
Equivariant Diffusion for Molecule Generation in 3D

Emiel Hoogeboom^{*1} Victor Garcia Satorras^{*1} Clément Vignac^{*2} Max Welling¹

Abstract

This work introduces a diffusion model for molecule generation in 3D that is equivariant to Euclidean transformations. Our E(3) Equivariant Diffusion Model (EDM) learns to denoise a diffusion process with an equivariant network that jointly operates on both continuous (atom coordinates) and categorical features (atom types). In addition, we provide a probabilistic analysis which admits likelihood computation of molecules using our model. Experimentally, the proposed method significantly outperforms previous 3D molecular generative methods regarding the quality of generated samples and efficiency at training time.

1. Introduction

Modern deep learning methods are starting to make an important impact on molecular sciences. Behind the success of AlphaFold in protein folding prediction (Jumper et al., 2021), an increasing body of literature develops deep learning models to analyze or synthesize (*in silico*) molecules (Simonovsky & Komodakis, 2018; Gebauer et al., 2019; Klicpera et al., 2020; Simm et al., 2021).

Molecules live in the physical 3D space, and as such are subject to geometric symmetries such as translations, rotations, and possibly reflections. These symmetries are referred to as the Euclidean group in 3 dimensions, E(3). Leveraging these symmetries in molecular data is important for good generalization and has been extensively studied (Thomas et al., 2018; Fuchs et al., 2020; Finzi et al., 2020).

Although developed for discriminative tasks, E(n) equivariant layers can also be used for molecule generation in 3D. In particular, they have been integrated into autoregressive models (Gebauer et al., 2018; 2019) which artificially in-

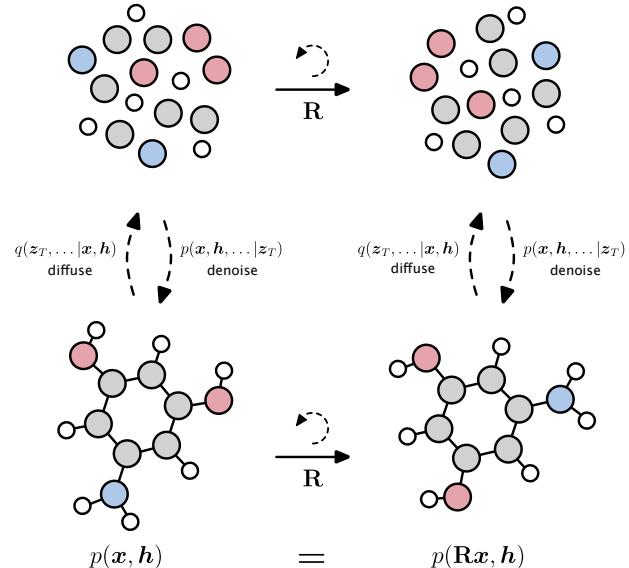


Figure 1. Overview of the EDM. To generate a molecule, a normal distributed set of points is denoised into a molecule consisting of atom coordinates x in 3D and atom types h . As the model is rotation equivariant, the likelihood is preserved when a molecule is rotated by R .

duce an order in the atoms and are known to be difficult to scale during sampling (Xu et al., 2021b). Alternatively, continuous-time normalizing flows such as Köhler et al. (2020) or E-NF (Satorras et al., 2021a) are expensive to train since they have to integrate a differential equation, leading to limited performance and scalability.

In this work, we introduce E(3) Equivariant Diffusion Models (EDMs). EDMs learn to denoise a diffusion process that operates on both continuous coordinates and categorical atom types. To the best of our knowledge, it is the first diffusion model that directly generates molecules in 3D space. Our method does not require a particular atom ordering (in contrast to autoregressive models) and can be trained much more efficiently than normalizing flows. To give an example, EDMs generate up to 16 times more stable molecules than E-NFs when trained on QM9, while requiring half of the training time. This favourable scaling behaviour allows EDMs to be trained on larger drug-like datasets such as GEOM-Drugs (Axelrod & Gomez-Bombarelli, 2020).

Our contributions can be summarized as follows. We intro-

^{*}Equal contribution ¹UvA-Bosch Delta Lab, University of Amsterdam, Netherlands ²EPFL, Lausanne, Switzerland. Correspondence to: Emiel Hoogeboom <e.hoogeboom@uva.nl>, Victor Garcia Satorras <v.garciasatorras@uva.nl>, Clément Vignac <clement.vignac@epfl.ch>.

duce an equivariant denoising diffusion model that operates on atom coordinates and categorical features. We add a probabilistic analysis which allows likelihood computation and this analysis is consistent with continuous and categorical features. We show that our method outperforms previous molecule generation models in log-likelihood and molecule stability.

2. Background

2.1. Diffusion Models

Diffusion models learn distributions by modelling the reverse of a diffusion process: a *denoising* process. Given a data point \mathbf{x} , a diffusion process that adds noise to \mathbf{z}_t for $t = 0, \dots, T$ is defined by the multivariate normal distribution:

$$q(\mathbf{z}_t | \mathbf{x}) = \mathcal{N}(\mathbf{z}_t | \alpha_t \mathbf{x}_t, \sigma_t^2 \mathbf{I}), \quad (1)$$

where $\alpha_t \in \mathbb{R}^+$ controls how much signal is retained and $\sigma_t \in \mathbb{R}^+$ controls how much noise is added. In general, α_t is modelled by a function that smoothly transitions from $\alpha_0 \approx 1$ towards $\alpha_T \approx 0$. A special case of noising process is the variance preserving process (Sohl-Dickstein et al., 2015; Ho et al., 2020) for which $\alpha_t = \sqrt{1 - \sigma_t^2}$. Following Kingma et al. (2021), we define the signal to noise ratio $\text{SNR}(t) = \alpha_t^2 / \sigma_t^2$, which simplifies notations. This diffusion process is Markov and can be equivalently written with transition distributions as:

$$q(\mathbf{z}_t | \mathbf{z}_s) = \mathcal{N}(\mathbf{z}_t | \alpha_{t|s} \mathbf{z}_s, \sigma_{t|s}^2 \mathbf{I}), \quad (2)$$

for any $t > s$ with $\alpha_{t|s} = \alpha_t / \alpha_s$ and $\sigma_{t|s}^2 = \sigma_t^2 - \alpha_{t|s}^2 \sigma_s^2$. The entire noising process is then written as:

$$q(\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_T | \mathbf{x}) = q(\mathbf{z}_0 | \mathbf{x}) \prod_{t=1}^T q(\mathbf{z}_t | \mathbf{z}_{t-1}). \quad (3)$$

The posterior of the transitions conditioned on \mathbf{x} gives the inverse of the noising process, the *true denoising process*. It is also normal and given by:

$$q(\mathbf{z}_s | \mathbf{x}, \mathbf{z}_t) = \mathcal{N}(\mathbf{z}_s | \mu_{t \rightarrow s}(\mathbf{x}, \mathbf{z}_t), \sigma_{t \rightarrow s}^2 \mathbf{I}), \quad (4)$$

where the definitions for $\mu_{t \rightarrow s}(\mathbf{x}, \mathbf{z}_t)$ and $\sigma_{t \rightarrow s}$ can be analytically obtained as

$$\mu_{t \rightarrow s}(\mathbf{x}, \mathbf{z}_t) = \frac{\alpha_{t|s} \sigma_s^2}{\sigma_t^2} \mathbf{z}_t + \frac{\alpha_s \sigma_{t|s}^2}{\sigma_t^2} \mathbf{x} \quad \text{and} \quad \sigma_{t \rightarrow s} = \frac{\sigma_{t|s} \sigma_s}{\sigma_t}.$$

The Generative Denoising Process In contrast to other generative models, in diffusion models, the generative process is defined with respect to the *true denoising process*. The variable \mathbf{x} , which is unknown to the generative process, is replaced by an approximation $\hat{\mathbf{x}} = \phi(\mathbf{z}_t, t)$ given by a neural network ϕ . Then the generative transition distribution

$p(\mathbf{z}_s | \mathbf{z}_t)$ is chosen to be $q(\mathbf{z}_s | \hat{\mathbf{x}}(\mathbf{z}_t, t), \mathbf{z}_t)$. Similarly to Eq. 4, it can be expressed using the approximation $\hat{\mathbf{x}}$ as:

$$p(\mathbf{z}_s | \mathbf{z}_t) = \mathcal{N}(\mathbf{z}_s | \mu_{t \rightarrow s}(\hat{\mathbf{x}}, \mathbf{z}_t), \sigma_{t \rightarrow s}^2 \mathbf{I}). \quad (5)$$

With the choice $s = t - 1$, a variational lower bound on the log-likelihood of \mathbf{x} given the generative model is given by:

$$\log p(\mathbf{x}) \geq \mathcal{L}_0 + \mathcal{L}_{\text{base}} + \sum_{t=1}^T \mathcal{L}_t, \quad (6)$$

where $\mathcal{L}_0 = \log p(\mathbf{x} | \mathbf{z}_0)$ models the likelihood of the data given \mathbf{z}_0 , $\mathcal{L}_{\text{base}} = -\text{KL}(q(\mathbf{z}_T | \mathbf{x}) || p(\mathbf{z}_T))$ models the distance between a standard normal distribution and the final latent variable $q(\mathbf{z}_T | \mathbf{x})$, and

$$\mathcal{L}_t = -\text{KL}(q(\mathbf{z}_s | \mathbf{x}, \mathbf{z}_t) || p(\mathbf{z}_s | \mathbf{z}_t)) \quad \text{for } t = 1, \dots, T.$$

While in this formulation the neural network directly predicts $\hat{\mathbf{x}}$, Ho et al. (2020) found that optimization is easier when predicting the Gaussian noise instead. Intuitively, the network is trying to predict which part of the observation \mathbf{z}_t is noise originating from the diffusion process, and which part corresponds to the underlying data point \mathbf{x} . Specifically, if $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$, then the neural network ϕ outputs $\hat{\epsilon} = \phi(\mathbf{z}_t, t)$, so that:

$$\hat{\mathbf{x}} = (1/\alpha_t) \mathbf{z}_t - (\sigma_t / \alpha_t) \hat{\epsilon} \quad (7)$$

As shown in (Kingma et al., 2021), with this parametrization \mathcal{L}_t simplifies to:

$$\mathcal{L}_t = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \mathbf{I})} \left[\frac{1}{2} (1 - \text{SNR}(t-1)/\text{SNR}(t)) \|\epsilon - \hat{\epsilon}\|^2 \right] \quad (8)$$

In practice the term $\mathcal{L}_{\text{base}}$ is close to zero when the noising schedule is defined in such a way that $\alpha_T \approx 0$. Furthermore, if $\alpha_0 \approx 1$ and \mathbf{x} is discrete, then \mathcal{L}_0 is close to zero as well.

2.2. Equivariance

A function f is said to be equivariant to the action of a group G if $T_g(f(\mathbf{x})) = f(S_g(\mathbf{x}))$ for all $g \in G$, where S_g, T_g are linear representations related to the group element g (Serre, 1977). In this work, we consider the Euclidean group $E(3)$ generated by translations, rotations and reflections, for which S_g and T_g can be represented by a translation \mathbf{t} and an orthogonal matrix \mathbf{R} that rotates or reflects coordinates. f is then equivariant to a rotation or reflection \mathbf{R} if transforming its input results in an equivalent transformation of its output, or $\mathbf{R}f(\mathbf{x}) = f(\mathbf{Rx})$.

Equivariant Distributions and Diffusion In our setting, a conditional distribution $p(y | x)$ is equivariant to the action of rotations and reflections when

$$p(y | x) = p(\mathbf{R}y | \mathbf{Rx}) \quad \text{for all orthogonal } \mathbf{R}. \quad (9)$$

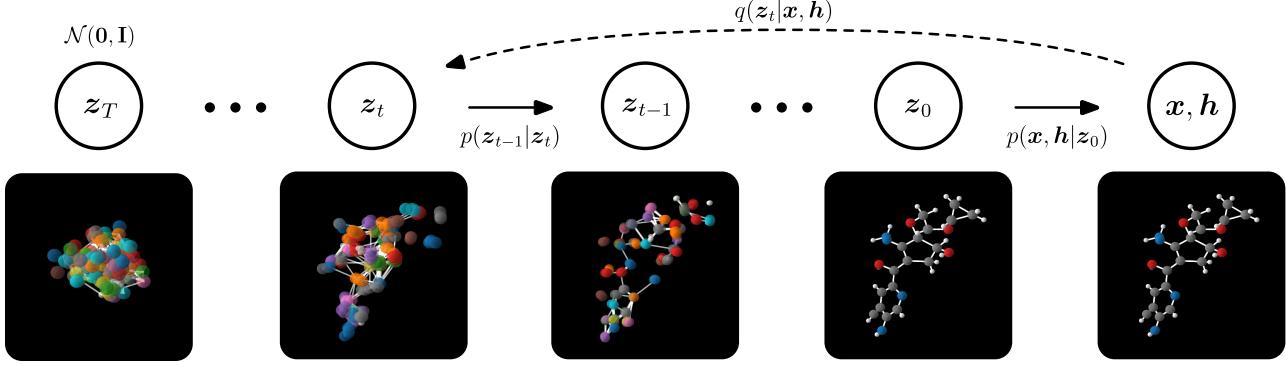


Figure 2. Overview of the Equivariant Diffusion Model. To generate molecules, coordinates \mathbf{x} and features \mathbf{h} are generated by denoising variables \mathbf{z}_t starting from standard normal noise \mathbf{z}_T . This is achieved by sampling from the distributions $p(\mathbf{z}_{t-1}|\mathbf{z}_t)$ iteratively. To train the model, noise is added to a datapoint \mathbf{x}, \mathbf{h} using $q(\mathbf{z}_t|\mathbf{x}, \mathbf{h})$ for the step t of interest, which the network then learns to denoise.

