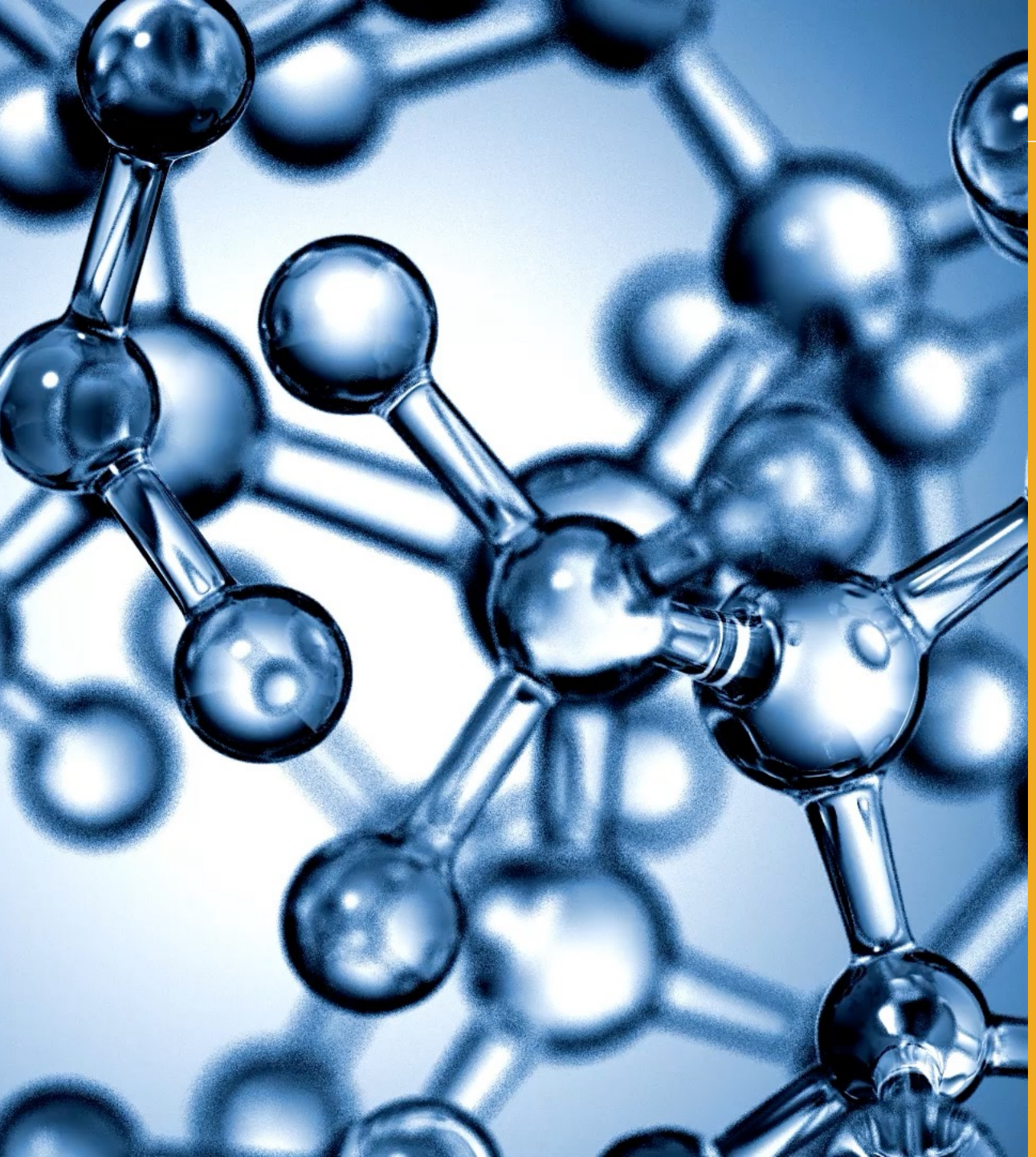
香港中文大學
The Chinese University of Hong Kong

Ron

Sampling 3D Molecular Conformers with Diffusion Transformers

11 August 2025



Contents

11 August 2025

Introduction

- Conformer generation with Diffusion model
- Less computationally expensive than DFT
- A small model in general (<100M parameters)

Results

Benchmark of this model against existing models

Precision	Recall	MAE of ensemble properties	Out-of-distribution generalization
SOTA	On par	SOTA	SOTA
The least amount of incorrect samples generated.	The most amount of samples within the training set generated.	Energy, dipole moment, HOMO-LUMO gap, minimum energy	Testing with larger molecules (>100 atoms) with models trained on molecules with ~44 atoms on average

