ZZ-Net: A Universal Rotation Equivariant Architecture for 2D Point Clouds

Georg Bökman[a], Fredrik Kahl[a], Axel Flinth[a,b]
bokman@chalmers.se, fredrik.kahl@chalmers.se, axel.flinth@umu.se
[a]Department of Electrical Engineering, Chalmers University of Technology
[b]Department of Mathematics and Mathematical Statistics, Umeå University

Abstract

In this paper, we are concerned with rotation equivariance on 2D point cloud data. We describe a particular set of functions able to approximate any continuous rotation equivariant and permutation invariant function. Based on this result, we propose a novel neural network architecture for processing 2D point clouds and we prove its universality for approximating functions exhibiting these symmetries.

We also show how to extend the architecture to accept a set of 2D-2D correspondences as indata, while maintaining similar equivariance properties. Experiments are presented on the estimation of essential matrices in stereo vision.

1. Introduction

The need to interpret and process point clouds arises in many different application areas such as autonomous driving, augmented reality and robotics [18]. Basic problem examples in computer vision are classification, segmentation and object detection as well as correspondence problems in multiple view geometry [34] . Considering as input object a point cloud or a pair of point clouds, it is a natural requirement that permuting the order of the points doesn't change the object in question. Such a permutation should therefore not change the way the points are processed. This permutation symmetry needs to be considered when designing a neural network for point cloud input, which is typically done by having *equivariant* network layers. Another possible symmetry is rotation of the point clouds about the origin. For an example of a single point cloud processing task that is rotation equivariant, see Figure 1. We will also consider rotational symmetries for pairs of point clouds.

Equivariance. Let us introduce some notation and provide a formal definition of equivariance. Given a group G we consider sets which exhibit G-symmetries (in a sense to be made precise shortly) and functions between such sets. A G-set is a set X equipped with a G-action, i.e., a group homomorphism φ from G to the group of bijections from X to X. One should think of a G-action φ as a way to relate elements of G to symmetries of X. Typically, we will suppress the group homomorphism in the notation and write gx or g^*x for $\varphi(g)(x)$. An example of a G-set is $\mathbb{C}^m$ acted on by the permutation group S_m. Note that S_m could act on $\mathbb{C}^m$ in different ways and we must specify the action to describe a G-set. The canonical action is to permute the m dimensions, but another obvious action is the trivial action given by $\pi Z = Z$ for all $\pi \in S_m$ and all $Z \in \mathbb{C}^m$. If we have two G-sets X and Y, we say that a function $f : X \to Y$ is (G-)equivariant if it commutes with the G-actions: $f(gx) = gf(x)$. A special case is when the action on Y is trivial and then we call f invariant: $f(gx) = f(x)$. For more information on groups, symmetries and equivariance in general, we refer to e.g. [22].

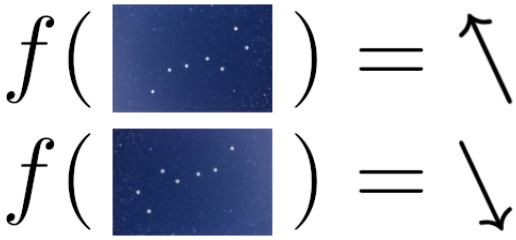

Figure 1. A simple example of rotation equivariance. The illustration shows the task of determining the direction to the North Star given other stars in the night sky. The input is the set of locations of the visible stars in some 2D coordinate frame. Rotation equivariance of the point cloud processor f means here that the determined direction should rotate if the night sky (or the observer) rotates. (Picture of stars [2].)

In this paper, we focus on the group $SO(2) \times S_m$. The $SO(2)$-action on a point cloud is given by rotating all points about the origin and the S_m-action is given by permuting the points.[1] More concretely, we are concerned with functions that are invariant to permutations, but equivariant to

[1]Technical note: These two actions commute with each other and hence define an $SO(2) \times S_m$-action.

rotations. Let us call the set of such functions $\mathcal{R}(m)$.

Additionally, we go further, and describe new results and neural network architectures for the case of clouds of pair of points, or correspondences. In this case, we deal with functions that are permutation invariant, rotation equivariant to one of the clouds, and rotation invariant with respect to the other. We call this set of functions $\mathcal{R}_2(m)$.

An obvious limitation with our work is that we only deal with $SO(2)$-equivariance and not higher order rotations. Still, it is an important case with many different applications. For instance, in many scenarios, invariance with respect to rotation around one axis is the correct model. Another example is essential matrix estimation [16], which we will explore in Section 5.2. Note that the derivations are simplified and the computations can be made more efficient as the group of 2D rotations is commutative, which is not the case for $SO(d)$ with $d > 2$.

The main contributions of this paper are as follows. First, we describe a dense set of equivariant functions on 2D point clouds (Theorem 2). With that set as a basis, we describe a neural network architecture for approximating the function space $\mathcal{R}(m)$ and prove its universality (Theorem 3). We then present how to extend that architecture to also cover $\mathcal{R}_2(m)$ and discuss the extension's universality properties. We test our architecture on a (toy) rotation estimation problem and the estimation of essential matrices in stereo vision.

1.1. Related work

Equivariance for regular image grids has been studied in various settings, ranging from classical CNNs for translation invariance [14, 24] to rotation and rigid transformation invariance [42, 44, 45]. Equivariance on more general domains and under general groups has also been investigated in a recent line of research. In particular there has been a focus on describing linear equivariant functions, which can be alternated with non-linearities to obtain equivariant neural network architectures [1, 4, 5, 12, 23]. Recent surveys of the theory include [3, 15, 43].

There exist a number of high-performing deep learning architectures for 3D point cloud processing, mostly targeted for recognition, classification and segmentation, including methods that do not take rotation equivariance into account [33, 54] and methods that do consider the effects of rotations [7, 13, 31, 37]. The approach most similar in spirit to ours is [47], but while we let every point in the point cloud gather information from all others to obtain rotation invariant and permutation equivariant features, they use the sorted Gram matrix of local neighbourhoods to obtain local rotation and permutation invariant features. They do not prove the universality of their approach.

