A SIMPLE AND UNIVERSAL ROTATION EQUIVARIANT POINT-CLOUD NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Equivariance to permutations and rigid motions is an important inductive bias for various 3D learning problems. Recently it has been shown that the equivariant Tensor Field Network architecture is universal- it can approximate any equivariant function. In this paper we suggest a much simpler architecture, prove that it enjoys the same universality guarantees and evaluate its performance on Modelnet40. The code to reproduce our experiments is available at `https://github.com/simpleinvariance/UniversalNetwork`

1 INTRODUCTION

Permutations and rigid motions are the basic shape-preserving transformations for point cloud data. In recent years multiple neural architectures that are equivariant to these transformations were proposed, and were demonstrated to outperform non-equivariant models on a variety of 3D learning tasks, such as shape classification Deng et al. (2021), molecule property prediction and the n-body problem Satorras et al. (2021).

A desired and much studied theoretical benchmark for equivariant neural networks is universality Zaheer et al. (2017); Maron et al. (2019b); Ravanbakhsh (2020) - the ability to approximate *any* continuous equivariant function. For networks jointly equivariant to permutations and rigid motions, universality was first achieved when Dym & Maron (2020) showed that the equivariant Tensor Field Network (TFN) architecture Thomas et al. (2018) is universal. However the disadvantages of TFN and similar architectures Klicpera et al. (2021); Fuchs et al. (2020) are (1) computational challenges: it requires maintaining intermediate high-dimensional *irreducible representations* and (2) complication: the network construction is based on the representation theory of $\mathcal{SO}(3)$, which limits the audience of this approach. Thus it is desirable to obtain simpler equivariant networks with universality guarantees. This goal was obtained for the simpler cases of 2D point clouds Bökman et al. (2021) or 3D point clouds with distinct principal eigenvalues Puny et al. (2021).

Here, we suggest a simple (though high dimensional) equivariant architecture with universality guarantees, which can be understood with a basic background in linear algebra. We present preliminary experimental results for this model. While these results are not state of the art, we believe this direction is worthy of further study, and may be a good first step towards the utlimate goal of achieving simple, equivariant networks with strong theoretical properties and empirical success.

1.1 PRELIMINARIES

Group actions and equivariance Given two (possibly different) vector spaces W_1, W_2, and a group G which acts on these vector spaces, we say that $f : W_1 \to W_2$ is equivariant if

$$f(gw) = gf(w), \forall w \in W_1, g \in G.$$

We say f is *invariant* in the special case where the action of G on W_2 is trivial, that is $gw_2 = w_2$ for all $g \in G$ and $w_2 \in W_2$. When G acts linearly on W we say W is a *representation* of G.

An important principle in the design of equivariant neural networks is that they be constructed by composition of simple equivariant functions. To achieve models with strong expressive power, the input low dimensional representations can be equivariantly mapped 'up' to high dimensional 'hidden' representations such as irreducible representations Thomas et al. (2018); Fuchs et al. (2020)

or tensor representations Kondor et al. (2018); Maron et al. (2018; 2019b;a), and them equivariantly mapped down again to the output low dimensional representation. We next introduce tensor representations which will be used in this paper.

Tensor representations Let $\mathcal{T}_k$ denote the vector spaces $\mathcal{T}_0 = \mathbb{R}$, $\mathcal{T}_1 = \mathbb{R}^3$, $\mathcal{T}_2 = \mathbb{R}^{3\times 3}$, $\mathcal{T}_3 = \mathbb{R}^{3\times 3\times 3}, \ldots$. An orthgonal matrix $R \in \mathcal{O}(3)$ acts on $\mathcal{T}_k$ via

$$\left(R^{\otimes k} V\right)_{i_1,\ldots,i_k} = \sum_{j_1,\ldots,j_k=1}^{3} R_{i_1,j_1} R_{i_2,j_2} \ldots R_{i_k,j_k} V_{j_1,j_2,\ldots,j_k}.$$

We note that $R^{\otimes k} : \mathcal{T}_k \to \mathcal{T}_k$ can be identified with a mapping $R^{\otimes k} : \mathbb{R}^{3^k} \to \mathbb{R}^{3^k}$, and as our notation suggests this mapping is the Kronecker product of R with itself k times. For $k = 0, 1, 2$ applying $R^{\otimes k}$ to the scalar/vector/matrix $V \in \mathcal{T}_k$ gives $R^{\otimes 0} V = V, \quad R^{\otimes 1} V = RV$ and $R^{\otimes 2} V = RVR^T$.

Equivariant mappings we review two basic basic mappings between tensor representations of $\mathcal{O}(3)$, which will later be used to define our equivariant layers.

Tensor product mappings are a standard method to equivariantly map lower order representations to higher order representations. The tensor product $\otimes : \mathcal{T}_k \times \mathcal{T}_\ell \to \mathcal{T}_{k+\ell}$ is defined for $V^{(1)} \in \mathcal{T}_k$ and $V^{(2)} \in \mathcal{T}_\ell$ by

$$\left(V^{(1)} \otimes V^{(2)}\right)_{i_1,\ldots,i_k,j_1,\ldots,j_\ell} = V^{(1)}_{i_1,\ldots,i_k} \cdot V^{(2)}_{j_1,\ldots,j_\ell}.$$

Contractions are a convenient method for equivariantly mapping high order representations to lower order representations: for $k \geq 2$, any pair of indices a, b with $1 \leq a < b \leq k$ defines a contraction mapping $C_{a,b} : \mathcal{T}_k \to \mathcal{T}_{k-2}$, which is defined by jointly marginalizing over the a, b indices. For example for $a = 2, b = k$ we have $\left(C_{2,k}(V)\right)_{i_1,i_2,\ldots,i_{k-2}} = \sum_{j=1}^{3} V_{i_1,j,i_2\ldots,i_{k-2},j}$. The equivariance of tensor products and contractions will be proved in Proposition 1 in the appendix.

