Equivariant Point Cloud Analysis via Learning Orientations for Message Passing

Shitong Luo[*1], Jiahan Li[*1,2], Jiaqi Guan[*3]
Yufeng Su[3], Chaoran Cheng[3], Jian Peng[3], Jianzhu Ma[2]
[1] HeliXon Research
[2] Peking University
[3] University of Illinois Urbana-Champaign
luost26@gmail.com,majianzhu@pku.edu.cn

Abstract

Equivariance has been a long-standing concern in various fields ranging from computer vision to physical modeling. Most previous methods struggle with generality, simplicity, and expressiveness — some are designed ad hoc for specific data types, some are too complex to be accessible, and some sacrifice flexible transformations. In this work, we propose a novel and simple framework to achieve equivariance for point cloud analysis based on the message passing (graph neural network) scheme. We find the equivariant property could be obtained by introducing an orientation for each point to decouple the relative position for each point from the global pose of the entire point cloud. Therefore, we extend current message passing networks with a module that learns orientations for each point. Before aggregating information from the neighbors of a point, the networks transforms the neighbors' coordinates based on the point's learned orientations. We provide formal proofs to show the equivariance of the proposed framework. Empirically, we demonstrate that our proposed method is competitive on both point cloud analysis and physical modeling tasks. Code is available at https://github.com/luost26/Equivariant-OrientedMP.

1. Introduction

3D point cloud has become a prevalent data structure for representing a wide range of 3D objects such as 3D scenes [1, 4, 8, 36], molecules [12, 15, 28], and physical particles [7, 31]. To capture the geometric relationship among points, message passing networks (graph neural networks) [9] and their variants such as DGCNN [33] and PointNet++ [24] have become standard tools to model 3D point cloud data. A characteristic of 3D point cloud is that most of the scalar features of the point cloud are *invariant*[1] to global rotation and translation, and the vector features of point clouds are *equivariant* to global rotation and translation. For instance, the category of a shape and its part semantics are not affected by their poses in the 3D space, and the normal vector of each point rotates together with the shape. Therefore, designing a new type of message passing network that preserves invariance and equivariance for point clouds is a critical challenge with broad impacts on both computer vision and other science domains.

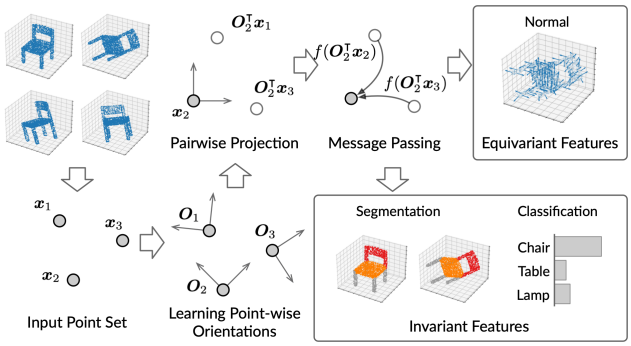

Figure 1. An illustration of the proposed method. It first learns orientations for each point in the point cloud. By projecting the relative coordinate of neighbor points before information aggregation, the model successfully decouples global rotation and achieves equivariance.

The key module of conventional message passing networks for modeling point clouds is to construct the message from point j to i using the difference between 3D coordinates $\boldsymbol{x}_j - \boldsymbol{x}_i$ which is usually transformed by non-equivariant layers such as regular MLPs. Hence, it is clear that the outputs of these models are not equivariant to the rotation of point clouds.

Recently, two notable lines of methods have been proposed to address the equivariance problem: tensor field-based neural networks [7, 31] and vector-based neural net-

[*] equal contribution.

[1] Note that invariance is a special case of equivariance.

works [5, 10, 27, 29]. Tensor field-based networks operate in the 3D space using continuous convolutions and the filters in the model are constrained to be the composite of a learnable radial function and a spherical harmonic. Though achieving equivariance, they suffer from high space and time complexity because a significant amount of the spherical harmonics need to be calculated on the fly. The complicated formulation of tensor field networks also makes them less accessible to the community. Vector-based neural networks are a good alternative to tensor field networks. The core part of vector neural networks is their linear layer which takes a list of vectors in 3D space as input and outputs multiple linear combinations of the input vectors. The output vectors are equivariant to input rotations by the nature of the linear combination. However, as the fully connected layer of vector networks linearly combines input vectors, different dimensions of the vectors are separated, which prohibits flexible vector transformations and hinders information flows between dimensions. In addition, vector-based neural networks are inadequate to model essential geometric relationships such as angles due to the linearity of the vector propagation.

To address this challenge, we introduce a novel equivariant message passing model by learning the orientation of each point during the message passing process. The overall learning process includes three key steps. First, it learns the orientation matrix for each point. Second, we project the relative position vector between these two points to the learned orientations, which decouples the relative position of two points from global rotations. Next, the projected relative position difference is integrated into neural messages by some specific network architecture. In the end, we can get both invariant and equivariant properties based on the output features of message passing layers along with the learned orientations.

In comparison to previous equivariant models, our method is much simpler but more flexible. Specifically, we can easily extend our model by applying arbitrary transforms to the projected relative position vector to compose the message, while vector-based networks only allow the linear combination of vectors.

To summarize, our main contributions include: (1) We propose a new framework for equivariant point cloud analysis and provide formal formulation and proof of the equivariance. (2) We show the framework's generality by using it to augment classical point cloud networks including DGCNN and RS-CNN. (3) We conduct extensive experiments on various tasks including point cloud classification, segmentation, normal estimation, and many-body modeling, and demonstrate that the proposed method achieves competitive performance.

2. Related Work

Message Passing Networks for Point Clouds Message passing neural networks (also known as graph neural networks) [9] have become a standard architecture to approach machine learning tasks on point clouds. Representative models include PointNet++ [24], DGCNN [33], RS-CNN [21], PointCNN [20, 35], and so on [6]. For each point in the point cloud, a message passing layer is first applied to aggregate information from features and relative positions of its neighbor points. The aggregated information is transformed to point representation by using standard neural networks such as MLPs. By definition, directly using MLPs to process coordinates is not equivariant to rotation and one can observe that a rotation of the 3D objects might yield different predictions for these models. Recently, there has been a growing interest in designing new deep learning architectures which can preserve equivariant and invariant properties.

Equivariance via Orientation Estimation Estimating canonical orientations for objects is a way to decouple the global rotation and achieve equivariance. PointNet [23] constructs a sub-network based on standard MLPs to estimate a canonical orientation in the form of a 3x3 matrix for each input shape. However, the matrix is not restricted to be an orthogonal rotation matrix and it is not equivariant to the input either. PointNet can only achieve approximate equivariance by data augmentation. GC-Conv [38] uses principal component analysis (PCA) to define canonical orientations at different scales but its orientations rely on handcrafted prior knowledge which is not available for many real-world applications. LGR-Net [42] relies on normal vectors as additional inputs, which could be viewed as a special form of orientation for each point. Yet, in most settings, normal vectors are hard to acquire or even not defined. Quaternion equivariant capsule network [43] uses quaternions to represents point-wise orientations. However, it requires initial orientations as input, while our methods learn orientations in a fully end-to-end manner. Other works [3, 19, 26, 30, 32] classified to this class mainly focus on pose estimation and most of them require the supervision by the ground-truth canonical pose and are only equivariant by data augmentation.

Equivariance via Vector-Based Networks Recently, vector-based neural networks have emerged as a new approach to achieve equivariance [5, 10, 27, 29]. They generalize scalar activations in standard neural networks to vector activations. To process these vector activations, vector-based linear layer is designed to take a list of vectors as input and output multiple linear combinations of the input vectors, which makes the entire network equivariant by na-

ture. For the activation functions, GVP [10] scale the vectors based on their norm. Consequently, vectors passed through such activation layers remain a linear combination of the input vectors. VNN [5] introduces a non-linear layer that outputs two sets of vectors and projects one to another. Since the two sets of vectors are both equivariant to inputs, the projected vectors are also equivariant. The major limitation of vector-based networks arises from linear combinations — as their fully connected layer linearly combines input vectors. This prohibits flexible vector transformations and hinders information flows among vector dimensions.

Other Equivariant Networks Tensor field-based networks [7, 22, 31] are another thread of works which could achieve equivariance. They rely on constraining the convolutional kernel to the spherical harmonics family. By restricting the space of learnable functions, features convolved with these kernels are provably equivariant to rotations and translations. However, it takes much space to cache the spherical harmonics calculated for irregular point clouds, leading to high space and time complexity. Concurrently, many works take efforts to design invariant operators [2, 11, 16, 17, 25, 28, 34], based on rotationally invariant measures such as distances and angles.

3. Preliminaries

3.1. Notations

We denote a point cloud as $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, where each point is represented as a 3D vector $\boldsymbol{x}_i \in \mathbb{R}^3$. The orientation of point i is defined as a 3x3 rotation matrices $\boldsymbol{O}_i$. The optimized orientation matrices should satisfy orthogonality constraint that $\boldsymbol{O}_i^\intercal = \boldsymbol{O}_i^{-1}$ and has unit determinant $(\det \boldsymbol{O}_i = 1)$[2].

