E2PN: Efficient SE(3)-Equivariant Point Network (Appendix)

Minghan Zhu
University of Michigan
minghanz@umich.edu

Maani Ghaffari
University of Michigan
maanigj@umich.edu

William A Clark
Cornell University
wac76@cornell.edu

Huei Peng
University of Michigan
hpeng@umich.edu

1. Preliminaries and notation

We first review some basic concepts in group theory and representation theory briefly. They are highly relevant for understanding the big picture from a theoretical perspective of our work and general equivariant deep learning literature.

Groups: A group G is a set equipped with a binary operation $\cdot$, satisfying the following conditions: (1) the set is closed under the operation: $x \cdot y \in G, \forall x, y \in G$; (2) the operation is associative: $x \cdot (y \cdot z) = (x \cdot y) \cdot z, \forall x, y, z \in G$; (3) there is an identity element e in the set such that $x \cdot e = e \cdot x = x, \forall x \in G$; (4) there is an inverse x^{-1} for each element x in the set such that $x \cdot x^{-1} = x^{-1} \cdot x = e$. For example, the integer set $\mathbb{Z}$ is a group under the addition operator with identity 0 and inverse $-x$ for any $x \in \mathbb{Z}$. Sometimes we omit the $\cdot$ notation.

Group actions and representations: We say a group *acts* on a set X if any element g in G corresponds to a transformation $\rho(g)$ on X, i.e., $[\rho(g)](x) \in X, \forall x \in X$, such that $\rho(g_1) \circ \rho(g_2) = \rho(g_1 g_2), \forall g_1, g_2 \in G$, where $\circ$ denotes function compositions, and $[\rho(e)](x) = x, \forall x \in X$. When X is a linear space and $\rho(g)$ is linear, we say ρ is a (linear) *representation* of G in X. When X is n (finite)-dimensional linear space, we have a representation $\rho : G \to \mathrm{GL}(n)$, i.e., we can write $[\rho(g)](x)$ as $\rho(g)x$, where $\rho(g)$ takes the form of n-by-n invertible matrices and acts by multiplication on the left. For example, the representations of 2D rotations $\mathrm{SO}(2)$ in $\mathbb{R}^2$ are the 2-by-2 orthonormal matrices with determinant 1. We also use (ρ, X) as a shorthand to denote the representation and space on which it acts.

Equivariance: Given spaces V_1 with representation ρ_1 of G and V_2 with representation ρ_2 of G, we say a mapping $\phi : V_1 \to V_2$ is *G-equivariant* if $\phi \circ \rho_1(g) = \rho_2(g) \circ \phi, \forall g \in G$. A G-equivariant linear map is also called an *intertwiner*. The space of intertwiners is denoted $\mathrm{Hom}_G(\rho_1, \rho_2)$, homomorphisms of group representations ρ_1, ρ_2 of G.

Subgroups, cosets, and quotient spaces: A subgroup H of G is a subset of G that is also a group, denoted $H \leq G$. For example, $\mathrm{SO}(2) \leq \mathrm{SO}(3)$. Given $H \leq G$ and $g \in G$, we can define a (left-)*coset* as $gH = \{gh | h \in H\}$. For a given H, all cosets are either equal or disjoint. Each coset is of the same size (contains the same number of elements), and they partition the whole group. The set of cosets forms a coset space (or *quotient space*) $G/H = \{gH | g \in G\}$. In short, a coset is both a subset in the group and an element in the quotient space.

Stabilizer subgroup: If a group G acts on set X by ρ, for $x \in X$, the stabilizer subgroup is defined as $\mathrm{Stab}_G(x) \triangleq \{g \in G | \rho(g)x = x\}$. By definition, the stabilizer subgroup $\mathrm{Stab}_G(e_{G/H})$ for the quotient space G/H is H.

Homogeneous spaces: Assume that a group G acts on a space X through action ρ, we call X a *homogeneous space* of G if G acts *transitively* on X, i. e., any two elements in X are connected by a group action, $\forall x_1, x_2 \in X, \exists g \in G$, s.t. $x_1 = \rho(g)x_2$. A quotient space G/H is a homogeneous space of G.

Induced representations: Here is an important known result [1, 4, 5]: given a representation ρ of subgroup H on vector space V, one can *induce* a representation $\pi = \mathrm{Ind}_H^G \rho$ of G for the space of functions $\mathcal{F} = \{f : G/H \to V\}$. It provides a way to define group actions in function spaces, a foundation of the research on equivariant feature learning.