2 METHOD

2.1 SETUP AND OVERVIEW

Our goal is to construct an architecture which produces equivariant functions $f : W_0 \to W_1$, where $W_0 = \mathbb{R}^{3\times n}$ is the space of point clouds, which is acted on by the group of orthogonal transformations and permutations $\mathcal{O}(3) \times S_n$. We note that translation equivariance/invariance can be easily added to our model by centralizing the input point cloud to have zero mean (see Dym & Maron (2020) for more details). The output representations W_1 we consider vary according to the task at hand: for example, classification tasks are invariant to rigid motions and permutations, predicting the trajectory of a dynamical system is typically equivariant to permutations and rigid motion, while segmentation tasks are permutation equivariant but invariant to rigid motions.

As in previous works our architecture is a concatenation of several equivariant layers that will be discussed in detail next. Specifically, it is composed of three main types of layers: *Ascending Layers* that produce higher order representations, *Descending Layers* which do the opposite and Linear layers that allow us to mix different channels. We use a U-net based Ronneberger et al. (2015) architecture that first uses ascending layers up to some predefined maximal order K, and then descends to the required output order (see Figure 1).

2.2 EQUIVARIANT LAYERS

In general we consider mappings between representations of $\mathcal{O}(3) \times S_n$ of the form $\mathcal{T}_k^{n\times C}$ where the group action is given by

$$\left[(R,\sigma)(V)\right]_{j,c} = R^{\otimes k}(V_{\sigma^{-1}(j),c}) \tag{1}$$

where we denote elements in $\mathcal{T}_k^{n\times C}$ by $V = (V_{jc})_{1\leq j\leq n, 1\leq c\leq C}$ and $V_{jc} \in \mathcal{T}_k$ for every fixed j, c. Note that for $k = 1, C = 1$ we get our input representation $\mathcal{T}_1^{n\times 1} = \mathbb{R}^{3\times n}$. Our construction is based on three basic layers:

Ascending Layers: Ascending layers are parametric mappings $\mathcal{A} : \mathcal{T}_k^{n\times C} \times \mathbb{R}^{3\times n} \to \mathcal{T}_{k+1}^{n\times C}$ which depend on parameters $\boldsymbol{\alpha} = (\alpha_{1c}, \alpha_{2c})_{c=1,\ldots,C}$. They use tensor products to obtain higher order represenations and are of the form[1] $V^{out} = \mathcal{A}(V^{in}, X|\boldsymbol{\alpha})$, where (using X_j to denote the j-th column of X)

$$V_{jc}^{out} = \alpha_{1c}\left(X_j \otimes V_{jc}\right) + \alpha_{2c} \sum_{i\neq j} X_i \otimes V_{ic} \tag{2}$$

Descending layers: Descending layers are parameteric mappings $\mathcal{D} : \mathcal{T}_k^{n\times C} \to \mathcal{T}_{k-2}^{n\times C}$ (defined for $k \geq 2$) which are of the form $V^{out} = \mathcal{D}(V^{in}|\boldsymbol{\beta})$, where $\boldsymbol{\beta} = (\beta_{a,b,c})_{1\leq a<b\leq k, 1\leq c\leq C}$. and are defined by:

$$V_{j,c}^{out} = \sum_{1\leq a<b\leq k} \beta_{a,b,c} C_{a,b}(V_{j,c}^{in})$$

Linear layers: We use the linear layers from Thomas et al. (2018). These layers are parametric mappings $\mathcal{L} : \mathcal{T}_k^{n\times C} \to \mathcal{T}_k^{n\times C'}$ of the form $V^{out} = \mathcal{L}(V^{in}|\boldsymbol{\gamma})$, where $\boldsymbol{\gamma} = (\gamma_{cc'})_{1\leq c\leq C, 1\leq c'\leq C'}$, which are defined by:

$$V_{jc'}^{out} = \sum_{c=1}^{C} \gamma_{cc'} V_{jc}^{in}.$$

The equivariance of these layers to orthogonal transformations and permutations follows rather easily from our previous discussion. We prove this formally in Proposition 2 in the appendix.

2.3 Architecture

The architecture we use depends on two hyper-parameters: the number of channels C (which we keep fixed throughout the network), and the maximal representation order used K. This choice of hyper-parameters defines a parametric function space $\mathcal{F}(K, C)$, containing functions which gradually map pointclouds up to K order representations, and then gradually map back down, using the ascending, descending and linear layers discussed above. The architecture is visualized in Figure 1 (with $C = 1$ and using the identification $\mathcal{T}_k = \mathbb{R}^{3^k}$). We next formally describe our architecture.

