the 1-hop neighborhood of the current graph are valid continuations.

Our solution to these challenges is simple yet effective. Given a partial molecular graph, a permutation-invariant set transformer [27] is trained to produce a token distribution that matches the boundary nodes of the subgraph (*i.e.*, all 1-hop neighbors of existing nodes). This training signal, which we refer to as neighborhood guidance, yields a probabilistic next-token distribution without relying on *ad hoc* canonicalization schemes. We combine it with a flow matching loss to model continuous atomic coordinates. Our approach enables training a standard transformer architecture for 3D molecular generation while maintaining permutation invariance to atom ordering.

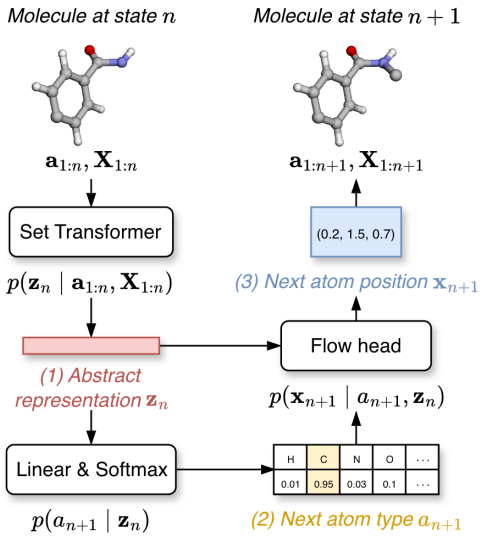


Figure 1: Autoregressive generation of 3D molecules with NEAT. (1) A set transformer produces an abstract representation $\mathbf{z}_n$ of the molecule at state n, given the atom types $\mathbf{a}_{1:n}$ and the coordinates $\mathbf{X}_{1:n}$. (2) The next atom type a_{n+1} is inferred from $\mathbf{z}_n$. (3) The next atom position $\mathbf{x}_{n+1}$ is modeled *via* a flow head, which concludes the molecule update at state $n+1$.

NEAT outperforms prior autoregressive models and matches state-of-the-art flow models on the QM9 and GEOM-Drug benchmarks, while sampling $5\times$ faster than SemlaFlow [21] and $16\times$ faster than FlowMol3 [9]. Its autoregressive design enables molecular prefix completion with no changes to the model or inference procedure—a task diffusion and flow models cannot perform out of the box. By establishing order-agnostic autoregressive generation as a competitive alternative to the SE(3)-equivariant diffusion and flow models that dominate the field, NEAT widens the space of viable approaches for 3D molecule generation. We hope to encourage the community to adopt and refine this direction for drug design tasks such as hit identification and lead optimization.

2 Related works

Autoregressive models generate sequences of tokens by conditioning each token on its predecessors. Formally, given a token sequence $x = (x_1, \ldots, x_N)$, the joint distribution $p(x)$ is factorized as:

$$p(x) = p(x_1) \prod_{n=1}^{N-1} p(x_{n+1} \mid x_{1:n}), \tag{1}$$

where $x_{1:n} = (x_1, \ldots, x_n)$ represents the tokens up to step n. Sampling is performed by initializing with x_1 and generating tokens sequentially until x_N. This is straightforward for sequentially ordered data such as text or images. In the graph domain, the generation order is less obvious, and has been an active focus of research. One way to model graphs as sequences is through canonicalization. However, predefined canonical orderings impose limitations, as the optimal generation order is often context-dependent. To alleviate strict assumptions, You et al. [59] proposed a breadth-first search

node ordering for autoregressive graph generation, while Wang et al. [56] introduced order policies to dynamically assign generation orders based on the current state. Although these approaches reduce reliance on fixed orderings, they do not fully eliminate order dependence. 3D molecules are geometric graphs whose structure is determined by connectivity and geometry. The generation of valid atom positions poses additional constraints on the graph generation process and recent work on this specific task is discussed in the following.

GNN models with focus atoms G-SchNet [15] is an $E(3)$-invariant GNN that uses the concept of focus atoms for generation order. In each generation step, the model selects a focus atom as the center of a 3D grid, chooses the next atom type, and infers its position probability from radial distances to previously placed atoms. G-SphereNet [34] adopts this approach but places atoms relative to the focus atom by generating atom distances, bond angles, and torsion angles with a normalizing flow. The $E(3)$-equivariant Symphony model [8] parameterizes the 3D probability around the focus atom using spherical harmonic projections. While all these models respect the permutation-invariance of atoms, they produce molecules with comparably low chemical validity.

Transformer models with discrete tokenization Transformers can be used for 3D molecule generation when combined with a suitable text format. Flam-Shepherd and Aspuru-Guzik [13] trained an LLM for 3D molecular generation directly on XYZ, CIF, or PDB files. An active line of research are tokenization schemata to allow LLMs more fine-grained control in the generation process. Geo2Seq [30] converts molecular geometries into $SE(3)$-invariant 1D discrete sequences for unconditional modeling, while Frag2Seq [14] uses a related technique for structure-based modeling. Uni-3DAR [33] employs a voxel-based discretization scheme with an octree-based compression technique, yielding sparse sequences.

Transformer models for continuous token generation While standard transformer architectures were initially developed for categorical data, they can be adapted for the generation of continuous values. Li et al. [29] modeled the next-token distribution $p(x_{n+1} \mid x_{1:n})$ without vector quantization using a diffusion-based approach, factorized as:

$$p(x_{n+1} \mid x_{1:n}) = p(x_{n+1} \mid \mathbf{z}_n) \cdot p(\mathbf{z}_n \mid x_{1:n}), \tag{2}$$

where $\mathbf{z}_n \in \mathbb{R}^k$ represents an abstract latent variable for the next token. This factorization enables to use a transformer to model $p(\mathbf{z}_n \mid x_{1:n})$, capturing interdependencies among tokens, followed by a lightweight diffusion-based multilayer perceptron (MLP) to map $\mathbf{z}_n$ to the continuous token space. Their approach enabled high-quality autoregressive image generation with improved inference speed. Cheng et al. [6] applied this idea to the generation of atomic positions, enabling autoregressive 3D molecular generation without spatial discretization. They employed a causal transformer with sequential embeddings derived from the atom order in XYZ coordinate files to predict discrete atom types, and a diffusion model, conditioned on these predictions, to predict the 3D atom positions.

Atom order matters to causal transformer models Cheng et al. [6] acknowledged that a limitation of a causal transformer formulation to 3D molecular generation is its sensitivity to atom ordering. They showed that replacing XYZ orderings with random atom-order augmentation during training substantially degrades model performance. This might seem surprising, since random token-order augmentation has been proposed as a way to address permutation invariance in autoregressive transformers for image generation [60]. In 3D molecular generation, however, this strategy has not yielded satisfactory results, motivating the need for an order-agnostic approach.

Our solution: an order-agnostic set transformer with neighborhood-guidance We propose that a set transformer [27] with bidirectional attention is better suited to autoregressive 3D molecular generation, as it naturally respects the permutation invariance of atom sets. However, this induces a new problem: *Without sequence information, how do we define the next token for model supervision?* Our neighborhood-guidance strategy addresses it by introducing a chemical inductive bias: the model is supervised to generate atoms that form bonds with the current partial molecule, encouraging connectedness throughout generation.

3 Methods

Overview Our architecture consists of three main components: (1) a data-augmentation module for neighborhood-guided training, (2) a set transformer trunk that produces permutation-invariant encodings of atom sets, and (3) a flow matching head that generates continuous atomic coordinates. During training, we split molecules into a *source set*, representing the state of the molecular graph at step n, and a *target set*, representing the set of possible token continuations at step $n+1$. The source set is processed by the set transformer to obtain a learned representation of the current state. This source set representation is then passed through a simple linear layer followed by a softmax to yield a categorical distribution over the next atoms' types. The continuous next atoms' positions are modeled by the flow matching head, conditioned on the source set representation and the next atom type. During training, we use teacher forcing to decouple the prediction of atom types from the prediction of atom positions. Because the expensive source set representation is computed only once per generated token, our model is computationally more efficient than non-autoregressive frameworks that rely entirely on diffusion or flow matching. The training procedure is summarized in Figure 2.

Molecular representation Molecules are represented as undirected graphs $G = (V, E, \mathbf{a}, \mathbf{X})$, where $V = [N]$ is the set of node indices, $E \subseteq \{\{i, j\} : 1 \leq i < j \leq N\}$ is the set of undirected edges, atom types $\mathbf{a} \in \mathcal{A}^N$ are elements of vocabulary $\mathcal{A}$ and positions $\mathbf{X} \in \mathbb{R}^{N\times 3}$ are the Cartesian coordinates after zero-centering the molecule. The vocabulary $\mathcal{A}$ is the set of possible atom types (hydrogen included), and a stop token, which stops the generation process. A subgraph $G' = (V', E')$ of G, denoted by $G' \subseteq G$, is defined as a subset of the nodes $V' \subseteq V$ and edges $E' \subseteq E$. Furthermore, $G' = G[V']$ is a node-induced subgraph of G, if $E' = (V' \times V') \cap E$. A connected graph contains at least one path between any pair of vertices, and a path is defined as a sequence of vertices $(v_0, \ldots, v_n)$ with $(v_i, v_{i+1}) \in E$ for $0 \leq i < n$. The neighborhood of a node index i is the set of all adjacent node indices $\mathcal{N}(i) = \{j \in V : (i, j) \in E\}$. Given a node-induced subgraph $G[V']$, we further define the set of boundary nodes $\partial V' = (\bigcup_{i\in V'} \mathcal{N}(i)) \setminus V'$. Finally, given a node subset V', we denote the corresponding subset of atom types as $\mathbf{a}[V']$ and the subset of atom positions as $\mathbf{X}[V']$.

Autoregressive 3D molecular graph generation In the build-up phase, the model iteratively predicts atom type a_{n+1} and position $\mathbf{x}_{n+1}$, given the current set of atoms and positions at step n. We will use the following compact notation $\mathbf{a}_{1:n} = (a_1, \ldots, a_n)$ to refer to the generated atom types and positions $\mathbf{X}_{1:n} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ at step n. The autoregressive build-up stops when the model predicts a stop token. Edges are then inferred from the N generated atoms *via* a predefined set of rules $E = \mathcal{R}(\mathbf{a}, \mathbf{X})$, which completes inference of the molecular graph G. While edges are inferred from atom types and positions after the build-up phase, the edge information can still be used for teacher guidance in the training phase. In the following, we describe how we use this information for model training.

Neighborhood guidance We propose a sampling strategy that mimics molecular growth during the autoregressive build-up phase. Given a molecular graph G in the training set, we sample a node-induced, connected subgraph $G[V_S]$ of G. We call V_S the *source set*. $G[V_S]$ simulates a molecular fragment with atom types $\mathbf{a}[V_S] = a_{1:n}$ and positions $\mathbf{X}[V_S] = \mathbf{x}_{1:n}$ in the autoregressive build-up at step $n = |V_S|$. The *target set* V_T is the set of valid token continuations at step $n+1$. It is the set of boundary nodes ∂V_S, given V_S, and induces atom types $\mathbf{a}[V_T]$ and positions $\mathbf{X}[V_T]$ for model supervision. The sampling of connected subgraphs is performed dynamically during training. First, we compute the eccentricity $\epsilon(v)$ (the maximum shortest-path distance from v to all other nodes in G) for each node $v \in V$. During runtime, a random seed node v_0 is selected from each molecular graph in the batch. Starting from this seed node, we perform m rounds of 1-hop neighborhood aggregation, where m is a random integer between 1 and $\epsilon(v_0)$.

Algorithm 1 Source-target split augmentation

Input: Graph $G = (V, E)$, β, γ
`# Source set construction`
$v_0 = \text{RandomSeedNode}(V)$
$V_S = \{v_0\}$
$m = \lfloor \beta \cdot \epsilon(v_0) \cdot u_0 \rfloor + 1,\ u_0 \sim \mathcal{U}_{[0,1)}$
for $i = 1$ **to** m **do**
  $\partial V_S \leftarrow (\bigcup_{j\in V_S} \mathcal{N}(j)) \setminus V_S$
  $\partial V_S' \leftarrow \{v \in \partial V_S \mid u_v > \gamma, u_v \sim \mathcal{U}_{(0,1)}\}$
  $V_S \leftarrow V_S \cup \partial V_S'$
end for
`# Target set construction`
$V_T \leftarrow (\bigcup_{j\in V_S} \mathcal{N}(j)) \setminus V_S$
Return: V_S, V_T

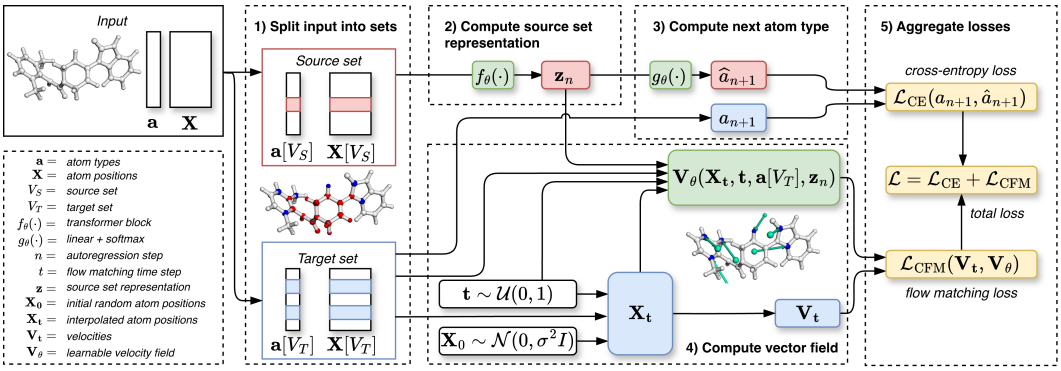


Figure 2: Overview of NEAT's training workflow through neighborhood guidance. The model takes molecular graphs as input. (1) For each training sample, we randomly construct a connected subgraph *via* Algorithm 1. The selected nodes form the source set, and their neighboring boundary nodes define the target set. (2) The source set is encoded using a set transformer, (3) and its representation is used to predict the distribution over the next atom type. (4) For each target-set node, we sample a position from a normal distribution and construct linear interpolation paths between the sampled and true positions. These interpolation paths provide supervision for learning the vector field *via* flow matching. (5) The cross-entropy and flow matching losses are summed to yield the total loss.

To avoid biasing the sampling process toward globular subgraphs, we introduce stochasticity by randomly dropping a subset of the boundary nodes in each aggregation round, controlled by a dropout factor γ. To counteract the resulting bias toward smaller subgraphs, we introduce a scaling parameter β, which adjusts the number of aggregation rounds per graph. By balancing these two parameters, our sampling strategy generates connected subgraphs of varying sizes and shapes, from a single node to the full molecular graph, thereby simulating molecular growth in an autoregressive manner.

The pseudo-code for our sampling strategy is provided in Algorithm 1. Additional information on the impact of γ and β on the sampling strategy and the values used in this work are available in Appendix C.4. In practice, this sampling strategy is efficiently implemented using batched operations and introduces only minimal overhead to the training process, which is dominated by the forward passes through the transformer.

The geometry of generated molecules can deviate from training examples, which can lead to error accumulation at inference. To improve robustness, we inject Gaussian noise $\mathcal{N}(0, \sigma^2 I)$ into a δ-fraction of source-set atoms during training, where σ and δ are model-specific hyperparameters (details in Appendix C.6). We hypothesize that this augmentation trains the model to tolerate inaccuracies from earlier generation stages. We also re-center atom positions in both the source and target sets w.r.t. the noised source-set atoms prior to computing the source-set representation. Both design choices improve NEAT's predictive performance. Their impact is quantified in Appendix C.7.

Transformer trunk The source set representation $f_\theta : (\mathbf{a}_{1:n}, \mathbf{X}_{1:n}) \to \mathbf{z}_n$ at step n is computed by encoding atom types $\mathbf{a}_{1:n}$ with a linear embedding layer and positions $\mathbf{X}_{1:n}$ with Fourier features [52]. Summing these encodings yields the initial node representations. For the transformer backbone f_θ, we adopt the NanoGPT implementation [22] of GPT-2 [39], removing positional embeddings and using bidirectional attention. After transformer updates, additive pooling [61] aggregates node representations into the set-level representation $\mathbf{z}_n$, characterizing the source set at step n. Due to conceptual similarity with the *set attention block* (SAB) by Lee et al. [27], we refer to this setup as *set transformer*. Using a set transformer ensures permutation invariance to the order of atoms in the *source set*.

