Pose-Transformed Equivariant Network for 3D Point Trajectory Prediction

Ruixuan Yu
Shandong University, Weihai
Weihai, 264209, China
yuruixuan@sdu.edu.cn

Jian Sun
Xi'an Jiaotong University
Xi'an, 710049, China
jiansun@xjtu.edu.cn

Abstract

Predicting 3D point trajectory is a fundamental learning task which commonly should be equivariant under Euclidean transformation, e.g., SE(3). The existing equivariant models are commonly based on the group equivariant convolution, equivariant message passing, vector neuron, frame averaging, etc. In this paper, we propose a novel pose-transformed equivariant network, in which the points are firstly uniquely normalized and then transformed by the learned pose transformations, upon which the points after motion are predicted and aggregated. Under each transformed pose, we design the point position predictor consisting of multiple Pose-Transformed Points Prediction blocks, in which the global and local motions are estimated and aggregated. This framework can be proven to be equivariant to SE(3) transformation over 3D points. We evaluate the pose-transformed equivariant network on extensive datasets including human motion capture, molecular dynamics modeling and dynamics simulation. Extensive experimental comparisons demonstrated our SOTA performance compared with the existing equivariant networks for 3D point trajectory prediction.

1. Introduction

3D Point trajectory prediction aims at forecasting the future positions of the given 3D points, and the motion should be equivariant under the transformation of the input points. It is widely applied to the areas such as particle dynamics modeling or simulation [14, 20, 30], human motion capture [14, 33], human robot interaction [22, 32], etc.

Various models have been proposed to achieve $SE(3)$ equivariance for 3D point analysis. The equivariant GNN-based mothods [14, 15, 26, 33] established equivariant message passing by learning linear combinations of the input vectors, ensuring explicit equivariance for point rotation and translation. The high-order-type methods [10, 11, 16, 29] achieve equivariance by building convolution kernels based on equivariant high-order representation such as spherical harmonics or lie group. The vector-based methods [6, 17, 19, 28] generalize traditional network layers with scalar neurons into vector neurons or update scalar-valued feature using the norm of vector-valued features. The frame averaging methods [8, 25] build equivariant models by averaging over all the transformation group elements. These equivariant models have achieved promising results over point clouds. However, the equivariant network has introduced additional computational cost for ensuring equivariance, and sometimes limits the prediction accuracy.

Under an equivariant framework, how to design the specific network architecture is a challenging task. For point trajectory prediction, the GNN-type models [14, 15, 26, 33] are based on $SE(3)$-invariant relation reasoning to aggregate geometric features such as input point coordinates. In the high-order-type models [11, 29] and vector-based models [6, 17, 19, 28], they adaptively design fundamental layers such as convolution, non-linear and batch normalization layer, etc., and build equivariant point trajectory network based on these layers. For frame averaging methods [8, 25], they directly utilize the architectures of classical deep learning models [13] to build their network architecture. As a summary, how to design the equivariant learning framework and the corresponding network architecture are still challenging and deserving to be investigated.

In this paper, for equivariant 3D point trajectory prediction, we propose a novel pose-transformed equivariant framework. It firstly normalizes the point cloud pose using the unique pose-normalization. Then it learns to transform the pose-normalized point clouds, and estimate/aggregate the point predictions upon these transformed point clouds. For the specific network design, we propose to learn the decomposed global and local motion displacements using the proposed Global-Local Motion Estimation Layer. Based on these ideas, we design a deep network architecture, dubbed Pose-Transformed Equivariant Network (PT-EvNet), for 3D point trajectory prediction. We apply the proposed PT-EvNet for 3D point trajectory prediction, including human motion capture, molecular dynamics modeling and dynamics simulation. Experiments show that PT-EvNet outperforms the state-of-the-art methods on these tasks. We also conduct ablation studies to show the effectiveness of our equivariant framework and the network designs.

2. Related Works

In this work, we focus on $SE(3)$ equivariant model for 3D point trajectory prediction. Given 3D point cloud $\mathcal{P}$, the equivariant model Ψ should predict point cloud $\Psi(\mathcal{P})$ that satisfy $g(\Psi(\mathcal{P})) = \Psi(g(\mathcal{P})), \ \forall g \in SE(3)$. We briefly review the related works on equivariant models for 3D point analysis including point trajectory prediction.

Group equivariant convolution [4] generalizes the spatial convolution to group convolution under actions of discrete symmetric group elements. It is further generalized to tensor features in [24, 37]. Those approaches are theoretically sound, however suffers from rapidly growing computational cost with the increase of group cardinality.

The spherical harmonics or lie group representation [1, 10–12, 16, 29] is popular for building equivariant network. The spherical harmonics are equivariant to continuous group transformation and utilized to design basic network layers for the equivariant models [5, 9, 11, 29]. The method of LieConv [10, 16] maps point position into group element on which one can conduct equivariant convolution with kernel parameterized as a neural network. These networks have achieved remarkable success for equivariant data analysis tasks, but rely on the high-order transformations with higher computational cost.

The equivariant graph neural networks of EGNN [26], GMN [15] and EGHN [14] are designed based on equivariant message passing by weighted combination of input vectors, and they keep equivariant under $SE(3)$ transformation of the input vectors. GMN [15] relies on equivariant geometrical constrains within the data during the message passing progress, and EGHN [14] is based on equivariant pooling and unpooling operations on the graph neural network. These models have shown effectiveness for equivariant 3D data analysis. However, they learn features with a scalar-based embedding to keep the equivariance of node position and may be insufficient for expressing more complex geometric quantities like torsion force [7].