2. Definition of the section functions

In Sec. 3.2.2, we define the convolution in a homogeneous space as Eq. (4), using the section function $s : X \to G$, mapping an element in the quotient space to a group element in the corresponding coset, i.e.,

$$s(x)H = x, \forall x \in X = G/H \tag{8}$$

In our work, for the continuous case, $H = \mathrm{SO}(2), G = \mathrm{SO}(3), X = S^2$, and Eq. (8) can be rewritten as

$$s(x)\mathbf{n} = x, \forall x \in S^2 \tag{9}$$

Since there are generally multiple group elements in a coset, section functions are not unique. Thus we need to define the section function to make the convolution well-defined. For any $R \in \mathrm{SO}(3)$, we can write $R = R_z(\alpha)R_y(\beta)R_z(\gamma)$ using Euler angles $\alpha \in [0, 2\pi), \beta \in [0, \pi], \gamma \in [0, 2\pi)$, and the coset it belongs to is $R\mathbf{n} = \{RR_z(\theta)|\theta \in [0, 2\pi)\} = \{R_z(\alpha)R_y(\beta)R_z(\gamma + \theta)|\theta \in [0, 2\pi)\} = R_z(\alpha)R_y(\beta)\mathbf{n}$. Thus a natural section from S^2 to $\mathrm{SO}(3)$ is

$$s(R\mathbf{n}) \triangleq R_z(\alpha)R_y(\beta) \tag{10}$$

which removes the last z-axis rotation in z-y-z Euler angle rotations. In the discretized setup, Eq. (10) does not work because for $R\mathbf{n} \in S^{2\prime} \subset S^2$, $s(R\mathbf{n}) \in \mathrm{SO}(3)$ may not be in $\mathrm{SO}(3)'$. In this case, we just arbitrarily select an element in each coset as the section so that $s' : S^{2\prime} \to \mathrm{SO}(3)'$ satisfies Eq. (8). While the selection is arbitrary, it should be fixed once selected so that the behavior of the function is deterministic and consistent for different inputs.

3. The derivation of our proposed convolution

3.1. The equivariance of our convolution

The derivation of the convolution in this paper is mostly built upon [4]. We do not discover new theorems. Our result is an application of the existing theoretical results in a specification that is not previously discussed in the literature. Here we start from the conclusions in [4] and show how it leads to our design of convolutions.

We first need to introduce another concept. For any $g \in G, x \in G/H$,

$$(gs(x))H = g(s(x)H) = g(x) = (s(gx))H \tag{11}$$

meaning that $gs(x)$ and $s(gx)$ are in the same coset, but $gs(x)$ and $s(gx)$ are not necessarily equal. We can relate these two using a function $\mathrm{h} : G/H \times G \to H$ as:

$$gs(x) = s(gx)\mathrm{h}(x, g) \tag{12}$$

i.e., $\mathrm{h}(x, g) \triangleq s(gx)^{-1}gs(x)$. This function h describes how the representative group element twists beyond jumping to another coset when applied with another group element, therefore heavily relying on s. We may denote it as h_s, but we will go with h in the following since we already selected and fixed s in Sec. 2.

With this h function, we can write down the form of induced representations. Given a space of functions $\mathcal{F} = \{f : G/H \to V\}$, assuming $\rho : H \to \mathrm{GL}(V)$ a representation of subgroup H in V, we define $\pi = \mathrm{Ind}_H^G\rho : G \to \mathrm{GL}(\mathcal{F})$ as:

$$[\pi(g)f](x) \triangleq \rho(\mathrm{h}(g^{-1}x, g))f(g^{-1}x) \tag{13}$$

It is shown in [4] that Eq. (13) is a valid representation. Denote $\mathcal{F}_1 = \{G/H \to V_1\}$ and $\mathcal{F}_2 = \{G/H \to V_2\}$ with representations (ρ_1, V_1) and (ρ_2, V_2) on H, any linear mapping $\mathcal{F}_1 \to \mathcal{F}_2$ equivariant to the induced representations $\mathrm{Ind}_H^G\rho_1$ and $\mathrm{Ind}_H^G\rho_2$ can be written as a cross-correlation with a twist:

$$[\kappa * f](x) = \int_{G/H} \kappa(s(x)^{-1}y)\rho_1(\mathrm{h}(y, s(x)^{-1}))f(y)\mathrm{d}y \tag{14}$$