A distribution is invariant to \mathbf{R} transformations if

$$p(y) = p(\mathbf{R}y) \quad \text{for all orthogonal } \mathbf{R}. \quad (10)$$

Köhler et al. (2020) showed that an invariant distribution composed with an equivariant invertible function results in an invariant distribution. Furthermore, Xu et al. (2022) proved that if $x \sim p(x)$ is invariant to a group and the transition probabilities of a Markov chain $y \sim p(y|x)$ are equivariant, then the marginal distribution of y at any time step is invariant to group transformations as well. This is helpful as it means that if $p(z_T)$ is invariant and the neural network used to parametrize $p(z_{t-1}|z_t)$ is equivariant, then the marginal distribution $p(\mathbf{x})$ of the denoising model will be an invariant distribution as desired.

Points and Features in E(3) In this paper, we consider point clouds $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_M) \in \mathbb{R}^{M \times 3}$ with corresponding features $\mathbf{h} = (\mathbf{h}_1, \dots, \mathbf{h}_M) \in \mathbb{R}^{M \times nf}$. The features \mathbf{h} are invariant to group transformations, and the positions are affected by rotations, reflections and translations as $\mathbf{R}\mathbf{x} + \mathbf{t} = (\mathbf{R}\mathbf{x}_1 + \mathbf{t}, \dots, \mathbf{R}\mathbf{x}_M + \mathbf{t})$ where \mathbf{R} is an orthogonal matrix¹. The function $(\mathbf{z}_x, \mathbf{z}_h) = f(\mathbf{x}, \mathbf{h})$ is $E(3)$ equivariant if for all orthogonal \mathbf{R} and $\mathbf{t} \in \mathbb{R}^3$ we have:

$$\mathbf{R}\mathbf{z}_x + \mathbf{t}, \mathbf{z}_h = f(\mathbf{R}\mathbf{x} + \mathbf{t}, \mathbf{h}) \quad (11)$$

E(n) Equivariant Graph Neural Networks (EGNNs) (Satorras et al., 2021b) are a type of Graph Neural Network that satisfies the equivariance constraint (11). In this work, we consider interactions between all atoms, and therefore assume a fully connected graph \mathcal{G} with nodes $v_i \in \mathcal{V}$. Each node v_i is endowed with coordinates $\mathbf{x}_i \in \mathbb{R}^3$ as well as features $\mathbf{h}_i \in \mathbb{R}^d$. In this setting, EGNN consists of the composition of Equivariant Convolutional Layers

¹As a matrix-multiplication the left-hand side would be written $\mathbf{x}\mathbf{R}^T$. Formally $\mathbf{R}\mathbf{x}$ can be seen as a group action of \mathbf{R} on \mathbf{x} .

$\mathbf{x}^{l+1}, \mathbf{h}^{l+1} = \text{EGCL}[\mathbf{x}^l, \mathbf{h}^l]$ which are defined as:

$$\begin{aligned} \mathbf{m}_{ij} &= \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, d_{ij}^2, a_{ij}), \quad \mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \sum_{j \neq i} \tilde{e}_{ij} \mathbf{m}_{ij}), \\ \mathbf{x}_i^{l+1} &= \mathbf{x}_i^l + \sum_{j \neq i} \frac{\mathbf{x}_i^l - \mathbf{x}_j^l}{d_{ij} + 1} \phi_x(\mathbf{h}_i^l, \mathbf{h}_j^l, d_{ij}^2, a_{ij}), \end{aligned} \quad (12)$$

where l indexes the layer, and $d_{ij} = \|\mathbf{x}_i^l - \mathbf{x}_j^l\|_2$ is the euclidean distance between nodes (v_i, v_j) , and a_{ij} are optional edge attributes. The difference $(\mathbf{x}_i^l - \mathbf{x}_j^l)$ in Equation 12 is normalized by $d_{ij} + 1$ as done in (Satorras et al., 2021a) for improved stability, as well as the attention mechanism which infers a soft estimation of the edges $\tilde{e}_{ij} = \phi_{inf}(\mathbf{m}_{ij})$. All learnable components (ϕ_e , ϕ_h , ϕ_x and ϕ_{inf}) are parametrized by fully connected neural networks (cf. Appendix B for details). An entire EGNN architecture is then composed of L EGCL layers which applies the following non-linear transformation $\hat{\mathbf{x}}, \hat{\mathbf{h}} = \text{EGNN}[\mathbf{x}^0, \mathbf{h}^0]$. This transformation satisfies the required equivariant property in Equation 11.

3. EDM: E(3) Equivariant Diffusion Model

In this section we describe EDM, an $E(3)$ Equivariant Diffusion Model. EDM defines a noising process on both node positions and features, and learns the generative *denoising* process using an equivariant neural network. We also determine the equations for log-likelihood computation.

3.1. The Diffusion Process

We first define an equivariant diffusion process for coordinates \mathbf{x}_i with atom features \mathbf{h}_i that adds noise to the data. Recall that we consider a set of points $\{(\mathbf{x}_i, \mathbf{h}_i)\}_{i=1, \dots, M}$, where each node has associated to it a coordinate representation $\mathbf{x}_i \in \mathbb{R}^n$ and an attribute vector $\mathbf{h}_i \in \mathbb{R}^{nf}$. Let $[\cdot, \cdot]$ denote a concatenation. We define the equivariant noising

Algorithm 1 Optimizing EDM

Input: Data point \mathbf{x} , neural network ϕ
 Sample $t \sim \mathcal{U}(0, \dots, T)$, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 Subtract center of gravity from $\epsilon^{(x)}$ in $\epsilon = [\epsilon^{(x)}, \epsilon^{(h)}]$
 Compute $\mathbf{z}_t = \alpha_t[\mathbf{x}, \mathbf{h}] + \sigma_t \epsilon$
 Minimize $\|\epsilon - \phi(\mathbf{z}_t, t)\|^2$

Algorithm 2 Sampling from EDM

Sample $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for t in $T, T-1, \dots, 1$ where $s = t-1$ **do**
 Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 Subtract center of gravity from $\epsilon^{(x)}$ in $\epsilon = [\epsilon^{(x)}, \epsilon^{(h)}]$
 $\mathbf{z}_s = \frac{1}{\alpha_{t|s}} \mathbf{z}_t - \frac{\sigma_{t|s}^2}{\alpha_{t|s} \sigma_t} \cdot \phi(\mathbf{z}_t, t) + \sigma_{t \rightarrow s} \cdot \epsilon$
end for
 Sample $\mathbf{x}, \mathbf{h} \sim p(\mathbf{x}, \mathbf{h} | \mathbf{z}_0)$

process on latent variables $\mathbf{z}_t = [\mathbf{z}_t^{(x)}, \mathbf{z}_t^{(h)}]$ as:

$$q(\mathbf{z}_t | \mathbf{x}, \mathbf{h}) = \mathcal{N}_{xh}(\mathbf{z}_t | \alpha_t[\mathbf{x}, \mathbf{h}], \sigma_t^2 \mathbf{I}) \quad (13)$$

for $t = 1, \dots, T$ where \mathcal{N}_{xh} is concise notation for the product of two distributions, one for the noised coordinates \mathcal{N}_x and another for the noised features \mathcal{N} given by:

$$\mathcal{N}_x(\mathbf{z}_t^{(x)} | \alpha_t \mathbf{x}, \sigma_t^2 \mathbf{I}) \cdot \mathcal{N}(\mathbf{z}_t^{(h)} | \alpha_t \mathbf{h}, \sigma_t^2 \mathbf{I}) \quad (14)$$

These equations correspond to Equation 1 in a standard diffusion model. Also, a slight abuse of notation is used to aid readability: technically \mathbf{x} , \mathbf{h} , \mathbf{z}_t are two-dimensional variables with an axis for the point identifier and an axis for the features. However, in the distributions they are treated as if flattened to a vector.

As explained in (Satorras et al., 2021a) it is impossible to have a non-zero distribution that is invariant to translations, since it cannot integrate to one. However, one can use distributions on the linear subspace where the center of gravity is always zero. Following (Xu et al., 2022) that showed that such a linear subspace can be used consistently in diffusion, \mathcal{N}_x is defined as a normal distribution on the subspace defined by $\sum_i \mathbf{x}_i = \mathbf{0}$ for which the precise definition is given in Appendix A.

Since the features \mathbf{h} are invariant to E(n) transformations, the noise distribution for these features can be the conventional normal distribution \mathcal{N} . Although similar to standard diffusion models, depending on whether data is categorical (for the atom type), ordinal (for atom charge), or continuous, different starting representations of \mathbf{h} may be desirable and require different treatment in \mathcal{L}_0 , on which we will expand in Section 3.3.

The Generative Denoising Process To define the generative process, the noise posteriors $q(\mathbf{z}_s | \mathbf{x}, \mathbf{h}, \mathbf{z}_t)$ of Equation 13 can be used in the same fashion as in Equation 4 by

replacing the data variables \mathbf{x}, \mathbf{h} by neural network approximations $\hat{\mathbf{x}}, \hat{\mathbf{h}}$:

$$p(\mathbf{z}_s | \mathbf{z}_t) = \mathcal{N}_{xh}(\mathbf{z}_s | \mu_{t \rightarrow s}([\hat{\mathbf{x}}, \hat{\mathbf{h}}], \mathbf{z}_t), \sigma_{t \rightarrow s}^2 \mathbf{I}) \quad (15)$$

where $\hat{\mathbf{x}}, \hat{\mathbf{h}}$ depend on \mathbf{z}_t, t and the neural network ϕ . As conventional in modern diffusion models, we use the noise parametrization to obtain $\hat{\mathbf{x}}, \hat{\mathbf{h}}$. Instead of directly predicting them, the network ϕ outputs $\hat{\epsilon} = [\hat{\epsilon}^{(x)}, \hat{\epsilon}^{(h)}]$ which is then used to compute:

$$[\hat{\mathbf{x}}, \hat{\mathbf{h}}] = \mathbf{z}_t / \alpha_t - \hat{\epsilon}_t \cdot \sigma_t / \alpha_t \quad (16)$$

If $\hat{\epsilon}_t$ is computed by an equivariant function ϕ then the denoising distribution in Equation 15 is equivariant. To see this, observe that rotating \mathbf{z}_t to $\mathbf{R}\mathbf{z}_t$ gives $\mathbf{R}\hat{\epsilon}_t = \phi(\mathbf{R}\mathbf{z}_t, t)$. Furthermore, the mean of the denoising equation rotates $\mathbf{R}\hat{\mathbf{x}} = \mathbf{R}\mathbf{z}_t^{(x)} / \alpha_t - \mathbf{R}\hat{\epsilon}_t^{(x)} \sigma_t / \alpha_t$ and since the noise is isotropic, the distribution is equivariant as desired.

To sample from the model, one first samples $\mathbf{z}_T \sim \mathcal{N}_{xh}(\mathbf{0}, \mathbf{I})$ and then iteratively samples $\mathbf{z}_{t-1} \sim p(\mathbf{z}_{t-1} | \mathbf{z}_t)$ for $t = T, \dots, 1$ and then finally samples $\mathbf{x}, \mathbf{h} \sim p(\mathbf{x}, \mathbf{h} | \mathbf{z}_0)$, as described in Algorithm 2.

