A functional approach to rotation equivariant non-linearities for Tensor Field Networks.

Adrien Poulenard
Stanford University
padrien@stanford.edu

Leonidas J. Guibas
Stanford University
guibas@cs.stanford.edu

Abstract

Learning pose invariant representation is a fundamental problem in shape analysis. Most existing deep learning algorithms for 3D shape analysis are not robust to rotations and are often trained on synthetic datasets consisting of pre-aligned shapes, yielding poor generalization to unseen poses. This observation motivates a growing interest in rotation invariant and equivariant methods. The field of rotation equivariant deep learning is developing in recent years thanks to a well established theory of Lie group representations and convolutions. A fundamental problem in equivariant deep learning is to design activation functions which are both informative and preserve equivariance. The recently introduced Tensor Field Network (TFN) framework provides a rotation equivariant network design for point cloud analysis. TFN features undergo a rotation in feature space given a rotation of the input pointcloud. TFN and similar designs consider nonlinearities which operate only over rotation invariant features such as the norm of equivariant features to preserve equivariance, making them unable to capture the directional information. In a recent work entitled "Gauge Equivariant Mesh CNNs: Anisotropic Convolutions on Geometric Graphs" Hann et al. interpret 2D rotation equivariant features as Fourier coefficients of functions on the circle. In this work we transpose the idea of Hann et al. to 3D by interpreting TFN features as spherical harmonics coefficients of functions on the sphere. We introduce a new equivariant nonlinearity and pooling for TFN. We show improvments over the original TFN design and other equivariant nonlinearities in classification and segmentation tasks. Furthermore our method is competitive with state of the art rotation invariant methods in some instances.

1. Introduction

In recent years many successful deep learning architectures for 3D geometric deep learning and point cloud analysis in particular have been developed, we refer to [14] for a comprehensive survey. Yet, most methods developed for point cloud analysis are not robust to rotations. Furthermore point cloud dataset like ModelNet [33] or ShapeNet [6] used for training and evaluating these methods often consist of synthetic pre-aligned shapes. In consequence, well established point cloud methods like [25, 27, 4, 31, 21] fail to generalise to unseen poses. An important performance gap between the aligned and unaligned or rotation augmented settings has been reported in in multiple publications [11, 24, 7, 37]. Recently numerous works [12, 22, 24, 7, 37, 36, 20, 39, 28] have addressed the issue of designing rotation invariant deep learning architectures for 3D data analysis with a variety of approaches. A particularly interesting property for 3D deep learning algorithms closely related to rotation invariance is rotation equivariance. At a high level a rotation equivariant neural network produces features that undergo a rotation in feature space given a rotation of the input. The key point is that this transform only depends on the rotation applied to the input. This is a precious property for learning, guaranteeing immediate generalization to unseen poses. Many rotation equivariant designs [30, 17, 11, 32, 2] have been proposed, thanks to a well established theory of $\mathrm{SO}(3)$ representations. A fundamental challenge in the design of equivariant networks is the design of equivariant non-linearities. To preserve equivariance such non-linearities must commute with the rotations of equivariant features, this constraint limits the possible designs. The aforementioned methods either rely on non-linearities applied to rotation invariant quantities like the norm of equivariant features which cannot capture directional information or, consider polynomial equivariant features which can be expensive to compute and complicates training. In the recent work [9] Hann *et al.* introduce a non linearity for local intrinsic rotation equivariant features over surfaces by interpreting the features as Fourier coefficients of functions over the circle, composing these functions with non linear activations and getting new equivariant features by computing the Fourier coefficients of the composed functions. In this work we explore a similar approach in the context of 3D rotation equivariant features from Tensor Field Networks [30] over point-clouds. We propose a simple yet effective design of rotation equivariant non-linearities for Tensor Field Networks. TFN relies on convolutional filters based on spherical harmonics and the resulting features share the same equivariant properties as the spherical harmonics basis. We interpret such feature vectors as the set of coefficients of a function on the unit sphere. Moving to a sphere function representation by applying inverse spherical harmonics transform we can apply non trivial activation functions or MLP's in a pointwise fashion over the sphere. We then return back to the equivariant feature representation by computing the spherical harmonics transform of the resulting function. This operation has a non trivial effect on both the norm and direction of equivariant features, allowing to capture directional information. We present the general setting of point cloud convolutions and TFN and briefly review the underlying theory in section (2). In section (3) we describe our method and specificities of our design compared to the general approach described in section (2). We present our results in section (4) where we compare our method to state of the art rotation invariant methods for shape classification and segmentation and show improvements over existing equivariant non-linearities. We believe our contibution is of general relevance to other equivaraint networks like [32] sharing similar structures to TFN.

2. Background and Related work

2.1. Point cloud & graph convolution

In recent years the field of point cloud processing has developed a lot thanks to many works introducing various notions of graph or point-cloud convolution [27, 21, 4, 31]. Graph convolution aggregates information over the neighbourhood of each vertex. Point cloud and graph convolutions are related as we can define a notion of point cloud convolution by considering the graph of k nearest neighbours of the point cloud. More generally, it is convenient to aggregate information in the neighborhoods of points from a second point cloud, typically a sub-sampled or an upsampled point cloud, similarly to the idea of strided convolutions. The general formulation for graph convolutions is as follows: Let $\mathcal{V}$ be the set of vertices and $\mathcal{X}, \mathcal{Y} \subset \mathcal{V}$ non empty subsets of $\mathcal{V}$. Given signals $x : \mathcal{X} \rightarrow \mathbb{R}^k$, $y : \mathcal{Y} \rightarrow \mathbb{R}^k$ and $f : \mathcal{X} \rightarrow \mathbb{R}$ and a relation kernel function $\kappa : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, the convolution of f by κ is the signal $g : \mathcal{Y} \rightarrow \mathbb{R}$ defined by:

