On Learning Deep O(n)-Equivariant Hyperspheres

Pavlo Melnyk [1] **Michael Felsberg** [1] **Mårten Wadenbäck** [1] **Andreas Robinson** [1] **Cuong Le** [1]

Abstract

In this paper, we utilize hyperspheres and regular n-simplexes and propose an approach to learning deep features equivariant under the transformations of nD reflections and rotations, encompassed by the powerful group of O(n)*. Namely, we propose* O(n)*-equivariant neurons with spherical decision surfaces that generalize to any dimension n, which we call* `Deep Equivariant Hyperspheres`*. We demonstrate how to combine them in a network that directly operates on the basis of the input points and propose an invariant operator based on the relation between two points and a sphere, which as we show, turns out to be a Gram matrix. Using synthetic and real-world data in nD, we experimentally verify our theoretical contributions and find that our approach is superior to the competing methods for* O(n)*-equivariant benchmark datasets (classification and regression), demonstrating a favorable speed/performance trade-off. The code is available on GitHub.*

1. Introduction

Spheres[1] serve as a foundational concept in Euclidean space while simultaneously embodying the essence of non-Euclidean geometry through their intrinsic curvature and non-linear nature. This motivated their usage as decision surfaces encompassed by spherical neurons (Perwass et al., 2003; Melnyk et al., 2021).

Felix Klein's Erlangen program of 1872 (Hilbert & Cohn-Vossen, 1952) introduced a methodology to unify non-Euclidean geometries, emphasizing the importance of studying geometries through their invariance properties under transformation groups. Similarly, geometric deep learning (GDL) as introduced by Bronstein et al. (2017; 2021) constitutes a unifying framework for various neural architectures. This framework is built from the first principles of geometry—symmetry and scale separation—and enables tractable learning in high dimensions.

Symmetries play a vital role in preserving structural information of geometric data and allow models to adjust to different geometric transformations. This flexibility ensures that models remain robust and accurate, even when the input data undergo various changes. In this context, spheres exhibit a maximal set of symmetries compared to other geometric entities in Euclidean space.

The orthogonal group O(n) fully encapsulates the symmetry of an nD sphere, including both rotational and reflection symmetries. Integrating these symmetries into a model as an inductive bias is often a crucial requirement for problems in natural sciences and the respective applications, e.g., molecular analysis, protein design and assessment, or catalyst design (Rupp et al., 2012; Ramakrishnan et al., 2014; Townshend et al., 2021; Jing et al., 2021; Lan et al., 2022).

In this paper, we consider data that live in Euclidean space (such as point clouds) and undergo rotations and reflections, i.e., transformations of the O(n)-group. Enriching the theory of steerable 3D spherical neurons (Melnyk et al., 2022), we present a method for learning O(n)-equivariant deep features using regular n-simplexes[2] and nD spheres, which we call `Deep Equivariant Hyperspheres` (see Figure 1). The name also captures the fact that the vertices of a regular n-simplex lie on an nD sphere, and that our results enable combining equivariant hyperspheres in multiple layers, thereby enabling *deep* propagation.

Our main contributions are summarized as follows:

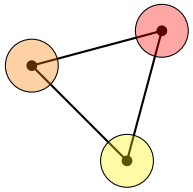
- We propose O(n)-equivariant spherical neurons, called `Deep Equivariant Hyperspheres`, readily generalizing to any dimension (see Section 4).
- We define and analyze generalized concepts for a network composed of the proposed neurons, including the invariant operator modeling the relation between two points and a sphere (21).

[1]Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, Sweden. Correspondence to: Michael Felsberg <michael.felsberg@liu.se>.

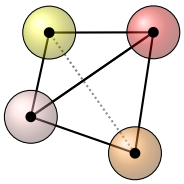
Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]By *sphere*, we generally refer to an nD sphere or a hypersphere; e.g., a circle is thus a 2D sphere.

[2]We use the fact that a regular n-simplex contains $n+1$ equidistant vertices in nD.



$$\mathbf{M}_2 = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & (\sqrt{3}-1)/2 & -(\sqrt{3}+1)/2 \\ 1 & -(\sqrt{3}+1)/2 & (\sqrt{3}-1)/2 \\ 1 & 1 & 1 \end{bmatrix}$$



$$\mathbf{M}_3 = \frac{1}{2}\begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure 1. The central components of `Deep Equivariant Hyperspheres` (best viewed in color): regular n-simplexes with the nD spherical decision surfaces located at their vertices and the simplex change-of-basis matrices $\mathbf{M}_n$ (displayed for $n = 2$ and $n = 3$).

- Conducting experimental validation on both synthetic and real-world data in nD, we demonstrate the soundness of the developed theoretical framework, outperforming the related equi- and invariant methods and exhibiting a favorable speed/performance trade-off.

2. Related Work

The concept of spheres is also an essential part of spherical convolutional neural networks (CNNs) and CNNs designed to operate on 360 imagery (Coors et al., 2018; Su & Grauman, 2017; Esteves et al., 2018; Cohen et al., 2018; Perraudin et al., 2019). In contrast, our method does not map input data *on* a sphere, $\mathcal{S}^2$, nor does it perform convolution on a sphere. Instead, it embeds input in a higher-dimensional Euclidean space by means of a quadratic function. Namely, our work extrapolates the ideas from prior work by Perwass et al. (2003); Melnyk et al. (2021), in which spherical decision surfaces and their symmetries have been utilized for constructing equivariant models for the 3D case (Melnyk et al., 2022; 2024). We carefully review these works in Section 3.

Similarly to the approach of (Ruhe et al., 2023), our `Deep Equivariant Hyperspheres` directly operate on the basis of the input points, not requiring constructing an alternative one, such as a steerable spherical harmonics basis. Constructing an alternative basis is a key limitation of many related methods (Anderson et al., 2019; Thomas et al., 2018; Fuchs et al., 2020). Our method also generalizes to the orthogonal group of any dimensionality.

Another type of method is such as by Finzi et al. (2021), a representation method building equivariant feature maps by computing an integral over the respective group (which is intractable for continuous Lie groups and hence, requires coarse approximation). Another category includes methods operating on scalars and vectors: they update the vector information by learning the parameters conditioned on scalar information and multiplying the vectors with it (Satorras et al., 2021; Jing et al., 2021), or by learning the latent equivariant features (Deng et al., 2021).

While the methods operating on hand-crafted O(n)-invariant features are generally not as relevant to our work, the methods proposed by Villar et al. (2021) and Xu et al. (2021) are: The central component of the scalars method (Villar et al., 2021) is the computation of the pair-wise scalar product between the input points, just as the O(3)-invariant descriptor by Xu et al. (2021) is a sorted Gram matrix (SGM) of input point coordinates, which encodes global context using relative distances and angles between the points. In Section 4.5, we show how this type of computation naturally arises when one considers the relation between two points and a sphere for computing invariant features in our approach, inspired by the work of Li et al. (2001).

3. Background

In this section, we present a comprehensive background on the theory of spherical neurons and their rotation-equivariant version, as well as on the general geometric concepts used in our work.

3.1. Spherical Neurons via Non-Linear Embedding

Spherical neurons (Perwass et al., 2003; Melnyk et al., 2021) are neurons with, as the name suggests, spherical decision surfaces. By virtue of conformal geometric algebra (Li et al., 2001), Perwass et al. (2003) proposed to embed the data vector $\mathbf{x} \in \mathbb{R}^n$ and represent the sphere with center $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{R}^n$ and radius $r \in \mathbb{R}$ respectively as

$$\begin{aligned} \boldsymbol{X} &= \left(x_1, \ldots, x_n, -1, -\frac{1}{2}\|\mathbf{x}\|^2\right) \in \mathbb{R}^{n+2}, \\ \boldsymbol{S} &= \left(c_1, \ldots, c_n, \frac{1}{2}(\|\mathbf{c}\|^2 - r^2), 1\right) \in \mathbb{R}^{n+2}, \end{aligned} \tag{1}$$

and used their scalar product $\boldsymbol{X}^\top \boldsymbol{S} = -\frac{1}{2}\|\mathbf{x} - \mathbf{c}\|^2 + \frac{1}{2}r^2$ as a classifier, i.e., the spherical neuron:

$$f_S(\boldsymbol{X}; \boldsymbol{S}) = \boldsymbol{X}^\top \boldsymbol{S}, \tag{2}$$

with learnable parameters $\boldsymbol{S} \in \mathbb{R}^{n+2}$.

The sign of this scalar product depends on the position of the point $\mathbf{x}$ relative to the sphere $(\mathbf{c}, r)$: inside the sphere if positive, outside of the sphere if negative, and on the sphere if zero (Perwass et al., 2003). Geometrically, the activation of the spherical neuron (2) determines the cathetus length of the right triangle formed by $\mathbf{x}$, $\mathbf{c}$, and the corresponding point on the sphere (see Figure 2 in Melnyk et al. (2021)).

We note that with respect to the data vector $\mathbf{x} \in \mathbb{R}^n$, a spherical neuron represents a non-linear function $f_S(\,\cdot\,;\boldsymbol{S}) : \mathbb{R}^{n+2} \to \mathbb{R}$, due to the inherent non-linearity of the embedding (1), and therefore, does not necessarily require an activation function, as observed by Melnyk et al. (2021). The components of $\boldsymbol{S}$ in (1) can be treated as *independent* learnable parameters. In this case, a spherical neuron learns a *non-normalized* sphere of the form $\widetilde{\boldsymbol{S}} = (s_1, \ldots, s_{n+2}) \in \mathbb{R}^{n+2}$, which represents the same decision surface as its normalized counterpart defined in (1), thanks to the homogeneity of the embedding (Perwass et al., 2003; Li et al., 2001).

