On Learning Sets of Symmetric Elements

Haggai Maron [1] **Or Litany** [2] **Gal Chechik** [3 1] **Ethan Fetaya** [3]

Abstract

Learning from unordered sets is a fundamental learning setup, recently attracting increasing attention. Research in this area has focused on the case where elements of the set are represented by feature vectors, and far less emphasis has been given to the common case where set elements themselves adhere to their own symmetries. That case is relevant to numerous applications, from deblurring image bursts to multi-view 3D shape recognition and reconstruction. In this paper, we present a principled approach to learning sets of general symmetric elements. We first characterize the space of linear layers that are equivariant both to element reordering and to the inherent symmetries of elements, like translation in the case of images. We further show that networks that are composed of these layers, called *Deep Sets for Symmetric elements* layers (DSS), are universal approximators of both invariant and equivariant functions. DSS layers are also straightforward to implement. Finally, we show that they improve over existing set-learning architectures in a series of experiments with images, graphs and point-clouds.

1. Introduction

Learning with data that consists of unordered sets of elements is an important problem with numerous applications, from classification and segmentation of 3D data (Zaheer et al., 2017; Qi et al., 2017; Su et al., 2015; Kalogerakis et al., 2017) to image deblurring (Aittala & Durand, 2018). In this setting, each data point consists of a set of elements, and the task is independent of element order. This independence induces a symmetry structure, which can be used to design deep models with improved efficiency and generalization. Indeed, models that respect set symmetries, e.g. (Zaheer et al., 2017; Qi et al., 2017), have become the leading approach for solving such tasks. However, in many cases, the elements of the set themselves adhere to certain symmetries, as happens when learning with sets of images, sets of point-clouds and sets of graphs. It is still unknown what is the best way to utilize these additional symmetries.

A common approach to handle per-element symmetries, is based on processing elements individually. First, one processes each set-element independently into a feature vector using a Siamese architecture (Bromley et al., 1994), and only then fuses information across all feature vectors. When following this process, the interaction between the elements of the set only occurs after each element has already been processed, possibly omitting low-level details. Indeed, it has been recently shown that for learning sets of images (Aittala & Durand, 2018; Sridhar et al., 2019; Liu et al., 2019), significant gain can be achieved with intermediate information-sharing layers.

In this paper, we present a principled approach to learning sets of symmetric elements. First, we describe the symmetry group of these sets, and then fully characterize the space of linear layers that are equivariant to this group. Notably, this characterization implies that information between set elements should be shared in all layers. For example, Figure 1 illustrates a DSS layer for sets of images. DSS layers provide a unified framework that generalizes several previously-described architectures for a variety of data types. In particular, it directly generalizes DeepSets (Zaheer et al., 2017). Moreover, other recent works can also be viewed as special cases of our approach (Hartford et al., 2018; Aittala & Durand, 2018; Sridhar et al., 2019).

A potential concern with equivariant architectures is that restricting layers to be equivariant to some group of symmetries may reduce the expressive power of the model (Maron et al., 2019c; Morris et al., 2018; Xu et al., 2019). We eliminate this potential limitation by proving two universal-approximation theorems for invariant and equivariant DSS networks. Simply put, these theorems state that if invariant (equivariant) networks for the elements of interest are universal, then the corresponding invariant (equivariant) DSS networks on sets of such elements are also universal.

To summarize, this paper has three main contributions: (1) We characterize the space of linear equivariant layers for sets

[1]NVIDIA Research [2]Stanford University [3]Bar Ilan University. Correspondence to: Haggai Maron <hmaron@nvidia.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

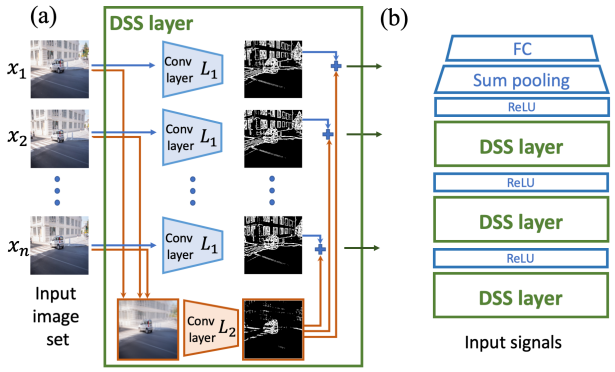


Figure 1. (a) A DSS layer for a set of images is composed of Siamese layer (blue) and an aggregation module (orange). The Siamese part is a convolutional layer (L_1) that is applied to each element independently. In the aggregation module, the *sum* of all images is processed by a different convolutional layer (L_2) and is added to the output of the Siamese part. (b) An example of a simple DSS-based invariant network.

of elements with symmetries. (2) We prove two universal approximation theorems for networks that are composed of DSS layers. (3) We demonstrate the empirical benefits of the DSS layers in a series of tasks, from classification through matching to selection, applied to diverse data from images to graphs and 3D point-clouds. These experiments show consistent improvement over previous approaches.

2. Previous work

Learning with sets. Several studies designed network architectures for set-structured input. Vinyals et al. (2015) suggested to extend the sequence-to-sequence framework of Sutskever et al. (2014) to handle sets. The prominent works of Ravanbakhsh et al. (2016); Edwards & Storkey (2016); Zaheer et al. (2017); Qi et al. (2017) proposed to use standard feed-forward neural networks whose layers are constrained to be equivariant to permutations. These models, when combined with a set-pooling layer, were also shown to be universal approximators of continuous permutation-invariant functions. Wagstaff et al. (2019) provided a theoretical study on the limitations of representing functions on sets with such networks. In another related work, Murphy et al. (2018) suggested to model permutation-invariant functions as an average of permutation-sensitive functions.

The specific case of learning sets of images was explored in several studies. Su et al. (2015); Kalogerakis et al. (2017) targeted classification and segmentation of 3D models by processing images rendered from several view points. These methods use a Siamese convolutional neural network to process the images, followed by view-pooling layer. Sridhar et al. (2019) tackled 3D shape reconstruction from multiple view points and suggest using several equivariant mean-removal layers in which the mean of all images is subtracted from each image in the set. Aittala & Durand (2018) targeted image burst deblurring and denoising, and suggested to use set-pooling layers after convolutional blocks in which for each pixel, the maximum over all images is concatenated to all images. Liu et al. (2019) proposed to use an attention-based information sharing block for face recognition tasks. In Gordon et al. (2020) the authors modify neural processes by adding a translation equivariance assumption, treating the inputs as a set of translation equivariant objects.

Equivariance in deep learning. The prototypical example for equivariance in learning is probably visual object recognition, where the prevailing Convolutional Neural Networks (CNNs) are constructed from convolution layers which are equivariant to image translations. In the past few years, researchers have used invariance and equivariance considerations to devise deep learning architectures for other types of data. In addition to set-structured data discussed above, researchers suggested equivariant models for interaction between sets (Hartford et al., 2018), graphs (Kondor et al., 2018; Maron et al., 2019b;a; Chen et al., 2019; Albooyeh et al., 2019) and relational databases (Graham & Ravanbakhsh, 2019). Another successful line of work took into account other image symmetries such as reflections and rotations (Dieleman et al., 2016; Cohen & Welling, 2016a;b; Worrall et al., 2017), spherical symmetries (Cohen et al., 2018; 2019b; Esteves et al., 2017), or 3D symmetries (Weiler et al., 2018; Winkels & Cohen, 2018; Worrall & Brostow, 2018; Kondor, 2018; Thomas et al., 2018; Weiler et al., 2018). From a theoretical point of view, several papers studied the properties of equivariant layers (Ravanbakhsh et al., 2017; Kondor & Trivedi, 2018; Cohen et al., 2019a) and characterized the expressive power of models that use such layers (Yarotsky, 2018; Maron et al., 2019c; Keriven & Peyré, 2019; Maehara & NT, 2019; Segol & Lipman, 2019).

3. Preliminaries

3.1. Notation and basic definitions

Let $x \in \mathbb{R}^\ell$ represent an input that adheres to a group of symmetries $G \leq S_\ell$, the symmetric group on ℓ elements. G captures those transformations that our task-of-interest is invariant (or equivariant) to. The action of G on $\mathbb{R}^\ell$ is defined by $(g \cdot x)_i = x_{g^{-1}(i)}$. For example, when inputs are images of size $h \times w$, we have $\ell = hw$ and G can be a group that applies cyclic translations, or left-right reflections to an image. A function is called G-equivariant if $f(g \cdot x) = g \cdot f(x)$ for all $g \in G$. Similarly, a function f is called G-invariant if $f(g \cdot x) = f(x)$ for all $g \in G$.