Technical details

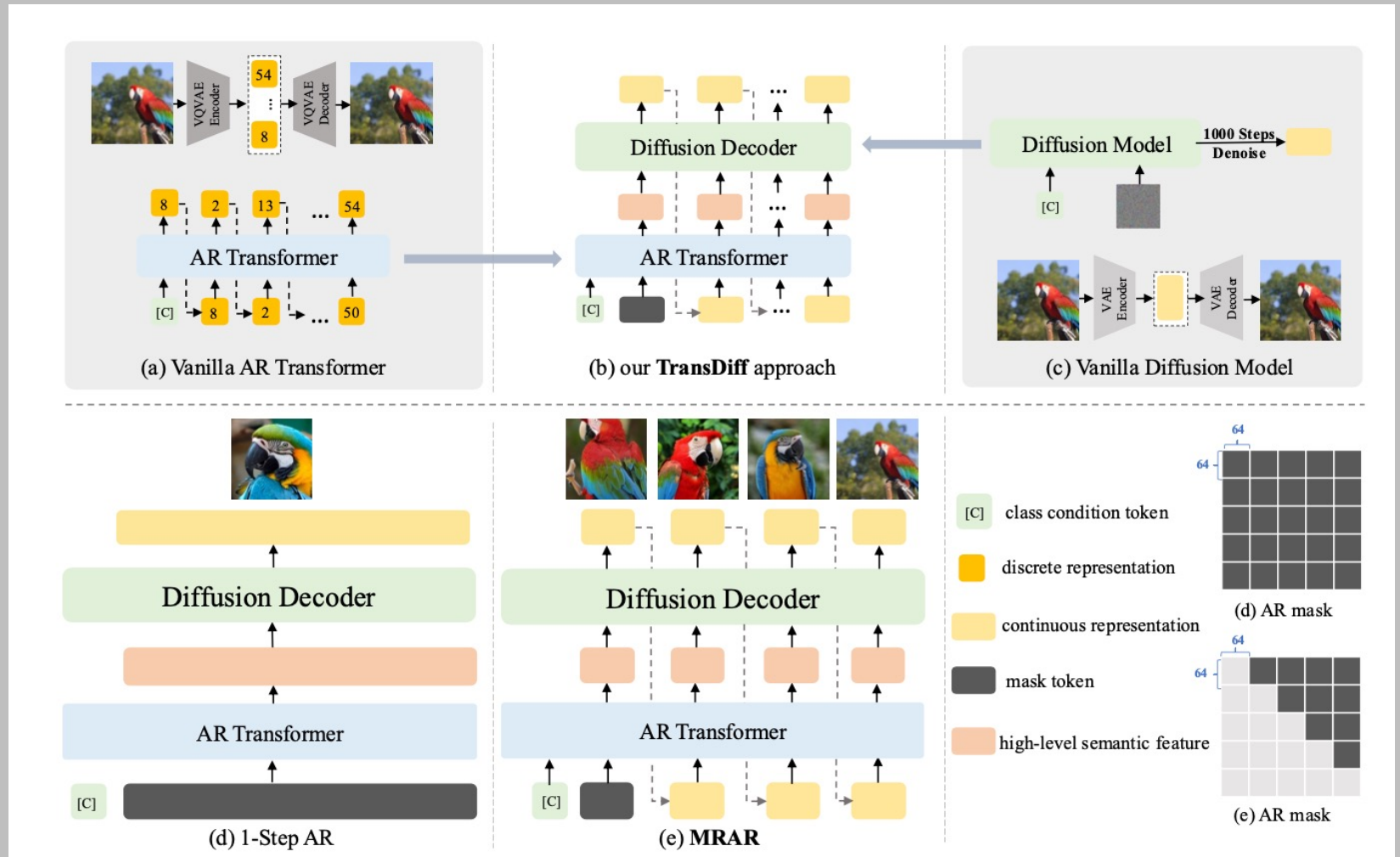
- Generative models
- Diffusion Transformers
- Flow matching
- Equivariance
- Conditioning
- Positional Embedding
- Self-attention

Generative models

A generative model is a joint probability distribution $p(x)$, for $x \in X$. In some cases, the model may be conditioned on inputs or covariates $c \in C$, which gives rise to a conditional generative model of the form $p(x|C)$.

- Probabilistic Machine Learning: Advanced Topics

Diffusion Transformers



Equivariance

That the output of a function is identical for different inputs that can be transformed into each other via certain permitted operations.

Slides from https://geometricdeeplearning.com/slides/Cambridge_1_Introduction_to_Groups_and_Representations.pdf