Next token prediction The set-level representation $\mathbf{z}_n$ is used to model the distribution of possible next tokens, represented as the joint distribution $p(a_{n+1}, \mathbf{x}_{n+1} \mid \mathbf{z}_n)$, which is factorized as follows:

$$p(a_{n+1}, \mathbf{x}_{n+1} \mid \mathbf{z}_n) = p(a_{n+1} \mid \mathbf{z}_n) \cdot p(\mathbf{x}_{n+1} \mid a_{n+1}, \mathbf{z}_n), \tag{3}$$

where $p(a_{n+1} \mid \mathbf{z}_n)$ represents the distribution over the next atom types, and $p(\mathbf{x}_{n+1} \mid a_{n+1}, \mathbf{z}_n)$ models the distribution over the next atom positions given the next atom type. The next atom type

probability $p(a_{n+1} \mid \mathbf{z}_n)$ is modeled as a categorical distribution over the vocabulary of atom types and is approximated with a linear layer followed by a softmax operation $g_\theta : \mathbf{z}_n \to \hat{a}_{n+1}$. For model supervision, we compute the cross-entropy loss between $\hat{a}_{n+1}$ and the observed distribution over target atom types, which is obtained by computing the mean over the one-hot-encoded atom types in the target set $a_{n+1} = \text{mean}(\text{onehot}(\mathbf{a}[V_T]))$. The next atom position is modeled as a conditional distribution given the next atom type. We model this distribution *via* conditional flow matching (CFM, see Appendix B for a more detailed description of the formalism) [1, 31]. During training, we employ teacher forcing to decouple the atom type prediction from the task of positional recovery. For the set of possible next positions $\mathbf{X}_1 = \mathbf{X}[V_T]$ corresponding to the target set V_T, we sample a set of random initial positions $\mathbf{X}_0 \sim p_0$, where $p_0 = \mathcal{N}(0, \sigma^2 I)$. We construct interpolation paths between the positions at the random position at $t = 0$ and the true position at $t = 1$:

$$\mathbf{X}_\mathbf{t} = (1 - \mathbf{t}) \cdot \mathbf{X}_0 + \mathbf{t} \cdot \mathbf{X}_1 \tag{4}$$

with random time steps $\mathbf{t} \in \mathcal{U}(0, 1)^{|V_T|}$, yielding a set of interpolated positions $\mathbf{X}_\mathbf{t}$ with velocities $\mathbf{V}_\mathbf{t} = \mathbf{X}_1 - \mathbf{X}_0$. We then train the conditional vector field $\mathbf{V}_\theta$ through atom-wise regression against the set of CFM path velocities:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{\mathbf{t}, \mathbf{X}_0, \mathbf{X}_1} \left[\frac{1}{|V_T|} \| \mathbf{V}_\theta(\mathbf{X}_\mathbf{t}, \mathbf{t}, \mathbf{a}[V_T], \mathbf{z}_n) - \mathbf{V}_\mathbf{t} \|_F^2 \right] \tag{5}$$

where $\|\cdot\|_F^2$ denotes the squared Frobenius norm. Aggregating atom-wise contributions *via* summation ensures permutation invariance to the order of atoms in the *target set*. The vector field $\mathbf{V}_\theta$ is parametrized using an MLP with adaptive layer normalization (adaLN), as done in [29]. Input to the MLP for velocity regression is the atom-wise summation of positional encoding, atom type embedding, time step embedding, and source set representation. The final loss is obtained by adding the cross-entropy loss and the flow matching loss. In order to reduce the transport cost between the source distribution p_0 and the target distribution p_1, we employ a simple coupling scheme. Given the positions $\mathbf{X}_1$ and $\mathbf{X}_0$, we solve the linear assignment problem on the Euclidean distance matrix $D(\mathbf{X}_1, \mathbf{X}_0)$ and permute the positions in $\mathbf{X}_0$ accordingly. This procedure avoids crossing paths, resulting in improved performance.

Inference The generation process begins by randomly sampling an initial atom type a_1 from the atom type vocabulary $\mathcal{A}$ and drawing its initial position $\mathbf{x}_1 \in \mathbb{R}^3$ from a normal distribution $\mathcal{N}(0, \sigma^2 I)$. At each subsequent autoregressive step, the current atom set $(\mathbf{a}_{1:n}, \mathbf{X}_{1:n})$ is passed through the transformer, producing the set-level representation $\mathbf{z}_n$. Note that the positions are re-centered between autoregressive steps. The distribution over the possible next atom types is predicted from $\mathbf{z}_n$ using a linear layer followed by the softmax function, from which the next atom type a_{n+1} is sampled. A random initial position $\mathbf{x}_{n+1}$ is then sampled from $\mathcal{N}(0, \sigma^2 I)$, and the transport equation is integrated from $t = 0$ to $t = 1$ to compute the next atom position. Details about atom type sampling and the integration scheme are provided in Appendix C.5. This iterative process continues until a stop token is predicted, signaling the end of the molecular generation. Molecular generation is efficiently implemented using batched operations, enabling parallel generation and high computational speed.

4 Experiments

Data We used the QM9 [40] and GEOM-Drugs [3] datasets. QM9 is a collection of small organic molecules composed of up to nine heavy atoms (C, N, O, F), with a total of up to 29 atoms when including hydrogen. We obtained the QM9 dataset from Ramakrishnan et al. [41] and, following standard practice [6, 18], randomly split it into training, validation, and test sets, yielding 100,000 molecules for training, 11,619 for validation, and 12,402 for testing after preprocessing. GEOM-Drugs contains 304,313 drug-like molecules, each with up to 30 conformers. Following Dunn and Koes [9], we loaded the split data from Koes [24]. Like Vignac et al. [54], we retained, for each molecule, the five lowest-energy conformers when available. After preprocessing, the dataset contained 1,172,436 conformers across 243,451 unique molecules for training, 146,394 conformers across 30,430 unique molecules for validation, and 146,788 conformers across 30,432 unique molecules for testing. The mean and median number of conformers per molecule are 4.8 and 5.0, respectively. In the following, we refer to GEOM-Drugs only as GEOM. Detailed information about data processing is provided in the Appendix C.1.

Model training We trained on QM9 and GEOM for 10,000 and 1,000 epochs, respectively, keeping the checkpoint with the lowest validation loss for inference. System specifications, implementation details, hyperparameters of the final models, scaling analyses w.r.t. transformer size, and ablations of key design decisions are provided in Appendix C.2-C.7. Our best-performing models (QM9 and GEOM) both have a 12-layer transformer with 12 heads and hidden dimension 768, and an MLP projector comprising 6 AdaLN blocks with hidden dimension 1536 (165M parameters). This configuration offers a favorable trade-off between generation quality and model size.

Evaluation metrics To evaluate the quality of generated molecules, we adhere to standard practices [6, 8, 18, 62] and report atomic and molecular stability, molecular validity, uniqueness, and novelty. An atom is considered stable if its valence satisfies the octet (heavy atoms) or duet (hydrogen atoms) rule. A molecule is stable if all its atoms are stable. Molecular uniqueness is the percentage of distinct canonical SMILES strings derived from valid molecules (*i.e.*, the percentage of valid and unique molecules). Novelty is the percentage of valid and unique molecules that are not present in the training set. A comparison of the distribution of molecular properties between the generated molecules and the training set can be found in Appendix E.1. Examples of molecules generated by NEAT are shown in Appendix G.1. Failure modes are discussed in Appendix E.3.

Molecule building For NEAT as well as for every other model that does not explicitly generate bonds, molecular validity depends on the algorithm used to construct molecules from 3D clouds of unconnected atoms. We investigate two approaches: (1) the widespread pipeline by Hoogeboom et al. [18] based on bond length lookup tables, (2) the pipeline introduced by Daigavane et al. [8] and adopted by Cheng et al. [6], where a molecule is considered valid if bonds can be assigned using RDKit's MolFromXYZBlock function and rdDetermineBonds module, which uses the xyz2mol package based on the work of Kim and Kim [23]. A notable limitation of the lookup-based pipeline is the absence of aromatic bond handling and a high sensitivity to bond lengths [8], which yields systematically lower scores for valid aromatic structures and thus reduces the informative value of these metrics. We also observed that the output of RDKit's rdDetermineBonds module depends on the RDKit version. Throughout this work, we use RDKit v2025.9.1. A detailed discussion of limitations arising from these bond-assignment-based evaluation pipelines is provided in Appendix D.

Baselines We compare NEAT trained on QM9 to five autoregressive models (G-SchNet [15], G-SphereNet [34], Symphony [8], QUETZAL [6], Geo2Seq [30]). The results are reported in Table 1, with additional comparisons to diffusion-based 3D molecular generation models provided in Appendix E.2. We compare NEAT trained on GEOM to three flow-based models (FlowMol3 [9], SemlaFlow [21], Megalodon-FM [42]) and two autoregressive models (QUETZAL [6], Geo2Seq [30]) and report the results in Table 1. Since the evaluation metrics are sensitive to the procedure used to infer bonds from atomic point clouds, we generated molecules with the source code and model weights of each baseline wherever possible and evaluated them using the same pipeline. For G-SchNet, G-SphereNet, and Symphony, we used the pre-computed XYZ files released by the authors of Symphony [7] and computed metrics with our pipeline. For Geo2Seq, we report the metrics from the original publication.

Unconditional generation performance On QM9, NEAT outperformed autoregressive models in terms of xyz2mol validity and uniqueness. NEAT ranked second to QUETZAL on lookup uniqueness, and third on lookup validity, atom stability, and molecular stability. While Geo2Seq achieved the highest scores for atom stability, molecular stability, and lookup validity, it performed poorly on molecular uniqueness. On the GEOM dataset, NEAT achieved the highest lookup validity and second-best lookup uniqueness and xyz2mol validity, uniqueness and novelty. All models yielded low molecular stability, close to that of the GEOM test set (2.8%). A detailed discussion of this value is provided in Appendix D.

Neighborhood guidance encourages a generation order in which the partial molecule remains connected, a prerequisite for molecular validity. To verify this, we examined the final molecular graphs by iterating over atoms in generation order and checking whether each new atom n was connected to the subgraph induced by atoms $1, \ldots, n-1$. This condition held for every generated molecule, confirming that the generation process follows the intended neighborhood-guided behavior.

Table 1: Benchmarking of NEAT against 3D molecular generators trained on QM9 and GEOM. Metrics are reported as percentages. The first row reports results on the QM9 test set (12,402 molecules). The eighth row reports results on the GEOM test set (146,788 conformers, 30,432 unique molecules). Other rows report results computed from 10,000 generated molecules. Entries for which generated molecules could be retrieved and re-evaluated with our pipeline are marked with a single star (*). Entries for which the source code and model weights were available to generate molecules are marked with two stars (**). For these models and for NEAT, we report the mean ± one standard deviation over three runs. Otherwise, values are taken directly from original publications.

Model	atom stable	mol stable	lookup valid	lookup unique	xyz2mol valid	xyz2mol unique	xyz2mol novel
QM9	99.3	94.8	97.5	97.5	100.0	100.0	99.8
G-SchNet*	93.7	63.6	80.1	74.7	73.0	70.6	**55.1**
G-SphereNet*	67.8	14.0	16.4	3.8	37.7	7.2	6.7
Symphony*	90.8	43.9	68.1	66.5	77.0	75.2	41.3
Geo2Seq	**98.9**	**93.2**	**97.1**	81.7	n.a.	n.a.	n.a.
QUETZAL**	98.3 ± 0.1	87.5 ± 0.1	94.6 ± 0.1	**90.1** ± 0.2	93.7 ± 0.3	89.1 ± 0.2	24.7 ± 0.3
NEAT (ours)	97.8 ± 0.1	83.1 ± 0.1	93.3 ± 0.1	88.8 ± 0.3	**95.2** ± 0.2	**91.6** ± 0.3	53.5 ± 0.6
GEOM	86.2	2.8	96.6	96.6	93.5	93.5	93.5
FlowMol3**	**88.8** ± 0.1	**5.0** ± 0.8	96.5 ± 0.6	96.5 ± 0.5	**95.8** ± 0.7	**95.7** ± 0.7	**95.7** ± 0.7
SemlaFlow**	84.1 ± 0.1	2.2 ± 0.1	92.9 ± 0.1	92.9 ± 0.1	86.6 ± 0.5	86.5 ± 0.5	86.5 ± 0.5
Megalodon-FM**	81.5 ± 0.1	1.8 ± 0.1	97.9 ± 0.2	**97.8** ± 0.2	88.2 ± 0.2	88.1 ± 0.2	88.1 ± 0.2
Geo2Seq	82.5	n.a.	96.1	n.a.	n.a.	n.a.	n.a.
QUETZAL**	86.7 ± 0.1	2.5 ± 0.1	95.7 ± 0.1	95.4 ± 0.1	90.1 ± 0.1	89.8 ± 0.1	89.8 ± 0.1
NEAT (ours)	81.5 ± 0.1	0.6 ± 0.1	**99.7** ± 0.1	96.9 ± 0.1	93.9 ± 0.2	91.6 ± 0.2	91.6 ± 0.2

Table 2: Prefix completion performance of NEAT, QUETZAL, and FlowMol3 trained on GEOM. Metrics are reported as percentages. For each of the 100 prefixes, we generate 1,000 molecules and compute average metric values. We then report the mean ± one standard deviation across the 100 prefix-level averages.

Model	atom stable	mol stable	lookup valid	lookup unique	xyz2mol valid	xyz2mol unique	xyz2mol novel
FlowMol3 (inpainted)	76.3 ± 6.5	0.4 ± 1.3	44.5 ± 16.3	36.3 ± 20.1	30.2 ± 7.8	29.4 ± 7.3	29.4 ± 7.3
QUETZAL	69.0 ± 1.8	0.1 ± 0.1	27.8 ± 6.8	13.6 ± 4.1	28.1 ± 4.7	17.7 ± 2.8	17.7 ± 2.8
NEAT (ours)	**77.0** ± 1.5	**0.5** ± 0.3	**97.8** ± 1.8	**82.2** ± 3.0	**92.7** ± 2.3	**77.9** ± 3.0	**77.9** ± 3.0

Prefix completion We demonstrate the utility of our approach by comparing NEAT to QUETZAL on the task of designing 3D molecules around a given scaffold, referred to as *prefix completion*. This task has practical relevance in drug discovery, for example for lead optimization. Both models are transformer-based autoregressive 3D molecule generators, but differ in one key aspect. QUETZAL uses a causal transformer with sequential encodings and is trained to generate molecules based on the canonical ordering of atoms in XYZ files, whereas NEAT uses a set transformer with bidirectional attention and is trained using neighborhood guidance. For QUETZAL to generate a valid molecule, the prefix must align with the first n atoms of the fully generated molecule after canonicalization. This requirement cannot be satisfied without prior knowledge of the final molecule.

We tasked both models (trained on GEOM and run with default settings) with generating 1,000 molecules from 100 incomplete prefixes extracted from GEOM, resulting in a total of 100,000 completions. The results are summarized in Table 2. The prefixes represent diverse ring systems with varying substitution patterns. 2D plots of all prefixes are provided in Appendix G.2. For each prefix, we generated 3D molecular coordinates using RDKit's implementation of the Universal Force Field (UFF), initially including hydrogen atoms at the substitution positions. These hydrogen atoms were subsequently removed to enable the model to continue molecular generation at the designated sites, and each prefix was randomly rotated for every generated sample.

Unlike QUETZAL, NEAT successfully generates valid molecules from arbitrary prefixes by learning the structural features of valid molecules independent of atom ordering conventions. Compared with unconditional generation (Table 1), prefix completion with NEAT (Table 2) largely preserves stability and validity, while reducing diversity. The reduced diversity is expected, since fixing a prefix

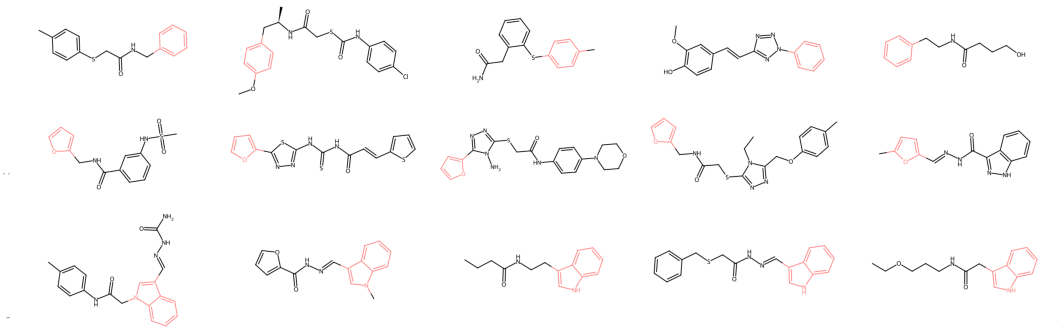


Figure 3: Randomly selected examples of molecules generated from three different prefixes: para-substituted benzene, 2,5-substituted furan, and 1,3-substituted indole.

constrains the space of possible molecules. The moderate decrease in atom and molecule stability can be explained by the fact that most benchmarked prefixes contain aromatic rings (Appendix Figures 19-21). These stability metrics fail for aromatic compounds because the underlying pipeline does not account for the length of aromatic bonds (see Appendix D for details). The small drop in the validity metrics may be due to the inclusion of rare prefixes, which are harder for the model to work with because it has seen fewer examples of these during training.