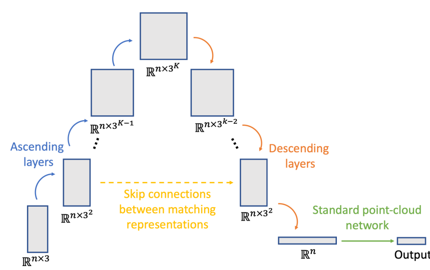

Figure 1: Architecture

We denote the input point cloud by $X \in \mathbb{R}^{3\times n}$, and define an initial degenerate representation $U^{(0)} \in \mathcal{T}_0^{n\times C}$ which is identically one in all $n \cdot C$ coordinates. we recursively define for $k = 1, \ldots, K$

$$U^{(k)} = \mathcal{L}\left(\mathcal{A}(U^{(k-1)}, X)\right).$$

where we suppress the dependence of $\mathcal{A}$ and $\mathcal{L}$ on the learned parameters $\boldsymbol{\alpha}^{(k)}$ and $\boldsymbol{\gamma}^{(k)}$ for notation simplicity. Each $U^{(k)} = U^{(k)}(X)$ contains $n \times C$ copies of a 3^k dimensional tensor in $\mathcal{T}_k$. Next we denote $U^{(K)} = V^{(K)}$ and recursively define for $k = K, K-2, \ldots, r+2$ where $r = K(\mod 2)$

$$V^{(k-2)} = \mathcal{L}\left(concat(\mathcal{D}(V^{(k)}), U^{(k-2)})\right) \tag{3}$$

where again we suppress the dependence of $\mathcal{D}$ and $\mathcal{L}$ on the learned parameters $\boldsymbol{\beta}^{(k-2)}$ and $\bar{\boldsymbol{\gamma}}^{(k-2)}$ for simplicity. Overall we get a function $f : \mathcal{T}_1^{n\times C} \to \mathcal{T}_r^{n\times C}$ of the form

$$V^{(r)} = f(X|\boldsymbol{\alpha}^{(1)}, \ldots, \boldsymbol{\alpha}^{(K)}, \boldsymbol{\beta}^{(K-2)}, \ldots, \boldsymbol{\beta}^{(r)}, \boldsymbol{\gamma}^{(1)}, \ldots, \boldsymbol{\gamma}^{(K)}, \bar{\boldsymbol{\gamma}}^{(K-2)}, \ldots, \bar{\boldsymbol{\gamma}}^{(r)}).$$

The function f is permutation and ortho-equivariant when $r = 1$ or ortho-*invariant* and permutation equivariant when $r = 0$. When $r = 0$ we can apply a permutation invariant/equivariant network such as PointNet (Qi et al., 2017) or DGCNN (Wang et al., 2019) to the output of our network to strengthen the expressive power of our network while maintaining the ortho-invariance and permutation invariance/equivariance of our overall construction.

[1]Note that this layer was already suggested in Dym & Maron (2020), and its structure resembles the structure of the basic layers in Zaheer et al. (2017); Maron et al. (2020); Thomas et al. (2018)

Expansion of basic model The basic model we presented up to now is sufficient to prove universality as discussed in Theorem 1 below. However, it is *only* able to compute polynomials up to degree K. To enable expression of non-polynomial functions we use the ReLU activation layer defined in Deng et al. (2021) (which easily generalizes to higher representations). To take local information into account, we expand our ascending layer in equation 2 by adding a summation over the K-nearest neighbors in feature space, giving

$$V_{jc}^{out} = \alpha_{1c} \left(X_j \otimes V_{jc}\right) + \alpha_{2c} \sum_{i \neq j} X_i \otimes V_{ic} + \alpha_{3c} \sum_{i \sim j} X_i \otimes V_{ic}.$$

Finally, we have experimented with adding linear layers which map $\mathcal{T}_k$ equivariantly to itself. When $k = 2$ these linear layers are spanned by the linear mappings

$$V \mapsto V,\, V \mapsto V^T,\, V \mapsto trace(V) I_3, \quad V \in \mathcal{T}_2 = \mathbb{R}^{3\times 3}.$$

We find that adding these mappings for $k = 2$ improves our results, and generalizing this to higher dimensions is an interesting challenge for further research.

3 THEORETICAL PROPERTIES

We now discuss the expressive power of the architecture $\mathcal{F}(K, C)$ defined above. Based on the proof methodology of Dym & Maron (2020), we prove the following theorem (stated formally in the appendix)

Theorem 1. *[non-formal statement] For every even K and large enough $C \geq C(K)$, every polynomial of degree $\leq K$ which is permutation equivariant (or invariant) and invariant to rigid motions can be expressed by functions in $\mathcal{F}(K, C)$ composed with simple pooling and centralizing operations.*

Since the invariant/equivariant polynomials are dense in the space of continuous invariant functions (uniformly on compact sets, see e.g., Lemma 1 in Dym & Maron (2020)) this theorem means that in the limit where $K, C \to \infty$ our architecture is able to approximate any function invariant to orthogonal transformations, translations and permutations.

A disadvantage of universality theorems is that it is unclear how big a network is needed to get a reasonable approximation of a given function. In Theorem 2, stated and proved in the appendix, we consider the function λ_{Cov} which computes the three eigenvalues of the covariance matrix of the point cloud X, ordered according to size. This is a classical global descriptor (see e.g.,Puny et al. (2021); Kazhdan et al. (2004)) which is invariant to permutations and rigid motions. We show that it can be computed by our networks with representations of order $K = 6$, composed with a continuous (non-invariant) function q which can be approximated by an MLP.