Group invariance

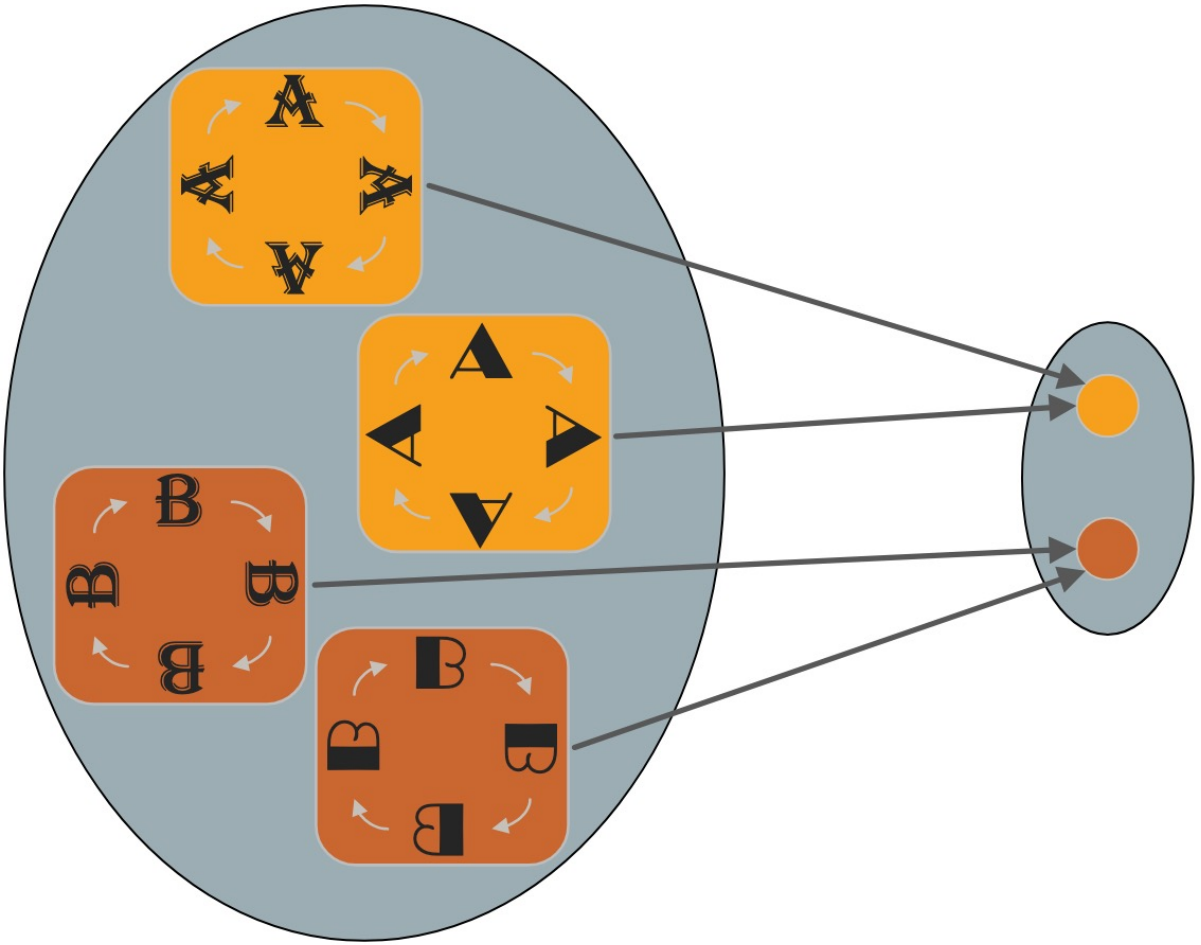
We can now *formally* describe how to *exploit* the symmetries in $\mathfrak{G}$!

A function $f : \mathcal{X}(\Omega) \to \mathcal{Y}$ is $\mathfrak{G}$-*invariant* if $f(\rho_{\mathcal{X}}(\mathfrak{g})x) = f(x)$ for all $\mathfrak{g} \in \mathfrak{G}$, i.e., its output is unaffected by the group action on the input.

e.g. **image classification**: output class won't depend on image **shifts**

$\rho_{\mathcal{X}}(\mathfrak{g})x$ $\qquad$ x

Orbits and equivalence relations



$$O_x = \{gx \mid x \in \mathcal{X}, g \in \mathfrak{G}\}$$

$\mathfrak{G}$-equivalence

$$x \sim_{\mathfrak{G}} y \Leftrightarrow \exists g \in \mathfrak{G} : gx = y$$

Satisfies the axioms of an equivalence relation:

1. Reflexivity: $x \sim_{\mathfrak{G}} x$
   - (Because $\mathfrak{G}$ contains the identity)
2. Transitivity:
   $$x \sim_{\mathfrak{G}} y \wedge y \sim_{\mathfrak{G}} z \Leftrightarrow x \sim_{\mathfrak{G}} z$$
   - (Because $\mathfrak{G}$ is closed under composition)
3. Symmetry: $x \sim_{\mathfrak{G}} y \Leftrightarrow y \sim_{\mathfrak{G}} x$
   - (Because $\mathfrak{G}$ is closed under inverses)