where the space $\mathcal{K}_\mathcal{C}$ of valid kernels $\kappa : G/H \to \mathrm{Hom}(V_1, V_2)$ is equivalent to the space:

$$\mathcal{K}_\mathcal{D} = \{\overline{\kappa} : H\backslash G/H \to \mathrm{Hom}(V_1, V_2)| \\ \overline{\kappa}(x) = \rho_2(h)\overline{\kappa}(x)\rho_1^x(h)^{-1}, \forall x \in H\backslash G/H, h \in H^{\eta(x)H}\} \tag{15}$$

where $H\backslash G/H = \{HgH|g \in G\}$ is the set of double cosets in which $HgH = \{h_1gh_2|h_1, h_2 \in H\}$ is called a double coset. $\eta : H\backslash G/H \to G$ is a section function for double cosets. $H^{\eta(x)H} = \{h \in H|h\eta(x)H = \eta(x)H\}$ is the set of stabilizers for double coset $x \in H\backslash G/H$, and ρ_1^x is the representation of $H^{\eta(x)H}$ defined as $\rho_1^x(h) = \rho_1(\eta(x)^{-1}h\eta(x))$ for $h \in H^{\eta(x)H}$.

While the above looks a bit involved, recall that in this work, we use scalar-type features, meaning that we choose the trivial identity representation for the subgroup $H = \mathrm{SO}(2)$, i.e., $\rho_1(h) = \mathrm{Id}_{V_1}, \rho_2(h) = \mathrm{Id}_{V_2} \forall h \in H$, which simplifies the equations. The induced representation $\pi = \mathrm{Ind}_H^G\rho : G \to \mathrm{GL}(\mathcal{F})$ is in the form:

$$[\pi(g)f](x) = f(g^{-1}x), \forall g \in G, x \in G/H \tag{16}$$

The convolution in Eq. (14) now looks like:

$$\begin{aligned}[\kappa * f](x) &= \int_{G/H} \kappa(s(x)^{-1}y)f(y)\mathrm{d}y \\ &= \int_{G/H} \kappa(y)f(s(x)y)\mathrm{d}y\end{aligned} \tag{17}$$

which is consistent with Eq. (4) in the main paper. The equivalent space of kernels is:

$$\mathcal{K}_\mathcal{D} = \{\overline{\kappa} : H\backslash G/H \to \mathrm{Hom}(V_1, V_2)\} \tag{18}$$

3.2. The specific form of our kernel

In this paper, we work with $G = \mathrm{SE}(3)$ and $H = \mathrm{SO}(2)$. In the following, we derive G/H and $H\backslash G/H$ in this setup since they do not appear commonly in the literature.

The group $\mathrm{SE}(3) = \mathrm{SO}(3) \ltimes \mathbb{R}^3$ is the semi-direct product of $\mathrm{SO}(3)$ and $\mathbb{R}^3$ (the latter is a normal subgroup). We can denote a group element of $\mathrm{SE}(3)$ as (R, t) where $R \in \mathrm{SO}(3), t \in \mathbb{R}^3$, such that the group action $\cdot$ is defined as $(R_1, t_1)\cdot(R_2, t_2) = (R_1R_2, R_1t_2 + t_1)$, and accordingly the group inverse is defined as $(R, t)^{-1} = (R^{-1}, -R^{-1}t)$.

The group $\mathrm{SO}(3)$ can be written as a subgroup of $\mathrm{SE}(3)$ as $(R, 0)$. Using Euler angles we have $\forall R \in$

$\mathrm{SO}(3), \exists \alpha \in [0, 2\pi), \beta \in [0, \pi], \gamma \in [0, 2\pi)$, such that $R = R_z(\alpha)R_y(\beta)R_z(\gamma)$, where R_z represents rotation around the z-axis, and similarly for R_y. We also have $\mathrm{SO}(2) \cong \{(R_z(\gamma), 0) \in \mathrm{SE}(3) | \gamma \in [0, 2\pi)\}$.