Vector Neuron [6] extends neurons from 1D scalars to 3D vectors, enables $SO(3)$ actions in the latent feature space, and provides a general framework for building equivariant neural layers. The similar scalar-vector transforms for equivariance can be found [18, 27]. These models achieve equivariant by weighted combination on input vectors, which maintain equivariance under $SE(3)$ transformation on the input vectors. However, different dimensions of the vectors are separated in their fully connected layer, prohibiting information flows between dimensions [23]. In addition, the vector-based layers may not well model the geometric relationships of points such as angles.

Frame Averaging [8, 25] achieves equivariance by aggregating prediction over the elements of a symmetry group computed by PCA. For N-dimensional data, the order of the group is 2^N, which is $2^3 = 8$ for 3D data. The frame averaging method relies on the predictions and averaging over these fixed transforms in the symmetry group.

Compared with frame averaging methods, we uniquely normalize the pose of a 3D point cloud instead of constructing a symmetry group of transformations. In network architecture, we learn to transform point clouds for prediction instead of using fixed transformations by symmetry group action. Our framework is proven to be $SE(3)$ equivariant, and extensive experiments demonstrate its superiority compared with the state-of-the-art methods, including the frame averaging method that uses 8 fixed pose-transformations.

3. Pose-Transformed Equivariant Network

In this section, we first introduce setting of equivariant trajectory prediction, and then present the equivariant framework followed by the equivariant network architecture.

Problem Setting. Following the common setting in [14, 15, 26], we are given the 3D point cloud $\mathcal{P} = \{X, V, H, E\}$, with point position $X \in \mathcal{R}^{N\times3}$, velocity $V \in \mathcal{R}^{N\times3}$ and $SE(3)$-invariant point attribute $H \in \mathcal{R}^{N\times d}$. The edge attribute (node distance or indicator for connection types as stick or hinge) $E = \{e_{ij}|i, j = 1, ..., N\}$ models point interactions between points. The goal is to predict the future point cloud $\mathcal{P}^*$ with point position $X^* \in \mathcal{R}^{N\times3}$ equivariant to $SE(3)$ transformation on $\mathcal{P}$. N denotes the number of points, and d is the dimension of point attribute such as particle charge or velocity norm. Note that $SE(3)$ transformation (including rotation and translation) is conducted on the point coordinates of the point cloud, and the same rotation is conducted on the velocity.

3.1. General Framework

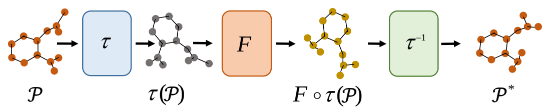

Figure 1. Illustration of the proposed equivariant framework Ψ.

For the given 3D point cloud $\mathcal{P}$, we first design a general equivariant framework as shown in Fig. 1. The 3D point cloud is firstly normalized by the transformation τ, and then we conduct trajectory prediction by the pose-transformed point position predictor F over the pose-normalized point cloud $\tau(\mathcal{P})$, and the output is the predicted point cloud after motion. Finally, the predicted point cloud $F \circ \tau(\mathcal{P})$ is transformed back to the pose of $\mathcal{P}$ by τ^{-1}. This framework is equivariant to $SE(3)$ transformation of $\mathcal{P}$, and the pre-

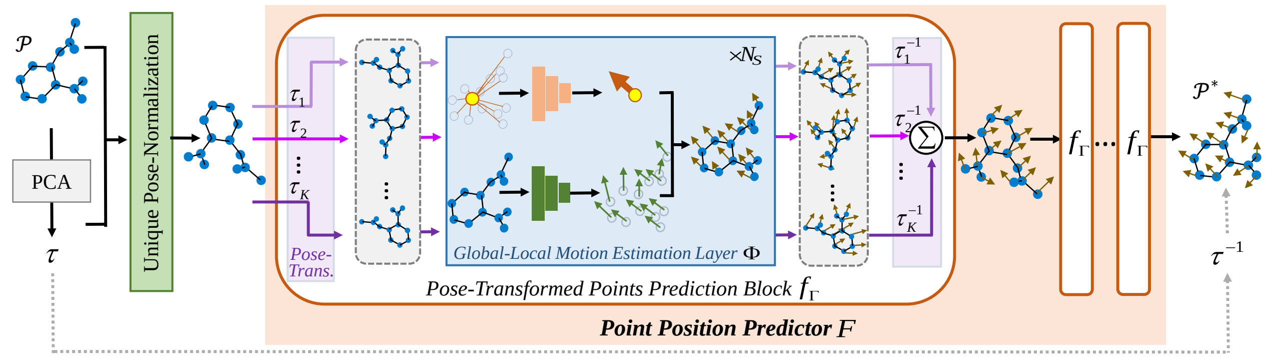

Figure 2. Pipeline of PT-EvNet. Point cloud $\mathcal{P}$ is firstly pose-normalized with τ which is uniquely computed with PCA. The trajectory is then predicted with Point Position Predictor F, which is a concatenation of Pose-Transformed Points Prediction Block f_Γ based on learnable pose transformations and iterative global-local motion estimation.

dicted point cloud $\mathcal{P}^*$ is

$$\mathcal{P}^* = \Psi(\mathcal{P}) = \tau^{-1} \circ F \circ \tau(\mathcal{P}). \tag{1}$$

We design the components τ and F of this framework, especially the dedicated design of F based on pose-transforms, presented in the following paragraphs and Sect. 3.2- 3.4.