4 EXPERIMENTS

We present initial results of our model on the ModelNet40 classification task with the protocol used in Deng et al. (2021). We examine the contribution of high representations by comparing different representation orders $K = 2, 4, 6$. As expected we find that higher representation lead to better accuracy. We find that our model does not perform as well as recent state of the art architectures with joint permutation and rigid motion invariance (see Table 1). We are currently working on additional expansions to our basic model, such as the study of equivariant linear mappings on $\mathcal{T}_k$ mentioned above, and believe this may lead to improved results on Modelnet40 and other equivariant learning tasks. For additional details and results on the experiment setup see Appendix A.

Methods	Accuracy
SFCNN	91.4
TFN	88.5
RI-Conv	86.5
SPHNet	87.7
ClusterNet	98.1
GC-Conv	89.0
RI-Framework	89.4
VN-PointNet	77.5
VN-DGCNN	89.5
Our method (K=2)	78.3
Our method (K=4)	80.4
Our method (K=6)	81.6

Table 1: ModelNet40.

REFERENCES

Georg Bökman, Fredrik Kahl, and Axel Flinth. Zz-net: A universal rotation equivariant architecture for 2d point clouds. *arXiv preprint arXiv:2111.15341*, 2021.

Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4989–4997, 2019. doi: 10.1109/CVPR.2019.00513.

Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12200–12209, 2021.

Nadav Dym and Haggai Maron. On the universality of rotation equivariant point cloud networks. *arXiv preprint arXiv:2010.02449*, 2020.

Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

Juan A Navarro González and Juan B Sancho de Salas. *C-differentiable spaces*, volume 1824. Springer, 2003.

Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Shape matching and anisotropy. In *ACM SIGGRAPH 2004 Papers*, pp. 623–629. 2004.

Johannes Klicpera, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. *Advances in Neural Information Processing Systems*, 34, 2021.

Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*, 2018.

Xianzhi Li, Ruihui Li, Guangyong Chen, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. A rotation-invariant framework for deep point cloud analysis. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2021. ISSN 2160-9306. doi: 10.1109/tvcg.2021.3092570. URL `http://dx.doi.org/10.1109/TVCG.2021.3092570`.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019a.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pp. 4363–4371. PMLR, 2019b.

Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On learning sets of symmetric elements. In *International Conference on Machine Learning*, pp. 6734–6744. PMLR, 2020.

Adrien Poulenard, Marie-Julie Rakotosaona, Yann Ponty, and Maks Ovsjanikov. Effective rotation-invariant point cnn with spherical harmonics kernels, 2019.

Omri Puny, Matan Atzmon, Heli Ben-Hamu, Edward J Smith, Ishan Misra, Aditya Grover, and Yaron Lipman. Frame averaging for invariant and equivariant network design. *arXiv preprint arXiv:2110.03336*, 2021.

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

Yongming Rao, Jiwen Lu, and Jie Zhou. Spherical fractal convolutional neural networks for point cloud recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 452–460, 2019.

Siamak Ravanbakhsh. Universal equivariant multilayer perceptrons. In *International Conference on Machine Learning*, pp. 7996–8006. PMLR, 2020.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International Conference on Machine Learning*, pp. 9323–9332. PMLR, 2021.

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

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

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

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Zhiyuan Zhang, Binh-Son Hua, David W. Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning, 2019.

A Implementation details

We evaluated our proposed network on a standard point cloud classification benchmark, ModelNet40 Wu et al. (2015), which consists of 40 classes with 12,311 pre-aligned CAD models, split into 80% for training and 20% for testing. Classification problems are invariant to rigid motions and permutations. We preprocess the point cloud to have zero mean to achieve translation invariance, and then apply the model described in the paper with K even to achieve function invariant to rigid motions and equivariant to permutations. Finally we apply sum pooling and a fully connected neural network to achieve a fully invariant model.

We compared our method to SFCNN Rao et al. (2019), TFN Thomas et al. (2018), RI-Conv Zhang et al. (2019), SPHNet Poulenard et al. (2019), ClusterNet , GC-Conv Chen et al. (2019), RI-Framework Li et al. (2021) and to two variations of Vector Neurons Deng et al. (2021). All these methods are invariant to rigid motions and permutations.

In the comparison showed in Table 2 we used the standard z/z evaluation protocal- that is, learning and test data are augmented with rotations around the z-axis. We verified that due to its invariance to rigid motions and permutations, our method, like the other methods we compare to, are hardly effected when the test data is augmented by general 3D rotations (z/SO(3)) or when both test and train are augmented by 3D rotations (SO(3)/SO(3)).

Implementation and other technical details We trained and tested our model on NVidia GTX A6000 GPUs with python 3.9, Cuda 11.3, PyTorch 1.10.0, PyTorch geometric, and pytorch3d 0.6.0. We trained our model for 100 epochs with a batch size of 32, a learning rate of 0.1, and a seed of 0.

Ablation on our method Without the use of the ReLU activation layer defined in Deng et al. (2021), the accuracy of our method decreases by 29.0%. When we remove the K-nearest neighbors summation as well, the accuracy of our method decreases by an additional 15.6%.