Therefore, a left coset of $H = \mathrm{SO}(2)$ in $G = \mathrm{SE}(3)$ is $gH = \{gh | h \in H\} = \{(R_g, t_g) \cdot (R_z(\gamma_h), 0) | \gamma_h \in [0, 2\pi)\} = \{(R_z(\alpha_g)R_y(\beta_g)R_z(\gamma_g + \gamma_h), t_g) | \gamma_h \in [0, 2\pi)\}$, meaning that a left coset can be parameterized by α_g, β_g, t_g. Then the set of left cosets G/H is homeomorphic to the Cartesian product $S^2 \times \mathbb{R}^3 = \{(R_z(\alpha)R_y(\beta)\mathbf{n}, t) | \alpha \in [0, 2\pi), \beta \in [0, \pi], t \in \mathbb{R}^3\}$, where S^2 is the surface of a sphere, $\mathbf{n} = t(0, 0, 1)$ is the unit vector pointing to the north pole. Here we abuse the notation $(x\mathbf{n}, y)$ as an ordered pair in the set $S^2 \times \mathbb{R}^3$. It can be understood as a point x on a sphere centered at some point y in $\mathbb{R}^3$. The group $G = \mathrm{SE}(3)$ acts on G/H by left multiplication: $(R_g, t_g)(R\mathbf{n}, t) = (R_g R\mathbf{n}, R_g t + t_g)$.

We further investigate the double coset space $H\backslash G/H$. An element in the set can be written as $HgH = \{h_1 g h_2 | h_1, h_2 \in H\} = \{(R_z(\alpha_g + \gamma_{h_1})R_y(\beta_g)R_z(\gamma_g + \gamma_{h_2}), R_z(\gamma_{h_1})t_g) | \gamma_{h_1}, \gamma_{h_2} \in [0, 2\pi)\}$. We can use $t(x, y, z)$ to specify the coordinate of an element in $\mathbb{R}^3$, and we can always rewrite $t(x, y, z) = R_z(\gamma_t)t(r_g, 0, z_g)$ where $r_g = \sqrt{x^2 + y^2} \geq 0$ and $\gamma_t = \arctan 2(y, x)$. Then we can rewrite

$$HgH = \{(R_z(\alpha_g + \gamma_{h_1})R_y(\beta_g)R_z(\gamma_g + \gamma_{h_2}), R_z(\gamma_{h_1} + \gamma_t)t(r_g, 0, z_g)) | \gamma_{h_1}, \gamma_{h_2} \in [0, 2\pi)\} \quad (19)$$

Let us rename $\gamma_1 \triangleq \gamma_{h_1} + \gamma_t, \theta_g \triangleq \alpha_g - \gamma_t, \gamma_2 \triangleq \gamma_g + \gamma_{h_2}$, then we have

$$HgH = \{(R_z(\theta_g + \gamma_1)R_y(\beta_g)R_z(\gamma_2), R_z(\gamma_1)t(r_g, 0, z_g)) | \gamma_1, \gamma_2 \in [0, 2\pi)\} \quad (20)$$

Now it is clear that an element in $H\backslash G/H$ can be determined by four parameters $(\theta_g, \beta_g, r_g, z_g)$, with $\theta_g \in [0, 2\pi), \beta_g \in [0, \pi], r_g \geq 0, z_g \in \mathbb{R}$. We denote an element in $H\backslash G/H$ as ${}_H g_H(\theta_g, \beta_g, r_g, z_g)$. We have $H\backslash G/H \cong S^2 \times \mathbb{R}^+ \times \mathbb{R}$. Geometrically, each point (r_g, z_g) in the $\mathbb{R}^+ \times \mathbb{R}$ plane corresponds to a circle around the z-axis with radius r_g at height z_g. On the other hand, θ_g, β_g parameterizes a point on the sphere S^2.

It follows that we can define a kernel $\overline{\kappa} \in \mathcal{K}_{\mathcal{D}} = \{\overline{\kappa} : S^2 \times \mathbb{R}^+ \times \mathbb{R} \to \mathrm{Hom}(V_1, V_2)\}$, and then injectively map it to $\kappa \in \mathcal{K}_{C_0} = \{\kappa : S^2 \times \mathbb{R}^3 \to \mathrm{Hom}(V_1, V_2)\}$. The fact that $S^2 \times \mathbb{R}^+ \times \mathbb{R} \subsetneq S^2 \times \mathbb{R}^3$ implies that $\mathcal{K}_{\mathcal{D}} \cong \mathcal{K}_{\mathcal{C}} \subsetneq \mathcal{K}_{C_0}$. In other words, there is a certain constraint on $\mathcal{K}_{C_0}$ to form the actual set of valid equivariant kernels $\mathcal{K}_{\mathcal{C}}$. As shown in [4], the general form of the constraint can be written as:

$$\kappa(hx) = \rho_2(h)\kappa(x)\rho_1(\mathrm{h}(x, h)^{-1}), \quad (21)$$

for $\kappa \in \mathcal{K}_{\mathcal{C}}, x \in G/H, h \in H$. As discussed in Sec. 3.1, ρ_1 and ρ_2 are both identity; thus Eq. (21) becomes $\kappa(hx) = \kappa(x)$, which is equivalent to Eq. (6) in the main paper in our specific case.