Unique Pose-Normalization τ by PCA. For point cloud $\mathcal{P}$, we define the pose-normalization based on PCA. Specifically, we first centralize the point coordinates by subtracting the points center μ, then compute the SVD over the covariance matrix, and the rotation matrix U is the singular-vector matrix from the SVD of $(X-\mu)^\top(X-\mu)$. The pose-normalized point cloud is derived as

$$\tau(\mathcal{P}) = \{\overline{X}, \overline{V}, H, E\}, \tag{2}$$

with the normalized position and velocity as

$$\overline{X} = (X-\mu)U, \quad \overline{V} = VU. \tag{3}$$

One challenge for PCA based pose-normalization is the uncertainty on direction of singular vectors [8, 25, 31, 34, 35], i.e., the columns of rotation matrix U. Inverting the singular vector directions leads to different rotation matrix $U' = US$ where $S = \text{diag}\{\pm1, \pm1, \pm1\}$ is diagonal matrix with elements as ±1. We uniquely compute the rotation matrix U by taking the strategy in [36]. Specifically, we determine the direction of singular vectors by estimating the angles between each singular vector with a predefined anchor point, i.e., the farthest point from centroid. The direction should be flipped if the corresponding angle is larger than $90°$. With this strategy, we determine U by singular vectors with unique directions and compute the pose-normalized point cloud $\tau(\mathcal{P})$ by Eqns. (2-3). $\tau^{-1}(\cdot)$ is the inverse transform of τ, which conducts inverse rotation $U^\top$ for point velocity and position, and further translation μ on point position.

Point Position Predictor F. This operator takes the pose-normalized point cloud $\tau(\mathcal{P})$ as input and predicts the future point position after motion in the trajectory. This operator is defined as multi-steps architecture for progressively refining the estimated point cloud, and each step is denoted as an operator f_Γ that relies on learnable $SE(3)$ pose transformations Γ. The operator F is conceptually defined as

$$F \circ \tau(\mathcal{P}) \triangleq f_\Gamma \circ \cdots \circ f_\Gamma \circ \tau(\mathcal{P}). \tag{4}$$

where f_Γ predicts future point cloud, by taking current normalized point cloud as input. It estimates point motions from multiple poses Γ with a global-local strategy. We will present its details in Sect. 3.3.

The equivariance of $\mathcal{P}^*$ estimated by Eqn. (1) can be proven in the following theorem, please refer to the supplementary material for the proof.

Theorem 1. *With τ as Unique Pose-Normalization by PCA and F as Point Position Predictor, $\Psi(\mathcal{P})$ is equivariant under SE(3) transformation of $\mathcal{P}$.*

3.2. Overview of Network Architecture

Based on above equivariant framework, we design the ***Pose-Transformed Equivariant Network*** (PT-EvNet) by concretizing the structure of F and f_Γ.

As shown in Figure 2, given point cloud $\mathcal{P}$, PT-EvNet first conducts **Pose-Normalization** by τ which is uniquely computed by PCA as in Sect. 3.1. Then **Point Position Predictor** (F) is utilized to predict the future point cloud over pose-normalized point cloud $\tau(\mathcal{P})$. The predicted point cloud are finally transformed back to the pose of $\mathcal{P}$ with inverse transformation τ^{-1} as final prediction of PT-EvNet.

The point position predictor F is designed as the cascade of several **Pose-Transformed Points Prediction Block** (f_Γ). In every block f_Γ, it is fed with predicted point cloud of its previous block (the first f_Γ takes $\tau(\mathcal{P})$ as input). These blocks estimate point cloud after movement from multiple pose-transformed point clouds with **Global-Local Motion Estimation Layer** (Φ), which conducts motion estimation with a combined global-local strategy. In the following subsections, we will introduce them in detail.

3.3. Pose-Transformed Points Prediction Block (f_Γ)

Taking the unique pose-normalized point cloud $\overline{\mathcal{P}} = \tau(\mathcal{P})$ as input, Pose-Transformed Points Prediction Block f_Γ is designed to estimate the corresponding point cloud after movement. We accomplish this goal by predicting point-wise motion and attribute on pose-transformed point clouds from $\overline{\mathcal{P}}$, which help us to explore the feature from multiple poses and coordinates. We design f_Γ using two fundamental layers, i.e., the *Pose-Transformation* layer that maps normalized point cloud $\overline{\mathcal{P}}$ to N_K poses with learnable transformations $\Gamma = \{\tau_k; k = 1, \cdots N_K\}$ and the *Global-Local Motion Estimation Layer* Φ that estimates the point-wise motion and attribute for the pose-transformed point cloud.

Specifically, for the pose-normalized point cloud $\overline{\mathcal{P}}$ with point position, velocity, point attribute and edge attribute as $\{\overline{X}, \overline{V}, H, E\}$, it is firstly transformed into N_K poses using learnable Euclidean transformations of Γ in the pose-transformation layer, arriving at pose-transformed points

$$\tau_k(\overline{\mathcal{P}}) = \{X_k, V_k, H_k, E\}, \quad k = 1, ..., N_K, \tag{5}$$

with its elements as

$$X_k = \overline{X} U_k + \mu_k, \quad V_k = \overline{V} U_k, \quad H_k = H, \tag{6}$$

where $\mu_k \in \mathcal{R}^{3\times 1}$ is the learnable translation vector, $U_k \in \mathcal{R}^{3\times 3}$ is rotation matrix with the 6D rotation representation [38] from 6D trainable parameters. These pose-transformed point clouds are invariant under $SE(3)$ transformations of the network input $\mathcal{P}$.