Optimization Objective Recall that a likelihood term of this model is given by $\mathcal{L}_t = -\text{KL}(q(\mathbf{z}_s | \mathbf{x}, \mathbf{z}_t) || p(\mathbf{z}_s | \mathbf{z}_t))$. Analogous to Equation 8, in this parametrization the term simplifies to:

$$\mathcal{L}_t = \mathbb{E}_{\epsilon_t \sim \mathcal{N}_{xh}(0, \mathbf{I})} \left[\frac{1}{2} w(t) \|\epsilon_t - \hat{\epsilon}_t\|^2 \right], \quad (17)$$

where $w(t) = (1 - \text{SNR}(t-1)/\text{SNR}(t))$ and $\hat{\epsilon}_t = \phi(\mathbf{z}_t, t)$. This is convenient: even though parts of the distribution of \mathcal{N}_{xh} operate on a subspace, the simplification in Equation 8 also holds here, and can be computed for all components belonging to \mathbf{x} and \mathbf{h} at once. There are three reason why this simplification remains true: firstly, \mathcal{N}_x and \mathcal{N} within \mathcal{N}_{xh} are independent, so the divergence can be separated into two divergences. Further, the KL divergence between the \mathcal{N}_x components are still compatible with the standard KL equation for normal distributions, as they rely on a Euclidean distance (which is rotation invariant) and the distributions are isotropic with equal variance. Finally, because of the similarity in KL equations, the results can be combined again by concatenating the components in \mathbf{x} and \mathbf{h} . For a more detailed argument see Appendix A. An overview of the optimization procedure is given in Algorithm 1.

Following (Ho et al., 2020) during training we set $w(t) = 1$ as it stabilizes training and it is known to improve sample quality for images. Experimentally we also found this to hold true for molecules: even when evaluating the probabilistic variational objective for which $w(t) = (1 - \text{SNR}(t-1)/\text{SNR}(t))$, the model trained with $w(t) = 1$ outperformed models trained with the variational $w(t)$.

In summary, we have defined a diffusion process, a denoising model and an optimization objective between them. To

further specify our model, we need to define the neural network ϕ that is used within the denoising model.

3.2. The Dynamics

We learn the E(n) equivariant dynamics function $[\hat{\epsilon}_t^{(x)}, \hat{\epsilon}_t^{(h)}] = \phi(\mathbf{z}_t^{(x)}, \mathbf{z}_t^{(h)}, t)$ of the diffusion model using the equivariant network EGNN introduced in Section 2.2 in the following way:

$$\hat{\epsilon}_t^{(x)}, \hat{\epsilon}_t^{(h)} = \text{EGNN}(\mathbf{z}_t^{(x)}, [\mathbf{z}_t^{(h)}, t/T]) - [\mathbf{z}_t^{(x)}, \mathbf{0}]$$

Notice that we simply input $\mathbf{z}_t^{(x)}, \mathbf{z}_t^{(h)}$ to the EGNN with the only difference that t/T is concatenated to the node features. The estimated noise $\hat{\epsilon}_t^{(x)}$ is given by the output of the EGNN from which the input coordinates $\mathbf{z}_t^{(x)}$ are removed. Importantly, since the outputs have to lie on a zero center of gravity subspace, the component $\hat{\epsilon}_t^{(x)}$ is projected down by subtracting its center of gravity. This then satisfies the rotational and reflection equivariance on \hat{x} with the parametrization in Equation 16.

3.3. The Zeroth Likelihood Term

In typical diffusion models (Ho et al., 2020), the data being modelled is ordinal which makes the design of $\mathcal{L}_0^{(h)} = \log p(\mathbf{h}|\mathbf{z}_0^{(h)})$ relatively simple. Specifically, under very small noise perturbations of the original data distribution $p_{\text{data}}(\mathbf{h})$ (when $\alpha_0 \approx 1$ and $\sigma_0 \approx 0$) we have

$$q(\mathbf{h}|\mathbf{z}_0^{(h)}) = \frac{q(\mathbf{z}_0^{(h)}|\mathbf{h})p_{\text{data}}(\mathbf{h})}{\sum_{\mathbf{h}} q(\mathbf{z}_0^{(h)}|\mathbf{h})p_{\text{data}}(\mathbf{h})} \approx 1,$$

when $\mathbf{z}_0^{(h)}$ is sampled from the noising process $q(\mathbf{z}_0^{(h)}|\mathbf{h})$. Because $q(\mathbf{z}_0^{(h)}|\mathbf{h})$ is a highly peaked distribution, in practice it tends to zero for all but one single discrete state of \mathbf{h} . Furthermore, p_{data} is constant over this small peak, and thus $q(\mathbf{h}|\mathbf{z}_0^{(h)}) \approx 1$ when \mathbf{h} is the closest integer value to $\mathbf{z}_0^{(h)}$. This can be used to model integer molecular properties such as the atom charge. Following standard practice we let:

$$p(\mathbf{h}|\mathbf{z}_0^{(h)}) = \int_{\mathbf{h}-\frac{1}{2}}^{\mathbf{h}+\frac{1}{2}} \mathcal{N}(\mathbf{u}|\mathbf{z}_0^{(h)}, \sigma_0) d\mathbf{u}, \quad (18)$$

which most likely equals 1 for reasonable noise parameters α_0, σ_0 and it computed as $\Phi((\mathbf{h} + \frac{1}{2} - \mathbf{z}_0^{(h)})/\sigma_0) - \Phi((\mathbf{h} - \frac{1}{2} - \mathbf{z}_0^{(h)})/\sigma_0)$ where Φ is the CDF of a standard normal distribution. For categorical features such as the atom types, this model would however introduce an undesired bias.

Categorical features For categorical features such as the atom type, the aforementioned integer representation is unnatural and introduces bias. Instead of using integers

for these features, we operate directly on a one-hot representation. Suppose \mathbf{h} is an array whose values represent categories in $\{c_1, \dots, c_d\}$ such as atom types. Then \mathbf{h} is encoded with a one-hot function $\mathbf{h} \mapsto \mathbf{h}^{\text{onehot}}$ such that $\mathbf{h}_{i,j}^{\text{onehot}} = \mathbb{1}_{h_i=c_j}$. The noising process over $\mathbf{z}_t^{(h)}$ can then directly be defined using the one-hot representation $\mathbf{h}^{\text{onehot}}$ equivalent to its definition for integer values, i.e. $q(\mathbf{z}_t^{(h)}|\mathbf{h}) = \mathcal{N}(\mathbf{z}_t^{(h)}|\alpha_t \mathbf{h}^{\text{onehot}}, \sigma_t^2 \mathbf{I})$ with the only difference that $\mathbf{z}_t^{(h)}$ has an additional dimension axis with the size equal to the number of categories. Since the data is discrete and the noising process is assumed to be well defined, by the same reasoning as for integer data we can define probability parameters \mathbf{p} to be proportional to the normal distribution integrated from $1 - \frac{1}{2}$ to $1 + \frac{1}{2}$. Intuitively, when a small amount of noise is sampled and added to the one-hot representation, then the value corresponding to the active class will almost certainly be between $1 - \frac{1}{2}$ and $1 + \frac{1}{2}$:

$$p(\mathbf{h}|\mathbf{z}_0^{(h)}) = \mathcal{C}(\mathbf{h}|\mathbf{p}), \mathbf{p} \propto \int_{1-\frac{1}{2}}^{1+\frac{1}{2}} \mathcal{N}(\mathbf{u}|\mathbf{z}_0^{(h)}, \sigma_0) d\mathbf{u}$$

where \mathbf{p} is normalized to sum to one and \mathcal{C} is a categorical distribution. In practice this distribution will almost certainly equal one for values $\mathbf{z}_0^{(h)}$ that were sampled from the diffusion process given \mathbf{h} .

Continuous positions For continuous positions, defining $\mathcal{L}_0^{(x)} = \log p(\mathbf{x}|\mathbf{z}_0^{(x)})$ is a little more involved than for discrete features. A similar analysis assuming $p_{\text{data}}(\mathbf{x})$ is constant results in:

$$q(\mathbf{x}|\mathbf{z}_0^{(x)}) = \frac{q(\mathbf{z}_0^{(x)}|\mathbf{x})p_{\text{data}}(\mathbf{x})}{\int_{\mathbf{x}} q(\mathbf{z}_0^{(x)}|\mathbf{x})p_{\text{data}}(\mathbf{x})} \approx \frac{q(\mathbf{z}_0^{(x)}|\mathbf{x})}{\int_{\mathbf{x}} q(\mathbf{z}_0^{(x)}|\mathbf{x})},$$

which by completing the square we find is equal to $\mathcal{N}_x(\mathbf{x}|\mathbf{z}_0^{(x)}/\alpha_0, \sigma_0^2/\alpha_0^2 \mathbf{I})$. Empirically, we find that our model achieves better likelihood performance, especially with lower SNR rates, when we use $\hat{\mathbf{x}}$ as a prediction for the mean. After simplifying the terms it turns out that this essentially adds an additional correction term containing the estimated noise $\hat{\epsilon}_0^{(x)}$, which originates from Equation 16 and can be written as:

$$p(\mathbf{x}|\mathbf{z}_0) = \mathcal{N}\left(\mathbf{x}|\mathbf{z}_0^{(x)}/\alpha_0 - \sigma_0/\alpha_0 \hat{\epsilon}_0, \sigma_0^2/\alpha_0^2 \mathbf{I}\right). \quad (19)$$

When this parametrization is chosen the log likelihood component $\mathcal{L}_0^{(x)}$ can be re-written to:

$$\mathcal{L}_0^{(x)} = \mathbb{E}_{\epsilon^{(x)} \sim \mathcal{N}_x(0, \mathbf{I})} \left[\log Z^{-1} - \frac{1}{2} \|\epsilon^{(x)} - \phi^{(x)}(\mathbf{z}_0, 0)\|^2 \right],$$

with the normalization constant Z . Conveniently, this allows Equation 17 for losses \mathcal{L}_t to also be used for $t = 0$ in its \mathbf{x} component, by defining $w(0) = -1$. The normalization constant then has to be added separately. This normalization constant $Z = (\sqrt{2\pi} \cdot \sigma_0/\alpha_0)^{(M-1)\cdot n}$ where the $(M-1) \cdot n$

arises from the zero center of gravity subspace is described in Appendix A.

Scaling Features Since coordinates, atom types and charges represent different physical quantities, we can define a relative scaling between them. While normalizing the features simply makes them easier to process for the neural network, the relative scaling has a deeper impact on the model: when the features \mathbf{h} are defined on a smaller scale than the coordinates \mathbf{x} , the denoising process tends to first determine rough positions and decide on the atom types only afterwards. Whereas scaling \mathbf{x} requires a correction in the log-likelihood since it is continuous, scaling \mathbf{h} does not require a correction and is not problematic as long as the difference in discrete values is large compared to σ_0 . We find empirically that defining the input to our EDM model as $[\mathbf{x}, 0.25 \mathbf{h}^{\text{onehot}}, 0.1 \mathbf{h}^{\text{atom charge}}]$ significantly improves performance over non-scaled inputs.

Number of Atoms In the above sections we have considered the number of atoms M to be known beforehand. To adapt to different molecules with different sizes, we compute the categorical distribution $p(M)$ of molecule sizes on the training set. To sample from the model $p(\mathbf{x}, \mathbf{h}, M)$, $M \sim p(M)$ is first sampled and then $\mathbf{x}, \mathbf{h} \sim p(\mathbf{x}, \mathbf{h}|M)$ are sampled from the EDM. For clarity this conditioning on M is often omitted, but it remains an important part of the generative process and likelihood computation.

3.4. Conditional generation

In this section we describe a straightforward extension to the proposed method to do conditional generation $\mathbf{x}, \mathbf{h} \sim p(\mathbf{x}, \mathbf{h}|c)$ given some desired property c . We can define the optimization lower bound for the conditional case as $\log p(\mathbf{x}, \mathbf{h}|c) \geq \mathcal{L}_{c,0} + \mathcal{L}_{c,\text{base}} + \sum_{t=1}^T \mathcal{L}_{c,t}$, where the different $\mathcal{L}_{c,t}$ for $1 \leq t < T - 1$ are defined similarly to Equation 17, with the important difference that the function $\hat{\epsilon}_t = \phi(\mathbf{z}_t, [t, c])$ takes as additional input a property c which is concatenated to the nodes features. Given a trained conditional model we define the generative process by first sampling the number of nodes M and a property value c from a parametrized distribution $c, M \sim p(c, M)$ defined in Appendix E. Next, we can generate molecules \mathbf{x}, \mathbf{h} given c, M using our Conditional EDM $\mathbf{x}, \mathbf{h} \sim p(\mathbf{x}, \mathbf{h}|c, M)$.

4. Related Work

Diffusion models (Sohl-Dickstein et al., 2015) are generative models that have recently been connected to score-based methods via denoising diffusion models (Song & Ermon, 2019; Ho et al., 2020). This new family of generative models has proven to be very effective for the generation of data such as images (Ho et al., 2020; Nichol & Dhariwal, 2021).