We focus on 2D rather than 3D. While the approaches for the 3D case could be modified to apply to the 2D case as well, doing so would not take advantage of the fact that the 2D case is simpler. Specifically, all rotations in 2D commute and this fact plays a crucial role in our proofs.

Our work is inspired by fundamental theoretical results in machine learning which aim to characterize equivariant point cloud networks. In the seminal work of [52], all permutation equivariant functions were shown to belong to a particular family of functions from which equivariant network architectures can be constructed. In more recent work, the theory has been further developed and additional symmetries have been considered [20, 29, 30, 41, 50]. In [8], the authors present a method for proving universality for rotation equivariant point cloud networks in 3D. Their proof technique is applicable to networks which allow latent features consisting of arbitrary high order tensors, such as for e.g Tensor Field Networks [37]. In contrast, our networks only need to handle tensors of order two.

While finalizing this work, we were made aware of the concurrent papers [40, 49], with an approach that is related to ours. In fact, Proposition 10 of [40] is similar to our Theorem 2 but for the group $O(2)$ instead of $SO(2)$ (in fact, they deal with a d-dimensional underlying space and the group $O(d)$). In particular, we make a more thorough description and analysis of neural network architectures.

From an application point-of-view, we are interested in correspondence problems and more generally, robust fitting problems in multiple view geometry. State-of-the-art deep learning approaches in this context include early work such as CNe [51] and OANet [53] but also the more recent approaches ACNe [36] where attentive context normalization is shown to improve permutation-equivariant learning and T-Net [55] which also consists of a permutation equivariant network that is able to capture both global and channel-wise contextual information. However, these methods only incorporate permutation equivariance, which make them dependent on the coordinate frame of the points. We give experimental comparisons to some of these approaches.

Notation. Throughout the entire paper, we will identify $\mathbb{R}^2$ with $\mathbb{C}$. The group $SO(2)$ of rotations is then naturally identified with the unit circle $\mathbb{S} \subset \mathbb{C}$. To keep things simple, we understand point clouds as vectors $Z = (z_0, \dots, z_{m-1}) \in \mathbb{C}^m$, where m is the number of points. Note that the action of $SO(2)$ on $\mathbb{C}^m$ can be simply written θZ, where $\theta \in \mathbb{S}$ and that this can be equivalently read as complex multiplication or an action of the rotation group. We write $[m]$ for the set of indices from 0 to $m-1$. The group of permutations is denoted S_m, and for $\pi \in S_m$, we let $\pi^* Z$ denote the permuted version of Z, i.e., $[\pi^* Z]_i = Z_{\pi^{-1}(i)}$. As in [29], we extend the latter to tensors: for $T \in (\mathbb{C}^m)^{\otimes 2}$, $[\pi^* T]_{ij} = T_{\pi^{-1}(i)\pi^{-1}(j)}$. Let us further denote the subgroup of permutations which fix the 0-element, i.e., $\{\pi \in S_m \,|\, \pi(0) = 0\}$ with $\mathrm{Stab}(0)$, which is called the *stabilizer* of 0. Finally, we let $\tau_i \in S_m$ be the transposition of i and 0.

2. Approximating functions in $\mathcal{R}(m)$

In this section we describe the theory underlying our permutation invariant, rotation equivariant neural network architecture. We denote the set of continuous rotation equivariant and permutation invariant functions, i.e., functions $f : \mathbb{C}^m \to \mathbb{C}$ with $f(\theta\pi^* Z) = \theta f(Z)$ for all $\pi \in S_m$ and $\theta \in \mathbb{S}$ with $\mathcal{R}(m)$.[2] Throughout the paper, m is fixed.

2.1. A dense set of functions in $\mathcal{R}(m)$

To get an idea of how to design a network for approximating functions on $\mathcal{R}(m)$, let us look at the DeepSet [52], or PointNet [33], architectures. In a nutshell, the reason that they are universal for approximating permutation invariant functions is that all such functions can be written as $\chi(\sum_{i\in[m]} \varrho(z_i))$ for some $K \in \mathbb{N}$ and functions $\varrho : \mathbb{C} \to \mathbb{R}^K$ and $\chi : \mathbb{R}^K \to \mathbb{C}$. A natural Ansatz for approximating functions in $\mathcal{R}(m)$ is therefore to use a network of the same structure, but letting ϱ and χ be rotation equivariant. Unfortunately, this simple idea will provably not succeed.

Proposition 1. *For any $m \geq 5$, there are functions $f \in \mathcal{R}(m)$ that cannot be uniformly approximated only using functions as $\chi(\sum_{i\in[m]} \varrho(z_i))$ for χ and ϱ rotation equivariant.*

The technical proof is given in Section A.2 in the supplementary material. An idea for a design is instead given by the following theorem.

Theorem 2. *The set of functions on the form*

$$f(Z) = \sum_{i\in[m]} \gamma(\tau_i^* Z) z_i, \tag{1}$$

where γ is an arbitrary continuous, rotation invariant and $\mathrm{Stab}(0)$-invariant function, is dense in $\mathcal{R}(m)$.

We remind the reader that τ_i is the transposition of 0 and i. The proof of Theorem 2, which rests upon the density of polynomials and algebraic manipulations of them, is found in Section A.3 in the supplementary material. Let us here instead concentrate on intuitively explaining it.

It is fruitful to interpret the values $(\gamma(\tau_i^* Z))_{i\in[m]}$ as scaled rotations $c_i\theta_i$, with $c_i \in \mathbb{R}$ and $\theta_i \in \mathbb{S}$. Considering this, (1) can be interpreted as a *weighted centroid* of the point cloud, where each point can be individually rotated prior to calculation of the weighted centroid.

To calculate the rotation invariant weight $\gamma(\tau_i^* Z)$ for point z_i, we are allowed to examine the entire cloud, and not only z_i. Hence, (1) can be interpreted as an attention mechanism (compare, e.g., [13, 19, 25, 36, 39, 46, 48]) – when calculating 'its' weight, z_i can attend to all other points in the network. It does not however do so in an arbitrary fashion: when calculating $\gamma(Z)$, because of the $\mathrm{Stab}(0)$-invariance, the point z_0 takes a special role, but the collective $(z_i)_{i\geq 1}$ is treated like a set. In the vector $\tau_i^* Z$, the special, first, position is occupied by z_i. Hence, when z_i calculate 'its' weight, it is allowed to attend to its own position z_i and to the positions of the rest of the points $(z_j)_{j\neq i}$ as a set. Finally, note that the weight calculation function γ is shared by all the points.

