Symmetry-Preserving Conformer Ensemble Networks for Molecular Representation Learning

Yanqiao Zhu* **Yidan Shi*** **Yuanzhou Chen** **Fang Sun** **Yizhou Sun** **Wei Wang**
Department of Computer Science, University of California, Los Angeles
{yzhu,adrianchen,fts,yzsun,weiwang}@cs.ucla.edu yidanshi@ucla.edu

Abstract

Molecular representation learning has emerged as a promising approach for modeling molecules with deep learning in chemistry and beyond. While 3D geometric models effectively capture molecular structure, they typically process single static conformers, overlooking the inherent flexibility and dynamics of molecules. In reality, many molecular properties depend on distributions of thermodynamically accessible conformations rather than single structures. Recent works show that learning from conformer ensembles can improve molecular representations, but existing approaches either produce unphysical structures through averaging or require restrictive molecular alignment. In this paper, we propose Symmetry-Preserving Conformer Ensemble networks (SPiCE), which introduces two key innovations: (1) geometric mixture-of-experts for selective processing of scalar and vector features, and (2) hierarchical ensemble encoding that combines ensemble-level representation with cross-conformer integration. Crucially, SPiCE ensures physically meaningful representations by maintaining joint equivariance to geometric transformations of individual conformers and conformer permutations. Extensive experiments demonstrate that SPiCE consistently outperforms existing conformer ensemble methods and state-of-the-art structural aggregation models across quantum mechanical and biological property prediction tasks.

1 Introduction

Molecular Representation Learning (MRL) has emerged as a powerful tool at the intersection of chemistry and machine learning, offering a data-driven approach to encode discrete molecular structures into continuous feature representations [1, 2]. This enables efficient predictions of molecular properties without expensive quantum mechanical calculations for various downstream tasks including drug discovery, materials design, and chemical reaction prediction [3–5].

The field has evolved from early appraoches using simplified molecular representations such as SMILES strings [6] and molecular fingerprints [7, 8] to sophisticated geometric deep learning methods that directly incorporate 2D topological and 3D geometric information [9]. Among these advances, 3D Graph Neural Networks (GNNs) have gained popularity by learning from molecular geometries while respecting fundamental physical symmetries. These models can be categorized as invariant (producing identical outputs regardless of molecular orientation and translation) [10–12], $\mathrm{SE}(3)$–equivariant (transforming consistently under rotations and translations) [13, 14], and $\mathrm{E}(3)$–equivariant (additionally respecting reflection symmetry) [15, 16]. By incorporating these inductive biases, geometric models achieve both improved sample efficiency and better generalization.

However, current 3D MRL models face a fundamental limitation: they typically encode individual conformer structures as if molecules were rigid, static entities. In reality, molecules exist as dynamic

*These authors made equal contribution to this research.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

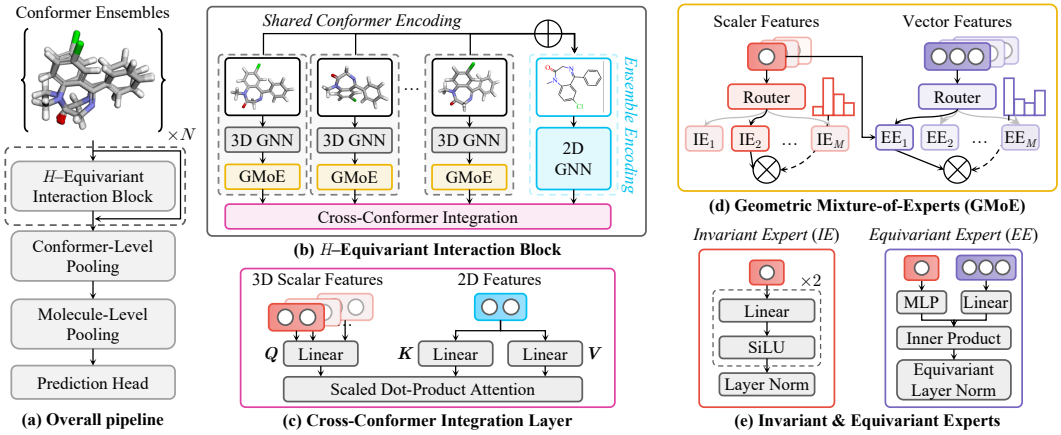

Figure 1: (a) SPiCE processes conformer ensembles through multiple interaction blocks, followed by node-level pooling to obtain conformer representations and ensemble-level pooling for final molecular embeddings. (b) H-equivariant interaction block that maintains joint equivariance to conformer permutation (S_n) and geometric transformations (G^n), composed of shared conformer encoding, geometric mixture-of-experts, and cross-conformer integration. (c) Cross-conformer integration layer combining ensemble encoding through a 2D GNN with attention-based integration. (d) Geometric mixture-of-experts (GMoE) with separate routing for scalar (type-0) and vector (type-1) features. (e) Invariant and equivariant expert networks: equivariant experts preserve rotational symmetry through specialized linear transformations with scalar feature gating for enhanced expressivity.

systems that continuously interconvert between different conformational states through bond rotations, vibrational motions, and intermolecular interactions [17]. Therefore, a conformer ensemble, which refers to the collection of thermodynamically accessible molecular conformations at equilibrium, provides a more complete molecular representation [18]. This is particularly important because many experimentally observable properties depend on the entire distribution of conformers rather than a single static structure. For example, protein-ligand binding often involves conformational selection where proteins recognize specific ligand conformations, and reaction mechanisms frequently depend on the accessibility of particular geometric configurations [19, 20].

Existing approaches to conformer ensemble modeling fall into three categories. Traditional cheminformatics methods like 4D-QSAR [21–23] extend classical mesh-based 3D-QSAR by incorporating conformational information into grid-based molecular descriptors. These methods map molecular properties onto regular 3D lattices for each conformer to produce molecular shape spectra, but require rigid molecular alignment and are limited to datasets with common substructures. Multi-instance learning methods have been adapted from computer vision to treat conformer ensembles as bags of instances [24–28]. However, these approaches typically process conformers independently and then aggregate conformer representations through simple pooling operations, which discard cross-conformer interactions and cannot maintain geometric symmetrices. Additionally, structural aggregation methods [29–31] attempt to combine information from multiple conformers through averaging or clustering procedures. While these can leverage geometric models, they often produce unphysical structures and are highly sensitive to alignment methods.

To address these limitations, we present Symmetry-Preservng Conformer Ensemble networks (SPiCE), a novel model that achieves joint equivariance to both permutations of conformer ensemble (S_n) and geometric transformations of individual conformers (G^n). SPiCE processes conformer ensembles through a series of H–equivariant interaction blocks, where $H = S_n \times G^n$. Each block is composed of three key components: (1) shared conformer encoding with weight-tied 3D GNNs that process each conformer while preserving geometric equivariance, (2) a Geometric Mixture-of-Experts (GMoE) layer that separately routes scalar and vector features through specialized expert networks, enabling type-aware selective processing, and (3) hierarchical ensemble encoding that combines molecular-level context with selective cross-conformer integration through attention mechanisms. Importantly, SPiCE is able to capture both geometric and topological relationships while respecting fundamental geometric symmetries, effectively processing conformer ensembles without requiring alignment or generating unphysical intermediate structures.

We validate SPiCE through comprehensive experiments across diverse molecular property prediction tasks, ranging from quantum mechanical properties to biological activities. Using various 3D GNN backbones and datasets of different scales, we demonstrate consistent improvements over existing conformer ensemble methods. Our analysis reveals scaling behaviors with dataset size, suggesting reliable deployment across different data regimes in real-world applications. Through extensive ablation studies, we justify key architectural choices including the GMoE design, sparse upcycling strategy, and optimal expert granularity for conformer ensemble modeling.

2 Preliminaries

2.1 Problem Definition

A molecule is represented as a molecular graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, where $\mathcal{V} = \{v_i\}_{i=1}^{|\mathcal{V}|}$ denotes the node set of atoms, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents chemical bonds between atoms. The node attributes are represented in $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times d_v}$. For a given molecule, we consider a set of n discrete conformers $\mathcal{C} = \{\boldsymbol{C}_i\}_{i=1}^{n}$, where each $\boldsymbol{C}_i \in \mathbb{R}^{|\mathcal{V}| \times 3}$ represents atomic 3D coordinates. These conformers are sampled from the thermodynamically-accessible conformational space. Each conformer is associated with a Boltzmann weight $p_i = \exp(-e_i/k_B T) \big/ \sum_j \exp(-e_j/k_B T)$, where e_i is the energy of conformer $\boldsymbol{C}_i$, k_B is the Boltzmann constant, and T is temperature. We note that these Boltzmann weights are *not* provided to the model but rather are used to compute ground-truth ensemble properties.

Our goal is to learn a mapping from the conformer ensemble $(\mathcal{G}, \mathcal{C})$ to molecular properties while preserving both *permutation* invariance of the ensemble and *geometric* symmetries of individual conformers. This requires learning functions that are equivariant to the product group $H = S_n \times G^n$, where S_n is the permutation group over n conformers and G is the geometric symmetry group. Typically, G is taken as the Euclidean group $\mathrm{E}(3)$ of translations, rotations and reflections, or the special Euclidean group $\mathrm{SE}(3)$ of translations and rotations. We allow individual transformations on each conformer individually and independently, hence the overall geometric transformation G^n.

2.2 Basics of Symmetry Properties

On permutation group S_n. Our work builds on the characterization of linear S_n–equivariant layers [32], which can be decomposed into intra-conformer interactions and a global G–invariant aggregation. We formalize this as a theorem and prove it in Appendix C.1. This provides a practical construction of H–equivariant layers, by processing individual conformers through G–equivariant functions and capturing ensemble interactions via a G–invariant function applied to aggregated features. This generalizes Deep Sets [33] to geometric symmetries in molecular conformers.

On geometry group G. We decompose geometric transformations into translations and rotations (including reflections for $E(3)$): $G = T \rtimes R$. Following Equiformer [34], we use *type-0* and *type-1* to express R–invariant and equivariant features and list basic symmetry-preserving operations below.

Lemma 1. *The following operations preserve symmetry:*

(1) Any function $f(s)$ of an R–invariant feature s is R–invariant;
(2) The product $s \cdot \boldsymbol{v}$ between R–invariant feature s and R–equivariant feature $\boldsymbol{v}$ is R–equivariant.
(3) The inner product $\langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle$ between two R–equivariant features $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ is R–invariant.

3 Method

The overall architecture of SPiCE is illustrated in Figure 1. SPiCE employs a hierarchical architecture that enables structural interaction across atoms and conformers. Its core is a series of H–equivariant interaction blocks designed to process conformer ensembles while maintaining the relevant symmetries. Intra-conformer learning is performed using weight-shared equivariant 3D GNNs, which jointly handle scalar and vector features. To support type-specific and selective information processing, we design a Geometric Mixture-of-Experts (GMoE) module. The resulting features are then pooled to form an ensemble representation with 2D GNNs that capture the topological structure of the molecular system. These molecule- and conformer-level features are subsequently combined through a cross-conformer integration module. Following the interaction blocks, we apply node-level pooling within each conformer to generate conformer-level embeddings and conformer-level pooling

across the ensemble to obtain the final molecular embedding. This embedding is then passed through a prediction head to produce the output property prediction. A PyTorch-like pseudocode of SPiCE is provided in Appendix B.

Overall H–equivariance. We specify the input and output invariance and equivariance types for every module in the following sections. Since each module preserves symmetry and the output symmetrical properties of each module match the input properties of each subsequent module, the overall symmetry with respect to H is guaranteed through the composition of all modules.

Other remarks. We batch over both conformers and molecules, hence our features shapes often start with $\mathbb{R}^{n\times|\mathcal{V}|\times\cdots}$. We denote $N = n \times |\mathcal{V}|$ as the total number of atoms across all conformers and interchangeably use $\mathbb{R}^{N\times\cdots}$ to express dimensionality. Typically, the second-to-last dimension of our features (`dim=-2`) is either 1 for R–invariant scalar features or 3 for R–equivariant vector features. We often concatenate these features into a single **hybrid** feature with size 4 over this dimension, and call a hybrid feature R–equivariant when the corresponding type-0 feature is R–invariant and type-1 feature is R–equivariant. Many modules of our model perform on each conformer *separately*, in which case S_n–equivariance automatically holds, and individual G–symmetries lead to global G^n–symmetry.