3.2. Equi- and Invariance under O(n)-Transformations

The elements of the orthogonal group $\mathrm{O}(n)$ can be represented as $n \times n$ matrices $\boldsymbol{R}$ with the properties $\boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^\top = \mathbf{I}_n$, where $\mathbf{I}_n$ is the identity matrix, and $\det \boldsymbol{R} = \pm 1$, geometrically characterizing nD rotations and reflections. The special orthogonal group $\mathrm{SO}(n)$ is a subgroup of $\mathrm{O}(n)$ and includes only orthogonal matrices with the positive determinant, representing rotations.

We say that a function $f : \mathcal{X} \to \mathcal{Y}$ is $\mathrm{O}(n)$-equivariant if for every $\boldsymbol{R} \in \mathrm{O}(n)$ there exists the transformation representation, $\rho(\boldsymbol{R})$, in the function output space, $\mathcal{Y}$, such that

$$\rho(\boldsymbol{R})\, f(\mathbf{x}) = f(\boldsymbol{R}\mathbf{x}) \quad \text{for all } \boldsymbol{R} \in \mathrm{O}(n),\ \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n. \tag{3}$$

We call a function $f : \mathcal{X} \to \mathcal{Y}$ $\mathrm{O}(n)$-invariant if for every $\boldsymbol{R} \in \mathrm{O}(n)$, $\rho(\boldsymbol{R}) = \mathbf{I}_n$. That is, if

$$f(\mathbf{x}) = f(\boldsymbol{R}\mathbf{x}) \quad \text{for all } \boldsymbol{R} \in \mathrm{O}(n),\ \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n. \tag{4}$$

Following the prior work convention (Melnyk et al., 2022; 2024) hereinafter, we write $\boldsymbol{R}$ to denote the same nD rotation/reflection as an $n \times n$ matrix in the Euclidean space $\mathbb{R}^n$, as an $(n+1) \times (n+1)$ matrix in the projective (homogeneous) space $P(\mathbb{R}^n) \subset \mathbb{R}^{n+1}$, and as an $(n+2) \times (n+2)$ matrix in $\mathbb{R}^{n+2}$. For the latter two cases, we achieve this by appending ones to the diagonal of the original $n \times n$ matrix without changing the transformation itself (Melnyk et al., 2021).

3.3. Steerable 3D Spherical Neurons and TetraSphere

Considering the 3D case, Melnyk et al. (2022) showed that a spherical neuron (Perwass et al., 2003; Melnyk et al., 2021) can be *steered*. In this context, *steerability* is defined as the ability of a function to be written as a linear combination of the rotated versions of itself, called *basis functions* (Freeman et al., 1991; Knutsson et al., 1992). For details, see the Appendix (Section A).

According to Melnyk et al. (2022), a 3D steerable filter consisting of spherical neurons needs to comprise a *minimum* of four 3D spheres: one learnable spherical decision surface $\boldsymbol{S} \in \mathbb{R}^5$ (1) and its three copies *rotated* into the other three vertices of the regular tetrahedron, using one of the results of Freeman et al. (1991) that the basis functions must be distributed in the space uniformly.

To construct such a filter, i.e., a steerable 3D spherical neuron, the main (learned) sphere center $\mathbf{c}_0$ needs to be rotated into $\frac{\|\mathbf{c}_0\|}{\sqrt{3}}\,(1,1,1)$ by the corresponding (geodesic) rotation $\boldsymbol{R}_O$. The resulting sphere center is then rotated into the other three vertices of the regular tetrahedron. This is followed by rotating all four spheres back to the original coordinate system. A steerable 3D spherical neuron can thus be defined by means of the 4×5 matrix $B(\boldsymbol{S})$ containing the four spheres:

$$\mathrm{F}(\boldsymbol{X};\boldsymbol{S}) = B(\boldsymbol{S})\boldsymbol{X}\,, \quad B(\boldsymbol{S}) = \Big[(\boldsymbol{R}_O^\top \boldsymbol{R}_{T_i} \boldsymbol{R}_O\, \boldsymbol{S})^\top\Big]_{i=1}^{4}\,, \tag{5}$$

where $\boldsymbol{X} \in \mathbb{R}^5$ is the input 3D point embedded using (1), $\{\boldsymbol{R}_{T_i}\}_{i=1}^4$ is the $\mathbb{R}^5$ rotation isomorphism corresponding to the rotation from the first vertex, i.e., $(1,1,1)$ to the i-th vertex of the regular tetrahedron[3].

Melnyk et al. (2022) showed that steerable 3D spherical neurons are $\mathrm{SO}(3)$-equivariant (or more precisely, $\mathrm{O}(3)$-equivariant, as remarked in Melnyk et al. (2024)):

$$V_{\boldsymbol{R}}\, B(\boldsymbol{S})\, \boldsymbol{X} = B(\boldsymbol{S})\, \boldsymbol{R}\boldsymbol{X}, \quad V_{\boldsymbol{R}} = \mathbf{M}^\top \boldsymbol{R}_O\, \boldsymbol{R}\, \boldsymbol{R}_O^\top \mathbf{M}\,, \tag{6}$$

where $\boldsymbol{R}$ is a representation of the 3D rotation in $\mathbb{R}^5$, and $V_{\boldsymbol{R}} \in G < \mathrm{SO}(4)$ is the 3D rotation representation in the filter bank output space, with $\mathbf{M} \in \mathrm{SO}(4)$ being a change-of-basis matrix that holds the homogeneous coordinates of the tetrahedron vertices in its columns as

$$\mathbf{M} = \begin{bmatrix}\mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 & \mathbf{m}_4\end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{7}$$

We note that with respect to the input vector $\mathbf{x} \in \mathbb{R}^3$, a steerable 3D spherical neuron represents a non-linear rotational-equivariant function $\mathrm{F}(\,\cdot\,;\boldsymbol{S}) : \mathbb{R}^5 \to \mathbb{R}^4$ with the learnable parameters $\boldsymbol{S} \in \mathbb{R}^5$.

TetraSphere As the first reported attempt to *learn* steerable 3D spherical neurons in an end-to-end approach, Melnyk et al. (2024) has presently introduced an approach for $\mathrm{O}(3)$-invariant point cloud classification based on said neurons and the VN-DGCNN architecture (Deng et al., 2021), called TetraSphere.

Given the point cloud input $\mathcal{X} \in \mathbb{R}^{N\times 3}$, the TetraSphere approach suggests to learn 4D features of each point by means of the TetraTransform layer $l_{\mathrm{TT}}(\,\cdot\,;\mathbf{S}) : \mathbb{R}^{N\times 3} \to \mathbb{R}^{N\times 4\times K}$ that consists of K steerable spherical neurons $B(\boldsymbol{S}_k)$ (see (5)) that are shared among the points. After the application of TetraTransform, pooling over the K dimensions

[3]Therefore, $\boldsymbol{R}_{T_1} = \mathbf{I}_5$, i.e., the original $\boldsymbol{S}$ remains at $\mathbf{c}_0$.

takes place, and the obtained feature map is then propagated through the VN-DGCNN network as-is. However, the questions of how to combine the steerable neurons in multiple layers and how to make them process data in dimensions other than 3 have remained open.

3.4. Regular Simplexes

Geometrically, a regular n-simplex represents $n+1$ equidistant points in nD (Elte, 2006), lying on an nD sphere with unit radius. In the 2D case, the regular simplex is an equilateral triangle; in 3D, a regular tetrahedron, and so on.

Following Cevikalp & Saribas (2023), we compute the Cartesian coordinates of a regular n-simplex as $n+1$ vectors $\mathbf{p}_i \in \mathbb{R}^n$:

$$\mathbf{p}_i = \begin{cases} n^{-1/2}\,\mathbf{1}, & i = 1 \\ \kappa\,\mathbf{1} + \mu\,\mathbf{e}_{i-1}, & 2 \le i \le n+1\,, \end{cases} \qquad \kappa = -\frac{1+\sqrt{n+1}}{n^{3/2}}\,,\ \ \mu = \sqrt{1+\frac{1}{n}}\,, \tag{8}$$

where $\mathbf{1} \in \mathbb{R}^n$ is a vector with all elements equal to 1 and $\mathbf{e}_i$ is the natural basis vector with the i-th element equal to 1.

For the case $n = 3$, we identify the following connection between (7) and (8): the columns of $\mathbf{M}$, $\mathbf{m}_i \in \mathbb{R}^4$, are the coordinates of the regular 3-simplex appended with a constant and normalized to unit length; that is,

$$\mathbf{m}_i = \frac{1}{p}\begin{bmatrix}\mathbf{p}_i \\ 1/\sqrt{3}\end{bmatrix} \text{ with } p = \left\|\begin{bmatrix}\mathbf{p}_i \\ 1/\sqrt{3}\end{bmatrix}\right\|, 1 \le i \le 4. \tag{9}$$

4. Deep Equivariant Hyperspheres

In this section, we provide a complete derivation of the proposed O(n)-equivariant neuron based on a learnable spherical decision surface and multiple transformed copies of it, as well as define and analyze generalized concepts of equivariant bias, non-linearities, and multi-layer setup.

While it is intuitive that in higher dimensions one should use more copies (i.e., vertices) than in the 3D case (Melnyk et al., 2022), it is uncertain exactly how many are needed. We hypothesize that the vertices should constitute a regular n-simplex ($n+1$ vertices) and rigorously prove it in this section.

4.1. The Simplex Change of Basis

We generalize the change-of-basis matrix (7) to nD by introducing $\mathbf{M}_n$, an $(n+1) \times (n+1)$ matrix holding in its columns the coordinates of the regular n-simplex appended with a constant and normalized to unit length:

$$\mathbf{M}_n = \left[\mathbf{m}_i\right]_{i=1}^{n+1}\,,\ \ \mathbf{m}_i = \frac{1}{p}\begin{bmatrix}\mathbf{p}_i \\ n^{-1/2}\end{bmatrix},\ \ p = \left\|\begin{bmatrix}\mathbf{p}_i \\ n^{-1/2}\end{bmatrix}\right\|, \tag{10}$$

where the norms p are constant, since $\|\mathbf{p}_i\| = \|\mathbf{p}_j\|$ for all i and j by definition of a regular simplex.