2.2. A universal architecture for $\mathcal{R}(m)$

We now describe how a neural network for approximation of functions in $\mathcal{R}(m)$ can be built. In the light of Theorem 2, we should design a weight unit $\alpha : \mathbb{C}^m \to \mathbb{C}$ which is invariant to both rotations and $\mathrm{Stab}(0)$-permutations, approximating the function γ. As for the rotation invariance, we propose to let the network simply act on the tensor $Z \otimes \overline{Z} = (z_i \overline{z}_j)_{i,j\in[m]}$ instead of Z – since $Z \otimes \overline{Z}$ is invariant to rotations of the network, the entire network will then automatically also be. Note that the real part of $Z \otimes \overline{Z}$ is the Gram matrix of scalar products $(\langle z_i, z_j \rangle)_{i,j\in[m]}$, where we see the z_i as vectors in $\mathbb{R}^2$. This strategy hence has clear connections to [47], which uses *sorted* Gram matrices of local neighbourhoods. Compared to them, we apply a different way of handling the $\mathrm{Stab}(0)$-invariance. We follow a canonical design idea for equivariant networks – first alternately apply equivariant linear layers and pointwise nonlinearities, add an invarizing step, and thereafter apply fully connected layers. We denote the resulting set of neural networks $\mathcal{NS}(m)$.[3] In the following closer description, 'linear layer' always refer to a real-linear layer with bias term.

The $\mathcal{NS}(m)$ architecture is constructed as follows (cf. Figure 2):

Early layers. The very first layer consists of applying a $\mathrm{Stab}(0)$-equivariant linear layer

$$B_0 : (\mathbb{C}^m)^{\otimes 2} \to (\mathbb{C}^m)^{\ell_1}$$

to $Z \otimes \overline{Z}$. Here, as in the following, ℓ_j refers to the number of channels in layer j. Then, a nonlinearity $\rho : \mathbb{C} \to \mathbb{C}$ is applied pointwise, i.e., $\rho(X)_i = \rho(x_i)$. Concretely, we use a standard activation function separately applied to real- and imaginary parts.

Subsequently, L $\mathrm{Stab}(0)$-equivariant layers

$$B_i : (\mathbb{C}^m)^{\ell_i} \to (\mathbb{C}^m)^{\ell_{i+1}}$$

are applied in alternation with a pointwise linearity $\rho : \mathbb{C} \to \mathbb{C}$. The final output of the early layers is a multivector $V \in (\mathbb{C}^m)^{\ell_L}$

Invarization step. Next, we calculate $v = \sum_{i\in[m]} v_i$. Note that this transforms the $\mathrm{Stab}(0)$-equivariant multivector V into a $\mathrm{Stab}(0)$-invariant multiscalar v. In fact, we

[2] $\mathcal{R}$ for *rotation*.

[3] $\mathcal{N}$ for *network* and $\mathcal{S}$ for *stabilizer*.

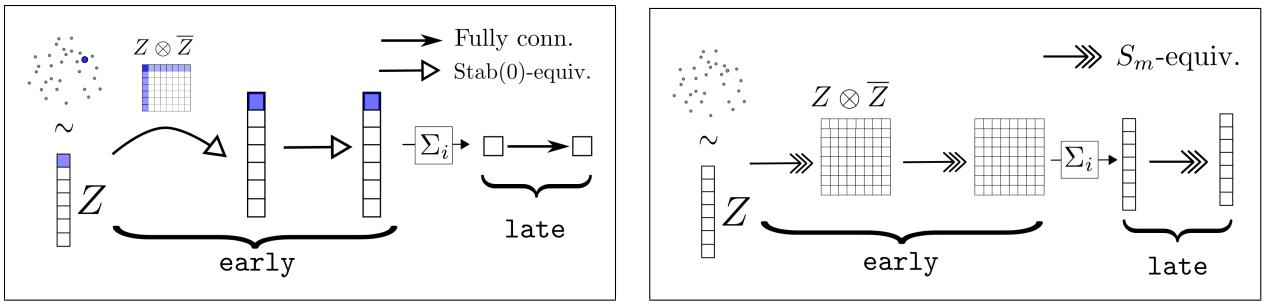

Figure 2. The $\mathcal{NS}(m)$ (left) and $\mathcal{NS}^+(m)$ (right) architectures. Note that one of the points take on a special role in the left architecture, whence the highlightings.

could here instead apply any $\mathrm{Stab}(0)$-invariant functional, but we concentrate on summation for simplicity.

Late layers. Finally, a number of fully connected layers are applied to v.

Importantly, the very first linear layer maps into a space of multivectors, rather than multitensors. This saves a significant amount of memory compared to letting all early layers handle multitensors, which would be the naive way to process the tensor $Z \otimes \overline{Z}$. In fact, it is even possible to apply the first layer without explicitly calculating $Z \otimes \overline{Z}$ – see Section C of the supplementary material.

When implementing $\mathcal{NS}(m)$, one of course needs a way to parametrize the $\mathrm{Stab}(0)$-equivariant linear layers. In Section A.4 of the supplementary material, heavily building on the results from [29] about permutation equivariant linear maps, we provide such a parametrization. It is not needed to know this construction in order to follow the rest of the paper. Let us just note that the number of parameters needed to describe each input-output-channel pair is independent of m (just as for the permutation invariant layers in [29]).