3.1 Input Representation and Feature Construction

For a conformer ensemble, each H–equivariant interaction block operates on graph node features that combine atomic and geometric information. For a molecule with m atoms, the input consists of: (a) atomic numbers $\boldsymbol{z} \in \mathbb{Z}_+^{|\mathcal{V}|}$, which are embedded into initial node features $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$ through a learnable embedding layer, and (b) atomic coordinates $\{\boldsymbol{c}_i\}_{i=1}^m$ where $\boldsymbol{c}_i \in \mathbb{R}^3$. From the coordinates, we compute relative position vectors $\boldsymbol{\rho}_{ij} = \boldsymbol{c}_i - \boldsymbol{c}_j$.

The H–equivariant interaction blocks process both features $\boldsymbol{x}_i$ and geometric vectors $\boldsymbol{\rho}_{ij}$ to construct messages that respect the underlying symmetries. To encode distance information effectively, we transform the relative distances $r = |\boldsymbol{\rho}_{ij}|$ using a set of radial basis functions. Specifically, we employ Gaussian radial basis functions (RBF) $\phi_k(r) = \exp(-\gamma(r-\mu_k)^2)$, where $\{\mu_k\}_{k=1}^K$ are equally spaced centers and γ controls the width of the Gaussians [10]. These radial features are then processed through a three-layer neural network, with layer normalization [35] and SiLU activation [36] after the first two linear layers. The output of this network parametrizes the message construction in the interaction blocks. Thanks to the RBF positional encoding, our encoded features are all **T–invariant**, therefore the T^n–invariance of the entire model is already guaranteed.

3.2 H–Equivariant Interaction Blocks

As the model is equipped with L interaction blocks, here we analyze the architecture within the l-th interaction block, and mostly omit the layer index l in the following discussions for simplicity.

3.2.1 Shared Conformer Encoding

With a conformer set with hybrid node feature as $\mathbf{X} = (\mathbf{X}^{\mathrm{s}}, \mathbf{X}^{\mathrm{v}}) \in \mathbb{R}^{n\times|\mathcal{V}|\times 4\times d}$, we employ shared 3D GNN layers to process conformer geometric information. A 3D GNN layer consists of two key learnable functions: message construction Φ and feature update Ψ, which can be formulated as

$$\boldsymbol{h}_i = \Psi\left(\boldsymbol{x}_i, \bigoplus_{j\in\mathcal{N}_i} \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{c}_{ij})\right), \tag{1}$$

where $\boldsymbol{h}_i = (\boldsymbol{h}_i^{\mathrm{s}}, \boldsymbol{h}_i^{\mathrm{v}})$, $\boldsymbol{h}_i^{\mathrm{s}} \in \mathbb{R}^{1\times d}$ represents the scaler (type-0, rotationally invariant) feature for atom i with 1 tensor component and d channels, and $\boldsymbol{h}_i^{\mathrm{v}} \in \mathbb{R}^{3\times d}$ represents the vector (type-1, rotationally equivariant) feature for atom i with 3 tensor components and d channels. For atom i in an arbitrary conformer, $\mathcal{N}_i$ denotes the set of its neighbors, $\boldsymbol{x}_i = (\boldsymbol{x}_i^{\mathrm{s}}, \boldsymbol{x}_i^{\mathrm{v}}) \in \mathbb{R}^{4\times d}$ represent the concatenated scalar and vector features, $\boldsymbol{c}_{ij} = (\boldsymbol{c}_{ij}^{\mathrm{s}}, \boldsymbol{c}_{ij}^{\mathrm{v}}) \in \mathbb{R}^{4\times d_m}$ are concatenated scalar and vector messages between atom i and j, d, d_m are feature dimensions, and $\bigoplus$ denotes message aggregation over neighbors $\mathcal{N}_i$ (typically summation).

For the first interaction block ($l = 0$), $\mathbf{X}^{\text{s}}$ (type-0) contains initial molecular features including embeddings of atomic numbers $\mathbf{z}$, distance-based radial basis functions, and topological descriptors, while $\mathbf{X}^{\text{v}}$ (type-1) contains initial geometric features including embeddings of relative position vectors and directional bond features. For subsequent interaction blocks ($l > 0$), the features include both the initial features above plus the processed outputs $\widetilde{\mathbf{H}}^{\text{s}}$ and $\widetilde{\mathbf{H}}^{\text{v}}$ from the previous interaction block.

The 3D GNN interaction layer outputs hybrid features $\mathbf{H} = (\mathbf{H}^{\text{s}}, \mathbf{H}^{\text{v}}) \in \mathbb{R}^{n\times|\mathcal{V}|\times 4\times d}$, where $\boldsymbol{h}_i$ denotes the hybrid node embedding for the i-th atomic node within this tensor. We also process $\mathbf{H}$ with a residual connection and an Equivariant Layer Normalization (ELN), which is a generalized layer normalization for G–equivariant features [34]. Here each conformer is operated separately, therefore the output $\mathbf{H}$ is S_n**–equivariant**; plus, all 3D GNNs are pre-selected to be equivariant, so it is also independently R**–equivariant** for each conformer.

We note that our framework is agnostic to the specific architecture of the message and update functions. They can be either Cartesian equivariant models (e.g., PaiNN [37]) that operate directly in Cartesian coordinates to construct messages and update features, or spherical equivariant models that employ spherical harmonics and tensor products for message passing (e.g., Equiformer [34]), with channel-mixing and gated non-linear functions for feature updates.

3.2.2 Geometric Mixture-of-Experts (GMoE)

When processing molecular conformers, certain atoms or structural features may be more relevant or important than others for specific properties. Graph pooling approaches [38–40] achieve this selective processing by focusing computation on important nodes through attention mechanisms. This naturally extends to Mixture-of-Experts (MoE) architectures [41, 42], which specialize multiple neural networks for selective information passing with a learnable routing mechanism.

Since scalar and vector features require distinct handling due to their different transformation properties under symmetry operations, we propose GMoE as shown in Figure 1(d) that maintains equivariance for vector features while ensuring invariance for scalar features through distinct router and expert groups. Note that all operations in this section operate on each node (and hence each conformer) separately, therefore S_n**–equivariance** is already guaranteed, and any claim on rotational R– invariance / equivariance automatically extends to R^n.

GMoE processes the two types of features in $\mathbf{H} = (\mathbf{H}^{\text{s}}, \mathbf{H}^{\text{v}}) \in \mathbb{R}^{N\times 4\times d}$ using separate modules. For a given node i, let $\boldsymbol{h}_i^{\text{s}} \in \mathbb{R}^{1\times d}$ and $\boldsymbol{h}_i^{\text{v}} \in \mathbb{R}^{3\times d}$ denote its scalar and vector features, respectively. We extend the standard Mixture of Experts (MoE) formulation [43, 44] to handle scalar and vector features independently, introducing two sets of routers and experts that process invariant and equivariant features respectively. The numbers of invariant and equivariant experts are denoted by N_{I} and N_{E}.

Permutation-equivariant router design. For each node i, the concatenated routing weights $\hat{\boldsymbol{r}}_i = (\hat{\boldsymbol{r}}_i^{\text{s}}, \hat{\boldsymbol{r}}_i^{\text{v}}) \in \mathbb{R}^{N_{\text{I}}+N_{\text{E}}}$ for scalar and vector branches are computed through a two-stage process. First, the corresponding routing networks $r^{\text{s}}(\boldsymbol{h}_i^{\text{s}})$ and $r^{\text{v}}(\boldsymbol{h}_i^{\text{v}})$, both with R**–invariant** outputs, compute initial scores $\boldsymbol{r}_i = (\boldsymbol{r}_i^{\text{s}}, \boldsymbol{r}_i^{\text{v}}) \in \mathbb{R}^{N_{\text{I}}+N_{\text{E}}}$ of $(N_{\text{I}}, N_{\text{E}})$ experts for scalar and vector features. Following SAGPool [38], our scalar router leverages a GCN layer [45] to compute routing scores, and our vector router calculates a linearly transformed inner product of features:

$$r^{\text{s}}(\boldsymbol{h}_i^{\text{s}}) = \text{softmax}\left(\sum_{j\in\mathcal{N}(i)} \boldsymbol{h}_j^{\text{s}}\boldsymbol{W}^{\text{s}}/\sqrt{d_i d_j} + \boldsymbol{b}\right), \tag{2}$$

$$r^{\text{v}}(\boldsymbol{h}_i^{\text{v}}) = \text{softmax}\left(\boldsymbol{h}_i^{\text{v}\mathsf{T}}\boldsymbol{h}_i^{\text{v}}\boldsymbol{W}^{\text{v}}\right). \tag{3}$$

where d_i represents the number of neighbors of node i. The R**-invariance** of r^{v} is due to (3) of Lemma 1: operation $\boldsymbol{h}_i^{\text{v}\mathsf{T}}\boldsymbol{h}_i^{\text{v}}$ performs inner products over the equivariant dimension. The scores then undergo top-k selection and normalization:

$$r'^{\text{s}}_{e,i} = \begin{cases} r^{\text{s}}_{e,i}, & \text{if } r^{\text{s}}_{e,i} \in \text{top-}k\big(\{r^{\text{s}}_{e,i}\}_{e=1}^{N_{\text{I}}}\big), \\ 0, & \text{otherwise,} \end{cases} \qquad \hat{r}^{\text{s}}_{e,i} = \frac{r'^{\text{s}}_{e,i}}{\sum_{e=1}^{k} r'^{\text{s}}_{e,i}}, \tag{4}$$

where $r^{\text{s}}_{e,i}$ refers to the score for the i-th atom of the e-th expert, and k is the pre-set number of selected experts for each type of hybrid features. The computation for $\hat{r}^{\text{v}}_{e,i}$ shares the same logic as $\hat{r}^{\text{s}}_{e,i}$. The resulting $\{\hat{\boldsymbol{r}}_i = (\hat{\boldsymbol{r}}_i^{\text{s}}, \hat{\boldsymbol{r}}_i^{\text{v}})\}_{i\in\mathcal{C}\times\mathcal{V}}$ is R^n**–invariant** from (1) of Lemma 1.

Invariant and equivariant experts. Each Invariant Expert (IE) is implemented as a two-layer MLP with SiLU activations followed by layer normalization. The output of the e-th IE can be defined as $\mathbf{F}_e^{\mathrm{s}} = f_e^{\mathrm{s}}(\mathbf{H}^{\mathrm{s}}) \in \mathbb{R}^{N\times 1\times d}$. These outputs are also R^n**–invariant** from (1) of Lemma 1. Meanwhile, for maintaining geometric symmetries, each Equivariant Expert (EE) is formulated as:

$$\boldsymbol{f}_{e,i}^{\mathrm{v}} = \boldsymbol{h}_i^{\mathrm{v}} \boldsymbol{W}_e \,\mathrm{diag}(\mathrm{MLP}(\boldsymbol{h}_i^{\mathrm{s}})), \tag{5}$$

where $\boldsymbol{f}_{e,i}^{\mathrm{v}}$ is the output of the e-th expert on node i, and the concatenated output $\mathbf{F}_e^{\mathrm{v}} = f_e^{\mathrm{v}}(\mathbf{H}^{\mathrm{v}}, \mathbf{H}^{\mathrm{s}}) \in \mathbb{R}^{N\times 4\times d}$. Here $\boldsymbol{W}_e \in \mathbb{R}^{d\times d}$ is a learnable weighting matrix, and diag constructs a diagonal matrix from a vector. EE is R^n**–equivariant** from (2) of Lemma 1 since $\boldsymbol{h}_i^{\mathrm{v}}$ is equivariant, while both $\boldsymbol{W}_e$ and $\mathrm{diag}(\mathrm{MLP}(\boldsymbol{h}_i^{\mathrm{s}}))$ are invariant, the latter being a consequence of (1) of Lemma 1.