We also compared NEAT's prefix completion capabilities to the ones of non-autoregressive models. To the best of our knowledge, no published diffusion- or flow-based baseline evaluates prefix-conditioned inpainting for unconditional 3D molecular generation on GEOM. We therefore implemented this setting in our strongest baseline, FlowMol3, by modifying only its inference procedure, without retraining. Details are provided in Appendix F. Our results show that inference-time inpainting with FlowMol3 is feasible, but substantially degrades validity and related quality metrics. A scaffold- or prefix-conditioned transport-based model trained specifically for this task could potentially reduce this gap, but would require substantial changes to the training objective, conditioning mechanism, and data construction. By comparison, NEAT supports prefix completion directly, without retraining or model modifications, highlighting its practical advantage.

Runtime We analyze NEAT's computational complexity starting with the worst-case scenario, where transformer calls dominate runtime. Its worst-case complexity is $\mathcal{O}(BN^3)$, with N as the number of atoms per molecule in a batch of B molecules. Diffusion and flow models have complexity $\mathcal{O}(BTN^2)$, where T is the number of integration steps. For drug-like molecules, N is typically smaller than T, giving NEAT an advantage even in the worst-case scenario. In practice, NEAT's complexity is lower than that of the worst-case scenario. During training, NEAT makes only one transformer call ($\mathcal{O}(BN^2)$) and one MLP call ($\mathcal{O}(BN)$) per forward pass. With batch size B constant, the transformer accounts for approximately 90% of the runtime. During inference, the dominant workload shifts to the ODE integration ($\mathcal{O}(BT)$). Generating N atoms scales as $\mathcal{O}(BTN)$. Another positive effect is that B decreases as molecules complete, while N grows linearly. Initially, transformer runtime increases but declines after generating 25 atoms. We illustrate these observations and report the total runtime in Figure 4.

NEAT operates 34 times faster than Megalodon-FM, 16 times faster than FlowMol3, and 5 times faster than SemlaFlow and QUETZAL. When the baselines are run with fewer integration steps than recommended by their authors to match NEAT's runtime, their generation quality falls below that of NEAT (See Appendix E.7), further highlighting NEAT's favorable quality–runtime trade-off. Because batched inference depends on molecule size, we report size statistics for completeness. The average number of heavy atoms was similar across models: 22 ± 5 for FlowMol3, 21 ± 6 for Megalodon, 21 ± 5 for NEAT, 24 ± 5 for QUETZAL, and 23 ± 5 for SemlaFlow. The largest molecules, including hydrogen atoms, contained 154 atoms for FlowMol3, 54 for Megalodon, 94 for NEAT, 94 for QUETZAL, and 72 for SemlaFlow.

Limitations NEAT inherits several standard limitations of autoregressive architectures. Specifically, error propagation may degrade sampling quality for larger molecules, and pooling node embeddings could cause oversquashing, leading to information loss. To address this, we analyzed the correlation

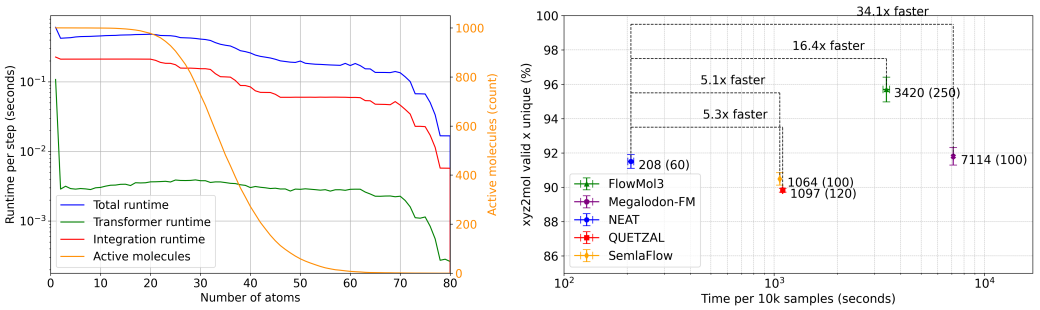


Figure 4: **Left**: Runtime per step when generating 1000 molecules, averaged over 10 runs. **Right**: Comparison of xyz2mol validity x uniqueness vs. runtime between NEAT and models trained on GEOM. The number attached to each data point corresponds to the runtime in seconds followed by the number of integration steps (chosen following the recommendations of the authors) in parenthesis. Values are averaged over three runs. Error bars correspond to one standard deviation. Batch size: largest to fit on GPU (FlowMol3: 500; SemlaFlow: 10,000; Megalodon-FM: 50; QUETZAL: 2,000; NEAT: 10,000).

between molecular size and validity (Appendix E.6) and found no evidence of size-dependent degradation. However, since NEAT does not extrapolate beyond the maximum atom count seen during training (Appendix E.1), this conclusion is restricted to drug-like molecules. Additionally, NEAT's neighborhood-guidance mechanism relies on edge information derived from the molecular graph. This dependency may limit its applicability in domains where bond assignment is ambiguous, such as inorganic crystal structures or metal-organic compounds. While spatial information, such as radius-based graphs, could serve as an alternative source of neighborhood information, evaluating the effectiveness of NEAT in such generalized settings is beyond the scope of this study and remains an avenue for future work. Finally, the transformer architecture employed in this study is a standard transformer and does not enforce SE(3)-equivariance. Despite this, our results demonstrate that combining this simple architecture with random rotation augmentation during training can achieve state-of-the-art generation performance. To quantify the impact of not enforcing SE(3)-equivariance, we conducted an invariance-sensitivity test (Appendix E.4) and observed only minor variance in predictions. While we conclude that the lack of architectural SE(3)-equivariance does not significantly impair model performance, orientation dependence in predictions cannot be entirely ruled out. Integrating NEAT with an SE(3)-equivariant transformer represents an interesting direction for future research.

5 Conclusion

We introduce a new method for training an autoregressive set transformer with bidirectional attention for 3D molecular generation using neighborhood guidance. Instead of sequential positional encodings, we rely on a subgraph sampling schema for model supervision. Treating the input as a permutation-invariant set removes the need for sequence canonicalization and makes the model well-suited for tasks like molecular prefix completion. On QM9 and GEOM, our approach achieves state-of-the-art performance, while offering permutation-invariance w.r.t. atoms, prefix completion, and fast sampling. We believe this provides a strong foundation for future work in conditional 3D molecular generation. Source code and model weights are available at: `https://github.com/molinfo-vienna/NEAT`.

Acknowledgments and Disclosure of Funding

The authors thank Leo Gaskin for testing the code and Steffen Hirte for proofreading the manuscript. The financial support received for the Christian Doppler Laboratory for Molecular Informatics in the Biosciences by the Austrian Federal Ministry of Labour and Economy, the National Foundation for Research, Technology and Development, the Christian Doppler Research Association, Boehringer-Ingelheim RCV GmbH & Co KG and BASF SE is gratefully acknowledged. There are no conflicts of interest to declare.

Author contributions

Conceptualization: DR, RAJ. Methodology: DR, RAJ. Software: DR, RAJ. Data curation: DR, RAJ. Formal analysis: DR, RAJ. Investigation: DR, RAJ. Visualization: DR, RAJ. Writing (original draft): DR, RAJ. Writing (review and editing): DR, RAJ, JK, TL. Supervision: JK, TL. Funding acquisition: JK, TL.

References

[1] M. S. Albergo and E. Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *Proceedings of the 11th International Conference on Learning Representations*. Curran Associates, Inc., 2023.

[2] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, and S. Chintala. PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, volume 2. ACM, 2024. doi: 10.1145/3620665.3640366.

[3] S. Axelrod and R. Gómez-Bombarelli. GEOM, energy-annotated molecular conformations for property prediction and molecular generation. *Sci. Data*, 9(185), 2022. doi: 10.1038/s41597-022-01288-4.

[4] A. Campbell, W. Harvey, C. Weilbach, V. De Bortoli, T. Rainforth, and A. Doucet. Trans-dimensional generative modeling via jump diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, pages 42217–42257. Curran Associates, Inc., 2023.

[5] N. D. Cao and T. Kipf. MolGAN: An implicit generative model for small molecular graphs, 2022. URL https://arxiv.org/abs/1805.11973.

[6] A. H. Cheng, C. Sun, and A. Aspuru-Guzik. Scalable autoregressive 3D molecule generation, 2025. URL https://arxiv.org/abs/2505.13791.

[7] A. Daigavane, S. E. Kim, M. Geiger, and T. Smidt. Sampled 3D molecular structures from generative models (symphony), 2023. URL https://figshare.com/s/a17ccface17f0c22f15a?file=42504807. Accessed 2025-12-15.

[8] A. Daigavane, S. E. Kim, M. Geiger, and T. Smidt. Symphony: Symmetry-equivariant point-centered spherical harmonics for 3D molecule generation. In *Proceedings of the 12th International Conference on Learning Representations*, pages 33975–34002. Curran Associates, Inc., 2024.

[9] I. Dunn and D. R. Koes. FlowMol3: Flow matching for 3D de novo small-molecule generation. *Digit. Discov.*, 2026. doi: 10.1039/d5dd00363f.

[10] W. Falcon and the PyTorch Lightning team. Pytorch lightning, 2025. URL https://github.com/Lightning-AI/lightning.

[11] S. Feng, Y. Ni, L. yan, Z.-M. Ma, W.-Y. Ma, and Y. Lan. UniGEM: A unified approach to generation and property prediction for molecules. In *Proceedings of the 13th International Conference on Learning Representations*, pages 12824–12849. Curran Associates, Inc., 2025.

[12] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric, 2019. URL https://arxiv.org/abs/1903.02428.

[13] D. Flam-Shepherd and A. Aspuru-Guzik. Language models can generate molecules, materials, and protein binding sites directly in three dimensions as XYZ, CIF, and PDB files, 2023. URL https://arxiv.org/abs/2305.05708.

[14] C. Fu, X. Li, B. Olson, H. Ji, and S. Ji. Fragment and geometry aware tokenization of molecules for structure-based drug design using language models. In *Proceedings of the 13th International Conference on Learning Representations*, pages 64203–64227. Curran Associates, Inc., 2025.

[15] N. W. A. Gebauer, M. Gastegger, and K. T. Schütt. Symmetry-adapted generation of 3D point sets for the targeted discovery of molecules. In *Advances in Neural Information Processing Systems*, volume 32, pages 7566–7578. Curran Associates, Inc., 2019.

[16] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4:268–276, 2018. doi: 10.1021/acscentsci.7b00572.

[17] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 9, pages 6840–6851. Curran Associates, Inc., 2020.

[18] E. Hoogeboom, V. G. Satorras, C. Vignac, and M. Welling. Equivariant diffusion for molecule generation in 3D. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 8867–8887. PMLR, 2022.

[19] C. Hua, S. Luan, M. Xu, R. Ying, J. Fu, S. Ermon, and D. Precup. MUDiff: Unified diffusion for complete molecule generation. In *Proceedings of the 2nd Learning on Graphs Conference*, volume 231. PMLR, 2023.

[20] J. D. Hunter. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.*, 9(3):90–95, 2007.

[21] R. Irwin, A. Tibo, J. P. Janet, and S. Olsson. SemlaFlow - efficient 3D molecular generation with latent attention and equivariant flow matching. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258, pages 3772–3780. PMLR, 2025.

[22] A. Karpathy. NanoGPT, 2025. URL https://github.com/karpathy/nanoGPT. Accessed 2025-11-15.

[23] Y. Kim and W. Y. Kim. Universal structure conversion method for organic molecules: From atomic connectivity to three-dimensional geometry. *Bull. Korean Chem. Soc.*, 36:1769–1777, 2015. doi: 10.1002/bkcs.10334.

[24] D. R. Koes, 2024. URL https://bits.csb.pitt.edu/files/geom_raw/. Accessed 2026-01-08.

[25] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1945–1954. PMLR, 2017.

[26] G. Landrum. RDKit: Open-source cheminformatics, 2025. URL https://www.rdkit.org.

[27] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3744–3753. PMLR, 2019.

[28] H. Li, W. Du, Y. Li, H. Guo, and S. Liu. InertialAR: Autoregressive 3D molecule generation with inertial frames. In *Proceedings of the 43rd International Conference on Machine Learning*. PMLR, 2026.

[29] T. Li, Y. Tian, H. Li, M. Deng, and K. He. Autoregressive image generation without vector quantization. In *Advances in Neural Information Processing Systems*, volume 37, pages 56424–56445. Curran Associates, Inc., 2024. doi: 10.52202/079017-1797.

[30] X. Li, L. Wang, Y. Luo, C. Edwards, S. Gui, Y. Lin, H. Ji, and S. Ji. Geometry informed tokenization of molecules for language model generation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 36096–36128. PMLR, 2025.

[31] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *Proceedings of the 11th International Conference on Learning Representations*. Curran Associates, Inc., 2023.

[32] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations*. Curran Associates, Inc., 2019.

[33] S. Lu, H. Lin, L. Yao, Z. Gao, X. Ji, Y. Liang, W. E, L. Zhang, and G. Ke. Unified cross-scale 3D generation and understanding via autoregressive modeling, 2025. URL https://arxiv.org/abs/2503.16278.

[34] Y. Luo and S. Ji. An autoregressive flow model for 3D molecular geometry generation from scratch. In *Proceedings of the 10th International Conference on Learning Representations*. Curran Associates, Inc., 2022.

[35] N. Ma, M. Goldstein, M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, and S. Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *Proceedings of the 18th European Conference on Computer Vision*, pages 23–40. Springer, 2024.

[36] A. Morehead and J. Cheng. Geometry-complete diffusion for 3D molecule generation and optimization. *Commun. Chem.*, 7(150), 2024. doi: 10.1038/s42004-024-01233-z.

[37] X. Peng, S. Luo, J. Guan, Q. Xie, J. Peng, and J. Ma. Pocket2Mol: Efficient molecular sampling based on 3D protein pockets. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 17644–17655. PMLR, 2022.

[38] X. Peng, J. Guan, Q. Liu, and J. Ma. MolDiff: Addressing the atom-bond inconsistency problem in 3D molecule diffusion generation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 27611–27629. PMLR, 2023.

[39] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 2019.

[40] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. V. Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(140022), 2014. doi: 10.1038/sdata.2014.22.

[41] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. V. Lilienfeld. "quantum chemistry structures and properties of 134 kilo molecules" figshare repository, 2019. URL https://springernature.figshare.com/collections/Quantum_chemistry_structures_and_properties_of_134_kilo_molecules/978904. Accessed 2025-12-16.

[42] D. Reidenbach, F. Nikitin, O. Isayev, and S. Paliwal. Applications of modular co-design for de novo 3D molecule generation. *Digit. Discov.*, 5:754–768, 2026.

[43] A. Schneuing, C. Harris, Y. Du, K. Didi, A. Jamasb, I. Igashov, W. Du, C. Gomes, T. L. Blundell, P. Lio, et al. Structure-based drug design with equivariant diffusion models. *Nat. Comput. Sci.*, 4:899–909, 2024. doi: 10.1038/s43588-024-00737-x.

[44] A. Schneuing, I. Igashov, A. W. Dobbelstein, T. Castiglione, M. Bronstein, and B. Correia. Multi-domain distribution learning for de novo drug design. In *Proceedings of the 13th International Conference on Learning Representations*, pages 84997–85033. Curran Associates, Inc., 2025.

[45] G. N. C. Simm, R. Pinsler, and J. M. Hernández-Lobato. Reinforcement learning for molecular design guided by quantum mechanics. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020.

[46] G. N. C. Simm, R. Pinsler, G. Csányi, and J. M. Hernández-Lobato. Symmetry-aware actor-critic for 3D molecular design. In *Proceedings of the 9th International Conference on Learning Representations*. Curran Associates, Inc., 2021.

[47] M. Simonovsky and N. Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. In *Proceedings of the 27th International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018. doi: 10.1007/978-3-030-01418-6_41.

[48] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *Proceedings of the 9th International Conference on Learning Representations*. Curran Associates, Inc., 2021.

[49] Y. Song, J. Gong, M. Xu, Z. Cao, Y. Lan, S. Ermon, H. Zhou, and W.-Y. Ma. Equivariant flow matching with hybrid probability transport for 3D molecule generation. In *Advances in Neural Information Processing Systems*, volume 36, pages 549–568. Curran Associates, Inc., 2023.

[50] Y. Song, J. Gong, H. Zhou, M. Zheng, J. Liu, and W.-Y. Ma. Unified generative modeling of 3D molecules via bayesian flow networks. In *Proceedings of the 12th International Conference on Learning Representations*, pages 47851–47869. Curran Associates, Inc., 2024.

[51] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568(127063), 2024. doi: 10.1016/j.neucom.2023.127063.

[52] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc., 2020.