Then the point position and attribute are estimated over all the pose-transformed point clouds with a shared Global-Local Motion Estimation Layer Φ (introduced in Sect. 3.4). It takes $\tau_k(\overline{\mathcal{P}})$ as input and outputs estimated point cloud $\Phi \circ \tau_k(\overline{\mathcal{P}})$ with updated point position X'_k and attribute H'_k

$$\Phi \circ \tau_k(\overline{\mathcal{P}}) = \{X'_k, V_k, H'_k, E\}. \tag{7}$$

Finally, the predicted point cloud $\Phi \circ \tau_k(\overline{\mathcal{P}})$ is transformed back with inverse transformation τ_k^{-1}. The predictions from all the pose-transformed point clouds are aggregated as prediction of f_Γ, i.e., the output of this block $f_\Gamma \circ \overline{\mathcal{P}} = \{X', \overline{V}, H', E\}$ is obtained by

$$f_\Gamma \circ \overline{\mathcal{P}} = \frac{1}{N_K} \sum_{k=1}^{N_K} \tau_k^{-1} \circ \Phi \circ \tau_k(\overline{\mathcal{P}}), \tag{8}$$

where the point position and attribute are aggregated by

$$X' = \frac{1}{N_K} \sum_{k=1}^{N_K} (X'_k - \mu_k) U_k^\top, H' = \frac{1}{N_K} \sum_{k=1}^{N_K} H'_k. \tag{9}$$

In the Pose-Transformed Points Prediction Block f_Γ, the point cloud is predicted from multiple transformed poses. The pose transformation with $\Gamma = \{\tau_k\}_{k=1}^{N_k}$ encourages Φ to explore rich positional information for point position prediction and attribute learning. Γ is adaptively learned in network driven by the training loss, providing flexibility for specific tasks. Note that both the point position and attribute are updated after this block, while the point velocity and edge attribute keep unchanged.

Pose-Transformed Points Prediction Block f_Γ is designed based on learnable pose transformations Γ in Eqn. (8). This is different to frame averaging [25] which conducts pose transformations using 8 elements of symmetry group computed from PCA. Experimental comparison is conducted in Sect. 4.4, demonstrating the superiority of our learnable pose-transformed strategy.

3.4. Global-Local Motion Estimation Layer (Φ)

We now introduce the Global-Local Motion Estimation layer Φ in Eqn. (8) for the point-wise position and attribute estimation on the pose-transformed point cloud. Inspired by global-local decomposition strategy of motion fields in physics and engineering, the point-wise motion is estimated with global and local displacements by three steps, i.e., *Decomposition* that separately represents global and local components, *Updating* that iteratively learns global and local displacement and attribute, *Combination* that combines learned global and local components together as prediction for Φ. An illustrative pipeline of the global-local motion estimation layer Φ is in Figure 2, which iteratively runs for N_S times to estimate point motions with global and local branches. The details of these three steps are as follows.

Decomposition. The input $\tau_k(\overline{\mathcal{P}}) = \{X_k, V_k, H_k, E\}$ is decomposed into global and local components in this step. We remove index k for brevity and denote point-wise position, velocity, attribute and edge attribute as x_i, v_i, h_i, e_{ij} with i, j as point index. The point position and velocity are decomposed into global and local components by

$$\begin{aligned} x^g &= \frac{1}{N} \sum_{i=1}^{N} x_i, \ v^g = \frac{1}{N} \sum_{i=1}^{N} v_i, \\ x_i^l &= x_i - x^g, \quad v_i^l = v_i - v^g, \end{aligned} \tag{10}$$

where superscripts g, l represent 'global' and 'local' respectively. x^g and v^g indicate the averaged position and velocity, while x_i^l and v_i^l reflect the relative geometric components. The local and global attributes are computed by

$$h_i^l = \rho([h_i, x_i^l, v_i^l, ||v_i^l||, ||v^g||]), \quad h^g = \text{MaxPool}\{h_i^l\}, \tag{11}$$

where 'MaxPool' is max-pooling operator by channel-wise maximization across points. $[\cdot]$ denotes concatenation operation, $||\cdot||$ denotes L_2-norm, and ρ is for point attribute learning that is designed as MLP.

Updating. Based on above decomposition, the global and local displacements Δx^g and Δx_i^l are iteratively updated in the global and local branches, with Δx^g, Δx_i^l initialized as zero vectors. In each of the N_S iterations, the global displacement is updated in the *global branch* based on global attribute h^g, i.e.,

$$\Delta x^g = \Delta x^g + \kappa(h^g), \tag{12}$$

where κ is designed as MLP for global displacement learning. In the *local branch*, the local displacement is updated based on local attribute and relationship by

$$\begin{aligned} \Delta x_i^l &= \Delta x_i^l + \beta(h_i^l) + \sum_{j \in \mathcal{N}(i)} \zeta(m_{ij}) \eta(x_i^l - x_j^l), \\ \textit{with} \quad m_{ij} &= \gamma([||x_i^l - x_j^l||, h_i^l, h_j^l, e_{ij}]), \end{aligned} \tag{13}$$

where $\beta, \zeta, \eta, \gamma$ are set as MLPs, $\mathcal{N}(i)$ denotes the neighbors of point x_i within two hops. The local and global point attributes are then updated by

$$h_i^l = \xi([h_i^l, \sum_{j \in \mathcal{N}(i)} m_{ij}]), \;\; h^g = \mathrm{MaxPool}\{h_i^l\}, \tag{14}$$

with ξ set as MLP. In this Updating step, with N_S times of iterations that sequentially conducts Eqns. (12-14), the global and local displacements Δx^g and Δx_i^l are derived, as well as the global and local point attribute h^g, h_i^l.
Combination. In this step, the global and local components after Updating step are combined to form the estimation of this layer. The estimated position $X' = \{x_i\}$ is obtained by combining global and local displacements $\Delta x^g, \Delta x_i^l$ with

$$x_i' = x_i + \Delta x^g + \Delta x_i^l. \tag{15}$$

The corresponding point attributes are taken as $H' = \{h_i'\}$ with $h_i' \triangleq h_i^l$.

In this global-local motion estimation layer, taking pose-transformed point cloud as input, we predict point position X' and attribute H' with above three steps. This layer is utilized as fundamental operation Φ in the pose-transformed points prediction block f_Γ as in Eqn. (8). The process of Φ is summarized as Algorithm 1 in supplementary material.

