SYMDIFF: EQUIVARIANT DIFFUSION VIA STOCHASTIC SYMMETRISATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose SYMDIFF, a novel method for constructing equivariant diffusion models using the recently introduced framework of stochastic symmetrisation. SYMDIFF resembles a learned data augmentation that is deployed at sampling time, and is lightweight, computationally efficient, and easy to implement on top of arbitrary off-the-shelf models. Notably, in contrast to previous work, SYMDIFF typically does not require any neural network components that are intrinsically equivariant, avoiding the need for complex parameterizations and the use of higher-order geometric features. Instead, our method can leverage highly scalable modern architectures as drop-in replacements for these more constrained alternatives. We show that this additional flexibility yields significant empirical benefit on $\mathrm{E}(3)$-equivariant molecular generation. To the best of our knowledge, this is the first application of symmetrisation to generative modelling, suggesting its potential in this domain more generally.

1 INTRODUCTION

For geometrically structured data such as N-body systems of molecules or proteins, it is often of interest to obtain a diffusion that is *equivariant* with respect to some group actions (Xu et al., 2022; Hoogeboom et al., 2022; Watson et al., 2023; Yim et al., 2023). By accounting for the large number of permutation and rotational symmetries within such systems, equivariant diffusion models aim to improve the data efficiency (Batzner et al., 2022) and generalisation (Elesedy & Zaidi, 2021) of the generative model. In previous work this has been achieved by the use of *intrinsically* equivariant neural networks, which constrain each of their linear and nonlinear layers individually to be equivariant, so that the overall model is also equivariant as a result (Bronstein et al., 2021).

There is a growing literature on the practical limitations of intrinsically equivariant neural networks (Wang et al., 2024; Canez et al., 2024; Abramson et al., 2024; Pertigkiozoglou et al., 2024). It has been observed that these models can suffer from degraded training dynamics due to their imposition of architectural constraints, as well as increased computational cost and implementation complexity (Duval et al., 2023a). To address these issues, there has been recent interest in *symmetrisation* techniques for obtaining equivariance instead (Murphy et al., 2018; Puny et al., 2021; Duval et al., 2023b; Kim et al., 2023; Kaba et al., 2023; Mondal et al., 2023; Dym et al., 2024; Gelberg et al., 2024). These approaches offer a mechanism for constructing equivariant neural networks using subcomponents that are not equivariant, which previous work has shown leads to better performing models (Yarotsky, 2022; Kim et al., 2023). However, the advantages of symmetrisation-based equivariance have not yet been explored for generative modelling. This is possibly in part because previous work has solely focused on deterministic equivariance, rather than the more complex condition of *stochastic* equivariance that is required in the context of generative models.

In this paper, we introduce SYMDIFF, a novel methodology for obtaining equivariant diffusion models through symmetrisation, rather than intrinsic equivariance. We build on the recent framework of *stochastic symmetrisation* developed by Cornish (2024) using the theory of Markov categories (Fritz, 2020). Unlike previous work on symmetrisation, which operates on deterministic functions, stochastic symmetrisation can be applied to *Markov kernels* directly in distribution space. We show that this leads naturally to a more flexible approach for constructing equivariant diffusion models than is possible using intrinsic architectures. We apply this concretely to obtain an $\mathrm{E}(3)$-equivariant diffusion architecture for modelling N-body systems, where $\mathrm{E}(3)$ denotes the Euclidean group. For

this task, we formulate a novel reverse process that is allowed to be non-Gaussian, for which we derive a tractable optimisation objective. Our model is stochastically $\mathrm{E}(3)$-equivariant overall without needing any intrinically $\mathrm{E}(3)$-equivariant neural networks as subcomponents. We also sketch how to extend SYMDIFF to score and flow-based generative models (Song et al., 2020; Lipman et al., 2022).

We implemented SYMDIFF for de novo molecular generation, using it as a drop-in replacement for the $\mathrm{E}(3)$-equivariant diffusion of Hoogeboom et al. (2022), which relies on intrinsically equivariant neural networks. In contrast, our model is able to leverage highly scalable off-the-shelf architectures such as Diffusion Transformers (Peebles & Xie, 2023) for all of its subcomponents. We demonstrate this leads to significantly improved empirical performance for both the QM9 and GEOM-Drugs datasets.

2 BACKGROUND

2.1 EQUIVARIANT MARKOV KERNELS

Markov kernels At a high level, a *Markov kernel* $k : \mathcal{X} \to \mathcal{Y}$ may be thought of as a *conditional distribution* or *stochastic map* that, when given an input $\mathbf{x} \in \mathcal{X}$, produces a random output in $\mathcal{Y}$ with distribution $k(d\mathbf{y}|\mathbf{x})$. For example, given a function $f : \mathcal{X} \times \mathcal{E} \to \mathcal{Y}$ and a random element η of $\mathcal{E}$, there is a Markov kernel $k : \mathcal{X} \to \mathcal{Y}$ for which $k(d\mathbf{y}|\mathbf{x})$ is the distribution of $f(\mathbf{x}, \eta)$.[1] As a special case, every deterministic function $f : \mathcal{X} \to \mathcal{Y}$ may be thought of as a Markov kernel $\mathcal{X} \to \mathcal{Y}$ also. When $k(d\mathbf{y}|\mathbf{x})$ has a *density* (or *likelihood*), we will denote this by $k(\mathbf{y}|\mathbf{x})$, although we note that we can still reason about Markov kernels even when they do not admit a likelihood in this sense.

Stochastic equivariance Let $\mathcal{G}$ be a group acting on spaces $\mathcal{X}$ and $\mathcal{Y}$. We will denote this using "dot" notation, so that the action on $\mathcal{X}$ is a function $(g, \mathbf{x}) \mapsto g \cdot \mathbf{x}$. Recall that a function $f : \mathcal{X} \to \mathcal{Y}$ is then *equivariant* if

$$f(g \cdot \mathbf{x}) = g \cdot f(\mathbf{x}) \qquad \text{for all } \mathbf{x} \in \mathcal{X} \text{ and } g \in \mathcal{G}.$$

Here f is purely deterministic, and so this concept must be generalised in order to encompass models whose outputs are stochastic (Bloem-Reddy & Teh, 2020). To this end, Cornish (2024) uses a notion defined for Markov kernels: a Markov kernel $k : \mathcal{X} \to \mathcal{Y}$ is *stochastically equivariant* if

$$k(d\mathbf{y}|g \cdot \mathbf{x}) = (g \cdot k)(d\mathbf{y}|\mathbf{x}) \qquad \text{for all } \mathbf{x} \in \mathcal{X} \text{ and } g \in \mathcal{G}, \tag{1}$$

where the right-hand side denotes the distribution of $g \cdot \mathbf{y}$ when $\mathbf{y} \sim k(d\mathbf{y}|\mathbf{x})$, or in other words the *pushforward* of $k(d\mathbf{y}|\mathbf{x})$ under g. When k is obtained from f and η as above, equation 1 holds iff

$$f(g \cdot \mathbf{x}, \eta) \stackrel{\mathrm{d}}{=} g \cdot f(\mathbf{x}, \eta) \qquad \text{for all } \mathbf{x} \in \mathcal{X} \text{ and } g \in \mathcal{G},$$

where $\stackrel{\mathrm{d}}{=}$ denotes equality in distribution.[2] If η is constant, this says f is deterministically equivariant in the usual sense. Likewise, when k has a conditional density $k(\mathbf{y}|\mathbf{x})$, equation 1 holds if

$$k(g \cdot \mathbf{y}|g \cdot \mathbf{x}) = k(\mathbf{y}|\mathbf{x}) \qquad \text{for all } \mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}, \text{ and } g \in \mathcal{G},$$

provided the action of $\mathcal{G}$ on $\mathcal{Y}$ has unit Jacobian (Cornish, 2024, Proposition 3.1), as will be the case for all the actions we consider. This latter condition recovers the usual formulation of stochastic equivariance considered in the diffusion literature by e.g. Xu et al. (2022); Hoogeboom et al. (2022).

2.2 EQUIVARIANT DIFFUSION MODELS

Denoising diffusion models Diffusion models construct a generative model $p_\theta(\mathbf{z}_0)$ of data $p_{\text{data}}(\mathbf{z}_0)$, living in a space $\mathcal{Z}$, by learning to reverse an iterative forward noising process $\mathbf{z}_t$. Following the notation of Kingma et al. (2021), the distribution of $\mathbf{z}_t$ is defined by $q(\mathbf{z}_t|\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_t; \alpha_t \mathbf{z}_0, \sigma_t^2 \mathbf{I})$ for some noise schedule $\alpha_t, \sigma_t > 0$ such that the signal-to-noise ration $\mathrm{SNR}(t) :=$

[1] For "nice" choices of $\mathcal{Y}$, the converse also holds by *noise outsourcing* (Kallenberg, 2002, Lemma 3.22).

[2] Note this is more general than the *almost sure* condition from equation (17) of Bloem-Reddy & Teh (2020).

α_t^2/σ_t^2 is strictly monotonically decreasing. The joint distribution of the forward and reverse processes respectively then have the forms:

$$q(\mathbf{z}_{0:T}) = q(\mathbf{z}_0)\prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}) \qquad p_\theta(\mathbf{z}_{0:T}) = p(\mathbf{z}_T)\prod_{t=1}^{T} p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t) \tag{2}$$

with $q(\mathbf{z}_0) \coloneqq p_{\text{data}}(\mathbf{z}_0)$, and $q(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t; \alpha_{t|t-1}\mathbf{z}_{t-1}, \sigma_{t|t-1}^2\mathbf{I})$, with constants defined as $\alpha_{t|t-1} \coloneqq \alpha_t/\alpha_{t-1}$ and $\sigma_{t|t-1}^2 \coloneqq \sigma_t^2 - \alpha_{t|t-1}^2\sigma_{t-1}^2$. We take $p(\mathbf{z}_T)$ to also be Gaussian. The reverse process is then trained to maximise the ELBO of $\log p_\theta(\mathbf{z}_0)$, which can be obtained as follows (Sohl-Dickstein et al., 2015):

$$\begin{aligned}\log p_\theta(\mathbf{z}_0) \geq \mathbb{E}_{q(\mathbf{z}_1|\mathbf{z}_0)}[\log p_\theta(\mathbf{z}_0|\mathbf{z}_1)] - D_{\mathrm{KL}}(q(\mathbf{z}_T|\mathbf{z}_0)||p(\mathbf{z}_T)) \\ - \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{z}_0)}[D_{\mathrm{KL}}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0)||p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t))].\end{aligned} \tag{3}$$

This objective can be efficiently optimised when the reverse process is Gaussian. This is due to the fact that the posterior distributions $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0) = \mathcal{N}(\mathbf{z}_{t-1}; \mu_q(\mathbf{z}_t, \mathbf{z}_0), \sigma_q^2(t)\mathbf{I})$ are Gaussian, and that the KL divergence between Gaussians can be expressed in closed form. To match these posteriors, the reverse process is then typically defined in terms of a neural network $\mu_\theta : \mathcal{Z} \to \mathcal{Z}$ as

$$p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t) \coloneqq \mathcal{N}(\mathbf{z}_{t-1}; \mu_\theta(\mathbf{z}_t), \sigma_q^2(t)\mathbf{I}). \tag{4}$$

Invariant and equivariant diffusion It is often desirable for the model $p_\theta(\mathbf{z}_0)$ to be *invariant* with respect to the action of a group $\mathcal{G}$. Invariance here means that

$$p_\theta(g \cdot \mathbf{z}_0) = p_\theta(\mathbf{z}_0) \qquad \text{for all } g \in \mathcal{G}.$$

Intuitively, this says that the density $p_\theta(\mathbf{z}_0)$ is constant on the orbits of this action. For the model in equation 2, invariance follows if $p_\theta(\mathbf{z}_T)$ is itself invariant, and if each $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$ is stochastically $\mathcal{G}$-equivariant (Xu et al., 2022). All previous work we are aware of has approached this problem by obtaining a deterministically equivariant μ_θ, which then implies that the reverse process is stochastically equivariant (Le et al., 2023).

2.3 N-BODY SYSTEMS AND $E(3)$-EQUIVARIANT DIFFUSION

Equivariant diffusion models are often applied to model N-body systems such as molecules and proteins (Xu et al., 2022; Hoogeboom et al., 2022; Yim et al., 2023). This is motivated by the large number of symmetries present in these system. For example, intuitively speaking, neither the coordinate system nor the ordering of bodies in the system should matter for sampling. We describe the standard components of such models below.