Node-wise weighted summarization. With selected expert weights and expert output for both scalar and vector features, a node-wise weighted summarization is employed for dynamic feature ensemble. For node $i \subseteq \mathcal{C} \times \mathcal{V}$, the scalar and vector representations can be calculated as:

$$\widetilde{\boldsymbol{h}}_i^{\mathrm{s}} = \sum_e \hat{r}_{e,i}^{\mathrm{s}} \boldsymbol{f}_{e,i}^{\mathrm{s}} : \mathbb{R}^{1\times d}, \qquad \widetilde{\boldsymbol{h}}_i^{\mathrm{v}} = \sum_e \hat{r}_{e,i}^{\mathrm{v}} \boldsymbol{f}_{e,i}^{\mathrm{v}} : \mathbb{R}^{3\times d}. \tag{6}$$

For a molecular conformer ensemble, the output of MoE module is represented as $\widetilde{\mathbf{H}} = (\widetilde{\mathbf{H}}^{\mathrm{s}}, \widetilde{\mathbf{H}}^{\mathrm{v}}) \in \mathbb{R}^{N\times 4\times d}$, where the $\widetilde{\mathbf{H}}^{\mathrm{s}}$ and $\widetilde{\mathbf{H}}^{\mathrm{v}}$ are R^n–invariant and R^n–equivariant, respectively from (1) and (2) of Lemma 1. Hence, the hybrid feature $\widetilde{\mathbf{H}}$ is R^n**–equivariant**.

Router training and regularization. To ensure balanced expert utilization, we employ three key mechanisms: (1) Router z-loss to punish extreme routing decisions [46]:

$$\mathcal{L}_R(\boldsymbol{r}) = \frac{1}{N}\sum_{i=1}^{N}\left(\log\sum_{e=1}^{N_{\mathrm{I}}}\exp(r_{e,i}^{\mathrm{s}})^2 + \log\sum_{e=1}^{N_{\mathrm{E}}}\exp(r_{e,i}^{\mathrm{v}})^2\right), \tag{7}$$

where $r_{e,i}^{\mathrm{v}}$ and $r_{e,i}^{\mathrm{s}}$ represents the R^n–invariant score from invariant and equivariant routers for the i-th node of the e-th expert, and N_{I} and N_{E} are the number of invariant and equivariant experts.

(2) Top-k expert selection with Gumbel-Sigmoid sampling [47, 48], where the Gumbel noise allows the experts to be better differentiated in the process of training:

$$\text{Gumbel-Sigmoid}(x) = \mathrm{Sigmoid}(x + G' - G''), \tag{8}$$

where routing logits $r_{i,e}^{\mathrm{s}}$ or $r_{i,e}^{\mathrm{v}}$ are input x, and G' and G'' are independent Gumbel noise samples.

(3) Expert upcycling strategy [49] that gradually increases the number of active experts during training. Upcycling combined with the Gumbel-Sigmoid technique allows us to overcome the limitations of static expert selection and achieve improved performance in our geometry-aware sparse architecture.

3.2.3 Ensemble Encoding

The ensemble encoding block enables cross-conformer interactions while preserving S_n–equivariance. Let $\bar{\mathbf{H}} = (1/n)\sum_{c=1}^{n}\widetilde{\mathbf{H}}_c^{\mathrm{s}} \in \mathbb{R}^{|\mathcal{V}|\times 1\times d}$ denote the mean-pooled representation of S_n–equivariant, R^n–invariant scalar features $\widetilde{\mathbf{H}}^{\mathrm{s}}$ across all conformers, which is S_n**–invariant** and R^n**–invariant**. This is processed by a Graph Isomorphism Network (GIN) layer [50], incorporated with 2D topological graph edge information. The propagation rule updates node i embeddings as:

$$\bar{\boldsymbol{h}}_i = \varphi\left((1+\epsilon)\bar{\boldsymbol{h}}_i + \sum_{j\in\mathcal{N}(i)}\bar{\boldsymbol{h}}_j\right), \tag{9}$$

where ϵ is learnable and φ is a two-layer perceptron with ReLU activation. Let $\bar{\mathbf{H}}^{\mathrm{s}} \in \mathbb{R}^{|\mathcal{V}|\times 1\times d}$ denote the final node embedding matrix after the GIN interaction layer, which aggregates information across the conformer ensemble, and is S_n**–invariant** and R^n**–invariant** from (1) of Lemma 1.

3.2.4 Gated Aggregation

While the GMoE mechanism handles self-attention within each conformer, we employ gated aggregation to facilitate information passing between conformer-level and molecular-level features.

This cross-attention mechanism integrates *conformer-level* features $\mathbf{H}^{\text{s}} \in \mathbb{R}^{(n\times|\mathcal{V}|)\times 1\times d}$ and the aggregated *molecular-level* GIN output features $\bar{\mathbf{H}}^{\text{s}} \in \mathbb{R}^{|\mathcal{V}|\times 1\times d}$. For ease of discussion, we squeeze the scalar dimension for both representations (`dim=-2`) in the following matrix multiplications. The gated integration is defined as $\mathbf{X}^{\text{s}} = \text{Attn}(\mathbf{H}^{\text{s}}, \bar{\mathbf{H}}^{\text{s}})$, where the attention mechanism computes:

$$\boldsymbol{Q} = \bar{\mathbf{H}}^{\text{s}}\boldsymbol{W}_Q, \qquad \boldsymbol{K} = \mathbf{H}^{\text{s}}\boldsymbol{W}_K, \qquad \boldsymbol{V} = \mathbf{H}^{\text{s}}\boldsymbol{W}_V, \tag{10}$$

$$\text{Attn}(\mathbf{H}^{\text{s}}, \bar{\mathbf{H}}^{\text{s}}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}}{\sqrt{d_k}}\right)\boldsymbol{V}. \tag{11}$$

Here, $\mathbf{W}_Q \in \mathbb{R}^{d\times d_k}, \mathbf{W}_K \in \mathbb{R}^{d\times d_k}, \mathbf{W}_V \in \mathbb{R}^{d\times d}$ are learnable projection matrices, d_k is the dimension of the attention space. After reshaping, we have the output $\mathbf{X}^{\text{s}} \in \mathbb{R}^{n\times|\mathcal{V}|\times 1\times d}$. This mechanism enables each conformer to selectively incorporate molecular-level and inter-conformer information, facilitating better ensemble representation learning. Since every input here are R^n–invariant, the output $\mathbf{X}^{\text{s}}$ is also R^n**–invariant** from (1) of Lemma 1. We further prove in Section C.2 that $\mathbf{X}^{\text{s}}$ is also S_n**–equivariant**.

Finally, we denote $\mathbf{X}^{(l+1)} = \left[\mathbf{X}^{s}, \widetilde{\mathbf{H}}^{v}\right] \in \mathbb{R}^{N\times 4\times d}$ as the next-level hybrid input feature that satisfies S_n**–equivariant** and R^n**–equivariant**, thus completing our single H–equivariant layer architecture.

3.3 Model Training

The proposed architecture consists of an equivariant graph neural network with L interaction layers. Each interaction layer contains N_{I} invariant experts and N_{E} equivariant experts to process type-0 and type-1 feature separately. For regression tasks, the objective is a weighted sum of the standard Mean Squared Error (MSE) loss and an auxiliary z-loss term. For classification tasks, a binary classification head equipped with sigmoid activation is attached to the final molecular representations and we use a binary cross entropy loss for model training.

4 Experiments

In this section, we present comprehensive experimental evaluations of SPiCE across multiple molecular datasets and tasks. We first describe our datasets and experimental setup, then present results and analyses. We aim to answer the following research questions:

- **RQ1: Performance.** How does SPiCE perform across different molecular property prediction tasks compared to state-of-the-art conformer ensemble methods?
- **RQ2: Scalability.** How does the performance of SPiCE scale with dataset size?
- **RQ3: Architecture design.** What is the impact of key architectural choices in SPiCE?

4.1 Experimental Setup

Datasets. We evaluate SPiCE on four datasets spanning both regression and classification tasks: (1) Drugs-7.5K, obtained by downsampling 10% of Drugs-75K [51] with a fixed random seed due to computational constraints, with three quantum mechanical properties: IP, EA, and χ, (2) Kraken [52] with four 3D ligand descriptors: (Sterimol B_5, Sterimol L, BurB_5, BurL), and (3-4) CoV2 and CoV2-3CL from GEOM-Drugs [53]: CoV2 measures general inhibition in human cells, while CoV2-3CL specifically targets the 3CL protease inhibition. Prior to training, we perform preprocessing including conformer deduplication, clustering, and selection following the methodology described in prior works [30, 51]. We present detailed statistics and descriptions of datasets in Appendix E.

Metrics. Following prior works, we use Mean Absolute Error (MAE) for regression tasks and Receiver Operating Characteristic (ROC) area under the curve for classification tasks, particularly appropriate for the highly imbalanced datasets CoV2 and CoV2-3CL.

Baselines. We evaluate SPiCE against three categories of baselines. (1) 1D string-based and 2D topological models, including fingerprints with random forest [54], extended 3D fingerprints with random forest [55], and 2D GNN models (GIN [50], GIN with virtual nodes [56], and GraphGPS [57]). (2) 3D single-conformer GNNs with random conformer sampling [51], including PaiNN [37], ViSNet [58], ClofNet [13], and Equiformer [34]. (3) Conformer ensemble methods: For each 3D GNN backbone, we implement three conformer aggregation strategies following [51]: mean

Table 1: Performance in terms of MAE (↓) for seven regression tasks (Drugs-7.5K, Kraken) and ROC scores (↑) for two classification tasks (CoV2, CoV2-3CL). **Bold** and underlined values indicate best and second-best overall performance, respectively.

Backbone	Ensemble Strategy	Drugs-7.5K (MAE, ↓)			Kraken (MAE, ↓)				CoV2	3CL
		IP	EA	χ	B_5	L	$BurB_5$	BurL	ROC (↑)	ROC (↑)
► *1D String-based and 2D Topological Approaches*										
Fingerprint+RF [54]		0.5833	0.5277	0.3130	0.4760	0.4303	0.2758	0.1521	0.6071	0.9013
E3FP+RF [55]		0.6217	0.5774	0.3464	0.6249	0.5535	0.3692	0.1908	0.6046	0.7676
GIN [50]		0.5575	0.5116	0.2892	0.3128	0.4003	0.1719	0.1200	0.3708	0.5942
GIN-VN [56]		0.5398	0.5160	0.2937	0.3567	0.4344	0.2422	0.1741	0.4832	0.7387
GraphGPS [57]		0.5480	0.5054	0.2863	0.3450	0.4363	0.2066	0.1500	0.5601	0.8387
► *3D Single-Conformer Graph Neural Networks with Random Conformer Sampling [51]*										
PaiNN [37]		0.5557	0.5127	0.2924	0.3443	0.4471	0.2395	0.1673	0.2997	0.8368
ClofNet [13]		0.6316	0.6008	0.3615	0.4473	0.6369	0.3216	0.2426	0.5233	0.7562
Equiformer [34]		0.5471	0.4898	0.2887	0.2709	0.3759	0.2019	0.1526	0.4577	0.8035
ViSNet [58]		0.5393	0.4855	0.2985	0.3828	0.4495	0.2400	0.1755	0.5011	0.4774
► *Conformer Ensemble Approaches*										
ConfNet [60]		0.5760	0.5359	0.3057	0.4469	0.4680	0.2686	0.1657	0.5010	0.4930
ConAN-FGW [31]		0.5471	0.4945	0.2891	0.3242	0.5178	0.2026	0.1492	0.6340	0.9180
PaiNN [37]	Mean	0.5410	0.4966	0.2963	0.2877	0.3950	0.1817	0.1472	0.5722	0.8850
	DeepSets	0.5396	0.5091	0.2982	0.2225	0.3619	0.1693	0.1324	0.5802	0.6808
	Attention	0.6318	0.5985	0.3488	0.3496	0.4109	0.2123	0.1506	0.4179	0.6984
	SPiCE	0.5281	0.4929	**0.2792**	0.2178	0.3548	**0.1564**	0.1292	0.5910	0.8880
ClofNet [13]	Mean	0.5935	0.5441	0.3121	0.3986	0.5674	0.2857	0.2327	0.3900	0.7580
	DeepSets	0.5912	0.5533	0.3153	0.3314	0.5375	0.2532	0.1983	0.6208	0.7628
	Attention	0.6694	0.5949	0.3578	0.4979	0.6118	0.3353	0.2502	0.3707	0.8182
	SPiCE	0.5747	0.5283	0.3059	0.3193	0.4903	0.2477	0.1913	0.6730	**1.0000**
Equiformer [34]	Mean	0.5457	0.4932	0.2977	0.2303	0.3830	0.1680	0.1259	0.5601	0.8387
	DeepSets	0.5404	0.4888	0.2990	0.2564	0.3772	0.1782	0.1234	0.5125	0.7134
	Attention	0.5488	0.4923	0.2896	0.3187	0.4508	0.1673	0.1425	0.3882	0.7881
	SPiCE	0.5318	0.4830	0.2816	0.2241	**0.3456**	0.1611	**0.1229**	0.5650	0.8405
ViSNet [58]	Mean	0.5593	0.4927	0.2862	0.2811	0.3970	0.1874	0.1469	0.6035	0.7447
	DeepSets	**0.5280**	0.4987	0.2846	0.3104	0.4113	0.1716	0.1314	0.6626	0.4160
	Attention	0.5593	0.4988	0.2944	0.3755	0.4195	0.2384	0.1394	0.5262	0.7158
	SPiCE	0.5384	**0.4538**	0.2814	**0.2715**	0.3807	0.1657	0.1277	**0.6890**	0.7195

pooling, DeepSets [33], and self-attention [59]. We also compare against specialized ensemble models: ConfNet [60] and ConAN-FGW [31]. Please refer to Appendix F for details of baselines.