3.2. G-invariant networks

G-equivariant networks are a popular way to model G-equivariant functions. These networks are composed of several linear G-equivariant layers, interleaved with activation functions like ReLU, and have the following form:

$$f = L_k \circ \sigma \circ L_{k-1} \cdots \circ \sigma \circ L_1, \tag{1}$$

Where $L_i : \mathbb{R}^{\ell \times d_i} \to \mathbb{R}^{\ell \times d_{i+1}}$ are linear G-equivariant layers, d_i are the feature dimensions and σ is a point-wise activation function. It is straightforward to show that this architecture results in a G-equivariant function. G-invariant networks are defined by adding an invariant layer on top of a G-equivariant function followed by a multilayer Perceptron (MLP), and have the form:

$$g = m \circ \sigma \circ h \circ \sigma \circ f\,, \tag{2}$$

where $h : \mathbb{R}^{\ell \times d_{k+1}} \to \mathbb{R}^{d_{k+2}}$ is a linear G-invariant layer and $m : \mathbb{R}^{d_{k+2}} \to \mathbb{R}^{d_{k+3}}$ is an MLP. It can be readily shown that this architecture results in a G-invariant function.

3.3. Characterizing equivariant layers

The main building block of G-invariant/equivariant networks are linear G-invariant/equivariant layers. To implement these networks, one has to characterize the space of linear G-invariant/equivariant layers, namely, L_i, h in Equations (1-2). For example, it is well known that for images with the group G of circular 2D translations, the space of linear G-equivariant layers is simply the space of all 2D convolutions operators (Puschel & Moura, 2008). Unfortunately, such elegant characterizations are not available for most permutation groups.

Characterizing linear G-equivariant layers can be reduced to the task of solving a set of linear equations in the following way: We are looking for a linear operator $L : \mathbb{R}^\ell \to \mathbb{R}^\ell$ that commutes with all the elements in G, namely:

$$L(g \cdot x) = g \cdot L(x),\ x \in \mathbb{R}^\ell,\ g \in G \tag{3}$$

Note that L can be realized as a $\ell \times \ell$ matrix (which will be denoted in the same way), and as in Maron et al. (2019b), Equation 3 is equivalent to the following linear system:

$$g \cdot L = L,\ g \in G, \tag{4}$$

where g acts on both dimensions of L. The solution space of Equation 4 characterizes the space of all G-equivariant linear layers, or equivalently, defines a parameter sharing scheme on the layer parameters for the group G (Wood & Shawe-Taylor, 1996; Ravanbakhsh et al., 2017). We will denote the dimension of this space as $E(G)$. We note that in many important cases (e.g., (Zaheer et al., 2017; Hartford et al., 2018; Maron et al., 2019b; Albooyeh et al., 2019)) $|G|$ is exponential in ℓ so it is not possible to solve the linear system naively, and one has to resort to other strategies.

3.4. Deep Sets

Since the current paper generalizes *DeepSets* (Zaheer et al., 2017), we summarize their main results for completeness. Let $\{x_1, \dots x_n\} \subset \mathbb{R}$ be a set, which we represent in arbitrary order as a vector $x \in \mathbb{R}^n$. DeepSets characterized all S_n-equivariant layers, namely, all matrices $L \in \mathbb{R}^{n \times n}$ such that $g \cdot L(x) = L(g \cdot x)$ for any permutation $g \in S_n$ and have shown that these operators have the following structure: $L = \lambda I_n + \beta \mathbf{1}\mathbf{1}^T$. When considering sets with higher dimensional features, i.e., $x_i \in \mathbb{R}^d$ and $X \in \mathbb{R}^{n \times d}$, this characterization takes the form:

$$L(X)_i = L_1(x_i) + L_2\left(\sum_{j \neq i}^{n} x_j\right), \tag{5}$$

where $L_1, L_2 : \mathbb{R}^d \to \mathbb{R}^d$ are general linear functions and the subscript represents the i-th row of the output. The paper then suggests to concatenate several such layers, yielding a deep equivariant model (or an invariant model if a set pooling layer is added on top). Zaheer et al. (2017); Qi et al. (2017) established the universality of invariant networks that are composed of DeepSets Layers and Segol & Lipman (2019) extended this result to the equivariant case.

4. DSS layers

Our main goal is to design deep models for sets of elements with non-trivial per-element symmetries. In this section, we first formulate the symmetry group G of such sets. The deep models we advocate are composed of linear G-equivariant layers (DSS layers), therefore, our next step is to find a simple and practical characterization of the space of these layers.

4.1. Sets with symmetric elements

Let $\{x_1, \dots x_n\} \subset \mathbb{R}^d$ be a set of elements with symmetry group $H \leq S_d$. We wish to characterize the space of linear maps $L : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ that are equivariant to both the natural symmetries of the elements, represented by the elements of the group H, as well as to the order of the n elements, represented by S_n.

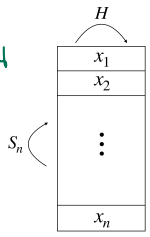


Figure 2. The input to a DSS layer is an $n \times d$ matrix, in which each row holds a d-dimensional element. $G = S_n \times H$ acts on it by applying a permutation to the columns and an element $h \in H$ to the rows.

In our setup, H operates on all elements x_i in the same way. More formally, the symmetry group is defined by $G = S_n \times H$, where S_n is the symmetric group on n elements. This group operates on $X \in \mathbb{R}^{n \times d}$

by applying the permutation $q \in S_n$ to the first dimension and the same element $h \in H$ to the second dimension, namely $((q,h)\cdot X)_{ij} = X_{q^{-1}(i)h^{-1}(j)}$. Notably, this setup generalizes several popular learning setups: (1) DeepSets, where $H = \{I_d\}$ is the trivial group. (2) Tabular data (Hartford et al., 2018), where $H = S_d$. (3) Sets of images, where H is the group of circular translations (Aittala & Durand, 2018). Figure 2 illustrates this setup.

One can also consider another setup, where the members of H that are applied to each element of the set may differ. Section C of the supplementary material formulates this setup and characterizes the corresponding equivariant layers in the common case where H acts transitively on $\{1,\dots,d\}$. While this setup can be used to model several interesting learning scenarios, it turns out that the corresponding equivariant networks are practically reduced to Siamese networks that were suggested in previous works.

4.2. Characterization of equivariant layers

This subsection provides a practical characterization of linear G-equivariant layers for $G = S_n \times H$. Our result generalizes DeepSets (equation 5) whose layers are tailored for $H = \{I_d\}$, by replacing the linear operators L_1, L_2 with linear H-equivariant operators. This result is summarized in the following theorem:

Theorem 1. *Any linear $G-$equivariant layer $L : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ is of the form*

$$L(X)_i = L_1^H(x_i) + L_2^H\left(\sum_{j\neq i}^{n} x_j\right),$$

where L_1^H, L_2^H are linear H-equivariant functions

Note that this is equivalent to the following formulations $L(X)_i = L_1^H(x_i) + L_2^H(\sum_{j=1}^n x_j) = L_1^H(x_i) + \sum_{j=1}^n L_2^H(x_j)$ due to linearity, and we will use them interchangeably throughout the paper. Figure 1 illustrates Theorem 1 for sets of images. In this case, applying a DSS layer amounts to: (i) Applying the same convolutional layer L_1 to all images in the set (blue); (ii) Applying another convolutional layer L_2 to the sum of all images (orange); and (iii) summing the outputs of these two layers. We discuss this theorem in the context of other widely-used data types such as point-clouds and graphs in section F of the Supplementary material.

We begin the proof by stating a useful lemma, that provides a formula for the dimension of the space of linear G-equivariant maps:

Lemma 1. *Let $G \leq S_\ell$, then the dimension of the space of G-equivariant linear functions $L : \mathbb{R}^\ell \to \mathbb{R}^\ell$ is*

$$E(G) = \frac{1}{|G|}\sum_{g\in G} tr(P(g))^2,$$

where $P(g)$ is the permutation matrix that corresponds to the permutation g.