[53] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport, 2024. URL https://arxiv.org/abs/2302.00482.

[54] C. Vignac, N. Osman, L. Toni, and P. Frossard. MiDi: Mixed graph and 3D denoising diffusion for molecule generation. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 560–576. Springer, 2023. doi: 10.1007/978-3-031-43415-0_33.

[55] Y. Wang, J. Lu, N. Jaitly, J. Susskind, and M. A. Bautista. SimpleFold: Folding proteins is simpler than you think, 2025. URL https://arxiv.org/abs/2509.18480.

[56] Z. Wang, J. Shi, N. Heess, A. Gretton, and M. Titsias. Learning-order autoregressive models with application to molecular graph generation. In *Proceedings of the 42nd International Conference on Machine Learning*. PMLR, 2025.

[57] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.*, 9(2):513–530, 2018.

[58] M. Xu, A. S. Powers, R. O. Dror, S. Ermon, and J. Leskovec. Geometric latent diffusion models for 3D molecule generation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 38592–38610. PMLR, 2023.

[59] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 5708–5717. PMLR, 2018.

[60] Q. Yu, J. He, X. Deng, X. Shen, and L.-C. Chen. Randomized autoregressive visual generation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18431–18441. IEEE Computer Society Conference Publishing Services (CPS), 2025.

[61] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[62] L. Zhang, K. Ashouritaklimi, Y. W. Teh, and R. Cornish. SymDiff: Equivariant diffusion via stochastic symmetrisation. In *Proceedings of the 13th International Conference on Learning Representations*, pages 47651–47673. Curran Associates, Inc., 2025.

A Broader societal impact

This paper aims to advance the field of machine learning in general, and of *de novo* molecular design in particular. While we recognize the potential misuse of our approach to generate harmful compounds, we emphasize that there are numerous beneficial applications, such as the development of new pharmaceuticals and safer agrochemicals. Importantly, the generation of molecules is only one step in a complex pipeline, requiring extensive validation, synthesis, and testing before any practical use, which mitigates immediate risks. Furthermore, the ethical deployment of such models should be guided by responsible practices, including rigorous oversight and adherence to safety standards. We do not foresee ethical concerns beyond those generally associated with generative models, but we stress the importance of responsible use to maximize societal benefit while minimizing risks.

B Flow matching

Flow matching [1, 31] offers a conceptually simple alternative to diffusion models and often achieves faster inference due to the need for fewer integration steps during the inference process. It is a simulation-free training framework for generative modeling, where a time-dependent velocity field ψ_t is learned to couple a tractable source distribution p_0 with a target distribution p_1. Typically, p_0 is chosen as a standard normal distribution, $p_0 = \mathcal{N}(0, I)$, and the transport is defined over a time interval $t \in [0, 1]$. The transformation of p_0 into intermediate distributions p_t is achieved *via* the push-forward operation $\#$, resulting in a trajectory of probability distributions $p_t(\mathbf{x}_t)$ at time t. The flow ψ_t is modeled using a parameterized function $\mathbf{v}_\theta(\mathbf{x}_t, t)$, which is trained to approximate the velocity field. Sampling from the target distribution p_1 is performed by drawing samples $\mathbf{x}_0 \sim p_0$ and integrating the ordinary differential equation (ODE) $\mathrm{d}\mathbf{x}_t = \mathbf{v}_\theta(\mathbf{x}_t, t)$.

Lipman et al. [31] introduced simulation-free training by constructing conditional probability paths. A widely used choice for these paths is the linear interpolant [1], defined as $\mathbf{x}_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$, with the conditional target velocity $\mathbf{v}_t = \mathbf{x}_1 - \mathbf{x}_0$. The training objective, known as conditional flow matching (CFM), minimizes the l_2 regression loss: $\mathcal{L}_{\text{CFM}} = \mathbb{E}[\|\mathbf{v}_\theta(\mathbf{x}_t, t) - \mathbf{v}_t\|]$. Flow-based molecular generation models include for instance EquiFM [49], SemlaFlow [21], Megalodon-FM [42], and FlowMol3 [9]. Independent coupling of p_0 and p_1 can result in high transport costs and intersecting conditional paths. These issues can be alleviated by employing optimal transport couplings between the source and target distributions, thereby improving convergence and enhancing flow performance [53]. In molecular generation, this has for example been approached through rigid alignment of noisy and true coordinates, combined with solving the assignment problem to minimize transport costs [9].

C Implementation details

C.1 Data processing

The QM9 dataset was retrieved from the figshare repository of Ramakrishnan et al. [41], licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 (CC BY-NC-SA 4.0) international license. The repository comprises 133,885 molecules and additionally provides a list of 3,054 molecule IDs for which OpenBabel failed to infer correct bonds from the XYZ geometry files. Following prior work [6, 18], we excluded these molecules. Because the dataset is distributed as XYZ files without explicit bond annotations, and our neighborhood-guidance strategy requires bond information, we reconstructed bonds for each molecule using RDKit (MolFromXYZBlock together with rdDetermineBonds). Bond assignment failed for 6810 molecules, yielding a processed dataset of 124,021 small molecules. We randomly split these into 100,000/11,619/12,402 molecules for the training/validation/test sets, respectively. While some prior works on 3D molecular generation obtained QM9 *via* the MoleculeNet project [57] rather than the original repository [41], we decided against using the MoleculeNet files. Although they include bond information, the procedure used to determine bonds is not documented, and a non-negligible fraction of molecules in that distribution are charged. This conflicts with the original QM9 specification [40], which describes the dataset as comprising *neutral* molecules with up to nine heavy atoms (C, N, O, F), excluding hydrogen atoms.

The GEOM-Drugs dataset was obtained from the FlowMol project's data repository [9] as distributed by Koes [24]. This dataset is licensed under a Creative Commons Attribution 4.0 (CC BY

4.0) international license. The files provide a predefined train/validation/test split totaling 304,313 molecules, which we adopted unchanged. We removed 35 molecules that failed RDKit sanitization. For molecules comprising multiple disconnected fragments, we retained only the largest fragment by atom count. Following Vignac et al. [54], we used up to five low-energy conformations per molecule. After preprocessing, the dataset contained 243,451/30,430/30,432 molecules and 1,172,436/146,394/146,788 conformations in the train/validation/test splits, respectively.

C.2 System and software specification

Training was performed on an AlmaLinux 9.7 system with an NVIDIA RTX Pro 6000 Blackwell GPU with 96 GB of GDDR7 VRAM and an AMD EPYC 9555 64-Core Processor. Inference and data analysis was performed on a Rocky Linux 9.7 system with an NVIDIA GeForce RTX 4090 GPU with 24 GB of GDDR6X VRAM and an AMD Ryzen 9 7950X 16-Core Processor. We use CUDA 13.0 for GPU acceleration. Our model is implemented in Python (v3.11.13) with PyTorch (v2.9.0) [2], PyTorch Geometric (v2.7.0) [12], and Pytorch Lightning (v2.5.5) [10]. Model training was monitored with Tensorboard (v2.20.0). For chemical data processing we employed RDKit (v2025.9.1) [26]. For visualization, we used Py3Dmol (v2.5.3) and matplotlib (v3.10.7) [20].

C.3 Model implementation

The basis of our implementation is the NanoGPT implementation of Karpathy [22], which provides a concise implementation of the GPT-2 [39] model. For positional encoding of the Cartesian coordinates, we employ Fourier features [52] as implemented in Wang et al. [55]. We experimented with qk-normalization and rotary position embeddings [51], but did not obtain improved results. We found additive pooling to be the most effective pooling strategy, but also tried mean and attention pooling. We also tried local attention, defined by a radius graph, instead of full attention over all atom nodes, which led to performance degradation. Inclusion of charged states into the atom type vocabulary did not lead to model improvement. Since our model architecture is not $SE(3)$-equivariant, we augment data samples by random rotation, which stabilizes the training. The flow head is the same adaptive layer-norm MLP as used in Li et al. [29], which was also used in the QUETZAL model [6] for implementation of the diffusion head.

For training the flow matching head, we sample random time steps analog to the SimpleFold model [55] from the following log-normal distribution:

$$p(t) = 0.02\mathcal{U}(0,1) + 0.98\text{LN}(0.8, 1.7) \tag{6}$$

with

$$\text{LN}(t; m, s) = \frac{1}{t(1-t)s\sqrt{2\pi}} \exp \frac{-(\text{logit}(t) - m)^2}{2s^2} \tag{7}$$

where we set $m = 0.8$ and $s = 1.7$. With these settings, time steps are sampled more densely around $t = 1$. We found that this sampling schema slightly improved training above time step sampling from a uniform distribution. For each conditional path, we sampled four random time steps, following Li et al. [29]. Using endpoint parametrization for construction of the conditional paths, like done in Dunn and Koes [9], led to performance degradation.

C.4 Source-target split

The splitting procedure in Algorithm 1 is dependent on the hyperparameters β and γ (see Section 3). Figure 5 and 6 report the absolute and relative (w.r.t. the original graphs) source and target set sizes for a batch of 25,000 randomly sampled training graphs. Across all runs, we used $\beta = 1.5$ and $\gamma = 0.45$. These settings produced an approximately uniform source set distribution with a slight bias toward smaller sets (corresponding to relative size). On QM9, the resulting target sets contain on average $\sim 18\%$ of the nodes of the original graphs. We also evaluated source-set distributions emphasizing mid-sized to larger sets, but near-uniform sampling performed best.

C.5 Inference details

Atom type sampling strategies The next atom type a_{n+1} is sampled from the predicted distribution over possible atom types using a hybrid sampling strategy between greedy and multinomial sampling.

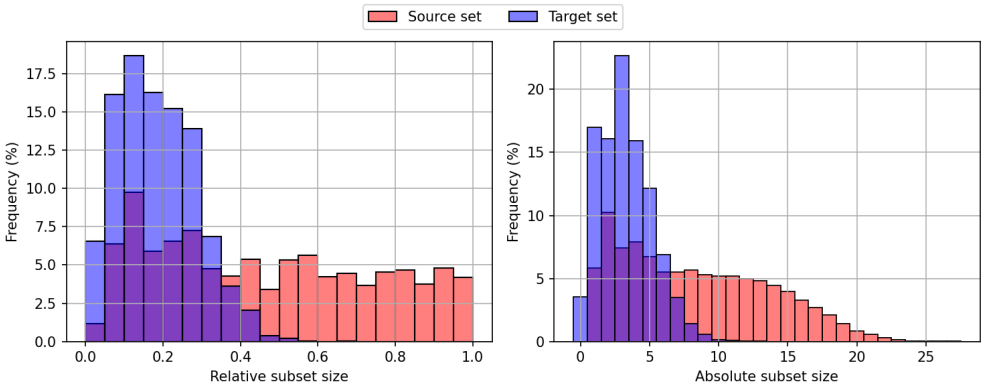


Figure 5: Relative and absolute size distribution of the source and target set w.r.t. the original graphs in the QM9 dataset. Sampling was performed with $\beta = 1.5$ and $\gamma = 0.45$.

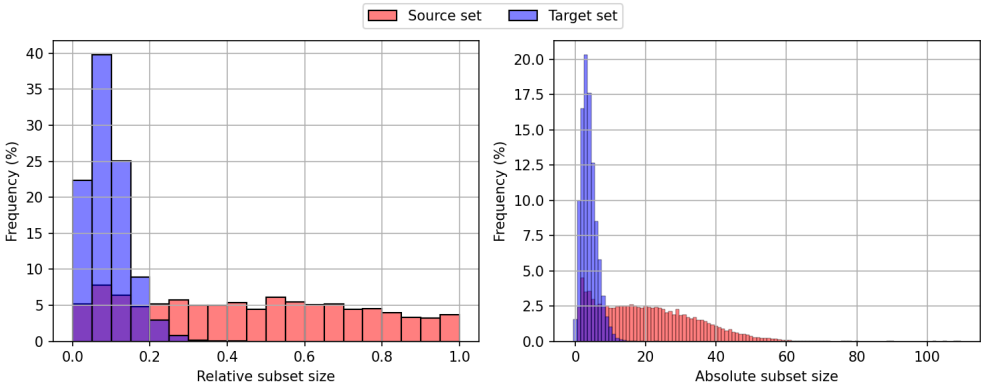


Figure 6: Relative and absolute size distribution of the source and target set w.r.t. the original graphs in the GEOM dataset. Sampling was performed with $\beta = 1.5$ and $\gamma = 0.45$.

Greedy sampling selects the atom type with the highest predicted probability, multinomial sampling draws samples from the predicted atom type distribution, and hybrid sampling greedily select the atom type with the highest probability if it is a hydrogen or the stop token, and otherwise samples from the multinomial distribution. An evaluation of the impact of this hybrid sampling strategy versus greedy and multinomial sampling can be found in the "Scaling and ablation" section (Appendix C.7). We also investigated the impact of temperature scaling on the sampling of the next atom type. These results can be found in a separate section (Appendix E.5).

Integration scheme for continuous coordinates We found Euler integration with $N = 60$ equidistant time steps to be best for integration of the flow ODE for the model trained on QM9 data. We also tried a Runge-Kutta-4 and an adaptive time step Dormand-Prince integrator, but did not observe any performance improvement. For the model trained on GEOM, the best results could be obtained by using the Euler-Maruyama integrator, which is described in the following. Sampling from a flow-based generative model can be performed by integrating the ODE $\mathrm{d}\mathbf{x}_t = \mathbf{v}_\theta(\mathbf{x}_t, t)\,\mathrm{d}t$. Beyond deterministic sampling, the velocity field can be used to simulate the following reverse-time stochastic differential equation (SDE) [35], thereby yielding a stochastic generative model:

$$\mathrm{d}\mathbf{x}_t = \mathbf{v}_\theta(\mathbf{x}_t, t)\mathrm{d}t + \frac{1}{2}w(t)\mathbf{s}_\theta(\mathbf{x}_t, t, c)\mathrm{d}t + \sqrt{\tau \cdot w(t)}\mathrm{d}\bar{\mathbf{W}}_t \tag{8}$$

where $\bar{\mathbf{W}}_t$ is a reverse-time Wiener process, w_t is the diffusion coefficient, and $\mathbf{s}_\theta(\mathbf{x}_t, t) = \nabla \log p_t(\mathbf{x}_t)$ is the Stein score function. We employ the same Euler-Maruyama integrator as described in Wang et al. [55]. The score $\mathbf{s}_\theta(\mathbf{x}_t, t) = \frac{t\mathbf{v}_\theta(\mathbf{x}_t, t) - \mathbf{x}_t}{1-t}$ can be derived from the velocity and the

diffusion coefficient is calculated as $w(t) = \frac{2(1-t)}{t+\eta}$, where $\eta = 10^{-3}$ for numerical stability. In all of our experiments we use $\tau = 0.3$, $N = 60$ integration steps, and equidistant time step spacing.

C.6 Hyperparameters

Table 3 and 4 summarize the hyperparameters of our best-performing models on QM9 and GEOM. The models both have 165M parameters. The loss function was minimized with the AdamW [32] optimizer with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.95$. We further used gradient clipping to 1.0 of the gradient norm. We implemented a cosine annealing learning rate schedule with linear warm-up epochs and set the minimum learning rate to 10% of the initial learning rate. Model training took 60 and 106 hours on QM9 and GEOM, respectively, given the hardware specifications listed above.

Table 3: Hyperparameters of the best performing NEAT model trained on the QM9 dataset.

Parameter	**Value**
batch size	2048
β	1.5
bias	true
dropout	0.1
epochs	10,000
flow matching head hidden dimension	1,536
flow matching head layers	6
flow matching noise standard deviation σ	1.4
γ	0.45
integrator	Euler
learning rate, weight decay	1.0×10^{-4}, 10^{-6}
learning rate decay number of epochs (cosine annealing scheduling)	10,000
learning rate minimum (cosine annealing scheduling)	10%
learning rate warm-up number of epochs (cosine annealing scheduling)	50
source set noise fraction δ	0.2
source set noise standard deviation	0.2
time step resampling	4
time step sampling	logit normal
transformer heads	12
transformer hidden dimension	768
transformer layers	12
vocabulary size	6

Table 4: Hyperparameters of the best performing NEAT model trained on the GEOM dataset.