Group equivariance

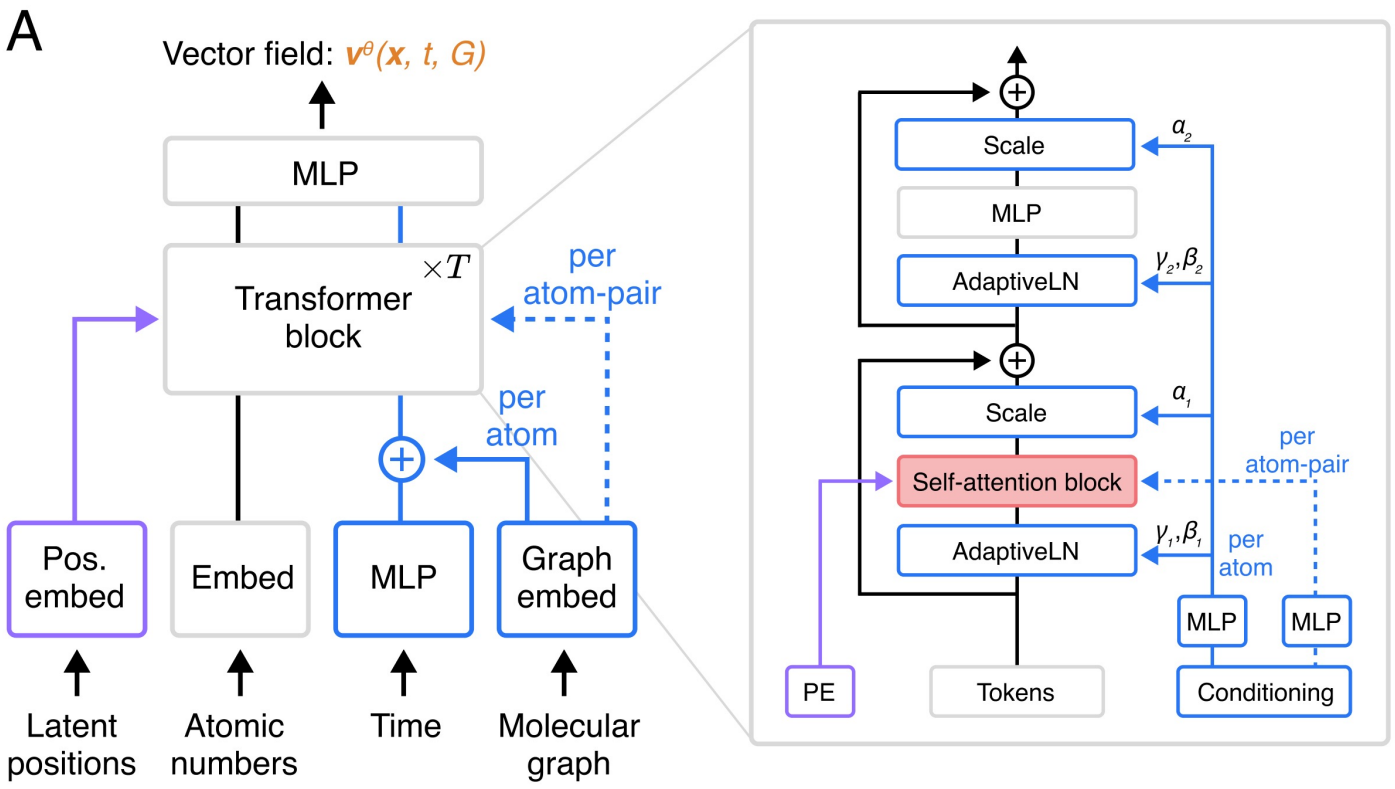
We proceed to define a more *fine-grained* notion of regularity:

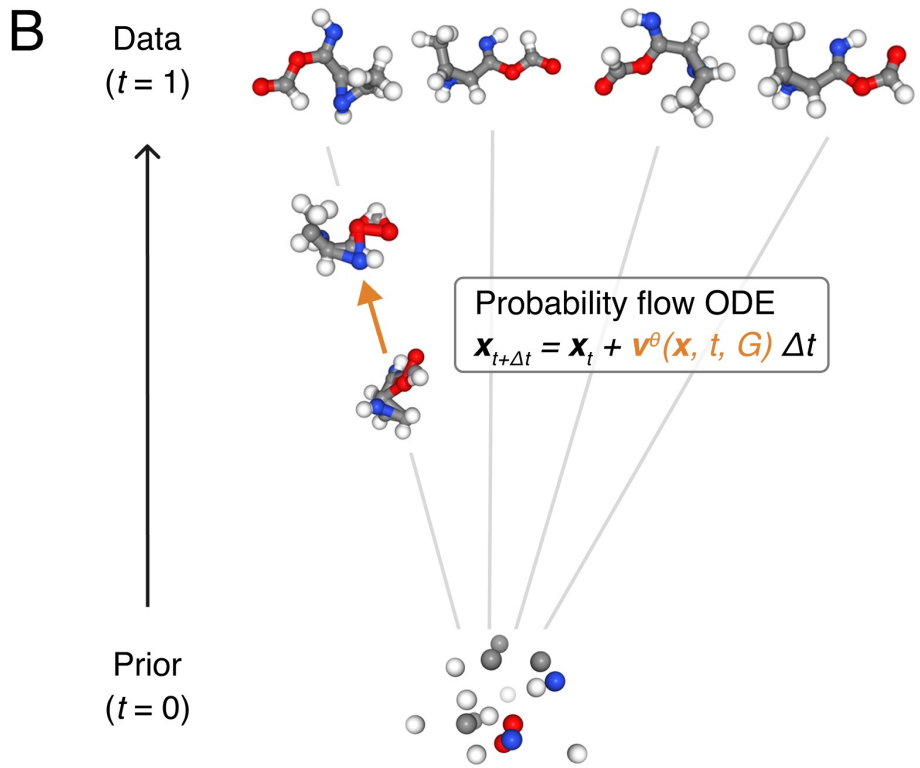
> A function $f : \mathcal{X}(\Omega) \to \mathcal{Z}(\Omega)$ is $\mathfrak{G}$-*equivariant* if, for all $\mathfrak{g} \in \mathfrak{G}$, $f(\rho_{\mathcal{X}}(\mathfrak{g})x) = \rho_{\mathcal{Z}}(\mathfrak{g})f(x)$, i.e., applying a group action on the input affects the output in the same way.

e.g. **image segmentation**: segmentation mask must **follow** any shifts in the input

Note that invariance is a *special case* of equivariance (for which $\rho_{\mathcal{Z}}$?)