Some recent methods directly generate molecules in 3D:

(Gebauer et al., 2019; Luo & Ji, 2021; Luo et al., 2021a; Gebauer et al., 2021) define an order-dependent autoregressive distribution from which atoms are iteratively sampled. (Ragoza et al., 2020) maps atoms to a fixed grid and trains a VAE using 3D convolutions. E-NF (Satorras et al., 2021a) defines an equivariant normalizing flow that integrates a differential equation. Instead, our method learns to denoise a diffusion process, which scales better during training.

A related branch of literature is concerned by solely predicting coordinates from molecular graphs, referred to as the conformation. Examples of such methods utilize conditional VAEs (Simm & Hernández-Lobato, 2019), Wasserstein GANs (Hoffmann & Noé, 2019), and normalizing flows (Noé et al., 2019), with adaptions for Euclidean symmetries in (Köhler et al., 2020; Xu et al., 2021a; Simm et al., 2021; Ganea et al., 2021; Guan et al., 2022) resulting in performance improvements. In recent works (Shi et al., 2021; Luo et al., 2021b; Xu et al., 2022) it was shown that score-based and diffusion models are effective at coordinate prediction, especially when the underlying neural network respects the symmetries of the data. Our work can be seen as an extension of these methods that incorporates discrete atom features, and furthermore derives the equations required for log-likelihood computation. In the context of diffusion for discrete variables, unrelated to molecule modelling, discrete diffusion processes have been proposed (Sohl-Dickstein et al., 2015; Hoogeboom et al., 2021; Austin et al., 2021). However, for 3D molecule generation these would require a separate diffusion process for the discrete features and the continuous coordinates. Instead we define a joint process for both of them.

Tangentially related, other methods generate molecules in graph representation. Some examples are autoregressive methods such as (Liu et al., 2018; You et al., 2018; Liao et al., 2019), and one-shot approaches such as (Simonovsky & Komodakis, 2018; De Cao & Kipf, 2018; Bresson & Laurent, 2019; Kosiorek et al., 2020; Krawczuk et al., 2021). However such methods do not provide conformer information which is useful for many downstream tasks.

5. Experiments

5.1. Molecule Generation — QM9

QM9 (Ramakrishnan et al., 2014) is a standard dataset that contains molecular properties and atom coordinates for 130k small molecules with up to 9 heavy atoms (29 atoms including hydrogens). In this experiment we train EDM to unconditionally generate molecules with 3-dimensional coordinates, atom types (H, C, N, O, F) and integer-valued atom charges. We use the train/val/test partitions introduced in (Anderson et al., 2019), which consists of 100K/18K/13K samples respectively for each partition.

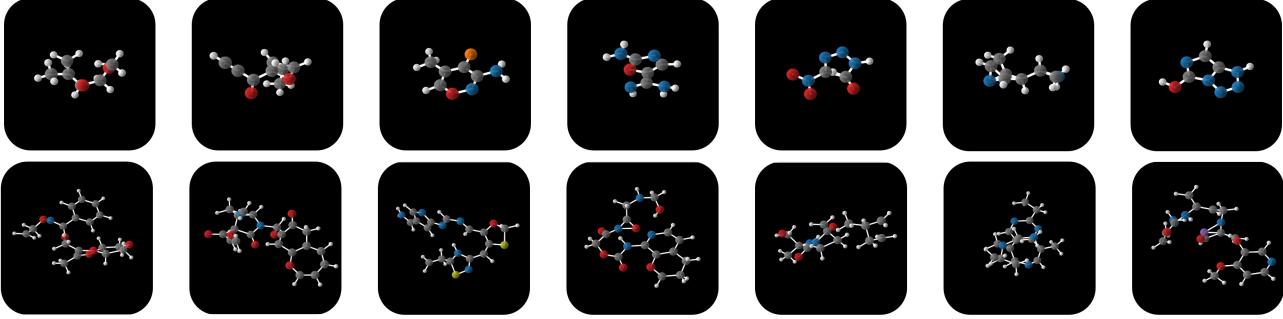


Figure 3. Selection of samples generated by the denoising process of our EDM trained on QM9 (up) and GEOM-DRUGS (down).

Table 1. Neg. log-likelihood – $\log p(\mathbf{x}, \mathbf{h}, M)$, atom stability and molecule stability with standard deviations across 3 runs on QM9, each drawing 10000 samples from the model.

# Metrics	NLL	Atom stable (%)	Mol stable (%)
E-NF	-59.7	85.0	4.9
G-Schnet	N.A	95.7	68.1
GDM	-94.7	97.0	63.2
GDM-aug	-92.5	97.6	71.6
EDM (ours)	-110.7\pm1.5	98.7\pm0.1	82.0\pm0.4
Data		99.0	95.2

Metrics Following (Satorras et al., 2021a), we use the distance between pairs of atoms and the atom types to predict bond types (single, double, triple or none). We then measure atom stability (the proportion of atoms that have the right valency) and molecule stability (the proportion of generated molecules for which all atoms are stable).

Baselines: We compare EDM to two existing E(3) equivariant models: G-Schnet (Gebauer et al., 2019) and Equivariant Normalizing Flows (E-NF) (Satorras et al., 2021a). For G-Schnet we extracted 10000 samples from the publicly available code to run the analysis. In order to demonstrate the benefits of equivariance, we also perform an ablation study and run a non-equivariant variation of our method that we call Graph Diffusion Models (GDM). The Graph diffusion model is run with the same configuration as our method, except that the EGNN is replaced by a non-equivariant graph network defined in Appendix C. We also experiment with GDM-aug, where the GDM model is trained on data augmented with random rotations. All models use 9 layers, 256 features per layer and SiLU activations. They are trained using Adam with batch size 64 and learning rate 10^{-4} .

Results are reported in Table 1. Our method outperforms previous methods (E-NF and G-Schnet), as well as its non-equivariant counterparts on all metrics. It is interesting to note that the negative log-likelihood of the EDM is much lower than other models, which indicates that it is able to create sharper peaks in the model distribution.

Further, EDMs are compared to one-shot graph-based molecule generation models that do not operate on 3D

Table 2. Validity and uniqueness over 10000 molecules with standard deviation across 3 runs. Results marked (*) are not directly comparable, as they do not use 3D coordinates to derive bonds. H: model hydrogens explicitly

Method	H	Valid (%)	Valid and Unique (%)
Graph VAE (*)		55.7	42.3
GTVAE (*)		74.6	16.8
Set2GraphVAE (*)		59.9 ± 1.7	56.2 ± 1.4
EDM (ours)		97.5 ± 0.2	94.3 ± 0.2
E-NF	✓	40.2	39.4
G-Schnet	✓	85.5	80.3
GDM-aug	✓	90.4	89.5
EDM (ours)	✓	91.9 ± 0.5	90.7 ± 0.6
Data	✓	97.7	97.7

coordinates: GraphVAE (Simonovsky & Komodakis, 2018), GraphTransformerVAE (Mitton et al., 2021), and Set2GraphVAE (Vignac & Frossard, 2021). For G-Schnet and EDM, the bonds are directly derived from the distance between atoms. We report validity (as measured by RDKit) and uniqueness of the generated compounds. Following (Vignac & Frossard, 2021) novelty is not included here. For a discussion on the issues with the novelty metric, see Appendix C. As can be seen in Table 2, the EDM is able to generate a very high rate of valid and unique molecules. This is impressive since the 3D models are at a disadvantage in this metric, as the rules to derive bonds are very strict. Interestingly, even when including hydrogen atoms in the model, the performance of the EDM does not deteriorate much. A possible explanation is that the equivariant diffusion model scales effectively and learn very precise distributions, as evidenced by the low negative log-likelihood.

5.2. Conditional Molecule Generation

In this section, we aim to generate molecules targeting some desired properties. This can be of interest towards the process of drug discovery where we need to obtain molecules that satisfy specific properties. We train our conditional diffusion model from Section 3.4 in QM9 conditioning the generation on properties α , gap, homo, lumo, μ and C_v described in more detail in Appendix E. In order to assess

Equivariant Diffusion for Molecule Generation in 3D

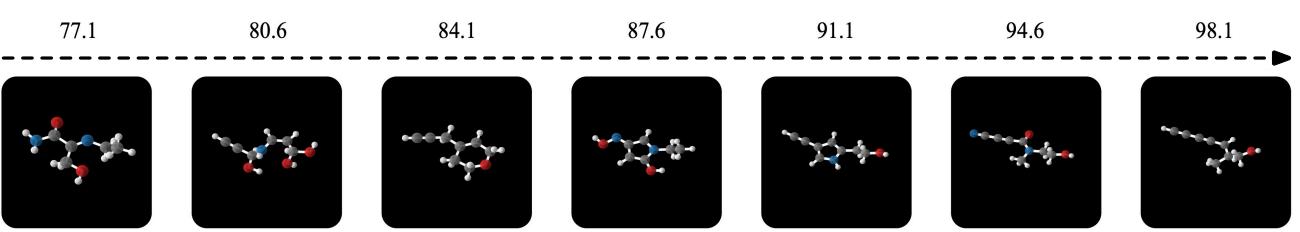


Figure 4. Generated molecules by our Conditional EDM when interpolating among different Polarizability α values with the same reparametrization noise ϵ . Each α value is provided on top of each image.

Table 3. Mean Absolute Error for molecular property prediction by a EGNN classifier ϕ_c on a QM9 subset, EDM generated samples and two different baselines "Naive (U-bounds)" and "# Atoms".

Task Units	α	$\Delta\epsilon$	ϵ_{HOMO}	ϵ_{LUMO}	μ	C_v
Naive (U-bound)	9.01	1470	645	1457	1.616	6.857
#Atoms	3.86	866	426	813	1.053	1.971
EDM	2.76	655	356	584	1.111	1.101
QM9 (L-bound)	0.10	64	39	36	0.043	0.040

the quality of the generated molecules w.r.t. to their conditioned property, we use the property classifier network ϕ_c from Satorras et al. (2021b). We split the QM9 training partition into two halves D_a, D_b of 50K samples each. The classifier ϕ_c is trained on the first half D_a , while the Conditional EDM is trained on the second half D_b . Then, ϕ_c is evaluated on the EDM conditionally generated samples. We also report the loss of ϕ_c on D_b as a lower bound named "QM9 (L-bound)". The better EDM approximates D_b the smaller the gap between "EDM" and "QM9 (L-bound)". Further implementation details are reported in Appendix E.

Baselines: We provide two baselines in which molecules are to some extent agnostic to their respective property c . In the first baseline we simply remove any relation between molecule and property by shuffling the property labels in D_b and then evaluating ϕ_c on it. We name this setting "Naive (Upper-Bound)". The second baseline named "#Atoms" predicts the molecular properties in D_b by only using the number of atoms in the molecule. If "EDM" overcomes "Naive (Upper-Bound)" it should be able to incorporate conditional property information into the generated molecules. If it overcomes "#Atoms" it should be able to incorporate it into the molecular structure beyond the number of atoms.

Results (quantitative): Results are reported in Table 3. EDM outperforms both "Naive (U-bound)" and "#Atoms" baselines in all properties (except μ) indicating that it is able to incorporate property information into the generated molecules beyond the number of atoms for most properties. However, we can see there is still room for improvement by looking at the gap between "EDM" and "QM9 (L-bound)".

Results (qualitative): In Figure 4, we interpolate the conditional generation among different Polarizability values α while keeping the noise ϵ fixed. The Polarizability is the tendency of a molecule to acquire an electric dipole moment

Table 4. Neg. log-likelihood, atom stability and Wasserstein distance between generated and training set energy distributions.

# Metrics	NLL	Atom stability (%)	\mathcal{W}
GDM	- 14.2	75.0	3.32
GDM-aug	- 58.3	77.7	4.26
EDM	-137.1	81.3	1.41
Data		86.5	0.0

when subject to an external electric field. We can expect less isometrically shaped molecules for large α values. This is the obtained behavior in Figure 4 – we show that this behavior is consistent across different runs in Appendix E.

5.3. GEOM-Drugs

While QM9 features only small molecules, GEOM (Axelrod & Gomez-Bombarelli, 2020) is a larger scale dataset of molecular conformers. It features 430,000 molecules with up to 181 atoms and 44.4 atoms on average. For each molecule, many conformers are given along with their energy. From this dataset we retain the 30 lowest energy conformations for each molecule. The models learn to generate the 3D positions and atom types of these molecules. All models use 4 layers, 256 features per layer, and are trained using Adam with batch size 64 and learning rate 10^{-4} .