3.2. Equivariance

Let $\mathbb{G}$ denote a transformation group (*e.g.* rotation group, permutation group, *etc.*), $g \in \mathbb{G}$ denote a specific transform in the group, and $\mathcal{T}_g(x) : \mathbb{X} \to \mathbb{X}$ denote the function that applies transform g to the the input $x \in \mathbb{X}$. A function $f : \mathbb{X} \to \mathbb{Y}$ is equivariant to $\mathbb{G}$ if there exists a transform function on the output domain $\mathcal{S}_g(y) : \mathbb{Y} \to \mathbb{Y}$ such that for all input $x \in \mathbb{X}$, the following equation holds:

$$f\left(\mathcal{T}_g\left(x\right)\right) = \mathcal{S}_g\left(f\left(x\right)\right). \tag{1}$$

In point cloud analysis, we consider the following types of equivariance:

- **Rotational and translational invariance**: The shape category of point clouds and point-wise semantic labels fall into this type of equivariance. Specifically, $\mathbb{G}$ is the group of rigid transform (rotation and translation) whose elements can be represented by a rotation matrix and a translation vector: $(\boldsymbol{R}, \boldsymbol{t})$. The transform function on the input domain is defined as $\mathcal{T}_{(\boldsymbol{R},\boldsymbol{t})}(\boldsymbol{X}) = (\boldsymbol{R}\boldsymbol{x}_1 + \boldsymbol{t}, \ldots, \boldsymbol{R}\boldsymbol{x}_N + \boldsymbol{t})$. The transform function on the output domain (the domain of shape categories and semantic labels) is simply an identity transform: $\mathcal{S}_{(\boldsymbol{R},\boldsymbol{t})}(\boldsymbol{c}) = \boldsymbol{c}$. In other words, a point cloud $\boldsymbol{X}$'s category or semantic labels $\boldsymbol{c}$ keeps unchanged regardless of the rotations and translations applied to $\boldsymbol{X}$.
- **Rotational equivariance and translational invariance**: Point-wise normal vectors and velocities of physical particles fall into this class. In particular, $\mathbb{G}$ is also the group of rigid transform. $\mathcal{T}_{(\boldsymbol{R},\boldsymbol{t})}$ is defined as $\mathcal{T}_{(\boldsymbol{R},\boldsymbol{t})}(\boldsymbol{X}) = (\boldsymbol{R}\boldsymbol{x}_1 + \boldsymbol{t}, \ldots, \boldsymbol{R}\boldsymbol{x}_N + \boldsymbol{t})$, and $\mathcal{S}_{(\boldsymbol{R},\boldsymbol{t})}$ is defined as $\mathcal{S}_{(\boldsymbol{R},\boldsymbol{t})}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N) = (\boldsymbol{R}\boldsymbol{v}_1, \ldots, \boldsymbol{R}\boldsymbol{v}_N)$. Here, $\boldsymbol{v}_i \in \mathbb{R}^3$ can represent the point-wise normal vector or velocity in the N-Body system modeling problem.
- **Permutation equivariance and invariance**: These types of equivariance are not among the interest of this work because most message passing networks are already permutation equivariant for point-wise features and invariant for global features. They can be trivially combined with the geometric equivariance discussed above. We state them formally here for clarity: $\mathcal{G}$ is the permutation group, and a permutation is denoted as a bijective mapping σ on $\{1, \ldots, N\}$. The transform function is $\mathcal{T}_\sigma(\boldsymbol{X}) = (\boldsymbol{x}_{\sigma(1)}, \ldots, \boldsymbol{x}_{\sigma(N)})$. For point-wise features, the outputs are equivariant to permutation: $\mathcal{S}_\sigma^{\text{eqv}}(\boldsymbol{Y}) = (\boldsymbol{y}_{\sigma(1)}, \ldots, \boldsymbol{y}_{\sigma(N)})$. For global features, the outputs are invariant: $\mathcal{S}_\sigma^{\text{inv}}(\boldsymbol{y}) = \boldsymbol{y}$.

Based on these definitions, it is clear that invariance is a special case of equivariance. In this paper, we use both of the terms "equivariance" and "invariance" to distinguish cases, but sometimes we also use "equivariance" as a general concept that implies both equivariance and invariance.

3.3. Intuition: Inherent Orientations of Points

One of the *key intuitions* behind our method is that: each point in a point cloud has an inherent orientation when considering its context, although an individual point itself is fully symmetric. As shown in 4, the seat of the chair has at least two inherent orientations: "UP" and "FRONT" — "UP" is the orientation to which the seat faces and "FRONT" is the direction opposite to the back of the chair. If the chair is represented as a point cloud, then points on the seat inherit these orientations, and we can denote the orientation using a rotation matrix $\boldsymbol{O}_i$ where we assume the first and second column vectors represent "UP" and "FRONT". These rotation matrices are naturally equivari-

[2]We can always choose the right-handed orientation so that the determinant is 1 instead of -1.

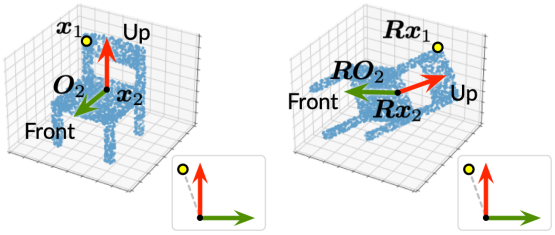

Figure 2. An illustration of the concept "inherent orientation". The coordinate of the yellow dot is $\boldsymbol{x}_1$, and the coordinate of the black dot is $\boldsymbol{x}_2$. The arrows are two basis vectors of the black dot's inherent orientation denoted as $\boldsymbol{O}_2$. The yellow dot's position relative to $\boldsymbol{x}_2$ and $\boldsymbol{O}_2$ keeps unchanged despite the global rotation $\boldsymbol{R}$.

ant to the global rotation of the chair. Assuming point i is on the seat and point j is any other point in the point cloud, then $\boldsymbol{O}_i^\intercal(\boldsymbol{x}_j - \boldsymbol{x}_i)$ essentially converts the difference of two points' global coordinates to the relative position of j to i in terms of the inherent orientation of i. The first component of $\boldsymbol{O}_i^\intercal(\boldsymbol{x}_j - \boldsymbol{x}_i)$ indicates the height of point j relative to the seat and the second component represents how far is the point in front of the chair. Therefore, $\boldsymbol{O}_i^\intercal(\boldsymbol{x}_j - \boldsymbol{x}_i)$ is invariant to the pose of the chair and can go through arbitrary transforms without breaking invariance. This concept is illustrated in Figure 2. Notice that no matter how the chair rotates, the relative coordinate of the yellow circle with respect to the FRONT-UP orientation remains unchanged. However, in most real-world applications, we have no prior knowledge about the orientation for each point. Therefore, it is necessary to design a network that can learn orientations for each point in an end-to-end fashion.

4. Method

The central idea of this work is to learn orientations for message passing. This section elaborates on each building block, organized as follows:

- In Section 4.1, we detail the principle of orientation learning and propose a neural network architecture to learn orientations.
- In Section 4.2, we show how the designed architecture could be applied to the learned orientations to message passing networks and how it preserves equivariance.
- In Section 4.3, we formulate the readout layers for estimating both equivariant and invariant properties.

4.1. Learning Orientations

As introduced in Section 3.3, the orientation of a point is inherent and contextual. Point-wise orientations can be represented as orthogonal orientation matrices $\boldsymbol{O}_i \in \mathbb{R}^{3\times3}$ $(i = 1, \ldots, N)$. Therefore, in order to learn effective point-wise orientations, there should be a sub-network that (1) captures the context of each point, (2) outputs orthogonal orientation matrices, and (3) is equivariant to the global pose. To fulfill objective (1) and (3), we employ a variant of GVP-GCN [10, 27] (a vector-based graph neural network) over the k-nearest-neighbor graph of the point cloud to estimate two equivariant vector features for each point. Subsequently, we leverage on the Gram-Schmidt process to construct orientation matrices from the vectors [44], which fulfills objective (2). Below is the description of the network for learning orientations:

Let $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ denote the input point cloud. The point cloud, with all zeros as its initial scalar and vector features, is passed to the first vector-based graph convolution layer denoted as V-GConv:

$$(\boldsymbol{H}^{(1)}, \boldsymbol{V}^{(1)}) \leftarrow \text{V-GConv}_1(\boldsymbol{X}, \boldsymbol{0}, \boldsymbol{0}). \tag{2}$$

Here, $\boldsymbol{H}^{(1)} = (\boldsymbol{h}_1^{(1)}, \ldots, \boldsymbol{h}_N^{(1)})$ is the scalar features of the point cloud, and $\boldsymbol{V}^{(1)} = \left[(\boldsymbol{v}_{11}^{(1)}, \boldsymbol{v}_{12}^{(1)}, \ldots), \ldots, (\boldsymbol{v}_{N1}^{(1)}, \boldsymbol{v}_{N2}^{(1)}, \ldots)\right]$ denote its vector features. Next, the scalar and vector features are iteratively updated through a stack of V-GConv layers:

$$(\boldsymbol{H}^{(\ell)}, \boldsymbol{V}^{(\ell)}) \leftarrow \text{V-GConv}_\ell(\boldsymbol{X}, \boldsymbol{H}^{(\ell-1)}, \boldsymbol{V}^{(\ell-1)}). \tag{3}$$

The final layer outputs two vectors for each point denoted as $\boldsymbol{V}^{(L)} = \left[(\boldsymbol{v}_{11}^{(L)}, \boldsymbol{v}_{12}^{(L)}), \ldots, (\boldsymbol{v}_{N1}^{(L)}, \boldsymbol{v}_{N2}^{(L)})\right]$. Details about V-GConv layers are presented in the supplementary material.