3.5. Details on Network Architecture and Training

PT-EvNet is designed based on the equivariant framework proposed in Sect. 3.1 and the basic operations proposed in Sect. 3.3-3.4. We design it using three pose-transformed points prediction blocks f_Γ in the point position predictor F. In each block, we set $N_K = 4$ in Eqn. (5), i.e., 4 pose transformations, to transform the pose-normalized point cloud, and we take shared transformations among the three blocks. In the global-local motion estimation layer Φ, we take $N_S = 4$ iterations to estimate global and local displacements. $\kappa, \beta, \zeta, \eta$ are set as two-layer MLPs with output as three-dimensional vector, and ξ, ρ, γ are set as two-layer MLPs to learn 64-dimensional attribute. PT-EvNet is trained by Adam optimizer with Mean Squared Error (MSE) loss between predicted point position and ground truth, and we set the batch-size to 12. Please refer to supplementary material for more details. Our code will be available at `https://github.com/yuruixuan/PT-EvNet`.

4. Experiments

We now evaluate our proposed PT-EvNet on multiple benchmark datasets for 3D point trajectory prediction, and conduct ablation studies to verify advantages of our designs.

4.1. Results on Motion Capture

Datasets. We conduct experiments on CMU Motion Capture Database [3] and evaluate different methods on Walking (Subject #35) and Running (Subject #9) subsets as in EGHN [14]. For the Walking subset, we take 200/600/600 frame pairs for training/validation/testing, and for Running subset, we take them as 200/240/240. For each pair of frames, the 3D coordinates corresponding to the body joints of the first frame are taken as input to estimate the coordinates of these points in the paired frame. The frame interval between each pair of frames is 30 in both scenarios. The performance is evaluated in MSE.
Implementation details. For our PT-EvNet, the same experiment setting is utilized as [14], and the input node attribute H is taken as L_2-norm of velocity, represented by the difference of positions in neighboring frames. The edge attribute e_{ij} in Eqn. (13) is set to 1 for edges connecting neighboring points, 2 for edges of points in two hops and 0 for others. All the results of the compared methods except EqMotion [33] and FA-GNN [25] are reported from [14], and we report the results of EqMotion and FA-GNN by running their codes.
Results and comparisons. The comparative results are presented in Table 1. We also present the results of PT-EvNet-B1 that uses only one pose-transformed points prediction block f_Γ in the point position predictor F. Our PT-EvNet achieves better performance than all the compared methods. Notably, PT-EvNet reduces the MSE by 1.0% and 1.6% on both datasets compared with the second best method. Even PT-EvNet-B1 with one pose-transformed points prediction block performs better than all these compared methods on the Walking subset.

In Figure 3, we show the trajectory estimation results by comparing with EGNN [26], EGHN [14], EqMotion [33] and FA-GNN [25]. Our PT-EvNet predicts the points coordinates more accurately.

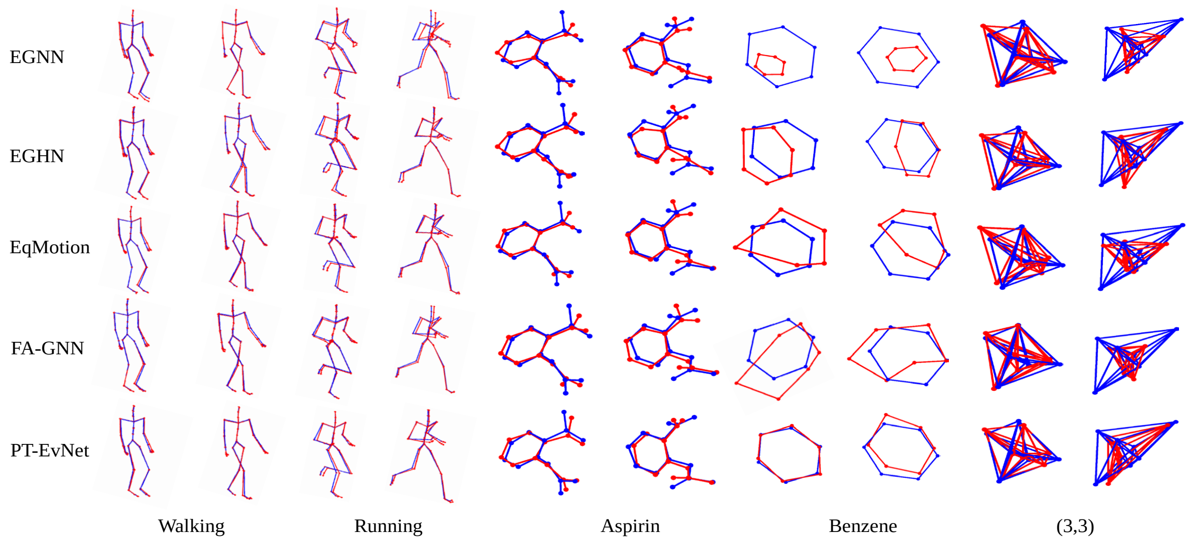

Figure 3. Visualization of trajectory estimation results on CMU (1-4 columns), MD17 (5-8 columns) and Simulation (9-10 columns) datasets. The ground truth are in blue, and the predictions are in red.

	Walking	Running
RF [21]	188.0	521.3
MPNN [13]	36.1	66.4
TFN[29]	32.0	56.6
SE(3)-Tr. [11]	31.5	61.2
EGNN [26]	28.7	50.9
GMN [15]	21.6	44.1
EGHN [14]	8.5	25.9
EqMotion* [33]	5.8	20.3
FA-GNN* [25]	7.2	37.3
PT-EvNet-B1	5.6	21.3
PT-EvNet	**4.8**	**18.7**

Table 1. Prediction error (MSE, $\times 10^{-2}$) on CMU dataset. * indicates that the results are reported by running their codes.