$$g_i = c_i \bigcirc_{j \sim i} \kappa(x_j, y_i) f_j \tag{1}$$

where $\bigcirc$ is a permutation invariant operator like $\sum$ or $\max$, j ranges across the neighbours in $\mathcal{X}$ of vertex i in $\mathcal{Y}$ and c_i is a normalization factor. We follow an approach similar to [4, 30, 24] for point cloud convolution where, $x : \mathcal{X} \rightarrow \mathbb{R}^3$, $y : \mathcal{Y} \rightarrow \mathbb{R}^3$ are the points euclidean coordinates of point clouds $\mathcal{X}$ and $\mathcal{Y}$ and, $\kappa : \mathbb{R}^3 \rightarrow \mathbb{R}$ is a kernel function:

$$g_i = f *_{x,y} \kappa = c_i \sum_{j \sim i} \kappa(x_j - y_i) f_j \tag{2}$$

where j ranges across the k-nearest neighbours of i in $\mathcal{Y}$ or points in $\mathcal{Y}$ within a ball of a given radius centred at i. A typical choice for c_i is to divide by the number of neighbours of i, $c_i = \frac{1}{|\mathcal{N}(i)|}$ but in practice we will choose a different normalization as we shall see later. In deep learning applications a collection of kernel functions κ_m is chosen and linear combinations of the $f *_{x,y} \kappa_m$ are learned.

Feature propagation We also borrow the feature propagation layers from [27] for segmentation tasks. Feature propagation allows to transfer signals between two point clouds. Given two point clouds x and y and a signal f over x, the feature propagation of f to y is defined as follows:

$$g_i = \left(\sum_{j \sim i} \frac{1}{\|x_j - y_i\|_2^2} \right)^{-1} \sum_{j \sim i} \frac{f_j}{\|x_j - y_i\|_2^2} \tag{3}$$

where j ranges across the 3 nearest neighbors of x_i in y. Feature propagation is used in the segmentation architecture of [27] which is a variant of U-Net architecture. The network computes series of conv like layers down sampling the signal, then feature propagation layers up sample the signal to concatenate it with the output of the conv like layer of corresponding resolution and, applies a fully connected layer to the channels axis followed by a batch normalization layer and a ReLu activation. An MLP is applied to the last layer for final point classification.

2.2. 3D rotation equivariant CNNs

In this subsection we describe the construction of 3D rotation equivariant CNNs and we refer to [18, 8, 16] for a more extensive view of the underlying theory. We especially consider Tensor Field Networks [30] which operate on point clouds but, our description also partly covers other works like [32, 17, 2] which also rely on equivariant convolution based on steerable kernels. 3D Rotation equivariant CNN's are built around the notion of 3D steerable kernels. A steerable kernel basis is a basis of the form:

$$\kappa_{rm}^{\ell}(x) = \varphi_r(\|x\|_2) Y_m^{\ell}\left(\frac{x}{\|x\|_2}\right) \tag{4}$$

where $Y^{\ell} : x \mapsto (Y_{-\ell}^{\ell}, \dots, Y_{\ell}^{\ell}) \in \mathbb{R}^{2\ell+1}$ is the vector of degree $\ell \in \mathbb{N}$ spherical harmonics and $(\varphi_r : \mathbb{R} \rightarrow \mathbb{R})_r$ are the radial component of the kernels. A key property of Y^{ℓ} is that it undergoes a rotation in $\mathbb{R}^{2\ell+1}$ given a rotation of its input. More precisely for any rotation $R \in \mathrm{SO}(3)$, there exist a rotation matrix $D^{\ell}(R) \in \mathrm{SO}(2\ell+1)$ associated to R called the Wigner matrix (of type ℓ) [8, 16, 18] such that, $Y^{\ell}(Rx) = D^{\ell}(R)Y^{\ell}(x)$. By construction the kernel basis from Eq. (4) satisfies the same equivariance property as spherical harmonics:

$$\kappa_r^{\ell}(Rx) = D^{\ell}(R)\kappa_r^{\ell}(x). \tag{5}$$

Many choices are possible for the radial component, we detail our exact kernel design in section (3). Given a point cloud $x \in \mathbb{R}^{n\times 3}$ and a signal $f \in \mathbb{R}^{n\times k}$ over x, we say that $v(x,f) \in \mathbb{R}^{2\ell+1}$ is a type ℓ equivariant feature if:

$$v(Rx, f) = D^{\ell}(R)v(x,f) \tag{6}$$

Type ℓ vector features can be global or point-wise. Type 0 features are rotation invariant scalars. Any rotation invariant descriptor or pose independent signal can be thought of as a type 0 feature. Type 1 features undergo rotations in $\mathbb{R}^3$ (up to permutation of axes), the yzx coordinates of the points are type 1 features. The convolution of a type 0 features with a type ℓ kernel produces type ℓ features:

$$\begin{aligned}\left(\kappa_r^{ell} *_{R.x,R.y} f\right)_i &= c_i \sum_{j\sim i} \kappa_r^{\ell}(R(x_j - y_i))f_j \\ &= D^{\ell}(R)\left(c_i \sum_{j\sim i} \kappa_r^{\ell}(x_j - y_i)f_j\right)\end{aligned} \tag{7}$$

Stacking convolutions with equivariant kernels involves computing convolutions between equivariant kernels and equivariant features of various types. The resulting features are linear combinations of tensor products of equivariant features. The Peter-Weyl theorem [16] implies that such tensor products can be decomposed into orthogonal sums of irreducible features of different types. More specifically, for any k,ℓ and any $J \in [\![m-\ell, m+\ell]\!]$ there exists an orthogonal projection matrix $Q^{(k,l),J}$ such that $Q^{(k,l),J}u\otimes v$ is a type J feature vector. The convolution of a signal f^{ℓ} of type ℓ vectors over x by a type k equivariant kernel κ_{rk} is the collection of tensors:

$$(\kappa_r^k *_{x,y} f^{\ell})_{ip}^J := Q^{(k,\ell),J}\left(c_i \sum_{j\sim i} \kappa_r^k(x_j - y_i) \otimes f_{jp}^{\ell}\right) \tag{8}$$

For $J \in [\![|l-k|, l+k]\!]$, the coefficients of the matrices $Q^{(k,\ell),J}$ are given by the so called Clebsch-Gordan coefficients [8]. TFN consist of a multi-head architecture where each head is a tensor stacking pointwise equivariant features of a given type. An equivariant convolution layer takes a collection $(f^{\ell})_{\ell}$ of such tensors as input, computes their convolution with kernels of different types, decomposes the results into irreducible features and, groups them by type and learns linear combinations between the resulting features:

$$\sum_p W_{qp}^J(\kappa^k *_{x,y} f^{\ell})_{ip}^J + \delta_{J0}b_q. \tag{9}$$

In practice we only take decomposed features up to type 3. A bias term b can be added for type 0 outputs since the bias is pose independent it is only compatible with type 0 (invariant) features. Notice that only linear combinations between vector features are taken. The reason is that the weights matrices must commute with the Wigner matrix. By a corollary of Schur's lemma [16], a linear map commutes with the Wigner matrix iff it is an homothety. Thus to preserve equivariance we can only take linear combinations of equivariant features of the same type and not between their coordinates. This is a limitation compared to traditional non-equivariant filters where we can consider any linear combinations of the basis filters.