N-body data In 3-dimensions, the state of an N-body system can be encoded as a pair $\mathbf{z} = [\mathbf{x}, \mathbf{h}] \in \mathbb{R}^{N\times(3+d)}$, where $\mathbf{x} \equiv (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}) \in \mathbb{R}^{N\times 3}$ describes a set of N points in 3D space, and $\mathbf{h} \equiv (\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(N)}) \in \mathbb{R}^{N\times d}$ describes a set of N feature vectors of dimension d. Each feature vector $\mathbf{h}^{(i)}$ is associated with the point $\mathbf{x}^{(i)}$. For example, Hoogeboom et al. (2022) encodes molecules in this way, where each $\mathbf{x}^{(i)}$ denotes the location of an atom, and $\mathbf{h}^{(i)}$ some corresponding properties such as atom type (represented as continuous quantities).

Center of mass free space Intuitively speaking, for many applications, the location of an N-body system in space should not matter. For this reason, instead of defining a diffusion on the full space of N-body systems directly, previous work (Garcia Satorras et al., 2021a; Xu et al., 2022; Hoogeboom et al., 2022) has set $\mathcal{Z} \coloneqq \mathcal{U}$, where $\mathcal{U}$ is the "center of mass (CoM) free" linear subspace of $\mathbb{R}^{N\times(3+d)}$ consisting of $[\mathbf{x}, \mathbf{h}]$ such that $\bar{\mathbf{x}} \coloneqq \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)} = 0$. In this way, samples from their model are always guaranteed to be centered at the origin.

CoM-free diffusions To construct their forward and reverse processes to now live entirely on $\mathcal{U}$ instead of $\mathbb{R}^{N\times(3+d)}$, Xu et al. (2022) defines the *projected Gaussian distribution* $\mathcal{N}_{\mathcal{U}}(\mu, \sigma^2\mathbf{I})$, for $\mu \in \mathcal{U}$ and $\sigma^2 > 0$, as the distribution of $\mathbf{z}$ obtained via the following process:

$$\epsilon \sim \mathcal{N}(0, \mathbf{I}) \qquad \mathbf{z} \coloneqq \mu + \sigma\,\pi(\epsilon),$$

where $\pi : \mathbb{R}^{N\times(3+d)} \to \mathcal{U}$ centers its input at the origin, so that $\pi([\mathbf{x}, \mathbf{h}]) \coloneqq [\mathbf{x} - (\bar{\mathbf{x}}, \dots, \bar{\mathbf{x}}), \mathbf{h}]$. By construction, the projected Gaussian distribution is then supported on the linear subspace $\mathcal{U}$. Xu et al. (2022) shows that this distribution has a density with Gaussian form $\mathcal{N}_{\mathcal{U}}(\mathbf{z}; \mu, \sigma^2\mathbf{I}) \propto \mathcal{N}(\mathbf{z}; \mu, \sigma^2\mathbf{I})$ defined for points $\mathbf{z} \in \mathcal{U}$ living in this subspace. The forward and reverse processes $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ and $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$ in equation 2 can then be defined exactly as before, but now as projected Gaussians. Since these are still Gaussian (albeit restricted to a linear subspace), the KL terms in equation 3 remain tractable, which permits optimising the ELBO in the usual way. This does require that $\mu_\theta : \mathcal{U} \to \mathcal{U}$ is now constrained to take values in the subspace $\mathcal{U}$. Prior work has enforced this by taking $\mu_\theta \coloneqq \pi \circ \varphi_\theta$, where $\varphi_\theta : \mathcal{U} \to \mathbb{R}^{N\times(3+d)}$ is a neural network.

Invariance and equivariance Intuitively, the ordering of the N points and the orientation of the overall system in 3D space should not matter. To formalise this, let S_N denote the symmetric group of permutations of the integers $\{1, \dots, N\}$, and $\mathrm{O}(3)$ denote the group of orthogonal 3×3 matrices. Their product $S_N \times \mathrm{O}(3)$ acts on N-body systems by reordering and orthogonally transforming points as follows:

$$(\sigma, R) \cdot [\mathbf{x}, \mathbf{h}] \coloneqq [R\mathbf{x}^{(\sigma(1))}, \dots, R\mathbf{x}^{(\sigma(N))}, \mathbf{h}^{(\sigma(1))}, \dots, \mathbf{h}^{(\sigma(N))}], \tag{5}$$

where $\sigma \in S_N$ and $R \in \mathrm{O}(3)$. Previous work (Xu et al., 2022; Hoogeboom et al., 2022) has then chosen the model $p_\theta(\mathbf{z}_0)$ to be invariant to this action. They enforce this via the approach described in Section 2.2, by ensuring that μ_θ is deterministically $(S_N \times \mathrm{O}(3))$-equivariant, which implies that the reverse process is stochastically equivariant also.

Following standard terminology in the literature (Garcia Satorras et al., 2021b; Xu et al., 2022; Hoogeboom et al., 2022), we refer to a $(S_N \times \mathrm{O}(3))$-diffusion defined on the CoM-free space $\mathcal{U}$ as an $\mathrm{E}(3)$-*equivariant diffusion*.

3 Equivariant Diffusion via Stochastic Symmetrisation

In this section, we introduce our methodology SymDiff, and apply this concretely to the problem of obtaining $\mathrm{E}(3)$-equivariant diffusion models for N-body systems. We also discuss extensions to score and flow-based generative models (Song et al., 2020; Lipman et al., 2022) in Appendix D.

3.1 Stochastic symmetrisation

Recently, Cornish (2024) gave a general theory of *symmetrisation* using the framework of *Markov categories* (Fritz, 2020). This work provides a characterisation of all possible symmetrisation procedures, recovering previous examples in the deterministic setting as special cases (Murphy et al., 2018; Puny et al., 2021; Kaba et al., 2023; Kim et al., 2023). The theory applies flexibly and compositionally to general groups and actions, including in the non-compact case, and extends beyond deterministic functions to provide a methodology for symmetrising *Markov kernels*, which had not previously been considered.

In this work, we will consider a special case of the symmetrisation framework of Cornish (2024). Concretely, for groups $\mathcal{H}$ and $\mathcal{G}$, we will consider the problem of symmetrising an $\mathcal{H}$-equivariant Markov kernel so that it becomes *both* $\mathcal{H}$-equivariant *and* $\mathcal{G}$-equivariant. Intuitively, this allows "upgrading" a "partially equivariant" Markov kernel to become "more equivariant". By taking $\mathcal{H}$ to be the trivial group consisting of just the identity element, this in particular allows converting arbitrary unconstrained Markov kernels to ones that are $\mathcal{G}$-equivariant. However, as we show later, it is often more computationally efficient to symmetrise a Markov kernel that is already "partially equivariant" as opposed to fully unconstrained, which motivates studying the case of general $\mathcal{H}$ here.

General procedure Suppose the product group $\mathcal{H} \times \mathcal{G}$ acts on both the spaces $\mathcal{X}$ and $\mathcal{Y}$. This means we have an action $(h, g) \cdot \mathbf{x}$ defined for $(h, g) \in \mathcal{H} \times \mathcal{G}$ and $\mathbf{x} \in \mathcal{X}$, and similarly for $\mathcal{Y}$ (see equation 5 below for a concrete example involving N-body systems). This implies that both $\mathcal{H}$ and $\mathcal{G}$ act on both $\mathcal{X}$ and $\mathcal{Y}$ individually: for example, $\mathcal{H}$ acts on $\mathcal{X}$ via $h \cdot \mathbf{x} \coloneqq (h, e_{\mathcal{G}}) \cdot \mathbf{x}$, where $e_{\mathcal{G}} \in \mathcal{G}$ is the identity element. The following result, which is a special case of Example 6.3 of Cornish (2024), then gives a procedure for symmetrising $\mathcal{H}$-equivariant Markov kernels to become $(\mathcal{H} \times \mathcal{G})$-equivariant ones. For the proof, see Appendix A.1.

Proposition 1. *Let $\gamma : \mathcal{X} \to \mathcal{G}$ be a Markov kernel that is $\mathcal{H}$-invariant (so that $\gamma(dg|h \cdot \mathbf{x}) = \gamma(dg|\mathbf{x})$) and $\mathcal{G}$-equivariant, where $\mathcal{G}$ acts on itself by left multiplication, i.e. $g \cdot g' \coloneqq gg'$. Then every $\mathcal{H}$-equivariant Markov kernel $k : \mathcal{X} \to \mathcal{Y}$ gives rise to an $(\mathcal{H} \times \mathcal{G})$-equivariant Markov kernel $\mathsf{sym}_\gamma(k) : \mathcal{X} \to \mathcal{Y}$, where $\mathsf{sym}_\gamma(k)(d\mathbf{y}|\mathbf{x})$ is the distribution of $\mathbf{y}$ sampled as follows:*

$$g \sim \gamma(dg|\mathbf{x}) \qquad \mathbf{y}_0 \sim k(d\mathbf{y}|g^{-1} \cdot \mathbf{x}) \qquad \mathbf{y} \coloneqq g \cdot \mathbf{y}_0. \tag{6}$$

Cornish (2024) shows that this is the only natural symmetrisation procedure that can be defined here without further assumptions on the groups, spaces, or Markov kernels involved.

Recursive symmetrisation Notice that the symmetrisation procedure defined by Proposition 1 requires γ already to satisfy some equivariance constraints. In effect this pushes back the problem of obtaining $(\mathcal{H} \times \mathcal{G})$-equivariant Markov kernels to the choice of γ, which mirrors the situation in the deterministic setting also (Puny et al., 2021; Kim et al., 2023). To work around this, Cornish (2024) proposes a *recursive* strategy for obtaining γ. Specifically, the idea is to set

$$\gamma \coloneqq \mathsf{sym}_{\gamma_0}(\gamma_1) \tag{7}$$

where $\gamma_0, \gamma_1 : \mathcal{X} \to \mathcal{G}$ are Markov kernels, and γ_0 is $\mathcal{H}$-invariant and $\mathcal{G}$-equivariant, but where now γ_1 is only required to be $\mathcal{H}$-invariant, and may behave arbitrarily with respect to $\mathcal{G}$. For compact $\mathcal{G}$, a suitable choice of γ_0 here is always available as $\gamma_0(dg|\mathbf{x}) \coloneqq \lambda(dg)$ (Cornish, 2024, Example 6.3), where λ denotes the *Haar measure* on $\mathcal{G}$ (Kallenberg, 1997).[3] We note that this use of the Haar measure exploits the stochastic nature of the procedure in Proposition 1 and would not be possible using deterministic symmetrisation methods here instead.

3.2 SymDiff: Symmetrised diffusion

We propose to use stochastic symmetrisation to obtain a diffusion process as in Section 2.2 whose reverse kernels are stochastically equivariant. Specifically, suppose some product group $\mathcal{H} \times \mathcal{G}$ acts on our state space $\mathcal{Z}$. For each timestep $t \in \{1, \dots, T\}$, we will choose some $\mathcal{H}$-equivariant Markov kernel $k_\theta : \mathcal{Z} \to \mathcal{Z}$ that admits a conditional density $k_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$. Similarly, we will choose some Markov kernel $\gamma_\theta : \mathcal{Z} \to \mathcal{G}$ that satisfies the conditions of Proposition 1 when $\mathcal{X} = \mathcal{Z}$. With these components, we will define our equivariant reverse process to be

$$p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t) \coloneqq \mathsf{sym}_{\gamma_\theta}(k_\theta)(\mathbf{z}_{t-1}|\mathbf{z}_t), \tag{8}$$

which is guaranteed to be $(\mathcal{H} \times \mathcal{G})$-equivariant by Proposition 1. This defines a conditional *density*, not just a Markov kernel, as a consequence of the next result. For the proof, see Appendix A.2.

Proposition 2. *Under the setup of Proposition 1, let $k(d\mathbf{y}|g, \mathbf{x})$ be the distribution of $\mathbf{y}$ sampled via*

$$\mathbf{y}_0 \sim k(d\mathbf{y}|g^{-1} \cdot \mathbf{x}) \qquad \mathbf{y} \coloneqq g \cdot \mathbf{y}_0.$$

If $k(d\mathbf{y}|\mathbf{x})$ has a density $k(\mathbf{y}|\mathbf{x})$, then so do $k(d\mathbf{y}|g, \mathbf{x})$ and $\mathsf{sym}_\gamma(k)(d\mathbf{y}|\mathbf{x})$, where the latter is

$$\mathsf{sym}_\gamma(k)(\mathbf{y}|\mathbf{x}) = \mathbb{E}_{\gamma(dg|\mathbf{x})}[k(\mathbf{y}|g, \mathbf{x})].$$

When the action of $\mathcal{G}$ on $\mathcal{Y}$ has unit Jacobian, we also have $k(\mathbf{y}|g, \mathbf{x}) = k(g^{-1} \cdot \mathbf{y}|g^{-1} \cdot \mathbf{x})$.