Proposition 1. *Let $\mathbf{M}_n$ be the-change-of-basis matrix defined in* (10)*. Then $\mathbf{M}_n$ is an $(n+1)$D rotation or reflection,* i.e.*, $\mathbf{M}_n \in \mathrm{O}(n+1)$ (see Section B in the Appendix for numeric examples).*

Proof. We want to show that $\mathbf{M}_n^\top \mathbf{M}_n = \mathbf{I}_{n+1}$, which will prove that $\mathbf{M}_n$ is orthogonal. The diagonal elements of $\mathbf{M}_n^\top \mathbf{M}_n$ are $\mathbf{m}_i^\top \mathbf{m}_i = \|\mathbf{m}_i\|^2 = 1$ since $\|\mathbf{m}_i\| = 1$. The off-diagonal elements are found as $\mathbf{m}_i^\top \mathbf{m}_j = (\mathbf{p}_i^\top \mathbf{p}_j + n^{-1})/p^2$ for $i \neq j$, where p is defined in (10). Note that $\mathbf{p}_i^\top \mathbf{p}_j$ is the same for all i and j with $i \neq j$ since, by definition of a regular simplex, the vertices $\mathbf{p}_i$ are spaced uniformly. Note that $\mathbf{p}_i^\top \mathbf{p}_j = -n^{-1}$ for all $i \neq j$ by definition (8). Hence, the off-diagonal elements of $\mathbf{M}_n^\top \mathbf{M}_n$ are zeros and $\mathbf{M}_n^\top \mathbf{M}_n = \mathbf{I}_{n+1}$. □

Since $\mathbf{M}_n \in \mathrm{O}(n+1)$, the sign of $\det \mathbf{M}_n$ is determined by the number of reflections required to form the transformation. In the case of a regular n-simplex, the sign of the determinant depends on the parity of n **and** the configuration of the simplex vertices. In our case, $\mathbf{M}_n$ is a rotation for odd n, i.e., $\det \mathbf{M}_n = 1$, and a reflection for even n. Consider, for example, the case $n = 3$. The matrix $\mathbf{M}_3$ shown in (7) has $\det \mathbf{M}_3 = 1$, thus, is a 4D rotation, as stated in Section 3.3.

Lemma 2. *Let $\mathbf{M}_n$ be the change-of-basis matrix defined in* (10)*, and $\mathbf{P}_n$ an $n \times (n+1)$ matrix holding the regular n-simplex vertices, $\mathbf{p}_i$, in its columns, and $p = \left\|\begin{bmatrix}\mathbf{p}_i \\ n^{-1/2}\end{bmatrix}\right\|$, as defined in* (10)*. Then*

$$\mathbf{M}_n \mathbf{P}_n^\top = p\begin{bmatrix}\mathbf{I}_n \\ \mathbf{0}^\top\end{bmatrix}. \tag{11}$$

Proof. We begin by elaborating on (10):

$$\mathbf{M}_n = \frac{1}{p}\begin{bmatrix}\mathbf{P}_n \\ n^{-1/2}\,\mathbf{1}^\top\end{bmatrix}. \tag{12}$$

We note that the norms of the rows of $\mathbf{P}_n$ are also equal to p since $\mathbf{M}_n \in \mathrm{O}(n+1)$ (as per Proposition 1). Recall that $\mathbf{P}_n$ is centered at the origin, and, therefore, for a constant $a \in \mathbb{R}$ and a vector of ones $\mathbf{1} \in \mathbb{R}^{n+1}$, we obtain $a\,\mathbf{1}^\top \mathbf{P}_n^\top = \mathbf{0}^\top$.

Remembering that the product $\mathbf{M}_n\mathbf{P}_n^\top$ is between $\mathbb{R}^{n+1}$ vectors, we plug (12) into the LHS of (11) and obtain

$$\mathbf{M}_n\mathbf{P}_n^\top = \frac{1}{p}\begin{bmatrix}\mathbf{P}_n\\ n^{-1/2}\,\mathbf{1}^\top\end{bmatrix}\mathbf{P}_n^\top = \frac{p^2}{p}\begin{bmatrix}\mathbf{I}_n\\ \mathbf{0}^\top\end{bmatrix} = p\begin{bmatrix}\mathbf{I}_n\\ \mathbf{0}^\top\end{bmatrix}. \tag{13}$$

□

4.2. Equivariant nD Spheres

In this section, we generalize steerable 3D spherical neurons reviewed in Section 3.3. We denote an equivariant nD-sphere neuron (an *equivariant hypersphere*) by means of the $(n+1)\times(n+2)$ matrix $B_n(\boldsymbol{S})$ for the spherical decision surface $\boldsymbol{S}\in\mathbb{R}^{n+2}$ with center $\mathbf{c}_0\in\mathbb{R}^n$ and an nD input $\mathbf{x}\in\mathbb{R}^n$ embedded as $\boldsymbol{X}\in\mathbb{R}^{n+2}$ as

$$\begin{aligned}\mathbf{F}_n(\boldsymbol{X};\boldsymbol{S}) &= B_n(\boldsymbol{S})\,\boldsymbol{X}\,,\\ B_n(\boldsymbol{S}) &= \Big[(\boldsymbol{R}_O^\top\,\boldsymbol{R}_{T_i}\,\boldsymbol{R}_O\,\boldsymbol{S})^\top\Big]_{i=1}^{n+1}\,,\end{aligned} \tag{14}$$

where $\{\boldsymbol{R}_{T_i}\}_{i=1}^{n+1}$ is the $\mathbb{R}^{n+2}$ rotation isomorphism corresponding to the rotation from the first vertex to the i-th vertex of the regular n-simplex, and $\boldsymbol{R}_O\in\mathrm{SO}(n)$ is the geodesic (shortest) rotation[4] from the sphere center $\mathbf{c}_0$ to $\|\mathbf{c}_0\|\,\mathbf{p}_1$. Therefore, $\boldsymbol{R}_{T_1}=\mathbf{I}_{n+2}$. (Technically, if the center $\mathbf{c}_0$ happens to be $-\mathbf{p}_1$, $\boldsymbol{R}_O$ is a reflection about the origin. In principle, we could just as well write $\boldsymbol{R}_O\in\mathrm{O}(n)$, since it makes no difference in our further derivations.)

We now need to prove that $\mathbf{F}_n(\,\cdot\,;\boldsymbol{S})$ is $\mathrm{O}(n)$-equivariant.

Proposition 3. *Let $\mathbf{F}_n(\,\cdot\,;\boldsymbol{S}):\mathbb{R}^{n+2}\to\mathbb{R}^{n+1}$ be the neuron defined in* (14) *and $\boldsymbol{R}\in\mathrm{O}(n)$ be an $n\times n$ rotation or reflection matrix. Then the transformation representation in the filter output space $\mathbb{R}^{n+1}$ is given by the $(n+1)\times(n+1)$ matrix*

$$V_n = \rho\,(\boldsymbol{R}) = \mathbf{M}_n^\top\boldsymbol{R}_O\,\boldsymbol{R}\,\boldsymbol{R}_O^\top\mathbf{M}_n\,, \tag{15}$$

where $\mathbf{M}_n\in\mathrm{O}(n+1)$ is the-change-of-basis matrix defined in (10) *and a* 1 *is appended to the diagonals of $\boldsymbol{R}_O$ and $\boldsymbol{R}$ to make them $(n+1)\times(n+1)$. Furthermore, $V_n\in G<\mathrm{O}(n+1)$.*

Proof. Since $\mathbf{M}_n\in\mathrm{O}(n+1)$, $\boldsymbol{R}_O\in\mathrm{SO}(n)$, and $\boldsymbol{R}\in\mathrm{O}(n)$ are orthogonal matrices, V_n in (15) is an orthogonal change-of-basis transformation that represents $\boldsymbol{R}\in\mathrm{O}(n)$ in the basis constructed by $\mathbf{M}_n$ and $\boldsymbol{R}_O$. Note that appending one to the diagonal of $\boldsymbol{R}\in\mathrm{O}(n)$ does not affect the sign of the determinant, which makes V_n a reflection representation if $\det\boldsymbol{R}=-1$, or a rotation representation if $\det\boldsymbol{R}=+1$. Since $\boldsymbol{R}\in\mathrm{O}(n)$ and $\boldsymbol{R}_O\in\mathrm{O}(n)$ embedded in the $(n+1)$D (by extending the matrix diagonal with 1), not all elements of $\mathrm{O}(n+1)$ can be generated by the operation in (15). Thus, we conclude that V_n belongs to a proper subgroup of $\mathrm{O}(n+1)=\mathrm{O}(n)\times\mathcal{S}^n$, i.e., $G<\mathrm{O}(n+1)$, where G is formed as $\mathrm{O}(n)\times\mathbf{M}_n\Big(\frac{\mathbf{c}_0}{\|\mathbf{c}_0\|},0\Big)$ with $\mathbf{M}_n\Big(\frac{\mathbf{c}_0}{\|\mathbf{c}_0\|},0\Big)\in\mathcal{S}^n$.

The original transformation is found directly as

$$\boldsymbol{R} = \boldsymbol{R}_O^\top\mathbf{M}_n\,V_n\,\mathbf{M}_n^\top\boldsymbol{R}_O\,, \tag{16}$$

followed by the retrieval of the upper-left $n\times n$ sub-matrix. □

Noteworthy, the basis determined by $\boldsymbol{R}_O\in\mathrm{SO}(n)$, which depends on the center $\mathbf{c}_0$ of the sphere $\boldsymbol{S}\in\mathbb{R}^{n+2}$ (see (14)), will be different for different $\mathbf{c}_0$. Therefore, the representation V_n will differ as well.

Theorem 4. *The neuron $\mathbf{F}_n(\,\cdot\,;\boldsymbol{S}):\mathbb{R}^{n+2}\to\mathbb{R}^{n+1}$ defined in* (14) *is $\mathrm{O}(n)$-equivariant.*

Proof. To prove the theorem, we need to show that (3) holds for $\mathbf{F}_n(\,\cdot\,;\boldsymbol{S})$.