Architecture (model)





Flow matching

We describe this transformation in terms of a *stochastic interpolant* $\boldsymbol{x}_t$ [26, 49, 50]. A noisy sample at time $t \in [0, 1]$ is defined as

$$\boldsymbol{x}_t = (1-t) \cdot \boldsymbol{x}_0 + t \cdot \boldsymbol{x}_1 + \sigma \cdot \boldsymbol{\epsilon}, \tag{1}$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^{N \times 3}$ is drawn from the standard normal distribution $\mathcal{N}(0, \boldsymbol{I})$ and scaled by a constant $\sigma \in \mathbb{R}_{\geq 0}$. We remark that t represents progress along this interpolation path, not physical time. Notably, stochastic interpolants enable transformations between arbitrary distributions and allow us to assess the performance of the generative process under varying prior distributions q_0. This contrasts with, e.g., score based diffusion methods [24, 51], which typically assume an isotropic Gaussian prior.

The stochastic interpolant induces a deterministic trajectory of densities $p_t(\boldsymbol{x})$, governed by an ODE known as the probability flow:

$$\mathrm{d}\boldsymbol{x} = u_t(\boldsymbol{x})\, \mathrm{d}t. \tag{2}$$

If the vector field $u_t(\boldsymbol{x})$ was tractable to sample, the weights of a neural network (NN) $\boldsymbol{v}^\theta(\boldsymbol{x}, t) : [0, 1] \times \mathbb{R}^d \mapsto \mathbb{R}^d$ could be optimized directly by minimizing

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}(0,1), \boldsymbol{x} \sim p_t(\boldsymbol{x})} \left|\left| u_t(\boldsymbol{x}) - \boldsymbol{v}^\theta(\boldsymbol{x}, t) \right|\right|. \tag{3}$$

The learned vector field $\boldsymbol{v}^\theta$ could then be used to generate new samples from the target distribution by starting from $\boldsymbol{x}_0 \sim q_0$ and integrating the probability flow ODE (Eq. 2), for example, using a numerical scheme such as Euler's method, i.e., $\boldsymbol{x}_{t+\Delta t} = \boldsymbol{x}_t + \boldsymbol{v}^\theta(\boldsymbol{x}_t, t)\Delta t$ for time step Δt.

However, for arbitrary distributions q_0 and q_1, the objective in Eq. 3 is computationally intractable [52]. Instead, we consider the expectation over interpolated point pairs from the two distributions. Eq. 1 defines a *conditional probability distribution* $p_t(\boldsymbol{x}|\boldsymbol{x}_0, \boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}|(1-t) \cdot \boldsymbol{x}_0 + t \cdot \boldsymbol{x}_1, \sigma^2)$, with *conditional vector field* $\boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_0, \boldsymbol{x}_1) = \boldsymbol{x}_1 - \boldsymbol{x}_0$ [27]. The ability to directly sample from the conditional probability via Eq. 1 allows formulating the conditional flow matching (CFM) objective

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}(0,1), \boldsymbol{x}_0 \sim q_0, \boldsymbol{x}_1 \sim q_1, \boldsymbol{x} \sim p_t(\boldsymbol{x}|\boldsymbol{x}_0, \boldsymbol{x}_1)} \left|\left| \boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_0, \boldsymbol{x}_1) - \boldsymbol{v}^\theta(\boldsymbol{x}, t) \right|\right|^2. \tag{4}$$

Conditioning

4.1 Conditioning

Each DiT block receives conditioning signals that encode information about the time step t, as well as atom- and pair-wise information derived from a molecular graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Pair-wise information is used during the self-attention update (see Eq. 5 and Sec. 4.3), whereas atom-wise information and the time conditioning signal are injected via adaptive layer norms (see Eq. 6).

Time conditioning The current time t of the latent state $\boldsymbol{x}_t$ is encoded via a two-layer MLP as

$$\boldsymbol{c}^t = \mathrm{MLP}(t)\,. \tag{7}$$

Atom-wise conditioning Atom-wise conditioning tokens are obtained from a GNN inspired by the processor module of the MeshGraphNet (MGN) framework [54]

$$\boldsymbol{c}_i^{\mathcal{G}} = \mathrm{GNN}_{\mathrm{node}}(\mathcal{V}, \mathcal{E})\,, \tag{8}$$

where $\boldsymbol{c}_i^{\mathcal{G}}$ denotes the final node representation for atom i. See Appendix I for details on the GNN.

Pair-wise conditioning In addition to node representations, the GNN also produces edge-level representations, which can be used to define pair-wise conditioning tokens:

$$\textbf{bond-pair:} \quad \boldsymbol{c}_{ij}^{\mathcal{G}} = \begin{cases} \mathrm{GNN}_{\mathrm{edge}}(\mathcal{V}, \mathcal{E}) & \forall (i,j) \in \mathcal{E} \\ \bar{\boldsymbol{c}}^{\mathcal{G}} & \forall (i,j) \notin \mathcal{E}\,. \end{cases} \tag{9}$$

These tokens only capture interactions between bonded atoms, i.e., when $(i,j) \in \mathcal{E}$. Conditioning tokens for non-bonded pairs are set to a learnable vector $\bar{\boldsymbol{c}}^{\mathcal{G}}$. Since self-attention operates on *all* atom pairs (i,j), even if they are not connected by a chemical bond, we also define an alternative pair-wise conditioning method inspired by the Graphormer architecture [55]

$$\textbf{all-pair:} \quad \boldsymbol{c}_{ij}^{\mathcal{G}} = \mathrm{MLP}\left(s(i,j)\right), \tag{10}$$

where $s(i,j)$ denotes the graph geodesic (i.e., the shortest path between atoms i and j via edges in $\mathcal{G}$). This formulation allows conditioning on all atom pairs, even if they are not directly connected. In the following, we refer to these two variants as **bond-pair** (Eq. 9) and **all-pair** (Eq. 10) conditioning, respectively (see Sec. 5 for empirical comparisons between both methods).