Parameter	**Value**
batch size	512
β	1.5
bias	true
dropout	0.1
epochs	1,000
flow matching head hidden dimension	1,536
flow matching head layers	6
flow matching noise standard deviation σ	2.5
γ	0.45
integrator	Euler-Maruyama
learning rate, weight decay	2.5×10^{-5}, 10^{-6}
learning rate decay number of epochs (cosine annealing scheduling)	1,000
learning rate minimum (cosine annealing scheduling)	10%
learning rate warm-up number of epochs (cosine annealing scheduling)	10
source set noise fraction δ	0.5
source set noise standard deviation	0.5
time step resampling	4
time step sampling	logit normal
transformer heads	12
transformer hidden dimension	768
transformer layers	12
vocabulary size	17

Table 5: Model scaling results on the QM9 dataset. Best in **bold**, second-best underlined. The configuration selected for the final model is highlighted in *italics*.

transformer heads	transformer layers	transformer dimension	model parameters $\times 10^6$	atom stable	mol stable	lookup valid	lookup unique	xyz2mol valid	xyz2mol unique	xyz2mol novel
6	6	384	89.46	96.8	74.8	89.6	84.3	91.5	86.9	52.5
8	8	512	104.32	97.4	79.9	92.4	86.7	93.7	88.9	53.2
10	10	640	128.62	97.6	81.6	93.2	88.0	94.2	90.0	53.3
12	*12*	*768*	*164.72*	**97.8**	**83.1**	**93.3**	88.8	**95.2**	**91.6**	53.5
14	14	896	214.97	97.6	81.1	93.0	**89.1**	94.3	91.2	**54.5**

Table 6: Hyperparameter screening results on the GEOM dataset. Best in **bold**, second-best underlined. The configuration selected for the final model is highlighted in *italics*, chosen to balance optimal performance metrics with model size.

transformer heads	transformer layers	transformer dimension	FM head dimension	model parameters $\times 10^6$	atom stable	lookup valid	lookup unique	xyz2mol valid	xyz2mol unique
6	6	384	1536	89.46	79.4	99.7	**98.3**	86.2	85.2
8	8	512	1536	104.32	80.4	**99.8**	97.6	91.7	89.9
10	10	640	1536	128.62	**81.5**	**99.8**	96.7	94.0	91.4
12	*12*	*768*	*1536*	*164.72*	**81.5**	99.7	96.9	93.9	91.6
14	14	896	1536	214.97	80.9	99.7	96.7	94.4	**91.8**
16	16	1024	1536	281.72	80.9	**99.8**	96.1	**94.7**	91.5

Table 7: Impact of centering, noising, and sampling strategies on the GEOM model. Best in **bold**, second-best underlined. The configuration selected for the final model is highlighted in *italics*.

centering	noising	sampling	atom stable	mol stable	lookup valid	lookup unique	xyz2mol valid	xyz2mol unique	xyz2mol novel
no	no	greedy	81.1	0.5	97.9	97.4	74.2	73.7	73.7
no	no	multinomial	78.1	1.8	98.6	91.8	50.8	47.1	47.1
no	no	hybrid	79.0	2.5	99.5	83.6	53.7	45.1	45.1
yes	no	greedy	82.3	1.6	99.2	91.7	87.8	80.9	80.9
yes	no	multinomial	81.6	1.6	99.2	97.7	85.4	84.2	84.2
yes	no	hybrid	**82.7**	**2.5**	**99.7**	95.5	88.7	85.2	85.2
yes	yes	greedy	79.9	0.1	99.6	69.1	**95.4**	65.1	65.1
yes	yes	multinomial	80.4	0.4	99.5	**98.8**	90.9	90.5	90.5
yes	*yes*	*hybrid*	81.5	0.6	**99.7**	96.9	93.9	**91.6**	**91.6**

Table 8: Impact of additive vs. attention pooling on the QM9 model.

pooling strategy	atom stable	mol stable	lookup valid	lookup unique	xyz2mol valid	xyz2mol unique	xyz2mol novel
additive	97.8	83.1	93.3	88.8	95.2	91.6	53.5
attention	95.0	63.2	83.6	81.4	89.5	87.4	56.0

C.7 Scaling and ablations

We studied performance across various transformer sizes (Table 5 and 6). While increasing the number of transformer layers from 6 to 12 improves performance on the QM9 dataset, gains are lost beyond 12 layers. We further ablate key design choices like re-centering to the positional mean of the source set atoms, noising of the source set atoms, and different sampling strategies of the next atom type. Overall trends on the best GEOM model are summarized in Table 7. Centering consistently improves results across most metrics, particularly for multinomial and hybrid sampling strategies, regardless of whether noise is used. When greedy sampling is employed, centering still improves most metrics, with the exception of uniqueness. Overall, centering brings the strongest and most consistent improvements across all three modifications. We observe a tradeoff between validity and uniqueness across different sampling strategies. Greedy sampling maximizes validity but sacrifices uniqueness. Multinomial sampling introduces stochasticity into the sampling process, which maximizes uniqueness at the cost of validity. Hybrid sampling strikes a good balance between greedy and multinomial strategies. Noising has mixed effects: It does not greatly impact the lookup

metrics (atom stability, molecule stability, lookup validity, and lookup uniqueness). However, it significantly improves xyz2mol metrics, especially when combined with multinomial or hybrid sampling strategies. The combination of noising and hybrid sampling appears to work synergistically, bringing gains that are less pronounced when either is used alone. In summary, centering is the most impactful improvement, while noising and sampling work together to bring additional gains, particularly in xyz2mol metrics. A comparison of additive pooling and attention pooling on the best QM9 model is given in Table 8.

D Discussion of evaluation metrics

Hoogeboom et al. [18] introduced a pipeline for generating molecular structures from atomic point clouds by using a look-up table of interatomic distances for single, double, and triple bonds. A bond is established if the interatomic distance is smaller than the corresponding table value plus a defined tolerance margin, starting with the shortest bond type and progressing sequentially to larger bond types. If no bond type matches the distance criteria, the atoms remain unbonded. Atom stability within this framework is defined by adherence to the octet or duet rule based on the assigned bonds. Molecular stability is achieved when all constituent atoms are stable, and molecular validity is determined by whether the molecule can be converted into a valid SMILES string after sanitization using RDKit.

A notable limitation of this pipeline is that it processes interatomic distances in a sequential manner, which does not guarantee convergence to an optimal solution. Moreover, this pipeline does not consider aromatic bonds, which have lengths between single and double bonds. Consequently, aromatic molecules are often falsely categorized as unstable. For example, evaluating benzene with this pipeline results in 0% atom stability. This occurs because all bonds in benzene are incorrectly assigned as "double" bonds, leading to a chemically invalid configuration where each carbon atom has five bonds (two double bonds to neighboring carbon atoms and one single bond to a hydrogen atom). Given the high prevalence of aromatic compounds in the GEOM-Drugs dataset, this issue leads to a surprisingly low molecular stability on the GEOM-Drugs test set and on the sets of molecules generated from models trained on GEOM-Drugs. Similar concerns have been raised by Daigavane et al. [8] and Irwin et al. [21].

Another peculiarity of the pipeline is that the code used in Hoogeboom et al. [18] and related works [6, 54] reduces all assigned bonds to single bonds when evaluating the GEOM-Drugs dataset. This simplification allows many SMILES strings to pass the RDKit sanitization test, even though they contain carbon atoms with incorrect multiplicities. To ensure fair comparison with other models, we have adhered to this widely used pipeline without modifications, but also evaluate our models using the xyz2mol metric, which is described in the following.

The xyz2mol metric tests whether generated 3D atomic point clouds can be converted into valid SMILES after bond assignment *via* RDKit's rdDetermineBonds. We would like to point out that this metric is sensitive to the RDKit version used. We observe decreases of up to 5% with RDKit versions later than v2023.3.3, likely due to stricter bond assignment. We use RDKit v2025.9.1 throughout this work, which was the latest available update at project start. We recomputed xyz2mol metrics where model weights or pre-generated samples were available. Literature-reported xyz2mol results with undocumented RDKit versions are marked accordingly. While xyz2mol appears more robust than lookup metrics, we advocate for alternative evaluation pipelines to improve the accuracy and reliability of molecular generator assessment.

E Additional results

E.1 Molecular properties of generated molecules

We compared the molecules generated by our GEOM model to the training set across eight molecular properties: molecular weight, Crippen logP, topological polar surface area (TPSA), ring count, fraction of rotatable bonds, fraction of heteroatoms, fraction of halogen atoms, and fraction of stereocenters (Figure 7). Overall, the distributions of these properties are broadly similar between the two sets. The generated molecules exhibit a small bias toward simpler structures with lower molecular weight, fewer rings, and few stereocenters. None of the generated molecules were larger than the largest molecule found in the training set.

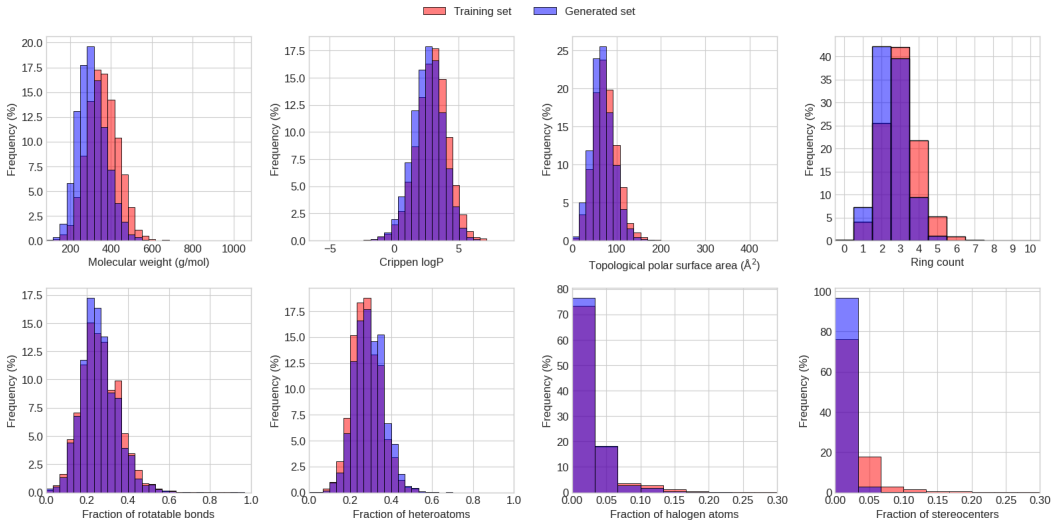


Figure 7: Comparison of generated molecules to the training set across eight molecular properties: molecular weight, Crippen logP, topological polar surface area, ring count, fraction of rotatable bonds, fraction of heteroatoms, fraction of halogen atoms, and fraction of stereocenters. Each panel shows overlaid histograms (training: red; generated: blue), reported as frequency in percent. Fractional properties are normalized by heavy atom count. Integer-valued properties use integer-centered bins.

E.2 Comparison to diffusion and Bayesian flow models

Diffusion models [17, 48] operate by progressively corrupting the training data with random noise through a forward diffusion process, and training a neural network to reverse this corruption *via* step-by-step denoising. During inference, new samples are generated by iteratively applying the learned reverse diffusion process to random noise. Prominent diffusion-based molecular generation models include EDM [18], GeoLDM [58], GCDM [36], SymDiff [62], and UniGEM [11]. Another alternative to 3D molecular generation are Bayesian flow networks like GeoBFN [50]. We compare NEAT to these six models in addition to the autoregressive and flow models already discussed in the main body of this work. Regarding QM9, we provide both lookup and xyz2mol metrics. For GEOM-Drugs, existing literature primarily reports stability and lookup metrics, and uniqueness and novelty are often omitted. Due to the unavailability of pre-trained weights for several baseline models, we report performance values directly from their original publications rather than performing independent re-evaluations. Consequently, due to this lack of available data, some metrics are missing in our comparison in Table 9. Given the observations discussed in Section D, both stability and lookup metrics should be interpreted cautiously. Overall, NEAT is competitive with the reported diffusion models, outperforming them in some metrics, but not all. As detailed in the main text, autoregressive frameworks offer practical advantages like adaptive atom counts, faster inference, and straightforward prefix conditioning.

E.3 Failure modes

We visually inspected generated molecules of our best GEOM model that failed xyz2mol bond assignment, examples are shown in Figure 8. Most issues were attributed to incorrect valency, such as issues with sp^3/sp^2 carbons, nitrogen atoms with four bonds (which fail under the assumption of a net charge of 0), or violations of aromaticity. In some cases, ring closure failed.

E.4 Impact of not using an SE(3)-equivariant model

NEAT uses a standard transformer, thereby benefiting from highly optimized building blocks, but it is not SE(3)-equivariant. We address this limitation with random rotation augmentations during training. We quantify the effect of rotations on prediction outcomes as follows. We begin with a single carbon atom positioned at the origin of the coordinate system and iteratively add atoms to construct a molecule. At each iteration, we sample $N = 32$ rotation matrices to generate N rotated

Table 9: Performance comparison of NEAT with diffusion or Bayesian flow (middle block) and autoregressive (lower block) models trained on the QM9 and GEOM dataset, respectively. All metrics are reported as percentages. The first row reports results on the QM9 and GEOM test set. All other rows report results computed from 10,000 generated molecules. The **best** and second-best entries are shown in bold and underlined, respectively. Entries for which generated molecules could be retrieved to be re-evaluated with our pipeline are marked with a single star (*). Entries for which the source code and model weights were available to generate molecules are marked with two stars (**). For these models and for NEAT, we report the mean and one standard deviation over three runs. Entries marked by a dagger (†) were taken from Cheng et al. [6]. Metrics for all other entries were taken from the original publications. xyz2mol entries marked with a double dagger (‡) are sourced from publications where the RDKit version is either v2023.3.3 or unspecified; these scores may decrease when re-evaluated with RDKit v2025.9.1.

	QM9						GEOM	
	atom stab.	mol. stab.	lookup valid	lookup unique	xyz2mol valid	xyz2mol unique	atom stab.	lookup valid
Test data	99.3	94.8	97.5	97.5	100	100	86.2	96.6
EDM	98.7†	82.0†	91.9†	90.7†	86.7†,‡	86.0†,‡	81.3†	92.6†
GeoLDM	98.9	89.4	93.8	92.7	91.3†,‡	90.3†,‡	84.4	99.3
GCDM	98.7	86.0	94.9	93.4	n.a.	n.a.	**88.1**	95.5
GeoBFN	**99.3**	**93.3**	96.9	92.4	n.a.	n.a.	86.2	91.7
SymDiff	98.9	89.7	96.4	**94.1**	92.8†,‡	91.4†,‡	86.2	99.3
UniGEM	99.0	89.8	95.0	93.2	n.a.	n.a.	85.1	98.4
G-SchNet*	93.7	63.6	80.1	74.7	73.0	70.6	n.a.	n.a.
G-SphereNet*	67.8	14.0	16.4	3.8	37.7	7.2	n.a.	n.a.
Symphony*	90.8	43.9	68.1	66.5	77.0	75.2	n.a.	n.a.
Geo2Seq	98.9	93.2	**97.1**	81.7	n.a.	n.a.	82.5	96.1
QUETZAL**	98.3 ± 0.1	87.5 ± 0.1	94.6 ± 0.1	90.1 ± 0.2	93.7 ± 0.3	89.1 ± 0.2	86.7 ± 0.1	95.7 ± 0.1
NEAT (ours)	97.8 ± 0.1	83.1 ± 0.1	93.3 ± 0.1	88.8 ± 0.3	**95.2** ± 0.2	**91.6** ± 0.3	81.5 ± 0.1	**99.7** ± 0.2

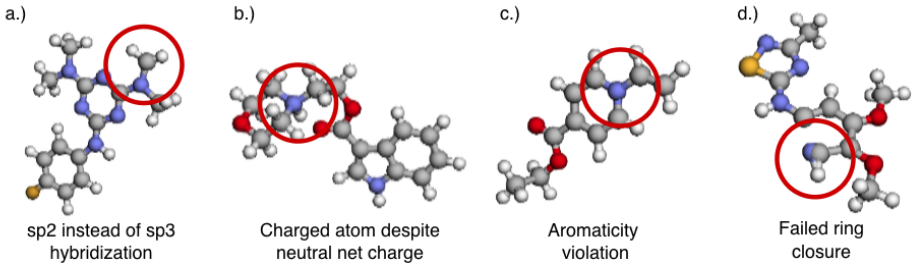


Figure 8: Qualitative illustration of possible failure modes.

versions of the partial molecule. For each rotated version, we compute its latent representation and derive the multinomial distribution over atom types. The next atom type is selected as the one with the highest probability and most frequent occurrence across the N rotated versions. Given the selected atom type, we compute distributions over atomic positions for all N rotated molecules. To achieve this, we sample an initial 3D Gaussian distribution with $K = 256$ points as the source distribution for flow matching. This distribution is rotated into the respective frames defined by the N rotation matrices. Flow matching then produces target distributions, which are rotated back to the reference frame using the inverse operation. The atom position for the next iteration is sampled randomly from the points in the aligned target distributions.

We use the following metrics for evaluation. For atom type prediction, we calculate the pairwise Jensen-Shannon divergence between the multinomial distributions of atom types across the N rotated molecules and report the mean divergence. For atomic coordinates, we compute the pairwise Sinkhorn divergence between the N re-aligned next-coordinate distributions and report the mean divergence.