We substitute (15) to the LHS and (14) to the RHS, and obtain

$$V_n\,B_n(\boldsymbol{S})\,\boldsymbol{X} = B_n(\boldsymbol{S})\,\boldsymbol{R}\boldsymbol{X}\,. \tag{17}$$

For the complete proof, please see the Appendix (refer to Section C). □

We note that with respect to the input vector $\mathbf{x}\in\mathbb{R}^n$, the equivariant hypersphere $\mathbf{F}_n(\,\cdot\,;\boldsymbol{S}):\mathbb{R}^{n+2}\to\mathbb{R}^{n+1}$ represents a non-linear $\mathrm{O}(n)$-equivariant function. It is also worth mentioning that the *sum* of the output $\boldsymbol{Y}=B_n(\boldsymbol{S})\,\boldsymbol{X}$ is an $\mathrm{O}(n)$-invariant scalar, i.e., the DC-component, due to the regular n-simplex construction.

This invariant part can be adjusted by adding a scalar *bias* parameter to the output $\boldsymbol{Y}$. The concept of bias is imperative for linear classifiers, but for spherical decision surfaces (Perwass et al., 2003), it is implicitly modeled by the embedding (1). We note, however, that adding a scalar bias parameter, $b\in\mathbb{R}$ to the output of an equivariant hypersphere (14) respects $\mathrm{O}(n)$-equivariance:

Proposition 5. *Let $\boldsymbol{Y}\in\mathbb{R}^{n+1}$ be the output of the $\mathrm{O}(n)$-equivariant hypersphere $\mathbf{F}_n(\,\cdot\,;\boldsymbol{S}):\mathbb{R}^{n+2}\to\mathbb{R}^{n+1}$* (14) *given the input $\boldsymbol{X}\in\mathbb{R}^{n+2}$, and $b\in\mathbb{R}$ be a bias parameter. Then $\boldsymbol{Y}'=\boldsymbol{Y}+b\,\mathbf{1}$, where $\mathbf{1}$ is the vector of ones in $\mathbb{R}^{n+1}$, is also $\mathrm{O}(n)$-equivariant.*

Proof. We need to show that (17) also holds when the bias b is added. First, we use V_n—the representation of $\boldsymbol{R}\in\mathrm{O}(n)$ from (15)—and the fact that $\boldsymbol{R}$ and $\boldsymbol{R}_O$ are both appended 1 to their main diagonal to make them $(n+1)\times(n+1)$. Then $V_n\,\mathbf{1} = \mathbf{M}_n^\top\boldsymbol{R}_O\,\boldsymbol{R}\,\boldsymbol{R}_O^\top\mathbf{M}_n\mathbf{1} = \mathbf{M}_n^\top\boldsymbol{R}_O\,\boldsymbol{R}\,\boldsymbol{R}_O^\top\begin{bmatrix}\mathbf{0}\\ p\sqrt{n}\end{bmatrix} =$

[4] In practice, we compute it utilizing the Householder (double-) reflection method, e.g., as described by Golub & Van Loan (2013).

$\mathbf{M}_n^\top \begin{bmatrix} \mathbf{0} \\ p\sqrt{n} \end{bmatrix} = \mathbf{1}$, where p is a scalar defined in (8). Since the bias b is a scalar, we use that $V_n\, b\mathbf{1} = bV_n\, \mathbf{1} = b\mathbf{1}$. We now consider the left-hand side of (17): $V_n\, \boldsymbol{Y}' = V_n\,(\boldsymbol{Y} + b\mathbf{1}) = V_n\, B_n(\boldsymbol{S})\, \boldsymbol{X} + V_n\, b\mathbf{1} = V_n\, B_n(\boldsymbol{S})\, \boldsymbol{X} + b\mathbf{1}$. Plugging the equality (17) into the last equation, we complete the proof: $V_n\, B_n(\boldsymbol{S})\, \boldsymbol{X} + b\mathbf{1} = B_n(\boldsymbol{S})\, \boldsymbol{R}\boldsymbol{X} + b\mathbf{1}$. □

This result allows us to increase the capacity of the equivariant hypersphere by adding the learnable parameter $b \in \mathbb{R}$. In addition, note that all $V_n \in G < \mathrm{O}(n+1)$ can be characterized by the fact that they all have an eigenvector equal to $\frac{1}{\sqrt{n+1}}\mathbf{1}$, where $\mathbf{1}$ is the vector of ones in $\mathbb{R}^{n+1}$.

4.3. Normalization and Additional Non-Linearity

An important practical consideration in deep learning is feature normalization (Ioffe & Szegedy, 2015; Ba et al., 2016). We show how the activations of the equivariant hypersphere (14) can be normalized maintaining the equivariance:

Proposition 6. *Let $\boldsymbol{Y} \in \mathbb{R}^{n+1}$ be the $\mathrm{O}(n)$-equivariant output of the hypersphere filter* (14)*. Then $\boldsymbol{Y}/\|\boldsymbol{Y}\|$, where $\|\boldsymbol{Y}\| \in \mathbb{R}$, is also $\mathrm{O}(n)$-equivariant.*

Proof. Since $\boldsymbol{Y}$ is $\mathrm{O}(n)$-equivariant, $\|\boldsymbol{Y}\|$ is $\mathrm{O}(n)$-invariant (the length remains unchanged under $\mathrm{O}(n)$-transformations). Hence, $\boldsymbol{Y}/\|\boldsymbol{Y}\|$ is also $\mathrm{O}(n)$-equivariant. □

To increase the descriptive power of the proposed approach, we can add non-linearity to the normalization step, following Ruhe et al. (2023):

$$\boldsymbol{Y} \mapsto \frac{\boldsymbol{Y}}{\sigma(a)\,(\|\boldsymbol{Y}\| - 1) + 1}, \tag{18}$$

where $a \in \mathbb{R}$ is a learnable scalar and $\sigma(\cdot)$ is the sigmoid function.

4.4. Extracting Deep Equivariant Features

We might want to propagate the equivariant output of $\mathbf{F}_n$ (14), $\boldsymbol{Y} = B_n(\boldsymbol{S})\, \boldsymbol{X}$, through spherical decision surfaces while maintaining the equivariance properties. One way to achieve it is by using $(n+1)$D spheres, i.e., $\mathbf{F}_{n+1}$, since the output $\boldsymbol{Y} \in \mathbb{R}^{n+1}$. Thus, the results established in the previous section not only allow us to use the equivariant hyperspheres (14) for nD inputs but also to cascade them in multiple layers, thus propagating equivariant representations by successively incrementing the feature space dimensionality with a unit step, i.e., $n\mathrm{D} \to (n+1)\mathrm{D}$.

Consider, for example, the point cloud patch $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ consisting of the coordinates of N points $\mathbf{x} \in \mathbb{R}^n$ as the input signal, which we can also consider as the $N \times n$ matrix $\mathbf{X}$. Given the equivariant neuron $\mathbf{F}_n(\,\cdot\,; \boldsymbol{S})$, a *cascaded* $n\mathrm{D} \to (n+1)\mathrm{D}$ feature extraction procedure using equivariant hyperspheres $\mathbf{F}_n(\,\cdot\,; \boldsymbol{S})$ for the given output dimensionality d (with $d > n$) can be defined as follows (at the first step, $\boldsymbol{X} \leftarrow \mathbf{x}$):

$$\begin{aligned} &\boldsymbol{X} \in \mathbb{R}^n \to \texttt{embed}(\texttt{normalize}(\boldsymbol{X})) \to \mathbf{F}_n(\boldsymbol{X}; \boldsymbol{S}) \to \\ &\texttt{embed}(\texttt{normalize}(\boldsymbol{X} + b)) \to \mathbf{F}_{n+1}(\boldsymbol{X}; \boldsymbol{S}) \to \\ &\ldots \to \mathbf{F}_d(\boldsymbol{X}; \boldsymbol{S}) \to \texttt{normalize}(\boldsymbol{X} + b) \to \boldsymbol{X} \in \mathbb{R}^d\,, \end{aligned} \tag{19}$$

where `embed` is the embedding according to (1), `normalize` is the optional activation normalization (see Proposition 6), and b is an optional scalar bias.

Proposition 7. *Given that all operations involved in the procedure 19 are $\mathrm{O}(n)$-equivariant, its output will also be $\mathrm{O}(n)$-equivariant.*

The proof is given in the Appendix (Section C).

Thus, given $\mathcal{X}$ as input, the point-wise cascaded application with depth d (19) produces the equivariant features $\mathcal{Y} = \{\boldsymbol{Y}_i\}_{i=1}^N$, $\boldsymbol{Y} \in \mathbb{R}^{n+d}$, which we can consider as the $N \times (n+d)$ matrix $\mathbf{Y}$.

In this case, we considered the width of each layer in (19) to be 1, i.e., one equivariant hypersphere. In practice and depending on the task, we often use K_l equivariant hyperspheres per layer l, with suitable connectivity between subsequent layers.

4.5. Modelling Higher-Order Interactions

The theoretical framework established above considers the interaction of *one* point and one spherical decision surface, copied to construct the regular n-simplex constellation for the equivariant neuron in (14). To increase the expressiveness of a model comprised of equivariant hyperspheres, we propose to consider the relation of *two* points and a sphere.

Namely, following the work of Li et al. (2001)[5], the relation between two points, $\mathbf{x}_1$, $\mathbf{x}_2$, and a sphere in $\mathbb{R}^n$, all embedded in $\mathbb{R}^{n+2}$ according to (1) as $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, and $\boldsymbol{S}$, respectively, is formulated as

$$\delta = e_{12}\, \boldsymbol{X}_1^\top \boldsymbol{S}\, \boldsymbol{X}_2^\top \boldsymbol{S}, \tag{20}$$

where $e_{12} := -\frac{1}{2}\left(\|\mathbf{x}_1 - \mathbf{x}_2\|^2\right) \in \mathbb{R}$ models the edge as the squared Euclidean distance between the points.