Equivariant Activations A major limitation in the design of rotation equivariant neural networks is the choice of activation functions. To preserve equivariance, the activation function must commute with the Wigner matrix. Equivariant non-linearities follow two trends. Works like [30, 32] multiply equivariant features by non-linear functions of invariant features. The resulting features still commute with the Wigner matrix as only their norm is affected and thus they are equivariant. However such non-linearities don't have effect on the direction of features which limits their expressivity. An other direction explored by [17] and [2] is to build equivariant features as polynomials of other equivariant features. While it has been proven in [10] that equivariant polynomial features can approximate equivariant functions, in practice they can be computationally heavy and are difficult to train as reported in section (8.1) of [2]. We now briefly review non-linearities from [30, 32, 17] and [2]. TFN [30] introduce the notion of norm non-linearities, for a multi head equivariant signal f it is defined by:

$$g_{i0c}^0 := \xi(f_{i0c}^0 + b_c^0),\ \forall \ell > 0,\ g_{i:c}^{\ell} := \xi(\|f_{i:c}^{\ell}\|_2 + b_c^{\ell})\frac{f_{i:c}^{\ell}}{\|f_{i:c}^{\ell}\|_2} \tag{10}$$

Where the ξ is the ReLu function and the b^{ℓ}'s are learnable bias terms. 3D steerable cnns [32] introduce a similar but slightly more elaborate approach called gated non-linearities. Instead of only producing a multi head equivariant signal f like in TFN, a layer also produces a rotation invariant signal γ^{ℓ} for each equivariant head f^{ℓ} with $\ell > 0$ (one can simply split the invariant head of the signal in two parts) the gated non linearity is defined by:

$$g_{i0c}^0 := \xi(f_{i0c}^0 + b_c^0),\ \forall \ell > 0,\ g_{imc}^{\ell} := \sigma(\gamma_{i0c}^{\ell} + b_c^{\ell})f_{imc}^{\ell} \tag{11}$$

where σ is the sigmoid function and the b^{ℓ} are learnable bias vectors. On the other hand [17] and [2] consider tensor product non-linearities. The idea is to take tensor products of equivariant features and decompose them using the Clebsch Gordan Decomposition producing quadratic equivariant features:

$$g^J_{i:,c} := \bigoplus_{J \in [\![|\ell-k|, \ell+k]\!]} Q^{(k,\ell),J} f^k_{i:,c} \otimes f^\ell_{i:,c} \tag{12}$$

where the $\oplus$ sign indicates concatenation on the last axis. The supplementary material of [32] also provides a description of the aforementioned non-linearities.

3. Our method

In this section we present our method and the specific changes we introduce to the pipeline presented in section (2). Our main contribution is to interpret rotation equivariant features as coefficients of functions in the spherical harmonics basis with two applications: Equivariant non-linearities and equivariant spatial max pooling. Before detailing these points we specify our kernel design and point cloud sampling strategy.

Our kernels and sampling strategy As explained in equation (2) of section (2) point cloud convolution requires two point clouds a first point cloud caries the signal and the second one (typically a sub-sampling or up-sampling of the first one) defines the locations at which to apply the convolution kernels. We adopt the same sampling strategy as [24], we consider point clouds whose cardinal is a power of two and index them by a balanced kd-tree, we can then sub-sample the point cloud by applying 1D average pooling with a pool size which also a power of two. This computes the means of sub-trees at a given depth. For convolutions we use a similar kernel to [30, 32, 24] but our normalization differs slightly as we empirically observed a slight improvement with the L2 normalization (14). Like [32, 24] we chose Gaussian shell functions for the radial component functions φ of equation (4) but, we divide by the sum of the radial functions so that each point in the kernel support is assigned a mass of 1 and, we restrict the kernel support to a ball of radius r. Denoting $r_j := \frac{jr}{d}$ we define the radial component of the steerable kernel of Eq. (4) by:

$$\varphi_{r_j m}(\|x\|_2) := \frac{\exp\left(-\ln(2)d^2(\|x\|_2 - r_j)^2\right)}{\sum_{k=0}^{d-1}\exp\left(-\ln(2)d^2(\|x\|_2 - r_k)^2\right)} \tag{13}$$

We normalize the kernels at each point by their average L2 norm, leading to the following formula for the normalization factor $c_{ir\ell}$ in formula (4):

$$c_{ir\ell} = \left(\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j\sim i}\|\kappa^\ell_r(x^k_j - x^{k+1}_i)\|_2^2\right)^{\frac{1}{2}}\right)^{-1}. \tag{14}$$

Equivariant features as functions The convolution layers presented in equations (4, 8, 9) of section (2) produce equivariant features tensors $(f^\ell)_\ell$ where the type ℓ ranges between 0 and $\ell_{\max} \geqslant 0$, we set the same number of channels for each head. We observe that type ℓ rotation equivariant features as defined in equation (6) of section (2) and spherical harmonics of degree ℓ share the same equivariance properties, that is they are multiplied by the Wigner matrix $D^\ell(R)$ when their input is rotated by $R \in \mathrm{SO}(3)$. This motivates us to interpret them as coefficients of functions in the spherical harmonics basis. For each point i and channel index c we can compute the inverse Spherical Harmonics Transform (SHT) resulting in a function on the unit sphere:

$$\mathcal{F}^+(f)_{ic}(x) := \sum_{\ell=0}^{\ell_{\max}}\langle f^\ell_{i:c}, Y^\ell(x)\rangle = \sum_{\ell=0}^{\ell_{\max}}\sum_{m=-\ell}^{\ell} f^\ell_{imc} Y^\ell_m(x) \tag{15}$$