Note that SPiCE is designed as a plug-and-play framework that requires compatible backbones with equivariant representations. While we primarily use equivariant models due to their higher representational capabilities, we also evaluate invariant 3D models as baselines in Appendix H.

Experimental settings. Following prior works, for regression datasets, we randomly partition data into training, validation, and test sets with a 7:1:2 ratio, while classification datasets use fixed public splits. We optimize models using AdamW [61] with a cosine decay scheduler. Training terminates if the loss shows no improvement for 400 consecutive epochs. To ensure consistent comparison, we set the latent feature dimension to 128 and limit each molecule to a maximum of 20 conformers. For classification tasks, we address label imbalance by maintaining a 1:1 ratio of positive to negative samples during training. Other configurations follow the original settings from respective papers. The details of experimental settings are provided in Appendix D.

4.2 Main Results (RQ1)

Table 1 summarizes the performance across all tasks. Previous studies have established that conformer ensemble learning presents unique challenges: explicit set encoding can improve performance but makes training more challenging due to computational complexity, and model performance often shows strong task dependencies [30, 51]. Despite these challenges, SPiCE consistently

achieves superior performance compared to baseline methods, **outperforming in 34 out of 36 total experimental configurations across all 9 tasks and 4 base models** and often surpassing the state-of-the-art model ConAN-FGW.

Notably, SPiCE demonstrates robust performance across datasets of varying sizes, from the smaller Kraken to the larger Drugs-7.5K and CoV2 datasets, suggesting that our architecture effectively balances computational efficiency with modeling capacity. The improvements are particularly significant with PaiNN, though performance gains vary across tasks, which is consistent with past observation in conformer ensemble modeling where different structural features may dominate different properties.

Regression tasks. On the Drugs-7.5K and Kraken datasets, SPiCE demonstrates consistent improvements across all backbone architectures. For Drugs-7.5K, SPiCE achieves relative MAE reductions of 2.79%, 7.89%, and 5.78% for IP, EA, and χ respectively, compared to the second-best strategy. This is due to its strong geometric feature extraction capabilities complementing our selective information processing. The mid-sized Kraken dataset shows even more improvements, with MAE reductions of 3.66%, 8.78%, 7.61%, and 3.53% across its four targets (B_5, L, $BurB_5$, and BurL). The most significant improvement is observed with ClofNet on $BurB_5$, where MAE reduces from 0.5375 to 0.4903. It demonstrates that SPiCE can effectively enhance even simple yet powerful equivariant backbones, a pattern we also observe in classification results.

Classification tasks. SPiCE shows strong performance on our highly imbalanced classification tasks. On the moderately imbalanced CoV2-3CL dataset ($\sim$1:10 positive-negative ratio), the PaiNN backbone achieves the most notable improvements, with ROC-AUC increasing by 10.95% and reaching perfect precision. The more challenging CoV2 dataset ($\sim$1:60 ratio) shows similar trends, increasing ROC-AUC by 8.41% and thus highlighting the effectiveness of SPiCE and its robustness to extreme class imbalance.

These comprehensive results highlight the effectiveness of our geometry-aware interaction networks in selectively integrating geometric and topological molecular information. The consistent improvements across diverse tasks, backbones, and data distributions suggest that SPiCE successfully captures meaningful patterns in conformer ensembles.

4.3 Analysis of Model Scaling (RQ2)

Processing conformer ensembles introduces significant computational overhead, making it essential to understand how model performance scales with dataset size. To investigate the scalability of SPiCE, we conduct experiments on the EA task using the PaiNN backbone. Starting with the full Drugs-75K dataset (75,099 molecules, 558,002 conformers) [51], we create random subsets ranging from 10% to 100% of the data at 10% intervals.

Our analysis reveals two findings. First, SPiCE shows consistent improvements with increasing data size, reducing MAE from 0.4929 (7.5K molecules) to 0.4386 (75K molecules). As shown in Figure 2(a), the approximately linear scaling relationship suggests that our geometry-aware architecture effectively leverages additional training data without encountering performance plateaus that often characterize capacity-limited models. Second, we plot the loss trajectories at 30%, 70%, and 100% dataset sizes for the first 200 epochs in Figure 2(b). They all show similar patterns despite varying data scales. This consistent optimization behavior suggests that SPiCE maintains stable learning characteristics across different dataset sizes, making it suitable for applications with varied data.

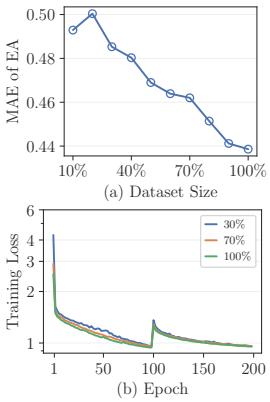

Figure 2: (a) Performance as the dataset size increases from 7.5K to 75K molecules. (b) Training loss trajectories of the first 200 epochs for 30%, 70%, and 100% of 75K molecules.

4.4 Ablation Studies (RQ3)

To better understand the key design choices, we conducted extensive ablation studies using PaiNN as the backbone and EA as the task. We disabled or modified individual components of SPiCE while keeping other parts unchanged. All experiments used the same protocol to ensure fair comparison.

Figure 3 presents our ablation results. First, the type-1 MoE component for vector feature processing demonstrates the highest importance, with its removal causing a substantial performance drop. This shows the necessity of dedicated equivariant feature handling. Similarly, the topological aggregator proves crucial, confirming that molecular topology provides essential information for conformer ensemble representations. Third, the router mechanisms, both Gumbel sampling and nonlinear activation, significantly influence model performance, indicating that sophisticated expert selection mechanisms are fundamental for effective specialization. Lastly, the gated aggregation component validates the selective cross-conformer integration, while the relatively smaller impact of atom-wise routing and layer-wise MoE suggests some implementation flexibility in these architectural aspects.

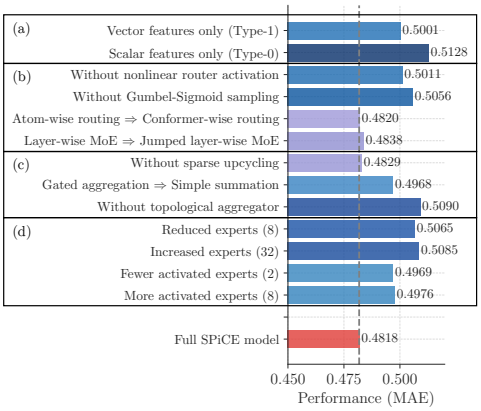

Figure 3: Results of ablation studies regarding (a) feature processing, (b) routing mechanism, (c) training & integration, and (d) expert granularity.

4.5 Additional Experiments

We conducted several extra experiments to further validate our approach. Ablation studies on MoE positioning and router z-loss regularization support our architectural choices. The experiments show that conformer-wise routing with layer-wise MoE placement achieves optimal performance, and moderate z-loss regularization ($\lambda = 1e^{-4}$) effectively balances expert utilization with training stability. Additionally, we evaluated SPiCE with invariant backbones and on the BDE dataset [51]. These additional experiments are documented in Appendices G and H.

5 Conclusions

We present SPiCE, a hierarchical framework for molecular conformer ensemble learning that combines three key architectural innovations: (1) shared conformer encoding for geometry-preserving molecular processing, (2) geometric mixture-of-experts for specialized handling of scalar and vector features, and (3) hierarchical ensemble encoding that integrates molecular topology with selective cross-conformer communication. SPiCE maintains essential symmetries, both in conformer permutation and geometric transformations. Our comprehensive evaluation demonstrates the effectiveness of SPiCE across diverse molecular prediction tasks.

Acknowledgments

This work was partially supported by NSF Center for Computer Assisted Synthesis (2202693), National Artificial Intelligence Research Resource (NAIRR) Pilot (240280, 240443), National Science Foundation (2106859, 2211557, 2119643, 2200274, 2303037, 2312501), National Institutes of Health (U54HG012517, U24DK097771, U54OD036472), NEC, Optum AI, SRC JUMP 2.0 Center, Amazon Research Awards, and Snapchat Gifts.

References

[1] Daniel S. Wigh, Jonathan M. Goodman, and Alexei A. Lapkin. A Review of Molecular Representation in the Age of Machine Learning. *WIREs Comput. Mol. Sci.*, 12(5):e1603, 2022. 1

[2] Jun Xia, Yanqiao Zhu, Yuanqi Du, and Stan Z. Li. A Systematic Survey of Chemical Pre-trained Models. In *IJCAI*, 2023. 1

[3] Laurianne David, Amol Thakkar, Rocío Mercado, and Ola Engkvist. Molecular Representations in AI-Driven Drug Discovery: A Review and Practical Guide. *J. Cheminformatics*, 12(1):56, 2020. 1

[4] Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, and Pascal Friederich. Graph Neural Networks for Materials Science and Chemistry. *Commun. Mater.*, 3(1):93, 2022. 1

[5] Mandana Saebi, Bozhao Nan, John E. Herr, Jessica Wahlers, Zhichun Guo, Andrzej M. Żurański, Thierry Kogej, Per-Ola Norrby, Abigail G. Doyle, Nitesh V. Chawla, and Olaf Wiest. On the Use of Real-World Datasets for Reaction Yield Prediction. *Chem. Sci.*, 14(19):4997–5005, 2023. 1

[6] David Weininger. SMILES, A Chemical Language and Information System. *J. Chem. Inf. Comput. Sci.*, 28(1):31–36, February 1988. 1

[7] H. L. Morgan. The Generation of a Unique Machine Description for Chemical Structures — A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.*, 5(2):107–113, 1965. 1

[8] Robert C. Glem, Andreas Bender, Catrin H. Arnby, Lars Carlsson, Scott Boyer, and James Smith. Circular Fingerprints: Flexible Molecular Descriptors with Applications from Physical Chemistry to ADME. *IDrugs*, 9(3):199–204, 2006. 1

[9] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *arXiv.org*, April 2021. 1

[10] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. SchNet: A Continuous-Filter Convolutional Neural Network for Modeling Quantum Interactions. In *NIPS*, pages 991–1001, 2017. 1, 4, 16

[11] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional Message Passing for Molecular Graphs. In *ICLR*, 2020. 1, 16

[12] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. GemNet: Universal Directional Graph Neural Networks for Molecules. In *NeurIPS*, pages 6790–6802, 2021. 1, 16

[13] Weitao Du, He Zhang, Yuanqi Du, Qi Meng, Wei Chen, Nanning Zheng, Bin Shao, and Tie-Yan Liu. SE(3) Equivariant Graph Neural Networks with Complete Local Frames. In *ICML*, pages 5583–5608, 2022. 1, 7, 8, 22, 26

[14] Fabian Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. In *NeurIPS*, pages 1970–1981, 2020. 1

[15] 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. *Nat. Commun.*, 13(1):2453, 2022. 1, 25

[16] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) Equivariant Graph Neural Networks. In *ICML*, pages 9323–9332, 2021. 1, 16

[17] Bharath Ramsundar, Peter Eastman, Patrick Walters, and Vijay Pande. *Deep Learning for the Life Sciences: Applying Deep Learning to Genomics, Microscopy, Drug Discovery, and More*. O'Reilly Media, 2019. 2

[18] Dmitry Zankov, Timur Madzhidov, Alexandre Varnek, and Pavel Polishchuk. Chemical Complexity Challenge: Is Multi-Instance Machine Learning A Solution? *WIREs Comput. Mol. Sci.*, page e1698, November 2023. 2