In order to build a provably universal architecture for $\mathcal{R}(m)$, it turns out that the above approximation of the γ-function is not enough. We instead need to add another component, a 'vector unit' $\psi : \mathbb{C} \to \mathbb{C}$ acting on the individual points z_i. These units use fully connected *complex-linear* linearities *without bias* and complex ReLUs $\rho_{\mathbb{C}}$,

$$\rho_{\mathbb{C}}(z, \eta) = \mathrm{ReLU}(|z| - \eta)_{\frac{z}{|z|}}$$

as nonlinearities. Here, $\eta \in \mathbb{R}_+$ is a learnable parameter, and ReLU is the real ReLU. Note that $\rho_{\mathbb{C}}$ is rotation equivariant. Since the complex-linear maps also are, the entire ψ-unit automatically becomes rotation equivariant. Let us call the set of such rotation equivariant networks $\mathcal{NC}$.[4]

Using α-units from $\mathcal{NS}(m)$ and ψ-units in $\mathcal{NC}$, we may now build a set $\mathcal{NR}(m)$[5] of rotation equivariant, permutation invariant Ψ networks through

$$\Psi(Z) = \sum_{i \in [m]} \alpha(\tau_i^* Z)\psi(z_i). \tag{2}$$

Our main result is now that this architecture is universal for $\mathcal{R}(m)$.

Theorem 3. *$\mathcal{NR}(m)$ is universal for $\mathcal{R}(m)$.*

Proof Sketch. The entire proof is too long to present here, and is postponed to Section A.5 of the supplementary material. Let's however sketch it.

Step 1: Universality of $\mathcal{NS}(m)$. First, one proves that for any $\epsilon > 0$, $\mathcal{NS}(m)$ is dense in the set of $\mathrm{Stab}(0)$- and rotation invariant function *when restricting to point clouds with* $|z_0| > \epsilon$. Intuitively, we apply the Stone-Weierstrass Theorem [35, 7.32] to show that α can approximate any function of the form $\phi(|z_0|^2, z_0\overline{Z})$, where ϕ is permutation invariant with respect to the second argument. Since we are only concerned with the case of $|z_0| \neq 0$, the map $Z \to (z_0, z_0\overline{Z})$ is injective. From that, we obtain the claim.

Step 2: Universality of $\mathcal{NR}(m)$. The first step shows that for any fixed $\epsilon > 0$, α can be chosen so that $\alpha(\tau_i^* Z) \approx \gamma(\tau_i^* Z)$ as long as $|z_i| > \epsilon$. However, since the product $\gamma(\tau_i^* Z) \cdot z_i$ is small if $|z_i| < \epsilon$, we can still achieve a good approximation anywhere. This is the technical reason for the inclusion of the vector unit – it can eliminate any problem with large $\alpha(\tau_i^* Z)$-values when z_i is small. □

3. Modifications of the universal architecture

Although the architecture in the previous section is universal, we will modify it in a number of ways before using them for our experiments.

3.1. A richer, parallel architecture

In the $\mathcal{NR}(m)$-nets, note that each permuted version $\tau_i^* Z$ of the cloud is sent through the α-unit individually. It

[4] $\mathcal{N}$ for *network* and $\mathcal{C}$ for *complex*.

[5] $\mathcal{N}$ for *network* and $\mathcal{R}$ for *rotation*.

would intuitively be better to calculate all weights in parallel, and in that process let the weight values 'communicate' with each other. A simple way to achieve this is the following modification, which we denote $\mathcal{NS}^+(m)$.

The $\mathcal{NS}^+(m)$ architecture consists of the following: (see also Figure 2).

Early layers. Apply an S_m-equivariant linear layer

$$B_0^+ \,:\, (\mathbb{C}^m)^{\otimes 2} \to ((\mathbb{C}^m)^{\otimes 2})^{\ell_1}$$

to $Z \otimes \overline{Z}$. Subsequently, apply, in alternation, a point-wise non-linearity and S_m-equivariant layers

$$B_i^+ \,:\, ((\mathbb{C}^m)^{\otimes 2})^{\ell_i} \to ((\mathbb{C}^m)^{\otimes 2})^{\ell_{i+1}}.$$

The final output of the early layers is then a multitensor $T \in ((\mathbb{C}^m)^{\otimes 2})^{\ell_L}$.

Invarization step. Next, $V = (\sum_{j\in[m]} T_{ij})_{i\in[m]}$ is calculated, which transforms the S_m-equivariant multitensor T to an S_m-equivariant multivector V.

Late layers. Now apply, in alternation, S_m-equivariant layers $C_i : (\mathbb{C}^m)^{\ell_{L+i}} \to (\mathbb{C}^m)^{\ell_{L+i+1}}$ and pointwise non-linearities. The final network output is $\alpha^+(Z) \in \mathbb{C}^m$.

We also modify the architecture for calculating the ψ-units: We still use $\rho_{\mathbb{C}}$ as the non-linearity and apply $\mathbb{C}$-linear layers, however S_m-equivariant such to the entire cloud Z. The final output of such networks is thus a vector $\psi^+(Z) \in \mathbb{C}^m$. The set of these networks are called $\mathcal{NC}^+$.

Given an $\alpha^+ \in \mathcal{NS}^+(m)$ and a $\psi^+ \in \mathcal{NC}^+$, we now build a network Ψ^+ through

$$\Psi^+(Z) = \sum_{i\in[m]} \alpha^+(Z)_i \cdot \psi^+(Z)_i$$

Let us denote the set of these networks $\mathcal{NR}^+(m)$. These networks are still equivariant, and are at least as expressive as the non-modified ones.

Proposition 4. *(i) The new architecture has the correct equivariance, i.e., $\mathcal{NR}^+(m) \subseteq \mathcal{R}(m)$.*
(ii) The new architecture is at least as expressive as the non-modified, i.e., $\mathcal{NR}(m) \subseteq \mathcal{NR}^+(m)$.

See Section A.6 in the supplement for a proof.

It does take more parameters to parametrize each input-output-channel pair of the linear layers in the $\mathcal{NR}^+(m)$, but this can be compensated by using less input-output-channel pairs. As for the memory requirements, we have to handle 2-tensors in memory, which leads to a quadratic cost. This is worse than the $\mathcal{NR}(m)$-architecture, whose memory cost is only linear. However, recall that we need to calculate m values $\alpha(\tau_i^* Z)$, $i \in [m]$ for each application of the network. If we want to parallelize those calculations, which we should do for efficiency, we need to handle m vectors, again resulting in a quadratic memory cost.