4. Equivariance of element-wise non-linear layers and normalization layers

For a feature map of shape $B \times C \times N \times A$, where B is the batch size, C is the feature channel, N is the number of spatial points in $\mathbb{R}^3$, A corresponds to the spherical coordinates in $S^{2\prime}$, a batch normalization (BatchNorm [6]) is to calculate the mean and variance across the $B \times N \times A$ channels and apply a constant scaling factor and shift for each $B \times N \times A$ tensor. For instance normalization (InstanceNorm [8]), one just need to change $B \times N \times A$ to $N \times A$. In either case, consider the feature map as a function $f : N \times A \to \mathbb{R}^{B \times C}$, a BatchNorm or InstanceNorm (denoted $\mathcal{N}$) behaves like an element-wise operation, i.e.,

$$[\mathcal{N} \cdot f](x) = af(x) + b = \mathcal{N} \cdot f(x), \ \forall x \in S^{2\prime} \times \mathbb{R}^3 \quad (22)$$

since a, b are constant vectors. Here $\cdot$ denotes applying a transformation.

Recall that our induced representation Eq. (16) is in a similar form of a regular representation, which is realized by a change of coordinate without modifying the function value. Such a representation π is commutative with element-wise operations (denoted as $\mathcal{E}$):

$$\begin{aligned}
&[(\pi(g) \circ \mathcal{E}) \cdot f](x) \\
&= [\pi(g) \cdot (\mathcal{E} \cdot f)](x) = [\mathcal{E} \cdot f](g^{-1}x) = \mathcal{E} \cdot f(g^{-1}x) \\
&= \mathcal{E} \cdot [\pi(g) \cdot f](x) = [\mathcal{E} \cdot (\pi(g) \cdot f)](x) \\
&= [(\mathcal{E} \circ \pi(g)) \cdot f](x), \forall g \in G
\end{aligned} \quad (23)$$

Or we can say:

$$\pi(g) \circ \mathcal{E} = \mathcal{E} \circ \pi(g), \forall g \in G \quad (24)$$

It shows that element-wise non-linear layers like ReLu and normalization layers, including BatchNorm and InstanceNorm, are G-equivariant.

5. Prediction heads and loss functions

5.1. Pose estimation task

The pose (rotation) estimation task is fulfilled with a prediction head designed as shown in Fig. 1. The inputs to the prediction head for each pair of point clouds are two $S^{2\prime} \times C$ features, where C is the number of feature channels. We call the $S^{2\prime}$ coordinates *anchors* in this section. We apply $R_i \in \mathcal{I}$ to the second point-cloud mentally, corresponding to 60 permutations of the anchors for f_2. If the two point clouds are different exactly by a rotation in $\mathcal{I}$, then