Since molecules in this dataset are bigger and have more complex structures, predicting the bond types using the atom types and the distance between atoms with lookup tables results in more errors than on QM9. For this reason, we only report the atom stability, which measures 86.5% stable atoms on the dataset. Intuitively, this metric describes the percentage of atoms that have bonds in typical ranges – ideally, generative models should generate a comparable number of stable atoms. We also measure the energy of generated compounds with the software used to generate the conformations of the GEOM dataset (Bannwarth et al., 2019). After computing the energies of sampled molecules and the dataset, we measure the Wasserstein distance between their histograms. In Table 4 we can see that the EDM outperforms its non-equivariant counterparts on all metrics. In particular, EDM is able to capture the energy distribution well, as can be seen on the histograms in Appendix C.

6. Conclusions

We presented EDM, an E(3) equivariant diffusion model for molecule generation in 3D. While previous non-autoregressive models mostly focused on very small molecules with up to 9 atoms, our model scales better and can generate valid conformations while explicitly modeling hydrogen atoms. We also evaluate our model on the larger GEOM-DRUGS dataset, setting the stage for models for drug-size molecule generation in 3D.

References

- Anderson, B., Hy, T. S., and Kondor, R. Cormorant: Covariant molecular neural networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL <https://proceedings.neurips.cc/paper/2019/file/03573b32b2746e6e8ca98b9123f2249b-Paper.pdf>.
- Austin, J., Johnson, D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34, 2021.
- Axelrod, S. and Gomez-Bombarelli, R. Geom: Energy-annotated molecular conformations for property prediction and molecular generation. *arXiv preprint arXiv:2006.05531*, 2020.
- Bannwarth, C., Ehlert, S., and Grimme, S. Gfn2-xtb—an accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions. *Journal of chemical theory and computation*, 15(3):1652–1671, 2019.
- Bresson, X. and Laurent, T. A two-step graph convolutional decoder for molecule generation. *arXiv preprint arXiv:1906.03412*, 2019.
- De Cao, N. and Kipf, T. Molgan: An implicit generative model for small molecular graphs. *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.
- Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3165–3176. PMLR, 2020.
- Fuchs, F., Worrall, D. E., Fischer, V., and Welling, M. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*, 2020.
- Ganea, O.-E., Pattanaik, L., Coley, C. W., Barzilay, R., Jensen, K. F., Green, W. H., and Jaakkola, T. S. Geomol: Torsional geometric generation of molecular 3d conformer ensembles. *arXiv preprint arXiv:2106.07802*, 2021.
- Gebauer, N. W., Gastegger, M., and Schütt, K. T. Generating equilibrium molecules with deep neural networks. *arXiv preprint arXiv:1810.11347*, 2018.
- Gebauer, N. W., Gastegger, M., and Schütt, K. T. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. *arXiv preprint arXiv:1906.00957*, 2019.
- Gebauer, N. W., Gastegger, M., Hessmann, S. S., Müller, K.-R., and Schütt, K. T. Inverse design of 3d molecular structures with conditional generative neural networks. *arXiv preprint arXiv:2109.04824*, 2021.
- Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.
- Guan, J., Qian, W. W., qiang liu, Ma, W.-Y., Ma, J., and Peng, J. Energy-inspired molecular conformation optimization. In *International Conference on Learning Representations*, 2022.
- Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.
- Hoffmann, M. and Noé, F. Generating valid euclidean distance matrices. *arXiv preprint arXiv:1910.03131*, 2019.
- Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34, 2021.
- Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.
- Kingma, D. P., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *arXiv preprint arXiv:2107.00630*, 2, 2021.

- Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. In *8th International Conference on Learning Representations, ICLR*, 2020.
- Köhler, J., Klein, L., and Noé, F. Equivariant flows: Exact likelihood generative learning for symmetric densities. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5361–5370. PMLR, 2020.
- Kosiorek, A. R., Kim, H., and Rezende, D. J. Conditional set generation with transformers. *Workshop on Object-Oriented Learning at ICML 2020*, 2020.
- Krawczuk, I., Abrantes, P., Loukas, A., and Cevher, V. Grgan: A geometric graph generative adversarial network, 2021. URL <https://openreview.net/forum?id=qiAxL3Xqx1o>.
- Liao, R., Li, Y., Song, Y., Wang, S., Nash, C., Hamilton, W. L., Duvenaud, D., Urtasun, R., and Zemel, R. S. Efficient graph generation with graph recurrent attention networks. *arXiv preprint arXiv:1910.00760*, 2019.
- Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. L. Constrained graph variational autoencoders for molecule design. *arXiv preprint arXiv:1805.09076*, 2018.
- Luo, S., Guan, J., Ma, J., and Peng, J. A 3d generative model for structure-based drug design. *Advances in Neural Information Processing Systems*, 34, 2021a.
- Luo, S., Shi, C., Xu, M., and Tang, J. Predicting molecular conformation via dynamic graph score matching. *Advances in Neural Information Processing Systems*, 34, 2021b.
- Luo, Y. and Ji, S. An autoregressive flow model for 3d molecular geometry generation from scratch. In *International Conference on Learning Representations*, 2021.
- Mitton, J., Senn, H. M., Wynne, K., and Murray-Smith, R. A graph vae and graph transformer approach to generating molecular graphs. *arXiv preprint arXiv:2104.04345*, 2021.
- Nichol, A. and Dhariwal, P. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021.
- Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457), 2019.
- Ragoza, M., Masuda, T., and Koes, D. R. Learning a continuous representation of 3d molecular structures with deep generative models. *arXiv preprint arXiv:2010.08687*, 2020.
- Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.
- Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. *CoRR*, abs/2202.00512, 2022.
- Satorras, V. G., Hoogeboom, E., Fuchs, F., Posner, I., and Welling, M. E(n) equivariant normalizing flows. *Advances in Neural Information Processing Systems*, 34, 2021a.
- Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021b.
- Serre, J.-P. *Linear representations of finite groups*, volume 42. Springer, 1977.
- Shi, C., Luo, S., Xu, M., and Tang, J. Learning gradient fields for molecular conformation generation. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML*, 2021.
- Simm, G. N. and Hernández-Lobato, J. M. A generative model for molecular distance geometry. *arXiv preprint arXiv:1909.11459*, 2019.
- Simm, G. N. C., Pinsler, R., Csányi, G., and Hernández-Lobato, J. M. Symmetry-aware actor-critic for 3d molecular design. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=jEYKjPE1xYN>.
- Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pp. 412–422. Springer, 2018.
- Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015.
- Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *CoRR*, abs/1907.05600, 2019. URL <http://arxiv.org/abs/1907.05600>.
- Thomas, N., Smidt, T., Kearnes, S. M., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *CoRR*, abs/1802.08219, 2018.
- Vignac, C. and Frossard, P. Top-n: Equivariant set and graph generation without exchangeability. *arXiv preprint arXiv:2110.02096*, 2021.

Xu, M., Wang, W., Luo, S., Shi, C., Bengio, Y., Gomez-Bombarelli, R., and Tang, J. An end-to-end framework for molecular conformation generation via bilevel programming. *arXiv preprint arXiv:2105.07246*, 2021a.

Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S., and Tang, J. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=PzcvxEMzvQC>.

Xu, Y., Song, Y., Garg, S., Gong, L., Shu, R., Grover, A., and Ermon, S. Anytime sampling for autoregressive models via ordered autoencoding. In *9th International Conference on Learning Representations, ICLR*, 2021b.

You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, 2018.

A. The zero center of gravity, normal distribution

Consider the Euclidean variable $\mathbf{x} \in \mathbb{R}^{M \times n}$ in the linear subspace $\sum_i \mathbf{x}_i = \mathbf{0}$. In other words, \mathbf{x} is a point cloud where its center of gravity is zero. One can place a normal distribution \mathcal{N}_x over this subspace and its likelihood can be expressed as:

$$\mathcal{N}_x(\mathbf{x}|\boldsymbol{\mu}, \sigma^2 \mathbf{I}) = (\sqrt{2\pi}\sigma)^{-(M-1)\cdot n} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x} - \boldsymbol{\mu}\|^2\right)$$

Here $\boldsymbol{\mu}$ also lies in the same subspace as \mathbf{x} . Also note a slight abuse of notation: $\mathbf{x}, \boldsymbol{\mu}$ are technically two-dimensional matrices but are treated in the distribution as single-dimensional (flattened) vectors. To sample from this distribution, there are multiple options. For instance, one could sample from a normal distribution with dimensionality $(M-1) \cdot n$ and then map the sample to the $M \cdot n$ dimensional ambient space so that its center of gravity equals zero. However there is an easier alternative: One can sample in the $M \cdot n$ dimensional ambient space directly, and subtract $\sum_i \mathbf{x}_i$. Because the normal distributions are isotropic (meaning its variance in any direction you pick is σ^2) this is equivalent to the aforementioned method. More detailed analysis are given in (Satorras et al., 2021a) and (Xu et al., 2022).

KL Divergence A standard KL divergence for between two isotropic normal distributions $q = \mathcal{N}(\boldsymbol{\mu}_1, \sigma_1^2 \mathbf{I})$ and $p = \mathcal{N}(\boldsymbol{\mu}_2, \sigma_2^2 \mathbf{I})$ is given by:

$$\text{KL}(q||p) = d \cdot \log \frac{\sigma_2}{\sigma_1} + \frac{1}{2} \left[\frac{d \cdot \sigma_1^2 + \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2}{\sigma_2^2} - d \right], \quad (20)$$

where d is the dimensionality of the distribution. Recall that in our case the diffusion and denoising process have the same variance $\sigma_{Q,s,t}^2$. If $\sigma_1 = \sigma_2 = \sigma$, then the KL divergence simplifies to:

$$\text{KL}(q||p) = \frac{1}{2} \left[\frac{\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2}{\sigma^2} \right]. \quad (21)$$

Suppose now that $\mathcal{N}_1(\tilde{\boldsymbol{\mu}}_1, \sigma \mathbf{I})$ and $\mathcal{N}_2(\tilde{\boldsymbol{\mu}}_2, \sigma \mathbf{I})$ are defined on a linear subspace, where the mean $\tilde{\boldsymbol{\mu}}$ is defined with respect to any coordinate system in the subspace. The KL divergence between these distributions then includes a term containing the Euclidean distance $\|\tilde{\boldsymbol{\mu}}_1 - \tilde{\boldsymbol{\mu}}_2\|^2$

Similar to the arguments in (Satorras et al., 2021a; Xu et al., 2022), an orthogonal transformation \mathbf{Q} can be constructed that maps an ambient space where $\sum_i \boldsymbol{\mu}_i = \mathbf{0}$ to the subspace in such a way that $\begin{bmatrix} \tilde{\boldsymbol{\mu}} \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}\boldsymbol{\mu}$. Observe that $\|\tilde{\boldsymbol{\mu}}\| = \left\| \begin{bmatrix} \tilde{\boldsymbol{\mu}} \\ \mathbf{0} \end{bmatrix} \right\| = \|\boldsymbol{\mu}\|$, and therefore $\|\tilde{\boldsymbol{\mu}}_1 - \tilde{\boldsymbol{\mu}}_2\|^2 = \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2$. This shows that Equation 21 can be consistently computed in the ambient space. This also shows an important caveat: in some diffusion models, different variances are used in the posterior of the diffusion process and the denoising process. In those cases one can see from Equation 20 that the divergence depends on the dimensionality of the subspace, not to be confused with the dimensionality of the ambient space.

The combined KL divergence for positions and features In the previous section we have shown that the KL divergence for distributions such as \mathcal{N}_x , can still be computed in the ambient space as long as standard deviations between two such distributions are the same. Let us now consider the combined KL divergence for distributions $q = \mathcal{N}_{xh}(\boldsymbol{\mu}_1, \sigma^2 \mathbf{I})$ and $p = \mathcal{N}_{xh}(\boldsymbol{\mu}_2, \sigma^2 \mathbf{I})$. Note that here the means consist of two parts $\boldsymbol{\mu} = [\boldsymbol{\mu}^{(x)}, \boldsymbol{\mu}^{(h)}]$ where the x part lies in a subspace and the h part is defined freely. The distributions factorize as $\mathcal{N}_{xh}(\boldsymbol{\mu}, \sigma^2 \mathbf{I}) = \mathcal{N}_x(\boldsymbol{\mu}^{(x)}, \sigma^2 \mathbf{I}) \cdot \mathcal{N}(\boldsymbol{\mu}^{(h)}, \sigma^2 \mathbf{I})$. Then the KL divergence simplifies as:

$$\begin{aligned} \text{KL}(q||p) &= \text{KL}\left(\mathcal{N}_x(\boldsymbol{\mu}_1^{(x)}, \sigma^2 \mathbf{I})||\mathcal{N}_x(\boldsymbol{\mu}_2^{(x)}, \sigma^2 \mathbf{I})\right) + \text{KL}\left(\mathcal{N}(\boldsymbol{\mu}_1^{(h)}, \sigma^2 \mathbf{I})||\mathcal{N}(\boldsymbol{\mu}_2^{(h)}, \sigma^2 \mathbf{I})\right) \\ &= \frac{1}{2} \left[\frac{\|\boldsymbol{\mu}_1^{(x)} - \boldsymbol{\mu}_2^{(x)}\|^2}{\sigma^2} \right] + \frac{1}{2} \left[\frac{\|\boldsymbol{\mu}_1^{(h)} - \boldsymbol{\mu}_2^{(h)}\|^2}{\sigma^2} \right] = \frac{1}{2} \left[\frac{\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2}{\sigma^2} \right]. \end{aligned} \quad (22)$$

Here we have used that products of independent distributions sum in their independent KL terms, and that the sum of the Euclidean distance of two vectors squared is equal to the squared Euclidean distance of the two vectors concatenated. In summary, even though parts of our distribution are defined on a linear subspace, all computation for the KL divergences is still consistent and does not require special treatment. This is however only valid under the condition that the variances of the denoising process and posterior noising process are the same.