[19] Emanuele Perola and Paul S. Charifson. Conformational Analysis of Drug-Like Molecules Bound to Proteins: An Extensive Study of Ligand Reorganization upon Binding. *J. Med. Chem.*, 47(10):2499–2510, 2004. 2

[20] R.D. Levine. *Molecular Reaction Dynamics*. Cambridge University Press, 2009. 2

[21] Andrew F. Zahrt, Jeremy J. Henle, Brennan T. Rose, Yang Wang, William T. Darrow, and Scott E. Denmark. Prediction of Higher-Selectivity Catalysts by Computer-Driven Workflow and Machine Learning. *Science*, 363(6424):eaau5631, January 2019. 2

[22] Carolina H. Andrade, Kerly F. M. Pasqualoto, Elizabeth I. Ferreira, and Anton J. Hopfinger. 4D-QSAR: Perspectives in Drug Design. *Molecules*, 15(5):3281–3294, 2010. 2

[23] Malini Ravi, Anton J. Hopfinger, Robert E. Hormann, and Laurence Dinan. 4D-QSAR Analysis of a Set of Ecdysteroids and a Comparison to CoMFA Modeling. *J. Chem. Inf. Comput. Sci.*, 41 (6):1587–1604, November 2001. 2

[24] Dmitry Zankov, Timur Madzhidov, Pavel Polishchuk, Pavel Sidorov, and Alexandre Varnek. Multi-Instance Learning Approach to the Modeling of Enantioselectivity of Conformationally Flexible Organic Catalysts. *J. Chem. Inf. Model.*, 63(21):6629–6641, November 2023. 2

[25] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the Multiple Instance Problem with Axis-Parallel Rectangles. *Artif. Intell.*, 89(1-2):31–71, 1997. 2

[26] Oded Maron and Tomás Lozano-Pérez. A Framework for Multiple-Instance Learning. In *NIPS*, pages 570–576, 1997. 2

[27] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based Deep Multiple Instance Learning. In *ICML*, pages 2132–2141, 2018. 2

[28] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, pages 77–85, 2017. 2

[29] W. Kabsch. A Solution for the Best Rotation to Relate Two Sets of Vectors. *Acta Cryst.*, 32(5): 922–923, Sep 1976. 2

[30] Simon Axelrod and Rafael Gómez-Bombarelli. Molecular Machine Learning with Conformer Ensembles. *Mach. Learn. Sci. Technol.*, 4(3):35025, 2023. 2, 7, 8

[31] Duy Minh Ho Nguyen, Nina Lukashina, Tai Nguyen, An T. Le, TrungTin Nguyen, Nhat Ho, Jan Peters, Daniel Sonntag, Viktor Zaverkin, and Mathias Niepert. Structure-Aware E(3)-Invariant Molecular Conformer Aggregation Networks. In *ICML*, 2024. 2, 8, 22

[32] Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On Learning Sets of Symmetric Elements. In *ICML*, pages 6734–6744, 2020. 3, 17, 22

[33] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan R. Salakhutdinov, and Alexander J. Smola. Deep Sets. In *NIPS*, pages 3391–3401, 2017. 3, 8, 17, 22

[34] Yi-Lun Liao and Tess E. Smidt. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs. In *ICLR*, 2023. 3, 5, 7, 8, 17, 22, 24, 26

[35] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv.org*, July 2016. 4

[36] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning. *Neural Netw.*, 107:3–11, 2018. 4

[37] Kristof Schütt, Oliver T. Unke, and Michael Gastegger. Equivariant Message Passing for the Prediction of Tensorial Properties and Molecular Spectra. In *ICML*, pages 9377–9388, 2021. 5, 7, 8, 16, 22, 24, 26

[38] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-Attention Graph Pooling. In *ICML*, pages 3734–3743. PMLR, 2019. 5

[39] Hongyang Gao, Yi Liu, and Shuiwang Ji. Topology-Aware Graph Pooling Networks. *arXiv.org*, October 2020. 5

[40] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *NIPS*, pages 4801–4811, 2018. 5

[41] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Comput.*, 3(1):79–87, 1991. 5, 17

[42] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning Factored Representations in a Deep Mixture of Experts. In *ICLR*, 2014. 5

[43] William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *J. Mach. Learn. Res.*, 23:120:1–120:39, 2022. 5, 17

[44] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In *ICLR*, 2021. 5, 17

[45] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017. 5

[46] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeffrey Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing Stable and Transferable Sparse Expert Models. *arXiv.org*, February 2022. 6, 17, 18

[47] Wouter Kool, Herke van Hoof, and Max Welling. Stochastic Beams and Where To Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement. In *ICML*, pages 3499–3508, 2019. 6

[48] Xi Victoria Lin, Akshat Shrivastava, Liang Luo, Srinivasan Iyer, Mike Lewis, Gargi Ghosh, Luke Zettlemoyer, and Armen Aghajanyan. MoMa: Efficient Early-Fusion Pre-training with Mixture of Modality-Aware Experts. *arXiv.org*, July 2024. 6, 17

[49] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse Upcycling: Training Mixture-of-Experts from Dense Checkpoints. In *ICLR*, 2023. 6, 23

[50] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful Are Graph Neural Networks? In *ICLR*, 2019. 6, 7, 8

[51] Yanqiao Zhu, Jeehyun Hwang, Keir Adams, Zhen Liu, Bozhao Nan, Brock Stenfors, Yuanqi Du, Jatin Chauhan, Olaf Wiest, Olexandr Isayev, Connor W. Coley, Yizhou Sun, and Wei Wang. Learning Over Molecular Conformer Ensembles: Datasets and Benchmarks. In *ICLR*, 2024. 7, 8, 9, 10, 22

[52] Tobias Gensch, Gabriel dos Passos Gomes, Pascal Friederich, Ellyn Peters, Théophile Gaudin, Robert Pollice, Kjell Jorner, AkshatKumar Nigam, Michael Lindner-D'Addario, Matthew S. Sigman, and Alán Aspuru-Guzik. A Comprehensive Discovery Platform for Organophosphorus Ligands for Catalysis. *J. Am. Chem. Soc.*, 144:1205–1217, January 2022. 7

[53] Simon Axelrod and Rafael Gómez-Bombarelli. GEOM, Energy-Annotated Molecular Conformations for Property Prediction and Molecular Generation. *Sci. Data*, 9(1):185, 2022. 7

[54] Joseph L. Durant, Burton A. Leland, Douglas R. Henry, and James G. Nourse. Reoptimization of MDL Keys for Use in Drug Discovery. *J. Chem. Inf. Comput. Sci.*, 42(6):1273–1280, 2002. 7, 8

[55] Seth D. Axen, Xi-Ping Huang, Elena L. Cáceres, Leo Gendelev, Bryan L. Roth, and Michael J. Keiser. A Simple Representation of Three-Dimensional Molecular Structure. *J. Med. Chem.*, 60(17):7393–7409, 2017. 7, 8

[56] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*, pages 22118–22133, 2020. 7, 8

[57] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer. In *NeurIPS*, pages 14501–14515, 2022. 7, 8

[58] Yusong Wang, Tong Wang, Shaoning Li, Xinheng He, Mingyu Li, Zun Wang, Nanning Zheng, Bin Shao, and Tie-Yan Liu. Enhancing Geometric Representations for Molecules with Equivariant Vector-Scalar Interactive Message Passing. *Nat. Commun.*, 15(1):313, 2024. 7, 8, 22, 26

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Uszkoreit Kaiser, and Illia Polosukhin. Attention is All You Need. In *NIPS*, pages 5998–6008, 2017. 8, 17, 22

[60] Meng Liu, Cong Fu, Xuan Zhang, Limei Wang, Yaochen Xie, Hao Yuan, Youzhi Luo, Zhao Xu, Shenglong Xu, and Shuiwang Ji. Fast Quantum Property Prediction via Deeper 2D and 3D Graph Networks. *arXiv.org*, June 2021. 8, 22

[61] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 8

[62] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *ICML*, pages 1263–1272, 2017. 16

[63] Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J. Bekkers, and Max Welling. Geometric and Physical Quantities Improve E(3) Equivariant Message Passing. In *ICLR*, 2022. 17

[64] Yi-Lun Liao, Brandon Wood, Abhishek Das, and Tess Smidt. EquiformerV2: Improved Equivariant Transformer for Scaling to Higher-Degree Representations. *arXiv.org*, June 2023. 17

[65] Alexandre Duval, Simon V. Mathis, Chaitanya K. Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D. Malliaros, Taco Cohen, Pietro Liò, Yoshua Bengio, and Michael M. Bronstein. A Hitchhiker's Guide to Geometric GNNs for 3D Atomic Systems. *arXiv.org*, December 2023. 17

[66] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *ICML*, pages 3744–3753, 2019. 17

[67] Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak A. Rao, and Bruno Ribeiro. Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs. In *ICLR*, 2019. 17

[68] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast Differentiable Sorting and Ranking. In *ICML*, pages 950–959, 2020. 17

[69] Yan Zhang, Jonathon S. Hare, and Adam Prügel-Bennett. FSPool: Learning Set Representations with Featurewise Sort Pooling. In *ICLR*, 2020. 17

[70] Chirag Pabbaraju and Prateek Jain. Learning Functions over Sets via Permutation Adversarial Networks. *arXiv.org*, 2019. 17

[71] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeffrey Dean. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *ICLR*, 2017. 17

[72] Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. Multimodal Contrastive Learning with LIMoE: the Language-Image Mixture of Experts. In *NeurIPS*, 2022. 17

[73] Damai Dai, Chengqi Deng, Chenggang Zhao, R X Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, Zhenda Xie, Y K Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. *arXiv.org*, January 2024. 18, 23

[74] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y. Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. Mixture-of-Experts with Expert Choice Routing. In *NeurIPS*, 2022. 18

[75] Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermis, Acyr Locatelli, and Sara Hooker. Pushing Mixture of Experts to the Limit: Extremely Parameter Efficient MoE for Instruction Tuning. In *ICLR*, 2024. 18

[76] Joan Puigcerver, Carlos Riquelme Ruiz, Basil Mustafa, and Neil Houlsby. From Sparse to Soft Mixtures of Experts. In *ICLR*, 2024. 18

[77] Benjamin Meyer, Boodsarin Sawatlon, Stefan Heinen, O. Anatole von Lilienfeld, and Clémence Corminboeuf. Machine Learning Meets Volcano Plots: Computational Discovery of Cross-Coupling Catalysts. *Chem. Sci.*, 9:7069–7077, 2018. 24

[78] Noel M. O'Boyle, Michael Banck, Craig A. James, Chris Morley, Tim Vandermeersch, and Geoffrey R. Hutchison. Open Babel: An Open Chemical Toolbox. *J. Cheminform.*, 3(1):1–14, 2011. 24

[79] Johannes Gasteiger, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. Fast and Uncertainty-Aware Directional Message Passing for Non-Equilibrium Molecules. In *ML4Molecules@NeurIPS 2020*, 2020. 24, 25

[80] Julia Balla, Siddharth Mishra-Sharma, Carolina Cuesta-Lazaro, Tommi Jaakkola, and Tess Smidt. A Cosmic-Scale Benchmark for Symmetry-Preserving Data Processing. *arXiv.org*, October 2024. 25

[81] Chaitanya K. Joshi, Cristian Bodnar, Simon V. Mathis, Taco Cohen, and Pietro Liò. On the Expressive Power of Geometric Graph Neural Networks. In *ICML*, pages 15330–15355, 2023. 25

Supplementary Material for SPiCE

A Related Work

A.1 Geometric Graph Neural Networks

Message-passing neural networks have been widely adopted in modeling atomistic systems, where nodes represent atoms and edges represent chemical bonds [62]. When incorporating 3D geometric information, these models can be broadly categorized into invariant and equivariant architectures, based on how they handle geometric transformations.

Invariant models satisfy the condition:

$$\phi(g \cdot G) = \phi(G), \quad \forall g \in \mathrm{E}(3)/\mathrm{SE}(3),$$

These models transform equivariant coordinates $\boldsymbol{X}$ into invariant scalar features (type-0) that remain unchanged under Euclidean transformations. Common invariant features include pairwise distances [10], triplet angles [11], torsion angles [12], etc.