To classify the relative position of the points to the sphere, we use the sign of $\delta \in \mathbb{R}$, and note that it is only determined by the respective sphere (scalar) activations, i.e., the scalar products $\boldsymbol{X}_i^\top \boldsymbol{S}$, since the edges e_{ij} are always negative. Thus, we may omit them, as we further demonstrate by the ablations in the Appendix (see Section E). Also, we note that in order to make δ an invariant quantity, we need to have *equivariant* activations. Since the single sphere activations

[5] See p. 22 in Li et al. (2001).

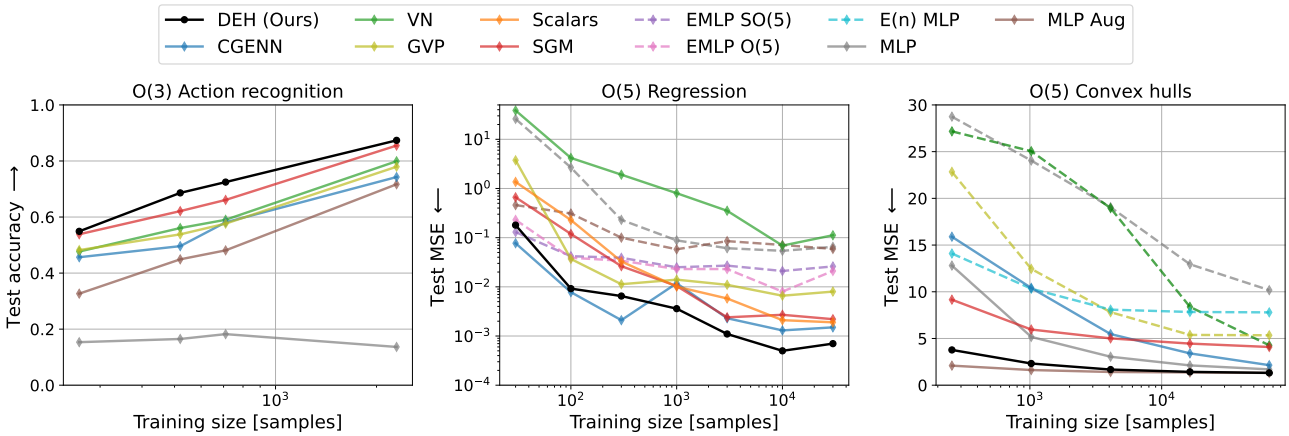


Figure 2. Left: real data experiment (the higher the accuracy the better); all the presented models are also permutation-invariant. Center and right: synthetic data experiments (the lower the mean squared error (MSE) the better); dotted lines mean that the results of the methods are copied from Finzi et al. (2021) (O(5) regression) or Ruhe et al. (2023) (O(5) convex hulls). Best viewed in color.

are not equivariant (see Section 3.1), we propose to substitute the single sphere $\boldsymbol{S}$ with our equivariant hyperspheres $B_n(\boldsymbol{S})$ (14).

Given the input $\mathbf{X} \in \mathbb{R}^{N\times n}$ and the corresponding extracted equivariant features $\mathbf{Y} \in \mathbb{R}^{N\times(n+d)}$, we compute

$$\Delta = \mathbf{X}^\top B_n(\boldsymbol{S})\,(\mathbf{X}^\top B_n(\boldsymbol{S}))^\top = \mathbf{Y}\,\mathbf{Y}^\top. \tag{21}$$

The O(n)-invariance of $\Delta \in \mathbb{R}^{N\times N}$ follows from the fact that it is comprised of the Gram matrix $\mathbf{Y}\,\mathbf{Y}^\top$ that consists of the pair-wise inner products of equivariant features, which are invariant (Deng et al., 2021; Melnyk et al., 2024), just as in the case of directly computing the auto-product of the points (Xu et al., 2021). When permutation-invariance is desired, we achieve it by aggregation over the points, first following the procedure by Xu et al. (2021) and sorting the rows/columns of Δ, and then applying max and/or mean pooling over N. If multiple (K_l) equivariant hyperspheres per layer are used, (21) is computed independently for all K_l, by computing K_l Gram matrices, resulting in $\Delta \in \mathbb{R}^{N\times N\times K_l}$.

We show the effectiveness of the proposed invariant operator (21) by the corresponding ablation in the Appendix (Section E).

5. Experimental Validation

In this section, we experimentally verify our theoretical results derived in Section 4 by evaluating our `Deep Equivariant Hyperspheres`, constituting feed-forward point-wise architectures, on real and synthetic O(n)-equivariant benchmarks. In each experiment, we train the models using the same hyperparameters and present the test-set performance of the models chosen based on their validation-set performance.

For the sake of a fair comparison, all the models have approximately the same number of learnable parameters, and their final fully-connected layer part is the same. A more detailed description of the used architectures is presented in the Appendix (see Section D). In addition to the performance comparison in Figure 2, we compare the time complexity (i.e., the inference speed) of the considered methods[6] in Figure 3. Furthermore, we present various ablations in the Appendix (see Section E). All the models are implemented in PyTorch (Paszke et al., 2019).

5.1. O(3) Action Recognition

First, we test the ability of our method to utilize O(3)-equivariance as the inductive bias. For this experiment, we select the task of classifying the 3D skeleton data, presented and extracted by Melnyk et al. (2022) from the UTKinect-Action3D dataset by Xia et al. (2012). Each skeleton is a 20×3 point cloud, belonging to one of the 10 action categories; refer to the work of Melnyk et al. (2022) for details. We formulate the task to be both permutation- and O(3)-invariant.

We construct an O(3)-equivariant point-wise feedforward model using layers with our equivariant hyperspheres (according to the blueprint of (19)) with the two-point interaction described in Section 4.5, which we call DEH (see the illustration in Figure 4). We also build a variant of the invariant SGM descriptor (Xu et al., 2021) computing the Gram matrix of the input points, point-wise equivariant VN (Deng et al., 2021), GVP (Jing et al., 2021), and CGENN

[6]Some of the results in Figure 2 are copied from Ruhe et al. (2023) since the implementation of the specific versions of some models is currently unavailable. Therefore, we could not measure their inference speed.

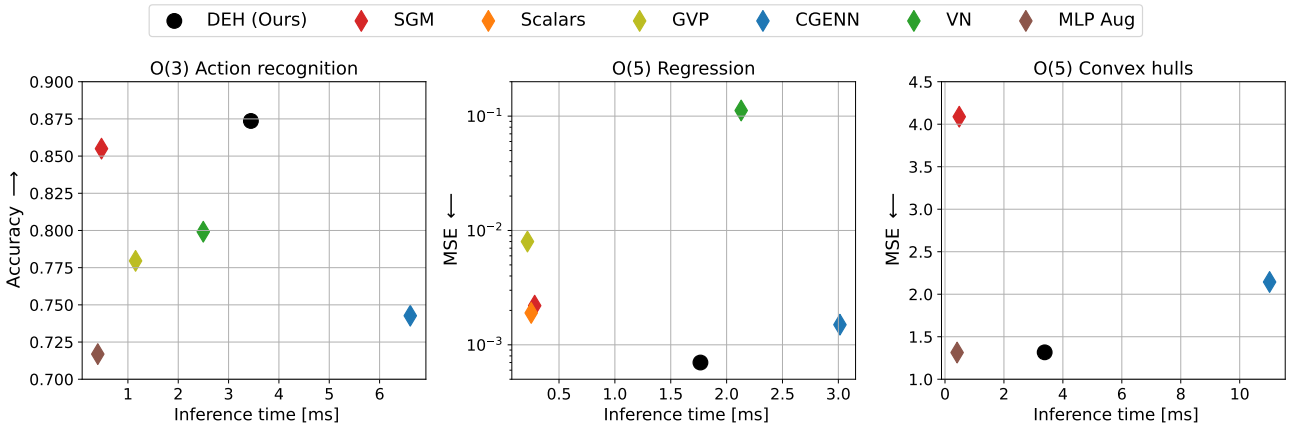


Figure 3. Speed/performance trade-off (the models are trained on all the available training data). Note that the desired trade-off is toward the top-left corner (higher accuracy and faster inference) in the left figure, and toward the bottom-left corner (lower error and faster inference) in the center and right figures. To measure inference time, we used an NVIDIA A100. Best viewed in color.

(Ruhe et al., 2023) models and, as non-equivariant baselines, point-wise MLPs, in which the equivariant layers are substituted with regular non-linear ones. We train one version of the baseline MLP with $\mathrm{O}(3)$-augmentation, whereas our method is *only trained on non-transformed skeletons*.

We evaluate the performance of the methods on the randomly $\mathrm{O}(3)$-transformed test data. The results are presented in Figure 2 (left): our DEH model, trained on the data in a single orientation, captures equivariant features that enable outperforming the non-equivariant baseline trained on the augmented data (MLP Aug). Moreover, DEH consistently outperforms the competing equivariant methods (VN, GVP, CGENN) and the invariant SGM model, demonstrating a favorable speed/performance trade-off, as seen in Figure 3 (left).

5.2. $\mathrm{O}(5)$ Regression

Originally introduced by Finzi et al. (2021), the task is to model the $\mathrm{O}(5)$-invariant function $f(\mathbf{x}_1, \mathbf{x}_2) := \sin(\|\mathbf{x}_1\|) - \|\mathbf{x}_2\|^3/2 + \frac{\mathbf{x}_1^\top \mathbf{x}_2}{\|\mathbf{x}_1\|\|\mathbf{x}_2\|}$, where the two vectors $\mathbf{x}_1 \in \mathbb{R}^5$ and $\mathbf{x}_2 \in \mathbb{R}^5$ are sampled from a standard Gaussian distribution to construct train, validation, and test sets. We use the same training hyperparameters and evaluation setup as Ruhe et al. (2023).