A subtle, but nonetheless reasonable, reason for using the $\mathcal{NR}^+(m)$-architecture instead of the $\mathcal{NR}(m)$ architecture is that it allows for more exchange of information between the points. As an example, notice that when calculating the weight $\alpha(\tau_i^* Z)$, each early layer in an $\mathcal{NR}(m)$-net is only allowed to attend to one vector, which can be seen as a preliminary version of the vector weight. In the $\mathcal{NR}(m)^+$-architecture, it is additionally allowed to attend to all the 'preliminary weight vectors', i.e., the other columns of the input tensor (as a set). This arguably makes the modified architecture more versatile.

4. Approximating functions in $\mathcal{R}_2(m)$

In our experiments, we will actually consider tasks which take *pairs* (Z, X) of point clouds as indata. Thereby, we assume that for each i, the points z_i and x_i correspond to each other, meaning that we only get invariance towards *simultaneous permutations of both clouds*. The tasks we consider will be (or will be transformed into ones that are) rotation equivariant with respect to one cloud, and rotation invariant with respect to the other. That is, we will have to approximate functions f such that for every $\pi \in S_m$ and $\theta, \omega \in \mathbb{S}$, we have

$$f(\theta\pi^* Z, \omega\pi^* X) = \theta f(Z, X).$$

We denote the set of such functions $\mathcal{R}_2(m)$.

We can use the same ideas as above to build an architecture for them. We propose to use the exact same scheme, with the only difference that the very first layer L of the α-unit depends on $Z \otimes \overline{Z}$ and $X \otimes \overline{X}$, as

$$L(Z, X) = A(Z \otimes \overline{Z}) + B(X \otimes \overline{X}),$$

where A and B are linear layers of the same flavor as for $\mathcal{NR}(m)$ and $\mathcal{NR}(m)^+$, respectively. This yields architectures $\mathcal{NR}_2(m)$ and $\mathcal{NR}_2^+(m)$. In Section A.7 of the supplementary material, we prove that $\mathcal{NR}_2(m)$ is *not* dense for the whole of $\mathcal{R}_2(m)$. We however also prove that if we only consider cloud pairs (Z, X) for which no points close to the origin in X correspond to points far away from the origin in Z, we again obtain universality for both versions.

4.1. A deeper architecture

We can easily combine several weight and vector units $\alpha_k \in \mathcal{NS}^+(m)$, $\psi_k \in \mathcal{NC}^+$, to build an iterative architecture. If $Z = Z^0$ is the input cloud, we iteratively define new clouds Z^k through

$$z_i^{k+1} = \alpha_k^+(Z^k)_i \cdot \psi_k^+(Z^k)_i,$$

$i \in [m]$. A particular case where such chains of units can be especially beneficial is the case when the cloud is filled with outliers. The weight units of early layers can then be

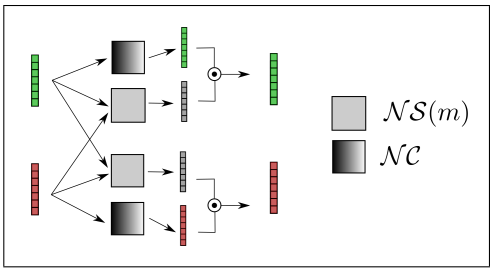

Figure 3. The architecture of a ZZ-unit. Two clouds are fed into $SO(2)$-invariant weight units and $SO(2)$-equivariant vector units, and then combined to produce a new pair of clouds. Best viewed in color.

used to filter those out, by giving the outliers small weights. They will then cluster around the origin, which can safely be ignored by later weight units. This is in spirit similar to (attentive) context normalization [36, 51].

In the cloud pair case, we can iteratively construct new pairs of clouds by chaining pairs of weight and vector units (see Figure 3):

$$z_i^{k+1} = \alpha_k^+(Z^k, X^k)_i \cdot \psi_k^+(Z^k)_i,$$
$$x_i^{k+1} = \beta_k^+(X^k, Z^k)_i \cdot \phi_k^+(X^k)_i.$$

The final output of such a network is then a pair of scalars $(F_0(Z,X), F_1(X,Z))$, where the first scalar is equivariant to rotations in the first cloud, and invariant to rotations in the second, and vice versa. If we let $\alpha_k = \beta_k$ and $\psi_k = \phi_k$, we will even obtain a network which is equivariant to switching the pairs. This is the version we are using in our experiments. Since the weight-units are using tensors of the form $Z \otimes \overline{Z}$ as input, we will refer to such layers as *ZZ-units*.

To obtain a rotation equivariant output of the network, we sum over i in the final (respective) cloud. The set of such obtained architecture will be referred to as *ZZ-nets*.

4.2. Limitations of the architecture

Although our architecture is provably universal, it has its limitations. First and foremost, it operates on tensors rather than vectors, making its memory requirement quadratic in the number of points per cloud. Secondly, all linear layers of our architecture are global in nature, which could hurt performance.

A simple way to mitigate these issues would be to let the weight units α only operate on the nearest neighbors to z_i when calculating the weight for i – we would then return to a memory requirement which is linear in the cloud sizes, and induce locality. However, such an architecture would not be universal.

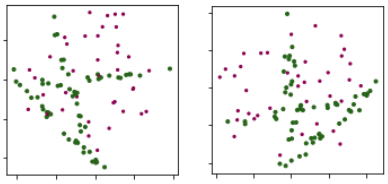

Figure 4. A pair of noisy point clouds as used in the rotation estimation experiments. The inlier points are larger and colored green for illustration purposes. Here the outlier ratio r is 0.4.

5. Experiments

Here we present two experiments to demonstrate our network in action. Further details about the experiments are given in Section B of the supplementary material. Code for the experiments is available at github.com/georg-bn/zz-net.

5.1. Estimating rotations between noisy point clouds

Let us, as a proof of concept more than anything else, test our model on a toy problem: Given a point pair (Z, X), estimate a rotation $R(Z,X)$ so that $X = R(Z,X)Z$. This rotation responds to rotations of either cloud through $R(\theta Z, \omega X) = \omega\bar{\theta}R(Z,X)$.