The proof is given in the supplementary material. Given this lemma we can now prove Theorem 1:

Proof of Theorem 1. We wish to prove that all linear G-equivariant layers $L : \mathbb{R}^{n\times k} \to \mathbb{R}^{n\times k}$ are of the form $L(X)_i = L_1^H(x_i) + L_2^H(\sum_{j\neq i}^n x_j)$. Clearly, layers of this form are linear and equivariant. Moreover, the dimension of the space of these operators is exactly $2E(H)$ since we need to account for two linearly independent H-equivariant operators. The linear independence follows from the fact that their support in the matrix representation of L is disjoint. On the other hand, using Lemma 1 we have:

$$\begin{aligned}
E(G) &= \frac{1}{|G|}\sum_{g\in G}\mathrm{tr}(P(g))^2 = \\
&= \frac{1}{|H|}\frac{1}{n!}\sum_{q\in S_n}\sum_{h\in H}\mathrm{tr}(P(q)\otimes P(h))^2 \\
&= \frac{1}{|H|}\frac{1}{n!}\sum_{q\in S_n}\sum_{h\in H}\mathrm{tr}(P(q))^2\mathrm{tr}(P(h))^2 \\
&= \left(\frac{1}{|H|}\sum_{h\in H}\mathrm{tr}(P(h))^2\right)\cdot\left(\frac{1}{n!}\sum_{q\in S_n}\mathrm{tr}(P(q))^2\right) \\
&= E(H)E(S_n) = 2E(H).
\end{aligned}$$

Here we used the fact that the trace is multiplicative with respect to the Kronecker product as well as the fact that $E(S_n) = 2$ (see (Zaheer et al., 2017) or Appendix 2 in (Maron et al., 2019b) for a generalization of this result).

To conclude, we have a linear subspace $\left\{L \mid L(X)_i = L_1^H(x_i) + L_2^H(\sum_{j\neq i}^n x_j)\right\}$, which is a subspace of the space of all linear G-equivariant operators, but has the same dimension, which implies that both spaces are equal. □

Relation to (Aittala & Durand, 2018; Sridhar et al., 2019). In the specific case of a set of images and translation equivariance, L_i^H are convolutions. In this setting, (Aittala & Durand, 2018; Sridhar et al., 2019) have previously proposed using set-aggregation layers after convolutional blocks. The main differences between these studies and the current paper are: (1) Our work applies to all types of symmetric elements and not just images; (2) We derive these layers from first principles; (3) We provide a theoretical analysis (Section 5); (4) We apply an aggregation step at each layer instead of only after convolutional blocks.

Generalizations. Section A of the supplementary material generalizes Theorem 1 to equivariant linear layers

with multiple features. It also generalizes to several additional types of equivariant layers: $L : \mathbb{R}^{n\times d} \to \mathbb{R}$, $L : \mathbb{R}^{n\times d} \to \mathbb{R}^n$ and $L : \mathbb{R}^{n\times d} \to \mathbb{R}^d$. In addition, see Section B of the supplementary material for further discussion and characterization of the space of equivariant maps for a product of arbitrary permutation groups.

5. A universal approximation theorem

When restricting a network to be invariant (equivariant) to some group action, one may worry that these restrictions could reduce the network expressive power (see Maron et al. (2019c) or Xu et al. (2019) for concrete examples). We now show that networks that are constructed from DSS layers do not suffer from loss of expressivity. Specifically, we show that for any group H that induces a *universal* H-invariant (equivariant) network, its corresponding G-invariant (equivariant) network is universal as well.

We first state a lemma, which we later use for proving our universal-approximation theorems. The lemma shows that one can uniquely encode orbits of a group H in an invariant way by using a polynomial function. The full proof is given in Section D of the supplementary material.

Lemma 2. *Let $H \leq S_d$ then there exists a polynomial function $u : \mathbb{R}^d \to \mathbb{R}^l$, for some $l \in \mathbb{N}$, for which $u(x) = u(y)$ if and only if $x = h \cdot y$ for some $h \in H$.*

Proof idea. This lemma is a generalization of Proposition 1 in (Maron et al., 2019a) and we follow their proof. The main idea is that for any such group H there exists a finite set of invariant polynomials whose values on $\mathbb{R}^d$ uniquely define each orbit of H in $\mathbb{R}^d$. □

5.1. Invariant functions

We are now ready to state and prove our first universal approximation theorem. As before, the full proof can be found in the supplementary material (Section D).

Theorem 2. *Let $K \subset \mathbb{R}^{n\times d}$ be a compact domain such that $K = \cup_{g\in G} gK$. G-invariant networks are universal approximators (in $\|\cdot\|_\infty$ sense) of continuous G-invariant functions on K if and only if H-invariant networks are universal*[1].

Proof idea. The "only if" part is straightforward. For the "if" part, let $f : K \to \mathbb{R}$ be a continuous G-invariant function we wish to approximate. The idea of the proof is as follows: (1) we encode each element x_i with a unique H-invariant polynomial descriptor $u_H(x_i) \in \mathbb{R}^{l_H}$ (2) we encode the resulting set of descriptors with a unique S_n-invariant polynomial set descriptor $u_{S_n}\left(\{u_H(x_i)\}_{i\in[n]}\right) \in \mathbb{R}^{l_{S_n}}$ (3) we map the unique set descriptor $u_{S_n}\left(\{u(x_i)\}_{i\in[n]}\right)$ to the appropriate value defined by f (4) we use the classic universal approximation theorem (Cybenko, 1989; Hornik et al., 1989) and our assumption on the universality of H-invariant networks to conclude that there exists a G-invariant network that can approximate each one of the previous stages to arbitrary precision on K. □

Siamese networks. The proof of Theorem 1 implies that a simple Siamese architecture that applies an H-invariant network to each element in the set followed by a sum aggregation and finally an MLP is also universal. In section 6, we compare this architecture to our DSS networks and show that DSS-based architectures perform better in practice.

Relation to (Maron et al., 2019c). The authors proved that for any permutation group G, G-invariant networks have a universal approximation property, if the networks are allowed to use high-order tensors as intermediate representations (i.e., $X \in \mathbb{R}^{d^l}$ for $2 \leq l \leq n^2$), which are computationally prohibitive. We strengthen this result by proving that if first-order[2] H-invariant networks are universal, so are first-order G-invariant networks.

5.2. Equivariant functions

Three possible types of equivariant functions can be considered. First, functions of the form $f : \mathbb{R}^{n\times d} \to \mathbb{R}^n$. For example, such a function can model a selection task in which we are given a set $\{x_1, \ldots, x_n\}$ and we wish to select a specific element from that set. Second, functions of the form $f : \mathbb{R}^{n\times d} \to \mathbb{R}^d$. An example for this type of functions would be an image-deblurring task in which we are given several noisy measurements of the same scene and we wish to generate a single high quality image (e.g., (Aittala & Durand, 2018)). Finally, functions of the form $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$. This type of functions can be used to model tasks such as image co-segmentation where the input consists of several images and the task is to predict a joint segmentation map.

In this subsection we will prove a universality result for the third type of G-equivariant functions that were mentioned above, namely $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$. We note that the equivariance of the first and second types can be easily deduced from this case. One can transform, for example, an $\mathbb{R}^{n\times d} \to \mathbb{R}^d$ G-equivariant function into a $\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ function by repeating the $\mathbb{R}^d$ vector n times and use our general approximation theorem on this function. We can get back a $\mathbb{R}^{n\times d} \to \mathbb{R}^d$ function by averaging over the first dimension.

Theorem 3. *Let $K \subset \mathbb{R}^{n\times d}$ be a compact domain such that $K = \cup_{g\in G} gK$. G-equivariant networks are universal ap-*

[1] We assume that there is a universal approximation theorem for the activation functions, e.g., ReLU.

[2] First-order networks use only first-order tensors.

proximators (in $\|\cdot\|_\infty$ sense) of continuous $\mathbb{R}^{n\times d}\to\mathbb{R}^{n\times d}$ G-equivariant functions on K if and only if H-equivariant networks are universal.

Proof idea. The proof follows a similar line to the universality proof in (Segol & Lipman, 2019): First, we use the fact that equivariant polynomials are dense in the space of continuous equivariant functions. This enables us to assume that the function we wish to approximate is a G-equivariant polynomial. Next we show that for every output element, the mapping $\mathbb{R}^{n\times d}\to\mathbb{R}^d$ can be written as a sum of H-equivariant base polynomials with invariant coefficients. The base polynomials can be approximated by our assumption on H and the invariant mappings can be approximated by leveraging a slight modification of theorem 2. Finally we show how we can combine all the parts and approximate the full function with a G-equivariant network. □

The full proof is given in Section D of the supplementary material. Similarly to the invariance case, using a Siamese network on each element separately followed by one DSS layer is sufficient for proving universality.