Training objective We would like to learn the parameters θ using the ELBO from equation 3. However, in general, we do not have access to the densities $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$ from equation 8 in closed form, since this requires computing the expectation as in Proposition 2. As such, we cannot compute the ELBO directly. However, since $\log$ is concave, Jensen's inequality allows us to bound

$$\log p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t) \geq \mathbb{E}_{\gamma_\theta(dg|\mathbf{z}_t)}[\log k_\theta(\mathbf{z}_{t-1}|g, \mathbf{z}_t)].$$

Since the ELBO in equation 3 depends linearly on $\log p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$, this allows us to also bound

$$\begin{aligned}\log p_\theta(\mathbf{z}_0) \geq \overbrace{\mathbb{E}_{q(\mathbf{z}_1|\mathbf{z}_0), \gamma_\theta(dg|\mathbf{z}_1)}[\log k_\theta(\mathbf{z}_0|g, \mathbf{z}_1)]}^{-\mathcal{L}_0} - D_{\mathrm{KL}}(q(\mathbf{z}_T|\mathbf{z}_0)||p(\mathbf{z}_T)) \\ - \sum_{t=2}^{T} \overbrace{\mathbb{E}_{q(\mathbf{z}_t|\mathbf{z}_0), \gamma_\theta(dg|\mathbf{z}_t)}[D_{\mathrm{KL}}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0)||k_\theta(\mathbf{z}_{t-1}|g, \mathbf{z}_t)]}^{\mathcal{L}_t},\end{aligned} \tag{9}$$

where the right-hand side is a tractable lower bound to the original ELBO in equation 3. We take this new bound as our objective used to train our SymDiff model.

[3] We could use this γ_0 directly in place of γ in Proposition 1, although we found it empirically better to use the more expressive model form in equation 7 instead.

Comparison with deterministic symmetrisation An alternative approach to equation 8 would be to obtain μ_θ in equation 4 via *deterministic* symmetrisation. An advantage of our method is that we require only a *single* pass through γ_θ and k_θ at sampling time. In contrast, many deterministic techniques (Puny et al., 2021; Kim et al., 2023) involve a Monte Carlo averaging step that requires multiple passes through the model instead, and then only return an approximation, which would introduce sampling bias. The canonicalisation method of Kaba et al. (2023) avoids this Monte Carlo step, but instead suffers from pathologies associated with its deterministic analogue of γ_θ, which Dym et al. (2024) show must become discontinuous at certain inputs. Additionally, canonicalisation requires an intrinsically $\mathcal{G}$-equivariant architecture for its analogue of γ_θ. In contrast, whenever $\mathcal{G}$ is compact, our recursive strategy in equation 7 requires no intrinsically $\mathcal{G}$-equivariant neural network components anywhere in the model at all.

3.3 SymDiff for N-body systems

We now apply SymDiff in the setting of N-body systems considered in Section 2.3. Specifically, we take $\mathcal{Z} \coloneqq \mathcal{U}$, $\mathcal{H} \coloneqq S_N$, and $\mathcal{G} \coloneqq \mathrm{O}(3)$, and consider the action on $\mathcal{Z}$ defined in equation 5. This means that we start with $k_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$ in equation 8 that is already equivariant with respect to reorderings of the N bodies, and then symmetrise this to obtain an $(S_N \times \mathrm{O}(3))$-equivariant reverse kernel overall. We choose to symmetrise in this way because highly scalable S_N-equivariant kernels based on Transformer architectures can be readily constructed for this purpose (Vaswani et al., 2017; Lee et al., 2019; Peebles & Xie, 2023), whereas intrinsically $\mathrm{O}(3)$-equivariant neural networks have not shown the same degree of scalability to-date (Abramson et al., 2024).

Choice of unsymmetrised kernels It remains now to choose $k_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$. We now do so in a way that equation 9 will resemble the standard diffusion objective in Ho et al. (2020), allowing for the scalable training of SymDiff. Specifically, we take

$$k_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t) \coloneqq \mathcal{N}_{\mathcal{U}}(\mathbf{z}_{t-1}; \mu_\theta(\mathbf{z}_t), \sigma_q^2(t)\mathbf{I}), \tag{10}$$

where $\mu_\theta : \mathcal{Z} \to \mathcal{Z}$ is an arbitrary S_N-equivariant neural network, which in turn means k_θ is stochastically S_N-equivariant. We highlight that μ_θ is otherwise unconstrained and can process $[\mathbf{x}, \mathbf{h}]$ jointly. In contrast, previous work using intrinsically $(S_n \times \mathrm{O}(3))$-equivariant components (Satorras et al., 2021; Thölke & De Fabritiis, 2022; Hua et al., 2024) has required complex parameterisations for μ_θ that handle the $\mathbf{x}$ and $\mathbf{h}$ inputs separately.

Form of KL terms We now show how our model yields a closed-form expression for the $\mathcal{L}_t$ terms in equation 9. Standard arguments show that each $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0) = \mathcal{N}_{\mathcal{U}}(\mathbf{z}_t; \mu_q(\mathbf{z}_t, \mathbf{z}_0), \sigma_q^2(t)\mathbf{I})$ is a (projected) Gaussian. We claim that our model also gives a (projected) Gaussian

$$k_\theta(\mathbf{z}_{t-1}|R, \mathbf{z}_t) = \mathcal{N}_{\mathcal{U}}(\mathbf{z}_{t-1}; R \cdot \mu_\theta(R^T \cdot \mathbf{z}_t), \sigma_q^2(t)\mathbf{I}). \tag{11}$$

Indeed, by the definition of this kernel in Proposition 2 and the definition of $\mathcal{N}_{\mathcal{U}}$ in Section 2.3, and since $R^{-1} = R^T$ for $R \in \mathrm{O}(3)$, for $\mathbf{z}_{t-1} \sim k_\theta(\mathbf{z}_{t-1}|R, \mathbf{z}_t)$ we have

$$\mathbf{z}_{t-1} \stackrel{\mathrm{d}}{=} R \cdot \big(\mu_\theta(R^T \cdot \mathbf{z}_t) + \sigma_q(t)\,\epsilon\big) = R \cdot \mu_\theta(R^T \cdot \mathbf{z}_t) + \sigma_q(t) R \cdot \epsilon,$$

where $\epsilon \sim \mathcal{N}_{\mathcal{U}}(0, \mathbf{I})$. Since $R \cdot \epsilon \stackrel{\mathrm{d}}{=} \epsilon$ for $R \in \mathrm{O}(3)$, equation 11 now follows. The same argument as Hoogeboom et al. (2022) now yields the closed-form expression

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{z}_t|\mathbf{z}_0), \gamma_\theta(dR|\mathbf{z}_t)} \left[\frac{1}{2\sigma_q^2(t)} \left\| \mu_q(\mathbf{z}_t, \mathbf{z}_0) - R \cdot \mu_\theta(R^T \cdot \mathbf{z}_t) \right\|^2 \right]. \tag{12}$$

We can obtain unbiased gradients of this quantity whenever $\gamma_\theta(dR|\mathbf{z}_t)$ is *reparametrisable* (Kingma, 2013). In other words, we should define $\gamma_\theta(dR|\mathbf{z}_t)$ to be the distribution of $\varphi_\theta(\mathbf{z}_t, \xi)$, where φ_θ is a deterministic neural network, and ξ is some noise variable whose distribution does not depend on θ. For a discussion of how we can handle the $\mathcal{L}_0$ term in equation 9 in our framework, we refer to Appendix C.1.

ϵ-parameterisation When μ_θ is taken to have the ϵ-form (Ho et al., 2020; Kingma et al., 2021)

$$\mu_\theta(\mathbf{z}_t) \coloneqq \frac{1}{\alpha_{t|t-1}} \mathbf{z}_t - \frac{\sigma_{t|t-1}^2}{\alpha_{t|t-1}\sigma_t} \epsilon_\theta(\mathbf{z}_t), \tag{13}$$

for some neural network $\epsilon_\theta : \mathcal{Z} \to \mathcal{Z}$, the same argument given by Ho et al. (2020) now allows us to rewrite equation 12 in a way that resembles the standard diffusion objective:

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{z}_0), \epsilon \sim \mathcal{N}_{\mathcal{U}}(0,\mathbf{I}), \gamma_\theta(dR|\mathbf{z}_t)} \left[\frac{1}{2} w(t) \left\| \epsilon - R \cdot \epsilon_\theta(R^T \cdot \mathbf{z}_t) \right\|^2 \right] \tag{14}$$

where $\mathbf{z}_t = \alpha_t \mathbf{z}_0 + \sigma_t \epsilon$, and $w(t) = (1 - \mathrm{SNR}(t-1)/\mathrm{SNR}(t))$. Recall we require μ_θ to be S_N-equivariant, which straightforwardly follows here whenever ϵ_θ is. In practice, we set $w(t) = 1$ during training as is commonly done in the diffusion literature (Kingma & Gao, 2024).

Recursive choice of γ_θ Recall that to apply Proposition 1, we require γ_θ to be S_N-invariant and $\mathrm{O}(3)$-equivariant. To obtain this, we apply the recursive procedure from equation 7, where $\gamma_0 : \mathcal{Z} \to \mathrm{O}(3)$ is obtained using the Haar measure on $\mathrm{O}(3)$ as described there, and $\gamma_1(dR|\mathbf{z}_t)$ is defined as the distribution of $f_\theta(\mathbf{z}_t, \eta)$, where f_θ is some S_N-invariant neural network, and η is sampled from some noise distribution $\nu(d\eta)$. Since the Haar measure does not depend on θ, the overall γ_θ obtained in this way remains reparametrisable. We emphasise that f_θ is *not* required to be $\mathrm{O}(3)$-equivariant, thus allowing for highly flexible choices such as Set Transformers (Lee et al., 2019). At sampling time, we use the procedure from Section 5 of Mezzadri (2006) to sample from the Haar measure on $\mathrm{O}(3)$, which is a negligible overhead compared with the cost of evaluating f_θ.

Algorithm 1 SymDiff training step

1: Sample $\mathbf{z}_0 \sim p_{\text{data}}(\mathbf{z}_0)$, $t \sim \mathrm{Unif}(\{1, \ldots, T\})$ and $\epsilon \sim \mathcal{N}_{\mathcal{U}}(0, \mathbf{I})$
2: $\mathbf{z}_t \leftarrow \alpha_t \mathbf{z}_0 + \sigma_t \epsilon$
3: Sample R_0 from the Haar measure on $\mathrm{O}(3)$ and $\eta \sim \nu(d\eta)$
4: $R \leftarrow R_0 \cdot f_\theta(R_0^T \cdot \mathbf{z}_t, \eta)$
5: Take gradient descent step with $\nabla_\theta \frac{1}{2} w(t) \left\| \epsilon - R \cdot \epsilon_\theta(R^T \cdot \mathbf{z}_t) \right\|^2$

Algorithm 2 SymDiff sampling process

1: Sample $\mathbf{z}_T \sim \mathcal{N}_{\mathcal{U}}(0, \mathbf{I})$
2: **for** $s = T, \ldots, 1$ **do**
3: Sample R_0 from the Haar measure on $\mathrm{O}(3)$, $\eta \sim \nu(d\eta)$, and $\epsilon \sim \mathcal{N}_{\mathcal{U}}(0, \mathbf{I})$
4: $R \leftarrow R_0 \cdot f_\theta(R_0^T \cdot \mathbf{z}_t, \eta)$
5: $\mathbf{z}_{t-1} \leftarrow \frac{1}{\alpha_{t|t-1}} \mathbf{z}_t - \frac{\sigma^2_{t|t-1}}{\alpha_{t|t-1}\sigma_t} R \cdot \epsilon_\theta(R^T \cdot \mathbf{z}_t) + \sigma_q(t)\epsilon$
6: **return** $\mathbf{z}_0 \sim p_\theta(\mathbf{z}_0|\mathbf{z}_1)$ ▷ See Appendix C.1 for an example of this output kernel

3.4 Data Augmentation is a Special Case of SymDiff

Data augmentation is a popular method for incorporating "soft" inductive biases within neural networks. Consider again our setup from the previous section, but suppose we now used the unsymmetrised kernels $k_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$ from equation 10 in place of $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$ in our backwards process directly. Suppose moreover that we trained our model using the standard diffusion objective, applying a uniform random orthogonal transformation to the input of our ϵ_θ network before each forward pass. This is equivalent to optimising the following objective:

$$\mathcal{L}_t^{\text{aug}} = \mathbb{E}_{q(\mathbf{z}_0), \epsilon \sim \mathcal{N}_{\mathcal{U}}(0,\mathbf{I}), \lambda(dR)} \left[\frac{1}{2} w(t) \|\epsilon - \epsilon_\theta(\alpha_t R \cdot \mathbf{z}_0 + \sigma_t \epsilon)\|^2 \right], \tag{15}$$

where λ is the Haar measure on $\mathrm{O}(3)$. (More general choices of λ could also be considered.) We then have the following result, proven in Appendix A.3.