Positional Embedding

4.2 Positional Embeddings (PE)

We inject information about the current atomic positions $\mathcal{R} = \{\vec{r}_1, \dots \vec{r}_N \,|\, \vec{r}_i \in \mathbb{R}^3\}$ of the latent state $\boldsymbol{x}_t$ via positional embeddings (PEs). To that end, we assume the positions to have zero center of mass (see section A.1). As mentioned in Sec. 3.1, conformer identity is determined only by the *relative* positions of atoms (i.e., it is invariant under rigid translations and rotations), but ensuring that these symmetries are preserved within the network often incurs computational overhead.

Therefore, we examine a representative range of PEs that vary in the number of Euclidean symmetries they respect by construction, which affects how the latent representations transform under translations and rotations. This includes absolute and relative PEs, motivated by a similar distinction made in other application domains [56, 57]. Inspired by recent advances in geometric deep learning [58–60], we also propose a PE strategy, obeying all Euclidean symmetries. Our aim is to find an acceptable tradeoff between model accuracy, efficiency and scalability, by scanning the design space of PE strategies.

Absolute Positional Embeddings (aPE) Following Refs. [31, 37] they are calculated as

$$\boldsymbol{p}_i^{\text{aPE}} = \text{MLP}(\vec{r}_i)\,, \tag{11}$$

where $\vec{r}_i \in \mathbb{R}^3$ is the Cartesian position of the i-th atom. This kind of PE is neither rotationally nor translationally invariant, and serves as a baseline without any symmetry constraints.

Relative Positional Embedding (rPE) We use displacements vectors $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ to build pairwise relative PEs as

$$\boldsymbol{p}_{ij}^{\text{rPE}} = \text{MLP}(\vec{r}_{ij})\,. \tag{12}$$

This formulation ensures translational invariance but not rotational invariance.

Euclidean Positional Embedding (PE(3)) Adapting ideas from equivariant message passing neural networks like PaiNN [61] or NequIP [62], we construct SO(3)-equivariant pairwise PEs as a concatenation of $L+1$ components

$$\boldsymbol{p}_{ij}^{\text{PE(3)}} = \bigoplus_{\ell=0}^{L} \boldsymbol{\phi}_\ell(r_{ij}) \odot \boldsymbol{Y}_\ell(\hat{r}_{ij})\,, \tag{13}$$

where $\boldsymbol{\phi}_\ell : \mathbb{R} \mapsto \mathbb{R}^{1\times H}$ is a radial filter function, $\hat{r} = \vec{r}/r$, and $\boldsymbol{Y}_\ell \in \mathbb{R}^{(2\ell+1)\times 1}$ are spherical harmonics of degree $\ell = 0 \dots L$. The element-wise multiplication '$\odot$' between radial filters and spherical harmonics is understood to be "broadcasting" along axes with size 1, such that $(\boldsymbol{\phi}_\ell \odot \boldsymbol{Y}_\ell) \in \mathbb{R}^{(2\ell+1)\times H}$ and (after concatenation) $\boldsymbol{p}_{ij}^{\text{PE(3)}} \in \mathbb{R}^{(L+1)^2\times H}$. Under rotation of the input positions, these PEs transform equivariantly (see Appendix B.2.2). Moreover, because displacement vectors are used as inputs, the embeddings are also invariant to translations. As a result, they respect the full set of Euclidean symmetries relevant to molecular geometry.

Self-attention

4.3 Self-Attention Operation

For ease of notation, we only describe self-attention with a single head, but employ multi-head attention [29] with n_{heads} heads in our experiments. All self-attention blocks rely on query, key and value vectors, which are obtained from the input tokens $\mathcal{H} = \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \mid \boldsymbol{h}_i \in \mathbb{R}^H\}$ as

$$\boldsymbol{q} = \boldsymbol{W}_q \tilde{\boldsymbol{h}}\,, \qquad \boldsymbol{k} = \boldsymbol{W}_k \tilde{\boldsymbol{h}}\,, \qquad \boldsymbol{v} = \boldsymbol{W}_v \tilde{\boldsymbol{h}}\,, \tag{14}$$

where $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{H \times H}$ are trainable weight matrices and $\tilde{\boldsymbol{h}}$ is either identical to $\boldsymbol{h}$ or combines it with a PE (see below). We define a slightly modified similarity kernel

$$\text{sim}(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}) = \exp\left(\frac{\boldsymbol{q}^\intercal \cdot (\boldsymbol{k} \odot \boldsymbol{u})}{\sqrt{H}}\right), \tag{15}$$

where $\boldsymbol{u} \in \mathbb{R}^H$ is used to inject additional information, e.g., conditioning signals and/or positional embeddings, and '$\odot$' denotes element-wise multiplication. Depending on the subset of Euclidean symmetries we aim to incorporate, we adopt slightly different formulations of the self-attention mechanism, as detailed below.

