A SIMPLE AND UNIVERSAL ROTATION EQUIVARIANT POINT-CLOUD NETWORK

Ben Finkelshtein & Chaim Baskin
Faculty of Computer Science
Technion
Haifa, Israel

Haggai Maron
NVIDIA Research
Israel

Nadav Dym
Faculty of Mathematics
Technion
Haifa, Israel
nadavdym@technion.ac.il

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 - the ability to approximate *any* continuous equivariant function. The universality of the translation equivariant convolutional neural networks was discussed in (Yarotsky, 2022). In contrast, the expressive power of standard permutation-equivariant graph neural networks such as message passing networks (Gilmer et al., 2017) is limited (Xu et al., 2018; Morris et al., 2019; 2021). Universality for graph neural networks can be attained using a 'lifted approach', where the input $n \times n$ graph is mapped to high dimensional hidden representations of the permutation group of dimension $n^K, K = 0, 1, 2, \ldots$. Universality can be achieved by taking a very large $K = n$ (Maron et al., 2019b; Ravanbakhsh, 2020).

Our focus in this paper is on equivariant networks for $3 \times n$ point clouds. In this context, universality of point cloud networks which are equivariant with respect to permutations (and not rigid motions) was discussed in (Zaheer et al., 2017; Qi et al., 2017), and the opposite case of universality with respect to rigid motions (and not permutations) was discussed in (Villar et al., 2021; Bogatskiy et al., 2020). In most of these constructions, The size of the hidden representations in these networks is proportional to the input dimension $3 \cdot n$ (though the number of channels may be large).

Many point cloud networks were suggested which are *jointly* equivariant to permutations and rigid motions (Deng et al., 2021; Poulenard et al., 2019; Satorras et al., 2021), and have hidden representations of size $\sim 3 \cdot n$. While it is currently unknown whether these networks are universal, there is reason to think they may not be, as separation of $d \times n$ point clouds up to equivalence is just as difficult as separation of graphs (Dym & Lipman, 2017), and this problem (known as the graph isomorphism problem) has no known polynomial time algorithm (Babai, 2016).

For $3 \times n$ point clouds, an interesting intermediate option between the standard low dimensional approach and the untractable lifted approach is the 'semi-lifted' approach. In this approach, intermediate hidden representations of dimension $3^K \cdot n, K = 0, 1, 2 \ldots$ are used, which are essentially high-dimensional representations of the rigid motion group combined with the standard representations of the permutation group. In contrast with the fully lifted approach which would require hidden representations of dimension $(3n)^K$, the complexity of the semi-lifted approach is linear in n (typically $n \gg 3$) and thus is tractable for moderately sized K. Several papers have proposed architectures based on the semi-lifted approach, including (Thomas et al., 2018; Klicpera et al., 2021; Fuchs et al., 2020).

Unlike the more standard low-dimensional approach, it was shown in (Dym & Maron, 2020) that semi-lifted jointly equivariant architectures can be universal. In particular, universality can be obtained using the Tensor Field Network (TFN) (Thomas et al., 2018; Klicpera et al., 2021; Fuchs et al., 2020) jointly equivariant layers. To the date, the construction in (Dym & Maron, 2020) seems to be the only universality result for jointly equivariant networks at this level of generality. Universality was obtained for the simpler cases of 2D point clouds (Bökman et al., 2021) (where the group of rotations is commutative) or 3D point clouds with distinct principal eigenvalues (Puny et al., 2021).

While universality can be obtained using TFN layers, the specific architecture used to obtain universality was not yet implemented. More importantly, TFN layers are based on the representation theory of $SO(3)$, which limits the audience of this approach and leads to cumbersome implementation.