Methods	z/z	z/SO(3)	SO(3)/SO(3)
SFCNN	91.4	84.8	90.1
TFN	88.5	85.3	87.6
RI-Conv	86.5	86.4	86.4
SPHNet	87.7	86.6	87.6
ClusterNet	87.1	87.1	87.1
GC-Conv	89.0	89.1	89.2
RI-Framework	89.4	89.4	89.3
VN-PointNet	77.5	77.5	77.2
VN-DGCNN	89.5	89.5	90.2
Our method (K=2)	78.3	78.4	77.7
Our method (K=4)	80.4	80.4	78.8
Our method (K=6)	81.6	81.5	80.1
Ours (K=4) w.o. VNReLU (K=4)	51.4	51.4	55.2
Ours (K=4) w.o. VNReLU & KNN (K=4)	35.8	35.8	33.8

Table 2: Test clasification accuracy on the ModelNet40 dataset in three train/test scenarios. z stands for the aligned data augmented by random rotations around the vertical axis and SO(3) indicates data augmented by random rotations

B Equivariance proofs

We begin by proving

Proposition 1. *Tensor products and contractions are $\mathcal{O}(3)$ equivariant.*

As a preliminary to this proof, recall that $T \in \mathcal{T}_k$ is a rank one tensor if there exists $t^{(1)}, \ldots, t^{(k)} \in \mathbb{R}^3$ such that

$$T_k = t^{(1)} \otimes t^{(2)} \otimes \ldots \otimes t^{(k)}. \tag{4}$$

Proof of Proposition 1. equivariance of tensor products

We need to show that for $T \in \mathcal{T}_k$ and $S \in \mathcal{T}_\ell$ we have

$$R^{\otimes k} T \otimes R^{\otimes \ell} S = R^{\otimes (k+\ell)} \left(T \otimes S\right), \quad \forall R \in \mathcal{O}(3). \tag{5}$$

since both sides of the equation are bilinear in (T, S), and since every tensor in $\mathcal{T}_k$ can be written as a linear combination of rank one tensors, it is sufficient to prove equation 5 for the special case where T and S are rank one tensors. If T is a rank one tensor as in equation 4, then $R^{\otimes k} T$ is given by

$$\begin{aligned}
\left[(Rt^{(1)}) \otimes (Rt^{(2)}) \otimes \ldots \otimes (Rt^{(k)})\right]_{i_1,\ldots,i_k} &= (Rt^{(1)})_{i_1} \times (Rt^{(2)})_{i_2} \times \ldots \times (Rt^{(k)})_{i_k} \\
&= \left(\sum_{j_1=1}^{3} R_{i_1 j_1} t^{(1)}_{j_1}\right)\left(\sum_{j_2=1}^{3} R_{i_2 j_2} t^{(2)}_{j_2}\right)\ldots\left(\sum_{j_k=1}^{3} R_{i_k j_k} t^{(k)}_{j_k}\right) \\
&= \sum_{j_1,\ldots,j_k=1}^{3} R_{i_1,j_1} R_{i_2,j_2} \ldots R_{i_k,j_k} T_{j_1,j_2,\ldots,j_k} \\
&= \left(R^{\otimes k} T\right)_{i_1,\ldots,i_k}
\end{aligned}$$

it follows that for every rank one tensors $T = t^{(1)} \otimes t^{(2)} \otimes \ldots \otimes t^{(k)}$ and $S = s^{(1)} \otimes s^{(2)} \otimes \ldots \otimes s^{(\ell)}$ we have for every $R \in \mathcal{O}(3)$

$$\begin{aligned}
R^{\otimes k} T \otimes R^{\otimes \ell} S &= (Rt^{(1)}) \otimes (Rt^{(2)}) \otimes \ldots \otimes (Rt^{(k)}) \otimes (Rs^{(1)}) \otimes (Rs^{(2)}) \otimes \ldots \otimes (Rs^{(\ell)}) \\
&= R^{\otimes (k+\ell)} \left[t^{(1)} \otimes t^{(2)} \otimes \ldots \otimes t^{(k)} \otimes s^{(1)} \otimes s^{(2)} \otimes \ldots \otimes s^{(\ell)}\right] \\
&= R^{\otimes (k+\ell)} (T \otimes S)
\end{aligned}$$

and thus we have shown correctness of equation 5 for all rank-one tensors and thus for all tensors. We note that this proof does not really require R to be an orthogonal matrix and would work for any square matrix. This is not the case for contractions, as we will see next:

equivariance of contractions For simplicity of notation we prove the equivariance of contractions $C_{a,b} : \mathcal{T}_k \to \mathcal{T}_{k-2}$ in the special case $a = k-1, b = k$. We need to show that for all $T \in \mathcal{T}_k$ and $R \in \mathcal{O}(3)$ we have

$$C_{k-1,k}(R^{\otimes k} T) = R^{\otimes (k-2)} C_{k-1,k}(T) \tag{6}$$

since both sides of the equation above are linear in T and every tensor of order k can be written as a linear combination of rank one tensors, it is sufficient to show that equation 6 holds for all rank one tensors. Let T be a rank one tensor as in equation 4. Note that by definition

$$\begin{aligned}
\left[C_{k-1,k}(T)\right]_{i_1,i_2,\ldots,i_{k-2}} &= \sum_{j=1}^{3} T_{i_1,\ldots,i_{k-2},j,j} \\
&= \langle t^{(k-1)}, t^{(k)} \rangle \left[t^{(1)} \otimes t^{(2)} \otimes \ldots \otimes t^{(k-2)}\right]
\end{aligned} \tag{7}$$

and so for every $R \in \mathcal{O}(3)$, since we saw in the first part of the proof that $R^{\otimes k} T$ is a rank one tensor given by

$$R^{\otimes k} T = (Rt^{(1)}) \otimes (Rt^{(2)}) \otimes \ldots \otimes (Rt^{(k)})$$