B. Additional Details for the Method

Noise schedule: A diffusion process requires a definition for α_t, σ_t for $t = 0, \dots, T$. Since $\alpha_t = \sqrt{1 - \sigma_t^2}$, it suffices to define α_t . The values should monotonically decrease, starting $\alpha_0 \approx 1$ and ending at $\alpha_T \approx 0$. In this paper we let

$$\alpha_t = (1 - 2s) \cdot f(t) + s \text{ where } f(t) = (1 - (t/T)^2),$$

for a precision value 10^{-5} that avoids numerically unstable situations. This schedule is very similar to the cosine noise schedule introduced in (Nichol & Dhariwal, 2021), but ours is somewhat simpler in notation. To avoid numerical instabilities during sampling, we follow the clipping procedure of (Nichol & Dhariwal, 2021) and compute $\alpha_{t|t-1} = \alpha_t / \alpha_{t-1}$, where we define $\alpha_{-1} = 1$. The values $\alpha_{t|t-1}^2$ are then clipped from below by 0.001. This avoids numerical instability as $1/\alpha_{t|t-1}$ is now bounded during sampling. Then the α_t values can be recomputed using the cumulative product $\alpha_t = \prod_{\tau=0}^t \alpha_{\tau|\tau-1}$.

Recall that $\text{SNR}(t) = \alpha_t^2 / \sigma_t^2$. As in (Kingma et al., 2021), we compute the negative log SNR curve defined as $\gamma(t) = -(\log \alpha_t^2 - \log \sigma_t^2)$ for $\sigma_t^2 = 1 - \alpha_t^2$. $\gamma(t)$ is a monotonically increasing function from which all required components can be computed with high numerical precision. For instance, $\alpha_t^2 = \text{sigmoid}(-\gamma(t))$, $\sigma_t^2 = \text{sigmoid}(\gamma(t))$, and $\text{SNR}(t) = \exp(-\gamma(t))$.

Log-likelihood estimator: As discussed, the simplified objective described in Algorithm 1 is optimized during training. However, when evaluating the log-likelihood of samples, the true weighting $w(t) = 1 - \text{SNR}(t-1)/\text{SNR}(t)$ needs to be used. For this purpose, we follow the procedure described in Algorithm 3. An important detail is that we choose to put an estimator over \mathcal{L}_t for $t = 1, \dots, T$ using $\mathbb{E}_{t \sim \mathcal{U}(1, \dots, T)}[T \cdot \mathcal{L}_t] = \sum_{t=1}^T \mathcal{L}_t$, but we require an additional forward pass for \mathcal{L}_0 . In initial experiments, we found the contribution of \mathcal{L}_0 very large compared to other loss terms, which would result in very high variance of the estimator. For that reason, the \mathcal{L}_0 is always computed at the expense of an additional forward pass. The resulting $\hat{\mathcal{L}}$ is an unbiased estimator for the log-likelihood.

Algorithm 3 Log-likelihood estimator for EDMs

Input: Data point \mathbf{x} , neural network ϕ

Sample $t \sim \mathcal{U}(1, \dots, T)$, $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, subtract center of gravity from $\epsilon_t^{(x)}$ in $\epsilon_t = [\epsilon_t^{(x)}, \epsilon_t^{(h)}]$

$\mathbf{z}_t = \alpha_t[\mathbf{x}, \mathbf{h}] + \sigma_t \epsilon_t$

$\mathcal{L}_t = \frac{1}{2}(1 - \text{SNR}(t-1)/\text{SNR}(t)) \|\epsilon_t - \phi(\mathbf{z}_t, t)\|^2$

Sample $\epsilon_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, subtract center of gravity from $\epsilon_0^{(x)}$ in $\epsilon_0 = [\epsilon_0^{(x)}, \epsilon_0^{(h)}]$

$\mathbf{z}_0 = \alpha_0[\mathbf{x}, \mathbf{h}] + \sigma_0 \epsilon_0$

$\mathcal{L}_0 = \mathcal{L}_0^{(x)} + \mathcal{L}_0^{(h)} = -\frac{1}{2}\|\epsilon - \phi(\mathbf{z}_0, 0)\|^2 - \log Z + \log p(\mathbf{h}|\mathbf{z}_0^{(h)})$

$\mathcal{L}_{\text{base}} = -\text{KL}(q(\mathbf{z}_T|\mathbf{x}, \mathbf{h})||p(\mathbf{z}_T)) = -\text{KL}(\mathcal{N}_{xh}(\alpha_T[\mathbf{x}, \mathbf{h}], \sigma_T^2 \mathbf{I})||\mathcal{N}_{xh}(\mathbf{0}, \mathbf{I}))$

Return $\hat{\mathcal{L}} = T \cdot \mathcal{L}_t + \mathcal{L}_0 + \mathcal{L}_{\text{base}}$

The Dynamics In Section 3.2 we explained that the dynamics of our proposed Equivariant Diffusion Model (EDM) are learned by the EGNN introduced in Section 2.2. The EGNN consists of a sequence of Equivariant Graph Convolutional Layers (EGCL). The EGCL is defined in Eq. 12. All its learnable components $\phi_e, \phi_h, \phi_x, \phi_{inf}$ by Multilayer Perceptrons:

Edge operation ϕ_e . Takes as input two node embeddings. The squared distance $d_{ij}^2 = \|\mathbf{x}_i^l - \mathbf{x}_j^l\|_2^2$, and the squared distance at the first layer as the optional attribute $a_{ij} = \|\mathbf{x}_i^0 - \mathbf{x}_j^0\|_2^2$ and outputs $\mathbf{m}_{ij} \in \mathbb{R}^{\text{nf}}$.

$$\text{concat}[\mathbf{h}_i^l, \mathbf{h}_j^l, d_{ij}^2, a_{ij}] \rightarrow \{\text{Linear}(\text{nf} \cdot 2 + 2, \text{nf}) \rightarrow \text{Silu} \rightarrow \text{Linear}(\text{nf}, \text{nf}) \rightarrow \text{Silu}\} \rightarrow \mathbf{m}_{ij}$$

Edge inference operation ϕ_{inf} . Takes as input the message \mathbf{m}_{ij} and outputs a scalar value $\tilde{e}_{ij} \in (0, 1)$.

$$\mathbf{m}_{ij} \rightarrow \{\text{Linear}(\text{nf}, 1) \rightarrow \text{Sigmoid}\} \rightarrow \tilde{e}_{ij}$$

Node update ϕ_h . Takes as input a node embedding and the aggregated messages and outputs the updated node embedding.

$$\text{concat}[\mathbf{h}_i^l, \mathbf{m}_{ij}] \rightarrow \{\text{Linear}(\text{nf} \cdot 2, \text{nf}) \rightarrow \text{Silu} \rightarrow \text{Linear}(\text{nf}, \text{nf}) \rightarrow \text{add}(\cdot, \mathbf{h}_i^l)\} \rightarrow \mathbf{h}_i^{l+1}$$

Coordinate update ϕ_x . Has the same inputs as ϕ_e and outputs a scalar value.

$$\text{concat}[\mathbf{h}_i^l, \mathbf{h}_j^l, d_{ij}^2, a_{ij}] \rightarrow \{\text{Linear}(\text{nf} \cdot 2 + 2, \text{nf}) \rightarrow \text{Silu} \rightarrow \text{Linear}(\text{nf}, \text{nf}) \rightarrow \text{Silu} \rightarrow \text{Linear}(\text{nf}, 1)\} \rightarrow \text{Output}$$

Equivariant Processes To be self-contained, a version of the proof from (Xu et al., 2022) is given here. It shows that if the transition distributions $p(\mathbf{z}_{t-1}|\mathbf{z}_t)$ are equivariant and $p(\mathbf{z}_T)$ is invariant, then every marginal distribution $p(\mathbf{z}_t)$ is invariant which importantly includes $p(\mathbf{z}_0)$. Here induction is used to derive the result.

Base case: Observe that $p(\mathbf{z}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ is equivariant with respect to rotations and reflections, so $p(\mathbf{z}_T) = p(\mathbf{R}\mathbf{z}_T)$.

Induction step: For some $t \in \{1, \dots, T\}$ assume $p(\mathbf{z}_t)$ to be invariant meaning that $p(\mathbf{z}_t) = p(\mathbf{R}\mathbf{z}_t)$ for all orthogonal \mathbf{R} . Let $p(\mathbf{z}_{t-1}|\mathbf{z}_t)$ be equivariant meaning that $p(\mathbf{z}_{t-1}|\mathbf{z}_t) = p(\mathbf{R}\mathbf{z}_{t-1}|\mathbf{R}\mathbf{z}_t)$ for orthogonal \mathbf{R} . Then:

$$\begin{aligned}
 p(\mathbf{R}\mathbf{z}_{t-1}) &= \int_{\mathbf{z}_t} p(\mathbf{R}\mathbf{z}_{t-1}|\mathbf{z}_t)p(\mathbf{z}_t) && \text{Probability Chain Rule} \\
 &= \int_{\mathbf{z}_t} p(\mathbf{R}\mathbf{z}_{t-1}|\mathbf{R}\mathbf{R}^{-1}\mathbf{z}_t)p(\mathbf{R}\mathbf{R}^{-1}\mathbf{z}_t) && \text{Multiply by } \mathbf{R}\mathbf{R}^{-1} = \mathbf{I} \\
 &= \int_{\mathbf{z}_t} p(\mathbf{z}_{t-1}|\mathbf{R}^{-1}\mathbf{z}_t)p(\mathbf{R}^{-1}\mathbf{z}_t) && \text{Equivariance \& Invariance} \\
 &= \int_{\mathbf{u}} p(\mathbf{z}_{t-1}|\mathbf{u})p(\mathbf{u}) \cdot \underbrace{\det \mathbf{R}}_{=1} && \text{Change of Variables } \mathbf{u} = \mathbf{R}^{-1}\mathbf{z}_t \\
 &= p(\mathbf{z}_{t-1}),
 \end{aligned}$$

and thus $p(\mathbf{z}_{t-1})$ is invariant. By induction, $p(\mathbf{z}_{T-1}), \dots, p(\mathbf{z}_0)$ are all invariant. Compared to (Xu et al., 2022), this proof makes explicit the dependency on a change of variables to rotate the reference frame of integration.

C. Additional Details on Experiments

Baseline model While our EDM model is parametrized by an $E(3)$ equivariant EGNN network, the GDM model used for the ablation study uses a non equivariant graph network. In this network, the coordinates are simply concatenated with the other node features: $\tilde{\mathbf{h}}_i^0 = [\mathbf{x}_i, \mathbf{h}]$. A message passing neural network (Gilmer et al., 2017) is then applied, that can be written:

$$\tilde{\mathbf{h}}_i^{l+1} = \phi_h(\tilde{\mathbf{h}}_i^l, \sum_{j \neq i} \tilde{e}_{ij} \mathbf{m}_{ij}) \quad \text{for } \mathbf{m}_{ij} = \phi_e(\tilde{\mathbf{h}}_i^l, \tilde{\mathbf{h}}_j^l, a_{ij})$$

The MLPs ϕ_e, ϕ_h are parametrized in the same way as in EGNN, with the sole exception that the input dimension of ϕ_e in the first layer is changed to accommodate the atom coordinates.