The goal of this short paper is to present for the first time an implementation of a universal architecture using the theoretical framework laid out in (Dym & Maron, 2020). This architecture is based on tensor representations (to be defined), and as a result can be easily understood with a basic background in linear algebra. The architecture depend on hyper-parameters K and C which define an architecture $\mathcal{F}(K, C)$ with C channels and representations of dimension at most $n \cdot 3^K$. The union over all possible (K, C) is dense in the space of jointly equivariant functions. Additionally, for fixed K and large enough C, this architecture can express all jointly equivariant polynomials of degree K. In particular, we show that the eigenvalues of the point cloud's covariance matrix, which are a common invariant shape descriptor, can be expressed by architectures with $K = 6$.

Experimentally, with the hardware currently available to us we have been able to train models for the ModelNet40 classification class with $K \leq 6$. These results are presented in Table 1. While our results are currently not state of the art, we believe this direction is worthy of further study, and may be a good first step towards the ultimate 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, and equivariant mappings between 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 orthogonal 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

$$R^{\otimes 0} V = V, \quad R^{\otimes 1} V = RV \text{ and } \quad R^{\otimes 2} V = RVR^T.$$

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

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

A proof of the equivariance of tensor product is given in Proposition A.1 in the appendix[1]. For now we give a simple example: when $k = \ell = 1$ the equivariance of the tensor product mapping can be seen by noting that for vectors $v, w \in \mathcal{T}_1$ and $R \in \mathcal{O}(3)$ we obtain

$$(Rv) \otimes (Rw) = (Rv)(Rw)^T = Rvw^T R^T = R^{\otimes 2}(v \otimes w).$$

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

$$(C_{2,k}(V))_{i_1,i_2,\ldots,i_{k-2}} = \sum_{j=1}^{3} V_{i_1,j,i_2\ldots,i_{k-2},j}.$$

The equivariance of contractions will be proved in Proposition A.1 in the appendix. Fir now we present a simple example is the case $k = 2, a = 1, b = 2$ where we get for every matrix $V \in \mathcal{T}_2$ that

$$C_{1,2}(V) = \sum_{j=1}^{3} V_{jj} = \mathrm{trace}(V)$$

and $\mathrm{trace} : \mathcal{T}_2 \to \mathcal{T}_0$ is indeed a $\mathcal{O}(3)$ equivariant mapping.

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 map lower order tensor representations up to higher order tensor 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

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

[1]The appendix can be found in the arxiv version of this article (Finkelshtein et al., 2022).

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 representations and are of the form[2] $V^{out} = \mathcal{A}(V^{in}, X|\boldsymbol{\alpha})$, 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 parametric 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 $\boldsymbol{\beta} = (\beta_{a,b,c})_{1\leq a<b\leq k, 1\leq c\leq C}$. They 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 $\boldsymbol{\gamma} = (\gamma_{cc'})_{1\leq c\leq C, 1\leq c'\leq C'}$. They 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 A.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 point clouds 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.

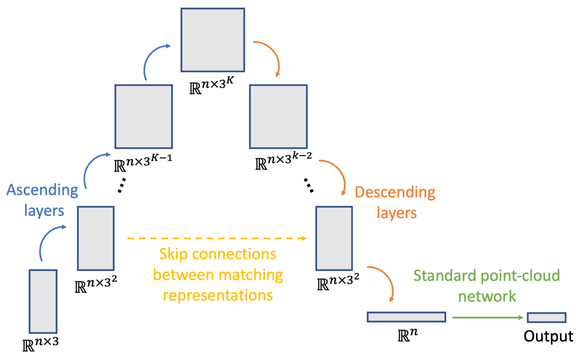
$\mathbb{R}^{n\times 3^K}$
$\mathbb{R}^{n\times 3^{K-1}}$
$\mathbb{R}^{n\times 3^{k-2}}$
Ascending layers
Descending layers
$\mathbb{R}^{n\times 3^2}$
Skip connections between matching representations
$\mathbb{R}^{n\times 3^2}$
Standard point-cloud network
$\mathbb{R}^{n\times 3}$
$\mathbb{R}^{n}$
Output

Figure 1: Our architecture $\mathcal{F}(K, C)$ with $C = 1$.

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 $r = K (\mod 2)$

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

[2]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)

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.