We then employ Gram-Schmidt process to normalize and orthogonalize the two vectors for each point:

$$\boldsymbol{u}_{i1} \leftarrow \frac{\boldsymbol{v}_{i1}^{(L)}}{\|\boldsymbol{v}_{i1}^{(L)}\|}, \tag{4}$$

$$\boldsymbol{u}'_{i2} \leftarrow \boldsymbol{v}_{i2}^{(L)} - \langle \boldsymbol{v}_{i2}^{(L)}, \boldsymbol{u}_{i1} \rangle \boldsymbol{u}_{i1}, \tag{5}$$

$$\boldsymbol{u}_{i2} \leftarrow \frac{\boldsymbol{u}'_{i2}}{\|\boldsymbol{u}'_{i2}\|}, \tag{6}$$

Here $\langle \cdot, \cdot \rangle$ is the inner product of two vectors. Finally, we construct point-wise orientation matrices by combining the two orthogonal unit vectors $(\boldsymbol{u}_{i1}, \boldsymbol{u}_{i2})$ and their cross product:

$$\boldsymbol{O}_i \leftarrow [\boldsymbol{u}_{i1}, \boldsymbol{u}_{i2}, \boldsymbol{u}_{i1} \times \boldsymbol{u}_{i2}] \quad (i = 1, \ldots, N). \tag{7}$$

Orientation matrices constructed in this way fulfill the three requirements summarized at the beginning of this section. In particular, the V-GConv layers enable the learned orientations to sense their contexts and guarantee equivariance. The Gram-Schmidt orthogonalization and Eq.7 ensure the orthogonality of learned orientations. The equivariance of learned orientations is stated formally in the following proposition:

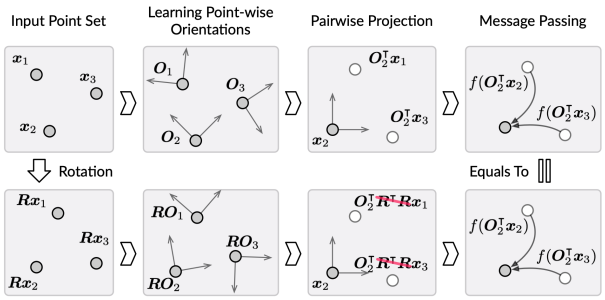

Figure 3. An illustration of message passing with orientations and its equivariant nature. Note that a proper rotation matrix $\boldsymbol{R}$ should satisfy the constraint $\boldsymbol{R}^\intercal \boldsymbol{R} = \boldsymbol{I}$.

Proposition 1. *The proposed network for learning orientations, denoted as* $f_{ort}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = (\boldsymbol{O}_1, \ldots, \boldsymbol{O}_N)$*, is equivariant to rotation and invariant to translation, satisfying* $f_{ort}\left(\boldsymbol{R}\boldsymbol{x}_1 + \boldsymbol{t}, \ldots, \boldsymbol{R}\boldsymbol{x}_N + \boldsymbol{t}\right) = (\boldsymbol{R}\boldsymbol{O}_1, \ldots, \boldsymbol{R}\boldsymbol{O}_N)$.

Proof. See the supplementary material. □

Extension: Learning Global Orientations The network for learning point-wise orientations can be adapted for learning global orientations. We first construct a hierarchy of the point cloud: $\boldsymbol{X}^{(L)} = \{\bar{\boldsymbol{x}}\} \subset \boldsymbol{X}^{(L-1)} \subseteq \ldots \subseteq \boldsymbol{X}^{(1)} \subseteq \boldsymbol{X}$, where $\bar{\boldsymbol{x}} = \frac{1}{N}\sum \boldsymbol{x}_i$. The ℓ-th V-GConv layer ($\ell < L$) finds the k-nearest-neighbors of a point $\boldsymbol{x}_i^{(\ell)}$ in $\boldsymbol{X}^{(\ell-1)}$ to aggregate information. The scalar and vector features of $\bar{\boldsymbol{x}}$ is obtained via global pooling over $\boldsymbol{X}^{(L-1)}$'s features. Finally, the orientation constructed from $\bar{\boldsymbol{x}}$'s vectors feature serves as the global orientation $\boldsymbol{O}_{\text{glob}}$. As the geometric center $\bar{\boldsymbol{x}}$ is equivariant, it is straightforward to see that $\boldsymbol{O}_{\text{glob}}$ is invariant to translation and equivariant to rotation.

4.2. Equivariant Message Passing with Learned Orientations

Typical message passing layers for point clouds use coordinate differences $\boldsymbol{x}_j - \boldsymbol{x}_i$ to compose the message passed from one point to another. They can be formulated in the following general form:

$$\boldsymbol{h}_i^{(\ell+1)} \leftarrow \underset{j \in \mathcal{N}(i)}{\text{Agg}}\ H\left(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_j^{(\ell)}, \boldsymbol{x}_j - \boldsymbol{x}_i\right), \tag{8}$$

where Agg is a permutation-invariant aggregation operator (common choices include max, sum, and mean). $H(\cdot)$ is an arbitrary neural network such as an MLP. To achieve invariance, we project $\boldsymbol{x}_j - \boldsymbol{x}_i$ using the learned orientation $\boldsymbol{O}_i$ and rewrite the formula as:

$$\boldsymbol{h}_i^{(\ell+1)} \leftarrow \underset{j \in \mathcal{N}(i)}{\text{Agg}}\ H\left(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_j^{(\ell)}, \boldsymbol{O}_i^\intercal(\boldsymbol{x}_j - \boldsymbol{x}_i)\right). \tag{9}$$

By applying $\boldsymbol{O}_i^\intercal$ to $\boldsymbol{x}_j - \boldsymbol{x}_i$, we decouple the coordinate differences to the pose of the point cloud — the rotated relative position $\boldsymbol{O}_i^\intercal(\boldsymbol{x}_j - \boldsymbol{x}_i)$ is invariant to global rotations of the point cloud. Figure 3 illustrates this message passing scheme. Formally, we have:

Proposition 2. *Under the assumption that* $(\boldsymbol{O}_1, \ldots, \boldsymbol{O}_N)$ *comes from an equivariant function* g *satisfying* $g(\boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}) = (\boldsymbol{R}\boldsymbol{O}_1, \ldots, \boldsymbol{R}\boldsymbol{O}_N)$*, and that* $\boldsymbol{H}$ *comes from an invariant function* h *satisfying* $h(\boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}) = \boldsymbol{H}$*, the message passing formula in Eq.9, denoted as* $f_{mp}^{(\ell)}$*, is invariant to rotation and translation and satisfies* $f_{mp}^{(\ell)}(\boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}) = f_{mp}^{(\ell)}(\boldsymbol{X})$.

Proof. See the supplementary material. □

Notice that the output feature $\boldsymbol{h}_i^{(\ell+1)}$ is invariant to rotations despite the form of H. This allows arbitrary flexible network design for H, while previous work imposes strong prior assumptions on H such as the linear combination of vectors, handcrafted invariant measures, or spherical harmonics filtering. In addition, projecting $\boldsymbol{x}_j - \boldsymbol{x}_i$ using $\boldsymbol{O}_i^\intercal$ involves the inner product between $\boldsymbol{x}_j - \boldsymbol{x}_i$ and $\boldsymbol{O}_i^\intercal$'s column vectors which explicitly encodes angular geometric information.

Augmented Message Passing Networks In this section, we demonstrate how to endow classical point-based message passing networks with equivariance by using RS-CNN and DGCNN as examples.