QM9 On QM9, the EDM and GDMs are trained using EGNNs with 256 hidden features and 9 layers. The models are trained for 1100 epochs, which is around 1.7 million iterations with a batch size of 64. The models are saved every 20 epochs when the validation loss is lower than the previously obtained number. The diffusion process uses $T = 1000$. Training takes approximately 7 days on a single NVIDIA GeForce GTX 1080Ti GPU. When generating samples the model takes on average 1.7 seconds per sample on the 1080Ti GPU. These times should not be taken as a fundamental limit of sampling performance, as more efficient samplers can be extracted from diffusion models after training (Salimans & Ho, 2022). For comparison, the E-NF takes 0.54 seconds per sample and G-Schnet 0.03 seconds. The EDM that only models heavy atoms and no hydrogens has the same architecture but is faster to train because it operates over less nodes: it takes about 3.2 days on a single 1080Ti GPU for 1100 epochs and converges even earlier to its final performance.

GEOM-DRUGS On GEOM, the EDM and GDMs are trained using EGNNs with 256 hidden features and 4 layers. The models are trained for 13 epochs, which is around 1.2 million iterations with a batch size of 64. Training takes approximately 5.5 days on three NVIDIA RTX A6000 GPUs. The model then takes on average 10.3 seconds to generate a sample.

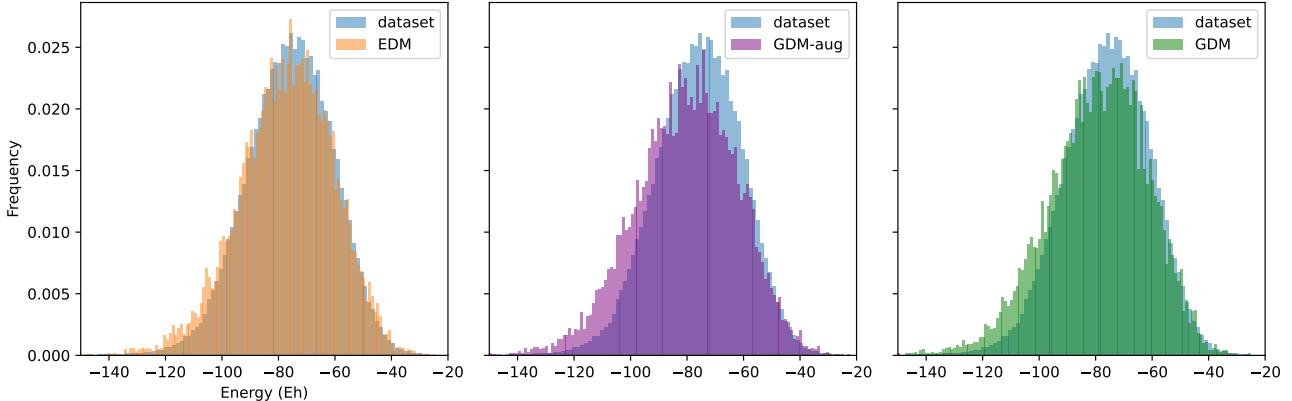


Figure 5. Distribution of estimated energies for the molecules generated by all methods trained on GEOM-DRUGS. We observe that EDM captures the dataset distribution well, while other methods tend to produce too many low-energy compounds.

Limitations of RDKit-based metrics Most commonly, unconstrained molecule generation methods are evaluated using RDKit. First, a molecule is built that contains only heavy atoms. Then, RDKit processes this molecule. In particular, it adds hydrogens to each heavy atoms in such a way that the valency of each atom matches its atom type. As a result, invalid molecules mostly appear when an atom has a valency bigger than expected.

Experimentally, we observed that validity could artificially be increased by reducing the number of bonds. For example, predicting only single bonds was enough to obtain close to 100% of valid molecules on GEOM-DRUGS. Such a change also increases the novelty of the generated molecules, since these molecules typically contain more hydrogens than the training set. On the contrary, our stability metrics directly model hydrogens and cannot be tricked as easily.

Regarding novelty, Vignac & Frossard (2021) argued that QM9 is the exhaustive enumeration of molecules that satisfy a predefined set of constraints. As a result, a molecule that is novel does not satisfy at least one of these constraints, which

means that the algorithm failed to capture some properties of the dataset. Experimentally, we observed that novelty decreases during training, which is in accordance with this observation. The final novelty metrics we obtain are the following:

Table 5. Novelty among valid and unique molecules (starting from 10000 molecules) with standard deviations across 3 runs on QM9. Experimentally, we observed that novelty is initially close to 100%, and decreases during training. On QM9, it reflects the fact that the algorithm progressively learns to capture the data distribution, which is an exhaustive enumeration under a predefined set of constraints.

Method	GDM-aug	EDM (with H)	EDM (no H)
Novelty (%)	74.6	65.7 \pm 1.3	34.5 \pm 0.9

Jensen–Shannon divergence between atomic distances In addition to the reported metrics in the QM9 experiment, we also report the Jensen-Shannon divergence between histograms of inter-atomic distances. Following the procedure from (Satorras et al., 2021a), we produce a histogram of relative distances between all pairs of atoms within each molecule. Then, we compute the Jensen-Shannon divergence between the normalized histograms produced with the generated and the training samples $JS_{div}(P_{gen} || P_{data})$. The lower the metric, the closer the distribution between generated and training samples will be. We report this metric in the following table for E-NF (Satorras et al., 2021a), G-Schnet (Gebauer et al., 2019) and EDM.

Table 6. Jensen-Shannon divergence between the normalized histograms of inter-atomic distances within atoms (lower is better). Results are reported for E-NF (Satorras et al., 2021a), G-Schnet (Gebauer et al., 2019) and our proposed method EDM.

	E-NF	G-Schnet	EDM (Ours)
JS-Div	.0049	.0027	.0002

Bond distances In order to check the validity and stability of the generated structures, we compute the distance between all pairs of atoms and use these distances to predict the existence of bonds and their order. Bond distances in Table 7, 8 and 9 are based on typical distances in chemistry²³. In addition, margins are defined for single, double, triple bonds $m_1, m_2, m_3 = 10, 5, 3$ which were found empirically to describe the QM9 dataset well. If an two atoms have a distance shorter than the typical bond length plus the margin for the respective bond type, the atoms are considered to have a bond between them. The allowed number of bonds per atom are: H: 1, C: 4, N: 3, O: 2, F: 1, B: 3, Al: 3, Si: 4, P: [3, 5], S: 4, Cl: 1, As: 3, Br: 1, I: 1. After all bonds have been created, we say that an atom is stable if its valency is precisely equal to the allowed number of bonds. An entire molecule is considered stable if *all* its atoms are stable. Although this metric does not take into account more atypical distances or aromatic bonds, it is still an extremely important metric as it measures whether the model is positioning the atoms precisely enough. On the QM9 dataset it still considers 95.2% molecules stable and 99.0% of atoms stable. For Geom-Drugs the molecules are much larger which introduces more atypical behaviour. Here the atom stability, which is 86.5%, can still be used since it describes how many atoms satisfy the typical bond length description. However, the molecule stability is 2.8% on the dataset, which is too low to draw meaningful conclusions.

Limitations of Log-Likelihood One of the metrics on which we compare the models is the log-likelihood, which is the negative cross-entropy between the data distribution and the model distribution. For discrete data, such a loss has a direct interpretation: it gives the number of bits required to compress the signal losslessly. For continuous data, no such interpretation exists. Further difficulty is that even though the representation is continuous, the underlying distribution may be discrete. When this happens, the log-likelihood is unbounded and can grow arbitrarily large. For the datasets used in this paper, the positional information (the conformation) is optimized with an iterative process to a local minimum, and thus has a discrete nature.

For this reason, the log-likelihoods are unbounded and should be treated with caution. They still provide insight into how the model is fitted: Higher log-likelihoods correspond to sharper model distributions on the correct locations, which is an important positive indication that the model is fitted well. However, it can also happen that part of the distribution (suppose for an x-coordinate) is extremely sharp whereas it is very blurred for another part (suppose a y-coordinate). The

²http://www.wiredchemist.com/chemistry/data/bond_energies_lengths.html

³<http://chemistry-reference.com/tables/Bond%20Lengths%20and%20Enthalpies.pdf>

Table 7. Typical bond distances for a single bond.

	H	C	O	N	P	S	F	Si	Cl	Br	I	B	As
H	74	109	96	101	144	134	92	148	127	141	161	119	152
C	109	154	143	147	184	182	135	185	177	194	214	-	-
O	96	143	148	140	163	151	142	163	164	172	194	-	-
N	101	147	140	145	177	168	136	-	175	214	222	-	-
P	144	184	163	177	221	210	156	-	203	222	-	-	-
S	134	182	151	168	210	204	158	200	207	225	234	-	-
F	92	135	142	136	156	158	142	160	166	178	187	-	-
Si	148	185	163	-	-	200	160	233	202	215	243	-	-
Cl	127	177	164	175	203	207	166	202	199	214	-	175	-
Br	141	194	172	214	222	225	178	215	214	228	-	-	-
I	161	214	194	222	-	234	187	243	-	-	266	-	-
B	119	-	-	-	-	-	-	-	175	-	-	-	-
As	152	-	-	-	-	-	-	-	-	-	-	-	-

Table 8. Typical bond distances for a double bond.

	C	O	N	P	S
C	134	120	129	-	160
O	120	121	121	150	-
N	129	121	125	-	-
P	-	150	-	-	186
S	-	-	-	186	-

Table 9. Typical bond distances for a triple bond.

	C	O	N
C	120	113	116
O	113	-	-
N	116	-	110

log-likelihood would still be high because of the x-coordinate, even though the y-coordinate is poorly fitted. Therefore, for this type of data, log-likelihoods should be considered in combination with other metrics such as the atom stability and molecule stability metrics, as done in this paper.

D. Samples from our models

Additional samples from the model trained on QM9 are depicted in Figure 6 and, and samples from the model trained on GEOM-DRUGS in Figure 7. These samples are not curated or cherry picked in any way. As a result, their structure may sometimes be difficult to see due to an unfortunate viewing angle.

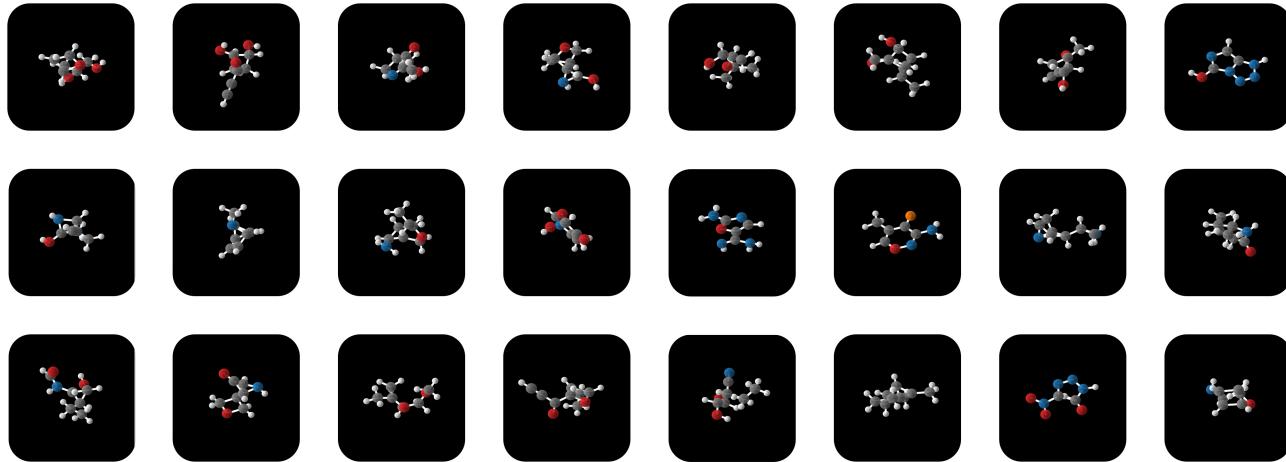


Figure 6. Random samples taken from the EDM trained on QM9.

The samples from the model trained on the drugs partition of GEOM show impressive large 3D structures. Interestingly, the model is sometimes generating disconnected component, which only happens QM9 models in early training stages. This may indicate that further training and increasing expressitivity of the models may further help the model bring these components together.