Equivariant non-linearities The inverse SHT functional representation allows us to apply non-linearities in a pointwise fashion over the sphere by composing $\mathcal{F}^+(f)$ with any activation function ξ and, compute the coefficients of the composition in the spherical harmonics basis using forward SHT:

$$\xi(f)^\ell_{imc} := \mathcal{F}(\xi\circ\mathcal{F}^+(f)_{ic})^\ell := \int_{\mathcal{S}_2}\xi(\mathcal{F}^+(f)_{ic}(x))Y^\ell_m(x)\mathrm{d}x \tag{16}$$

Our contribution relies on the following observation:

Theorem 3.1 *Our continuous non linearity $\xi(\bullet)$ is $\mathrm{SO}(3)$ equivariant. That is, for any rotation $R \in \mathrm{SO}(3)$ we have $\xi(R.f)^\ell_{i:c} = D^\ell(R)\xi(f)^\ell_{i:c}$.*

We refer to the supplementary material for a proof. Unfortunately there is no closed form for the integral of equation (16) so, in practice we have to rely on a discrete approximation of the SHT and its inverse. For that we consider a discrete sampling of the unit sphere given by a finite set of points $p_1, \ldots, p_N \in \mathcal{S}_2$, we define the discretized inverse SHT by:

$$\mathcal{F}^+(f)_{ijc} := \mathcal{F}^+(f)_{ic}(p_j) \tag{17}$$

We then apply the non linearity ξ at all points $(p_j)_j$ and compute the coefficients of this discrete approximation of $\mathcal{F}^+(f)_{ic}$ into the spherical harmonic basis using the discretized SHT (18). Given a discrete signal $\{F_{ijc}\}$ over the points $P = (p_j)_j$ for all point i of the input point cloud and all channel index c we define our SHT layer by:

$$\mathcal{F}(F)^\ell_{imc} := \frac{4\pi}{N}\sum_{j=1}^{N} F_{ijc}Y^\ell_m(p_j) \tag{18}$$

We use the the intermediate representation $\mathcal{F}^+(f)$ to apply pointwise operations at each sample p_j like a dense layer over the channel axis and non-linearities or more generaly

MLP's while maintaining equivariance up to our discretization. We apply our activation function ξ to f by:

$$\xi[f]^{\ell}_{imc} := \mathcal{F}(\xi \circ \mathcal{F}^{+}(f))_{imc} \tag{19}$$

We consider different samplings of the sphere, a first choice motivated by theorem (3.2) is to use the vertices of polyhedrons with non trivial symmetry groups. Specifically we chose the Regular dodecahedron with its 20 vertices and the Pentakis decahedron with its 32 vertices.

Theorem 3.2 *Our non linearity $\xi[\bullet]$ is equivariant w.r.t. the symmetry group of the sampling. That is, for any rotation $R \in \mathrm{SO}(3)$ in the symmetry group of the sampling P we have $\xi[R.f]^{\ell}_{i:c} = D^{\ell}(R)\xi[f]^{\ell}_{i:c}$.*

We refer to the supplementary material for a proof. We also consider the Fibonaci Sampling from [23] providing a more systematic and flexible sampling allowing to choose the number of points.

Equivariant pooling An other benefit of our approach is that we can use our functional representation to perform max pooling. Given a sub-sampling y of the point cloud x we define local max pooling by:

$$\mathrm{MP}(f)^{l}_{ic} = \mathcal{F}(\max_{i \sim j} \mathcal{F}^{+}(f)_{i,\dots})^{\ell}_{ic} \tag{20}$$

where j ranges across the k-nearest neighbours of y_i in x. We also define global max-pooling by:

$$\mathrm{GMP}(f)^{\ell}_{c} := \mathcal{F}(\max_{i} \mathcal{F}^{+}(f)_{i,\dots})^{\ell}_{c} \tag{21}$$

3.1. Architecture and models

In our experiments we consider a classification and a segmentation architecture. The basic brick of our architectures is our equivariant point cloud convolution layer described in figure (1). It has been shown in [10] that simply concatenate each conv layer of TFN with the yzx coordinates of the underlying point clouds without applying non-linearities is enough to ensure that the resulting class of networks is a universal approximator of equivariant features (up to certain choices of radial functions in the definition of the kernels). Here we adopted this construction to compensate for the loss of information induced by the truncation of the features type. We adopt following notations to describe our layers and network architecture:
We denote our equivariant convolution layer by $EC(K, r, d, \ell_{\max}, [u_0, \dots, u_k])$ where:

- K is the number of centroid points at which to apply equivariant kernels.
- r is the radius of the K convolution windows.
- d is the number of shells of the kernels.
- $\ell_{\max}$ is the highest equivariant output type of the layer.
- u_0 is the number of units of the dense layers applied to the output of the equivariant convolution.
- $[u_1, \dots, u_k]$ are the number of units of the MLP layers.

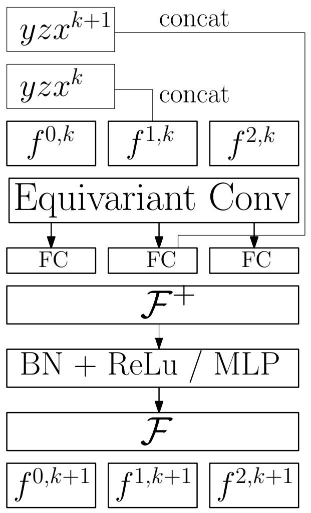

Figure 1. Our convolution layer takes a dictionary of tensors of equivariant features $(f^{\ell,k})_{\ell,k}$ for $\ell \in [\![0, \ell^{k}_{\max}]\!]$ as input. First we concatenate type 1 features with the yzx coordinates of the underlying point cloud, then we perform equivariant convolution (9) and again concatenate type 1 output features with the yzx coordinates of the next point cloud. We apply inverse SHT from Eq. (17) followed by a batch norm layer and ReLu activation optionally followed by an MLP. Finally we obtain a new set of equvariant features $(f^{\ell,k+1})_{\ell,k}$ for $\ell \in [\![0, \ell^{k+1}_{\max}]\!]$ by computing SHT from Eq. (18).