If Z and X are completely noise-free, this is of course trivial (one can e.g. calculate z_0/x_0). In order to make the problem more challenging, we consider a setting with both inlier and outlier noise.

Data. We synthetically generate data. The details of the data generation are presented in the supplementary B.1. Each point cloud pair (Z, X) contains $m = 100$ correspondences out of which a fraction r are outliers. The inliers lie on a triangle with low-level inlier noise. An example pair is shown in Figure 4.

Models. We test two versions of our model: A 'broad' and a 'deep' one. The 'broad' model consists of a single ZZ-unit, with 2 early and 3 late layers in the weight unit, and 2 layers in the vector unit. The 'deep' model consists of three ZZ-units, where each unit only has 1 early and 2 late weight-layer units, and 1 vector layer, and each such layer is smaller than for the broad model. The broad unit has around 4k, and the deep around 7k, parameters in total.

We train a unit with weights shared, thus outputting two scalars $F(Z,X)$ and $F(X,Z)$. The final output of our model $\widehat{\theta}(X,Z) = F(X,Z)\overline{F(Z,X)} \in \mathbb{C}$ then responds correctly to rotations of either cloud.

For comparison, we implement two alternative models. A PointNet and a simplified version of ACNe [36] which we call 'ACNe–'. They have 34k and 11k parameters respectively. Details about these models are presented in the supplementary B.1.

Experiments. We test each of the models on four outlier ratios: $0.4, 0.6, 0.8$ and 0.85. We use an ℓ_2-loss between

Outlier ratio	$r = 0.4$			$r = 0.6$		
Threshold	1°	5°	10°	1°	5°	10°
Broad ZZ-net	.42	.97	.99	.21	.87	.96
Deep ZZ-net	**.85**	**.99**	**1.0**	**.84**	**.99**	**.99**
PointNet	.02	.45	.78	.03	.34	.67
ACNe–	.05	.63	.96	.04	.54	.90
Outlier ratio	$r = 0.8$			$r = 0.85$		
Threshold	1°	5°	10°	1°	5°	10°
Broad ZZ-net†	.03	.46	.81	.02	.24	.50
Deep ZZ-net	**.32**	**.90**	**.96**	**.11**	**.73**	**.90**
PointNet	.03	.25	.54	.03	.21	.37
ACNe–	.01	.27	.69	.02	.45	.75

Table 1. Results for rotation estimation with varying outlier ratios. †This experiment was stopped early due to severe overfitting.

the ground truth rotation and the output of the networks, and manually tune hyperparameters to optimize the mean error on the validation set.

To evaluate the experiments, we test how many of the ground truth rotations the models can estimate within an error that corresponds to a difference 1°, 5° and 10° for two normalized complex numbers (note that the output of our models is not necessarily normalized), respectively. The results are presented in Table 1. The broad model easily beats the PointNet model, and also the 'ACNe–'-model for low outlier ratios, but starts to struggle against the context-normalization based model for $r = 0.85$. The deep model however easily outperforms all other models.

We notice that some models struggled somewhat on the $r = 0.8$-data set. We had to stop the broad model early due to severe overfitting, and the 'ACNe–' model did worse on the 0.8-set than on the 0.85-set. We suspect that this ultimately boils down to the fact that due to our data generation method, the actual outlier ratios are random. Therefore, the 0.8 dataset could contain some especially hard examples just by chance.

5.2. Essential matrix estimation

The input in the problem is a (noisy) set of calibrated 2D-2D correspondences $\{(p_1, p_2)\}$, where $p_1, p_2 \in \mathbb{R}^2$ are points of interest in two images of the same object. The task is then to estimate the essential matrix $E \in \mathbb{R}^{3\times3}$ such that $\tilde{p}_2^T E \tilde{p}_1 = 0$ for the (correct) correspondences. Here, $\tilde{p}$ is the homogeneous representation of p obtained by adding a third coordinate 1 to p. See [16] for an in depth description of essential matrices. Considering the points $\{p_1\}$ as elements of $\mathbb{C}$ and stacking them into a vector yields the Z vector considered in earlier sections, and similar for $\{p_2\}$ and X.

Rotation equivariance of E. If $\tilde{p}_2^T E \tilde{p}_1 = 0$ for a set of correspondences $\{(p_1, p_2)\}$, it follows that if we rotate p_1 by an image plane rotation $R \in SO(2)$, say to $q_1 = Rp_1$, then $\tilde{p}_2^T E \tilde{R}^T \tilde{q}_1 = 0$ where $\tilde{R} \in SO(3)$ is the rotation obtained by applying R as a rotation around the z-axis. Hence, $E\tilde{R}^T$ is an essential matrix for the correspondences $\{(q_1, p_2)\}$. Similarly one shows that a rotation of p_2 to $q_2 = Rp_2$ yields an essential matrix $\tilde{R}E$ for the correspondences $\{(p_1, q_2)\}$.

The essential matrix has an SVD of the form $E = USV^T$, where U and V are orthogonal and $S = \mathrm{diag}(1, 1, 0)$. Since E is only determined up to scale, we can choose U and V as rotation matrices. It is then possible to decompose U and V into Euler rotations about the z- and y-axes: $E = R_{z,2}R_{y,2}R_{z',2}SR_{z',1}^T R_{y,1}^T R_{z,1}^T$ where we can merge $R_{z',2}$ and $R_{z',1}$ as they commute with S. We obtain $E = R_{z,2}R_{y,2}R_{z'}SR_{y,1}^T R_{z,1}^T$ and we have one degree of freedom for each R, thus five in total, as expected.

The equivariance properties of E imply that $R_{z,1}$ is equivariant to rotations in p_1 and $R_{z,2}$ is equivariant to rotations in p_2, both while being invariant to rotations of the other cloud. The other matrices are invariant to rotations in both clouds. We design the network to output five complex numbers on the unit circle $\mathbb{S}$, where two of them lie in $\mathcal{R}_2(m)$, and three are invariant to rotations in either cloud.