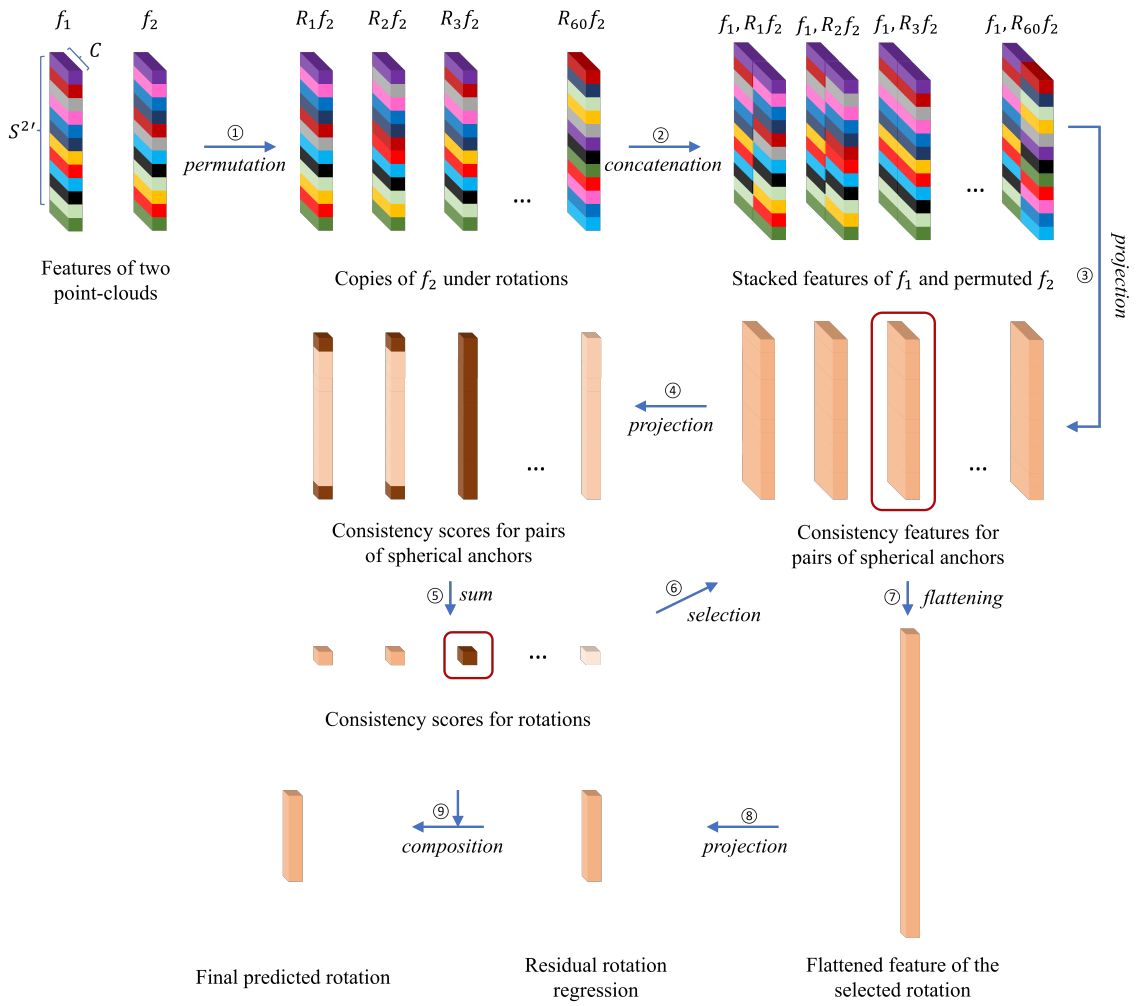

Figure 1. Illustration of the prediction head for rotation estimation. The numbers show the sequence of operations. The colors in the top row correspond to different spherical anchors. The shade of color after step 4 and after step 5 represents the matching scores for pairs of anchors. Darker means higher.

one of the permuted f_2 should be exactly the same as f_1. We stack the features together and use several linear layers to find the match. Notice that the matching is defined as a binary classification problem for each pair of anchors instead of a multi-class classification problem for the overall feature corresponding to a certain rotation. It aligns better with the underlying geometry because a subset of anchors may align even under a wrong rotation since any rotations in $\mathrm{SO}(2)$ keep the north-pole and south-pole vertices static. We can find the correct rotation class by summing over all anchor pairs and picking the rotation with the highest overall matching score. After finding the correct permutation (equivalent to the $R_i \in \mathcal{I} \subset \mathrm{SO}(3)$), we flatten the feature and regress the residual rotation using quaternions in a way similar to [3].

Accordingly, the loss functions are the binary cross entropy loss for anchor-pair matching and L2 loss on the residual rotation regression.

5.2. Object classification task

For the classification task, we follow a similar philosophy as the rotation estimation task. Here we do not have a pair of inputs from which to find the relative rotation. Therefore, we imagine that there is a *reference* object for each category, with a canonical permutation of the features representing the underlying canonical pose. We learn the features of the reference object in each class and use them to classify input point clouds.

The core learnable parameter is the reference feature X of shape $S^{2\prime} \times C \times N$ where N is the number of object classes. We can denote $X_n \in \mathbb{R}^{S^{2\prime} \times C}$ as the reference feature for object class n. We directly calculate the inner product between the reference features and the permuted input features. The score of rotations under every object-class hypothesis is calculated by summing over the inner products across all anchors. The score of each class is de-