Equivariant models, which satisfy:

$$\phi(g \cdot G) = g \cdot \phi(G), \quad \forall g \in \mathrm{E}(3)/\mathrm{SE}(3),$$

can be further divided into two subcategories. Cartesian equivariant models (e.g., EGNN [16], PaiNN [37]) operate directly in Cartesian coordinates, processing scalar (type-0) and vector (type-1)

features while preserving equivariance through restricted operations. In contrast, spherical equivariant models (e.g., SEGNN [63], Equiformer [34, 64]) convert geometric information into steerable features using spherical harmonics, process rotations through Wigner-D matrices, and combine features using Clebsch-Gordan tensor products. This formulation naturally extends to higher-order geometric features, making it particularly effective for capturing complex geometric patterns in molecular structures.

Readers of interest may refer to Duval et al. [65] for a complete review of geometric graph neural networks for 3D atomic systems. We would like to remark that our SPiCE framework is agnostic to the choice of geometric GNN architecture, allowing practitioners to select the most suitable backbone for their specific application while maintaining the benefits of our conformer ensemble processing approach.

A.2 Learning over Sets

Learning permutation-invariant functions over sets is a fundamental problem in machine learning, with approaches broadly falling into several categories: direct permutation invariance, sorting-based methods, and approximate invariance.

Direct permutation invariance. Deep Sets [33] introduced a simple yet powerful architecture where a permutation-sensitive network processes each element independently, followed by a permutation-invariant aggregation operation. Set Transformers [66] extend this using attention mechanisms for more expressive set processing. DSSNet [32] generalizes this framework to handle symmetries beyond permutation invariance. They show that when working with symmetries that combine permutations with element-wise transformations, the processing can be decomposed into two parts: one that handles individual elements and another that processes information aggregated across the entire set. This framework naturally fits our molecular setting, where we need to maintain both the permutation invariance of conformer ensembles and the geometric symmetries of individual conformers. More sophisticated approaches like Janossy pooling [67] capture higher-order interactions by processing subsets of elements together. Self-attention mechanisms [59] offer another perspective, effectively comparing pairs of elements through learned relationships between queries and keys.

Sorting-based methods. Another strategy achieves permutation invariance by first arranging elements into a canonical order. Recent advances include methods for learning the sorting function itself [68] and approaches that sort based on learned features [69]. While these methods are computationally efficient, they face challenges in training due to the discrete nature of sorting operations.

Approximate invariance. Some methods trade exact invariance for computational efficiency. These include stochastic approaches that sample random permutations [67] and adversarial training techniques that encourage approximate invariance [70].

Our work, while informed by these approaches, focuses on maintaining exact invariance through the principled decomposition provided by DSSNet [32]. We propose a novel type-separated Mixture-of-Experts mechanism to enable selective information processing while achieving the required symmetry.

A.3 Mixture-of-Experts

When processing molecular conformers, certain atoms or structural features may be more relevant than others for specific properties. Graph pooling approaches like SAGPool achieve this selective processing by focusing computation on important nodes through attention mechanisms. This naturally extends to Mixture-of-Experts (MoE) architectures that employ multiple specialized neural networks with a learned routing mechanism.

The concept of MoE originated from adaptive mixtures of local experts [41], where separate networks handle different subsets of training cases. Recent developments have transformed MoEs from standalone models to components within deeper architectures [71], enabling selective computation at various scales. Modern MoE architectures have demonstrated remarkable success in large-scale models [43, 44, 46, 48, 72], achieving improved data efficiency and computational performance through sparse activation patterns.

Key design considerations of MoE include balanced expert utilization through load balancing, router design, sparse gating for computational efficiency, and appropriate capacity factors. Router z-loss [46] penalizes extreme routing decisions to prevent expert collapse and improve training stability. Expert capacity factors control the maximum number of tokens per expert, balancing computational efficiency with model performance [73]. While traditional MoEs use token-to-expert routing, Expert Choice routing [74] reverses this by allowing experts to select tokens, improving load balancing and computational efficiency. Recent innovations include parameter-efficient MoE variants [75], and Soft MoE [76], which replaces discrete expert assignment with differentiable soft routing to address training instability and scaling limitations. These developments collectively enable more efficient and stable MoE architectures while preserving the benefits of specialized computation.

This MoE approach is suitable for conformer ensemble modeling as experts can specialize in different geometric patterns or conformational states. We further leverage a gated aggregation mechanism which can adaptively weigh the importance of different conformers based on both local geometric features and global molecular context.

B PyTorch-like Pseudocode

A PyTorch-like pseudocode is given in Algorithm S1.

C Theoretical Details

C.1 S_n–equivariant Linear Layer Structure

Theorem 1. *Consider S_n–equivariant linear layer L that also respects G–symmetry.*

1. *If L takes as input G–equivariant features $\mathbb{R}^{n\times 3\times d}$, then $L(\boldsymbol{X})_i = L_0(\boldsymbol{x}_i)$ operates on each conformer i separately;*
2. *If L takes as input G–invariant features $\mathbb{R}^{n\times 1\times d}$, then L can be decomposed as $L(\boldsymbol{X})_i = L_1(\boldsymbol{x}_i) + L_2\big(\sum_{j=1}^n \boldsymbol{x}_j\big)$, i.e. a local interaction module and a global aggregation module.*

Proof of Theorem 1. For the first claim, we begin by noting that any linear function of equivariant features, if it respects G–symmetry, must be a G–equivariant linear function, hence the output $L(\boldsymbol{X})$ is also G–equivariant, and operates on the 3 coordinate channels of the second dimension of L simultaneously. Next we decompose L into separate components from each individual conformer:

$$L(\boldsymbol{X})_i = \sum_{j=1}^{n} L_{i,j}(\boldsymbol{x}_j), \tag{S1}$$

where each $L_{i,j}$ must be a G–equivariant function. Consider the transformation $g = (g_1, \cdots, g_n) \in G^n$ on input $\boldsymbol{X}$: $g \circ \boldsymbol{X} = \big[g_1 \circ \boldsymbol{x}_1, \cdots, g_n \circ \boldsymbol{x}_n\big]$, where $g_1, \cdots, g_n$ are independent transformations in G. From the G^n–equivariance constraint, we have for any $g \in G^n$ that

$$L(g \circ \boldsymbol{X}) = g \circ L(\boldsymbol{X}) \Leftrightarrow \sum_{j=1}^{n} L_{i,j}(g_j \circ \boldsymbol{x}_j) = g_i \circ \sum_{j=1}^{n} L_{i,j}(\boldsymbol{x}_j), \forall i$$

By fixing g_i and varying g_j for all $j \neq i$, we see the right hand side remains constant, while each $L_{i,j}(g_j \circ \boldsymbol{x}_j) = g_j \circ L_{i,j}(\boldsymbol{x}_j)$ on the left hand side individually changes with different g_j's. Thus the equation can only hold for any g if $L_{i,j}(\boldsymbol{x}_j) = \boldsymbol{0}$ for any $i \neq j$, i.e. there are no cross-conformer interactions, and consequently (S1) becomes

$$L(\boldsymbol{X})_i = L_{i,i}(\boldsymbol{x}_i).$$

From here we finally consider S_n–equivariance: for a permutation $p \in S_n$, defined by $p : [x_i]_{i=1}^n \mapsto [x_{p(i)}]_{i=1}^n$, we have

$$p \circ L(\boldsymbol{X}) = L(p \circ \boldsymbol{X}) \Leftrightarrow L_{p(i),p(i)}(\boldsymbol{x}_{p(i)}) = L_{i,i}(\boldsymbol{x}_{p(i)}), \forall i,$$

which suggests by the arbitrariness of p that $L_{i,i} = L_{j,j}, \forall i, j$. Writing $L_0 = L_{i,i}$ finishes the proof.

Algorithm S1 Pseudocode of SPiCE in a PyTorch-like style.

```
# Inputs:
# batch = [z, c, atom_idx]
# z : (Nx1) atomic numbers for all M conformers
# c : (Nx3) atomic coordinates for all M conformers
# atom_idx : (Nx1) index mapping each atom to its molecule ID
# conf_idx : (Nx1) index mapping each atom to its conformer ID

# --- 1) Preprocess ---
# build scalar & vector features and auxiliary graph attributes
x_s, x_v, 3d_aux = 3d_gnn.preprocess(batch)
_, 2d_aux = GIN.preprocess(batch)

# --- 2) Interaction block ---
for i in range(num_blocks):
    # type-0 feature x_s: Nx1xC
    # type-1 feature x_v: Nx3xC
    h_s, h_v, 3d_aux = 3d_gnn[i].forward(x_s, x_v, 3d_aux)

    # normalized type-0 feature h_s: Nx1xC
    # normalized type-1 feature h_v: Nx3xC
    h_s = ELN_s[i](x_s + h_s)
    h_v = ELN_v[i](x_v + h_v)

    # GMoE block, Eqn. (2)-(8)
    # moe_x: Nx1xC, moe_v: Nx3xC
    # router scores: Nx(N_I+N_E)
    moe_x, moe_v, scores = gmoe_block[i](h_s, h_v)

    # conformer set information sharing
    # h_bar, h_bar_s: MxC
    h_bar = scatter(moe_x, conf_idx, dim=0, reduce='mean')
    h_bar_s = GIN[i](h_bar, 2d_aux)

    # gated aggregation, Eqn. (10)-(11)
    # rev_idx: mapping from conformer-level back to atom ordering
    x_g = gated_aggr[i](h_s, h_bar_s[rev_idx])
    x_g = x_g.unsqueeze(1) # x_g: Nx1xC

    # feature update
    x_s = x_f
    x_v = moe_v

# --- 3) Postprocess ---
# convert final node-level features to molecule-level outputs
out = 3DGNN.postprocess(x_s, x_v, 3d_aux, atom_idx)
```

For the second claim, clearly the output of any G–invariant function is also G–invariant, and cannot contain any type-1 features. We similarly decompose $L(\boldsymbol{X})$ as

$$L(\boldsymbol{X})_i = \sum_{j=1}^{n} L_{i,j}(\boldsymbol{x}_j), \tag{S2}$$

where each $L_{i,j}$ is an G–invariant linear function. Consider $p \in S_n$ again, we have

$$\begin{aligned} p \circ L(\boldsymbol{X}) = L(p \circ \boldsymbol{X}) &\Leftrightarrow L(\boldsymbol{X})_{p(i)} = L(p \circ \boldsymbol{X})_i, \forall i \\ &\Leftrightarrow \sum_{j=1}^{n} L_{p(i),j}(\boldsymbol{x}_j) = \sum_{j=1}^{n} L_{i,j}(\boldsymbol{x}_{p(j)}), \forall i \\ &\Leftrightarrow \sum_{j=1}^{n} L_{p(i),j}(\boldsymbol{x}_j) = \sum_{k=1}^{n} L_{i,p^{-1}(k)}(\boldsymbol{x}_k), \forall i \\ &\Leftrightarrow \sum_{j=1}^{n} \left[L_{p(i),j}(\boldsymbol{x}_j) - L_{i,p^{-1}(j)}(\boldsymbol{x}_j) \right] = 0, \forall i, \end{aligned}$$

where in the third equivalence we used $k = p(j)$ to rewrite the summation. From the independent variabilities of $\boldsymbol{x}_i$, each member of the above summation must take value 0, in other words

$$L_{p(i),j} = L_{i,p^{-1}(j)}, \forall i, j, p,$$

which is equivalent to

$$L_{p(i),p(k)} = L_{i,k}, \forall i, k, p,$$

via substitution of $k = p^{-1}(j)$. This means:

- $L_{i,i} = L_{j,j} = L_{\mathrm{e}}, \forall i, j$;
- $L_{i_1,j_1} = L_{i_2,j_2} = L_{\mathrm{n}}, \forall i_1 \neq j_1, i_2 \neq j_2$;

and therefore (S2) becomes

$$\begin{aligned} L(\boldsymbol{X})_i &= L_{\mathrm{e}}(\boldsymbol{x}_i) + \sum_{j \neq i} L_{\mathrm{n}}(\boldsymbol{x}_j) \\ &= (L_{\mathrm{e}} - L_{\mathrm{n}})(\boldsymbol{x}_i) + L_{\mathrm{n}}\left(\sum_{j=1}^{n} \boldsymbol{x}_j \right), \end{aligned}$$

where in the second equality we used the fact that the summation of a shared linear mapping of components is equal to the same linear mapping of the summation of components. Taking $L_1 = L_{\mathrm{e}} - L_{\mathrm{n}}$ and $L_2 = L_{\mathrm{n}}$ finishes the proof. □

C.2 S_n–equivariance of Cross-Attention Output $\boldsymbol{X}^{\mathrm{s}}$

Here we elaborate on the symmetry preservation of the gated aggregation operation discussed in Section 3.2.4, and prove the S_n–equivariance of the attention output. Looking at (10) and (11), notice the input $\mathbf{H}^{\mathrm{s}}$ is S_n–equivariant while $\bar{\mathbf{H}}^{\mathrm{s}}$ is S_n–invariant, therefore for a permutation $p \in S_n$,

$$\begin{aligned} \mathrm{Attn}(p \circ \mathbf{H}^{\mathrm{s}}, p \circ \bar{\mathbf{H}}^{\mathrm{s}}) = \mathrm{Attn}(p \circ \mathbf{H}^{\mathrm{s}}, \bar{\mathbf{H}}^{\mathrm{s}}) &= \mathrm{softmax}\left(\frac{(p \circ \mathbf{H}^{\mathrm{s}} \boldsymbol{W}_Q)(\bar{\mathbf{H}}^{\mathrm{s}} \boldsymbol{W}_K)^{\mathsf{T}}}{\sqrt{d_k}} \right) \bar{\mathbf{H}}^{\mathrm{s}} \boldsymbol{W}_V \\ &= p \circ \mathrm{softmax}\left(\frac{(\mathbf{H}^{\mathrm{s}} \boldsymbol{W}_Q)(\bar{\mathbf{H}}^{\mathrm{s}} \boldsymbol{W}_K)^{\mathsf{T}}}{\sqrt{d_k}} \right) \bar{\mathbf{H}}^{\mathrm{s}} \boldsymbol{W}_V = p \circ \mathrm{Attn}(\mathbf{H}^{\mathrm{s}}, \bar{\mathbf{H}}^{\mathrm{s}}), \end{aligned} \tag{S3}$$

where the third equality holds since individual operations on each conformer preserves permutation equivariance. This proves the overall S_n–equivariance of the attention module.