RS-CNN. The message passing formula of vanilla RS-CNN is:

$$\boldsymbol{r}_{ij} \leftarrow \boldsymbol{x}_j - \boldsymbol{x}_i, \tag{10}$$

$$\boldsymbol{h}_i^{(\ell+1)} \leftarrow \max_{j \in \mathcal{N}(i)} \left(M\left(\|\boldsymbol{r}_{ij}\|, \boldsymbol{r}_{ij}\right) \odot \boldsymbol{h}_j^{(\ell)}\right), \tag{11}$$

where max denotes the max pooling operation, M is an MLP network, and $\odot$ denotes element-wise multiplication. We can rewrite RS-CNN by incorporating Eq.7 as the follow:

$$\boldsymbol{h}_i^{(\ell+1)} \leftarrow \max_{j \in \mathcal{N}(i)} \left(M\left(\|\boldsymbol{O}_i^\intercal \boldsymbol{r}_{ij}\|, \boldsymbol{O}_i^\intercal \boldsymbol{r}_{ij}\right) \odot \boldsymbol{h}_j^{(\ell)}\right), \tag{12}$$

In vanilla RS-CNN, the initial point-wise features $\boldsymbol{h}_i^{(0)}$ are set to the points' 3D coordinates, which would break equivariance in our setting. As a remedy, we can either initialize $\boldsymbol{h}_i^{(0)}$ with a constant or coordinates aligned to the global orientation $\boldsymbol{O}_{\text{glob}}$.

DGCNN. The formulations of the first and subsequent layers of vanilla DGCNN are:

$$\boldsymbol{h}_i^{(1)} \leftarrow \max_{j \in \mathcal{N}(\boldsymbol{x}_i)} \sigma\left(M_0(\boldsymbol{x}_j - \boldsymbol{x}_i, \boldsymbol{x}_i)\right), \tag{13}$$

$$\boldsymbol{h}_i^{(\ell+1)} \leftarrow \max_{j \in \mathcal{N}(\boldsymbol{h}_i^{(\ell)})} \sigma\left(M_\ell(\boldsymbol{h}_j^{(\ell)} - \boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_i^{(\ell)})\right), \tag{14}$$

where σ is the activation function and M_ℓ is a MLP layer. To obtain the equivariant property, we modify Eq.13 to:

$$\boldsymbol{h}_i^{(1)} \leftarrow \max_{j \in \mathcal{N}(\boldsymbol{x}_i)} \sigma\left(M_0(\boldsymbol{O}_i^\intercal(\boldsymbol{x}_j - \boldsymbol{x}_i))\right). \tag{15}$$

Here, we ignore $\boldsymbol{x}_i$ from the message passing formula, but we can still apply $\boldsymbol{O}_{\text{glob}}^\intercal$ to $\boldsymbol{x}_i$ to decouple the point's coordinate from global rotation.

The above formulations are kept simple for clarity, we introduce the following extensions that may further increase the power of the network:

Extension 1: Multiple Orientations The relative coordinates can be projected using multiple orientation matrices. Differently projected relative coordinates can be concatenated to feed to the message network $H(\cdot)$.

Extension 2: Incorporating Global Information In some cases, global spatial information can be useful. We can project global coordinates using $\boldsymbol{O}_{\text{glob}}$ to obtain global invariant canonical coordinates and use them as point-wise invariant features.

4.3. Estimating Equivariant and Invariant Properties

So far, we have introduced how to extract equivariant point-wise features in previous sections. The last step is to predict properties based on the learned features. We divide properties into two classes: invariant properties and equivariant properties. For example, invariant properties include the category of a point cloud, point-wise semantics, and so on. They are independent of the global pose of the point cloud and therefore belong to the first case in Section 3.2. Equivariant properties such as normal vectors and velocities rotate together with the global rotation, and therefore they belong to the second case discussed in Section 3.2.

Invariant Property Estimation Point-wise features output by the proposed message passing layers are already invariant according to Propos.2. Therefore, to predict point-wise invariant property, we can directly feed the features into an arbitrary neural network (commonly an MLP) to estimate the property. For global properties, we can perform pooling operations over all the features. The pooled global features are clearly invariant and we can safely feed it to a neural network to estimate global invariant properties.

Equivariant Property Estimation The learned point-wise features are invariant to rotation. Hence, in order to predict equivariant properties, we leverage on the learned orientations again. Specifically, we first use an MLP to transform point-wise features to 3D vectors. Then, we apply the orientations to the vectors to obtain the equivariant vector properties. These two steps can be formulated as:

$$\boldsymbol{p}_i \leftarrow \text{MLP}\left(\boldsymbol{h}_i^{(L)}\right), \tag{16}$$

$$\boldsymbol{e}_i \leftarrow \boldsymbol{O}_i \boldsymbol{p}_i, \tag{17}$$

where $\boldsymbol{e}_i$ is the desired equivariant for point i. The equivariance of $\boldsymbol{e}_i$ is stated formally in the following proposition:

Proposition 3. *Under the assumption that* $(\boldsymbol{O}_1, \dots, \boldsymbol{O}_N)$ *comes from an equivariant function* g *satisfying* $g(\boldsymbol{RX} + \boldsymbol{t}) = (\boldsymbol{RO}_1, \dots, \boldsymbol{RO}_N)$*, and that* $\boldsymbol{H}^{(L)}$ *comes from an invariant function* h *satisfying* $h(\boldsymbol{RX} + \boldsymbol{t}) = \boldsymbol{H}^{(L)}$*, The property estimator* f_{prop} *defined by Eq.16 and Eq.17 is equivariant to rotation and invariant to translation, satisfying* $f_{prop}(\boldsymbol{RX} + \boldsymbol{t}) = \boldsymbol{R} f_{prop}(\boldsymbol{X})$.

Proof. See the supplementary material. □

By combining proposition 1, 2, and 3, we can easily see that message passing networks in the proposed framework are equivariant.

5. Experiments

We evaluate our method on three types of tasks including point cloud analysis (Section 5.1), physical modeling (Section 5.2), and analytical study (Section 5.3). In the point cloud analysis, the shape classification task and part segmentation task examine the method's capability of predicting global and point-wise invariant properties. The normal estimation task evaluates the model's ability on estimating equivariant properties. The task of physical modeling is to predict particle velocities in an n-body system, which heavily relies on the model's equivariance to achieve accurate predictions. In the end, we provide more insight into the behavior of our model by visualizing learned orientations of different 3D objects.

5.1. Point Cloud Analysis

5.1.1 Classification (Invariant)

A point cloud's category is its *invariant* property. This task can demonstrate the model's capability of learning invariant representations.

Setup We use the ModelNet40 [36] dataset and its official train-test split for the point cloud classification task. The dataset consists of 12,311 shapes from 40 different categories, of which 2,468 shapes are left out for test. Each point cloud contains 1,024 points.

Following the setup in the previous work [5, 18, 39], we adopt three different train-test rotation settings: (1) z/z: both training and test point clouds are rotated around the gravitational axis; (2) $z/\text{SO}(3)$: training point clouds are rotated around the gravitational axis and test point clouds

Table 1. Point cloud segmentation results in the IoU (*Inter-over-Union*). In z/SO(3) setting, our model outperforms other baselines on 11 out of 16 shapes.

z/SO(3)	plane	bag	cap	car	chair	earph.	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate	table
PointNet [23]	40.4	48.1	46.3	24.5	45.1	39.4	29.2	42.6	52.7	36.7	21.2	55.0	29.7	26.6	32.1	35.8
PointNet++ [24]	51.3	66.0	50.8	25.2	66.7	27.7	29.7	65.6	59.7	70.1	17.2	67.3	49.9	23.4	43.8	57.6
PointCNN [20]	21.8	52.0	52.1	23.6	29.4	18.2	40.7	36.9	51.1	33.1	18.9	48.0	23.0	27.7	38.6	39.9
DGCNN [33]	37.0	50.2	38.5	24.1	43.9	32.3	23.7	48.6	54.8	28.7	17.8	74.4	25.2	24.1	43.1	32.3
ShellNet [41]	55.8	59.4	49.6	26.5	40.3	51.2	53.8	52.8	59.2	41.8	28.9	71.4	37.9	49.1	40.9	37.3
RI-Conv [40]	80.6	80.0	70.8	68.8	86.8	70.3	87.3	84.7	77.8	80.6	57.4	91.2	71.5	52.3	66.5	78.4
GC-Conv [39]	80.9	**82.6**	81.0	70.2	88.4	70.6	87.1	**87.2**	**81.8**	78.9	58.7	91.0	77.9	52.3	66.8	80.3
RI-Fwk. [18]	81.4	82.3	**86.3**	75.3	88.5	72.8	90.3	82.1	81.3	81.9	67.5	92.6	75.5	54.8	75.1	78.9
Ours	**81.7**	79.0	85.0	**78.1**	**89.7**	**76.5**	**91.6**	85.9	81.6	**82.1**	**67.6**	**95.0**	**79.6**	**64.4**	**76.9**	**80.7**

Table 2. Consine distance between the predicted and ground truth normals. Our model outperforms all the baselines by a clear margin. In addition, compared to non-equivariant baselines, our model shows consistent performance thanks to equivariance.