Proposition 3. *When $\gamma_\theta(dR|\mathbf{z}_t) = \lambda(dR)$ for all $\mathbf{z}_t \in \mathcal{Z}$, our* SymDiff *objective recovers the data augmentation objective exactly, so that $\mathcal{L}_t = \mathcal{L}_t^{\text{aug}}$.*

In this way, SymDiff may be understood as a strict generalisation of data augmentation in which a more flexible augmentation process γ_θ is *learned* during training. Moreover, our γ_θ is then deployed at sampling time, which guarantees stochastic equivariance. In contrast, data augmentation is usually only applied during training, and the model deployed at sampling time then becomes only approximately equivariant.

4 EXPERIMENTS

We evaluated our $E(3)$-equivariant SYMDIFF model as a drop-in replacement for the $E(3)$-equivariant diffusion (EDM) of Hoogeboom et al. (2022) on both the QM9 and GEOM-Drugs datasets for molecular generation. We implemented this by taking the code base of Hoogeboom et al. (2022) and substituting our symmetrised reverse process for their one, which relies on intrinsically equivariant neural networks. We made minimal other changes to their codebase and experimental setup otherwise. We found SYMDIFF led to significantly improved performance on both tasks. We additionally compared against several other more sophisticated models than Hoogeboom et al. (2022) that model more complex molecular structure. Despite the relative simplicity of our approach, we obtained results that were on par or superior to these additional methods also. We give an overview of these results now and provide full details in Appendix C.

4.1 QM9

Dataset QM9 (Ramakrishnan et al., 2014) is a common benchmark dataset used for evaluating molecular generation. It consists of molecular properties and atom coordinates for 130k small molecules with up to 9 heavy atoms and a total of 29 atoms including hydrogen. For our experiments, we trained our SYMDIFF method to generate molecules with 3D coordinates, atom types, and atom charges where we explicitly modeled hydrogen atoms. We used the same train-val-test split of 100K-8K-13K as in Anderson et al. (2019).

Our model We took our model to be $\mathsf{sym}_{\gamma_\theta}(k_\theta)(\mathbf{z}_{t-1}|\mathbf{z}_t)$ from equation 8 above. We chose the "backbone" neural network ϵ_θ to be a Diffusion Transformer (DiT) (Peebles & Xie, 2023). This component had 29M parameters, matching the smallest model considered by Peebles & Xie (2023). Likewise, we chose the "backbone" neural network f_θ of the component γ_θ to be a DiT with 2.2M parameters. In this way, our k_θ was much larger than γ_θ, following a similar approach taken by earlier work on deterministic symmetrisation (Kim et al., 2023; Kaba et al., 2023). By construction, DiT architectures are S_N-equivariant, and can be made S_N-invariant using a Set Transformer (Lee et al., 2019) approach, which allowed us to achieve the requirements for these components from Section 3.3. However, DiTs are not required to be $\mathrm{O}(3)$-equivariant, unlike the backbone E-GNN components used by Hoogeboom et al. (2022). Overall, our model had 31.2M parameters in total. To test its scalability, we also trained a larger version of our method with a backbone of 115.6M parameters (SymDiff*), matching the DiT-B model from Peebles & Xie (2023). For this model, γ_θ had 2.2M parameters, giving 117.8M parameters total.

Metrics To measure the quality of generated molecules, we follow standard practice (Hoogeboom et al., 2022; Garcia Satorras et al., 2021a) and report atom stability, molecular stability, validity and uniqueness. We exclude results for the novelty metric for the same reasons as discussed in Vignac & Frossard (2021) and refer the reader to these works for a more extensive discussion of these metrics. For all of our metrics, we used 10,000 samples and report the mean and standard deviation over three evaluation runs. To demonstrate the efficiency of our approach, we also report the number of seconds per epoch, time taken to generate one sample and vRAM.

Baselines We compared our method with EDM (Hoogeboom et al., 2022), whose framework is the most similar to ours. We also considered several other baselines that baked in more sophisticated inductive biases related to molecular generation. These methods included END (Cornet et al., 2024), GeoLDM (Xu et al., 2023) and MUDiff (Hua et al., 2024). For EDM, we trained their 5.3M parameter model using their experimental setup and for the remaining methods we state the results reported in the papers, noting that all the papers used the same train-val-test splits. In addition, we trained an unsymmetrised reverse process with a 29M parameter DiT backbone (DiT), as well as the same model using data augmentation as described in Section 3.4 (DiT-Aug). We also compared our SymDiff model to the case where we simply let γ_θ be obtained using the Haar measure on $\mathrm{O}(3)$, rather than learning this component (SymDiff-H).

Results From Table 3, we see that our SYMDIFF model comfortably beats its EDM counterpart on all metrics, bar uniqueness. Additionally, we see that our model is also competitive with the more recent, sophisticated baselines from the literature, outperforming all of them on validity. Im-

portantly, it is significantly more memory and time efficient (see Table 2). We also see from Table 3 that our method scales significantly better than the EDM approach, both in terms of seconds/epoch, sampling time and vRAM. This is not surprising since the intrinsically equivariant E-GNN used by the EDM model uses message passing for training and inference, which is computationally very costly. In contrast, our symmetrisation approach allows us to use computationally efficient DiT components that parallelise and scale much more effectively (Fei et al., 2024).

Table 1: Test NLL, atom stability, molecular stability, validity and uniqueness on QM9 for 10,000 samples and 3 evaluation runs. We omit the results for NLL where not available.

Method	NLL ↓	Atm. stability (%) ↑	Mol. stability (%) ↑	Val. (%) ↑	Uniq. (%) ↑
GeoLDM	–	98.90 ± 0.10	89.40 ± 0.50	93.80 ± 0.40	92.70 ± 0.50
MUDiff	**-135.50**± 2.10	98.80± 0.20	**89.90**± 1.10	95.30± 1.50	**99.10**± 0.50
END	–	98.90 ± 0.00	89.10 ± 0.10	94.80 ± 0.10	92.60 ± 0.20
EDM	-110.70± 1.50	98.70± 0.10	82.00 ± 0.40	91.90± 0.50	90.70± 0.60
SymDiff*	-133.79± 1.33	**98.92**± 0.03	89.65± 0.10	**96.36**± 0.27	97.66± 0.22
SymDiff	-129.35 ± 1.07	98.74± 0.03	87.49 ± 0.23	95.75 ± 0.10	97.89 ± 0.26
SymDiff-H	-126.53± 0.90	98.57± 0.07	85.51± 0.18	95.22± 0.18	97.98± 0.09
DiT-Aug	-126.81± 1.69	98.64± 0.03	85.85± 0.24	95.10± 0.17	97.98± 0.08
DiT	-127.78± 2.49	98.23± 0.04	81.03± 0.25	94.71± 0.31	97.98± 0.12
Data		99.00	95.20	97.8	100

We attribute the improved performance of our method to the extra architectural flexibility provided by our approach to symmetrisation, enabling it to not only outperform its EDM counterpart but also perform competitively with more sophisticated alternatives (see Tables 1, 2). Our largest model, SymDiff*, outperformed all our baselines on atom stability and validity, and is within variance for molecular stability. By training the model for 500 more epochs, we further found that it surpassed all our baselines on all metrics except uniqueness (see Appendix C.2.3). We conjecture that similar performance improvements can be achieved by using our SYMDIFF approach as a drop-in replacement for the reverse kernels in more sophisticated methods.

Moreover, from Table 1 we note that DiT-Aug performs notably better than the DiT model on all metrics, highlighting its strength as a baseline. Moreover, we observe that our SymDiff model outperforms both SymDiff-H and DiT-Aug on all metrics apart from uniqueness. This shows the benefit of our approach as a more effective way of incorporating $\mathrm{O}(3)$ equivariance than data augmentation.

Table 2: Seconds per epoch, sampling time and vRAM for SymDiff and our baselines. Results for END are omitted as their code was not publicly available.

Method	# Parameters	Sec./epoch (s) ↓	Sampling time (s) ↓	vRAM (GB) ↓
GeoLDM	11.4M	210.93	0.26	27
MuDiff	9.7M	230.87	0.89	36
END	9.4M	–	–	–
EDM	5.4M	88.80	0.27	14
SymDiff*	117.8M	53.40	0.21	16
SymDiff	31.2M	27.20	0.09	7

Ablations As an ablation study, we also tested the effect of making SYMDIFF smaller, and EDM *larger*. For SymDiff, we trained two models of 23.5M (SymDiff$^-$) and 13.5M (SymDiff^{--}) parameters respectively. For EDM, we trained two additional models with 9.5M (EDM$^+$) and 12.4M (EDM^{++}) parameters respectively. For full details see Appendix C.2.1. We found that even our smaller SymDiff models remained competitive. In particular, SymDiff$^-$ gave comparable molecular stability as the second largest EDM model, EDM$^+$, while being approximately 5 times faster in terms of seconds/epoch.

Table 3: NLL, molecular stability, seconds per epoch, sampling time and vRAM for different sizes of SymDiff and EDM. For additional performance metrics see Appendix C.2.

Method	NLL ↓	Mol. stability (%) ↑	Sec./epoch (s) ↓	Sampling time (s) ↓	vRAM (GB) ↓
EDM^{++}	-119.12$\pm$1.41	85.68$\pm$0.83	160.60	0.56	23
EDM^{+}	-110.97$\pm$1.42	84.63$\pm$0.16	192.60	0.46	23
EDM	-110.70$\pm$1.50	82.00 $\pm$0.40	88.80	0.27	14
SymDiff	-129.35$\pm$1.07	87.49$\pm$0.23	27.20	0.09	7
SymDiff^{-}	-125.40$\pm$0.63	83.51$\pm$0.24	24.87	0.08	6
SymDiff^{--}	-110.68 $\pm$2.55	71.25 $\pm$0.50	20.60	0.07	5

4.2 GEOM-DRUGS

Dataset, model and training GEOM-Drugs (Axelrod & Gomez-Bombarelli, 2022) is a larger and more complicated dataset than QM9, containing 430,000 molecules with up to 181 atoms. We processed the dataset in the same way as Hoogeboom et al. (2022), where we again model hydrogen explicitly. We used the SymDiff model from earlier and trained our model for 40 epochs to match the same total number of gradient steps as used by Hoogeboom et al. (2022).

Metrics and baselines We report the same metrics as for QM9 but exclude molecular stability and uniqueness for the same reasons discussed in Hoogeboom et al. (2022). We compared our method to the EDM model used by Hoogeboom et al. (2022) for GEOM-Drugs, as well as the other baseline architectures reported for QM9. Due to the increased computational costs involved for GEOM-Drugs, we restate the results reported in the original papers rather than retraining these models ourselves.

Results From Table 4 we again see that our approach comfortably outperforms its EDM counterpart. It is also again competitive with the more sophisticated baselines. Like with QM9, our SymDiff models were significantly less costly in terms of compute time and memory usage compared with EDM (see Appendix C.3). In fact, when we tried to run the EDM model it resulted in out-of-memory errors on our NVIDIA H100 80GB GPU. (Hoogeboom et al. (2022) avoid this by training EDM on 3$\times$ NVIDIA RTX A6000 48GB GPUs.)

Table 4: Test NLL, atom stability and validity on GEOM-Drugs for 10,000 samples and 3 evaluation runs. GeoLDM and EDM ran their results for just one evaluation run. We omit the results for NLL and validity where not available.