4.2. Results on Molecular Dynamics Modeling

Datasets. We conduct experiment for molecular trajectory prediction on MD17 dataset [2], which has trajectories of eight molecules. Taking current system state (particle positions) as input, we aim to predict the future particle positions. We take the same experiment setting as GMN [15], i.e., randomly split the dataset into training/validation/test set containing 500/2000/2000 frame pairs respectively, and taking 5000 frames as interval between frames pairs.

Implementation details. The initial node attribute H for our PT-EvNet is taken as concatenation of velocity L_2-norm and particle charge. The same edge attribute E is taken as GMN [15], i.e., the concatenation of point distance, charges of edge connected points, indication for stick or hinge, and indication for edge connected points within one or two hops. The results of EGHN [14], EqMotion [33] and FA-GNN [25] are reported by running their codes, and the results of the other compared methods are from [15].

Results and comparisons. Prediction errors are presented in Table 2. Our PT-EvNet surpasses most of the compared methods and obtains the best performance on six out of eight molecules. In particular, PT-EvNet achieves a substantial reduction in MSE from 48.12×10^{-2} to 26.25×10^{-2} on the Benzene subset. Figure 3 presents the predicted trajectory by PT-EvNet and the SOTA methods on Aspirin (5-6 columns) and Benzene (7-8 columns) subsets. Our model generates more reasonable predictions for both of the two molecules.

4.3. Results on Simulation Dataset

Datasets. We further conduct experiments on the Simulation Dataset [14], which includes single and multiple systems of particles. Given initial particles positions and velocities, the goal is to predict their future positions. All the experiments are conducted with the same setting as EGHN [14], i.e., the training/validation/test set contains 1000/2000/2000 frame pairs respectively, and we take 1500 frames as the interval between every frame pairs.

Implementation details. We follow EGHN [14] and assign node attribute as the L_2-norm of velocity. The edge attribute e_{ij} connecting point i and j is set as the concatenation of point charges product, point distance, and edge indicator for stick. Edge indicator is 1 if a stick exists and 0 otherwise. All the results of compared methods are reported from [14] except EqMotion [33] and FA-GNN [25], which are reported by running their official codes.

Results and comparisons. As shown in Table 3, PT-EvNet

	Aspirin	Benzene	Ethanol	Malonaldehyde	Naphthalene	Salicylic	Toluene	Uracil
RF [21]	10.94	103.72	4.64	13.93	0.50	1.23	10.93	0.64
TFN [29]	12.37	58.48	4.81	13.62	0.49	1.03	10.89	0.84
SE(3)-Tr. [11]	11.12	68.11	4.74	13.89	0.52	1.13	10.88	0.79
EGNN [26]	14.41	62.40	4.64	13.64	0.47	1.02	11.78	0.64
EGNNReg [26]	13.82	61.68	6.06	13.49	0.63	1.68	11.05	0.66
GMN [15]	10.14	48.12	4.83	13.11	**0.40**	0.91	10.22	**0.59**
GMN-L [15]	9.76	54.17	4.63	**12.82**	0.41	0.88	10.45	**0.59**
EGHN* [14]	11.68	59.34	5.60	13.74	0.43	0.90	21.52	0.62
EqMotion* [33]	13.14	66.70	5.82	13.75	0.63	0.93	20.31	0.67
FA-GNN* [25]	12.83	117.50	5.81	13.70	0.41	0.88	20.15	0.61
PT-EvNet-B1	7.64	26.87	**4.02**	13.73	0.42	0.92	9.67	0.63
PT-EvNet	**6.63**	**26.25**	4.34	13.22	**0.40**	**0.87**	**8.63**	0.62

Table 2. Prediction errors on MD17 dataset measured in MSE ($\times 10^{-2}$). * denotes the results obtained by running the official codes.

	Single System				Multiple System			
	(3,3)	(5,5)	(5,10)	(10,10)	(3,3)	(5,5)	(5,10)	(10,10)
Linear [14]	35.15	35.22	30.14	31.44	35.91	35.29	30.88	32.49
TFN [29]	25.11	29.35	26.01	OOM	27.33	29.01	25.57	OOM
SE(3)-Tr. [11]	27.12	28.87	24.48	OOM	28.14	28.66	25.00	OOM
MPNN [13]	16.00	17.55	16.15	15.91	16.76	17.58	16.55	16.05
RF [21]	14.20	18.37	17.08	18.57	15.17	18.55	17.24	19.34
EGNN [26]	12.69	15.37	15.12	14.64	13.33	15.48	15.29	15.02
EGHN [14]	11.58	14.42	14.29	13.09	12.80	14.85	14.50	**13.11**
EqMotion* [33]	13.52	16.28	16.61	14.57	14.91	16.88	15.81	13.38
FA-GNN* [25]	11.14	13.95	15.44	14.37	13.27	15.56	15.30	13.80
PT-EvNet-B1	11.08	13.38	14.76	14.62	12.78	14.67	14.98	13.59
PT-EvNet	**10.06**	**12.34**	**13.64**	**12.99**	**11.14**	**14.44**	**14.45**	14.25

Table 3. Prediction errors on Simulation dataset measured in MSE ($\times 10^{-2}$). The 'Multiple System' contains 5 different systems, and (A,B) indicates the system containing A complexes of average size B. * denotes the results obtained by running the official codes.

achieves the best performance compared with other methods on all the single systems and the challenging multiple systems except the multiple system of $(10, 10)$. Even PT-EvNet-B1 with one pose-transformed points prediction block in the point position predictor generates competitive results. In the last two columns of Figure 3, we show the predictions by our PT-EvNet and compared methods on the single simulation dataset of (3,3), and PT-EvNet predicts the system trajectories more accurately.