The experiment is repeated ten times, and we report the mean and 90% confidence interval (CI) of the evaluation metrics, as shown in Figure 9 and 10. To contextualize the scaling of these metrics, we also provide lower and upper bounds. For atom type prediction, the upper bound is modeled with randomly sampled multinomial distributions, representing a dummy classifier. The lower bound is a divergence of zero, corresponding to perfect SE(3) equivariance. For atomic coordinate prediction, the upper bound is calculated as the average Sinkhorn divergence between N target distributions without realignment back to the reference frame. The lower bound is the average Sinkhorn divergence between N target distributions without randomly rotating partial molecules but initialized with N different 3D Gaussian distributions as source distribution for flow matching.

As shown in Figure 9 and 10, we observe only small variations in the mean pairwise Jensen-Shannon divergences of next atom-type distributions and the mean pairwise Sinkhorn divergences of next-coordinate distributions. The Jensen-Shannon divergence appears to slightly increase with the number of atoms, but remains relatively low compared to the worst-case upper bound. While the worst-case upper bound of the Sinkhorn divergence increases significantly with the number of atoms, the observed divergence remains unaffected by atom count and stays close to the lower bound. We conclude that the lack of architectural SE(3)-equivariance does not significantly impact the performance of our model. Although our architecture is not explicitly SE(3)-equivariant, the incorporation of random rotations during training effectively compensates for this limitation.

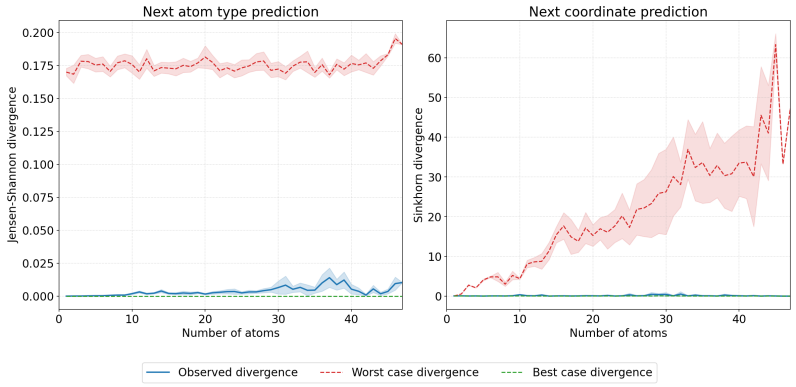


Figure 9: Mean pairwise Jensen-Shannon divergence of next atom-type distributions (left) and mean pairwise Sinkhorn divergence of next-coordinate distributions (right) across $N = 32$ rotated versions of partial molecules, plotted against the number of atoms in the molecule. The shaded regions represent the 90% confidence intervals (CI) across 10 repeated runs.

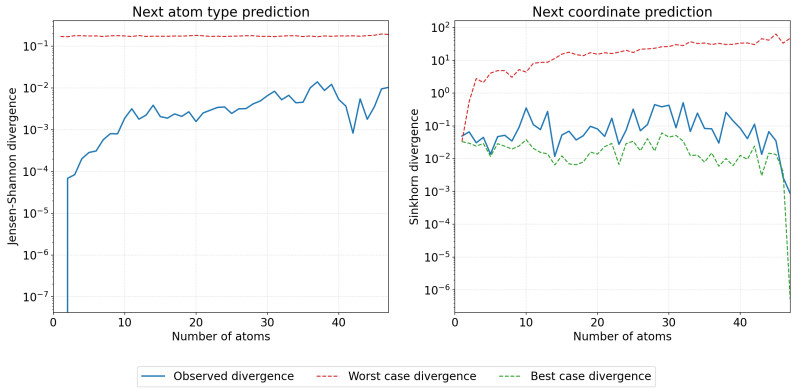


Figure 10: Log-scaled pairwise Jensen-Shannon divergence of next atom-type distributions (left) and mean pairwise Sinkhorn divergence of next-coordinate distributions (right) across $N = 32$ rotated versions of partial molecules, plotted against the number of atoms in the molecule. Confidence intervals (CI) are omitted for clarity.

E.5 Impact of temperature scaling

In language modeling, it is standard practice to use a temperature parameter to adjust the sharpness or flatness of the predicted next-token distribution, thereby influencing the generation process. As outlined in Section C.5, our best GEOM model employs a hybrid sampling approach for predicting the next atom type, utilizing multinomial sampling for heavy atom tokens. To explore the impact of temperature scaling on the generated molecules, we modify this multinomial distribution by introducing a temperature parameter. Figure 11 demonstrates how temperature influences token probabilities during generation, revealing distinct patterns in the prediction of hydrogens and heavy atoms across low and high temperatures. Figure 12 shows that higher temperatures decrease atom count, hydrogen and carbon content, and aromatic bonds, while increasing heteroatom content. Lower temperatures improve validity but reduce uniqueness, with $T = 1$ offering the best balance of validity and uniqueness based on the xyz2mol metric. Additionally, molecule size decreases as temperature rises. These findings highlight the role of temperature in balancing molecular properties during generation.

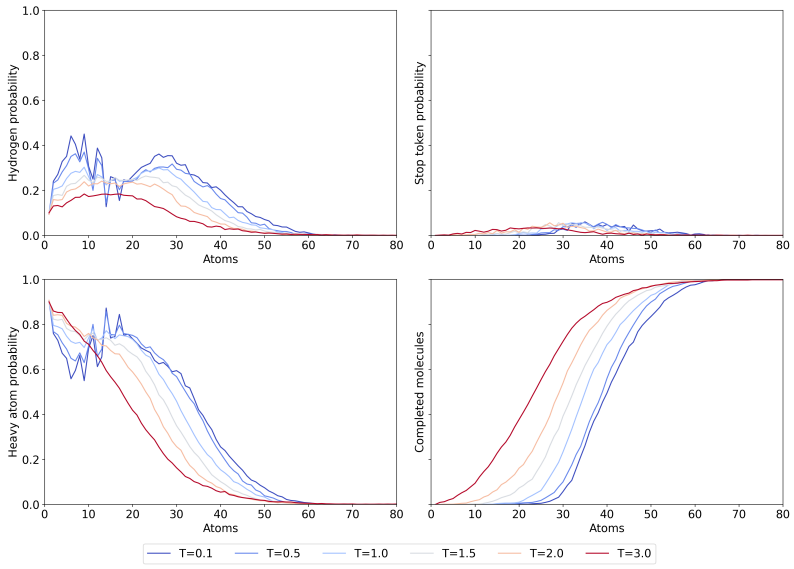


Figure 11: Mean probability of the next atom type, averaged over 1000 molecules during the generation process. The plot shows the probabilities of the atom type prediction head before temperature scaling, evaluated across different temperatures.

E.6 Impact of molecular size on validity

We binned molecules generated by our best GEOM-model based on their total number of atoms and computed three metrics for each bin: the total number of molecules, the absolute number of xyz2mol-valid molecules, and the ratio of xyz2mol-valid molecules to the total number of molecules. Results can be seen in Figure 13. We find no definitive correlation between molecular size and validity, indicating that NEAT's performance does not degrade as molecular size increases within the range of the training data distribution, although we observe a slight variation in validity for molecules whose sizes deviate significantly from the mean. This trend could be due to two possible factors: either a sample size effect, as the tails of the distribution contain relatively few molecules, or genuine model errors. If the second explanation is correct, the observed variation cannot be attributed to error accumulation during autoregressive generation, as small molecules fail at similar rates to large ones. A more plausible explanation is that the model has encountered fewer examples of molecules at the extremes of the size distribution during training, compared to those closer to the mean, leading to a higher likelihood of mistakes for these less familiar cases. We perform the same analysis of the ratio of valid molecules *vs.* total number of atoms using the bond lookup pipeline instead of the xyz2mol pipeline, results are presented in Figure 14. We found an even less pronounced correlation: the Pearson correlation coefficient was 0.07, compared to the previous 0.12 observed with the xyz2mol pipeline.

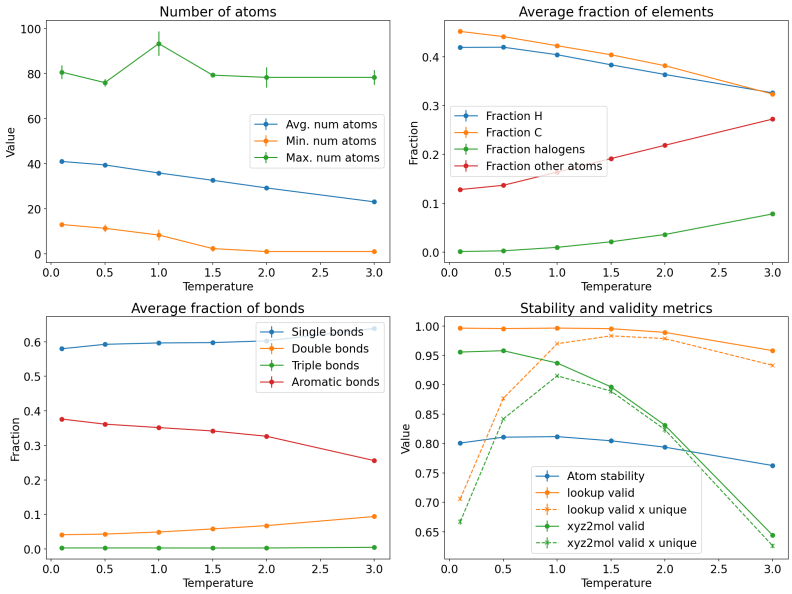


Figure 12: Correlation between evaluation metrics and various statistics of the generated data distribution w.r.t. temperature-scaled atom type sampling. For each temperature setting, we generate 10,000 molecules and report the mean and standard deviation (error bars) across three random seeds.

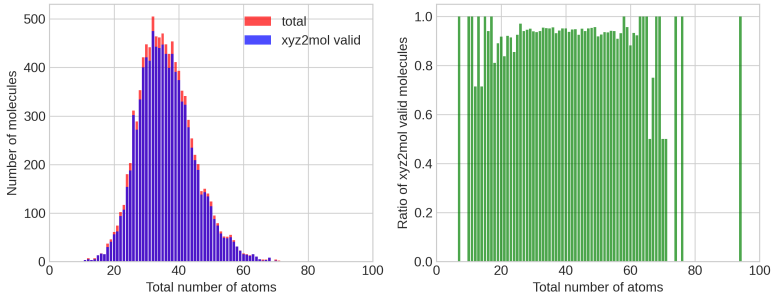


Figure 13: xyz2mol validity vs. total number of atoms. Left: distribution of number of atoms. The red bars represent the total number of molecules in each bin, the blue bars represent the absolute number of xyz2mol valid molecules. Right: ratio of xyz2mol valid molecules to the total number of molecules for each bin.

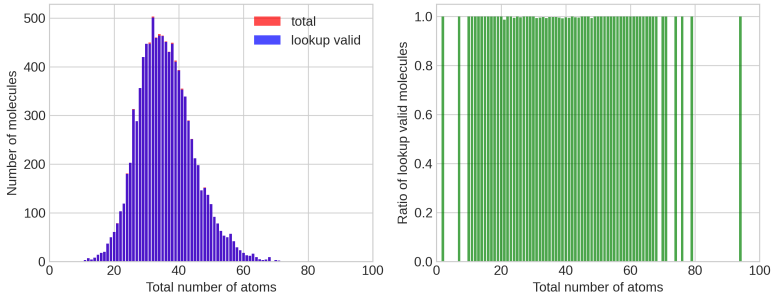


Figure 14: Lookup validity vs. total number of atoms. Left: distribution of number of atoms. The red bars represent the total number of molecules in each bin, the blue bars represent the absolute number of lookup valid molecules. Right: ratio of lookup valid molecules to the total number of molecules for each bin.

E.7 Impact of integration steps on baseline validity

We investigate the impact of the number of integration steps on the molecular validity of the SemlaFlow [21] and FlowMol3 [9] baselines. We report the average runtime and performance metrics over three independent runs for both models with three choices for the number of integration steps: a) the number recommended by the authors of SemlaFlow and FlowMol3, b) 60 (same as NEAT), and c) 15 (resulting in runtimes matching NEAT's). In the latter setting, SemlaFlow and FlowMol3 achieve xyz2mol validities of 80.4% and 55.5%, respectively, compared to NEAT's xyz2mol validity of 93.9%. The results are summarized in Table 10.

Table 10: Average runtime and performance metrics for both SemlaFlow and FlowMol3 with three choices for the number of integration steps: a) the number recommended by the authors of SemlaFlow and FlowMol3, b) 60 (same as NEAT), and c) 15 (resulting in runtimes matching NEAT's). Mean $\pm$ one standard deviation are reported over 3 independent runs.

Model	Steps	Runtime	xyz2mol valid	xyz2mol unique
SemlaFlow	100	1064	86.6 ± 0.5	86.5 ± 0.5
SemlaFlow	60	662	85.5 ± 0.1	85.5 ± 0.1
SemlaFlow	15	188	80.4 ± 0.4	80.4 ± 0.4
FlowMol3	250	3420	95.8 ± 0.7	95.7 ± 0.7
FlowMol3	60	855	92.2 ± 0.3	91.9 ± 0.3
FlowMol3	15	236	55.5 ± 0.2	55.5 ± 0.2
NEAT	60	208	93.9 ± 0.2	91.6 ± 0.2

F FlowMol3 inpainting procedure

FlowMol3 [9] represents a molecule as a graph $g = (X, A, C, E)$ with atom positions X, atom type A, charges C, and edges E. Given a fixed prefix g_1^{prefix} with n atoms, we sample a corresponding prefix state g_0^{prefix} from the prior and construct the same straight interpolation path used during FlowMol3 training:

$$g_t^{\text{prefix}} = (1-t)g_0^{\text{prefix}} + tg_1^{\text{prefix}} \tag{9}$$

At inference, we initialize the full molecule g_0 from the prior and define a binary mask M, where $M = 1$ denotes the prefix atoms to be kept fixed and $M = 0$ denotes the remaining atoms to be generated (edges are handled analogously). During integration, after each model update to g_t, we overwrite the masked prefix coordinates/features with the corresponding point on the prefix interpolation path:

$$g_t[M] \leftarrow g_t^{\text{prefix}} \tag{10}$$

Thus, the prefix follows the prescribed noised-to-clean path while the unmasked part is generated conditionally on it through the model dynamics. This provides a direct inference-time inpainting baseline for FlowMol3 under the same prefix constraints as our method, without requiring any additional training. For each of the 100 prefixes, we sample 1000 molecules. The results are reported in Table 2.

G Visualizations

G.1 Examples of generated molecules

Figure 15, 16, 17, and 18 show 2D and 3D plots of randomly selected molecules generated by NEAT trained on the QM9 and on the GEOM-Drugs datasets.

G.2 Prefixes

The complete list of prefixes extracted from the GEOM dataset is shown in Figure 19, 20, and 21.

G.3 Vector field visualization

Figure 22–24 illustrate iterative molecular construction by the NEAT model trained on QM9. Each row corresponds to one construction step. From left to right, panels show Gaussian-initialized points (orange) advected by the NEAT velocity field (gray arrows, with length indicating speed). Trajectories are integrated using explicit Euler with a uniform time step for $N = 60$ steps. The green diamond marks the point selected as the atom position for the next iteration. We use QM9 for visualization, as examples from the larger GEOM dataset become visually cluttered.

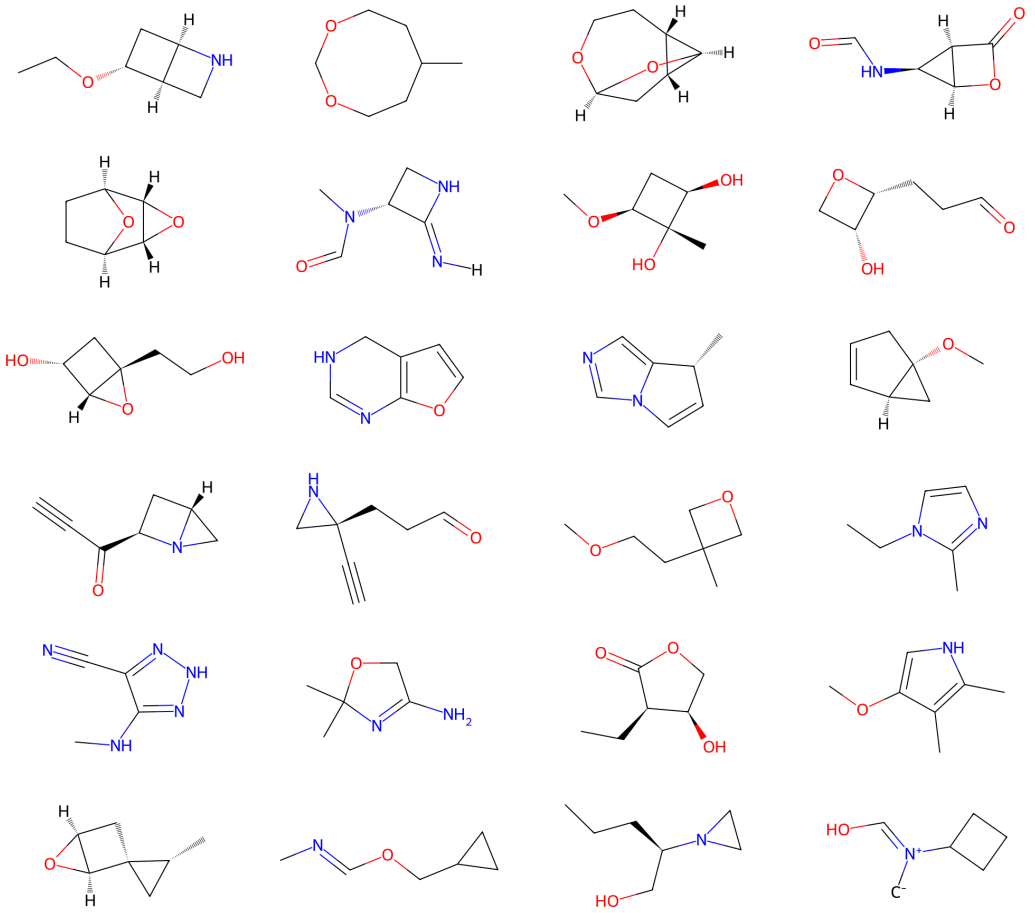


Figure 15: Randomly selected molecules generated by NEAT trained on the QM9 dataset. Implicit hydrogen atoms have been removed for the sake of clarity. 3D plots of the same molecules are shown in the next figure.