Method	NLL ↓	Atm. stability (%) ↑	Val. (%) ↑
GeoLDM	–	84.4	99.3
END	–	**87.8**$\pm$0.99	92.9$\pm$0.3
EDM	-137.1	81.3	–
SymDiff	**-301.21**$\pm$0.53	86.16$\pm$0.05	99.27$\pm$0.1
Data		86.50	99.9

5 CONCLUSION

We have introduced SYMDIFF: a novel, lightweight, and scalable framework for constructing equivariant diffusion models based on stochastic symmetrisation. We applied this approach to $\mathrm{E}(3)$-equivariance for N-body data, obtaining an overall model that is stochastically equivariant but that does not rely on any intrinsically equivariant neural network subcomponents. Our approach leads to significantly greater modelling flexibility, which allows leveraging powerful off-the-shelf architectures such as Transformers (Vaswani et al., 2017). We showed empirically that this leads overall to improved performance on several relevant benchmarks.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pp. 1–3, 2024.

Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *Advances in neural information processing systems*, 32, 2019.

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

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

Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1): 2453, 2022.

Benjamin Bloem-Reddy and Yee Whye Teh. Probabilistic symmetries and invariant neural networks. *Journal of Machine Learning Research*, 21(90):1–61, 2020.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Diego Canez, Nesta Midavaine, and Thijs Stessen. Effect of equivariance on training dynamics. `https://gram-blogposts.github.io/blog/2024/relaxed-equivariance/`, 2024. Accessed: 2024-09-30.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

François RJ Cornet, Grigory Bartosh, Mikkel N Schmidt, and Christian A Naesseth. Equivariant neural diffusion for molecule generation. In *ICML 2024 AI for Science Workshop*, 2024.

Rob Cornish. Stochastic Neural Network Symmetrisation in Markov Categories. *arXiv preprint arXiv:2406.11814*, 2024.

Alexandre Duval, Simon V Mathis, Chaitanya K Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D Malliaros, Taco Cohen, Pietro Lio, Yoshua Bengio, and Michael Bronstein. A hitchhiker's guide to geometric gnns for 3d atomic systems. *arXiv preprint arXiv:2312.07511*, 2023a.

Alexandre Agm Duval, Victor Schmidt, Alex Hernández-Garcıa, Santiago Miret, Fragkiskos D Malliaros, Yoshua Bengio, and David Rolnick. Faenet: Frame averaging equivariant gnn for materials modeling. In *International Conference on Machine Learning*, pp. 9013–9033. PMLR, 2023b.

Nadav Dym, Hannah Lawrence, and Jonathan W Siegel. Equivariant frames and the impossibility of continuous canonicalization. *arXiv preprint arXiv:2402.16077*, 2024.

Bryn Elesedy and Sheheryar Zaidi. Provably strict generalisation benefit for equivariant models. In *International conference on machine learning*, pp. 2959–2969. PMLR, 2021.

Zhengcong Fei, Mingyuan Fan, Changqian Yu, Debang Li, and Junshi Huang. Scaling diffusion transformers to 16 billion parameters. *arXiv preprint arXiv:2407.11633*, 2024.

Tobias Fritz. A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020.

Victor Garcia Satorras, Emiel Hoogeboom, Fabian Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows. *Advances in Neural Information Processing Systems*, 34:4181–4192, 2021a.

Victor Garcia Satorras, Emiel Hoogeboom, Fabian Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 4181–4192. Curran Associates, Inc., 2021b. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/21b5680d80f75a616096f2e791affac6-Paper.pdf.

Yoav Gelberg, Tycho FA van der Ouderaa, Mark van der Wilk, and Yarin Gal. Variational inference failures under model symmetries: Permutation invariant posteriors for bayesian neural networks. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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

Chenqing Hua, Sitao Luan, Minkai Xu, Zhitao Ying, Jie Fu, Stefano Ermon, and Doina Precup. Mudiff: Unified diffusion for complete molecule generation. In *Learning on Graphs Conference*, pp. 33–1. PMLR, 2024.

Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *International Conference on Machine Learning*, pp. 15546–15566. PMLR, 2023.

Olav Kallenberg. *Foundations of modern probability*, volume 2. Springer, 1997.

Olav Kallenberg. *Foundations of Modern Probability*. Springer, 2 edition, 2002.

Jinwoo Kim, Dat Nguyen, Ayhan Suleymanzade, Hyeokjun An, and Seunghoon Hong. Learning probabilistic symmetrization for architecture agnostic equivariance. *Advances in Neural Information Processing Systems*, 36:18582–18612, 2023.

Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36, 2024.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching. *Advances in Neural Information Processing Systems*, 36, 2024.

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

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Shengjie Luo, Tianlang Chen, Yixian Xu, Shuxin Zheng, Tie-Yan Liu, Liwei Wang, and Di He. One transformer can understand both 2d & 3d molecular data. In *The Eleventh International Conference on Learning Representations*, 2022.

Francesco Mezzadri. How to generate random matrices from the classical compact groups. *arXiv preprint math-ph/0609050*, 2006.

Arnab Kumar Mondal, Siba Smarak Panigrahi, Oumar Kaba, Sai Rajeswar Mudumba, and Siamak Ravanbakhsh. Equivariant adaptation of large pretrained models. *Advances in Neural Information Processing Systems*, 36:50293–50309, 2023.

Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *arXiv preprint arXiv:1811.01900*, 2018.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Stefanos Pertigkiozoglou, Evangelos Chatzipantazis, Shubhendu Trivedi, and Kostas Daniilidis. Improving equivariant model training via constraint relaxation. *arXiv preprint arXiv:2408.13242*, 2024.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Omri Puny, Matan Atzmon, Heli Ben-Hamu, Ishan Misra, Aditya Grover, Edward J Smith, and Yaron Lipman. Frame averaging for invariant and equivariant network design. *arXiv preprint arXiv:2110.03336*, 2021.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

P. Selinger. *A Survey of Graphical Languages for Monoidal Categories*, pp. 289–355. Springer Berlin Heidelberg, 2010. ISBN 9783642128219. doi: 10.1007/978-3-642-12821-9_4. URL http://dx.doi.org/10.1007/978-3-642-12821-9_4.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Philipp Thölke and Gianni De Fabritiis. Torchmd-net: equivariant transformers for neural network based molecular potentials. *arXiv preprint arXiv:2202.02541*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Clement Vignac and Pascal Frossard. Top-n: Equivariant set and graph generation without exchangeability. *arXiv preprint arXiv:2110.02096*, 2021.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Yuyang Wang, Ahmed AA Elhag, Navdeep Jaitly, Joshua M Susskind, and Miguel Ángel Bautista. Swallowing the bitter pill: Simplified scalable conformer generation. In *Forty-first International Conference on Machine Learning*, 2024.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

Minkai Xu, Alexander S Powers, Ron O Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pp. 38592–38610. PMLR, 2023.

Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55(1):407–474, 2022.

Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se(3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023.

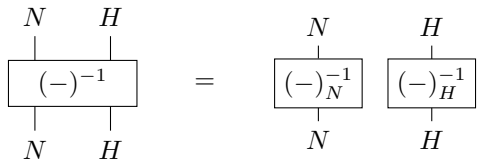
Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

A PROOFS

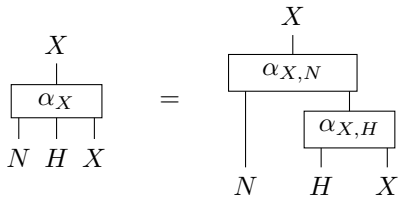
A.1 PROOF OF PROPOSITION 1

Proof. This follows by appropriately instantiating Example 6.3 of Cornish (2024), whose definitions and notation we will import freely here. We will also make use of *string diagrams* (Selinger, 2010), an introduction to which can be found in Section 2 of Cornish (2024). Intuitively, these diagrams represent (possibly stochastic) computational processes that should be read up the page, with the inputs applied at the bottom, and outputs produced at the top.

First, we will set ρ in Example 6.3 of Cornish (2024) to be the trivial action of H on N, so that the semidirect product $N \rtimes_\rho H$ becomes simply the product $N \times H$. As per Definition 3.7 of Cornish (2024), the inversion operation for $N \times H$ can then be written as

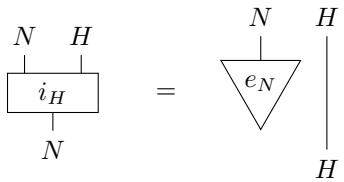


Similarly, by Remark 3.4 of Cornish (2024), we can write the action α_X on X in the form

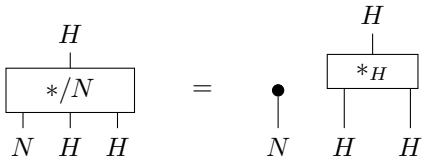


(16)

for some action $\alpha_{X,H}$ of H on X, and some action $\alpha_{X,N}$ of N and X. We can do similarly for α_Y. In addition, as per Example 3.23 of Cornish (2024), the inclusion homomorphism i_H, which plays the role of the i_N-coset map in Example 6.3, can be written as



Finally, the coset action $*/H$ of $N \times H$ on H is just

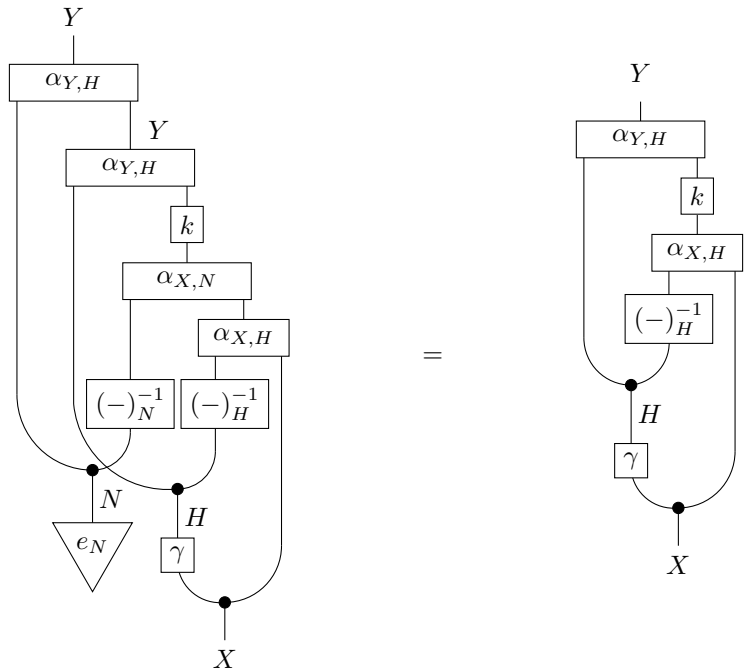


where $*_H$ denotes the group multiplication operation for H. Notice that, like in equation 16, the right-hand side here is decomposed into a pair of an N-action (namely the trivial action) and an H-action (namely $*_H$). By Remark 3.4 of Cornish (2024), it therefore follows that $\gamma : X \to H$ is equivariant with respect to α_X and $*/H$ if and only if it is

- N-equivariant with respect to $\alpha_{X,N}$ and the trivial action on H, and
- H-equivariant with respect to $\alpha_{X,H}$ and the action $*_H$

In other words, γ must be N-invariant and H-equivariant.

Now suppose we have such a γ. Substituting all the components above into the end-to-end procedure from Section 5.4 of Cornish (2024), we obtain $\mathsf{sym}_\gamma(k) : X \to Y$ as follows:



(17)

where on the right-hand side we have simplified the operations involving the identity term e_N via the basic group axioms (e.g. $(-)_N^{-1} \circ e_N$ becomes simply e_N). As shown by Cornish (2024), when $k : X \to Y$ is H-equivariant with respect to $\alpha_{X,N}$ and $\alpha_{Y,N}$, it always holds that $\mathsf{sym}_\gamma(k)$ is equivariant with respect to the $(N \times H)$-actions α_X and α_Y.