and so we get that

$$\begin{aligned}
C_{k-1,k}(R^{\otimes k} T) &= \langle Rt^{(k-1)}, Rt^{(k)} \rangle (Rt^{(1)}) \otimes (Rt^{(2)}) \otimes \ldots (Rt^{(k)}) \\
&= \langle t^{(k-1)}, t^{(k)} \rangle (Rt^{(1)}) \otimes (Rt^{(2)}) \otimes \ldots (Rt^{(k)}) \\
&= \langle t^{(k-1)}, t^{(k)} \rangle \left[R^{\otimes (k-2)} \left(t^{(1)} \otimes t^{(2)} \otimes \ldots \otimes t^{(k-2)}\right)\right] \\
&= R^{\otimes (k-2)} C_{k-1,k}(T).
\end{aligned}$$

Thus we have shown that equation 6 holds for all rank one tensors and thus for all tensors. This concludes the prove of Proposition 1. $\square$

Layer equivariance We now prove equivariance of our layers. In the following discussion we use the notation $\rho_k(R,\sigma)$ for the action of (R,σ) on $\mathcal{T}_k^{n\times C}$ as defined in equation 1.

Proposition 2. *For any choice of parameters, the layers $\mathcal{A}, \mathcal{D}$ and $\mathcal{L}$ are equivariant.*

Proof of Proposition 2. **Equivariance of ascending layers** We need to show that for every given input $X \in \mathbb{R}^{3\times n}, V \in \mathcal{T}_k^{n\times C}$, for every $R \in \mathcal{O}(3), \sigma \in S_n$ and any fixed parameter vector $\boldsymbol{\alpha}$

$$\rho_{k+1}(R,\sigma)V^{out} = \mathcal{A}(\rho_k(R,\sigma)V^{in}, \rho_1(R,\sigma)X|\boldsymbol{\alpha})$$

Indeed using the definition of the action ρ_{k+1} from equation 1 and the equivariance of the tensor prouct we proved above, we have

$$\begin{aligned}
\left[\rho_{k+1}(R,\sigma)V^{out}\right]_{jc} &= R^{\otimes(k+1)}[V^{out}_{\sigma^{-1}(j),c}] \\
&= \alpha_{1c}R^{\otimes(k+1)}\left(X_{\sigma^{-1}(j)} \otimes V_{\sigma^{-1}(j)c}\right) + \alpha_{2c}\sum_{i\neq\sigma^{-1}(j)} R^{\otimes(k+1)}\left(X_i \otimes V_{ic}\right) \\
&= \alpha_{1c}(RX_{\sigma^{-1}(j)}) \otimes (R^{\otimes k}V_{\sigma^{-1}(j)c}) + \alpha_{2c}\sum_{i\neq\sigma^{-1}(j)} (RX_i) \otimes (R^{\otimes k}V_{ic}) \\
&= [\mathcal{A}(\rho_k(R,\sigma)V^{in}, \rho_1(R,\sigma)X|\boldsymbol{\alpha})]_{j,c}
\end{aligned}$$

Equivariance of descending layers We need to show that for all $R \in \mathcal{O}(3), \sigma \in S_n$, for all $V^{in} \in \mathcal{T}_k^{n\times C}, V^{out} \in \mathcal{T}_{k-2}^{n\times C}$ and for all choice of a parameter vector $\boldsymbol{\beta}$,

$$\rho_{k-2}(R,\sigma)V^{out} = \mathcal{D}(\rho_k(R,\sigma)V^{in}|\boldsymbol{\beta}).$$

Using the definition of the action ρ_k from equation 1 and the equivariance of contraction we proved above, we have

$$\begin{aligned}
[\rho_{k-2}(V^{out})]_{j,c} = R^{\otimes(k-2)}[V^{out}_{\sigma^{-1}(j),c}] &= \sum_{1\leq a<b\leq k} \beta_{a,b,c}R^{\otimes(k-2)}C_{a,b}(V^{in}_{\sigma^{-1}(j),c}) \\
&= \sum_{1\leq a<b\leq k} \beta_{a,b,c}C_{a,b}(R^{\otimes k}V^{in}_{\sigma^{-1}(j),c}) = [\mathcal{D}(\rho_k(R,\sigma)V^{in}|\boldsymbol{\beta})]_{j,c}
\end{aligned}$$

Equivariance of linear layers We need to show that for all $R \in \mathcal{O}(3), \sigma \in S_n$, for all $V^{in}, V^{out} \in \mathcal{T}_k^{n\times C}$ and for all choice of a parameter vector $\boldsymbol{\gamma}$,

$$\rho_k(R,\sigma)V^{out} = \mathcal{L}(\rho_k(R,\sigma)V^{in}|\boldsymbol{\gamma}).$$

Indeed

$$\begin{aligned}
[\rho_k(R,\sigma)V^{out}]_{jc'} = R^{\otimes k}[V^{out}_{\sigma^{-1}(j),c'}] &= \sum_{c=1}^{C} \gamma_{cc'}R^{\otimes k}V^{in}_{\sigma^{-1}(j)c} \\
&= [\mathcal{L}(\rho_k(R,\sigma)V^{in}|\boldsymbol{\gamma})]_{jc'}.
\end{aligned}$$

This concludes the prove of Proposition 2. $\square$

C EXPRESSIVE POWER PROOFS

We now reformulate and prove Theorem 1.