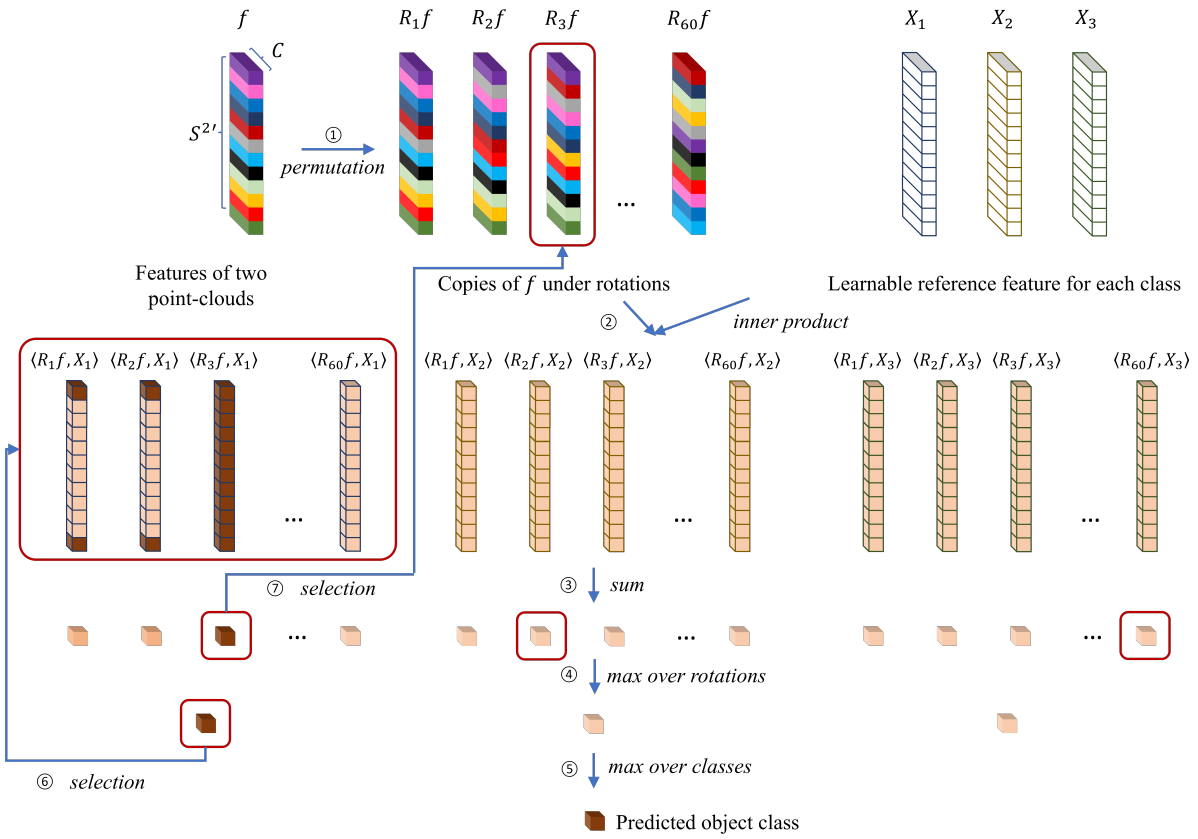

Figure 2. Illustration of the prediction head for object classification. The numbers show the sequence of operations. The solid colors in the top row correspond to different spherical anchors. The line colors on the top right represent different semantic classes. Here we only use three classes for illustration. The shade of color after step 2 represents the scores. Darker means higher. $\langle \cdot, \cdot \rangle$ denotes the inner product of finite-dim vectors.

fined as the maximum rotation score in each class hypothesis. We first generate the final prediction of the object class using the class score. Then we go back to the inner product matrix corresponding to this class and use it as the anchor-matching score prediction. Finally, the best rotation in this class is used to retrieve the specific permutation of the input feature, which forms a rotation-invariant feature for this object.

The basic assumption here is that only the correct object class yields a high-quality matching under the actual rotation. Thus we first solve for the classification and then determine the optimal rotation only in this class, which is used for generating rotation-invariant features.

The loss functions applied are the cross entropy loss for object classification and the binary cross entropy loss for anchor-matching prediction.

5.3. Keypoint matching task

For the keypoint matching task, there is no definition of a canonical pose for a local patch of points around a keypoint. Because the feature learning in this task does not involve the corresponding patch in another point cloud, we cannot define relative poses as well. Therefore, we do not apply the permutation layer in this task. Instead, we simply follow the same design as in [3] using GA pooling, except that our attentive pooling is not over $\mathcal{I}$, but $S^{2\prime}$.

The loss function applied here is the batch-hard triplet loss, also consistent with [3].

6. More specifications in the experiments

The batch size used in the efficiency comparison in Tab. 1 is specified in Tab. 5. For the keypoint matching task, the number of global scans processed (n_g) and the number of local patches extracted from each global scan (n_l) define the input size. We use $n_g \times n_l$ as the notation in Tab. 5.