Table S1: Hyperparameters for each backbone model on dataset Drugs-7.5K and Kraken.

Dataset	Backbone	Epochs	Batch	LR	Patience	Experts	Act. Experts	Upcycle	τ	β
Drugs-7.5K	PaiNN	2000	32	2e-4	400	8	2	100	0.1	1e-3
	ViSNet	2000	32	1.5e-4	600	8	4	50	0.1	1e-3
	ClofNet	2000	32	1.5e-4	400	8	4	50	0.1	1e-3
	Equiformer	2000	32	2e-4	400	8	4	50	0.1	1e-3
Kraken	PaiNN	2000	16	3e-4	400	16	2	50	0.1	1e-4
	ViSNet	2000	32	4e-4	400	16	4	100	1.0	0.1
	ClofNet	2000	16	1e-4	400	8	2	100	1.5	1e-3
	Equiformer	2000	8	3e-4	400	16	2	50	1.0	1e-4

D Details of Experimental Settings

Different backbones have varying architecture complexities and capacities to learn molecular representations. Datasets also differ in size and complexity of the prediction task. Hyperparameter tuning for each backbone-dataset pair ensures optimal performance by balancing model capacity, data size, and task difficulty. For example, larger datasets like Kraken may require more experts and epochs to sufficiently learn, while simpler backbones like ClofNet need fewer epochs on Drugs-7.5K. The hyperparameters include number of experts, number of activated experts, Gumbel-Softmax sampling temperature, auxiliary z-loss weight, and upcycling epochs.

For regression tasks, each backbone model with a different aggregation strategy is trained with the same set of hyperparameters, optimized through 20 iterations of Bayesian Optimization. Specific settings are summarized in Table S1 for regression datasets. For classification tasks, all experiments use the same settings as PaiNN on Drugs-7.5K, as classification is less sensitive to hyperparameters than regression and Drugs-7.5K is a reliable molecular property prediction benchmark.

E Details of Datasets

We evaluate our method on four diverse molecular datasets. Their statistics is summarized in Table S2.

Drugs-7.5K. A subset of 7,500 molecules downsampled from GEOM-Drugs dataset, with three quantum mechanical properties:

- Ionization Potential (IP): The energy required to remove an electron from a neutral molecule (IP $= E_{\text{cation}} - E_{\text{neutral}}$).
- Electron Affinity (EA): The energy change when adding an electron (EA $= E_{\text{neutral}} - E_{\text{anion}}$).
- Electronegativity (χ): Measuring electron attraction tendency ($\chi = -\partial E/\partial N$).

Kraken. A dataset of 1,552 monodentate organophosphorus(III) ligands with DFT-computed conformer ensembles. We focus on four 3D steric descriptors (measured in Å): Sterimol B_5, Sterimol L, buried Sterimol B_5, and buried Sterimol L, which are crucial for QSAR modeling in catalyst design.

CoV2 and CoV2-3CL. Two highly imbalanced classification datasets derived from GEOM-Drugs, containing experimental data for SARS-CoV-2 inhibition:

- CoV2-3CL: Tests specific inhibition of the SARS-CoV-2 3CL protease.
- CoV2: Evaluates general SARS-CoV-2 inhibition in human cell assays.

The class imbalance is summarized in Table S2b, where positive samples (hits) represent only a small fraction of the total molecules.

Table S2: Dataset statistics for all four datasets and dataset splits for COVID-related datasets.

Dataset	#Molecules	#Conformers	#Heavy atoms	#Rot. bonds
Drugs-7.5K	7,509	54,202	30.67	7.45
Kraken	1,552	21,287	23.70	9.05
CoV2	5,466	72,744	24.57	4.83
CoV2-3CL	755	7,742	14.51	2.41

(a) Molecular composition statistics. Numbers of heavy atoms and rotatable bonds ("rot. bonds") are averaged per molecule.

Split	CoV2-3CL	CoV2
Train	50 (485)	53 (3,294)
Validation	15 (157)	17 (1,096)
Test	11 (162)	22 (1,086)
Total	76 (804)	92 (5,476)

(b) Dataset partitioning showing number of active compounds (total compounds in parentheses)

F Details of Baselines

We evaluate SPiCE against two categories of baselines: ensemble learning methods with explicit set encoders and specialized conformer ensemble models.

Ensemble learning with set encoders. We implement our framework with four representative 3D backbones that guarantee E(3) or SE(3) equivariance: PaiNN [37], ViSNet [58], ClofNet [13], and Equiformer [34]. For each backbone, we compare GAIN against three conformer aggregation strategies following Zhu et al. [51]:

- Mean pooling: The simplest approach that computes the average of conformer embeddings $\{\boldsymbol{h}_i\}_{i=1}^n$ generated by 3D GNNs:

$$\boldsymbol{h}_{\text{MEAN}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{h}_i. \tag{S4}$$

- DeepSets [33]: A permutation-invariant function that processes the ensemble through a Multi-Layer Perceptron (MLP) ϕ, followed by sum pooling and another MLP ρ:

$$\boldsymbol{h}_{\text{DS}} = \text{MLP}_\rho\left(\sum_{i=1}^{n}\text{MLP}_\phi(\boldsymbol{h}_i)\right), \tag{S5}$$

  where MLP_ϕ transforms individual embeddings and MLP_ρ processes the aggregated features. This approach preserves more individual conformer information than mean pooling at the cost of additional non-linear transformations.

- Self-attention [59]: Computes a weighted sum of embeddings using attention scores:

$$\boldsymbol{h}_{\text{ATTN}} = \sum_{i=1}^{n}\alpha_i\boldsymbol{h}_i, \tag{S6}$$

$$\alpha_i = \text{softmax}(\boldsymbol{h}_i^\top\boldsymbol{W}\boldsymbol{h}_i). \tag{S7}$$

  This approach captures pairwise interactions between conformers through learned attention weights.

After obtaining the ensemble embeddings through these set encoders, a linear projection head generates the final predictions.

Specialized conformer ensemble models. We also compare against two state-of-the-art models specifically designed for conformer ensemble learning:

- ConfNet [60]: Extends DSSNet [32] as an explicit set encoder, applying permutation-invariant operations directly to the conformer ensemble while maintaining geometric equivariance.
- ConAN-FGW [31]: Introduces a novel 2D–3D aggregation mechanism based on the Fused Gromov-Wasserstein Barycenter problem. It combines this with efficient conformer generation using distance geometry through RDKit, enabling joint optimization of conformer generation and property prediction.

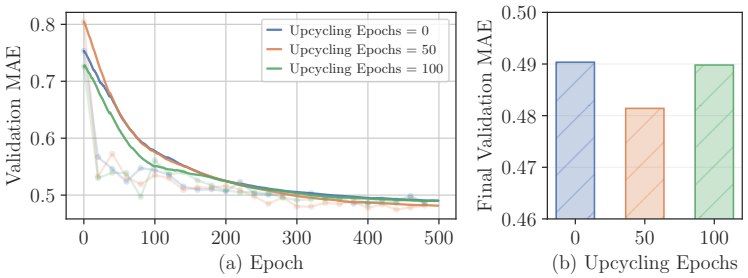

Figure S1: Impact of upcycling epochs on model performance. (a) Validation loss curves for different upcycling epochs on the Drugs-7.5K dataset using ViSNet backbone for EA prediction. (b) MSE comparison of validation errors.

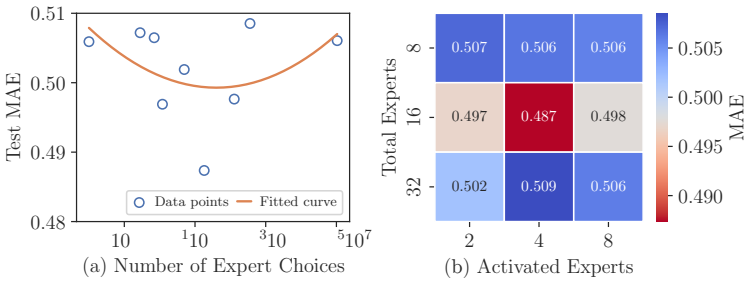

Figure S2: Analysis of expert combinations on model performance. Results show how varying the number of total experts (8–32) and active experts (2–8) affects prediction accuracy on the Drugs-7.5K dataset using PaiNN backbone for EA prediction.

G Detailed Ablation Studies

To better understand the key design choices in SPiCE, we conduct comprehensive ablation studies examining four critical aspects: (1) the effectiveness of sparse upcycling for stabilizing early-stage training, (2) the impact of expert granularity on model performance, (3) the optimal positioning of TS-MoE within the architecture, and (4) the influence of router z-loss on training stability and expert utilization. Here we present analyses of the first two aspects, while full results for the rest are provided in Appendix G.

Sparse upcycling. Training both GNN representation and MoE components simultaneously can be challenging in early stages. The sparse upcycling strategy [49] addresses this by separating the training into two phases: first training a single expert to optimize representation learning, then cloning it into multiple experts with a newly initialized router for specialization.

As shown in Figure S1, we evaluate different upcycling epochs (0, 50, 100) on the Drugs-7.5K dataset using ViSNet for EA prediction. Results indicate that 50 upcycling epochs yield optimal performance, while excessive upcycling epochs can slow convergence. This confirms that appropriate upcycling improves both convergence speed and final accuracy.

Expert granularity. While finer expert granularity can increase model flexibility through more possible combinations (e.g., $\binom{8}{2} = 28$ vs. $\binom{16}{4} = 1820$) [73], it also introduces computational overhead and potential training instability. Figure S2 presents a systematic study varying the number of total experts (8, 16, 32) and activated experts (2, 4, 8) on the Drugs-7.5K dataset with PaiNN backbone. The results reveal a U-shaped performance curve, with optimal performance at $\binom{16}{4} = 1820$ combinations. This suggests a sweet spot balancing model flexibility with computational efficiency for molecular conformer ensemble representation.

MoE positioning. Table S3a compares different MoE configurations on the Kraken dataset's BurB_5 target using PaiNN. Our experiments examine three key design choices: conformer-wise vs. atom-wise routing, layer-wise MoE placement, and non-linear router activation. Results show that conformer-wise routing with layer-wise MoE and softmax activation achieves the best performance, supporting our design choice of treating conformers rather than atoms as routing objectives.

Table S3: Ablation studies on (a) different architectural choices, (b) router z-loss weight (λ). Results show validation MAE on Kraken ($BurB_5$) and Drugs-7.5K (EA) datasets using PaiNN and ViSNet respectively.