We denote by $FC(k, d)$ a fully connected layer with k units followed by a batch normalization and ReLu layer and a dropout layer with dropout rate d. Our classification architecture is as follows:

$$\begin{aligned} &EC(256, 0.2, 3, 3, [32, 32, 32]) \\ &\to EC(64, 0.4, 3, 3, [64, 64, 64]) \\ &\to EC(16, 0.8, 3, 3, [128, 128, 256]) \\ &\to \mathrm{GMP} \to \mathrm{norm} \\ &\to FC(512, 0.5) \to FC(256, 0.5) \to FC(K) \end{aligned} \tag{22}$$

It takes a point cloud with 1024 points index it by a kd-tree and computes coarser point clouds of size 256, 64 and 16 using average kd-tree pooling. Then it computes three successive equivariant convolution layers as in figure (1), taking the the constant signal equal to 1 as a type 0 feature input to the first layer. After the last convolution we have equivariant tensors of types 0 to 3 representing a signal over the 16 points of the last point cloud, we apply a global max pooling layer (21), compute the norms of the resulting global features and pass them through a final two layers MLP for the class prediction.
For segmentation we adopt a U-Net like architecture inspired by the segmentation architecture of [27]. Like our classification architecture we have three equivariant convolution layers going from a point cloud with 2048 points to a coarse point cloud of 32 points. We then up sample the

signal back to 2048 points using feature the propagation layers from [27] described in equation (3). The output of each feature propagation layer is concatenated with the output of the convolutional layer on the corresponding point cloud if available or the yzx coordinates of the input point cloud for the last feature propagation layer. Then we apply fully connected layers over the channel axis and equivariant activation as described in figure (1) (with no MLP to reduce computational cost). We denote our feature propagation layer by $FP(d)$ where d is the number of units of the fully connected layers. Our segmentation architecture is as follows:

$$\begin{aligned} &EC(512, 0.2, 3, 3, [16]) \rightarrow EC(128, 0.4, 3, 3, [32]) \\ &\rightarrow EC(32, 0.8, 3, 3, [64]) \\ &\rightarrow FP(64) \rightarrow FP(32) \rightarrow FP(32) \rightarrow \text{norm} \\ &\rightarrow FC(128, 0.1) \rightarrow FC(128, 0.1) \rightarrow FC(K) \end{aligned} \tag{23}$$

4. Experiments and results

We evaluate our method in point cloud classification and segmentation tasks. We consider three different experimental train / test settings: z/z, $z/\mathrm{SO}(3)$ and $\mathrm{SO}(3)/\mathrm{SO}(3)$ where z stands for aligned data augmented by random rotations around the vertical axis and $\mathrm{SO}(3)$ indicates data augmented by random $\mathrm{SO}(3)$ rotations. The data is randomly rotated at each training iteration. We consider different non-linearities for TFN: 'ReLu' where we simply use a ReLu activation in our equivariant convolution layer (EC) described in Figure (1), 'mlp' where we stack a two layer mlp on top of the ReLu activation, 'norm' which is the original TFN non linearity and 'gated' wich is the gated non linearity from [32]. For each non linearity x we denote TFN[x] the TFN version based on x. Thus TFN[norm] is similar to the original TFN. We use the following abbreviations to refer to the sampling pattern: R for the regular dodecahedron, P for the Pentakis dodecahedron and F for Fibonacci sampling.
We compare to two categories of methods: rotation sensitive and, rotation robust methods which are specifically designed for handling non aligned data. The z/z case offers a baseline for rotation sensitive methods. The $z/\mathrm{SO}(3)$ setting allows to measure the robustness of a method to the observation of new poses unseen during training while, the $\mathrm{SO}(3)/\mathrm{SO}(3)$ is the most practical setting as in practice it is more representative of real world where the pose of objects is not known in advance nor consistent and, it is always possible to apply random rotation augmentation.
For classification we consider two datasets. The ModelNet40 dataset [33] consists of 12,311 synthetic shapes in canonical pose divided into 40 categories with a 9,843/2,468 train / test split and the ScanObjectNN dataset [3] which consists of 2902 point clouds sampled from real-world objects across 15 different categories with a 2309/581 train test split. For segmentation we consider the ShapeNet part dataset [6] which contains 16,881 shapes from 16 classes, annotated with 50 parts labels in total. We trained all our models with a TensorFlow [1] implementation using the Adam optimizer [15] with learning rate 0.001. We used a batch size of 32 for classification reported in section (4.1) and 24 for segmentation reported in section (4.2).

4.1. Classification

Table (1) shows our results for classification on ModelNet40 and compares it to state of the art methods. We considered an alternative variant of our method using pre-alignment with PCA refereed to by the indication (PCA). We trained our models for 200 epochs and observed convergence in around 150 epochs. We report two different metrics, since the ModelNet40 dataset does not have an official validation set we used the test set as validation set and select the best model over training based on validation score. We average the test score over 40 randomly rotated copies of the test set to smooth the variability introduced by random rotations augmentation. We use the same best z/z model for both the z/z and $z/\mathrm{SO}(3)$ scores. We refer to these results with the indication (best). We observed oscillations of validation accuracy even after convergence we included these results to indicate an upper bound of the performance of our models. For fair comparison we also report the average validation accuracy over the last 10 epochs which is more representative of actual performance. While our method is mostly robust to rotations as suggested by the $z/\mathrm{SO}(3)$ setting, we still observe a performance drop but this is due to our pooling and activation being only approximately equivariant in practice. We notice that our PCA pre-aligned version produced more consistent results across the 3 different settings z/z, $z/\mathrm{SO}(3)$ and $\mathrm{SO}(3)/\mathrm{SO}(3)$.
Our method does not quite match the best state of the art methods yet, these methods either rely on richer input data or slightly different representation. EMVnet [12] consider a multi-view image based representation of the shapes based on renderings of meshes. LGR-Net [39] relies on surface normals, SFCNN [28] propose an original approach similar to multi-view by mapping input point-clouds to a sphere. Our results are close to GC-Conv [36] and RI-Framework [20] which both rely only on the point coordinates.
A fundamental question is whether it is enough to stack purely invariant layers for rotation invariant tasks, it has been stated in in GC-Conv [36] that rotation invariant features are usually not as distinctive as non invariant ones. In this work we rely on equivariant layers which unlike invariant layers can transmit directional information to the next layers. Our baseline for rotation invariant layers is SPHNet [24] as we use very similar spherical harmonics based kernels in a 3 layered conv architecture, the difference is that SPHNet takes the norm of each equivariant features right