To prove Proposition 1 here, we now simply instantiate the above discussion in the Markov category $\mathsf{C} \coloneqq \mathsf{Stoch}$, so that all the components involved become Markov kernels. Concretely, we take $X \coloneqq \mathcal{X}$, $Y \coloneqq \mathcal{Y}$, $N \coloneqq \mathcal{H}$, and $H \coloneqq \mathcal{G}$. A procedure for sampling from the symmetrised kernel $\mathsf{sym}(\gamma)(d\mathbf{y}|\mathbf{x})$ may now be read off from equation 17 as follows:

$$g \sim \gamma(dg|\mathbf{x}) \qquad \mathbf{y}_0 \sim k(d\mathbf{y}|g^{-1} \cdot \mathbf{x}) \qquad \mathbf{y} \coloneqq g \cdot \mathbf{y}_0,$$

as in the statement of this result. □

A.2 Proof of Proposition 2

Proof. For simplicity, we assume $k(d\mathbf{y}|\mathbf{x})$ has a density $k(\mathbf{y}|\mathbf{x})$ with respect to the Lebesgue measure μ on $\mathcal{Y} = \mathbb{R}^n$. To derive the density of $k(d\mathbf{y}|g, \mathbf{x})$ where $g \in \mathcal{G}$ is some fixed group element, we note that for $\mathbf{y} \sim k(d\mathbf{y}|g, \mathbf{x})$, we have $\mathbf{y} = g \cdot \mathbf{y}_0$ where $\mathbf{y}_0 \sim k(\mathbf{y}|g^{-1} \cdot \mathbf{x})$. Hence, by the change-of-variables formula, we can conclude that the density of $k(\mathbf{y}|g, \mathbf{x})$ exists and has the form:

$$k(\mathbf{y}|g, \mathbf{x}) = k(g^{-1} \cdot \mathbf{y}|g^{-1} \cdot \mathbf{x}) \left| \frac{\partial (g^{-1} \cdot \mathbf{y})}{\partial \mathbf{y}} \right|.$$

Hence, we see that when the action of $\mathcal{G}$ has unit Jacobian, the density of $k(\mathbf{y}|g, \mathbf{x}) = k(g^{-1} \cdot \mathbf{y}|g^{-1} \cdot \mathbf{x})$.

Further, suppose we have $g \sim \gamma(dg|\mathbf{x})$ and $\mathbf{y} \sim k(\mathbf{y}|g, \mathbf{x})$ - i.e. $\mathbf{y} \sim \mathsf{sym}_\gamma(k)(d\mathbf{y}|\mathbf{x})$. It is the case that for an arbitrary (Borel) measurable set A in $\mathcal{Y}$, we have

$$\mathsf{sym}_\gamma(k)(\mathbf{y} \in A|\mathbf{x}) = \int_{\mathcal{G}} k(\mathbf{y} \in A|g, \mathbf{x}) \, \gamma(dg|\mathbf{x}).$$

Since we have shown above that $k(d\mathbf{y}|g, \mathbf{x})$ has a density, we can express this as

$$\begin{aligned} \mathsf{sym}_\gamma(k)(\mathbf{y} \in A|\mathbf{x}) &= \int_{\mathcal{G}} \left(\int_A k(\mathbf{y}|g, \mathbf{x}) \, \mu(d\mathbf{y}) \right) \gamma(dg|\mathbf{x}) \\ &= \int_A \mathbb{E}_{\gamma(dg|\mathbf{x})} \left[k(\mathbf{y}|g, \mathbf{x}) \right] \mu(d\mathbf{y}). \end{aligned}$$

where we use Fubini's theorem for the second line as all quantities are non-negative. Hence, we can conclude that the density of $\mathsf{sym}_\gamma(k)$ exists and has the form $\mathsf{sym}_\gamma(k) = \mathbb{E}_{\gamma(dg|\mathbf{x})}[k(\mathbf{y}|g, \mathbf{x})]$. □

A.3 Proof of Proposition 3

Proof. The standard diffusion objective with data augmentation distributed according to the Haar measure λ is given by

$$\mathcal{L}_t^{\text{aug}} = \mathbb{E}_{q(\mathbf{z}_0),\epsilon\sim\mathcal{N}_{\mathcal{U}}(0,\mathbf{I}),\lambda(dR)}\left[\frac{1}{2}w(t)\|\epsilon - \epsilon_\theta(\alpha_t R\cdot\mathbf{z}_0 + \sigma_t\epsilon)\|^2\right] \tag{18}$$

$$= \mathbb{E}_{q(\mathbf{z}_0),\epsilon\sim\mathcal{N}_{\mathcal{U}}(0,\mathbf{I}),\lambda(dR)}\left[\frac{1}{2}w(t)\left\|\epsilon - \epsilon_\theta(R\cdot(\alpha_t\mathbf{z}_0 + \sigma_t R^T\cdot\epsilon))\right\|^2\right] \tag{19}$$

$$= \mathbb{E}_{q(\mathbf{z}_0),\epsilon'\sim\mathcal{N}_{\mathcal{U}}(0,\mathbf{I}),\lambda(dR)}\left[\frac{1}{2}w(t)\|R\cdot\epsilon' - \epsilon_\theta(R\cdot(\alpha_t\mathbf{z}_0 + \sigma_t\epsilon'))\|^2\right] \tag{20}$$

$$= \mathbb{E}_{q(\mathbf{z}_0),\epsilon\sim\mathcal{N}_{\mathcal{U}}(0,\mathbf{I}),\lambda(dR)}\left[\frac{1}{2}w(t)\left\|\epsilon - R^T\cdot\epsilon_\theta(R\cdot(\alpha_t\mathbf{z}_0 + \sigma_t\epsilon))\right\|^2\right], \tag{21}$$

where we use the fact that $\epsilon' = R^T\cdot\epsilon$ is distributed according to $\mathcal{N}_{\mathcal{U}}(0,\mathbf{I})$ as R, ϵ are independent in the expectation, and that the action of $R \in \mathrm{O}(3)$ preserves the L2 norm. To conclude, we note that it is a standard result that if $R \sim \lambda$, the inverse R^T is also distributed according to the Haar measure λ. Hence, we see that $\mathcal{L}_t^{\text{aug}}$ coincides with

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{z}_0),\epsilon\sim\mathcal{N}_{\mathcal{U}}(0,\mathbf{I}),\lambda(dR)}\left[\frac{1}{2}w(t)\left\|\epsilon - R\cdot\epsilon_\theta(R^T\cdot\mathbf{z}_t)\right\|^2\right], \tag{22}$$

where $\mathbf{z}_t = \alpha_t\mathbf{z}_0 + \sigma_t\epsilon$. □

B Model Architecture

Below, we outline the architectures used for ϵ_θ and γ_θ. Both components rely on Diffusion Transformers (DiTs) (Peebles & Xie, 2023) using the official PyTorch implementation at `https://github.com/facebookresearch/DiT`. We also state the hyperparameters that we kept fixed for both our QM9 and GEOM-Drugs experiments. Any hyperparameters that differed between the datasets are discussed in their respective sections later in the Appendix.

We emphasise that our architecture choices were not extensively tuned as the main purpose of our experiments was to show that we can use generic architectures for equivariant diffusion models. We arrived at the below architecture through small adjustments from experimenting with DiT models in the context of molecular generation, which we stuck with in our final experiments.

B.1 Architecture of ϵ_θ

As we need ϵ_θ to be an S_N-equivariant architecture, we take ϵ_θ to be parametrised in terms of a DiT model ϵ'_θ with n_{layers} layers, n_{head} attention heads and hidden size n_{size} where to ensure the output lies in $\mathcal{U}$ we project the output via π. For the MLP layers, we use SwiGLU activations Shazeer (2020) instead of the standard GELU, where the ratio of the hidden size of the SwiGLU to the hidden size of ϵ'_θ is 2. We do not use the default Fourier embeddings for the inputs and instead use linear layers to project our inputs to the correct size; for time embeddings, we use the default. We use Gaussian positional embeddings from Luo et al. (2022) as additional features that we concatenate to our inputs. To compute this, we let

$$\psi^k_{(i,j)} = -\frac{1}{\sqrt{2\pi}|\sigma^k|}\exp\left(-\frac{1}{2}\left(\frac{\left\|\mathbf{x}^{(j)} - \mathbf{x}^{(j)}\right\| - \mu^k}{|\sigma^k|}\right)\right),$$

where $k = 1, \ldots, K$ is the number of basis kernels we use and $\mu^k, \sigma^k \in \mathbb{R}$ are learnable parameters. We define $\psi_{(i,j)} = (\psi^1_{(i,j)}, \ldots, \psi^K_{(i,j)})^T \in \mathbb{R}^{K\times 1}$. We then compute our positional embeddings by $\Psi_i = \frac{1}{N}\sum_{j=1}^N \psi_{(i,j)}W_D$ where $W_D \in \mathbb{R}^{K\times n_{\text{emb}}}$ is a learnable matrix, and we concatenate these to form our embeddings $\mathbf{\Psi} = [\Psi_1, \ldots, \Psi_N] \in \mathbb{R}^{N\times n_{\text{emb}}}$.

To compute ϵ_θ, given the input $\mathbf{z} = [\mathbf{x}, \mathbf{h}] \in \mathbb{R}^{N\times(3+d)}$, we use a learnable linear layer $W_I \in \mathbb{R}^{(3+d)\times n_z}$ to project $\mathbf{z}$ to $\mathbf{z}' = \mathbb{R}^{N\times n_z}$, and we compute the positional embeddings $\mathbf{\Psi} \in \mathbb{R}^{N\times n_{\text{emb}}}$, as outlined above, where $n_{\text{emb}} = n_{\text{size}} - n_z$. We then pass $[\mathbf{z}', \mathbf{\Psi}] \in \mathbb{R}^{N\times n_{\text{size}}}$ to our DiT model ϵ'_θ.

B.2 Architecture of γ_θ

We construct γ_θ following the recursive setup in Section 3.3. We take f_θ to be comprised of a DiT model with m_{layers} layers, m_{head} attention heads and hidden size m_{size}, and the same SwiGLU setup as above. To ensure that the architecture is S_N-invariant, we use the approach in Zaheer et al. (2017), where we further take the token-wise average of the representations from the DiT model (before the final output layer) to give a vector in $\mathbb{R}^{m_{\text{size}}}$. We then pass this into a small MLP $\mathbf{x} \mapsto \text{GELU}(\mathbf{x}W_1)W_2$ where $W_1 \in \mathbb{R}^{m_{\text{size}}\times m'_{\text{size}}}, W_2 \in \mathbb{R}^{m'_{\text{size}}\times(3\times 3)}$ are learnable linear layers, which produces a matrix $\mathbb{R}^{3\times 3}$. We apply the QR decomposition to convert this into an orthogonal matrix which we then return.

To compute γ_θ given the input $\mathbf{z} = [\mathbf{x}, \mathbf{h}] \in \mathbb{R}^{N\times(3+d)}$, we discard $\mathbf{h}$ and compute $\mathbf{\Psi} \in \mathbb{R}^{N\times n_{\text{emb}}}$ with the same parameters used in ϵ_θ from $\mathbf{x}$. We also sample $\eta \sim \mathcal{N}_{\mathcal{U}}(0, \mathbf{I})$ of size $\mathbb{R}^{N\times m_{\text{noise}}}$ and an orthogonal matrix $R \in \mathbb{R}^{3\times 3}$ from the Haar measure on $\text{O}(3)$. We then project the concatenated input $[R^T \cdot \mathbf{x}, \eta, \mathbf{\Psi}] \in \mathbb{R}^{N\times(3+m_{\text{noise}}+n_{\text{emb}})}$ by a learnable linear layer $W_G \in \mathbb{R}^{(3+m_{\text{noise}}+n_{\text{emb}})\times m_{\text{size}}}$ to $\mathbf{x}' = \mathbb{R}^{N\times m_{\text{size}}}$ (we note that $\mathbf{\Psi}$ is invariant to $\text{O}(3)$). We then pass $\mathbf{x}'$ through f_θ and apply R to the output - i.e. we return $R \cdot f_\theta(\mathbf{x}')$.

B.3 Hyperparameters for γ_θ and ϵ_θ

For both QM9 and GEOM-Drugs we fixed the following hyperparameters. For ϵ_θ, we set $n_{\text{size}} = K \approx \frac{1}{2}n_{\text{size}}$. For γ_θ, we set $m_{\text{noise}} = 3$ and $m'_{\text{size}} = 4m_{\text{size}}$.

C Experimental details

C.1 Zeroth likelihood term $\mathcal{L}_0$

We have a presented a framework for parametrising and optimising $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$ for $t > 1$ in Section 3.3 where p_θ is obtained via stochastic symmetrisation. This corresponds to the $\mathcal{L}_t$ terms where $t > 0$ in our objective in equation 9. We note however that standard diffusion models usually choose a different parametrisation for $p_\theta(\mathbf{z}_0|\mathbf{z}_1)$ as this corresponds to the final generation step. Depending on the modelling task, this requires a different approach compared to the other reverse kernels. For example, in Hoogeboom et al. (2022), $p_\theta(\mathbf{z}_0|\mathbf{z}_1)$ is defined as the product of densities $p_\theta^{\text{cont}}(\mathbf{x}_0|\mathbf{z}_1)p_\theta^{\text{disc}}(\mathbf{h}_0|\mathbf{z}_1)$ where p_θ^{disc} implements a quantisation step converting the continuous latent $\mathbf{z}_1$ to discrete values $\mathbf{h}_0$, while p_θ^{cont} is still a Gaussian distribution generating continuous geometric features $\mathbf{x}_0$ from $\mathbf{z}_1$. In particular, we have that $p_\theta^{\text{cont}}(\mathbf{x}_0|\mathbf{z}_1) = \mathcal{N}_{\mathcal{U}}(\mathbf{x}_0; \mathbf{x}_1/\alpha_1 - \sigma_1/\alpha_1\epsilon_\theta^{(\mathbf{x})}(\mathbf{z}_1), \sigma_1^2/\alpha_1^2\mathbf{I})$ where $\epsilon_\theta : \mathcal{Z} \to \mathcal{Z}$ is some S_N-invariant neural network and $\epsilon_\theta^{(\mathbf{x})}$ denotes the $\mathbf{x}$ component of the output of ϵ_θ.