Theorem (Reformulation of Theorem 1)**.** *For every even number K and every large enough $C = C(K)$, every polynomial $p : \mathbb{R}^{3\times n} \to \mathbb{R}^n$ of degree $\leq K$ which is permutation equivariant and invariant to rigid motions can be obtained as the first channel of a function $f \in \mathcal{F}(K,C)$, that is*

$$p_j(X) = f_{j1}(X), \forall j = 1, \ldots, n$$

This immediately implies an analogous result where we replace the permutation equivariant assumption by a permutation invariant assumption:

Corollary 1. *For every even number K and every large enough $C = C(K)$, every polynomial $p : \mathbb{R}^{3\times n} \to \mathbb{R}$ of degree $\leq K$ which is* invariant *to permutations and rigid motions can be obtained by applying sum pooling to the first channel of a function $f \in \mathcal{F}(K, C)$, that is*

$$p(X) = \sum_{j=1}^{n} f_{j1}(X - \frac{1}{n}X1_n1_n^T)$$

proof of Theorem 1. Our proof is based on the general framework for proving universality laid out in Dym & Maron (2020). In this paper it is shown that polynomials $p : \mathbb{R}^{3\times n} \to \mathbb{R}^n$ of degree K which are permutation equivariant and invariant to translations and orthogonal transformations[2]can be written for large enough C as

$$p(X) = \sum_{c=1}^{C} \hat{\Lambda}_c\left(g_c(X)\right) \tag{8}$$

where

1. g_c is a member of a function space $\mathcal{F}_{\text{feat}}$ which maps $\mathbb{R}^{3\times n}$ to W_{feat}^n, where W_{feat} is a representation of $\mathcal{O}(3)$.
2. Λ_c is a member of a space of functions $\mathcal{F}_{\text{pool}}$ from W_{feat} to $\mathbb{R}$ and $\hat{\Lambda}_c : W_{\text{feat}}^n \to \mathbb{R}^n$ is the function induced by elementwise application of Λ_c.
3. The function spaces $\mathcal{F}_{\text{pool}}$ has the linear universality property. This means that $\mathcal{F}_{\text{pool}}$ is precisely the set of linear functionals $\Lambda : W_{\text{feat}} \to \mathbb{R}$ which are $\mathcal{O}(3)$ equivariant.
4. the function space $\mathcal{F}_{\text{feat}}$ has the K-spanning property. This means that all functions in $\mathcal{F}_{\text{feat}}$ are required to by $\mathcal{O}(3) \times S_n$ equivaraint and invariant to translations, and additionally, that any permutation equivariant and translation invariant (ortho-equivariance not required) degree D polynomial $\tilde{p} : \mathbb{R}^{3\times n} \to \mathbb{R}^n$ can be obtained by an expression as in equation 8 where Λ_c are linear but are not required to be ortho-invariant.

It is also shown in Lemma 3 in Dym & Maron (2020) that the spaces of functions obtained by applying ascending layers recursively $k = 1, \ldots, K$ times form a K-spanning family. Here we use ascending layers independently on each channel and essentially remove the linear layers by setting $\gamma_{cc'}^{(k)} = \delta_{cc'}$ for all $k = 1, \ldots, K$. Applying ascending layers k times gives us a function from $\mathbb{R}^{3\times n}$ to $\mathcal{T}_k$, and since we are basically considering a collection of different functions to different representations $\mathcal{T}_1, \ldots, \mathcal{T}_K$ (depending on the number of ascending layers used) we choose to embed all these representations into a joint representation $W_{\text{feat}} = \oplus_{k=1}^{K}\mathcal{T}_k$. Thus we see that every permutation equivaraint and translation and ortho-invariant polynomial p can be written in the form equation 8 where the g_c are obtained by applying our ascending layers k_c times for some $1 \leq k_c \leq K$ and $\Lambda_c : \oplus_{k=1}^{K}\mathcal{T}_k \to \mathbb{R}$ is linear and ortho-invariant. Since g_c maps into a single representation $\mathcal{T}_{k_c}$ in practice we can think of Λ_c is a linear equivariant functional on this single representation.

Using an analogous argument to the proof of Proposition 1 in Appendix 5 in Dym & Maron (2020), it can be shown that the space of linear invariant functionals $\Lambda : \mathcal{T}_k \to \mathbb{R}$ are spanned by the functionals $\{\Lambda_\sigma | \, \sigma \in S_k\}$ which are defined uniquely be the requirement that for every $x^{(1}, \ldots, x^{(k)} \in \mathbb{R}^3$,

$$\Lambda_\sigma(x^{\sigma(1)} \otimes \ldots \otimes x^{\sigma(k)}) = \langle x^{\sigma(1)}, x^{\sigma(2)}\rangle \times \langle x^{\sigma(3)}, x^{\sigma(4)}\rangle \ldots \langle x^{\sigma(k-1)}, x^{\sigma(k)}\rangle.$$

These Λ_σ are given by (see also the derivation of equation 7)

$$\Lambda_\sigma(T) = C_{\sigma(1),\sigma(2)} \circ C_{\sigma(3),\sigma(4)} \circ \ldots \circ C_{\sigma(k-1),\sigma(k)}(T).$$

In particular we see that if k is odd there is no non-zero linear equivariant functional from $\mathcal{T}_k$ to $\mathbb{R}$, so we can assume that k_c is even for all $c = 1, \ldots, C$, and Λ_c can be obtained by applying the last

[2]Actually the argument in this paper discusses rotation invariance rather than orthogonal transformations (that is, we also discuss invariance to reflections). However the arguments there hold in this case as well.