Methods	z/z	z/SO(3)	SO(3)/SO(3)
PointNet++ [24]	0.34	0.81	0.55
RS-CNN [21]	0.26	0.83	0.50
DGCNN [33]	0.29	0.32	0.22
RI-Conv [40]	1.33	1.30	1.30
GC-Conv [39]	0.42	0.44	0.42
Ours	**0.20**	**0.20**	**0.20**

Table 3. Classification accuracy on ModelNet40. The upper rows are non-equivariant models, and the lower rows are equivariant models. The performance of our model is on par with equivariant baselines.

Methods	z/z	z/SO(3)	SO(3)/SO(3)
PointNet [23]	85.9	19.6	74.7
PointNet++ [24]	91.8	28.4	85.0
RS-CNN [21]	90.3	48.7	82.6
DGCNN [33]	90.3	33.8	88.6
RI-Conv [40]	86.5	86.4	86.4
GC-Conv [39]	89.0	89.1	89.2
Ours-RSCNN	87.6	87.6	87.5
Ours-DGCNN	88.4	88.4	88.9

are rotated arbitrarily. This setting examines the model's quality of equivariance-by-construction. (3) SO(3)/SO(3): both training and test point clouds are rotated arbitrarily.

Results We compare our model with both equivariant and non-equivariant baselines. Table 3 shows the test classification accuracy of different methods. Our model achieves competitive performance compared to other baseline models.

5.1.2 Part Segmentation (Invariant)

Point-wise labels are also an *invariant* property of a point cloud. This task can examine the model's performance on modeling invariant detail properties.

Setup We evaluate the models on the ShapeNet [37] dataset which consists of 16 categories with 16,881 shapes, and are labeled in 50 parts in total. Following the convention, we sample 2,048 points for each shape. We use two train-set rotation settings for this task: z/SO(3) and SO(3)/SO(3). Our model employs the equivariant adaptation DGCNN (see Section 4.2) as the backbone. We calculate the IoU (Inter-over-Union) metric to measure the segmentation quality for each category.

Results Table 1 summarizes the quantitative comparisons with baseline methods in z/SO(3) setting. The results of SO(3)/SO(3) can be found in the supplementary material. Our model outperforms all the baseline models on 12 out of 16 categories in z/SO(3) setting and 10 of 16 categories in SO(3)/SO(3) setting. This improvement demonstrates that our equivariant model is effective in learning fine-grained details.

5.1.3 Normal Estimation (Equivariant)

Normal vectors are *equivariant* property of a point cloud as they rotate together with the global rotation of the point cloud. This task tests the models' capabilities of modeling equivariant properties.

Setup We validate our model and other baselines using ModelNet40 [36], where each point in the point cloud is labeled with a 3D normal vector. We use the same data split and rotation settings as the ones used in the classification task. Our model uses the equivariant adaptation of DGCNN (see Section 4.1 for detail) as the backbone network. Hyper-parameters and other implementation details are put to supplementary materials.

Results Table 2 shows the test cosine-distance errors ($1 - \cos\langle \boldsymbol{n}_{\text{true}}, \boldsymbol{n}_{\text{pred}} \rangle$) [39] of all the methods to be compared on

Table 4. Mean squared errors (MSE) of predicted future coordinates in N-body system modeling task.

Methods	10 Particles	20 Particles
Linear(vel) [27]	0.339	0.408
GNN [14]	0.183	0.256
SE(3) Tr. [7]	0.351	0.371
TFN [31]	0.236	0.273
EGNN [27]	0.126	0.212
Ours	0.103	0.206

the normal estimation benchmark. It is obvious that the performance of other non-equivariant models degrades when we apply more flexible rotation on the point clouds. Thanks to the equivariance, our model presents consistency of performance on different settings of train-test rotation, and outperforms both non-equivariant and equivariant baselines by a clear margin. This result reveals the inherent generalizability of our model with respect to arbitrary poses and confirms the benefit of using equivariant models to estimate equivariant properties.

5.2. N-Body System Modeling

Modeling a dynamic particle system is a fundamental yet challenging task in many other research areas. It is a also prototype task of many other specific tasks in various domains, such as molecule simulation. Though controlled by simple physical rules, N-body dynamic systems exhibit complex dynamics and it is expensive to predict their future states solely based on simulation. Therefore, the goal of this task is to predict each particle's future states (position and velocity) based on its current states.

Setup We test our model and baselines on two systems: 10-particle system and 20-particle system. For each system, we run the simulation program provided by [7] to collect particle trajectories. We collected 3,000 trajectories for training and 2,000 for test. Each trajectory has 1,000 timesteps. The target is to predict the coordinates of all the particles after 1,000 timesteps, given their current positions, velocities, and types (positively charged or negatively charged). We use the mean squared error (MSE) of the predicted future positions as the evaluation metric. Hyperparameters and other implementation details of our model are put to supplementary materials. Our baselines include a simple linear model introduced in [27], a non-equivariant graph neural network model [13], as well as equivariant models including tensor field networks [31], SE(3) transforms [7], and EGNN [27].

Results As shown in Table 2, our model exhibits competitive performance, suggesting its power in modeling physical interactions among points.

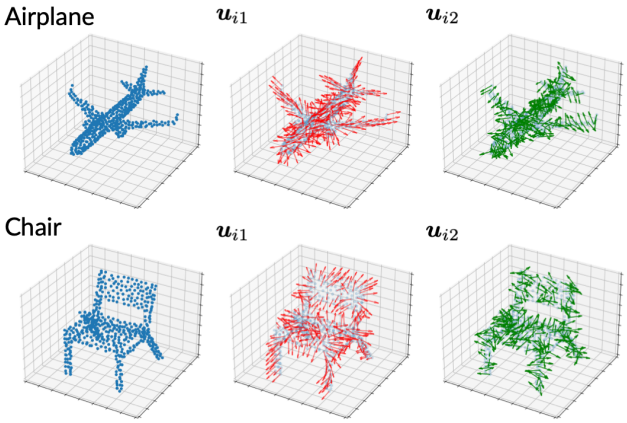

Figure 4. *Left*: Input point clouds. *Middle & Right*: The first and second column vectors of learned orientations in the classification model.

5.3. Visualizing Orientations

We visualize two examples of learned orientations from the point cloud classification model in Figure 4. We find that the learned orientations are correlated with the global structure. For example, the first orientation vectors (red arrows) on the airplane's wings and tails head to the direction to which the wings and tails stretch and the second orientation vectors (green arrows) are perpendicular to the wings. In addition, the first orientation vectors (red) on the chair's both front legs consistently head to the front direction, indicating the learned orientations' awareness of structural similarities.

6. Conclusions and Limitations

In this paper, we introduce a scheme for equivariant point set analysis. The core ingredient is to learn point-wise orientations and project neighbor coordinates using the learned orientation when aggregating information. By extensive experiments, we demonstrate our model's effectiveness and generality.

However, the proposed method requires an additional sub-network to estimate point-wise orientations. This is a non-negligible overhead and it is important to reduce the cost of orientation learning. Another concern of the method is its robustness. It is unclear to what extent the learned orientations are vulnerable to noisy data, which is inevitable in real-world settings. It is necessary to study the robustness in future work. Lastly, this paper only demonstrates few applications, it would be valuable to examine the method in other more challenging tasks such as molecular modeling.

Acknowledgement

This work was supported by National Key R&D Program of China No. 2021YFF1201600.

References

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016. 1

[2] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. Se (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *arXiv preprint arXiv:2101.03164*, 2021. 3

[3] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014. 2

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1

[5] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas Guibas. Vector neurons: A general framework for so (3)-equivariant networks. *arXiv preprint arXiv:2104.12229*, 2021. 2, 3, 6

[6] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 195–205. Computer Vision Foundation / IEEE Computer Society, 2018. 2

[7] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 3, 8

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1

[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017. 1, 2

[10] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael John Lamarre Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2020. 2, 3, 4, 11

[11] Mor Joseph-Rivlin, Alon Zvirin, and Ron Kimmel. Momen (e) t: Flavor the moments in learning to classify shapes. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019. 3

[12] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. 1

[13] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR, 2018. 8

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 8

[15] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2019. 1

[16] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020. 3

[17] Feiran Li, Kent Fujiwara, Fumio Okura, and Yasuyuki Matsushita. A closer look at rotation-invariant deep point cloud analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16218–16227, 2021. 3

[18] Xianzhi Li, Ruihui Li, Guangyong Chen, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. A rotation-invariant framework for deep point cloud analysis. *IEEE Transactions on Visualization and Computer Graphics*, 2021. 6, 7, 13

[19] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3706–3715, 2020. 2

[20] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018. 2, 7, 13

[21] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019. 2, 7

[22] Adrien Poulenard and Leonidas J Guibas. A functional approach to rotation equivariant non-linearities for tensor field networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13174–13183, 2021. 3

[23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2, 7, 13

[24] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017. 1, 2, 7, 13

[25] Zhuoran Qiao, Anders S Christensen, Matthew Welborn, Frederick R Manby, Anima Anandkumar, and Thomas F Miller III. Unite: Unitary n-body tensor equivariant network with applications to quantum chemistry. *arXiv preprint arXiv:2105.14655*, 2021. 3