We note that our above methodology can still account for this case by defining the symmetrised kernel in terms of

$$p_\theta(\mathbf{z}_0|\mathbf{z}_1) = \mathsf{sym}_{\gamma_\theta}(k_\theta)(\mathbf{z}_0|\mathbf{z}_1), \qquad k_\theta(\mathbf{z}_0|\mathbf{z}_1) = p_\theta^{\text{cont}}(\mathbf{x}_0|\mathbf{z}_1)p_\theta^{\text{disc}}(\mathbf{h}_0|\mathbf{z}_1)$$

We can follow the same discussion in Section 3.3 to conclude that

$$p_\theta(\mathbf{z}_0|\mathbf{z}_1) = \mathbb{E}_{\gamma_\theta(dR|\mathbf{z}_1)}\left[k_\theta(\mathbf{z}_0|R, \mathbf{z}_1)\right], \qquad k_\theta(\mathbf{z}_0|R, \mathbf{z}_1) = k_\theta^{\text{cont}}(\mathbf{x}_0|R, \mathbf{z}_1)k_\theta^{\text{disc}}(\mathbf{h}_0|R, \mathbf{z}_1),$$

where $k_\theta^{\text{cont}}(\mathbf{z}_0|R, \mathbf{z}_1) = \mathcal{N}(\mathbf{x}_0; \mathbf{x}_1/\alpha_1 - \sigma_1/\alpha_1 R \cdot \epsilon_\theta(R^T \cdot \mathbf{z}_1), \sigma_1^2/\alpha_1^2\mathbf{I})$ and $k_\theta^{\text{disc}}(\mathbf{h}_0|R, \mathbf{z}_1) = p_\theta^{\text{disc}}(\mathbf{h}_0|R^T \cdot \mathbf{z}_1)$. This allows us to decompose $\mathcal{L}_0$ into the form in equation 9 and to tractably optimise this objective since we have access to the density $k_\theta(\mathbf{z}_0|R, \mathbf{z}_1)$.

C.2 QM9

C.2.1 MODEL HYPERPARAMETERS

SymDiff Table 5 shows the hyperparameters for the ϵ_θ backbone of the k_θ component of the SYMDIFF models used for QM9. The remaining hyperparameters were kept the same as in Appendix B.3.

Table 5: Choice of n_{size}, n_{layers}, n_{heads} for the ϵ_θ of the SYMDIFF models used for QM9.

Model	# Parameters	n_{size}	n_{layers}	n_{heads}
SymDiff*	115.6M	768	12	12
SymDiff	29M	384	12	6
SymDiff$^-$	21.3M	360	12	6
SymDiff^{--}	11.3M	294	12	6

For the f_θ backbone of the γ_θ component, we set $m_{\text{size}} = 128$, $m_{\text{layers}} = 12$, $m_{\text{heads}} = 6$.

EDM Table 6 shows the hyperparameters for the EDM models that we used for our QM9 experiments. The remaining model hyperparameters were kept the same as those in Hoogeboom et al. (2022).

Table 6: Choices of the hyperparameters n_f (# features per layer), n_l (number of layers) for the EDM models used for QM9.

Model	# Parameters	n_f	n_l
EDM^{++}	12.4M	332	12
EDM^{+}	9.5M	256	16
EDM	5.3M	256	9

C.2.2 OPTIMISATION

For our optimisation, we followed Peebles & Xie (2023) and used AdamW (Loshchilov & Hutter) with a batch size of 256. We chose a learning rate of 10^{-4} and weight decay of 10^{-12} for our 31.2M parameter model by searching over a small grid of 3 values for each. To match the same number of steps as in Hoogeboom et al. (2022), we trained our model for 4350 epochs.

We applied the same optimization hyperparameters from our 31.2M model to all other SYMDIFF models. For the EDM models, we followed the default hyperparameters from Hoogeboom et al. (2022). In our augmentation experiments, we first tuned the learning rate and weight decay for the DiT model, keeping all other optimization hyperparameters unchanged. These tuned values were then applied to DiT-Aug.

C.2.3 ADDITIONAL RESULTS

Here we restate the results from Table 7 but now include the results for SymDiff* when trained for an additional 500 epochs (we refer to this as SymDiff-Long*). We see that SymDiff-Long* outperforms all our baselines bar uniqueness. For NLL it is within variance with that of MUDiff.

Table 7: Test NLL, atom stability, molecular stability, validity and uniqueness on QM9 for 10,000 samples and 3 evaluation runs. We omit the results for NLL where not available.

Method	NLL ↓	Atm. stability (%) ↑	Mol. stability (%) ↑	Val. (%) ↑	Uniq. (%) ↑
GeoLDM	–	98.90 $\pm$0.10	89.40 $\pm$0.50	93.80 $\pm$0.40	92.70 $\pm$0.50
MUDiff	**-135.50**$\pm$2.10	98.80$\pm$0.20	89.90$\pm$1.10	95.30$\pm$1.50	**99.10**$\pm$0.50
END	–	98.90 $\pm$0.00	89.10 $\pm$0.10	94.80 $\pm$0.10	92.60 $\pm$0.20
EDM	-110.70$\pm$1.50	98.70$\pm$0.10	82.00 $\pm$0.40	91.90$\pm$0.50	90.70$\pm$0.60
SymDiff-Long*	-135.04$\pm$0.64	**98.98**$\pm$0.05	**90.26**$\pm$0.38	**96.38**$\pm$0.27	97.60$\pm$0.10
SymDiff*	-133.79$\pm$1.33	**98.92**$\pm$0.03	89.65$\pm$0.10	**96.36**$\pm$0.27	97.66$\pm$0.22
Data		99.00	95.20	97.8	100

C.3 GEOM-DRUGS

Like with QM9, we report the seconds per epoch, sampling time (s) and vRAM for the models used in Table 4. We exclude END as their code is not publicly available. We omit the results for EDM and GeoLDM as were unable to run their code on our NVIDIA H100 80GB GPU.

Table 8: Total wall time, iterations per second sampling time for different sizes of SymDiff.

Method	# Parameters	Sec./epoch	Sampling time (s)	vRAM (GB)
GeoLM	5.5M	–	–	–
EDM	2.4M	–	–	–
SymDiff	31.2M	4336.82	0.39	63

C.4 PRETRAIN-FINETUNING

To further explore the flexibility of our approach, we experimented with using it in the pretrain-finetune framework, similar to Mondal et al. (2023). Using QM9, we took the trained DiT model from Table 1 and substituted it as the ϵ_θ for our SymDiff model, while keeping the same architecture and hyperparameters for f_θ. We tested two setups: finetuning both ϵ_θ and f_θ (DiT-FT) and freezing ϵ_θ while tuning only f_θ (DiT-FT-Freeze). The same training procedure and optimization hyperparameters were used, except we now trained our models for 800 epochs and used a larger grid for learning rate and weight decay tuning. Specifically, we searched first for the optimal learning rate in $[10^{-3}, 8 \times 10^{-4}, 2 \times 10^{-4}, 10^{-4}]$ and for the optimal weight decay in $[0, 10^{-12}, 2 \times 10^{-12}]$. We found the optimal learning rate and weight decay to be 10^{-3} and 2×10^{-12}.

Table 9: Test NLL, atom stability, molecular stability, validity and uniqueness on QM9 for 10,000 samples and 3 evaluation runs.

Method	NLL ↓	Atm. stability (%) ↑	Mol. stability (%) ↑	Val. (%) ↑	Uniq. (%) ↑
SymDiff	**-129.35**$\pm$1.07	**98.74**$\pm$0.03	**87.49**$\pm$0.23	**95.75**$\pm$0.10	97.89$\pm$0.26
DiT-FT	-111.66$\pm$1.22	98.43$\pm$0.03	83.27 $\pm$0.39	94.19$\pm$0.16	98.17$\pm$0.26
DiT-FT-Freeze	-43.29$\pm$3.73	95.68$\pm$0.02	55.02$\pm$0.38	90.48$\pm$0.24	**99.06**$\pm$0.13
DiT	-127.78$\pm$2.49	98.23$\pm$0.04	81.03$\pm$0.25	94.71$\pm$0.31	97.98$\pm$0.12

From Table 9, we observe that finetuning both ϵ_θ and f_θ improves performance over the DiT model, even with our minimal optimization tuning. However, finetuning only f_θ leads to worse results, indicating that end-to-end training or finetuning the whole model is necessary. This underscores the flexibility of our approach and its potential for easy and efficient symmetrisation of pretrained DiT models with an unconstrained f_θ.

D EXTENSION TO SCORE MATCHING AND FLOW MATCHING

In this section, we discuss how to extend stochastic symmetrisation to score and flow-based generative models to give an analogue of SYMDIFF to these paradigms. For clarity of presentation, we

consider all models to be defined for N-body systems living in the full space $\mathcal{Z} = \mathbb{R}^{N\times 3}$ where we wish to obtain a $S_N \times \mathrm{O}(3)$-equivariant model - i.e. we do not consider non-geometric features or translation invariance. Although, we note that the below discussion can be extended to such settings in the natural way as presented for diffusion models above.

D.1 Score Matching

Score-based generative models (SGMs) (Song et al., 2020) are the continuous-time analogue of diffusion models. SGMs consider the forward noising process $\mathbf{x}_t \sim p_t$ for $t \in [0, T]$ defined by the following stochastic differential equation (SDE) with the initial condition $\mathbf{x}_0 \sim p_{\text{data}}$:

$$d\mathbf{x}_t = f(\mathbf{x}_t, t)dt + g(t)d\mathbf{w}, \tag{23}$$

for some choice of functions $f : \mathcal{Z} \times [0, T] \to \mathcal{Z}$ and $g : [0, T] \to \mathbb{R}$, and where $\mathbf{w}$ is a standard Weiner process.

The corresponding backward process is shown in Anderson (1982) to take the form:

$$d\mathbf{x}_t = \left[f(\mathbf{x}_t, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\right] dt + g(t)d\bar{\mathbf{w}}, \tag{24}$$

where $\bar{\mathbf{w}}$ is a standard Weiner process and time runs backwards from T to 0. Hence, given samples $\mathbf{x}_T \sim p_T$ and access to the score of the marginal distributions $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$, we can obtain samples from p_{data} by simulating the backward process in equation 24.

By considering the Euler–Maruyama discretisation of equation 24, we can represent the sampling scheme of a SGM in terms of the Markov chain $p_T(\mathbf{x}_T) \prod_{i=1}^{n} p(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i})$, where the time-points t_i are uniformly spaced in $[0, T]$ - i.e. $t_i = i\Delta t$ where $\Delta t = T/n$ - and the reverse transition kernels are given by:

$$p(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i}) = \mathcal{N}\left(\mathbf{x}_{t_{i-1}}; \mathbf{x}_{t_i} + \Delta t \left\{f(\mathbf{x}_{t_i}, t_i) - g(t_i)^2 \nabla_x \log p_{t_i}(\mathbf{x}_{t_i})\right\}, g(t_i)^2 \Delta t \mathbf{I}\right).$$

In what follows, we additionally assume that $f(\cdot, t)$ is $S_N \times \mathrm{O}(3)$-equivariant for all $t \in [0, T]$. This is true for common choices of f which take f to be linear in $\mathbf{x}_t$.