Here, we employ a DEH architecture similar to that in Section 5.1, and compare it to the equivariant EMLPs (Finzi et al., 2021), CGENN, VN, and GVP, and non-equivariant MLPs. Refer to the Appendix (Section D) for the architecture details. Our results together with those of the related methods are presented in Figure 2 (center). As we can see, our DEH is more stable than CGENN, as shown by the dependency over the training set size, and outperforms it in most cases. Our method also outperforms the vanilla MLP and the MLP trained with augmentation (MLP Aug), as well as the $\mathrm{O}(5)$- and $\mathrm{SO}(5)$-equivariant EMLP and VN, and the invariant SGM and Scalars Villar et al. (2021)[7] methods.

5.3. $\mathrm{O}(5)$ Convex Hull Volume Prediction

Our third experiment addresses the task of estimating the volume of the convex hull generated by 16 5D points, described by Ruhe et al. (2023). The problem is $\mathrm{O}(5)$-invariant in nature, i.e., rotations and reflections of a convex hull do not change its volume. We exploit the same network architecture as in Section 5.1 (see the Appendix for details).

We present our results alongside those of the related methods in Figure 2: our DEH model outperforms all of the equi- and invariant competing methods (including an MLP version of the $\mathrm{E}(n)$-equivariant approach Satorras et al. (2021)) and in all the scenarios, additionally exhibiting a superior speed/performance trade-off, as seen in Figure 3 (right). We outperform Ruhe et al. (2023) as well as their MLP result. However, our point-wise MLP implementation slightly outperforms our method on low-data regimes.

6. Conclusion

In this manuscript, we presented `Deep Equivariant Hyperspheres` — nD neurons based on spheres and regular n-simplexes — equivariant under orthogonal transformations of dimension n. We defined and analyzed generalized components for a network composed of the proposed neurons, such as equivariant bias, non-linearity, and multi-

[7]For this task, permutation-invariance is not required, and Villar et al. (2021) only used 3 unique elements of the Gram matrix constructed from the input. For the other two tasks, we refer to the results we obtained with SGM (where the entire Gram matrix with sorted rows is used).

layer configuration (see Section 4 and the ablations in the Appendix).

In addition, we proposed the invariant operator (21) modeling the relation between two points and a sphere, inspired by the work of Li et al. (2001), and demonstrated its effectiveness (see the Appendix). We evaluated our method on both synthetic and real-world data and demonstrated the utility of the developed theoretical framework in nD by outperforming the competing methods and achieving a favorable speed/performance trade-off (see Figure 3). Investigating the design of more advanced equivariant architectures of the proposed equivariant hyperspheres forms a clear direction for future work.

Limitations

The focus of this paper is on the $\mathrm{O}(n)$-equivariance, with $n > 3$, and our model architecture employed in the experiments was designed to compare with recent related methods, e.g., Finzi et al. (2021); Ruhe et al. (2023), using the small-scale $\mathrm{O}(5)$ tasks presented in them. To address scalability, one could wrap our equivariant hyperspheres in some kind of graph neural network (GNN) framework, e.g., DGCNN (Wang et al., 2019), thus utilizing the spheres in local neighborhoods.

Additionally, all the tasks considered in the experiments require invariant model output. Therefore in the current version, we perform modeling of the interaction between the points using their $\mathrm{O}(n)$-equivariant features (outputs of the proposed equivariant hyperspheres) only to produce invariant features (21). For tasks that require equivariant output, not considered in this work, this interaction needs to happen such that the equivariance is preserved (e.g., using Lemma 3 by Villar et al. (2021)).

Finally, if translation equivariance is additionally desired, i.e., the full $\mathrm{E}(n)$ group, a common way to address this is by centering the input point set by subtracting the mean vector, and then adding this vector to the model output.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here. An exception is possibly the area of molecular physics with applications in material science; the development of new materials might have a significant impact on sustainability.

Acknowledgments

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), by the Swedish Research Council through a grant for the project Uncertainty-Aware Transformers for Regression Tasks in Computer Vision (2022-04266), and the strategic research environment ELLIIT. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) partially funded by the Swedish Research Council through grant agreement no. 2022-06725, and by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre.

We highly appreciate the useful feedback from the reviewers. We thank Erik Darpö from the Department of Mathematics, Linköping University, for sharing his insights on group theory with us.

References

Anderson, B., Hy, T. S., and Kondor, R. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, pp. 14510–14519, 2019.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Cevikalp, H. and Saribas, H. Deep simplex classifier for maximizing the margin in both euclidean and angular spaces. In *Image Analysis: 23rd Scandinavian Conference, SCIA 2023, Sirkka, Finland, April 18–21, 2023, Proceedings, Part II*, pp. 91–107. Springer, 2023.

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. Spherical CNNs. *arXiv preprint arXiv:1801.10130*, 2018.

Coors, B., Condurache, A. P., and Geiger, A. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *European Conference on Computer Vision (ECCV)*, September 2018.

Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., and Guibas, L. J. Vector neurons: A general framework for SO(3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12200–12209, 2021.

On Learning Deep O(n)-Equivariant Hyperspheres

Elte, E. L. *The Semiregular Polytopes of the Hyperspaces*. United States: University of Michigan Press, 2006. ISBN 9781418179687. URL https://books.google.se/books?id=9SxtPwAACAAJ.

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. Learning SO(3) Equivariant Representations with Spherical CNNs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Finzi, M., Welling, M., and Wilson, A. G. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International conference on machine learning*, pp. 3318–3328. PMLR, 2021.

Freeman, W. T., Adelson, E. H., et al. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

Fuchs, F., Worrall, D., Fischer, V., and Welling, M. Se(3)-transformers: 3d roto-translation equivariant attention networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1970–1981. Curran Associates, Inc., 2020.

Golub, G. H. and Van Loan, C. F. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 4 edition, 2013. ISBN 978-1-4214-0794-4. doi: 10.56021/9781421407944.

Hilbert, D. and Cohn-Vossen, S. *Geometry and the Imagination*. Chelsea Publishing Company, New York, 1952.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Jing, B., Eismann, S., Suriana, P., Townshend, R. J. L., and Dror, R. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=1YLJDvSx6J4.

Knutsson, H., Haglund, L., Bårman, H., and Granlund, G. H. A framework for anisotropic adaptive filtering and analysis of image sequences and volumes. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pp. 469–472 vol.3, 1992. doi: 10.1109/ICASSP.1992.226174.

Lan, J., Palizhati, A., Shuaibi, M., Wood, B. M., Wander, B., Das, A., Uyttendaele, M., Zitnick, C. L., and Ulissi, Z. W. Adsorbml: Accelerating adsorption energy calculations with machine learning. *arXiv preprint arXiv:2211.16486*, 2022.

Li, H., Hestenes, D., and Rockwood, A. Generalized homogeneous coordinates for computational geometry. In *Geometric Computing with Clifford Algebras*, pp. 27–59. Springer, 2001.

Melnyk, P., Felsberg, M., and Wadenbäck, M. Embed Me if You Can: A Geometric Perceptron. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1276–1284, 2021.

Melnyk, P., Felsberg, M., and Wadenbäck, M. Steerable 3D Spherical Neurons. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15330–15339. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/melnyk22a.html.

Melnyk, P., Robinson, A., Felsberg, M., and Wadenbäck, M. TetraSphere: A Neural Descriptor for O(3)-Invariant Point Cloud Analysis. *arXiv preprint arXiv:2211.14456, Accepted at CVPR 2024*, 2024.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Perraudin, N., Defferrard, M., Kacprzak, T., and Sgier, R. Deepsphere: Efficient spherical convolutional neural network with healpix sampling for cosmological applications. *Astronomy and Computing*, 27:130–146, 2019.

Perwass, C., Banarer, V., and Sommer, G. Spherical decision surfaces using conformal modelling. In *Joint Pattern Recognition Symposium*, pp. 9–16. Springer, 2003.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Ruhe, D., Brandstetter, J., and Forré, P. Clifford Group Equivariant Neural Networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=n84bzMrGUD.

Rupp, M., Tkatchenko, A., Müller, K.-R., and Von Lilienfeld, O. A. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

10

Su, Y.-C. and Grauman, K. Learning spherical convolution for fast features from 360 imagery. *Advances in Neural Information Processing Systems*, 30, 2017.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation-and translation-equivariant neural networks for 3D point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Townshend, R. J., Vögele, M., Suriana, P., Derry, A., Powers, A., Laloudakis, Y., Balachandar, S., Jing, B., Anderson, B., Eismann, S., et al. Atom3d: Tasks on molecules in three dimensions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Villar, S., Hogg, D. W., Storey-Fisher, K., Yao, W., and Blum-Smith, B. Scalars are universal: Equivariant machine learning, structured like classical physics. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=ba27-RzNaIv.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.*, 38(5), oct 2019. ISSN 0730-0301. doi: 10.1145/3326362.

Xia, L., Chen, C.-C., and Aggarwal, J. K. View invariant human action recognition using histograms of 3d joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 20–27, 2012. doi: 10.1109/CVPRW.2012.6239233.

Xu, J., Tang, X., Zhu, Y., Sun, J., and Pu, S. SGMNet: Learning rotation-invariant point cloud representations via sorted Gram matrix. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10468–10477, 2021.

A. Additional Background

A.1. Steerability

According to Freeman et al. (1991), a function is called *steerable* if it can be written as a linear combination of rotated versions of itself, as also alternatively presented by Knutsson et al. (1992). In 3D, $f^{\boldsymbol{R}}(x, y, z)$ is thus said to steer if

$$f^{\boldsymbol{R}}(x, y, z) = \sum_{j=1}^{M} v_j(\boldsymbol{R}) f^{\boldsymbol{R}_j}(x, y, z) \ , \tag{22}$$

where $f^{\boldsymbol{R}}(x, y, z)$ is $f(x, y, z)$ rotated by $\boldsymbol{R} \in \mathrm{SO}(3)$, and each $\boldsymbol{R}_j \in \mathrm{SO}(3)$ orients the corresponding jth basis function.