5.3. Examples

We can use Theorems (2-3) to show that DSS-based networks are universal in two important cases. For tabular data, which was considered by Hartford et al. (2018), the symmetries are $G = S_n \times S_d$. From the universality of S_n-invariant and equivariant networks (Zaheer et al., 2017; Segol & Lipman, 2019) we get that G-invariant (equivariant) networks are universal as well[3]. For sets of images, when H is the group of circular translations, it was shown in Yarotsky (2018) that H-invariant/equivariant networks are universal[4], which implies universality of our DSS models.

6. Experiments

In this section we investigate the effectiveness of DSS layers in practice, by comparing them to previously suggested architectures and different aggregation schemes. We use the experiments to answer two basic questions: **(1)** ***Early or late aggregation?*** Can *early aggregation* architectures like DSS and its variants improve learning compared to *Late aggregation* architectures, which fuse the set information at the end of the data processing pipeline? and **(2)** ***How to aggregate?*** What is the preferred early aggregation scheme?

[3] Hartford et al. (2018) also considered interactions between more than two sets with $G = S_n \times S_{d_1} \times \cdots \times S_{d_k}$. Our theorems can be extended to that case by induction on k.

[4] We note that this paper considers convolutional layers with full size kernels and no pooling layers

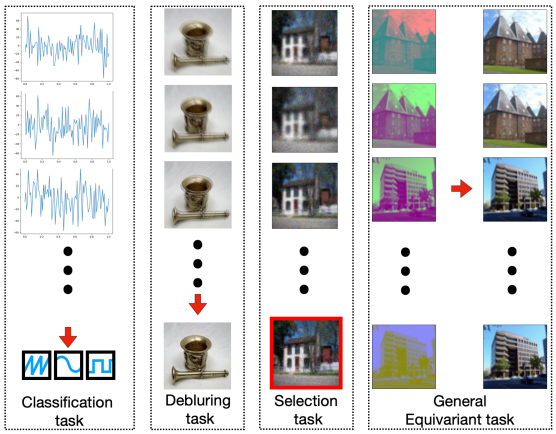


Figure 3. We consider all possible types of invariant and equivariant learning tasks in our settings: classification ($\mathbb{R}^{n\times d}\to\mathbb{R}$), selection ($\mathbb{R}^{n\times d}\to\mathbb{R}^n$), merging ($\mathbb{R}^{n\times d}\to\mathbb{R}^d$) and general equivariant tasks ($\mathbb{R}^{n\times d}\to\mathbb{R}^{n\times d}$).

Tasks. We evaluated DSS in a series of six experiments spanning a wide range of tasks: from classification ($\mathbb{R}^{n\times d}\to\mathbb{R}$), through selection ($\mathbb{R}^{n\times d}\to\mathbb{R}^n$) and burst image deblurring ($\mathbb{R}^{n\times d}\to\mathbb{R}^d$) to general equivariant tasks ($\mathbb{R}^{n\times d}\to\mathbb{R}^{n\times d}$). The experiments also demonstrate the applicability of DSS to a range of data types, including point-clouds, images and graphs. Figure 3 illustrates the various types of tasks evaluated. A detailed description of all tasks, architectures and datasets is given in the supplementary material (Section E).

Competing methods. We compare DSS to four other models: (1) MLP; (2) DeepSets (DS) (Zaheer et al., 2017); (3) Siamese network; (4) Siamese network followed by DeepSets (Siamese+DS).

We also compare several variants of our DSS layers: **(1) DSS(sum)**: our basic DSS layer from Theorem 1 **(2) DSS(max)**: DSS with max-aggregation instead of sum-aggregation **(3) DSS(Aittala)**: DSS with the aggregation proposed in (Aittala & Durand, 2018), namely, $L(x)_i \mapsto [L^H(x_i), \max_{j=1}^n L^H(x_j)]$ where [] denotes feature concatenation and L^H is a linear H-equivariant layer **(4) DSS(Sridhar)**: DSS layers with the aggregation proposed in (Sridhar et al., 2019) ,i.e., $L(x)_i \mapsto L^H(x_i) - \frac{1}{n}\sum_{j=1}^n L^H(x_j)$.

Evaluation protocol. For a fair comparison, for each particular task, all models have roughly the same number of parameters. In all experiments, we report the mean and standard deviation over 5 random initializations. Experiments were conducted using NVIDIA DGX with V100 GPUs.

Dataset	Data type	Late Aggregation	Early Aggregation				Random choice
		Siamese+DS	DSS (sum)	DSS (max)	DSS (Sridhar)	DSS (Aittala)	
UCF101	Images	36.41% ± 1.43	76.6% ± 1.51	76.39% ± 1.01	60.15% ± 0.76	**77.96%** ± 1.69	12.5%
Dynamic Faust	Point-clouds	22.26% ± 0.64	42.45% ± 1.32	28.71% ± 0.64	**54.26%** ± 1.66	26.43% ± 3.92	14.28%
Dynamic Faust	Graphs	26.53% ± 1.99	44.24% ± 1.28	30.54% ± 1.27	**53.16%** ± 1.47	26.66% ± 4.25	14.28%

Table 1. Frame selection tasks for images, point-clouds and graphs. Numbers represent average classification accuracy.

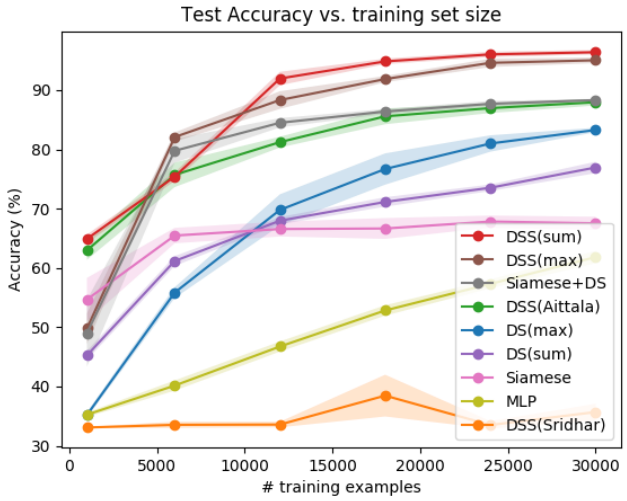


Figure 4. Comparison of set learning methods on the signal classification task. Shaded area represents standard deviation.

6.1. Classification with multiple measurements

To illustrate the benefits of DSS, we first evaluate it in a signal-classification task using a synthetic dataset that we generated. Each sample consists of a set of $n = 25$ noisy measurements of the same 1D periodic signal sampled at 100 time-steps (see Figure 3). The clean signals are sampled uniformly from three signal types - sine, saw-tooth and square waves - with varying amplitude, DC component, phase-shift and frequency. The task is to predict the signal type given the set of noisy measurements. Figure 4 depicts the classification accuracy as a function of varying training set sizes, showing that DSS(sum) outperforms all other methods. Notably, DSS(sum) layers achieve significantly higher accuracy then the DeepSets architecture which takes into account the set structure but not within-element symmetry. DSS(sum) also outperforms the the *Siamese* and *Siamese+DS* architectures, which do not employ early aggregation. *DSS(Sridhar)* fails, presumably because it employs a mean removal aggregation scheme which is not appropriate for this task (removes the signal and leaves the noise).

6.2. Selection tasks

We next test DSS layers on selection tasks. In these tasks, we are given a set and wish to choose one element of the set that obeys a predefined property. Formally, each task is modelled as a G-equivariant function $f : \mathbb{R}^{n\times d} \to \mathbb{R}^n$, where the output vector represents the probability of selecting each element. The architecture comprises of three convolutional blocks employing Siamese or DSS variants, followed by a DeepSets block. We note that the *Siamese+DS* model was suggested for similar selection tasks in (Zaheer et al., 2017).

Frame selection in images and shapes. The first selection task is to find a particular frame within an unordered set of frames extracted from a video/shape sequence. For videos, we used the UCF101 dataset (Soomro et al., 2012). Each set contains $n = 8$ frames that were generated by randomly drawing a video, a starting position and frame ordering. The task is to select the "first" frame, namely, the one that appeared earliest in the video. Table 1 details the accuracy of all compared methods in this task, showing that *DSS(sum)* and *DSS(Aittala)* outperform *Siamese+DS* and *DSS(Sridhar)* by a large margin.

In a second selection task, we demonstrate that DSS can handle multiple data types. Specifically, we showcase how DSS operates on point-clouds and graphs. Given a short sequences of 3D human shapes preforming various activities, the task is to identify which frame was the center frame in the original non-shuffled sequence. These human shapes are represented as point-clouds in the first experiment and as graphs (point-clouds + connectivity) in the second.