4.4. Model Analysis

Effectiveness of the proposed equivariant framework. In our framework in Sect. 3, we conduct pose normalization using Unique Pose-Normalization by PCA, and then conduct pose transformations on the uniquely normalized pose with learnable pose transformations Γ to estimate point motions. We conduct ablation studies on multiple datasets to demonstrate the effectiveness of our framework with learned pose transformations. In comparison, we set 8 pose transformations in Γ of our PT-EvNet, denoted as PT. We further replace these transformations by symmetry group transformations computed by PCA [25] (denoted as FA) or random rotations (denoted as RT) that are randomly set for each instance in each training/validation/test step. Our framework using learned 8 transformations achieves the best performance in all datasets as shown in Table 4, demonstrating the effectiveness of our pose-transformed equivariant framework. Please refer to supplementary material for visualizing these learned transformations.

Effectiveness of global-local motion estimation layer Φ. We replace Φ in PT-EvNet with linear operation, MLP, and other basic point position prediction layers in SOTA methods, including layers used in MPNN [13], EGNN [26], GMN [15] and EGHN [14] (please refer to supplementary material for details). The results of these modified PT-EvNets are shown in Table 5. PT-EvNet with Φ achieves the best performance on both Walking and Aspirin datasets. PT-EvNet-local that predicts displacement with only local

Dataset	Subdataset	FA	RT	PT
CMU	Walking	4.83	10.04	4.82
	Running	19.51	55.18	18.67
MD17	Aspirin	7.13	11.45	6.58
	Benzene	72.88	48.47	26.33
	Ethanol	5.20	5.51	4.28
Simulation	(3,3,1)	11.57	10.60	10.11
	(5,5,1)	13.42	13.84	12.25
	(3,3,5)	12.93	12.16	11.32

Table 4. Comparison of our network using frame averaging (FA), random pose transformation (RT) and proposed pose transformation (PT) for equivaraint tasks on various datasets. The lower MSE results demonstrate effectiveness of our framework.

	Walking	Aspirin
*-linear	158.46	16.62
*-mlp	143.22	13.04
*-mpnn	34.42	7.56
*-egnn	7.26	9.04
*-gmn	9.42	10.87
*-eghn	11.09	8.36
*-local	6.89	7.27
PT-EvNet	**4.83**	**6.63**

Table 5. Prediction error (MSE, $\times 10^{-2}$) on Walking and Aspirin datasets. * denotes networks that replace our global-local motion estimation layer Φ with other layers.

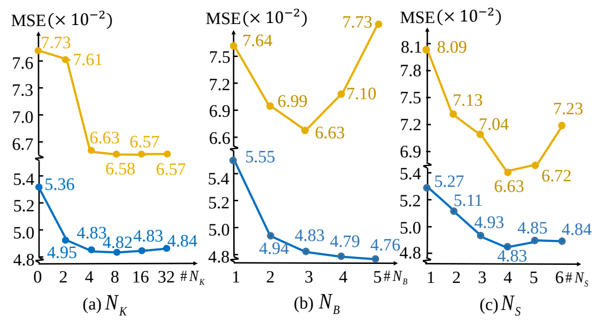

Figure 4. Ablation study on hyper-parameters in PT-EvNet on the Walking (blue lines) and Aspirin (yellow lines) datasets.

branch achieves competitive but higher MSE values, showing the necessity of both the global and local branches.

Effect of the number of pose transformation. Figure 4 (a) shows the results of PT-EvNet using different number (N_K) of pose transformations on Walking and Aspirin datasets. When $N_K = 0$, we directly predict point motion on the pose-normalized point cloud without transformations. PT-EvNet with pose transformation ($N_K > 0$) achieves better performance than network without it ($N_K = 0$). But with the increase of N_K, the performance becomes stable.

	Memory (MB)	FLOPs $\times 10^8$	Parameters $(\times 10^5)$	MSE $(\times 10^{-2})$
MPNN [13]	2479	35.85	2.05	36.1
RF [21]	983	0.02	0.02	188.0
TFN [29]	4395	47.10	4.40	32.0
SE(3)-Tr. [11]	7573	138.45	12.76	31.5
EGNN [26]	995	1.28	1.34	28.7
GMN [15]	997	1.39	2.02	21.6
EGHN [14]	1103	1.68	3.18	8.5
EqMotion [33]	1289	23.55	2.86	5.8
FA-GNN [25]	995	6.86	0.92	7.2
PT-EvNet	1243	8.18	4.88	4.8

Table 6. Computational cost of PT-EvNet and compared methods, with MSE reported on Walking dataset.

Effect of the number of pose-transformed points prediction block. We show the performance curves of PT-EvNet with different pose-transformed points prediction block number (N_B) in Figure 4 (b). As shown in the figure, setting three blocks, i.e., $N_B = 3$ is enough to produce a stable result, and even with one block, our network can achieve better results than state-of-the-art methods on Waling and Aspirin datasets.

Effect of the iteration number for displacement estimation. In the global-local motion estimation layer Φ, the displacements are estimated in N_S times. Figure 4 (c) show the performance with different N_S on Walking and Aspirin datasets. Results show that PT-EvNet with 4 times of displacement estimation performs best.

Computational cost. Table 6 presents the GPU memory, FLOPs and parameters of PT-EvNet and SOTA methods (please refer to supplementary material for details), as well as their MSE results on Walking dataset. Even with pose transformations and multiple MLPs in the network, our model exhibits tolerable/comparable computational costs, but achieves the overall best performance.