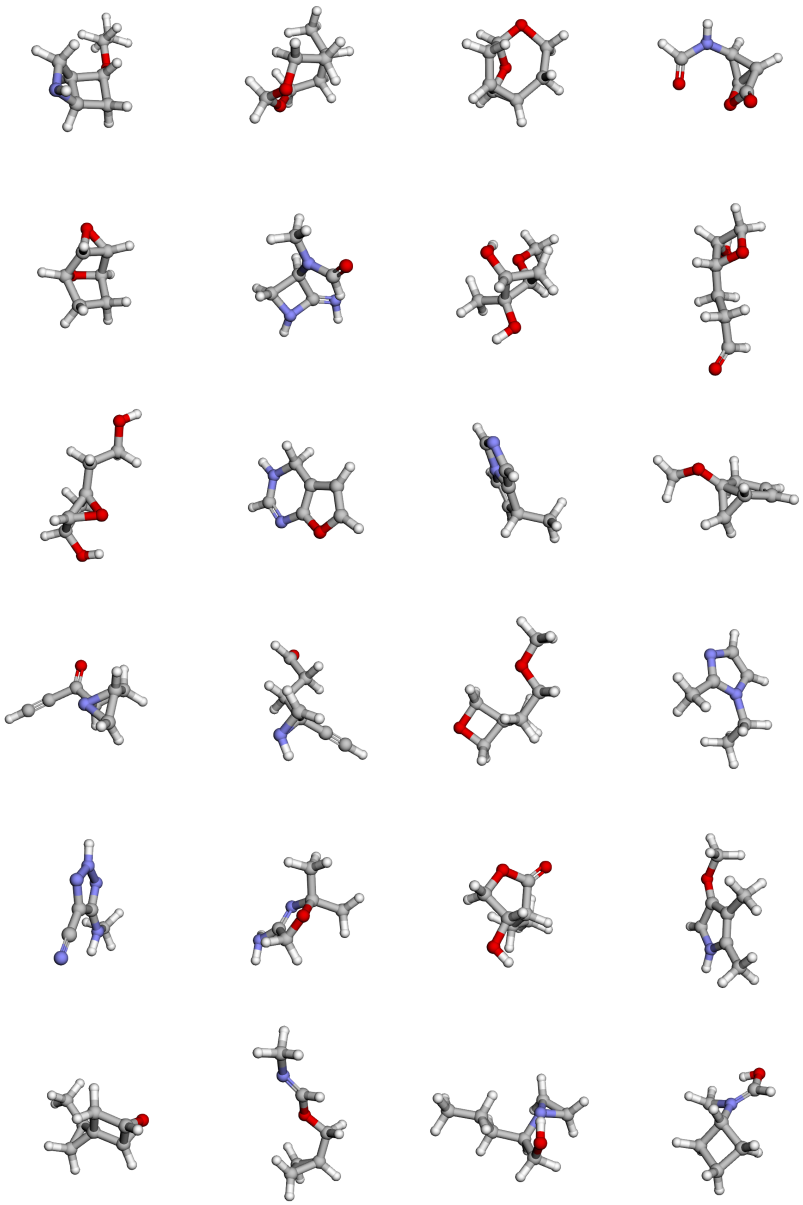


Figure 16: Randomly selected molecules generated by NEAT trained on the QM9 dataset (white: hydrogen, gray: carbon, blue: nitrogen, red: oxygen, brown: fluorine). 2D plots of the same molecules are shown in the previous figure.

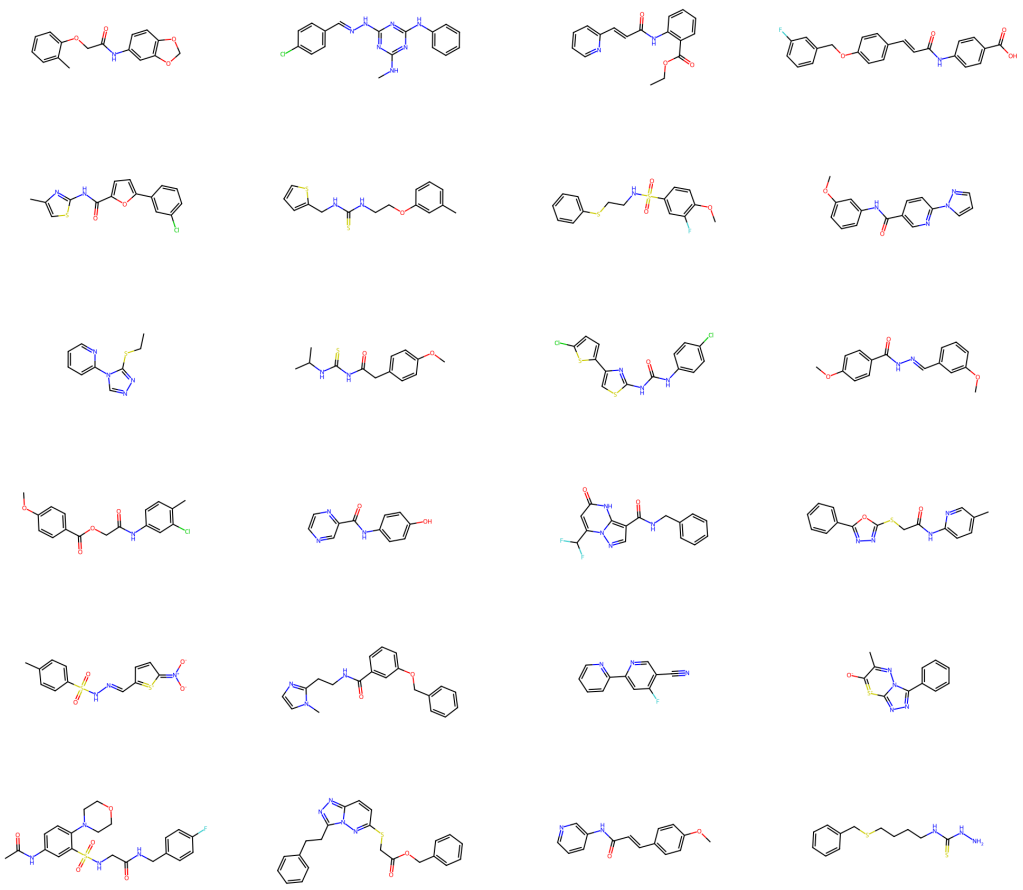


Figure 17: Randomly selected molecules generated by NEAT trained on the GEOM-Drugs dataset. Implicit hydrogen atoms have been removed for the sake of clarity. 3D plots of the same molecules are shown in the next figure.

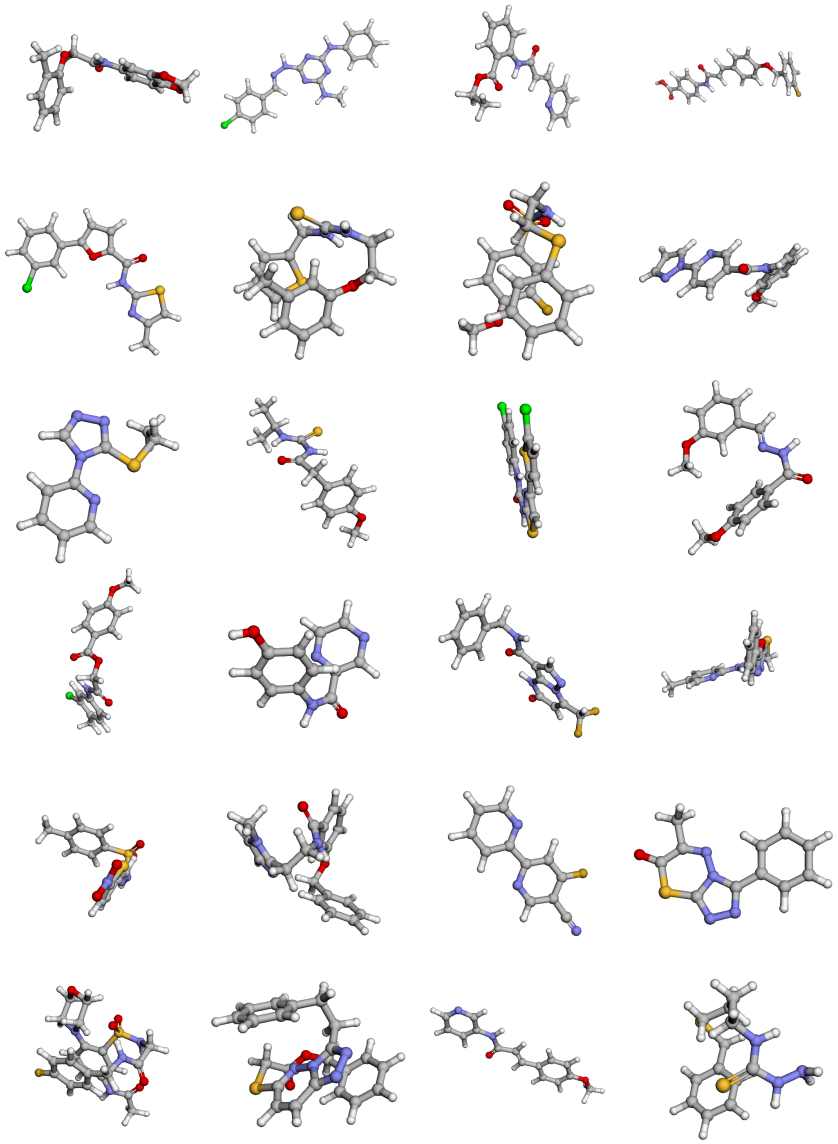


Figure 18: Randomly selected molecules generated by NEAT trained on the GEOM-Drugs dataset (white: hydrogen, gray: carbon, blue: nitrogen, red: oxygen, yellow: sulfur, brown: fluorine, green: chlorine). 2D plots of the same molecules are shown in the previous figure.

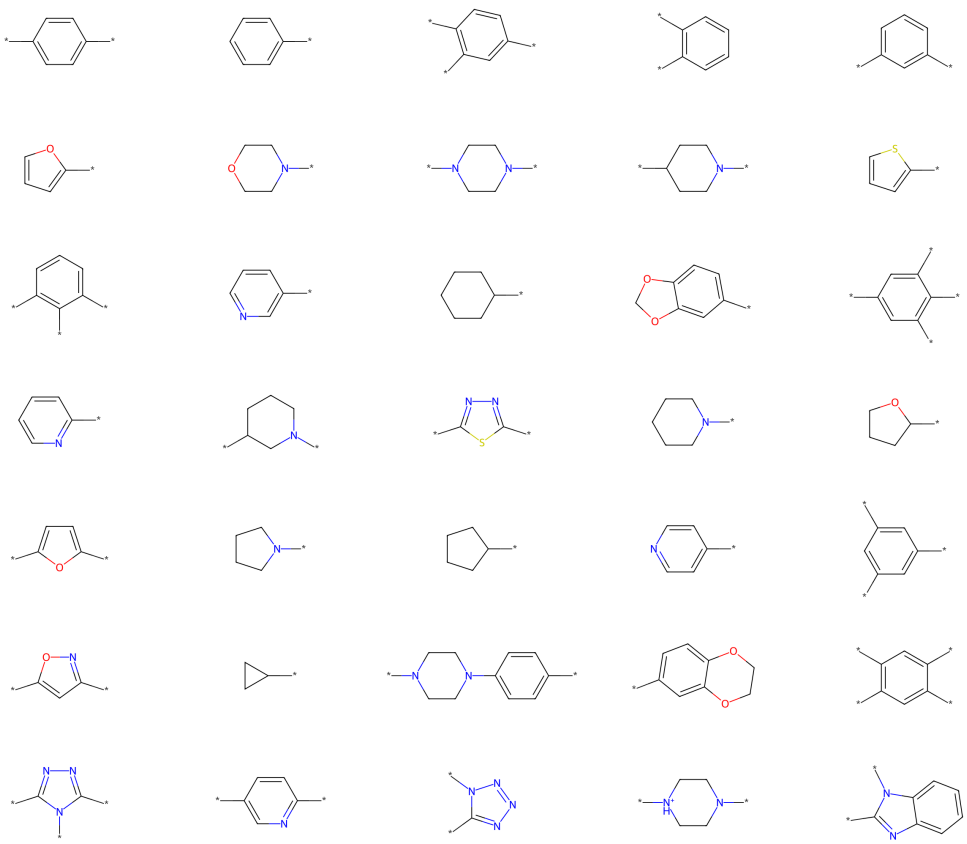


Figure 19: Prefixes used for the prefix completion task (1/3).

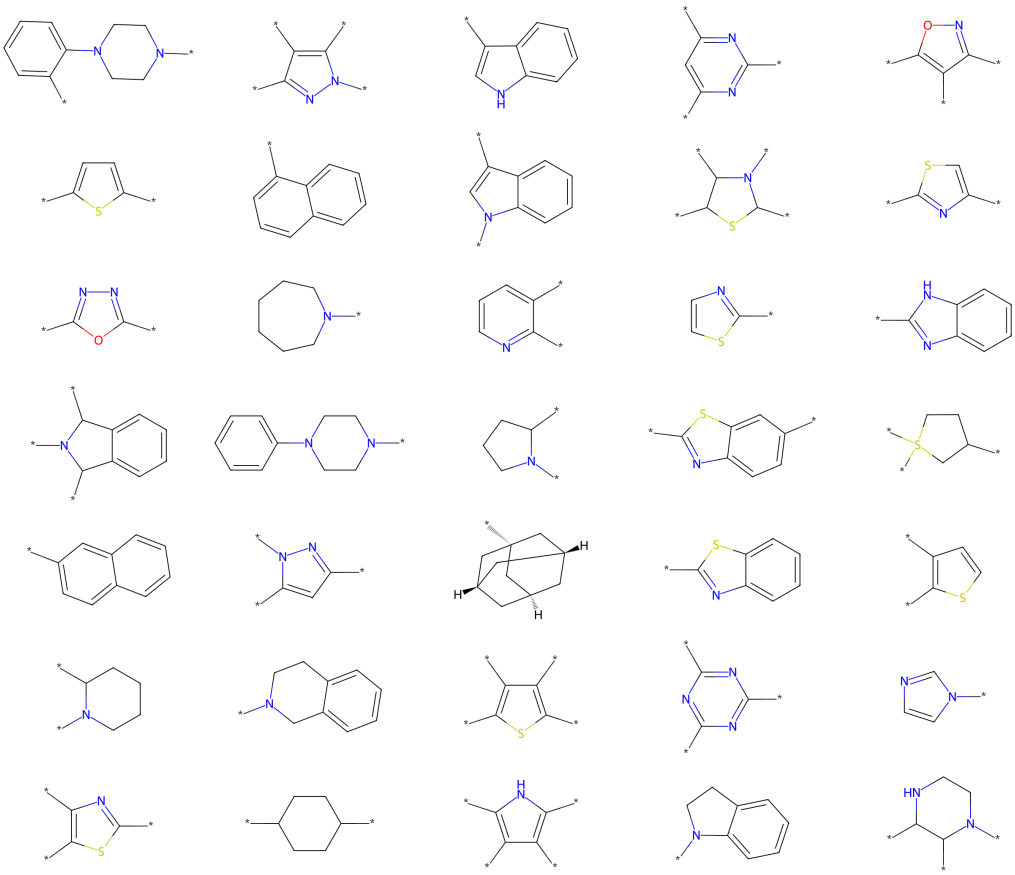


Figure 20: Prefixes used for the prefix completion task (2/3).