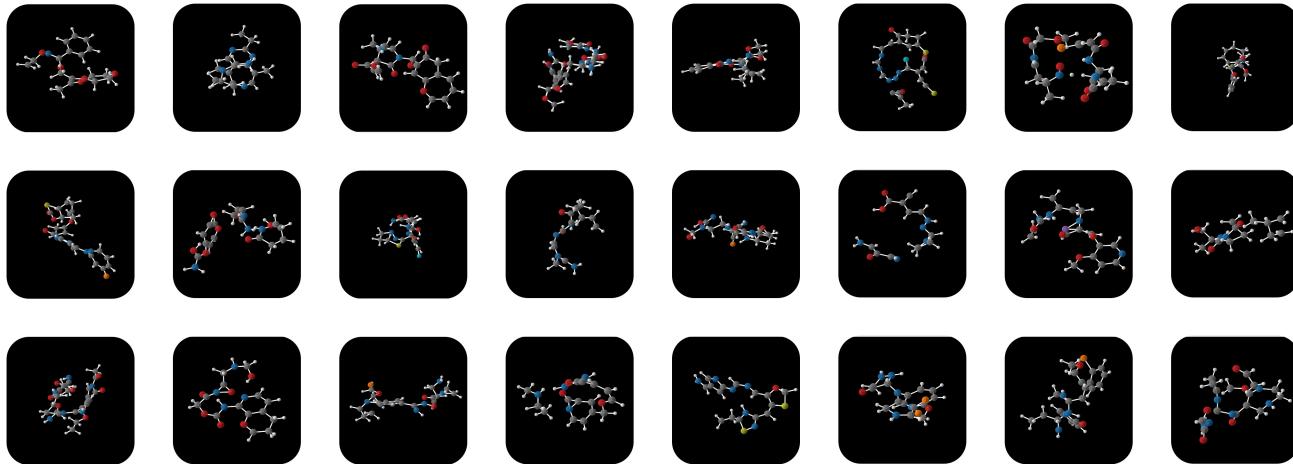


Figure 7. Random samples taken from the EDM trained on geom drugs. While most samples are very realistic, we observe two main failure cases: some molecules that are disconnected, and some that contain long rings. We note that the model does not feature any regularization to prevent these phenomena.

Figure 8 depicts the generation of molecules from a model trained on GEOM-Drugs. The model starts at random normal noise at time $t = T = 1000$ and iteratively sample $\mathbf{z}_{t-1} \sim p(\mathbf{z}_{t-1} | \mathbf{z}_t)$ towards $t = 0$ to obtain \mathbf{x}, \mathbf{h} , which is the resulting sample from the model. The atom type part of $\mathbf{z}_t^{(h)}$ is visualized by taking the argmax of this component.

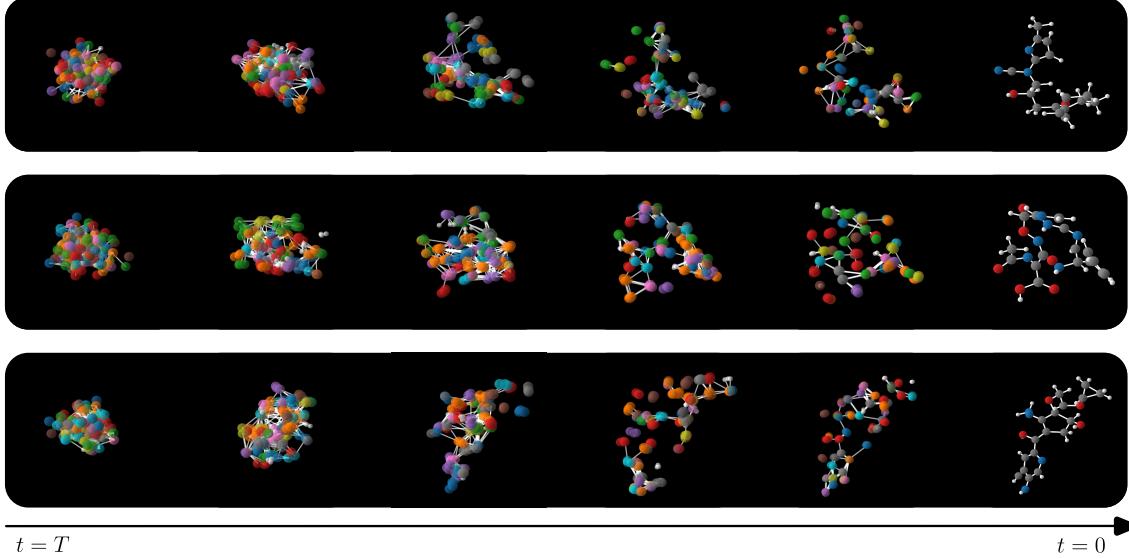


Figure 8. Selection of sampling chains at different steps from a model trained on GEOM-Drugs. The final column shows the resulting sample from the model.

Ablation on scaling features In Table 10 a comparison between the standard and proposed scaling is shown. Interestingly, there is quite a large difficulty in performance when measuring atom and molecule stability. From these results, it seems that it is easier to learn a denoising process where the atom type is decided later, when the atom coordinates are already relatively well defined.

Table 10. Ablation study on the scaling of features of the EDM. Comparing our proposed scaling to no scaling.

# Metrics	Scaling	NLL	Atom stable (%)	Mol stable (%)
EDM (ours)	$[\mathbf{x}, 1.00 \mathbf{h}^{\text{onehot}}, 1.0 \mathbf{h}^{\text{atom charge}}]$	-103.4	95.7	46.9
EDM (ours)	$[\mathbf{x}, 0.25 \mathbf{h}^{\text{onehot}}, 0.1 \mathbf{h}^{\text{atom charge}}]$	-110.7 \pm 1.5	98.7 \pm 0.1	82.0 \pm 0.4
Data		99.0	95.2	

E. Conditional generation

Conditional Method The specific definition for the loss components $\mathcal{L}_{c,t}$ is given in Equation 23. Essentially, a conditioning on a property c is added where relevant. The diffusion process that adds noise is not altered. The generative denoising process is conditioned on c by adding it as input to the neural network ϕ :

$$\begin{aligned}\mathcal{L}_{c,t} &= \mathbb{E}_{\epsilon_t \sim \mathcal{N}_{xh}(0, \mathbf{I})} \left[\frac{1}{2} (1 - \text{SNR}(t-1)/\text{SNR}(t)) \|\epsilon_t - \phi(\mathbf{z}_t, t, c)\|^2 \right], \\ \mathcal{L}_{c,0}^{(h)} &= \log p(\mathbf{h} | \mathbf{z}_0^{(h)}) \approx 0, \\ \mathcal{L}_{c,0}^{(x)} &= \log p(\mathbf{x} | \mathbf{z}_0, c) = \mathbb{E}_{\epsilon^{(x)} \sim \mathcal{N}(0, \mathbf{I})} \left[\log Z^{-1} - \frac{1}{2} \|\epsilon^{(x)} - \phi^{(x)}(\mathbf{z}_0, 0, c)\|^2 \right], \\ \mathcal{L}_{c,\text{base}} &= \mathcal{L}_{\text{base}} = -\text{KL}(q(\mathbf{z}_T | \mathbf{x}, \mathbf{h}) | p(\mathbf{z}_T)) \approx 0.\end{aligned}\tag{23}$$

Given a trained conditional model $p(\mathbf{x}, \mathbf{h} | c, M)$, we define the generative process by first sampling $c, M \sim p(c, M)$ and then $\mathbf{x}, \mathbf{h} \sim p(\mathbf{x}, \mathbf{h} | c, M)$. We compute $c, M \sim p(c, M)$ on the training partition as a parametrized two dimensional categorical distribution where we discretize the continuous variable c into small uniformly distributed intervals.

Implementation details: In this conditional experiment, our Equivariant Diffusion Model uses an EGNN with 9 layers, 192 features per hidden layer and SiLU activation functions. We used the Adam optimizer with learning rate 10^{-4} and batch size 64. Only atom types (categorical) and positions (continuous) have been modelled but not atom charges. All methods have been trained for ~ 2000 epochs while doing early stopping by evaluating the Negative Log Likelihood on the validation partition proposed by (Anderson et al., 2019).

Additionaly, the obtained molecule stabilities in the conditional generative case was similar to the ones obtained in the non-conditional setting. The reported molecule stabilities for each conditioned property evaluated on 10K generated samples are: (80.4%) α , (81.73%) $\Delta\epsilon$, (82.81%) ϵ_{HOMO} , (83.6 %) ϵ_{LUMO} , (83.3%) μ , (81.03 %) C_v .

QM9 Properties

α : Polarizability: Tendency of a molecule to acquire an electric dipole moment when subjected to an external electric field.

ϵ_{HOMO} : Highest occupied molecular orbital energy.

ϵ_{LUMO} : Lowest unoccupied molecular orbital energy.

$\Delta\epsilon$ Gap: The energy difference between HOMO and LUMO.

μ : Dipole moment.

C_v : Heat capacity at 298.15K

Conditional generation results In this Section we sweep over 9 different α values in the range [73.6, 101.6] while keeping the reparametrization noise ϵ fixed and the number of nodes $M = 19$. We plot 10 randomly selected sweeps in Figure 9 with different reparametrization noises ϵ each. Samples have been generated using our Conditional EDM. We can see that for larger Polarizability values, the atoms are distributed less isotropically encouraging larger dipole moments when an electric field is applied. This behavior is consistent among all reported runs.

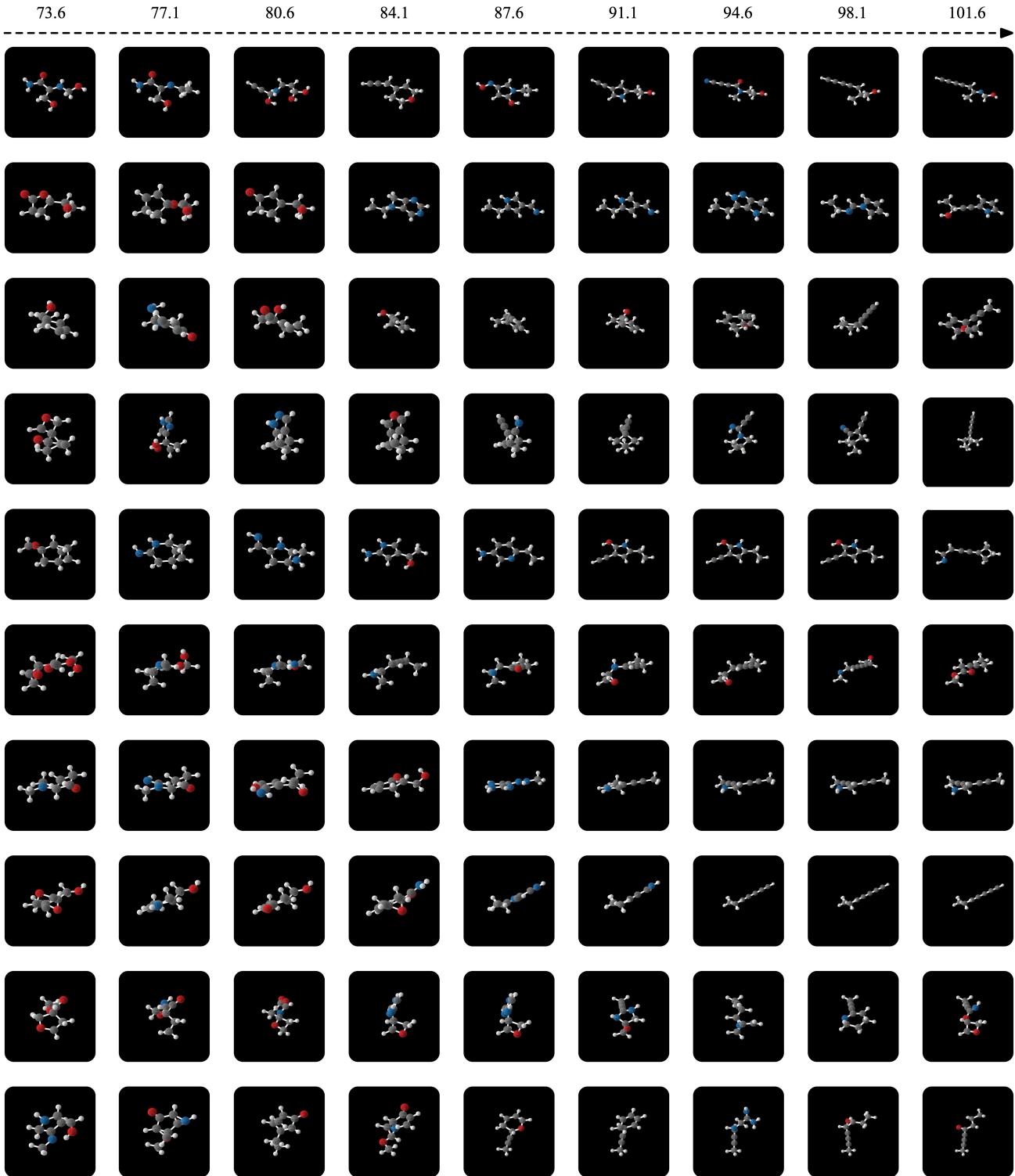


Figure 9. Molecules generated by our Conditional EDM when interpolating among different α polarizability values (from left to right). α 's are reported on top of the image. All samples within each row have been generated with the same reparametrization noise ϵ .