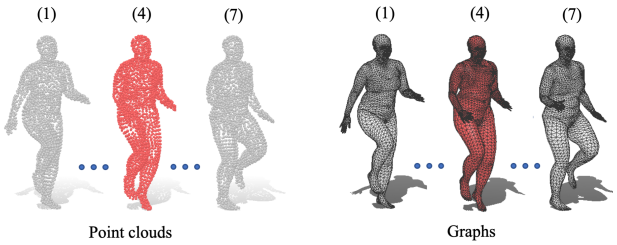


Figure 5. Shape-selection task on human shape sequences. Shapes are represented as graphs or as point-clouds. The task is to select the central frame (red). Numbers indicate frame order.

To generate the data, we cropped 7-frame-long sequences from the Dynamic Faust dataset (Bogo et al., 2017) in which the shapes are given as triangular meshes. To generate point-clouds, we simply use the mesh vertices. To generate graphs, we use the graph defined by the triangular mesh [5]. See Figure 5 for an illustration of this task.

[5] In (Bogo et al., 2017) the points of each mesh are ordered con-

Noise type and strength	Late Aggregation	Early Aggregation				Random choice
	Siamese+DS	DSS (sum)	DSS (max)	DSS (Sridahr)	DSS (Aittala)	
Gaussian $\sigma = 10$	77.2% $\pm$ 0.37	**78.48%** $\pm$ 0.48	77.99% $\pm$ 1.1	76.8% $\pm$ 0.25	78.34% $\pm$ 0.49	5%
Gaussian $\sigma = 30$	65.89% $\pm$ 0.66	**68.35%** $\pm$ 0.55	67.85% $\pm$ 0.40	61.52% $\pm$ 0.54	66.89% $\pm$ 0.58	5%
Gaussian $\sigma = 50$	59.24% $\pm$ 0.51	**62.6%** $\pm$ 0.45	61.59% $\pm$ 1.00	55.25% $\pm$ 0.40	62.02% $\pm$ 1.03	5%
Occlusion 10%	82.15% $\pm$ 0.45	83.13%$\pm$ 1.00	**83.27** $\pm$ 0.51	83.21% $\pm$ 0.338	83.19% $\pm$ 0.67	5%
Occlusion 30%	77.47% $\pm$ 0.37	78% $\pm$ 0.89	78.69% $\pm$ 0.32	**78.71%** $\pm$ 0.26	78.27% $\pm$ 0.67	5%
Occlusion 50%	76.2% $\pm$ 0.82	**77.29%** $\pm$ 0.40	76.64% $\pm$ 0.45	77.04% $\pm$ 0.75	77.03% $\pm$ 0.58	5%

Table 2. Highest-quality image selection. Values indicate the mean accuracy.

Task	Late Aggregation	Early Aggregation				TP
	Siamese+DS	DSS (sum)	DSS (max)	DSS (Sridahr)	DSS (Aittala)	
Color matching (places)	8.06 $\pm$ 0.06	1.78 $\pm$ 0.03	1.92 $\pm$ 0.07	1.97 $\pm$ 0.02	**1.67** $\pm$ 0.06	14.68
Color matching (CelebA)	6 $\pm$ 0.13	1.27 $\pm$ 0.07	1.34 $\pm$ 0.07	1.35 $\pm$ 0.03	**1.17** $\pm$ 0.04	18.72
Burst deblurring (Imagenet)	6.15 $\pm$ 0.05	6.11 $\pm$ 0.08	5.87 $\pm$ 0.05	21.01 $\pm$ 0.08	**5.7** $\pm$ 0.13	16.75

Table 3. Color-channel matching and burst deblurring tasks. Values indicate mean absolute error per pixel over the test set where the pixel values are in $[0, 255]$. TP stands for the trivial grey-scale predictor.

Results are summarized in Table 1, comparing DSS variants to a late-aggregation baseline (Siamese +DS) and to random choice. We further compared to a simple yet strong baseline. Using the mapping between points across shapes, we computed the mean of each point, and searched for the shape that was closest to that mean in L_1 sense. Frames in the sequence are $80msec$ apart, which limits the deviations around the mean, making it a strong baseline. Indeed, it achieved an accuracy of 34.47, which outperforms both late aggregation, DSS(max) and DSS(Aitalla). In contrast, sum-based early aggregation methods reach significantly higher accuracy. Interestingly, using a graph representation provided a small improvement over point-clouds for almost all methods .

Highest quality image selection. Given a set of $n = 20$ degraded images of the same scene, the task is to select the highest-quality image. We generate data for this task from the Places dataset (Zhou et al., 2017), by adding noise and Gaussian blur to each image. The target image is defined to be the image that is the most similar in L_1 norm sense to the original image (see Figure 3 for an illustration). Notably, DSS consistently improves over Siamese+DS with a margin of 1% to 3%. See Table 2.

6.3. Color-channel matching

To illustrate the limitation of late-aggregation, we designed a very simple image-to-image task that highlights why early aggregation can be critical: learning to combine color channels into full images. Here, each sample consists of six images, generated from two randomly selected color images, by separating each image into three color channels. In each mono-chromatic image two channels were set to zero, yielding a $d = 64 \times 64 \times 3$ image. The task is to predict the fully colored image (i.e., imputing the missing color channels) for each of the set element. This can be formulated as a $\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ G-equivariant task. See Figure 3 for an example.

We use a U-net architecture (Ronneberger et al., 2015), where convolutions and deconvolutions are replaced with Siamese layers or DSS variants. A DeepSets block is placed between the encoder and the decoder. Table 3 shows that layers with early aggregation significantly outperform DS+Siamese. For context, we add the error value of a trivial predictor which imputes the zeroed color channels by replicating the input color channel, resulting in a gray-scale image. This experiment was conducted on two datasets: *CelebA* (Liu et al., 2018), and *Places* (Zhou et al., 2017).

6.4. Burst image deblurring

Finally, we test DSS layers in a task of deblurring image bursts as in (Aittala & Durand, 2018). In this task, we are given a set of $n = 5$ blurred and noisy images of the same scene and aim to generate a single high quality image. This can be formulated as a $\mathbb{R}^{n\times d} \to \mathbb{R}^{d}$ G-equivariant task. See results in Table 3, where we also added the mean absolute error of a trivial predictor that outputs the median pixel of the images in the burst at each pixel. More details can be found in the supplementary material.

6.5. Summary of experiments

The above experiments demonstrate that applying early aggregation using DSS layers improves learning in various tasks and data types, compared with earlier architectures like *Siamese+DS*. More specifically, the basic DSS layer, *DSS(sum)*, performs well on all tasks, and *DSS(Aittala)* has also yielded strong results. *DSS(Sridhar)* performs well on some tasks but fails on others. See Section G of the supplementary materials for additional experiments on a multi-view reconstruction task.

sistently, providing point-to-point correspondence across frames. When this correspondence is not available, a shape matching algorithm like (Litany et al., 2017; Maron & Lipman, 2018) can be used as preprocessing.

Acknowledgments

This research was supported by an Israel science foundation grant 737/18. We thank Srinath Sridhar and Davis Rempe for useful discussions.

References

Aittala, M. and Durand, F. Burst image deblurring using permutation invariant convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 731–747, 2018.

Albooyeh, M., Bertolini, D., and Ravanbakhsh, S. Incidence networks for geometric deep learning. *arXiv preprint arXiv:1905.11460*, 2019.

Bogo, F., Romero, J., Pons-Moll, G., and Black, M. J. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pp. 737–744, 1994.

Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gnns. *arXiv preprint arXiv:1905.12560*, 2019.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999, 2016a.

Cohen, T. S. and Welling, M. Steerable CNNs. (1990):1–14, 2016b. URL http://arxiv.org/abs/1612.08498.

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

Cohen, T. S., Geiger, M., and Weiler, M. A general theory of equivariant cnns on homogeneous spaces. In *Advances in Neural Information Processing Systems*, pp. 9142–9153, 2019a.

Cohen, T. S., Weiler, M., Kicanaoglu, B., and Welling, M. Gauge equivariant convolutional networks and the icosahedral cnn. *arXiv preprint arXiv:1902.04615*, 2019b.

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

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dieleman, S., De Fauw, J., and Kavukcuoglu, K. Exploiting cyclic symmetry in convolutional neural networks. *arXiv preprint arXiv:1602.02660*, 2016.

Edwards, H. and Storkey, A. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. 3d object classification and retrieval with spherical cnns. *arXiv preprint arXiv:1711.06721*, 2017.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Fulton, W. and Harris, J. *Representation theory: a first course*, volume 129. Springer Science & Business Media, 2013.