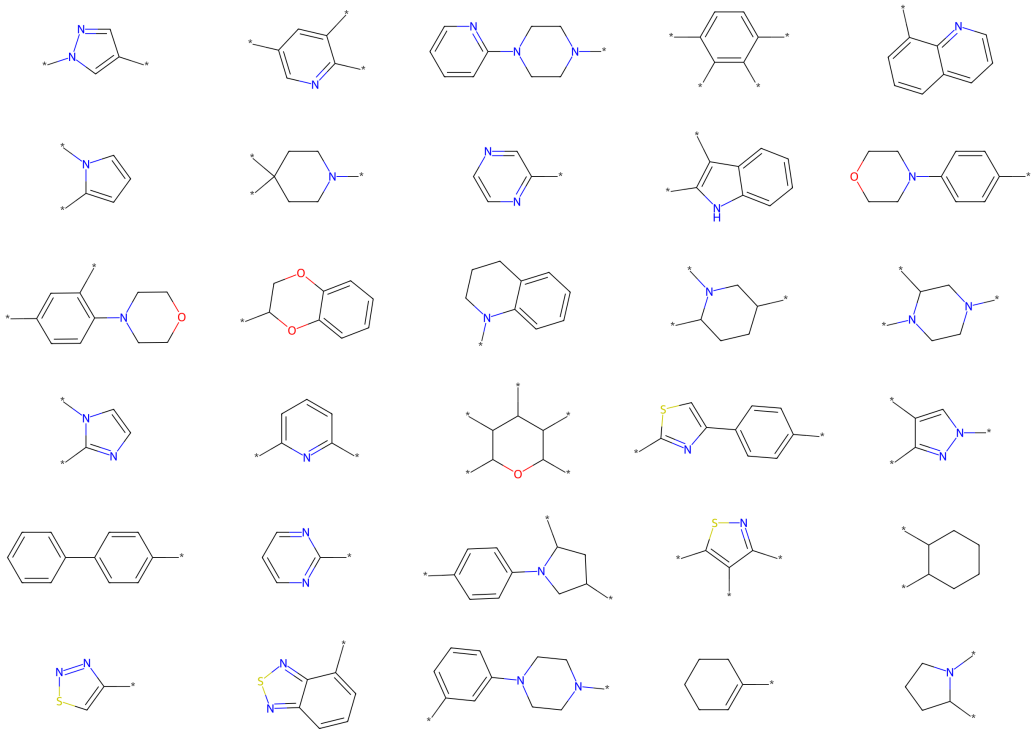


Figure 21: Prefixes used for the prefix completion task (3/3).

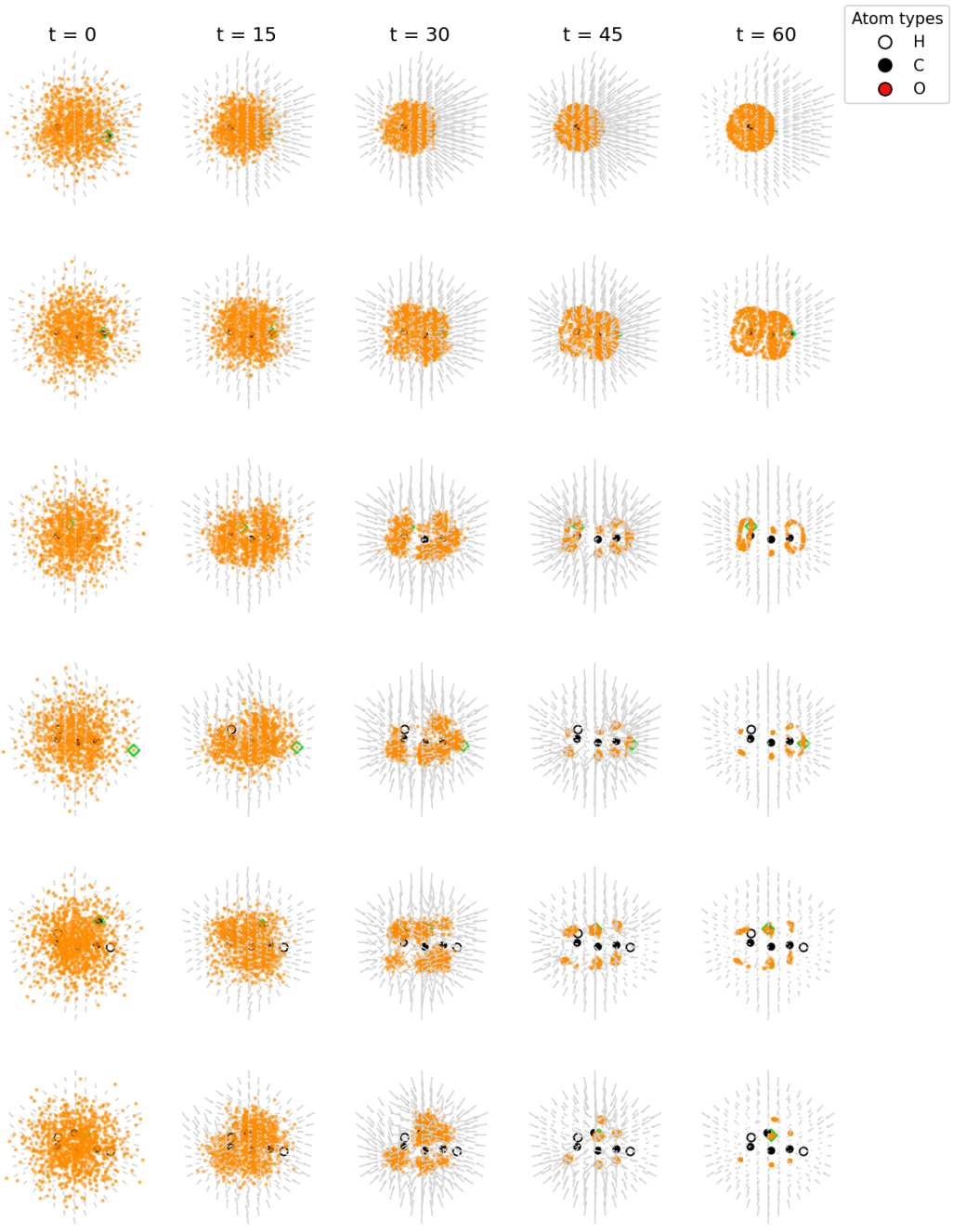


Figure 22: Iterative build of a molecule with NEAT, trained on QM9 (1/3).

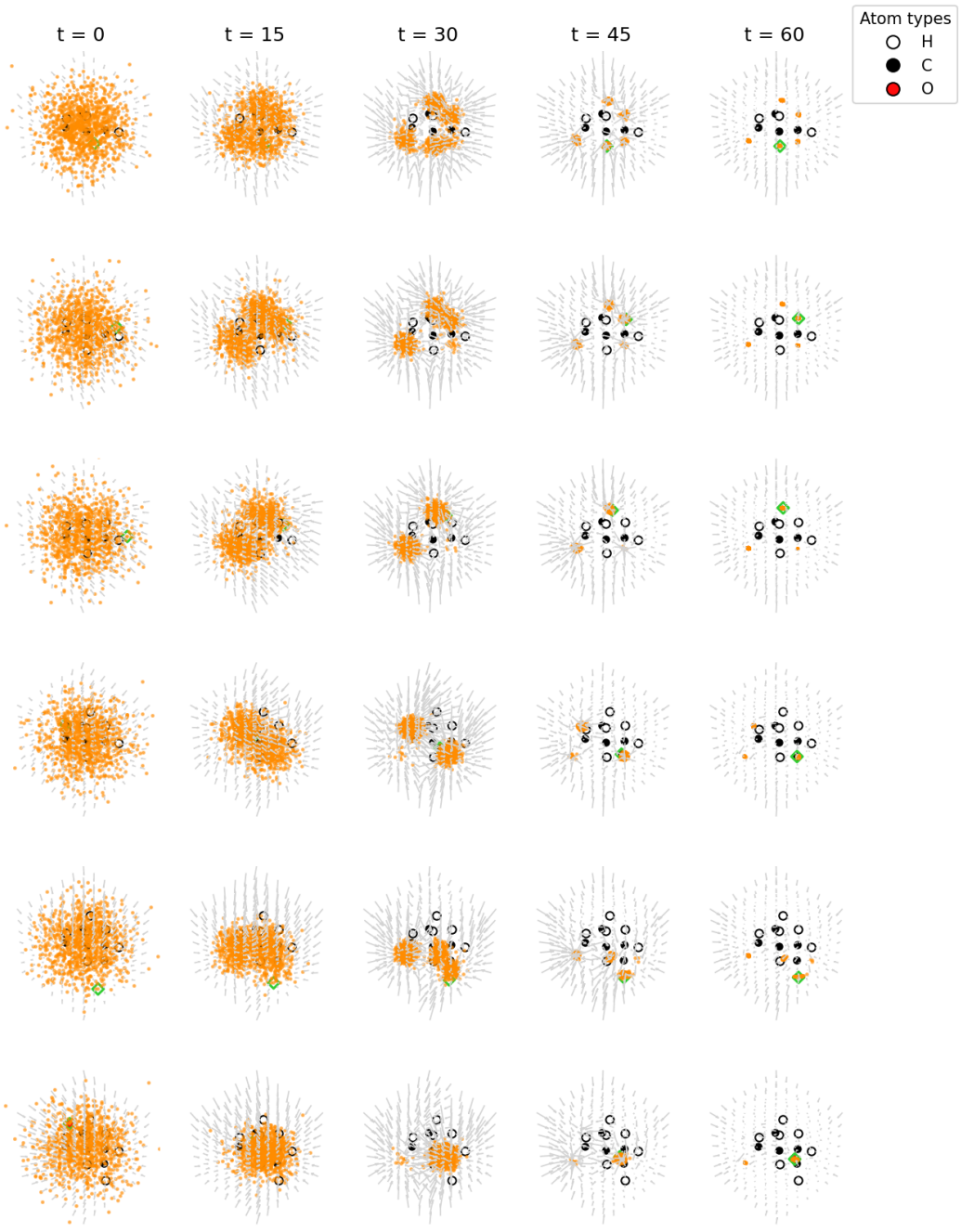


Figure 23: Iterative build of a molecule with NEAT, trained on QM9 (2/3).

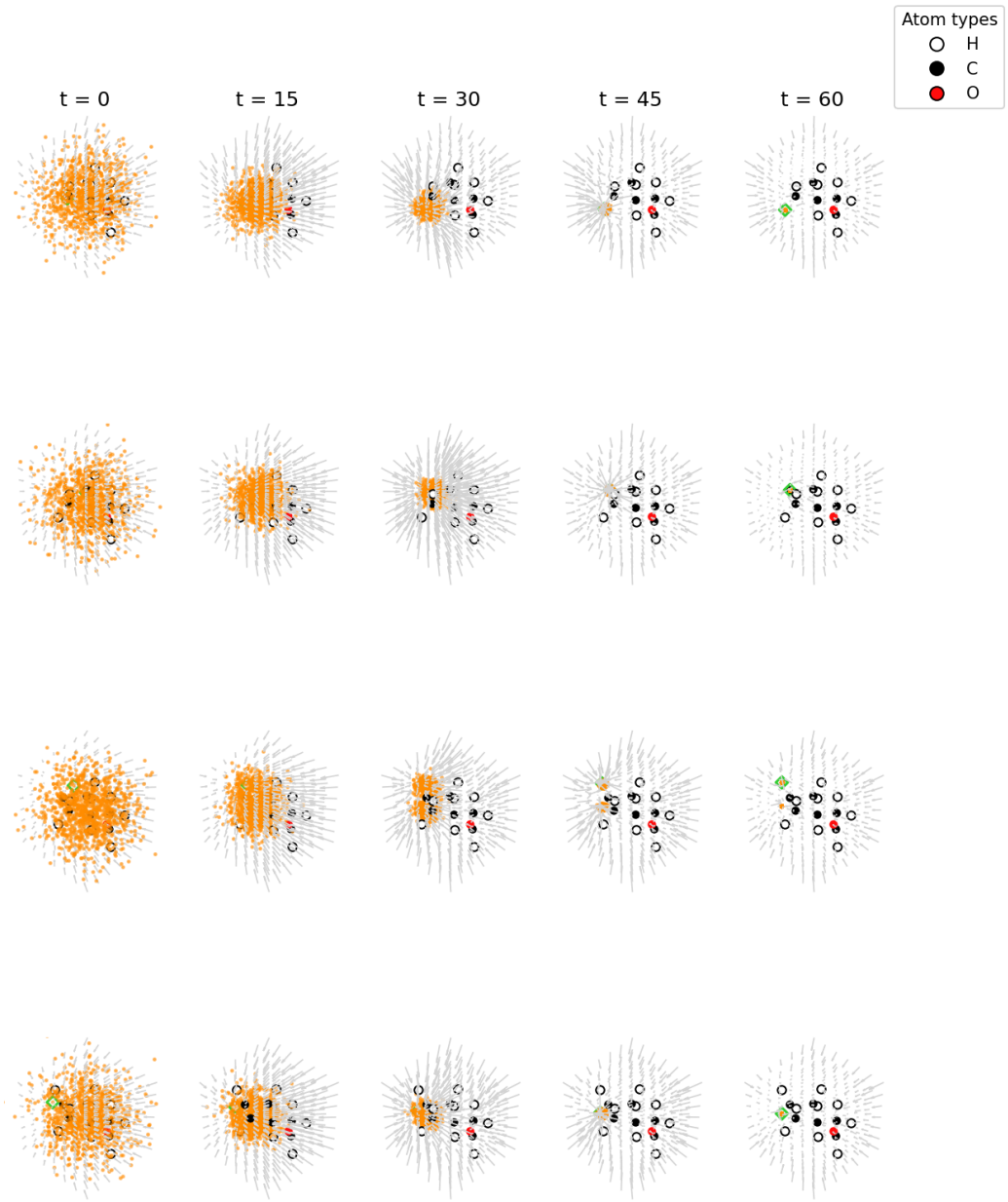


Figure 24: Iterative build of a molecule with NEAT, trained on QM9 (3/3).

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Our main contributions are elucidated in the abstract and the introduction.

   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: The limitations of our work are discussed in the main text in a dedicated paragraph and further elaborated in the Appendix.

   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: The paper does not include theoretical results.

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: The pseudo-code for our proposed method and a detailed description of the model architecture are given in the "Methods" section of the paper. The experimental setting is described in the "Experiments" section of the paper. Additional information on the model architecture and the experimental setting are provided in the "Implementation details" section of the Appendix. Additionally, the submission includes an anonymized link to our code repository and pretrained model weights. We will include a de-anonymized link to the official code repository upon acceptance. All datasets used are publicly available. We trust that the provided information is complete and sufficient to reproduce all results presented in this work.

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: [Yes]

   Justification: This submission includes an anonymized link to our code repository, including pretrained model weights. We will include a de-anonymized link to the official code repository upon acceptance. Our repository includes scripts for data and model download, data preprocessing, model training, model inference, and evaluation of the results. System specification are detailed in the Appendix.

   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: We provide comprehensive information on the experimental setting in the "Experiments" section of the core of the paper and present detailed information in the "Implementation details" section of the Appendix. We trust that the provided information, together with the released code and model weights, is complete and sufficient to reproduce all results presented in this work.

   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: In our performance evaluation, we report the mean and 1-sigma standard deviation interval over three inference runs with different random seed initialization. When the source code and trained model weights were available, we re-evaluated baselines using our evaluation pipeline to maximize compatibility. All this is stated in the paper.

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: The paper specifies the hardware used for training and inference, including GPU models, memory, and processors. It also specified the version of the software and libraries employed. The size, computational cost, and runtime of the proposed model are discussed in the main part of the paper.

   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: The research did not involve human subjects or participants. The datasets do not contain personally identifiable information and are still available for use. The QM9 and GEOM-Drugs datasets that were used in this work have defined licensing agreements, which we respected. We discussed the societal impact and potential harmful consequences of our work in the "Broader societal impact" section of the Appendix.

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: [Yes]

    Justification: We discussed the societal impact and potential harmful consequences of our work and proposed adequate mitigation measures in the "Broader societal impact" section of the Appendix.

    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: We do not foresee high risk of misuse for our model.

    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: The original owners of the QM9 and GEOM-Drugs datasets used in this work are properly referenced and credited and the licenses of both datasets are explicitly mentioned in the "Data processing" section of the Appendix. We do not repackage the datasets.

    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: [Yes]

    Justification: The paper introduces new assets in the form of code and trained model weights. We provide an anonymized links to our code repository and pretrained model weights. We will include a de-anonymized link to the official code repository / model weights upon acceptance.

    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: This work 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: This work 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: [N/A]

    Justification: LLMs were used only for spell-checking, editing of self-written text, formatting plots and implementing standard elements of code. They were not involved in the development of the core methods presented in this work.

    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.