[26] Helge Rhodin, Mathieu Salzmann, and Pascal Fua. Unsupervised geometry-aware representation for 3d human pose

estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 750–767, 2018. 2

[27] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021. 2, 4, 8

[28] Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018. 1, 3

[29] Kristof T Schütt, Oliver T Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. *arXiv preprint arXiv:2102.03150*, 2021. 2

[30] Weiwei Sun, Andrea Tagliasacchi, Boyang Deng, Sara Sabour, Soroosh Yazdani, Geoffrey Hinton, and Kwang Moo Yi. Canonical capsules: Unsupervised capsules in canonical pose. *arXiv preprint arXiv:2012.04718*, 2020. 2

[31] 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, 3, 8

[32] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. 2

[33] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 1, 2, 7, 13

[34] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, 2018. 3

[35] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 2

[36] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 1, 6, 7

[37] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016. 7

[38] Zhiyuan Zhang, Binh-Son Hua, Wei Chen, Yibin Tian, and Sai-Kit Yeung. Global context aware convolutions for 3d point cloud understanding. In *2020 International Conference on 3D Vision (3DV)*, pages 210–219. IEEE, 2020. 2

[39] Zhiyuan Zhang, Binh-Son Hua, Wei Chen, Yibin Tian, and Sai-Kit Yeung. Global context aware convolutions for 3d point cloud understanding. In *2020 International Conference on 3D Vision (3DV)*, pages 210–219. IEEE, 2020. 6, 7, 13

[40] Zhiyuan Zhang, Binh-Son Hua, David W Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International Conference on 3D Vision (3DV)*, pages 204–213. IEEE, 2019. 7, 13

[41] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1607–1616, 2019. 7, 13

[42] Chen Zhao, Jiaqi Yang, Xin Xiong, Angfan Zhu, Zhiguo Cao, and Xin Li. Rotation invariant point cloud classification: Where local geometry meets global topology. *arXiv preprint arXiv:1911.00195*, 2019. 2

[43] Yongheng Zhao, Tolga Birdal, Jan Eric Lenssen, Emanuele Menegatti, Leonidas Guibas, and Federico Tombari. Quaternion equivariant capsule networks for 3d point clouds. In *European Conference on Computer Vision*, pages 1–19. Springer, 2020. 2

[44] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 4

Supplementary Material

Equivariant Point Cloud Analysis via Learning Orientations for Message Passing

A. Orientation Learning Networks

This section elaborates on the neural network architecture for learning orientations described in Section 4.1.

The building block of the network is GVP-based graph convolutional networks (GVP-GCN) [10]. To understand GVP-GCN, we first briefly introduce GVP (geometric vector perceptron). GVP is a generalization of the traditional perceptron (a linear layer followed by a non-linear function). While traditional perceptrons take only an array of scalar features as input and outputs another array of scalar values, GVP takes an array of scalars and an array of 3D vectors as input, and output another array of scalars and array of vectors:

$$(\boldsymbol{s}', \boldsymbol{V}') \leftarrow \mathrm{GVP}(\boldsymbol{s}, \boldsymbol{V}), \quad \text{where } \boldsymbol{s} = [s_1, \dots, s_n] \in \mathbb{R}^n \text{ and } \boldsymbol{V} = [\boldsymbol{v}_i, \dots, \boldsymbol{v}_m] \in \mathbb{R}^{3\times m}. \tag{18}$$

The most relevant property of GVP is its equivariance to rotation:

$$(\boldsymbol{s}', \boldsymbol{R}\boldsymbol{V}') \leftarrow \mathrm{GVP}(\boldsymbol{s}, \boldsymbol{R}\boldsymbol{V}). \tag{19}$$

The above formulation and equivariance property is sufficient for building the rest part of our model. Therefore, if the reader is interested in more technical details and formal proofs about GVP, we refer the reader to [10].

Next, we introduce GVP-based graph convolutional networks (GVP-GCN). Let $\boldsymbol{h}_i$ denote the scalar features, $\boldsymbol{V}_i$ denote the vector features of point i, and $\boldsymbol{h}_{ij}$, $\boldsymbol{V}_{ij}$ denote the scalar and vector of edge ij. The message passing step of GVP-GCN is defined as:

$$(\boldsymbol{h}_{(i\leftarrow j)}, \boldsymbol{V}_{(i\leftarrow j)}) := \mathrm{GVP}\left(\mathrm{concat}(\boldsymbol{h}_i, \boldsymbol{h}_{ij}), \mathrm{concat}(\boldsymbol{V}_i, \boldsymbol{V}_{ij})\right), \tag{20}$$

$$(\boldsymbol{h}'_i, \boldsymbol{V}'_i) \leftarrow (\boldsymbol{h}_i, \boldsymbol{V}_i) + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} (\boldsymbol{h}_{(i\leftarrow j)}, \boldsymbol{V}_{(i\leftarrow j)}), \tag{21}$$

$$(\boldsymbol{h}^{\text{out}}_i, \boldsymbol{V}^{\text{out}}_i) \leftarrow (\boldsymbol{h}'_i, \boldsymbol{V}'_i) + \mathrm{GVP}(\boldsymbol{h}'_i, \boldsymbol{V}'_i). \tag{22}$$

For short, we denote the above message passing scheme as $(\boldsymbol{h}^{\text{out}}_i, \boldsymbol{V}^{\text{out}}_i) \leftarrow \text{GVP-GConv}(\boldsymbol{h}_i, \boldsymbol{h}_{ij}, \boldsymbol{V}_i, \boldsymbol{V}_{ij})$. As proven in [10], GVP-GConv is equivariant to rotation:

$$(\boldsymbol{h}^{\text{out}}_i, \boldsymbol{R}\boldsymbol{V}^{\text{out}}_i) \leftarrow \text{GVP-GConv}(\boldsymbol{h}_i, \boldsymbol{h}_{ij}, \boldsymbol{R}\boldsymbol{V}_i, \boldsymbol{R}\boldsymbol{V}_{ij}). \tag{23}$$

To learn orientations from point clouds, we modify GVP-GConv according to the characteristics of 3D point clouds. Specifically, we assign distance to the scalar feature of edges and coordinate difference to the vector feature of edges:

$$\boldsymbol{h}_{ij} = [\|\boldsymbol{x}_i - \boldsymbol{x}_j\|], \ \boldsymbol{V}_{ij} = [(\boldsymbol{x}_i - \boldsymbol{x}_j)]. \tag{24}$$

The message passing formula over the point cloud (denoted as V-GConv) is then formulated as:

$$(\boldsymbol{h}_{(i\leftarrow j)}, \boldsymbol{V}_{(i\leftarrow j)}) := \mathrm{GVP}\left(\mathrm{concat}(\boldsymbol{h}_i, [\|\boldsymbol{x}_i - \boldsymbol{x}_j\|]), \mathrm{concat}(\boldsymbol{V}_i, [(\boldsymbol{x}_i - \boldsymbol{x}_j)])\right), \tag{25}$$

$$(\boldsymbol{h}'_i, \boldsymbol{V}'_i) \leftarrow (\boldsymbol{h}_i, \boldsymbol{V}_i) + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} (\boldsymbol{h}_{(i\leftarrow j)}, \boldsymbol{V}_{(i\leftarrow j)}), \tag{26}$$

$$(\boldsymbol{h}^{\text{out}}_i, \boldsymbol{V}^{\text{out}}_i) \leftarrow (\boldsymbol{h}'_i, \boldsymbol{V}'_i) + \mathrm{GVP}(\boldsymbol{h}'_i, \boldsymbol{V}'_i). \tag{27}$$

The formula involve point-wise coordinates $\boldsymbol{X}$, scalar and vector features $\boldsymbol{H}$, $\boldsymbol{V}$, so we denote the layer as $(\boldsymbol{h}^{\text{out}}_i, \boldsymbol{V}^{\text{out}}_i) \leftarrow \text{V-GConv}(\boldsymbol{X}, \boldsymbol{h}_i, \boldsymbol{V}_i)$ for short. To ensure equivariance, we assign all zero as the initial scalar and vector features of the point cloud, as defined in Eq.2.

B. Proofs

In this section, we prove the three propositions in Section 4.

Proposition 1. *The proposed network for learning orientations, denoted as* $f_{ort}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_N) = (\boldsymbol{O}_1, \dots, \boldsymbol{O}_N)$*, is equivariant to rotation and invariant to translation, satisfying* $f_{ort}(\boldsymbol{R}\boldsymbol{x}_1 + \boldsymbol{t}, \dots, \boldsymbol{R}\boldsymbol{x}_N + \boldsymbol{t}) = (\boldsymbol{R}\boldsymbol{O}_1, \dots, \boldsymbol{R}\boldsymbol{O}_N)$.

Proof. First, we show the equivariance of the message passing formula defined by Eq.25-27. Let $\boldsymbol{R} \in \mathrm{SO}(3)$ denote an arbitrary proper rotation matrix and $\boldsymbol{t} \in \mathbb{R}^3$ denote an arbitrary vector in the 3D space.