$k_c/2$ descending layers of our construction to the output $U^{(k_c)}$ of g_c with an appropriate choice of $0-1$ parameters. Recall that $U^{(k_c)} = g_c(X)$ was obtained from k_c ascending layers. When $k_c < K$ we achieve this using our K-dimensional U-shaped architecture by 'short-circuiting' at the k_c level, that is by setting the parameters of the linear layer $\mathcal{L}$ in equation 3 to erase $\mathcal{D}(U^{(k_c+2)})$ and maintain only $U^{(k_c)}$.

We have seen that we can write p as in equation 8 where $\hat{\Lambda}_c$ and g_c can be constructed to be the c-th channel of our network. We obtain the sum of $\hat{\Lambda}_c \circ g_c$ in the first channel of the output representation $V^{(0)}$ by setting the first column of the parameter matrix $\boldsymbol{\gamma}^{(0)}$ of the last linear layer
to be $\gamma_{c1}^{(0)} = 1, \forall c = 1, \ldots, C$.

□

Computing eigenvalues of the covariance matrix We prove
Theorem 2. *Let $\lambda_{Cov} : \mathbb{R}^{3\times n} \to \mathbb{R}^3$ be the function which, given a point cloud $X \in \mathbb{R}^{3\times n}$, computes the ordered eigenvalues of the covariance matrix $(X - \frac{1}{n}X1_n1_n^T)(X - \frac{1}{n}X1_n1_n^T)^T$. For large enough C there exists $f \in \mathcal{F}(6, C)$ and a continuous $q : \mathbb{R}^3 \to \mathbb{R}^3$ such that*

$$\lambda_{Cov}(X) = q\left(\sum_{j=1}^{n} f_{j1}(X - \frac{1}{n}X1_n1_n^T), \sum_{j=1}^{n} f_{j2}(X - \frac{1}{n}X1_n1_n^T), \sum_{j=1}^{n} f_{j3}(X - \frac{1}{n}X1_n1_n^T)\right)$$

proof of Theorem 2. The covariance matrix of $X \in \mathbb{R}^{3\times n}$ is given by

$$\bar{X}\bar{X}^T, \text{ where } \bar{X} = X - \frac{1}{n}1_n1_n^T.$$

The covariance matrix is a symmetric positive semi-definite matrix and we denote its eigenvalues by $\lambda_1(X) \geq \lambda_2(X) \geq \lambda_3(X) \geq 0$ and define

$$\lambda_{Cov}(X) = (\lambda_1(X), \lambda_2(X), \lambda_3(X))$$

We now define polynomials p_2, p_4, p_6 of degree $2, 4, 6$ which are invariant to permutations and rigid motions, by

$$\begin{aligned} p_2(X) &= \bar{X}\bar{X}^T = \lambda_1(X) + \lambda_2(X) + \lambda_3(X) \\ p_4(X) &= \left(\bar{X}\bar{X}^T\right)^2 = \lambda_1^2(X) + \lambda_2^2(X) + \lambda_3^2(X) \\ p_6(X) &= \left(\bar{X}\bar{X}^T\right)^3 = \lambda_1^3(X) + \lambda_2^3(X) + \lambda_3^3(X) \end{aligned}$$

Since p_2, p_4, p_6 are invariant and of degree ≤ 6 we can approximated them with our architecture $\mathcal{F}(6, C)$ with C large enough.

It remains to show that there exists a continuous mapping $q : \mathbb{R}^3 \to \mathbb{R}^3$ such that (dropping the dependence of the eigenvalues on X)

$$q\left(\lambda_1 + \lambda_2 + \lambda_3, \lambda_1^2 + \lambda_2^2 + \lambda_3^2, \lambda_1^3 + \lambda_2^3 + \lambda_3^3\right) = (\lambda_1, \lambda_2, \lambda_3)$$

This follows from the fact that the three polynomials $s_1, s_2, s_3 : \mathbb{R}^3 \to \mathbb{R}$

$$\begin{aligned} s_1(\lambda_1, \lambda_2, \lambda_3) &= \lambda_1 + \lambda_2 + \lambda_3 \\ s_2(\lambda_1, \lambda_2, \lambda_3) &= \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \\ s_3(\lambda_1, \lambda_2, \lambda_3) &= \lambda_1^3 + \lambda_2^3 + \lambda_3^3 \end{aligned}$$

are permutation invariant polynomials known as the power sum polynomials, which generate the ring of permutation invariant polynomials on $\mathbb{R}^3$, and as such, the map $\boldsymbol{s} = (s_1, s_2, s_3)$ induces a homeomorphism of $\mathbb{R}^3/S_3$ onto the image of $\boldsymbol{s}$ (González & de Salas (2003), Lemma 11.13) . Similarly, the sorting function $\mathrm{sort} : \mathbb{R}^3 \to \mathbb{R}^3$ is an injective permutation invariant mapping which induces a homeomorphism of $\mathbb{R}^3/S_3$ onto its image. Thus the sets

$$\{\boldsymbol{s}(\lambda_1, \lambda_2, \lambda_3) | \, (\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}^3\} \text{ and } \{\mathrm{sort}(\lambda_1, \lambda_2, \lambda_3) | \, (\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}^3\}$$

are homeomorphic, and we can choose q to be a homeomorphism (in particular, q is continuous). □