Atom-Wise Routing	Layer-Wise MoE	Router Activation	MAE
✗	✓	✓	0.1645
✓	✗	✓	0.1515
✓	✓	✗	0.1546
✓	✓	✓	**0.1470**

(a) Impact of architectural choices

λ	MAE
$1e^{-6}$	0.4973
$1e^{-5}$	0.4960
$1e^{-4}$	**0.4888**
$1e^{-3}$	0.4985

(b) Z-loss λ

Router z-loss. Router z-loss penalizes extreme routing decisions and helps balance expert utilization. Table S3b shows the impact of different z-loss weights (λ) on validation error using the ViSNet backbone on Drugs-75K. A moderate weight of $1e^{-4}$ achieves optimal performance, suggesting that proper regularization improves model stability without overly constraining expert specialization. Larger weights, while promoting more balanced expert utilization, can lead to training instability due to excessive penalization.

H Additional Experiments

BDE dataset. To showcase our method is effective and scalable to reaction-based prediction task, we conducted experiments on the BDE dataset [77]. The BDE dataset comprises 5,915 organometallic catalysts (ML_1L_2) with diverse ligands and metal centers, including DFT-computed binding energies for conformer ensembles of unbound and bound states. Conformers are generated via Open Babel [78] and force field optimizations, approximating global minima. Due to the high cost of DFT optimization, precise conformer ensembles are generally unknown at inference, making this a realistic and challenging benchmark. The goal is to predict binding energies from either individual or ensemble conformers of the catalyst in both states.

Different from the experimental settings in the main paper, this task involves two distinct sets of conformers as input, corresponding to the unbound and bound states of the catalyst. To accommodate this, we employ two parameter-independent copies of SPiCE, with their regression heads removed, to separately process each conformer set. The resulting molecular-level features are then concatenated and passed through a regression head to predict the final binding energy. Table S4a presents the performance of SPiCE on this dataset in comparison with the two other ensemble models Mean and DeepSets with PaiNN [37] and Equiformer [34] as backbones. It is seen that SPiCE consistently outperforms the baselines, demonstrating its effectiveness in reaction-based prediction tasks.

Invariant Type-0-only model. Our SPiCE framework is designed as a plug-and-play architecture that can be adapted to work with different geometric neural network backbones. While our method

Table S4: Additional experiments on the BDE dataset and invariant models. **Bold** values indicate best performance.

Backbone	Model	Binding Energy
PaiNN [37]	Mean	1.8744
	DeepSets	1.9164
	SPiCE	**1.8528**
Equiformer [34]	Mean	1.9136
	DeepSets	1.9540
	SPiCE	**1.8978**

(a) Reaction-based regression task (BDE) in MAE (↓).

Model	B_5	L
Mean	0.3172	0.4258
DeepSets	0.2627	0.3777
SPiCE (Invariant)	**0.2446**	**0.3379**

(b) Invariant type-0-only model on Kraken dataset with DimeNet++ [79] as backbone.

can accommodate both invariant and equivariant backbones, we primarily focus on equivariant 3D GNNs (PaiNN, ClofNet, Equiformer, and VisNet) due to their superior representational capabilities. Equivariant features have been shown to provide steeper learning curves, improved data efficiency, and finer angular resolution for capturing molecular structural details [15, 80, 81].

To demonstrate the versatility of SPiCE and address potential questions about backbone compatibility, we developed a simplified variant that operates exclusively on invariant (type-0) features, compatible with invariant-only backbones such as DimeNet++ [79]. This adaptation removes the type-1 processing branch including vector features, type-1 router, and the corresponding expert network to accommodate scalar-only representations.

We evaluate this invariant variant on the Kraken dataset against strong baselines including DeepSets and mean pooling (results in Table S4b). We note that this experiment is not intended as a direct comparison between invariant and equivariant approaches. Rather, it validates the effectiveness of our design in a constrained setting and confirms the adaptability of our model across different architectural paradigms. Even without equivariant information, our adapted framework can outperform the invariant baselines, demonstrating that the core design remains beneficial across different geometric representations.

I Code and Dataset Availability

The implementation of this work can be found in this repository: https://github.com/DanieSYD/SPiCE.

J Limitations and Future Work

While SPiCE achieves strong performance across a variety of molecular property prediction tasks, the hierarchical architecture and dependence on equivariant GNN backbones introduce additional computational overhead compared to simpler pooling methods, potentially limiting scalability in resource-constrained environments or applications requiring real-time inference. Furthermore, although SPiCE facilitates cross-conformer interactions through attention-based integration, the framework does not explicitly incorporate thermodynamic priors or statistical mechanical principles that govern conformational ensembles.

Future work could address these limitations through several directions: developing more efficient attention mechanisms or approximation strategies to reduce computational cost, incorporating physical distribution information, and exploring hybrid approaches that balance computational efficiency with physical grounding.

K Raw Performance Data

The raw performance data with standard deviation of Table 1 is summarized in Table S5.

Table S5: Raw performance (mean ± standard deviation) in terms of MAE (↓) for seven regression tasks (Drugs-7.5K, Kraken) and ROC scores (↑) for two classification tasks (CoV2, CoV2-3CL).

Backbone	Ensemble Strategy	Drugs-7.5K (MAE, ↓)			Kraken (MAE, ↓)				CoV2	3CL
		IP	EA	χ	B_5	L	$BurB_5$	BurL	ROC (↑)	ROC (↑)
PaiNN [37]	Mean	$0.5410_{\pm 0.0462}$	$0.4966_{\pm 0.0336}$	$0.2963_{\pm 0.0190}$	$0.2877_{\pm 0.0252}$	$0.3950_{\pm 0.0233}$	$0.1817_{\pm 0.0091}$	$0.1472_{\pm 0.0039}$	$0.5722_{\pm 0.0518}$	$0.8850_{\pm 0.1209}$
	DeepSets	$0.5396_{\pm 0.0534}$	$0.5091_{\pm 0.0129}$	$0.2982_{\pm 0.0052}$	$0.2225_{\pm 0.0218}$	$0.3619_{\pm 0.0192}$	$0.1693_{\pm 0.0111}$	$0.1324_{\pm 0.0091}$	$0.5802_{\pm 0.0356}$	$0.6808_{\pm 0.1239}$
	Attention	$0.6318_{\pm 0.0327}$	$0.5985_{\pm 0.0160}$	$0.3488_{\pm 0.0126}$	$0.3496_{\pm 0.0140}$	$0.4109_{\pm 0.0167}$	$0.2123_{\pm 0.0005}$	$0.1506_{\pm 0.0029}$	$0.4179_{\pm 0.0357}$	$0.6984_{\pm 0.1994}$
	SPiCE	$0.5281_{\pm 0.0666}$	$0.4929_{\pm 0.0331}$	$0.2792_{\pm 0.0030}$	$0.2178_{\pm 0.0376}$	$0.3548_{\pm 0.0199}$	$0.1564_{\pm 0.0154}$	$0.1292_{\pm 0.0031}$	$0.5910_{\pm 0.0831}$	$0.8880_{\pm 0.2113}$
ClofNet [13]	Mean	$0.5935_{\pm 0.0672}$	$0.5441_{\pm 0.0072}$	$0.3121_{\pm 0.0266}$	$0.3986_{\pm 0.0211}$	$0.5674_{\pm 0.0423}$	$0.2857_{\pm 0.0332}$	$0.2327_{\pm 0.0176}$	$0.3900_{\pm 0.0042}$	$0.7580_{\pm 0.1898}$
	DeepSets	$0.5912_{\pm 0.0447}$	$0.5533_{\pm 0.0292}$	$0.3153_{\pm 0.0174}$	$0.3314_{\pm 0.0187}$	$0.5375_{\pm 0.0154}$	$0.2532_{\pm 0.0043}$	$0.1983_{\pm 0.0008}$	$0.6208_{\pm 0.0354}$	$0.7628_{\pm 0.0184}$
	Attention	$0.6694_{\pm 0.0264}$	$0.5949_{\pm 0.0352}$	$0.3578_{\pm 0.0096}$	$0.4979_{\pm 0.0199}$	$0.6118_{\pm 0.0328}$	$0.3353_{\pm 0.0109}$	$0.2502_{\pm 0.0099}$	$0.3707_{\pm 0.0149}$	$0.8182_{\pm 0.1042}$
	SPiCE	$0.5747_{\pm 0.0362}$	$0.5283_{\pm 0.0186}$	$0.3059_{\pm 0.0035}$	$0.3193_{\pm 0.0234}$	$0.4903_{\pm 0.0311}$	$0.2477_{\pm 0.0113}$	$0.1913_{\pm 0.0098}$	$0.6730_{\pm 0.0347}$	$1.0000_{\pm 0.0972}$
Equiformer [34]	Mean	$0.5457_{\pm 0.0349}$	$0.4932_{\pm 0.0125}$	$0.2977_{\pm 0.0160}$	$0.2303_{\pm 0.0059}$	$0.3830_{\pm 0.0291}$	$0.1680_{\pm 0.0004}$	$0.1259_{\pm 0.0011}$	$0.5601_{\pm 0.0351}$	$0.8387_{\pm 0.0982}$
	DeepSets	$0.5404_{\pm 0.0247}$	$0.4888_{\pm 0.0154}$	$0.2990_{\pm 0.0016}$	$0.2564_{\pm 0.0159}$	$0.3772_{\pm 0.0008}$	$0.1782_{\pm 0.0120}$	$0.1234_{\pm 0.0023}$	$0.5125_{\pm 0.0346}$	$0.7134_{\pm 0.0194}$
	Attention	$0.5488_{\pm 0.0205}$	$0.4923_{\pm 0.0371}$	$0.2896_{\pm 0.0247}$	$0.3187_{\pm 0.0074}$	$0.4508_{\pm 0.0352}$	$0.1673_{\pm 0.0058}$	$0.1425_{\pm 0.0195}$	$0.3882_{\pm 0.0377}$	$0.7881_{\pm 0.0614}$
	SPiCE	$0.5318_{\pm 0.0254}$	$0.4830_{\pm 0.0453}$	$0.2816_{\pm 0.0332}$	$0.2241_{\pm 0.0102}$	$0.3456_{\pm 0.0291}$	$0.1611_{\pm 0.0004}$	$0.1229_{\pm 0.0021}$	$0.5650_{\pm 0.0457}$	$0.8405_{\pm 0.0823}$
ViSNet [58]	Mean	$0.5593_{\pm 0.0392}$	$0.4927_{\pm 0.0451}$	$0.2862_{\pm 0.0347}$	$0.2811_{\pm 0.0163}$	$0.3970_{\pm 0.0461}$	$0.1874_{\pm 0.0036}$	$0.1469_{\pm 0.0022}$	$0.6035_{\pm 0.0623}$	$0.7447_{\pm 0.0376}$
	DeepSets	$0.5280_{\pm 0.0449}$	$0.4987_{\pm 0.0515}$	$0.2846_{\pm 0.0248}$	$0.3104_{\pm 0.0247}$	$0.4113_{\pm 0.0222}$	$0.1716_{\pm 0.0116}$	$0.1314_{\pm 0.0242}$	$0.6626_{\pm 0.0036}$	$0.4160_{\pm 0.0425}$
	Attention	$0.5593_{\pm 0.0455}$	$0.4988_{\pm 0.0355}$	$0.2944_{\pm 0.0108}$	$0.3755_{\pm 0.0129}$	$0.4195_{\pm 0.0336}$	$0.2384_{\pm 0.0172}$	$0.1394_{\pm 0.0044}$	$0.5262_{\pm 0.0472}$	$0.7158_{\pm 0.1312}$
	SPiCE	$0.5384_{\pm 0.0298}$	$0.4538_{\pm 0.0491}$	$0.2814_{\pm 0.0155}$	$0.2715_{\pm 0.0130}$	$0.3807_{\pm 0.0399}$	$0.1657_{\pm 0.0120}$	$0.1277_{\pm 0.0077}$	$0.6890_{\pm 0.0629}$	$0.7195_{\pm 0.0837}$

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Our main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitation in the Appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: We provide proof based on lemmas listed in Section 2.2.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide the full experiments setting only relying on open-source data and models in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: We provide detailed experiment details in Section 4.1 and Appendix.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide detailed experiment details in Section 4.1 and Appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: In Appendix.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: In Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss the impacts in the Appendix.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: [NA]

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: [NA]

    Guidelines:

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

13. **New assets**

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

    Answer: [NA]

    Justification: [NA]

    Guidelines:

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

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

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

Answer: [NA]

Justification: [NA]

Guidelines:

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

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

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

Answer: [NA]

Justification: [NA]

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.