According to the equivariance of GVP (Eq.19), Eq.25 is equivariant by:

$$\begin{aligned}
(\boldsymbol{h}_{(i\leftarrow j)}, \boldsymbol{R}\boldsymbol{V}_{(i\leftarrow j)}) &= \mathrm{GVP}\left(\mathrm{concat}(\boldsymbol{h}_i, [\|\boldsymbol{x}_i - \boldsymbol{x}_j\|]), \mathrm{concat}(\boldsymbol{R}\boldsymbol{V}_i, [\boldsymbol{R}(\boldsymbol{x}_i - \boldsymbol{x}_j)])\right) \\
&= \mathrm{GVP}\left(\mathrm{concat}(\boldsymbol{h}_i, [\|\boldsymbol{R}(\boldsymbol{x}_i - \boldsymbol{x}_j)\|]), \mathrm{concat}(\boldsymbol{R}\boldsymbol{V}_i, [\boldsymbol{R}(\boldsymbol{x}_i - \boldsymbol{x}_j)])\right) \\
&= \mathrm{GVP}\left(\mathrm{concat}(\boldsymbol{h}_i, [\|\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t} - \boldsymbol{R}\boldsymbol{x}_j - \boldsymbol{t}\|]), \mathrm{concat}(\boldsymbol{R}\boldsymbol{V}_i, [\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t} - \boldsymbol{R}\boldsymbol{x}_j - \boldsymbol{t}])\right) \\
&= \mathrm{GVP}\left(\mathrm{concat}(\boldsymbol{h}_i, [\|(\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t}) - (\boldsymbol{R}\boldsymbol{x}_j + \boldsymbol{t})\|]), \mathrm{concat}(\boldsymbol{R}\boldsymbol{V}_i, [(\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t}) - (\boldsymbol{R}\boldsymbol{x}_j + \boldsymbol{t})])\right)
\end{aligned}$$

Next, it is straightforward to see that Eq.26 is equivariant as it simply sums up vectors, and Eq.27 is equivariant by Eq.19. By chaining the equivariance of Eq.25-27, we can see a V-GConv layer is equivariant. That is, formally:

$$(\boldsymbol{h}_i^{\text{out}}, \boldsymbol{R}\boldsymbol{V}_i^{\text{out}}) \leftarrow \text{V-GConv}(\boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}, \boldsymbol{h}_i, \boldsymbol{R}\boldsymbol{V}_i).$$

In addition, it is easy to see that stacking multiple V-GConv layers results to an equivariant function:

$$(\boldsymbol{h}_i^{(L)}, \boldsymbol{R}\boldsymbol{V}_i^{(L)}) \leftarrow \text{V-GConv}_L(\boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}, \text{V-GConv}_{L-1}(\boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}, (\dots \text{V-GConv}_1(\boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}, \boldsymbol{h}_i^{(0)}, \boldsymbol{R}\boldsymbol{V}_i^{(0)}) \dots))).$$

Now, the problem is how to choose initial scalar and vector features $\boldsymbol{h}_i^{(0)}, \boldsymbol{V}_i^{(0)}$ for an input point cloud $\boldsymbol{X}$. From the above equation, we can see that the initial vector features $\boldsymbol{V}^{(0)}$ should be aligned to the rotation $\boldsymbol{R}$ in order to preserve equivariance. However, in most settings, the pose of the point cloud (parameterized by $\boldsymbol{R}$ and $\boldsymbol{t}$) is unknown. Therefore, we simply set $\boldsymbol{V}^{(0)} = \boldsymbol{0}$ which is irrelevant to the global pose[3]. As for the initial scalar features $\boldsymbol{H}^{(0)} = [\boldsymbol{h}_1^{(0)}, \dots, \boldsymbol{h}_N^{(0)}]$, they should be invariant to the global pose, so we also simply set them to $\boldsymbol{0}$. Setting both initial scalar and vector features to zero leads to the design of the first layer of the orientation learning network (Eq.2).

So far, we have shown that our designed stack of V-GConv layers is equivariant. Note that the final layer of the network outputs two vectors for each point. They are equivariant to rotation and invariant to translation according to the above results.

Next, we consider the equivariance of the Gram-Schmidt process for constructing orientation matrices (Eq.4-6). For Eq.4, we have:

$$\widetilde{\boldsymbol{u}}_{i1} := \frac{\boldsymbol{R}\boldsymbol{v}_{i1}^{(L)}}{\|\boldsymbol{R}\boldsymbol{v}_{i1}^{(L)}\|} = \frac{\boldsymbol{R}\boldsymbol{v}_{i1}^{(L)}}{\|\boldsymbol{v}_{i1}^{(L)}\|} = \boldsymbol{R}\boldsymbol{u}_{i1},$$

and for Eq.6, we have:

$$\widetilde{\boldsymbol{u}}_{i2} := \frac{\widetilde{\boldsymbol{u}}'_{i2}}{\|\widetilde{\boldsymbol{u}}'_{i2}\|} = \boldsymbol{R}\boldsymbol{v}_{i2}^{(L)} - \langle \boldsymbol{R}\boldsymbol{v}_{i2}^{(L)}, \boldsymbol{R}\boldsymbol{u}_{i1} \rangle \boldsymbol{R}\boldsymbol{u}_{i1} = \boldsymbol{R}\left[\boldsymbol{v}_{i2}^{(L)} - \langle \boldsymbol{v}_{i2}^{(L)}, \boldsymbol{u}_{i1} \rangle \boldsymbol{u}_{i1}\right] = \boldsymbol{R}\boldsymbol{u}_{i2}.$$

These prove the equivariance of Eq.4-6. For Eq.7, we show its equivariance by:

$$\widetilde{\boldsymbol{O}_i} := [\boldsymbol{R}\boldsymbol{u}_{i1}, \boldsymbol{R}\boldsymbol{u}_{i2}, \boldsymbol{R}\boldsymbol{u}_{i1} \times \boldsymbol{R}\boldsymbol{u}_{i2}] = \boldsymbol{R}[\boldsymbol{u}_{i1}, \boldsymbol{u}_{i2}, \boldsymbol{u}_{i1} \times \boldsymbol{u}_{i2}] = \boldsymbol{R}\boldsymbol{O}_i.$$

Finally, as the whole orientation learning network for is a composite of the V-GConv network and the Gram-Schmidt-based matrix construction process. Therefore, the network, denoted as f_{ort}, is equivariant to the rotation and invariant to translation, satisfying $f_{\text{ort}}(\boldsymbol{R}\boldsymbol{x}_1 + \boldsymbol{t}, \dots, \boldsymbol{R}\boldsymbol{x}_N + \boldsymbol{t}) = (\boldsymbol{R}\boldsymbol{O}_1, \dots, \boldsymbol{R}\boldsymbol{O}_N)$. □

Proposition 2. *Under the assumption that* $(\boldsymbol{O}_1, \dots, \boldsymbol{O}_N)$ *comes from an equivariant function* g *satisfying* $g(\boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}) = (\boldsymbol{R}\boldsymbol{O}_1, \dots, \boldsymbol{R}\boldsymbol{O}_N)$*, and that* $\boldsymbol{H}$ *comes from an invariant function* h *satisfying* $h(\boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}) = \boldsymbol{H}$*, the message passing formula in Eq.9, denoted as* $f_{mp}^{(\ell)}$*, is invariant to rotation and translation and satisfies* $f_{mp}^{(\ell)}(\boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}) = f_{mp}^{(\ell)}(\boldsymbol{X})$.

[3]There are some other choices for $\boldsymbol{V}^{(0)}$ such as local principle components of each point. These features align with the global rotation and can be used as initial vector features. In this work, we choose the simplest solution.

Proof.

$$\begin{aligned}\widetilde{\boldsymbol{h}}_i^{(\ell+1)} := f_{\text{mp}}^{(\ell+1)}\left(\boldsymbol{R}\boldsymbol{X}+\boldsymbol{t}\right) &= \underset{j\in\mathcal{N}(i)}{\text{Agg}}\, H\left(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_j^{(\ell)}, (\boldsymbol{R}\boldsymbol{O})_i^{\intercal}\left((\boldsymbol{R}\boldsymbol{x}_j+\boldsymbol{t})-(\boldsymbol{R}\boldsymbol{x}_i+\boldsymbol{t})\right)\right)\\ &= \underset{j\in\mathcal{N}(i)}{\text{Agg}}\, H\left(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_j^{(\ell)}, \boldsymbol{O}_i^{\intercal}\boldsymbol{R}^{\intercal}\boldsymbol{R}(\boldsymbol{x}_j-\boldsymbol{x}_i)\right)\\ &= \underset{j\in\mathcal{N}(i)}{\text{Agg}}\, H\left(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_j^{(\ell)}, \boldsymbol{O}_i^{\intercal}(\boldsymbol{x}_j-\boldsymbol{x}_i)\right)\\ &= f_{\text{mp}}^{(\ell+1)}\left(\boldsymbol{X}\right) = \boldsymbol{h}_i^{(\ell+1)}.\end{aligned}$$

Note that initial point-wise features $\boldsymbol{h}_i^{(0)}(i=1,\ldots,N)$ are all set to zero, which are invariant to $\boldsymbol{X}$. Hence the above equation hold. When $\ell>0$, by induction, $\boldsymbol{h}_i^{(\ell-1)}(i=1,\ldots,N)$ are invariant to $\boldsymbol{X}$, so the equations also hold.