The model: ZZ-net. We use a back-bone architecture $\mathcal{B}$ with three ZZ-units, the first two units having 2 early, 2 late and 2 vector layers and the last unit having 1 early, 1 late and 1 vector layer. We also add skip-connections between the units in the back-bone for ease of training. This back-bone outputs 8 channels of point clouds which are fed into two further units. One is a ZZ-unit $\mathcal{E}$ which is responsible for predicting the *equivariant* $R_{z,2}$ and $R_{z,1}$. The second is a PointNet $\mathcal{I}$ that takes as input the α^+-values of the last layer of the back-bone (which are rotation invariant) to predict the *invariant* $R_{y,2}$, $R_{y,1}$ and $R_{z'}$.

To account for the symmetry of changing order of the clouds, we approximate $R_{z,1}$ with $\mathcal{E}(\mathcal{B}(Z, X))$, and $R_{z,2}$ with $\mathcal{E}(\mathcal{B}(X, Z))$. In turn, $\mathcal{I}(\mathcal{B}(Z, X))$ yields two rotations: $R_{y,1}$ and $R_{z',1}$, while $\mathcal{I}(\mathcal{B}(X, Z))$ yields $R_{y,2}$ and $R_{z',2}$. $R_{z',1}$ and $R_{z',2}$ are combined to form $R_{z'} = R_{z',2}R_{z',1}^T$. In total, the architecture thus outputs five rotations. It has around 55k parameters.

Similar to OANet [53], we use a geometric loss based on virtual matches generated from the ground truth essential matrix. For further information on the model and training setup, see the supplementary B.2.

Data. We use the subset of the YFCC100M data [38] corresponding to the sequence 'Reichstag' compiled by [17]. Two example images can be seen in Figure 5. The image sequence is processed to obtain SIFT-matches [28] between image pairs using code supplied by the authors of CNe [51]. Some image pairs are discarded due to visibility issues and for each remaining image pair 2000 correspondences are found, many of which might be incorrect matches. The obtained dataset is quite small – the training

Figure 5. Two images from the 'Reichstag' data.

set consists of 3302, the validation set of 56 and the test set of 52 point cloud pairs[6]. Therefore our experiments should be viewed as a limited data case study.

Evaluation metric. From the essential matrix we can recover the rotation between the two views and the translation between the views up to scale. We evaluate the estimated essential matrix in terms of the mAP score proposed by [51], which is a measure of error in angle of the estimated translation and rotation axes.

Comparisons. We compare against CNe [51], OANet [53] and ACNe [36]. These methods build on the idea of learning inlier weights for the correspondences and using a weighted formulation of the 8 point method [27] as a final layer in the network. They are all very good at handling outliers, as they are explicitly trained on classifying each correspondence as an inlier or outlier as well as outputting a reasonable essential matrix. In contrast, our network is only trained to output a reasonable essential matrix but does it in a way that is resilient to rotations of the data, which is not part of the other frameworks. We do not compare against T-net [55] as they have not published their code at the time of writing.

We retrain the implementation of the authors of CNe, OANet and ACNe on the 'Reichstag' dataset. For the sake of fairness, we do not use RANSAC at test time. Note that therefore our reported numbers for CNe are below what they report in their paper. CNe has 394k parameters, ACNe 400k parameters and OANet 2347k parameters.

Rotated test data. To demonstrate the resilience of our method to rotation perturbations of the data, we evaluate both on the original test data as well as versions of the test data where the p_1 points are rotated a random amount (and the ground truth essential matrix is altered correspondingly, as described earlier). We sample rotations for each test example uniformly in the interval $(-a, a)$ and consider three different values for the maximum rotation angle: $a = 30°, 60°, 180°$. All methods are evaluated on the same rotated versions of the test set for consistency.

Results. We present results in Table 2 for mAP at 20°. The results for our method are averaged over two training runs. The maximum difference in mAP scores between the two runs was 0.01. mAP scores at 10° and 30° are presented in the supplementary B.2 and they tell a similar story.

[6]In fact half of the 3302 (resp. 56, 52) pairs correspond to the other half but with the two images in the pair swapped.

Max. test rot. $a =$	0°	30°	60°	180°
ZZ-net (Ours)	0.26	**0.26**	**0.26**	**0.26**
ACNe	**0.67**	0.25	0.15	0.038
CNe	0.43	0.14	0.12	0.0048
OANet	0.42	0.24	0.077	0.0048

Table 2. Results for essential matrix estimation. mAP at 20° error in the estimated translation and rotation vectors for different values of image plane rotations a at test time.

Discussion. Our method does not compete well on the base problem ($a = 0°$). This may in part be due to the order of magnitude fewer parameters of our network. Note that we had to limit the number of parameters due to the quadratic memory cost of the weight-units. We however demonstrate the resilience to rotation perturbations of ZZ-net. Already at modest rotations uniformly sampled from $-30°$ to $30°$ it is on par with the more mature competitors. At larger rotations ZZ-net is superior. It should however be noted that for this dataset, all images are oriented close to parallel with the ground. There is hence a clear bias in the training data, so that the comparison to the other models on artificially rotated test data is not completely fair.

We still believe that rotation equivariance can add robustness to methods attacking the essential matrix estimation problem and regard it as an interesting future research direction to try to merge our approach with the outlier robust methods, using for instance the weighted 8-point method. Furthermore, it would be interesting to develop methods which are equivariant only to small rotations – rotations larger than 60° will typically not be seen in practice. This would require leaving the mathematical framework of group theory, as such bounded rotations do not form a group.

6. Conclusions

We have presented a foundational framework for learning tasks based on a rotation equivariant and permutation invariant neural network architecture. A proof is given showing that this architecture is indeed universal. We have described several ways of modifying the architecture, in particular, how to extend it to pairs of point clouds as appearing in correspondence problems and how to perform efficient computations. As for limitations, the framework is only applicable in two dimensions. Our architecture further lacks locality and has a high memory requirement. To mitigate the latter issues are examples of interesting future work.