Freeman et al. (1991) further describe the conditions under which the 3D steerability constraint (22) holds and how to find the minimum number of basis functions, that must be uniformly distributed in space.

In this context, Melnyk et al. (2022) showed that in order to steer a spherical neuron defined in (2) (Perwass et al., 2003; Melnyk et al., 2021), one needs to have a minimum of fours basis functions, i.e., rotated versions of the original spherical neuron. This, together with the condition of the uniform distribution of the basis functions, leads to the regular tetrahedron construction of the steerable 3D spherical neuron in (5).

B. Numeric Instances for $n = \{2, 3, 4\}$

To facilitate the reader's understanding of the algebraic manipulations in the next section, herein, we present numeric instances of the central components of our theory defined in (8) and (10), for the cases $n = 2$, $n = 3$, and $n = 4$. For convenience, we write the vertices of the regular simplex (8) as the $n \times (n+1)$ matrix $\mathbf{P}_n = \big[\mathbf{p}_i\big]_{i=1\ldots n+1}$.

$$n = 2: \quad \mathbf{P}_2 = \tfrac{1}{\sqrt{2}} \begin{bmatrix} 1 & (\sqrt{3}-1)/2 & -(\sqrt{3}+1)/2 \\ 1 & -(\sqrt{3}+1)/2 & (\sqrt{3}-1)/2 \end{bmatrix}, \qquad p = \sqrt{3/2},$$

$$\mathbf{M}_2 = \tfrac{1}{\sqrt{3}} \begin{bmatrix} 1 & (\sqrt{3}-1)/2 & -(\sqrt{3}+1)/2 \\ 1 & -(\sqrt{3}+1)/2 & (\sqrt{3}-1)/2 \\ 1 & 1 & 1 \end{bmatrix}.$$

$$n = 3: \quad \mathbf{P}_3 = \tfrac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}, \qquad p = 2/\sqrt{3},$$

$$\mathbf{M}_3 = \tfrac{1}{2} \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

$$n = 4: \quad \mathbf{P}_4 = \tfrac{1}{2} \begin{bmatrix} 1 & (3\sqrt{5}-1)/4 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 \\ 1 & -(\sqrt{5}+1)/4 & (3\sqrt{5}-1)/4 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 \\ 1 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 & (3\sqrt{5}-1)/4 & -(\sqrt{5}+1)/4 \\ 1 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 & (3\sqrt{5}-1)/4 \end{bmatrix},$$

$$p = \sqrt{5}/2,$$

$$\mathbf{M}_4 = \tfrac{1}{\sqrt{5}} \begin{bmatrix} 1 & (3\sqrt{5}-1)/4 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 \\ 1 & -(\sqrt{5}+1)/4 & (3\sqrt{5}-1)/4 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 \\ 1 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 & (3\sqrt{5}-1)/4 & -(\sqrt{5}+1)/4 \\ 1 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 & -(\sqrt{5}+1)/4 & (3\sqrt{5}-1)/4 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

On Learning Deep O(n)-Equivariant Hyperspheres

C. Complete Proofs

In this section, we provide complete proof of the propositions and theorems stated in the main paper.

Theorem. *(Restating Theorem 4:)The neuron* $\mathbf{F}_n(\,\cdot\,;\boldsymbol{S}) : \mathbb{R}^{n+2} \to \mathbb{R}^{n+1}$ *defined in* (14) *is* O(n)*-equivariant.*

Proof. We need to show that (3) holds for $\mathbf{F}_n(\,\cdot\,;\boldsymbol{S})$.

We substitute (15) to the LHS and (14) to the RHS, and obtain

$$V_n\, B_n(\boldsymbol{S})\, \boldsymbol{X} = B_n(\boldsymbol{S})\, \boldsymbol{R}\boldsymbol{X}\,. \tag{23}$$

Keeping in mind that the $(n+1)$-th and $(n+2)$-th components, s_{n+1} and s_{n+2}, of the sphere $\boldsymbol{S} \in \mathbb{R}^{n+2}$ with center $\mathbf{c}_0 \in \mathbb{R}^n$ (1) are O(n)-invariant, as well as our convention on writing the rotation matrices (see the last paragraph of Section 3.2), we rewrite the $(n+1) \times (n+2)$ matrix $B_n(\boldsymbol{S})$ using its definition (14):

$$B_n(\boldsymbol{S}) = \Big[(\boldsymbol{R}_O^\top \boldsymbol{R}_{T_i}\, \boldsymbol{R}_O\, \boldsymbol{S})^\top\Big]_{i=1}^{n+1} = \Big[\mathbf{c}_0^\top \boldsymbol{R}_O^\top \boldsymbol{R}_{T_i}^\top\, \boldsymbol{R}_O \quad s_{n+1} \quad s_{n+2}\Big]_{i=1}^{n+1}\,. \tag{24}$$

By definition of the rotation $\boldsymbol{R}_O$ (14), we have that $\boldsymbol{R}_O\, \mathbf{c}_0 = \|\mathbf{c}_0\|\mathbf{p}_1$, where $\mathbf{p}_1 \in \mathbb{R}^n$ is the first vertex of the regular simplex according to (8). Since $\boldsymbol{R}_{T_i}$ rotates $\mathbf{p}_1$ into $\mathbf{p}_i$, we obtain

$$\boldsymbol{R}_{T_i}\boldsymbol{R}_O\, \mathbf{c}_0 = \|\mathbf{c}_0\|\mathbf{p}_i\,, \quad 1 \le i \le n+1\,. \tag{25}$$

Thus, we can write the RHS of (23) using the sphere definition (1) as

$$B_n(\boldsymbol{S})\, \boldsymbol{R}\boldsymbol{X} = \Big[\|\mathbf{c}_0\|\mathbf{p}_i^\top\, \boldsymbol{R}_O \quad s_{n+1} \quad s_{n+2}\Big]_{i=1}^{n+1} \boldsymbol{R}\boldsymbol{X} = \Big[\|\mathbf{c}_0\|\, \mathbf{P}_n^\top\, \boldsymbol{R}_O\boldsymbol{R} \quad s_{n+1}\,\mathbf{1} \quad s_{n+2}\,\mathbf{1}\Big]\, \boldsymbol{X}. \tag{26}$$

We now use the definition of V_n from (15) along with (11), (12), and (25) to rewrite the LHS of (23) as

$$\begin{aligned}
V_n\, B_n(\boldsymbol{S})\boldsymbol{X} &= \mathbf{M}_n^\top \boldsymbol{R}_O\, \boldsymbol{R}\, \boldsymbol{R}_O^\top\, \mathbf{M}_n \left[\Big[\|\mathbf{c}_0\|\, \mathbf{P}_n^\top \quad s_{n+1}\,\mathbf{1}\Big]\, \boldsymbol{R}_O \quad s_{n+2}\,\mathbf{1}\right] \boldsymbol{X} \\
&= \mathbf{M}_n^\top \boldsymbol{R}_O\, \boldsymbol{R}\, \boldsymbol{R}_O^\top \left[\begin{bmatrix} p\,\|\mathbf{c}_0\| \begin{bmatrix}\mathbf{I}_n \\ \mathbf{0}^\top\end{bmatrix} & \begin{matrix}\mathbf{0} \\ p\sqrt{n}\, s_{n+1}\end{matrix}\end{bmatrix} \boldsymbol{R}_O \quad \begin{matrix}\mathbf{0} \\ p\sqrt{n}\, s_{n+2}\end{matrix}\right] \boldsymbol{X} \\
&= \mathbf{M}_n^\top \boldsymbol{R}_O\, \boldsymbol{R} \left[\begin{bmatrix} p\,\|\mathbf{c}_0\|\, \mathbf{I}_n, & \begin{matrix}\mathbf{0} \\ p\sqrt{n}\, s_{n+1}\end{matrix} \\ \mathbf{0}^\top & \end{bmatrix} \boldsymbol{R}_O^\top \boldsymbol{R}_O \quad \begin{matrix}\mathbf{0} \\ p\sqrt{n}\, s_{n+2}\end{matrix}\right] \boldsymbol{X} \\
&= \frac{1}{p}\Big[\mathbf{P}_n^\top\, \boldsymbol{R}_O\, \boldsymbol{R} \quad n^{-1/2}\mathbf{1}\Big] \begin{bmatrix} p\,\|\mathbf{c}_0\|\, \mathbf{I}_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^\top & p\sqrt{n}\, s_{n+1} & p\sqrt{n}\, s_{n+2}\end{bmatrix} \boldsymbol{X} \\
&= \Big[\,\|\mathbf{c}_0\|\, \mathbf{P}_n^\top\, \boldsymbol{R}_O\, \boldsymbol{R} \quad \tfrac{\sqrt{n}}{\sqrt{n}}\, s_{n+1}\,\mathbf{1} \quad \tfrac{\sqrt{n}}{\sqrt{n}}\, s_{n+2}\,\mathbf{1}\Big]\, \boldsymbol{X} \quad = \quad B_n(\boldsymbol{S})\, \boldsymbol{R}\boldsymbol{X}.
\end{aligned} \tag{27}$$

□

Proposition. *(Restating Proposition 7:) Given that all operations involved in the procedure 19 are* O(n)*-equivariant, its output will also be* O(n)*-equivariant.*