□

Proposition 3. *Under the assumption that $(\boldsymbol{O}_1,\ldots,\boldsymbol{O}_N)$ comes from an equivariant function g satisfying $g(\boldsymbol{R}\boldsymbol{X}+\boldsymbol{t}) = (\boldsymbol{R}\boldsymbol{O}_1,\ldots,\boldsymbol{R}\boldsymbol{O}_N)$, and that $\boldsymbol{H}^{(L)}$ comes from an invariant function h satisfying $h(\boldsymbol{R}\boldsymbol{X}+\boldsymbol{t}) = \boldsymbol{H}^{(L)}$, The property estimator f_{prop} defined by Eq.16 and Eq.17 is equivariant to rotation and invariant to translation, satisfying $f_{prop}\left(\boldsymbol{R}\boldsymbol{X}+\boldsymbol{t}\right) = \boldsymbol{R}f_{prop}\left(\boldsymbol{X}\right)$.*

Proof.

$$\widetilde{\boldsymbol{e}}_i := f_{\text{prop}}\left(\boldsymbol{R}\boldsymbol{X}+\boldsymbol{t}\right) = \boldsymbol{R}\boldsymbol{O}_i\,\text{MLP}(\boldsymbol{h}_i^{(L)}) = \boldsymbol{R}f_{\text{prop}}\left(\boldsymbol{X}\right) = \boldsymbol{R}\boldsymbol{e}_i.$$

□

C. Additional Results

Point Cloud Part Segmentation: SO(3)/SO(3) The following table is supplementary to Table 1 in Section 5.1.2, showing the part segmentation performance of our model and baselines in SO(3)-train/SO(3)-test setting.

Table 5. Point cloud segmentation results in the IoU (*Inter-over-Union*). In SO(3)/SO(3) setting, our model outperforms other baselines on 10 out of 16 shapes.

SO(3)/SO(3)	plane	bag	cap	car	chair	earph.	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate	table
PointNet [23]	81.6	68.7	74.0	70.3	87.6	68.5	88.9	80.0	74.9	83.6	56.5	77.6	75.2	53.9	69.4	79.9
PointNet++ (MSG) [24]	79.5	71.6	**87.7**	70.7	88.8	64.9	88.8	78.1	79.2	94.9	54.3	92.0	76.4	50.3	68.4	81.0
PointCNN [20]	78.0	80.1	78.2	68.2	81.2	70.2	82.0	70.6	68.9	80.8	48.6	77.3	63.2	50.6	63.2	**82.0**
DGCNN [33]	77.7	71.8	77.7	55.2	87.3	68.7	88.7	85.5	81.8	81.3	36.2	86.0	77.3	51.6	65.3	80.2
ShellNet [41]	79.0	79.6	80.2	64.1	87.4	71.3	88.8	81.9	79.1	95.1	57.2	91.2	69.8	55.8	73.0	79.3
RI-Conv [40]	80.6	80.2	70.7	68.8	86.8	70.4	87.2	84.3	78.0	80.1	57.3	91.2	71.3	52.1	66.6	78.5
GC-Conv [39]	81.2	82.6	81.6	70.2	88.6	70.6	86.2	**86.6**	81.6	79.6	58.9	90.8	76.8	53.2	67.2	81.6
RI-Fwk. [18]	81.4	**84.5**	85.1	75.0	88.2	72.4	90.7	84.4	80.3	**84.0**	**68.8**	92.6	76.1	52.1	74.1	80.0
Ours	**81.8**	78.8	85.4	**78.0**	**89.6**	**76.7**	**91.6**	85.7	**81.7**	82.1	67.6	**95.0**	**79.1**	**63.5**	**76.5**	81.0

Additional Visualization of Learned Orientations Figure 5 and 6 present more examples of learned orientations from the point cloud classification model. We observe that though not explicitly supervised, the learned orientations are correlated with the structure of shapes. For example, from the first two rows of Figure 5, we can see that red arrows on the airplane's wings and tails head to the direction to which the wings and tails stretch. We also find that the orientations are aware of structural similarity — the red arrows on chair legs consistently point to the front of the chair. We also note that the learned orientations are not trivial. An example is the second shape on the bottom row of Figure 6. The red arrows are obviously correlated to the vase's surface normals, rather than simply diverge from the center of the vase.

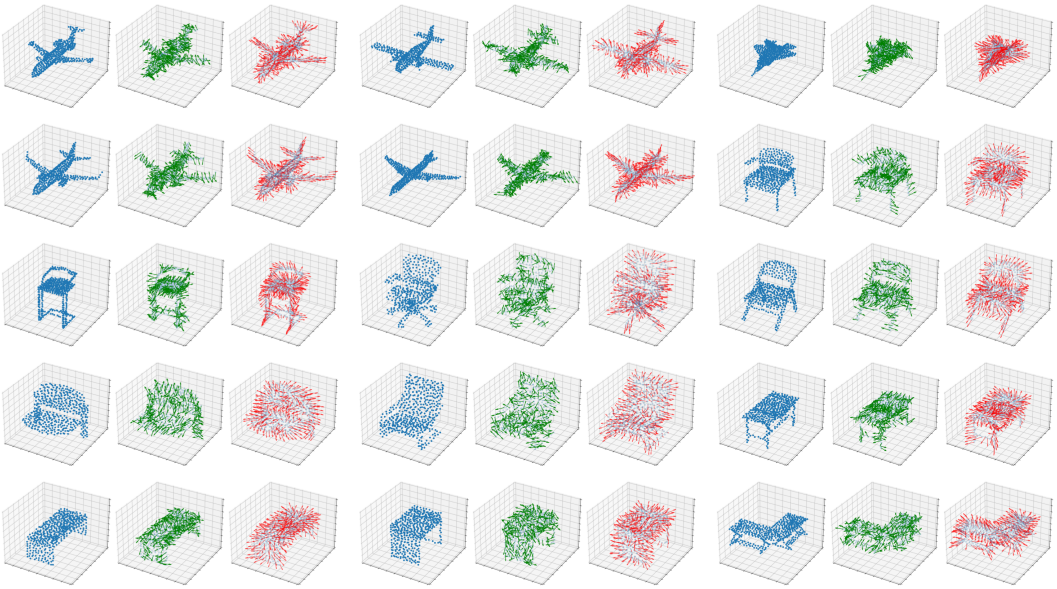

Figure 5. Green and red arrows indicate the first two column vectors of learned orientation matrices.

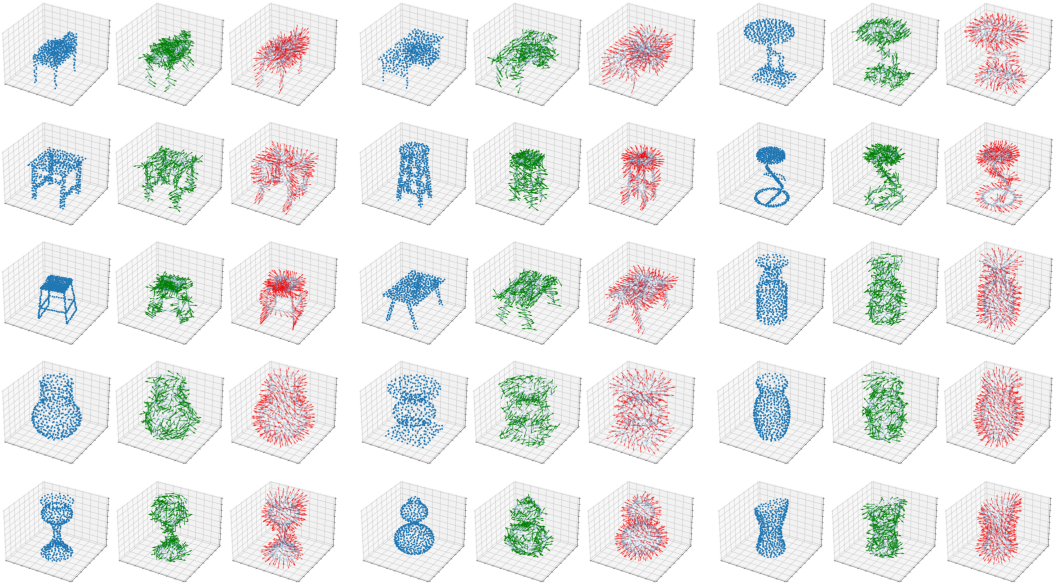

Figure 6. Green and red arrows indicate the first two column vectors of learned orientation matrices.