Standard Self-Attention For absolute and relative PEs, we slightly modify standard self-attention to allow injecting pair-wise information into the values in addition to using our modified similarity kernel:

$$\mathrm{ATT}(\mathcal{H})_i = \frac{\sum_{j=1}^{N} \mathrm{sim}(\boldsymbol{q}_i, \boldsymbol{k}_j, \boldsymbol{u}_{ij}) \cdot (\boldsymbol{v}_j \odot \boldsymbol{u}_{ij})}{\sum_{j=1}^{N} \mathrm{sim}(\boldsymbol{q}_i, \boldsymbol{k}_j, \boldsymbol{u}_{ij})} \,. \tag{16}$$

Queries, keys, and values are obtained with Eq. 14 from different (position-encoded) tokens $\tilde{\boldsymbol{h}}_i$ depending on the chosen PEs; further, the injected pair-wise information $\boldsymbol{u}_{ij}$ differs:

$$\tilde{\boldsymbol{h}}_i = \begin{cases} \boldsymbol{h}_i + \boldsymbol{p}_i^{\mathrm{aPE}} & \text{for absolute PEs}\,, \\ \boldsymbol{h}_i & \text{for relative PEs}\,, \end{cases} \quad \text{and} \quad \boldsymbol{u}_{ij} = \begin{cases} \boldsymbol{c}_{ij}^{\mathcal{G}} & \text{for absolute PEs}\,, \\ \boldsymbol{c}_{ij}^{\mathcal{G}} + \boldsymbol{p}_{ij}^{\mathrm{rPE}} & \text{for relative PEs}\,. \end{cases} \tag{17}$$

Here $\boldsymbol{c}_{ij}^{\mathcal{G}} \in \mathbb{R}^H$ are pair-wise graph conditioning tokens (see Eqs. 9 and 10) and $\boldsymbol{p}_i^{\mathrm{aPE}} \in \mathbb{R}^H$ and $\boldsymbol{p}_{ij}^{\mathrm{rPE}} \in \mathbb{R}^H$ are the absolute and relative PEs described above (see Eqs. 11 and 12).

SO(3)-Equivariant Self-Attention To preserve all Euclidean symmetries throughout the network, every token must transform equivariantly. One way to achieve this is by separating out the rotational degrees of freedom, encoding them with irreducible representations of the rotation group $\mathrm{SO}(3)$. This introduces a "degree-axis" of size $(L+1)^2$, which encodes angular components of increasing order. The maximum degree L is chosen to ensure high fidelity at a reasonable computational cost. For example, setting $L = 1$ restricts the representation to scalars and vectors, as used in models like PaiNN [61] or TorchMDNet [34]. An $\mathrm{SO}(3)$-equivariant formulation of self-attention is then given as

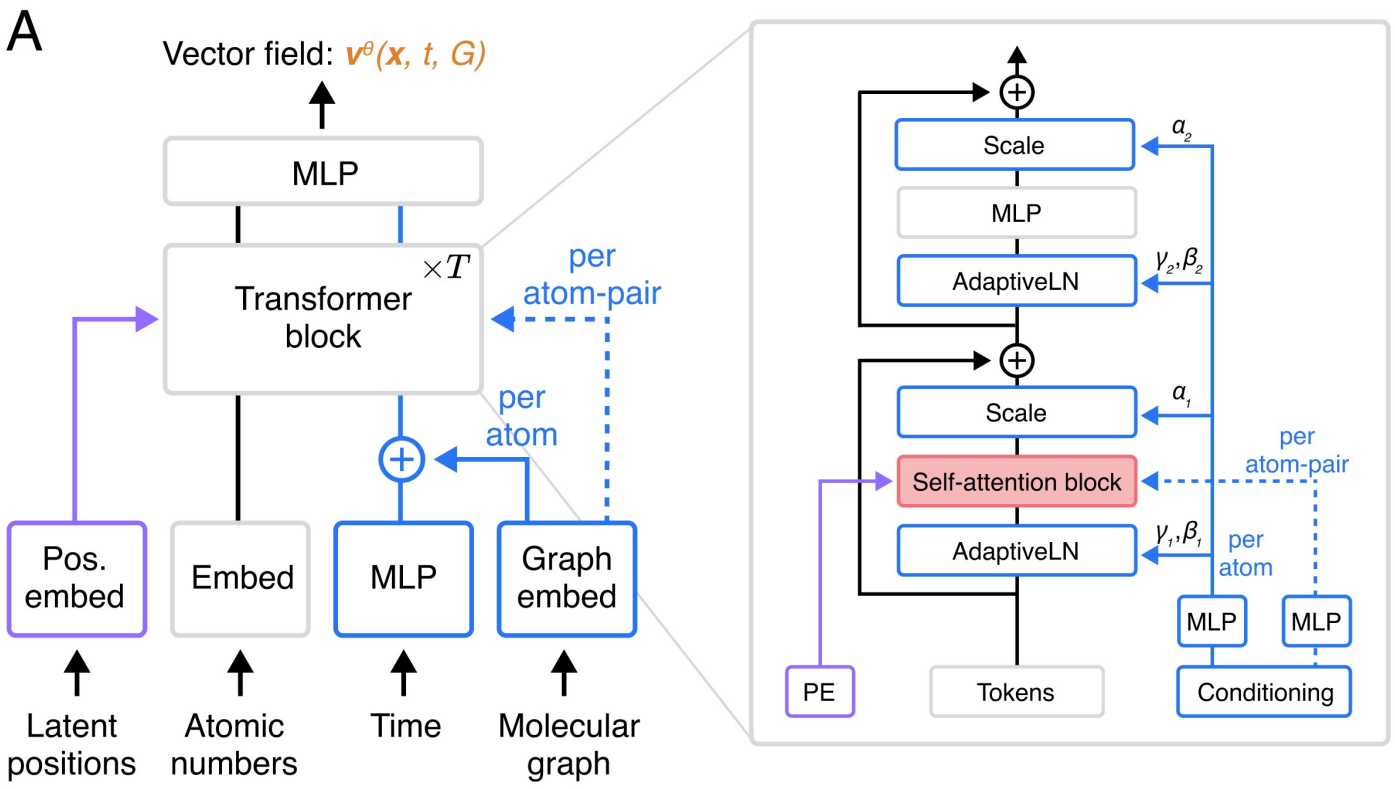
$$\mathrm{ATT}_{\mathrm{SO}(3)}(\mathcal{H})_i = \frac{\sum_{j=1}^{N} \mathrm{sim}(\boldsymbol{q}_i, \boldsymbol{k}_j, \boldsymbol{u}_{ij}) \cdot (\hat{\boldsymbol{u}}_{ij} \otimes \boldsymbol{v}_j)}{\sum_{j=1}^{N} \mathrm{sim}(\boldsymbol{q}_i, \boldsymbol{k}_j, \boldsymbol{u}_{ij})} \,, \tag{18}$$