Acknowledgements

The authors acknowledge support from CHAIR, SSF, as well as WASP funded by the Knut and Alice Wallenberg Foundation. The computations were enabled by resources provided by SNIC at C3SE.

References

[1] Jimmy Aronsson. Homogeneous vector bundles and G-equivariant convolutional neural networks. *arXiv:2105.05400 [cs, math, stat]*, May 2021. 2

[2] BreakdownDiode. Big Dipper 20210116.jpg, used under Creative Commons Attribution-ShareAlike 4.0 International license // Stars in main constellation brightened. https://commons.wikimedia.org/wiki/File:Big_Dipper_20210116.jpg, 2021. 1

[3] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *arXiv:2104.13478 [cs, stat]*, May 2021. 2

[4] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Int. Conf. Machine Learning*, 2016. 2

[5] Taco S. Cohen. *Equivariant Convolutional Networks (PhD Thesis)*. PhD thesis, University of Amsterdam, June 2021. 2

[6] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. 17

[7] 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 (ICCV)*, pages 12200–12209, October 2021. 2

[8] Nadav Dym and Haggai Maron. On the universality of rotation equivariant point cloud networks. In *ICLR*, 2021. 2, 13

[9] JA Eidswick. A proof of Newton's power sum formulas. *The American Mathematical Monthly*, 75(4):396–397, 1968. 18

[10] William Falcon and The PyTorch Lightning team. PyTorch Lightning, Mar. 2019. 24

[11] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A Practical Method for Constructing Equivariant Multilayer Perceptrons for Arbitrary Matrix Groups. *arXiv:2104.09459 [cs, math, stat]*, Apr. 2021. 15

[12] Marc Finzi, Max Welling, and Andrew Gordon Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3318–3328. PMLR, 18–24 Jul 2021. 2

[13] Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D roto-translation equivariant attention networks. In *NeurIPS*, 2020. 2, 3

[14] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics*, 36:193—202, 1980. 2

[15] Jan E. Gerken, Jimmy Aronsson, Oscar Carlsson, Hampus Linander, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson. Geometric Deep Learning and Equivariant Neural Networks. *arXiv:2105.13926 [hep-th]*, May 2021. 2

[16] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK ; New York, 2nd ed edition, 2003. 2, 7

[17] Jared Heinly, Johannes L. Schonberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the world* in six days. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3287–3295, Boston, MA, USA, June 2015. IEEE. 7

[18] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *UIST '11 Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, October 2011. 1

[19] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 3

[20] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. *NeurIPS*, 32:7092–7101, 2019. 2, 11

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 26

[22] Yvette Kosmann-Schwarzbach. *Groups and Symmetries*. Springer New York, New York, NY, 2010. 1

[23] Leon Lang and Maurice Weiler. A wigner-eckart theorem for group equivariant convolution kernels. In *International Conference on Learning Representations*, 2021. 2

[24] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. 2

[25] 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 *Proceedings of the 36th International Conference on Machine Learning*, pages 3744–3753, 2019. 3

[26] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018. 24

[27] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, Sept. 1981. 8

[28] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004. 7

[29] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *ICLR*, 2018. 2, 4, 14, 15, 17, 27

[30] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *Int. Conf. Machine Learning*, pages 4363–4371, 2019. 2, 11, 13

[31] Pavlo Melnyk, Michael Felsberg, and Mårten Wadenbäck. Embed me if you can: A geometric perceptron. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1276–1284, October 2021. 2

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 24

[33] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, pages 652–660, 2017. 2, 3

[34] Joshua Rapp, Julian Tachella, Yoann Altmann, Stephen McLaughlin, and Vivek K Goyal. Advances in single-photon lidar for autonomous vehicles: Working principles, challenges, and recent advances. *IEEE Signal Processing Magazine*, 37(4):62–71, 2020. 1

[35] Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 1953. 4, 17

[36] Weiwei Sun, Wei Jiang, Eduard Trulls, Andrea Tagliasacchi, and Kwang Moo Yi. ACNe: Attentive context normalization for robust permutation-equivariant learning. In *CVPR*, 2020. 2, 3, 6, 8, 25

[37] 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:1802.08219 [cs]*, May 2018. 2

[38] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *Commun. ACM*, 59(2):64–73, Jan. 2016. 7

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 3

[40] Soledad Villar, David W. Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. *Preprint. arXiv: 2106.06610*, 2021. 2

[41] Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. On the limitations of representing functions on sets. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6487–6494. PMLR, 09–15 Jun 2019. 2

[42] Maurice Weiler and Gabriele Cesa. General e(2)-equivariant steerable cnns. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2

[43] Maurice Weiler, Patrick Forré, Erik Verlinde, and Max Welling. Coordinate Independent Convolutional Networks – Isometry and Gauge Equivariant Convolutions on Riemannian Manifolds. *arXiv:2106.06020 [cs, stat]*, June 2021. 2

[44] Maurice Weiler, Fred A. Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant CNNs. In *CVPR*, 2018. 2

[45] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *CVPR*, 2017. 2

[46] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional ShapeContextNet for Point Cloud Recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, June 2018. 3

[47] Jianyun Xu, Xin Tang, Yushi Zhu, Jie Sun, and Shiliang Pu. Sgmnet: Learning rotation-invariant point cloud representations via sorted gram matrix. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10468–10477, October 2021. 2, 3

[48] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2048–2057. PMLR, June 2015. 3

[49] Weichi Yao, Kate Storey-Fisher, David W. Hogg, and Soledad Villar. A simple equivariant machine learning method for dynamics based on scalars. *arXiv:2110.03761 [cs]*, Oct. 2021. 2

[50] Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, pages 1–68, 2021. 2, 13

[51] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 2, 6, 7, 8, 26, 27

[52] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *NeurIPS*, volume 30, 2017. 2, 3

[53] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. *International Conference on Computer Vision (ICCV)*, 2019. 2, 7, 8

[54] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *ICCV*, pages 16259–16268, October 2021. 2

[55] Zhen Zhong, Guobao Xiao, Linxin Zheng, Yan Lu, and Jiayi Ma. T-Net: Effective permutation-equivariant network for two-view correspondence learning. In *ICCV*, 2021. 2, 8