Stochastic symmetrisation In order to learn an approximation to the transition kernels via stochastic symmetrisation, we can parametrise the reverse transition kernels, in a similar fashion as for diffusion models, by

$$p_\theta(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i}) = \mathsf{sym}_{\gamma_\theta}(k_\theta)(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i}),$$

where we take $k_\theta(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i}) = \mathcal{N}(\mathbf{x}_{t_{i-1}}; \mu_\theta(\mathbf{x}_{t_i}), g(t_i)^2 \Delta t \mathbf{I})$. We define μ_θ by the following parametrisation[4]:

$$\mu_\theta(\mathbf{x}_{t_i}) = \mathbf{x}_{t_i} + \Delta t \left\{f(\mathbf{x}_{t_i}, t_i) - g(t_i)^2 s_\theta(\mathbf{x}_{t_i})\right\},$$

where we take $s_\theta : \mathcal{Z} \to \mathcal{Z}$ to be a S_N-equivariant neural network which aims to learn an approximation to the true score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$. This ensures that k_θ is S_N-invariant. Additionally, we assume that $\gamma_\theta : \mathcal{Z} \to \mathrm{O}(3)$ is some choice of a S_N-invariant and $\mathrm{O}(3)$-equivariant Markov kernel. Hence, we can conclude that $p_\theta : \mathcal{Z} \to \mathcal{Z}$ is a $S_N \times \mathrm{O}(3)$-equivariant Markov kernel by Proposition 1. We can also guarantee that p_θ admits a density by Proposition 2.

Training To learn θ for $p_\theta(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i})$, a natural objective is to minimise the KL divergence between the true reverse kernels and our parametrised reverse kernels

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \lambda_0(t_i)\mathcal{L}_i(\theta), \qquad \mathcal{L}_i(\theta) = \mathbb{E}_{p_{t_i}(\mathbf{x}_{t_i})}\left[D_{\mathrm{KL}}(p(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i})||p_\theta(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i}))\right],$$

where λ_0 is some time weighting function. We note that we run into the same issue as with SymDiff in that we do not have access to $p_\theta(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i})$ in closed-form, since this is expressed in terms of an expectation. However, as $\mathcal{L}_i(\theta)$ is a linear function of $-\log p_\theta(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i})$ and $-\log$ is a convex

[4]Similar to our discussion on diffusion models, we leave the time dependency implicit in here.

function, we can apply Jensen's inequality again to provide the following upper bound to our original objective

$$\mathcal{L}'(\theta) = \sum_{i=1}^{n} \lambda_0(t_i)\mathcal{L}'_i(\theta), \qquad \mathcal{L}'_i(\theta) = \mathbb{E}_{p_{t_i}(\mathbf{x}_{t_i}),\gamma_\theta(dR|\mathbf{x}_{t_i})}\left[D_{\mathrm{KL}}(p(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i})||k_\theta(\mathbf{x}_{t_{i-1}}|R,\mathbf{x}_{t_i}))\right],$$

which we can now use to train θ. To further simplify $\mathcal{L}'_i(\theta)$, we note that $k_\theta(\mathbf{x}_{t_{i-1}}|R,\mathbf{x}_{t_i}) = \mathcal{N}(\mathbf{x}_{t_{i-1}}; R\cdot\mu_\theta(R^T\cdot\mathbf{x}_{t_i}), g(t_i)^2\Delta t\mathbf{I})$ with a similar derivation as before. This allows us to evaluate the KL divergences in closed form since p, k_θ are defined in terms of Gaussians. We can show that this gives

$$\mathcal{L}'_i(\theta) = \mathbb{E}_{p_{t_i}(\mathbf{x}_{t_i}),\gamma_\theta(dR|\mathbf{x}_{t_i})}\left[\frac{1}{2}g(t_i)^2\Delta t\left\|R\cdot s_\theta(R^T\cdot\mathbf{x}_{t_i}) - \nabla_{\mathbf{x}}\log p_{t_i}(\mathbf{x}_{t_i})\right\|^2\right], \tag{25}$$

where we use the fact that $f(\cdot,t)$ is $S_N \times \mathrm{O}(3)$-equivariant. To express equation 25 in a tractable form (as we do not have access to the true score), we can apply the standard technique of employing the score matching identity (Vincent, 2011) to give

$$\mathcal{L}'_i(\theta) = \mathbb{E}_{p(\mathbf{x}_0),p(\mathbf{x}_{t_i}|\mathbf{x}_0),\gamma_\theta(dR|\mathbf{x}_{t_i})}\left[\frac{1}{2}g(t_i)^2\Delta t\left\|R\cdot s_\theta(R^T\cdot\mathbf{x}_{t_i}) - \nabla_{\mathbf{x}}\log p_{t_i}(\mathbf{x}_{t_i}|\mathbf{x}_0)\right\|^2\right] + C_i,$$

where $p(\mathbf{x}_{t_i}|\mathbf{x}_0)$ denotes the conditional distribution of $\mathbf{x}_{t_i}$ given $\mathbf{x}_0$ under the forward noising process p, and C_i is some constant. In practice, the choice of forward noising SDE in equation 23 is made to ensure that we have access to $p(\mathbf{x}_{t_i}|\mathbf{x}_0)$ in closed-form and that the distribution is easy to sample from.

By making the choice that $\lambda_0(t) = 2\lambda(t)/(g(t_i)^2T)$ for some suitable time weighting function λ, we can show as $\Delta t \to 0$ that our objective $\mathcal{L}'(\theta)$ will converge to (modulo some constant)

$$\mathbb{E}_{p(\mathbf{x}_0),t\sim U(0,T),p(\mathbf{x}_t|\mathbf{x}_0),\gamma_\theta(dR|\mathbf{x}_t)}\left[\lambda(t)\left\|R\cdot s_\theta(R^T\cdot\mathbf{x}_t) - \nabla_{\mathbf{x}}\log p(\mathbf{x}_t|\mathbf{x}_0)\right\|^2\right], \tag{26}$$

where $U(0,T)$ denotes the uniform distribution on $[0,T]$. We see that our final objective in equation 26 now resembles the standard score matching objective.

D.2 Flow Matching

Continuous normalising flows (CNFs) (Chen et al., 2018) construct a generative model of data $\mathbf{x}_1 \sim q = p_{\text{data}}$ by the pushforward of an ordinary differential equation (ODE) taking the form

$$\frac{d}{dt}\phi_t(\mathbf{x}) = u_t(\phi_t(\mathbf{x})), \qquad \phi_0(\mathbf{x}) = \mathbf{x}, \tag{27}$$

where $u_t : \mathcal{Z}\times[0,T] \to \mathcal{Z}$ is the vector field function defining the ODE, and $\phi_t : \mathcal{Z}\times[0,T] \to \mathcal{Z}$ denotes the flow implicitly defined by solutions to the above ODE. By letting p_0 be some simple prior distribution, the above ODE defines a generative model $\mathbf{x}_t \sim p_t$ by the pushforward of p_0 through the flow ϕ_t

$$p_t = [\phi_t]_{\#}p_0 \tag{28}$$

If u_t is chosen in such a way that $p_1 \approx q = p_{\text{data}}$, we can then generate samples from p_{data} by sampling some $\mathbf{x}_0 \sim p_0$, then solving the ODE in equation 27 with this initial condition[5]. Furthermore, as in previous work (Klein et al., 2024), we assume that u_t is $S_N \times \mathrm{O}(3)$-equivariant.

By considering the Euler discretisation of equation 27, we can represent the generation process by the Markov chain $p_0(\mathbf{x}_0)\prod_{i=1}^{T} p(d\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}})$ where the time-points t_i are uniformly spaced in $[0,T]$ - i.e. $t_i = i\Delta t$ where $\Delta t = T/n$ - and the transition kernels are given by the Markov kernels

$$p(d\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}}) = \delta(\mathbf{x}_{t_{i-1}} + u_{t_{i-1}}(\mathbf{x}_{t_{i-1}})\Delta t), \tag{29}$$

where $\delta(\cdot)$ denotes the Dirac measure at some point.

[5]The use of time here reverse the convention used in the diffusion literature.

Stochastic symmetrisation To learn the transition kernels induced by the vector field u_t via stochastic symmetrisation, we parametrise our transition kernels by

$$p_\theta(d\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}}) = \mathsf{sym}_{\gamma_\theta}(k_\theta)(\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}}), \tag{30}$$

where we take $k_\theta(d\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}}) = \delta(\mathbf{x}_{t_{i-1}} + v^\theta_{t_{i-1}}(\mathbf{x}_{t_{i-1}})\Delta t)$ in which $v^\theta_t : \mathcal{Z} \to \mathcal{Z}$ is some S_N-equivariant neural network which aims to learn an approximation to the true vector field u_t. We further assume $\gamma_\theta : \mathcal{Z} \to \mathrm{O}(3)$ is some S_N-invariant and $\mathrm{O}(3)$-equivariant Markov kernel. We can again conclude that $p_\theta : \mathcal{Z} \to \mathcal{Z}$ is a $S_N \times \mathrm{O}(3)$-equivariant Markov kernel by Proposition 1.

Training A natural objective to learn p_θ is to minimise the 2-Wasserstein distance $\mathcal{W}_2$ (Peyré et al., 2019) between $p(d\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}})$ and $p_\theta(d\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}})$ since these are defined in terms of Dirac measures. We can write our objective as

$$\mathcal{L}(\theta) = \sum_{i=1}^{T} \lambda_0(t_{i-1})\mathcal{L}_i(\theta), \qquad \mathcal{L}_i(\theta) = \mathbb{E}_{p_{t_{i-1}}(\mathbf{x}_{t_{i-1}})}\left[\mathcal{W}_2^2(p(d\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}}), p_\theta(d\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}}))\right]$$

where λ_0 is some time weighting function, and the 2-Wasserstein distance $\mathcal{W}_2$ is defined as $\mathcal{W}_2^2(\pi_1, \pi_2) = \inf_\pi \int \|\mathbf{x} - \mathbf{y}\|^2 \, d\pi(\mathbf{x}, \mathbf{y})$ where π is taken over the space of possible couplings between the measures π_1, π_2. We note that as $p(d\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}})$ is Dirac, there only exists a single coupling between the two kernels given by the product of the Markov kernels. This allows us to evaluate $\mathcal{L}_{i+1}$ as

$$\mathcal{L}_{i+1}(\theta) = \mathbb{E}_{p_{t_i}(\mathbf{x}_{t_i}), \gamma_\theta(dR|\mathbf{x}_{t_i})}\left[\Delta t^2 \left\| R \cdot v^\theta_{t_i}(R^T \cdot \mathbf{x}_{t_i}) - u_{t_i}(\mathbf{x}_{t_i}) \right\|^2\right]. \tag{31}$$

To express equation 31 in a tractable form (as we do not have access to u_t), we can take u_t to be constructed by the same setup used in Flow Matching (Lipman et al., 2022). This framework allows us to express equation 31 in the now tractable form

$$\mathcal{L}_{i+1}(\theta) = \mathbb{E}_{q(\mathbf{x}_1), p_{t_i}(\mathbf{x}_{t_i}|\mathbf{x}_1), \gamma_\theta(dR|\mathbf{x}_{t_i})}\left[\Delta t^2 \left\| R \cdot v^\theta_{t_i}(R \cdot \mathbf{x}_{t_i}) - u_{t_i}(\mathbf{x}_{t_i}|\mathbf{x}_1) \right\|\right] + C_{i+1}, \tag{32}$$

by the use of the Conditional Flow Matching objective, where C_{i+1} is some constant. Here $p_t(\mathbf{x}_t|\mathbf{x}_1)$ is a family of conditional distributions where $p_0(\mathbf{x}_0|\mathbf{x}_1) = p_0(\mathbf{x}_0)$ equals our prior distribution and $p_1(\mathbf{x}_1|\mathbf{x}_1) \approx \delta(\mathbf{x}_1)$, and for which $u_t(\mathbf{x}_t|\mathbf{x}_1)$ is a vector field generating $p_t(\mathbf{x}_t|\mathbf{x}_1)$ by an ODE of the form in equation 28. These are constructed to be easy to sample from and evaluate. The true vector field u_t, which provides a generative model of $q = p_{\text{data}}$, is then defined by some expectation of the conditional vector fields $u_t(\mathbf{x}_t|\mathbf{x}_0)$ over $p_t(\mathbf{x}_t|\mathbf{x}_0)$ and $q(\mathbf{x}_1)$.

Hence, by taking $\lambda_0(t) = \frac{\lambda(t)}{T\Delta t}$ for some suitable time weighting function λ, we can show as $\Delta t \to 0$, our objective $\mathcal{L}(\theta)$ will converge to (modulo some constant)

$$\mathbb{E}_{q(\mathbf{x}_1), t\sim U(0,T), p_t(\mathbf{x}_t|\mathbf{x}_1), \gamma_\theta(dR|\mathbf{x}_t)}\left[\lambda(t)\left\| R \cdot v^\theta_t(R^T \cdot \mathbf{x}_t) - u_t(\mathbf{x}_t|\mathbf{x}_1) \right\|^2\right]. \tag{33}$$

We see that our final objective in equation 33 now resembles the standard flow matching objective.