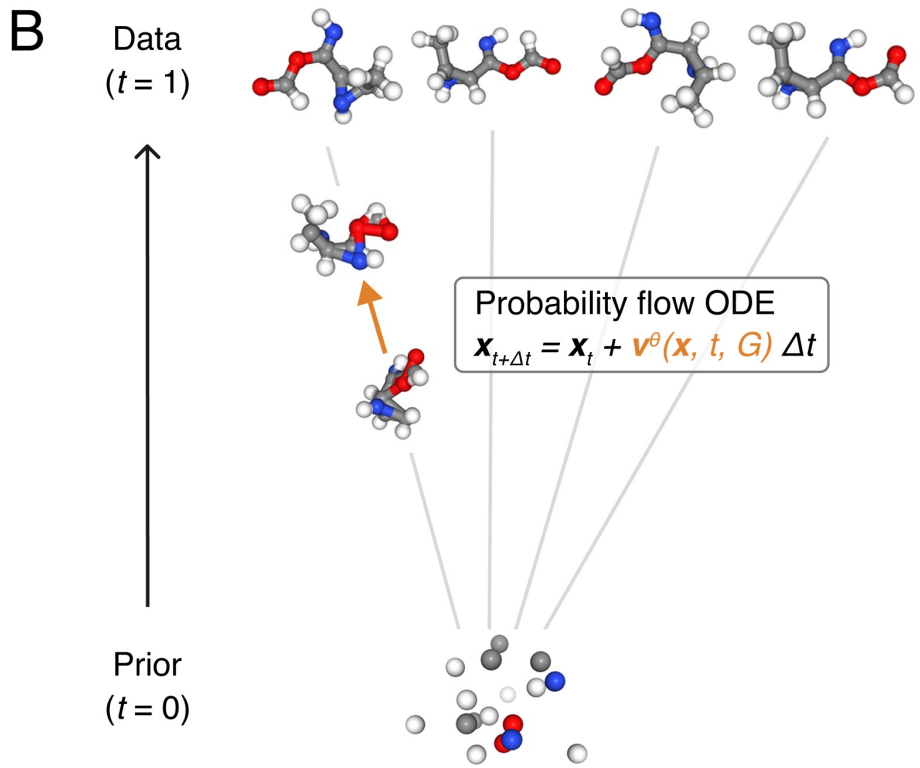
where equivariant queries, keys and values can be calculated similarly to Eq. 14 and '$\otimes$' denotes a Clebsch-Gordan (CG) tensor product contraction [60]. The dot-product in the similarity measure is taken along both feature and degree axes, such that the overall update preserves equivariance (see Appendix for details). Tokens and scaling vectors are calculated as

$$\tilde{\boldsymbol{h}}_i = \boldsymbol{h}_i \,, \qquad \boldsymbol{u}_{ij} = \boldsymbol{\phi}(r_{ij}) \odot \boldsymbol{c}_{ij}^{\mathcal{G}} \,, \qquad \hat{\boldsymbol{u}}_{ij} = \boldsymbol{p}_{ij}^{\mathrm{PE(3)}} \odot \boldsymbol{c}_{ij}^{\mathcal{G}} \,, \tag{19}$$

where $\boldsymbol{\phi}(r_{ij}) \in \mathbb{R}^{(L+1)^2 \times H}$ is a radial filter, and the element-wise products with the pair-wise conditioning tokens $\boldsymbol{c}_{ij}^{\mathcal{G}} \in \mathbb{R}^{1 \times H}$ are broadcast along the degree axis. Importantly, the $2\ell + 1$ subcomponents of the radial filter for degree ℓ are obtained by repeating per-degree filter functions $\boldsymbol{\phi}_\ell(r_{ij}) \in \mathbb{R}^{1 \times H}$ along the degree axis to preserve equivariance (see also Eq. 13).

Architecture (model)





Thank You