Gordon, J., Bruinsma, W. P., Foong, A. Y. K., Requeima, J., Dubois, Y., and Turner, R. E. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Skey4eBYPS.

Graham, D. and Ravanbakhsh, S. Deep models for relational databases. *arXiv preprint arXiv:1903.09033*, 2019.

Hartford, J. S., Graham, D. R., Leyton-Brown, K., and Ravanbakhsh, S. Deep models of interactions across sets. In *ICML*, 2018.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Kalogerakis, E., Averkiou, M., Maji, S., and Chaudhuri, S. 3d shape segmentation with projective convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3779–3788, 2017.

Keriven, N. and Peyré, G. Universal invariant and equivariant graph neural networks. *CoRR*, abs/1905.04943, 2019. URL http://arxiv.org/abs/1905.04943.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Kondor, R. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv preprint arXiv:1803.01588*, 2018.

Kondor, R. and Trivedi, S. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *arXiv preprint arXiv:1802.03690*, 2018.

Kondor, R., Son, H. T., Pan, H., Anderson, B., and Trivedi, S. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*, 2018.

Litany, O., Remez, T., Rodolà, E., Bronstein, A., and Bronstein, M. Deep functional maps: Structured prediction for dense shape correspondence. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5659–5667, 2017.

Liu, X., Guo, Z., Li, S., Kong, L., Jia, P., You, J., and Kumar, B. Permutation-invariant feature restructuring for correlation-aware image set-based recognition. *arXiv preprint arXiv:1908.01174*, 2019.

Liu, Z., Luo, P., Wang, X., and Tang, X. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15: 2018, 2018.

Maehara, T. and NT, H. A simple proof of the universality of invariant/equivariant graph neural networks, 2019.

Maron, H. and Lipman, Y. (probably) concave graph matching. In *Advances in Neural Information Processing Systems*, pp. 408–418, 2018.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. *arXiv preprint arXiv:1905.11136*, 2019a.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=Syx72jC9tm.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. On the universality of invariant networks. In *International conference on machine learning*, 2019c.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. *arXiv preprint arXiv:1810.02244*, 2018.

Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *arXiv preprint arXiv:1811.01900*, 2018.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Puschel, M. and Moura, J. M. Algebraic signal processing theory: Foundation and 1-d time. *IEEE Transactions on Signal Processing*, 56(8):3572–3585, 2008.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Deep learning with sets and point clouds. *arXiv preprint arXiv:1611.04500*, 2016.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Equivariance through parameter-sharing. *arXiv preprint arXiv:1702.08389*, 2017.

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

Segol, N. and Lipman, Y. On universal equivariant set networks. *arXiv preprint arXiv:1910.02421*, 2019.

Simmons, G. F. *Introduction to topology and modern analysis*. Tokyo, 1963.

Soomro, K., Zamir, A. R., and Shah, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Sridhar, S., Rempe, D., Valentin, J., Bouaziz, S., and Guibas, L. J. Multiview aggregation for learning category-specific shape reconstruction. *arXiv preprint arXiv:1907.01085*, 2019.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

Sutskever, I., Vinyals, O., and Le, Q. Sequence to sequence learning with neural networks. *Advances in NIPS*, 2014.

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

Vinyals, O., Bengio, S., and Kudlur, M. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Wagstaff, E., Fuchs, F. B., Engelcke, M., Posner, I., and Osborne, M. On the limitations of representing functions on sets. *arXiv preprint arXiv:1901.09006*, 2019.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data. 2018. URL http://arxiv.org/abs/1807.02547.

Winkels, M. and Cohen, T. S. 3d g-cnns for pulmonary nodule detection. *arXiv preprint arXiv:1804.04656*, 2018.

Wood, J. and Shawe-Taylor, J. Representation theory and invariant neural networks. *Discrete applied mathematics*, 69(1-2):33–60, 1996.

Worrall, D. and Brostow, G. Cubenet: Equivariance to 3d rotation and translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 567–584, 2018.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yarotsky, D. Universal approximations of invariant maps by neural networks. *arXiv preprint arXiv:1804.10306*, 2018.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Supplementary material

A. Generalizations of equivariant layer characterization

A.1. Equivariant layers for multiple features

The following generalization to sets of elements with multiple features can be proved in a similar way to the section 3.1 in (Maron et al., 2019b).

Theorem 4. *Any linear $G-$equivariant layer $L : \mathbb{R}^{n\times d\times f} \to \mathbb{R}^{n\times d\times f'}$ is of the form*

$$L(X)_i = L_1(x_i) + L_2(\sum_{j\neq i}^{n} x_j),$$

where $L_i,\ i = 1, 2$ are linear H-equivariant functions. The dimension of the space of these layers is $2E(H)ff'$.

A.2. General equivariant and invariant layers

In the main text we characterized all G-invariant functions of the form $L : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$. Here, we characterize all other possibilities of equivariant and invariant functions. The proof is identical to the proof of Theorem 1 in the main paper.

Theorem 5. *1. Any linear $G-$equivariant layer $L : \mathbb{R}^{n\times d} \to \mathbb{R}^{n}$ is of the form $L(X)_i = L_1^H(x_i) + L_2^H(\sum_{j\neq i}^{n} x_j)$, where $L_i^H,\ i = 1, 2$ are linear H-invariant functions.*

2. *Any linear $G-$equivariant layer $L : \mathbb{R}^{n\times d} \to \mathbb{R}^{d}$ is of the form $L(X) = L^H(\sum_{j=1}^{n} x_j)$, where L^H is linear H-equivariant function.*

3. *Any linear G-invariant layer $L : \mathbb{R}^{n\times d} \to \mathbb{R}$ is of the form $L(X) = L^H(\sum_{j=1}^{n} x_j)$, where L^H is linear H-invariant function.*

B. Products of arbitrary permutation groups

Here, we show that Theorem 1 can be generalized to products of arbitrary permutation groups. Our first step is noting that the second part of the proof of Theorem 1 can be easily modified to show that $E(H_1 \times H_2) = E(H_1) \cdot E(H_2)$ for any permutation groups H_1, H_2.

Indeed, Theorem 1 is a special case of the following theorem, which characterizes the space of linear equivariant maps for arbitrary products of permutation groups.

Theorem 6. *Let $H_1 \leq S_n$, $H_2 \leq S_d$ and $\{L_i^j\}_{i=1}^{E(H_j)},\ j = 1, 2$ are bases for the spaces of linear H_j-equivariant maps. Let $G = H_1 \times H_2$ act on $\mathbb{R}^{n\times d}$ by multiplication, $(h_1, h_2) \cdot X := h_1 X h_2^T$. Then, a basis for the space of linear $G-$equivariant layers $L : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ is given by*

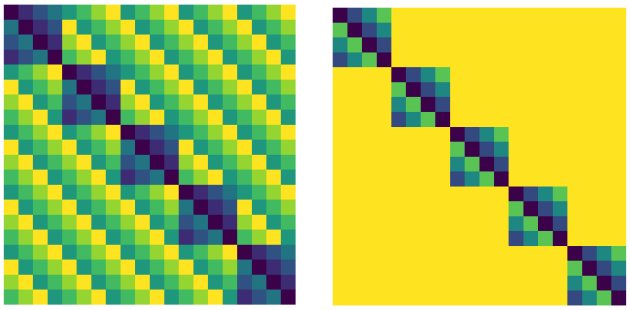
$$T_{i_1,i_2} = L_{i_1}^1 \otimes L_{i_2}^2, \quad i_1 = 1, \ldots, E(H1),\ i_2 = 1, \ldots, E(H_2)$$

Proof. $\{T_{i_1,i_2}\}$ is G-equivariant and linearly independent as a tensor product of linearly independent sets. Moreover, its size is exactly $E(H_1) \cdot E(H_2)$ so it must span the whole space of G-equivariant layers. □

The basis mentioned in Theorem 6 can be implemented using the Kronecker product identity:

$$T_{i_1,i_2}(X) = L_{i_1}^1 X {L_{i_2}^2}^T \tag{6}$$

In other words, these operators can be implemented by applying $L_{i_1}^1$ to the columns of X and $L_{i_2}^2$ to the rows of X.



(a) (b)

Figure 6. parameter sharing schemes for (a) $G = S_n \times H$ and (b) $G = \oplus_{i=1}^{n} H \rtimes S_n$, where $d = 4, n = 5$ and $H = C_4$ the cyclic group of four elements. Each color represents a parameter.