after convolution instead of computing equivariant convolution. These observation together with our results showing a noticeable improvement over SPHNet suggest that simply stacking invariant layers is not optimal.
We propose an ablation study of our ModelNet40 experiment in table (2) to compare our equivariant non linearity (19) to the norm non linearity (11) introduced in [30] and gated non linearity (12) introduced in [32]. We also study the impact of the number of samples and the sampling pattern on the performance of our non linearity. For better comparison with other non-linearities we consider TFN[ReLu] so that we only measure the effect of our new non linearity. We apply the norm and gated non linearites on top of the fully connected layers following the equivariant convolution step. To produce a global descriptor layers in our classification architecture we apply norm in a pointwise fashion, reducing equivariant features to invariant scalars, allowing to apply rotation invariant global max pooling. Results are shown in table 2, we observe a noticeable improvement over the norm and gated non-linearities baselines and interestingly the results of TFN[norm] are very close to the results of SPHNet [24] shown in table (1). Although the later uses purely invariant layers both rely on non similar linearities applied only to the norm of equivariant features. As for the effect of the number of samples we observe as expected that more samples improves robustness to rotations as shown by the $z/\mathrm{SO}(3)$ setting. We add a qualitative study of the equivariance error w.r.t. the number of samples in the supplementary material.
We report results on ScanObjectNN [3] in table (4). We compare our method to several state of the art methods, in particular the recent $\mathrm{SE}(3)$-Transformers framework [13] introducing a transformer architecture built on top of TFN. We follow the same experimental setting as [13], training our network until convergence (100 epochs) and average over 5 runs on the basic OBJ_ONLY version of the dataset with no rotation augmentation and using the z coordinate as a 0 type input (indicated by +z). We achieve comparable results to $\mathrm{SE}(3)$-Transf.+z [13] and noticeably better results than TFN+z. We note that $\mathrm{SE}(3)$-Transf uses less input points but in (Appendix D.1.2) of [13] the authors report that increasing the number of point did not improve the performance. We also adapted the radius of our kernels(0.14, 0.28, 0.56 instead of 0.2, 0.4, 0.8 for Modelnet40.

4.2. Segmentation

For our segmentation experiment on ShapeNet [6] we used the official train/val/test split we select the best model according to the the validation accuracy over a 150 epochs training which is a fair metric since validation and test sets are distinct. Table (3) compares our per class and average class intersection over union scores (IoU) to state of the art methods. Our method falls in between GC-Conv [36] and RI-Framework [20] which both use only xyz coordinates of the point clouds as we do unlike the best performing method LGR-Net [39] which also uses normals. We again observe a noticeable improvement over other TFN non-linearities.

Rot-sensitive Methods	z/z	$z/\mathrm{SO}(3)$	$\mathrm{SO}(3)/\mathrm{SO}(3)$
SubVolSup MO [26]	89.5	45.5	85.0
PointNet [25]	87.0	21.6	80.3
PointNet++ [27]	91.8	28.4	85
PointCNN [21]	92.5	41.2	84.5
DGCNN [31]	92.9	20.0	81.1
ShellNet [38]	93.1	19.9	87.8
PCNN [4]	92.3	11.9	85.1
Rot-robust Methods	z/z	$z/\mathrm{SO}(3)$	$\mathrm{SO}(3)/\mathrm{SO}(3)$
EMVnet [12]	94.4	-	91.1
SPHNet [24]	87.7	86.6	87.6
Spherical-CNN [11]	88.9	76.7	86.9
a^3S-CNN [22]	89.6	87.9	88.7
ClusterNet [7]	87.1	87.1	87.1
RI-Conv [37]	86.5	86.4	86.4
GC-Conv [36]	89.0	89.1	89.2
RI-Framework [20]	89.4	89.4	89.3
LGR-Net [39]	90.9	90.9	91.1
SFCNN(xyz) [28]	91.4	84.8	90.1
SFCNN(xyz+n) [28]	92.3	85.3	91.0
TFN[mlp] P	89.4	87.9	89.0
TFN[mlp] P + PCA	88.6	88.7	88.8
TFN[mlp] P (best)	90.5	88.2	89.3
TFN[mlp] P + PCA (best)	89.7	89.7	89.7

Table 1. Test classification accuracy on the modelnet40 dataset [33] in three train / test scenarios. z stands for aligned data augmented by random rotations around the vertical axis and $\mathrm{SO}(3)$ indicates data augmented by random $\mathrm{SO}(3)$ rotations. (best) stands for the best model selected over 200 training epochs.

Methods	# samples	z/z	$z/\mathrm{SO}(3)$	$\frac{\mathrm{SO}(3)}{\mathrm{SO}(3)}$	time
TFN[norm]	-	87.7	86.7	87.2	27s
TFN[gated]	-	88.3	86.7	87.8	27s
TFN[ReLu] R	20	89.1	87.0	88.5	25s
TFN[ReLu] P	32	89.1	88.1	88.1	25s
TFN[ReLu] F	16	89.8	81.8	88.6	25s
TFN[ReLu] F	32	89.6	85.4	88.8	25s
TFN[ReLu] F	64	89.3	87.6	88.2	26s
TFN[norm] (b)	-	88.5	85.3	87.6	27s
TFN[gated] (b)	-	88.7	87.1	88.5	27s
TFN[ReLu] R (b)	20	89.6	86.5	89.0	25s
TFN[ReLu] P (b)	32	89.8	87.2	89.1	25s
TFN[ReLu] F (b)	16	90.3	83.4	88.6	25s
TFN[ReLu] F (b)	32	89.6	84.7	89.0	25s
TFN[ReLu] F (b)	64	89.9	87.2	89.2	26s