5. Conclusion

In this paper, we propose a pose-transformed equivariant network framework for 3D point trajectory prediction. The equivariance is achieved based on the unique pose-normalization and learnable pose-transformations, over which the global-local motion estimation is conducted. Experimental results show that the PT-EvNet outperforms the compared SOTA methods on benchmark datasets, and ablation studies justify the effectiveness of the network design. In the future work, we plan to further improve the network architecture and apply it for sequential trajectory prediction.

Acknowledgment This work was supported by NSFC (62306167, 12125104, U20B2075) and AI Super Fusion Private Cloud Module of Shandong University, Weihai.

References

[1] Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *ICML*, pages 992–1002, 2020. 2

[2] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5):e1603015, 2017. 6

[3] CMU. Carnegie-mellon motion capture database. 2003. 5

[4] Taco S. Cohen and Max Welling. Group equivariant convolutional networks. In *ICML*, pages 2990–2999, 2016. 2

[5] Taco S. Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016. 2

[6] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so(3)-equivariant networks. In *ICCV*, pages 12200–12209, 2021. 1, 2

[7] 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. 2

[8] Alexandre Agm Duval, Victor Schmidt, Alex Hernández-Garcıa, Santiago Miret, Fragkiskos D Malliaros, Yoshua Bengio, and David Rolnick. Faenet: Frame averaging equivariant gnn for materials modeling. In *ICML*, pages 9013–9033, 2023. 1, 2, 3

[9] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so(3) equivariant representations with spherical cnns. In *ECCV*, pages 52–68, 2018. 2

[10] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *ICML*, pages 3165–3176, 2020. 1, 2

[11] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *NIPS*, pages 1970–1981, 2020. 1, 2, 6, 7, 8

[12] Fabian B. Fuchs, Edward Wagstaff, Justas Dauparas, and Ingmar Posner. Iterative se(3)-transformers. In *Geometric Science of Information*, pages 585–595, 2021. 2

[13] 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. 1, 6, 7, 8

[14] Jiaqi Han, Wenbing Huang, Tingyang Xu, and Yu Rong. Equivariant graph hierarchy-based neural networks. In *NIPS*, 2022. 1, 2, 5, 6, 7, 8

[15] Wenbing Huang, Jiaqi Han, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Equivariant graph mechanics networks with constraints. In *ICLR*, 2022. 1, 2, 6, 7, 8

[16] Michael J Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. Lietransformer: Equivariant self-attention for lie groups. In *ICML*, pages 4533–4543, 2021. 1, 2

[17] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020. 1

[18] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *ICLR*, 2020. 2

[19] Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *ICML*, pages 15546–15566, 2023. 1

[20] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *ICML*, pages 2688–2697, 2018. 1

[21] Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019. 6, 7, 8

[22] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. In *CVPR*, pages 3595–3603, 2019. 1

[23] Shitong Luo, Jiahan Li, Jiaqi Guan, Yufeng Su, Chaoran Cheng, Jian Peng, and Jianzhu Ma. Equivariant point cloud analysis via learning orientations for message passing. In *CVPR*, pages 18932–18941, 2022. 2

[24] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *ICCV*, pages 5048–5057, 2017. 2

[25] Omri Puny, Matan Atzmon, Heli Ben-Hamu, Ishan Misra, Aditya Grover, Edward J. Smith, and Yaron Lipman. Frame averaging for invariant and equivariant network design. In *ICLR*, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[26] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *ICML*, pages 9323–9332, 2021. 1, 2, 5, 6, 7, 8

[27] Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *ICML*, pages 9377–9388, 2021. 2

[28] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B. Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipulation. In *International Conference on Robotics and Automation (ICRA)*, pages 6394–6400, 2022. 1

[29] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018. 1, 2, 6, 7, 8

[30] Fang Wu and Stan Z Li. Diffmd: a geometric diffusion model for molecular dynamics simulations. In *AAAI*, pages 5321–5329, 2023. 1

[31] Zelin Xiao, Hongxin Lin, Renjie Li, Lishuai Geng, Hongyang Chao, and Shengyong Ding. Endowing deep 3d

models with rotation invariance based on principal component analysis. In *IEEE International Conference on Multimedia and Expo*, pages 1–6, 2020. 3

[32] Chenxin Xu, Siheng Chen, Maosen Li, and Ya Zhang. Invariant teacher and equivariant student for unsupervised 3d human pose estimation. In *AAAI*, pages 3013–3021, 2021. 1

[33] Chenxin Xu, Robby T Tan, Yuhong Tan, Siheng Chen, Yu Guang Wang, Xinchao Wang, and Yanfeng Wang. Eqmotion: Equivariant multi-agent motion prediction with invariant interaction reasoning. In *CVPR*, pages 1410–1420, 2023. 1, 5, 6, 7, 8

[34] Ruixuan Yu, Xin Wei, Federico Tombari, and Jian Sun. Deep positional and relational feature learning for rotation-invariant point cloud analysis. In *ECCV*, pages 217–233, 2020. 3

[35] Zhiyuan Zhang, Binh-Son Hua, Wei Chen, Yibin Tian, and Sai-Kit Yeung. Global context aware convolutions for 3d point cloud understanding. In *International Conference on 3D Vision (3DV)*, pages 210–219, 2020. 3

[36] Chen Zhao, Jiaqi Yang, Xin Xiong, Angfan Zhu, Zhiguo Cao, and Xin Li. Rotation invariant point cloud analysis: Where local geometry meets global topology. *Pattern Recognition*, 127:108626, 2022. 3

[37] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. In *CVPR*, pages 519–528, 2017. 2

[38] Yi Zhou, Connelly Barnes, Lu Jingwan, Yang Jimei, and Li Hao. On the continuity of rotation representations in neural networks. In *CVPR*, 2019. 4