The training optimizer and learning rate schedule follow the default setup of EPN [3]. The number of feature channels (i.e., width) and the number of network layers (i.e., depth) also follow the settings in EPN, except that for the object classification task on ModelNet40, we reduced the backbone width by half compared with the original EPN setting (first layer width changed from 64 to 32, later layers in the backbone changed accordingly). The network width is the same across KPConv [7], EPN, and our E2PN in

Table 5. The batch sizes used in the efficiency comparison in terms of the GPU memory consumption and the running speed between EPN [3] and our method on three tasks as in Tab. 1.

Tasks	ModelNet40 Pose		ModelNet40 Classification		3DMatch Keypoint Matching	
Modes	Training	Inference	Training	Inference	Training	Inference
Batch size	8	8	12	24	1×16	8×24

our experiments, therefore their comparison remains valid. For all the three networks, the classification accuracy is not harmed by the width reduction.

Due to the page limit in the main paper, we only mentioned "all results are trained and tested with rotational augmentation" in the classification task section. Here we provide more details. During training, random rotations are applied, following the practice in EPN. EPN and our network are equivariant to a discretization of $\mathrm{SO}(3)$; therefore, we apply rotation augmentations to let the network interpolate among the discretizations well and regress the residual rotation adding to the discretized rotations. During testing, we use fixed rotations for each test input so that results are repeatable and comparable.

We also provide more details here on the steerable CNN baseline based on ESCNN [2] in the classification task experiment. [2] established a general framework for steerable CNNs equivariant to $\mathrm{O}(3)$ and its subgroups. It is relevant to the discussion in our paper and reported results on shape classification on the ModelNet10 dataset, which is a similar task to our experiments on ModelNet 40 dataset. Therefore we include it as one of our baselines. While the library for this work is open-sourced, the specific implementation of the network for the tasks mentioned in [2] is not provided. Therefore we implemented the network using the library according to the specifics stated in [2]. We also implemented the data conversion to transform the shapes into voxel grids as described in [2]. The Gaussian kernel radius in voxel generation and the learning rate are not specified in [2], so we did a hand tuning and reported the result under the best settings. We applied the final values $\sigma = 0.03$ for the Gaussian kernel in voxel generation and $lr = 10^{-3}$ as the initial learning rate. The multiplicity of irreps in the backbone is not specified either, and we found that using band-limited regular representations yields better results than using equal proportions of irreps. Thus we use multiple and-limited regular representations and align the total number of channels to the numbers specified in [2]. We implemented the $\mathrm{SO}(3)$ equivariant version with frequency up to 3. Among the invariant maps discussed in [2], group pooling and norm pooling are implemented in their library. Group pooling is not recommended for continuous groups in [2]; therefore, we use norm pooling as the invariant layer.

The result is shown in Table 3 in the main paper. We did not compare the efficiency because the forms of input (voxels vs. point clouds) and network structures (number of layers, channels, and connections) are both quite different. We can see that the network of [2] underperforms both EPN [3] and our network. One of the reasons could be that both EPN and our model for the classification task are also trained with the auxiliary task of rotation estimation (for the GA pooling layer or the permutation layer). In contrast, [2] is only trained with the classification task as described in their paper, which may cause more information loss in the invariant layer. Another reason could be that voxel inputs lose some details compared with point clouds. Transitioning from ModelNet10 to ModelNet40 dataset may require some scale-up of the network in terms of depth and width and some other careful tuning.

References

[1] Tullio Ceccherini-Silberstein, A Machì, Fabio Scarabotti, and Filippo Tolli. Induced representations and Mackey theory. *Journal of Mathematical Sciences*, 156(1):11–28, 2009. 1

[2] Gabriele Cesa, Leon Lang, and Maurice Weiler. A program to build E(N)-equivariant steerable CNNs. In *International Conference on Learning Representations*, 2022. 6

[3] Haiwei Chen, Shichen Liu, Weikai Chen, Hao Li, and Randall Hill. Equivariant point network for 3D point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 14514–14523, 2021. 4, 5, 6

[4] Taco S Cohen, Mario Geiger, and Maurice Weiler. Intertwiners between induced representations (with applications to the theory of equivariant neural networks). *arXiv preprint arXiv:1803.10743*, 2018. 1, 2, 3

[5] Taco S Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016. 1

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 3

[7] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 5

[8] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 3