To verify that Theorem 6 is indeed a generalization of Theorem 1, consider $H_1 = S_n$. A basis for S_n-invariant layers is given by $L_1^1 = I_n,\ L_2^1 = \mathbf{1}_n \mathbf{1}_n^T$. From Theorem 6 it follows that a basis for the space of G-invariant linear layers is given by $I_n \otimes L_i^2$ and $\mathbf{1}_n \mathbf{1}_n^T \otimes L_i^2$ which, by Equation 6, gives the basis from Theorem 1.

C. Equivariant layers for order dependent action

As mentioned in the main text, we can consider a different learning setup, where tasks are equivariant to applying different elements of H to different elements in the set. In this section, we formulate this setup and prove that when H acts transitively on $\{1, \dots, d\}$, for example, in the case of images and sets, the corresponding equivariant layers are Siamese H-equivariant layers with an additional global summation term. In this setup, $G = \{(h_1, \dots, h_n, \sigma)\}$ is the semi-direct product $\oplus_{i=1}^{n} H \rtimes S_n$ (also called restricted wreath product) and the action of G on $\mathbb{R}^{n \times d}$ is defined as $((h_1, \dots, h_n, \sigma) \cdot X)_{ij} = X_{\sigma^{-1}(i), h_i^{-1}(j)}$.

We can now characterize the set of G-equivariant layers for this setup.

Theorem 7. *If H acts transitively on $\{1, \dots, d\}$ then any linear G-equivariant layer L is of the form:*

$$L(X)_i = L_1(x_i) + \beta \left(\sum_{j=1}^{n} \sum_{k=1}^{d} x_{jk} \right).$$

L_1 is an H-equivariant layer and $\beta \in \mathbb{R}$. The dimension of the space of linear G-equivariant maps is $E(H) + 1$.

Proof. We want to characterize the space of G-equivariant maps. According to equation 4, we need to find the null space of the following fixed point equation $g \cdot L = L,\ g \in G$. As shown in (Wood & Shawe-Taylor, 1996; Ravanbakhsh et al., 2017), this is equivalent to revealing the parameter-sharing scheme that is induced by G, which we will define next. Let $L \in \mathbb{R}^{nd \times nd}$ represent a linear G-equivariant map, where we think of the input $X \in \mathbb{R}^{n \times d}$ as a row-stack $x \in \mathbb{R}^{nd}$. The works mentioned above assert that $L_{st} = L_{kl}$ if and only if there exists an element $g \in G$ such that $g(s) = l, g(t) = k$. Namely, the indices (s, t) and (k, l) share a parameter if and only if they belong to the same orbit of G when acting on $\{1, \dots, nd\}^2$.

We now find this parameter-sharing scheme for G. For readability, we use two indices (i, j) to represent an index in $s \in \{1, \dots, nd\}$. Given two such indices $(s, t) = (i_s, j_s, i_t, j_t)$ we wish to find their orbit under the action of G. We split this question into two cases and treat them one by one: (1) We first consider the case where $i_s \neq j_s$. In this case, the orbit of (i_s, j_s, i_t, j_t) consists of all indices $(l, k) = (i_l, j_l, i_k, j_k)$ such that $i_l \neq i_k$ which, in turn, implies that all the elements of L that are not on the $d \times d$ block diagonal share their parameter. (2) In the case where $i_s = i_t$, applying the group action shows that all the $d \times d$ diagonal blocks represent the same H-equivariant function. □

Figure 6 illustrates parameter sharing schemes for G-equivariant layers for (a) $G = S_n \times H$ and (b) $G = H^n \rtimes S_n$, where $d = 4, n = 5$ and $H = C_4$ the cyclic group of four elements. Here, each color represents a parameter. Note that all off-diagonal elements in (b) are represented by the same parameter in contrast to (a). The invariant universality proof of Theorem 2 applies in this case as well.

D. Proofs

D.1. Proof of Lemma 1

Proof. As discussed in Section 3, L can be realized as a $\ell \times \ell$ matrix, and the problem of finding all linear G-equivariant functions L can be reduced to solving the following *fixed-point equation*: $g \cdot L = L$. Recall that we are interested in the dimension of the space of linear G-equivariant layers, or equivalently, the dimension of the null space of the fixed-point equation. One way to obtain it, is by applying the trace function to the projection operator onto this null-space. See (Fulton & Harris, 2013), section 2.2 for a derivation. In our case, this projection is given by $\phi = \frac{1}{|G|}\sum_{g\in G} P(g) \otimes P(g)$ which implies:

$$\begin{aligned} E(G) = \text{tr}(\phi) &= \frac{1}{|G|}\sum_{g\in G} \text{tr}(P(g) \otimes P(g)) \\ &= \frac{1}{|G|}\sum_{g\in G} \text{tr}(P(g))^2, \end{aligned}$$

where $\otimes$ is a Kronecker product, $P(g) \otimes P(g)$ is the matrix representation of the action of G on $\mathbb{R}^{\ell\times\ell}$ and we use the fact that the trace is multiplicative with respect to the Kronecker product. □

D.2. Proof of Theorem 2

Proof of lemma 2. This lemma is a generalization of Proposition 1 in (Maron et al., 2019a) and we follow their proof idea. By Noether's theorem (see, e.g., (Yarotsky, 2018; Maron et al., 2019c)), there is a finite set of invariant polynomials $\left(p_i : \mathbb{R}^d \to \mathbb{R}\right)_{i=1}^{l}$ that generate the ring of invariant polynomials, that is, any invariant polynomial $p(x)$ can be written as $p(x) = q(\left(p_i(x)\right\}_{i=1}^{l})$ where $q : \mathbb{R}^l \to \mathbb{R}$ is some general polynomial. We define $u(x) = (p_i(x))_{i=1}^{l}$. On one hand, assume that $y = g\cdot x$ then by the invariance of the polynomials p_i we get $u(y) = u(g\cdot x) = u(x)$. On the other hand, if $u(x) = u(y)$ assume towards contradiction that $g \cdot y \neq x$ for all $g \in G$, then the orbits $G \cdot x$, $G \cdot y$ are disjoint. As both sets are finite, there is a continuous function $f : \mathbb{R}^d \to \mathbb{R}$ such that $f|_{G\cdot x} \leq -2$ and $f|_{G\cdot y} \geq 2$. Using the Stone-Weierstrass theorem (Simmons, 1963) we can get a polynomial p with the property $p|_{G\cdot x} \leq -1$ and $p|_{G\cdot y} \geq 1$. Define $\overline{p} = \frac{1}{|G|}\sum_{g\in G} p(g \cdot x)$ then $\overline{p}$ is a G-invariant polynomial and using the discussion above we can write $\overline{p}(x) = q(\left(p_i(x)\right\}_{i=1}^{l})$ for some polynomial q. This, in turn, implies the following contradiction:

$$1 \leq \overline{p}(y) = q(u(y)) = q(u(x)) = \overline{p}(x) \leq -1$$

□

Proof of Theorem 2. For the "only if" part, assume that G-invariant networks are universal and let $f : K' \to \mathbb{R}$ be a function we would like to approximate on some compact domain $K' \subset \mathbb{R}^d$ using an H-invariant network. We define a new function $\hat{f} : \{(x, \ldots, x) \mid x \in K'\} \to \mathbb{R}$ by $\hat{f}(x, \ldots, x) = f(x)$, and note that the domain of $\hat{f}$ is compact as well. We now use our assumption that G invariant networks are universal to get a G-invariant function that approximates $\hat{f}$ and note that any G-equivariant layer in this network can be seen as an H-equivariant layer since it applies H-equivariant functions to x and $\sum_{i=1}^{n} x$ (and similarly for the G-invariant layer).

The "if" part is a bit more challenging. As mentioned above, our first task is to encode each element x_i with a unique polynomial descriptor. Let u_H be the unique H-invariant descriptor that exists due to Lemma 2, we define the following map $U_H : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d\times l_H}$ by applying u_H in the following way:

$$U_H(X)_{i,j,:} = u_H(x_i), \quad j = 1, \ldots, d$$

In other words, U_H encodes each x_i using u_H and repeats this encoding d times on the second dimension of the output tensor. Note that since u_H is H-invariant then each component of U_H that is applied to a specific element x_i is H-equivariant.