Expansion of basic model The basic model we presented up to now is sufficient to prove universality as discussed in Theorem 3.1 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 3.1. *[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 B.3, 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

In this section, we evaluate our model. In Section 4.1, we describe in details the experimental setup we have used and in Section 4.2 we present our main results.

Methods	z/z	z/SO(3)	SO(3)/SO(3)
SFCNN Rao et al. (2019)	91.4	84.8	90.1
TFN Thomas et al. (2018)	88.5	85.3	87.6
RI-Conv Zhang et al. (2019)	86.5	86.4	86.4
SPHNet Poulenard et al. (2019)	87.7	86.6	87.6
ClusterNet Chen et al. (2019)	87.1	87.1	87.1
GC-Conv Zhang et al. (2020)	89.0	89.1	89.2
RI-Framework Li et al. (2021)	89.4	89.4	89.3
VN-PointNet Deng et al. (2021)	77.5	77.5	77.2
VN-DGCNN Deng et al. (2021)	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 1: 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. All models presented in the table are rotation and permutation invariant (up to numerical inaccuracies).

4.1 Experimental Setting

Data 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.

Implementation 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.

4.2 Experimental Results

We present initial results of our model on the ModelNet40 classification task, under the three standard evaluation protocols for jointly equivariant architectures: in the first protocol (z/z) train and test data are augmented with rotations around the z-axis, in the second (z/SO(3)) train data is augmented by rotations around the z-axis and test data is augmented by general 3D rotations, and in the third protocol (SO(3)/SO(3)) train and test data are augmented by 3D rotations.

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 (Chen et al., 2019), GC-Conv (Chen et al., 2019), RI-Framework (Li et al., 2021) and to two variations of Vector Neurons (Deng et al., 2021). The results of the comparison are shown in Table 1. In general we find that our model does not perform as well as many of the models we compared too. We do note that our results are obtained with a relatively small amount of parameters: the variation of our method with the highest empirical success and largest amount of parameters (K=6) uses 370,000 parameters, while VN-point net uses five times more parameters and VC-DGCNN uses more than seven times more.

All methods we compared to, as well as our own method, are invariant to rigid motions and permutations. Thus they 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)) (for comparison, see the results for the three augmentation protocols obtained for methods without rotation invariance, as shown e.g., in (Deng et al., 2021)).

Ablation on our method 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. 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%.

5 Conclusion and future work

In this short paper we first presented an implementation of a simple, universal, neural network architecture which is jointly equivariant of permutations and rigid motions. Experimentally, we find that our model does not perform as well as recent state of the art architectures with joint permutation and rigid motion invariance. 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.

At the same time, we believe the inability of our method, as well as other semi-lifted approaches such as TFN, to outperform methods with low-dimensional representations is worthy of further study and raises many interesting questions for further research. For example: Are prevalent low dimensional models universal? If they aren't, what equivariant tasks are they likely to fail in? Is there a theoretical explanation to their success in other tasks? Is it possible that semi-lifted models are more expressive but are difficult to optimize? If so can this difficulty be alleviated? We hope to address these questions in future work, and hope others will be inspired to do the same.

References

László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 684–697, 2016.

Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *International Conference on Machine Learning*, pp. 992–1002. PMLR, 2020.

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 Yaron Lipman. Exact recovery with symmetries for procrustes matching. *SIAM Journal on Optimization*, 27(3):1513–1530, 2017.

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

Ben Finkelshtein, Chaim Baskin, Haggai Maron, and Nadav Dym. A simple and universal rotation equivariant point-cloud network. *arXiv preprint arXiv:2203.01216*, 2022.

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.

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

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.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019.

Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: The story so far. *arXiv preprint arXiv:2112.09992*, 2021.

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.

Vıćctor 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.

Soledad Villar, David W Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. *Advances in Neural Information Processing Systems*, 34:28848–28863, 2021.

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.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55(1):407–474, 2022.

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.

Zhiyuan Zhang, Binh-Son Hua, Wei Chen, Yibin Tian, and Sai-Kit Yeung. Global context aware convolutions for 3d point cloud understanding, 2020.