Table 2. Comparison of equiavriant non-linearities for TFN on the ModelNet40 classification dataset. ReLu is our non linearity (19), norm is the norm non linearity of Eq. (11) introduced in [30], gated is the gated non linearity of Eq. (12) introduced in [32]. (b) stands for the best model selected over 200 training epochs. Timings in seconds for 1 epoch are given for an RTX 3090 gpu.

method	aero	bag	cap	car	chair	earph.	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate	table	avg. mIoU
$z/\mathrm{SO}(3)$																	
PointNet [25]	40.4	48.1	46.3	24.5	45.1	39.4	29.2	42.6	52.7	36.7	21.2	55.0	29.7	26.6	32.1	35.8	37.8
PointNet++ [27]	51.3	66.0	50.8	25.2	66.7	27.7	29.7	65.6	59.7	70.1	17.2	67.3	49.9	23.4	43.8	57.6	48.3
PointCNN [21]	21.8	52.0	52.1	23.6	29.4	18.2	40.7	36.9	51.1	33.1	18.9	48.0	23.0	27.7	38.6	39.9	34.7
DGCNN [31]	37.0	50.2	38.5	24.1	43.9	32.3	23.7	48.6	54.8	28.7	17.8	74.4	25.2	24.1	43.1	32.3	37.4
ShellNet [38]	55.8	59.4	49.6	26.5	40.3	51.2	53.8	52.8	59.2	41.8	28.9	71.4	37.9	49.1	40.9	37.3	47.2
RI-Conv [37]	80.6	80.0	70.8	68.8	86.8	70.3	87.3	84.7	77.8	80.6	57.4	91.2	71.5	52.3	66.5	78.4	75.3
GC-Conv [36]	80.9	82.6	81.0	70.2	88.4	70.6	87.1	87.2	81.8	78.9	58.7	91.0	77.9	52.3	66.8	80.3	77.2
RI-Framework [20]	81.4	82.3	86.3	75.3	88.5	72.8	90.3	82.1	81.3	81.9	67.5	92.6	75.5	54.8	75.1	78.9	79.2
LGR-Net [39]	81.5	80.5	81.4	75.5	87.4	72.6	88.7	83.4	83.1	86.8	66.2	92.9	76.8	62.9	80.0	80.0	80.0
TFN[norm]	80.1	76.5	84.7	71.0	88.1	67.6	90.1	80.8	75.8	79.3	57.6	92.5	73.9	58.0	73.6	78.7	76.8
TFN[gated]	80.0	78.8	82.4	71.1	88.4	73.5	89.9	75.5	76.3	78.9	60.0	91.2	75.4	48.3	72.9	79.2	77.0
TFN[ReLu]	81.1	77.8	79.8	74.5	89.1	77.2	90.8	82.8	77.7	78.6	60.3	93.4	77.0	54.7	74.4	79.5	78.1
TFN[ReLu] + PCA	80.9	75.2	81.9	73.8	89.0	61.0	90.8	83.0	76.9	80.2	58.5	92.8	76.3	54.0	74.5	79.1	76.7
$\mathrm{SO}(3)/\mathrm{SO}(3)$																	
PointNet [25]	81.6	68.7	74.0	70.3	87.6	68.5	88.9	80.0	74.9	83.6	56.5	77.6	75.2	53.9	69.4	79.9	74.4
PointNet++[27]	79.5	71.6	87.7	70.7	88.8	64.9	88.8	78.1	79.2	94.9	54.3	92.0	76.4	50.3	68.4	81.0	76.7
PointCNN [21]	78.0	80.1	78.2	68.2	81.2	70.2	82.0	70.6	68.9	80.8	48.6	77.3	63.2	50.6	63.2	82.0	71.4
DGCNN [31]	77.7	71.8	77.7	55.2	87.3	68.7	88.7	85.5	81.8	81.3	36.2	86.0	77.3	51.6	65.3	80.2	73.3
ShellNet [38]	79.0	79.6	80.2	64.1	87.4	71.3	88.8	81.9	79.1	95.1	57.2	91.2	69.8	55.8	73.0	79.3	77.1
RI-Conv [37]	80.6	80.2	70.7	68.8	86.8	70.4	87.2	84.3	78.0	80.1	57.3	91.2	71.3	52.1	66.6	78.5	75.3
GC-Conv [36]	81.2	82.6	81.6	70.2	88.6	70.6	86.2	86.6	81.6	79.6	58.9	90.8	76.8	53.2	67.2	81.6	77.3
RI-Framework [20]	81.4	84.5	85.1	75.0	88.2	72.4	90.7	84.4	80.3	84.0	68.8	92.6	76.1	52.1	74.1	80.0	79.4
LGR-Net [39]	81.7	78.1	82.5	75.1	87.6	74.5	89.4	86.1	83.0	86.4	65.3	92.6	75.2	64.1	79.8	80.5	80.1
Ours norm	79.9	79.6	81.6	68.0	87.6	73.4	90.3	82.3	76.2	79.8	56.0	92.2	73.3	46.1	74.1	78.4	76.2
TFN[gated]	79.6	77.8	81.2	71.1	88.4	75.0	90.2	81.9	78.0	77.7	61.2	92.5	75.4	52.1	73.2	78.8	76.9
TFN[ReLu]	80.8	74.5	82.8	74.4	89.4	75.7	90.6	81.0	77.8	80.5	62.4	93.3	78.5	55.8	74.7	79.5	78.2
TFN[ReLu] + PCA	80.3	77.3	82.6	74.7	88.8	76.3	90.7	81.7	77.4	82.4	60.7	93.2	79.4	54.3	74.7	79.6	78.4

Table 3. Test segmentation mean per class and mean intersection over union (IoU) on the ShapeNet shape segmentation benchmark [6]. z stands for aligned data augmented by random rotations around the vertical axis and $\mathrm{SO}(3)$ indicates data augmented by random $\mathrm{SO}(3)$ rotations. PCA stands for PCA pre-alignment.