Proof. Let $\boldsymbol{R} \in \mathrm{O}(n)$ be an orthogonal transformation, $\rho_i(\boldsymbol{R})$ the representation of $\boldsymbol{R}$ in the respective space, e.g., (15) for the equivariant hypersphere output, and $\boldsymbol{x} \in \mathbb{R}^n$ be the input to the procedure 19. We denote the output of the procedure 19 as $\mathbf{F}(\mathbf{x})$, where $\mathbf{F}$ is the composition of all operations in the procedure 19. Since each operation is equivariant, (3) holds for each operation $\boldsymbol{\Phi}$, i.e., we have $\boldsymbol{\Phi}_i(\rho_i(\boldsymbol{R})\boldsymbol{X}) = \rho_{i+1}(\boldsymbol{R})\boldsymbol{\Phi}(\boldsymbol{X})$. Consider now the output $\mathbf{F}(\mathbf{x})$ and the transformed output $\mathbf{F}(\boldsymbol{R}\mathbf{x})$. Since each operation in $\mathbf{F}$ is equivariant, we have: $\mathbf{F}(\boldsymbol{R}\mathbf{x}) = \boldsymbol{\Phi}_d(\boldsymbol{\Phi}_{d-1}(\ldots \boldsymbol{\Phi}_2(\boldsymbol{\Phi}_1(\boldsymbol{R}\mathbf{x})))) = \rho_d(\boldsymbol{R})\boldsymbol{\Phi}_d(\boldsymbol{\Phi}_{d-1}(\ldots \boldsymbol{\Phi}_2(\boldsymbol{\Phi}_1(\mathbf{x})))) = \rho_d(\boldsymbol{R})\mathbf{F}(\mathbf{x})$. Thus, the output of the procedure in (19) is equivariant, as desired. □

13

Methods	O(3) Action recognition	O(5) Regression	O(5) Convex hulls
CGENN	9.1K	467 *	58.8K
VN	8.3K	924	N/A**
GVP	8.8K	315	N/A**
Scalars	-	641	-
SGM	8.5K	333	58.9K
MLP	8.3K	447K (Finzi et al., 2021)	N/A** | 58.2K
DEH (Ours)	**8.1K**	**275**	**49.8K**

Table 1. Total number of parameters of the models in the experiments presented in Figure 2. *see Section D.1. **an unknown exact number of parameters, somewhere in the range of the other numbers in the column, as indicated by Ruhe et al. (2023).

D. Architecture Details

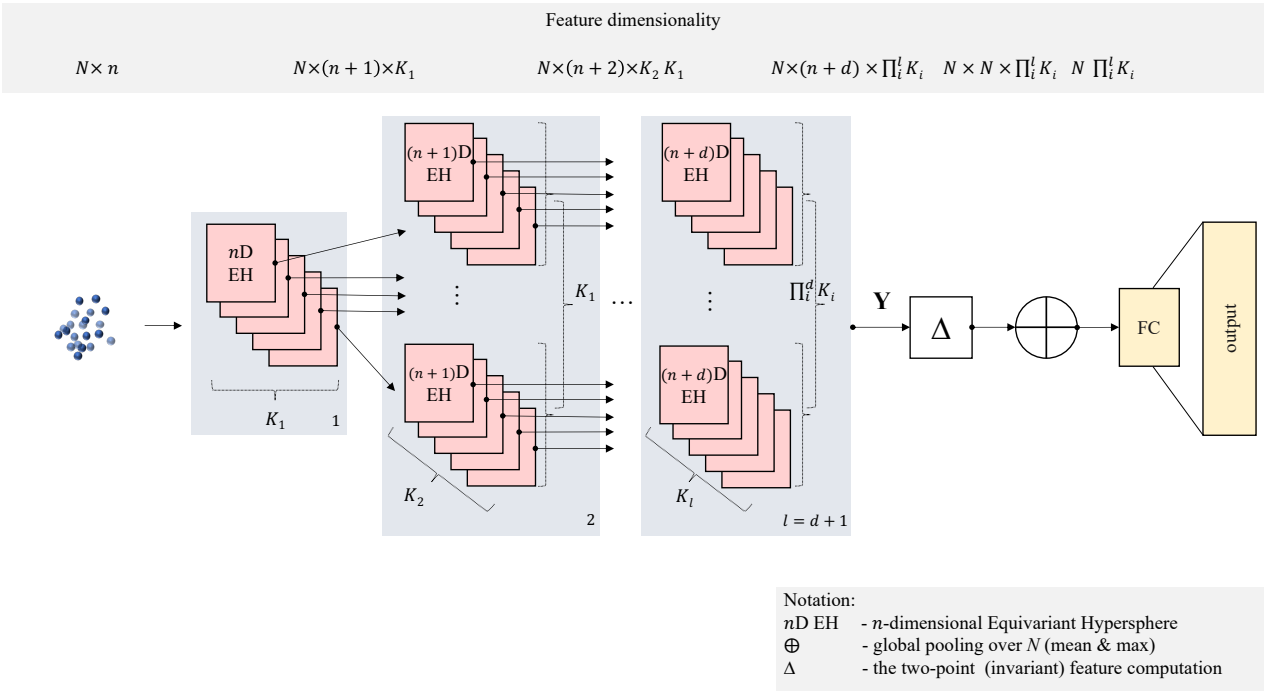


Figure 4. Architecture of our DEH model. All the operations are point-wise, i.e., shared amongst N points. Each subsequent layer of equivariant hyperspheres contains K_l neurons for each of the $\prod_i^d K_i$ preceding layer channels. The architectures of the non-permutation-invariant variants differ only in that the global aggregation function over N is substituted with the flattening of the feature map.

In this section, we provide an illustration of the architectures of our DEH model used in the experiments in Section 5. By default, we learned non-normalized hyperspheres and equipped the layers with the equivariant bias and the additional non-linearity (non-linear normalization in (18)). The number of learnable parameters corresponds to the competing methods in the experiments, as seen in Table 1. The DEH architectures are presented in Table 2.

D.1. O(5) Regression Architectures Clarification

Note that we used the CGENN model architecture, containing 467 parameters, from the first version of the manuscript (Ruhe et al., 2023), and the corrected evaluation protocol from the latest version. Their model in the latest version has

On Learning Deep O(n)-Equivariant Hyperspheres

Methods	Equiv. layer sizes $[K_1, K_2, \dots]$	Invariant operation	FC-layer size	Total #params	Performance
		O(3) Action recognition			Acc., % (↑)
DEH	[8, 6, 2]	sum	32	7.8K	69.86
DEH	[3, 2]	Δ_E	32	8.1K	82.92
DEH	[3, 2]	Δ	32	8.1K	**87.36**
		O(5) Regression			MSE (↓)
DEH	[2]	l^2-norm	32	343	0.0084
DEH	[2]	Δ_E	32	275	0.0033
DEH	[2]	Δ	32	275	**0.0007**
		O(5) Convex hulls			MSE (↓)
DEH	[32, 24]	l^2-norm	32	57.2K	7.5398
DEH	[8, 6]	Δ_E	32	49.8K	1.3843
DEH	[8, 6]	Δ	32	49.8K	**1.3166**

Table 2. Our DEH model architectures employed in the experiments and ablation on the invariant feature computation. The models are trained on all the available training data. The results of the models in **bold** are presented in Figure 2. DEH for the first and the third tasks is also permutation-invariant.

three orders of magnitude more parameters, which is in the range of the EMLPs (Finzi et al., 2021) (see Figure 2, center) containing 562K parameters. However, the error reduction thus achieved is only of one order of magnitude (Ruhe et al., 2023) and only in the maximum training data size regime, which is why we compared the models within the original size range (see Table 2 and Figure 2).

Besides, since the number of points in this task is only 2 and the permutation invariance is not required (no aggregation over N; see Figure 4 and the caption), we used only three out of four entries of Δ (21) in our model, i.e., only one of the identical off-diagonal elements. Also, we disabled the bias component in our model for this experiment and achieved a lower error (0.0007 vs. 0.0011).

E. Ablations

E.1. Invariant Feature Computation

In Table 2, we show the effectiveness of the invariant operator Δ (21), modeling the relation between two points and a sphere (see Section 4.5), over other invariant operations such as sum or l^2-norm, applied to the $N \times (n+d)$ matrix $\mathbf{Y}$ (see Section 4.4 and Figure 4) row-wise.

We also considered including the edge computation in Δ, as discussed in Section 4.5, in the following way:

$$\Delta_E = \mathbf{E} \odot \mathbf{Y}\mathbf{Y}^\top, \tag{28}$$

where $\mathbf{E} := \frac{1}{2}\left(\|\mathbf{x}_i - \mathbf{x}_j\|^2 + \mathbf{I}_N\right) \in \mathbb{R}^{N\times N}$ models the edges as the squared distances between the points (with the identity matrix included to also model interactions between a single point and a sphere). This formulation is slightly closer to the original formulation from Li et al. (2001) than (21) that we used. In Table 2, we present the respective model results.

E.2. Architecture

In Table 3, we present a comparison between a single- vs. a two-layer DEH (which was employed in the experiments with the results in Figure 1). We note that already with one layer, our model exhibits high performance for the presented tasks. Increasing the number of layers in our DEH is therefore only marginally advantageous in these cases.

Bias and learnable normalization ablations are presented in Table 4. As we see, the performance of DEH is further improved if the bias is removed, which was also noted in Section D.1. A minor improvement is obtained by removing the learnable parameters from the non-linear normalization, α (one per neuron), (18), while keeping the bias. However, removing both the bias and the learnable parameters from the normalization, results in lower performance.

15

Methods	Equiv. layer sizes $[K_1, K_2, \dots]$	Total #params	Avg.* performance
	O(3) Action recognition		Acc., % (↑)
DEH	[6]	8.1K	70.05
DEH	[3, 2]	8.1K	**70.84**
	O(5) Convex hulls		MSE (↓)
DEH	[48]	49.6K	2.1633
DEH	[8, 6]	49.8K	**2.1024**

Table 3. Our DEH model: single- and two-layer (the results of which are presented in Figure 2). *The performance is averaged across the models trained on the *various training set sizes* (see Figure 2).

Bias	Normalization with learnable parameters, α_k	Total #params	Avg.* MSE (↓)
✓	✓	49.8K	2.1024
✓	✗	49.7K	2.0936
✗	✓	49.7K	**2.0623**
✗	✗	49.7K	2.1468

Table 4. Hyperparameter ablation (using the O(5) convex hull volume prediction task from Section 5: our main DEH model with and without equivariant bias, learnable parameters in the normalization (18). *The MSE is averaged across the models trained on the *various training set sizes* (see Figure 1).