Methods	# points	test acc
DeepSet [35]	1024	71.4
3DmFV [5]	1024	73.8
Set Transformer [19]	1024	74.1
PointNet [25]	1024	79.2
SpiderCNN [34]	1024	79.5
TFN+z [30]	128	81.0
PointNet++ [27]	1024	84.3
SE(3)-Transf.+z [13]	128	85.0
PointCNN [21]	1024	85.5
DGCNN [31]	1024	86.2
PointGLR [29]	1024	87.2
TFN[norm] + z	1024	80.7
TFN[norm] + z (best)	1024	83.2
TFN[gated] + z	1024	81.1
TFN[gated] + z (best)	1024	83.0
TFN[MLP]+z	1024	85.3
TFN[MLP]+z (best)	1024	88.1

Table 4. Classification accuracy on the ScanObjectNN [3] OBJ_ONLY dataset the z stands for z coordinate as 0 type feature input. The central column indicates the number of input points. (best) stands for the best model selected over 100 training epochs. TFN is the original TFN implementation while TFN[norm] is our implementation of TFN.

5. Conclusion and future work

In this work we highlight the importance of the design of equivariant non-linearities. We introduced a novel rotation equivariant non linearity for Tensor Field Networks [30] based on a functional interpretation of equivariant features. In future work we would like to explore this idea further. An interesting direction would be to use equivariant features to produce rotation equivariant functions over the 3D space with potential applications such as rotation equivariant auto encoder based on implicit surface representation. In our experiments we observed that better performing methods such as [39, 28] often use point cloud normals. Tensor Field Networks are direction agnostic as they are $\mathrm{SO}(3)$ equivariant. Adding local direction information like normals allows to reduce the dimension of the rotation group by restricting $\mathrm{SO}(3)$ tor rotations around the normals which might help designing more informative and cost effective equivariant non-linearities.

Acknowledgements

This work was supported by a Vannevar Bush Faculty Fellowship, a grant from the Stanford-Toyota Center for AI Research, and gifts from Adobe, Autodesk and the Amazon AWS ML program.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016. 6

[2] Brandon Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *arXiv preprint arXiv:1906.04015*, 2019. 1, 2, 3

[3] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. *arXiv*, pages arXiv–1908, 2019. 6, 7, 8

[4] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018. 1, 2, 7

[5] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, 3(4):3145–3152, 2018. 8

[6] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1, 6, 7, 8

[7] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4994–5002, 2019. 1, 7

[8] Gregory S Chirikjian, Alexander B Kyatkin, and AC Buckingham. Engineering applications of noncommutative harmonic analysis: with emphasis on rotation and motion groups. *Appl. Mech. Rev.*, 54(6):B97–B98, 2001. 2, 3

[9] Pim de Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. *arXiv preprint arXiv:2003.05425*, 2020. 1

[10] Nadav Dym and Haggai Maron. On the universality of rotation equivariant point cloud networks. *arXiv preprint arXiv:2010.02449*, 2020. 3, 5

[11] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018. 1, 7

[12] Carlos Esteves, Yinshuang Xu, Christine Allen-Blanchette, and Kostas Daniilidis. Equivariant multi-view networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1568–1577, 2019. 1, 6, 7

[13] Fabian B Fuchs, Daniel E Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*, 2020. 7, 8

[14] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 1

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[16] Anthony W Knapp. *Representation theory of semisimple groups: an overview based on examples*, volume 36. Princeton university press, 2001. 2, 3

[17] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch-gordan nets: a fully fourier space spherical convolutional neural network. *arXiv preprint arXiv:1806.09231*, 2018. 1, 2, 3

[18] Leon Lang and Maurice Weiler. A wigner-eckart theorem for group equivariant convolution kernels. *arXiv preprint arXiv:2010.10952*, 2020. 2, 3

[19] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019. 8

[20] 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. *arXiv preprint arXiv:2003.07238*, 2020. 1, 6, 7, 8

[21] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NIPS*, pages 820–830, 2018. 1, 2, 7, 8

[22] Min Liu, Fupin Yao, Chiho Choi, Ayan Sinha, and Karthik Ramani. Deep learning 3d shapes using alt-az anisotropic 2-sphere convolution. In *International Conference on Learning Representations*, 2018. 1, 7

[23] Ricardo Marques, Christian Bouville, Mickaël Ribardière, Luis Paulo Santos, and Kadi Bouatouch. Spherical fibonacci point sets for illumination integrals. In *Computer Graphics Forum*, volume 32, pages 134–143. Wiley Online Library, 2013. 5

[24] Adrien Poulenard, Marie-Julie Rakotosaona, Yann Ponty, and Maks Ovsjanikov. Effective rotation-invariant point cnn with spherical harmonics kernels. In *2019 International Conference on 3D Vision (3DV)*, pages 47–56. IEEE, 2019. 1, 2, 4, 6, 7

[25] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 7, 8

[26] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 7

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on

point sets in a metric space. In *NIPS*, pages 5099–5108, 2017. 1, 2, 5, 6, 7, 8

[28] Yongming Rao, Jiwen Lu, and Jie Zhou. Spherical fractal convolutional neural networks for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 452–460, 2019. 1, 6, 7, 8

[29] Yongming Rao, Jiwen Lu, and Jie Zhou. Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5376–5385, 2020. 8

[30] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018. 1, 2, 3, 4, 7, 8

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018. 1, 2, 7, 8

[32] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *NIPS*, pages 10381–10392, 2018. 1, 2, 3, 4, 6, 7

[33] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proc. CVPR*, pages 1912–1920, 2015. 1, 6, 7

[34] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018. 8

[35] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017. 8

[36] Zhiyuan Zhang, Binh-Son Hua, Wei Chen, Yibin Tian, and Sai-Kit Yeung. Global context aware convolutions for 3d point cloud understanding. *arXiv preprint arXiv:2008.02986*, 2020. 1, 6, 7, 8

[37] Zhiyuan Zhang, Binh-Son Hua, David W Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International Conference on 3D Vision (3DV)*, pages 204–213. IEEE, 2019. 1, 7, 8

[38] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1607–1616, 2019. 7, 8

[39] Chen Zhao, Jiaqi Yang, Xin Xiong, Angfan Zhu, Zhiguo Cao, and Xin Li. Rotation invariant point cloud classification: Where local geometry meets global topology. *arXiv preprint arXiv:1911.00195*, 2019. 1, 6, 7, 8