Let $Y \in \mathbb{R}^{n\times d\times l_H}$ denote the output of U_H and $y_i = u_H(x_i)$ the unique H-invariant descriptors. Our second step is to map this set of unique descriptors to a unique set descriptor. By using Lemma 2 again, there exists a function $u_{S_n} : \mathbb{R}^{n\times l_H} \to \mathbb{R}^{l_{S_n}}$ that computes this encoding. Moreover, in the specific case of S_n, u_{S_n} can be chosen to be in the

following form: $u_{S_n}(y_1,\dots,y_n) = \sum_{i=1}^{n} p(y_i)$ where $p : \mathbb{R}^{l_H} \to \mathbb{R}^{l_{S_n}}$ is a multivariate polynomial (see section 4 in (Maron et al., 2019a) for more details). We define:

$$U_{S_n}(Y) = \frac{1}{d}\sum_{i=1}^{n}\sum_{j=1}^{d} p(Y_{i,j,:})$$

and note that $U_{S_n}(Y) = u_{S_n}(y_1,\dots,y_n)$ is exactly the unique S_n-invariant set descriptor, and that H_{S_n} is a G-invariant function as it is composed of applying feature-wise polynomials and summation.

Up until now, we have mapped our set of elements X to a unique set descriptor $U_{S_n}(U_H(X))$. Our next step is to map each such set descriptor to the value $f(X)$. Intuitively, we would have liked to apply the function $r = f \circ (U_{S_n} \circ U_H)^{-1}$ to the output of $U_{S_n} \circ U_H$ but unfortunately, $U_{S_n} \circ U_H$ is not injective so an inverse function is not well defined. Because f and $u = U_{S_n} \circ U_H$ are invariant to the action of $G = S_n \times H$ there exists unique continuous maps $\tilde{f}, \tilde{u}$ from the quotient space $\mathbb{R}^{n\times d}/G$ such that $f = \tilde{f} \circ \pi$ and $u = \tilde{u} \circ \pi$ where π is the projection map to the quotient space. From the fact that our domain $K \subset \mathbb{R}^{n\times d}$ is compact we get that $\tilde{K} = \pi(K)$ is compact and $\tilde{u}$ is bijective between $\tilde{K}$ and its image. We can now write $f = (\tilde{f} \circ \tilde{u}^{-1}) \circ \tilde{u} \circ \pi$ and define $r = \tilde{f} \circ \tilde{u}^{-1}$, which is continuous from lemma 3

In the last stage of the proof, we use the universal approximation properties of MLPs (Cybenko, 1989; Hornik et al., 1989) in order to approximate the three functions mentioned above, i.e., U_{S_n}, U_H, r, using a G-invariant network.

We start with U_H which is defined as an element-wise application of the H-invariant function u_H. We note that U applies a continuous H-invariant function element-wise which can be approximated by an H-invariant network according to our assumption. Furthermore, an element-wise application of an H-invariant network is a G- equivariant network which implies that for any $\epsilon > 0$ there is a G-equivariant network $N_H : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d\times l_H}$ that uniformly approximates it.

Next, we would like to approximate U_{S_n}. From the universality of MLPs there exists MLP an $M_1 : \mathbb{R}^{l}_{H} \to \mathbb{R}^{l_{S_n}}$ and such that M_1 approximates p, which implies that $\sum_{i=1}^{n} M_1(y_i)$ approximates $u_{S_n}(\{y_i\}_{i=1}^{n})$. We define the next equivariant layers to apply M_1 to the feature dimension of Y. We then apply a scaled G-invariant summation function in order to get $\frac{1}{d}\sum_{i=1}^{n}\sum_{j=1}^{d} M_1(y_i)$ as output. Our last function to approximate is r and since it is a continuous function defined on a compact domain we can approximate it with an MLP M_2.

To summarize, we have written our function of interest f as a composition of three functions U_H, U_{S_n}, r, and constructed a networks that uniformly approximates each one of these functions, which, by using the uniform continuity of the functions, gives us a uniform approximation of their composition. □

Lemma 3. *Let $K \subset \mathbb{R}^m$ be a compact domain and $f : K \to \mathbb{R}$ be a continuous function such that $f = h \circ g$. If g is continuous, then h is continuous on $g(K)$.*

Proof. Assume that this is incorrect, then there is a sequence $y_i = g(x_i)$ such that $y_i \to y_0$ but $h(y_i) \nrightarrow h(y_0)$. Without loss of generality, assume that $x_i \to x_0 \in K$ (otherwise choose a converging sub sequence). We have

$$f(x_i) = h(g(x_i)) = h(y_i) \nrightarrow h(y_0) = h(g(x_0)) = f(x_0)$$

which is a contradiction to the continuity of f. □

D.3. Proof of Theorem 3

Proof of Theorem 3. The "only if" part is proved in the same was as in the proof of Theorem 2. For the other side, we first note that G-equivariant polynomials are dense in the space of continuous G-equivariant functions over a compact domain. For proof, see Lemma 4 in (Segol & Lipman, 2019): while the statement in the paper is about S_n equivariant polynomial, the proof trivially extends for every finite group. We therefore start by approximating $P : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$, an equivariant polynomial map of degree at most m. We look at $P_1 = \mathbb{R}^{n\times d} \to \mathbb{R}^{d}$ the first element in the output of P. If P is $S_n \times H$ equivariant then by lemma 4 P_1 is S_{n-1} invariant when S_{n-1} operates on the last $n-1$ rows and H equivariant. If we fix $x_2, ..., x_n$ then $P_1(x_1, ..., x_n)$ is a H-equivariant polynomial in x_1. The space of H-equivariant polynomials of bounded degree is a finite dimensional linear space and therefore has a basis $q_1, ..., q_T$. We can therefore write $P_1(x_1, ..., x_n) = \sum \alpha_k(x_2, ..., x_n) \cdot q_k(x_1)$ where $\alpha_k : \mathbb{R}^{n-1\times d} \to \mathbb{R}$ are the coefficients. Because P_1 is S_{n-1}-invariant,

H-equivariant and q_k are a basis it is easy to see that α_k must be $S_{n-1} \times H$ -invariant: If σ is a permutation on the last n-1 elements then $P_1(x_1, ..., x_n) = P_1(x_1, x_{\sigma(2)}..., x_{\sigma(n)})$ since P_1 is invariant to σ. We then have $\sum \alpha_k(x_2, ..., x_n) \cdot q_k(x_1) = \sum \alpha_k(x_{\sigma(2)}, ..., x_{\sigma(n)}) \cdot q_k(x_1)$ and because q_k form a basis this means that for each k, $\alpha_k(x_2, ..., x_n)$ is equal to $\alpha_k(x_{\sigma(2)}, ..., x_{\sigma(n)})$ proving S_{n-1} invariance. The same idea shows H-invariance.

Next, we note that since P is G-equivariant we have $[P(x_1, .., x_n)]_i = \sum_k \alpha_k(x_1, ...x_{i-1}, x_{i+1}, x_n) \cdot q_k(x_i)$ by applying a permutation that only switches 1 and i. We will now show how this can be approximated using our G-equivariant network. This is the key for the proof as we break down the equivariant function to invariant functions that we can already approximate (up to the fact that they are S_{n-1}-invariant and not S_n invariant), and H-equivariant functions that can be approximated by the assumption on H. The fact that α_k are invariant to permutations of the other $n-1$ elements and not the whole set is not an issue, as that can be implemented easily in our framework as our basic layers separates the sum over other elements with the operation over the current one (see theorem 1) which is exactly the operation needed.

We will use function names from the proof of theorem 2 when applicable for clarity. The first of approximating P will be $U_H : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d \times l_H + 1}$ that maps each element to a unique H-invariant descriptor (same as in the proof of Theorem 2) plus the original information on a separate channel. The second mapping is $U_{S_{n-1}} : \mathbb{R}^{n \times d \times l_H + 1} \to \mathbb{R}^{n \times d \times l_{S_{n-1}} + 1}$ that computes an $S_{n-1} \times H$-invariant representation of the other $n-1$ inputs at each point plus the original input. Next, we need to compute the equivariant polynomial base elements and invariant coefficients. The coefficients are a continuous mapping $r : \mathbb{R}^{l_{S_{n-1}}} \to \mathbb{R}^T$ (proof of continuity is the same as in the proof of Theorem 2) and can be approximated by an MLP, the equivariant polynomials $q_k : \mathbb{R}^d \to \mathbb{R}^d$ are H-equivariant and continuous and can be approximated by an H-equivariant network that is applied to each element independently. The last operation is the multiplication and summation over basis elements, which can be approximated by an MLP on the channel dimension. This breaks down the computation of P into parts that each can be approximated by an equivariant neural network and therefore so can P. Since all polynomials are dense in the space of equivariant functions this shows that each equivariant function can be approximated by an equivariant neural network. □

Lemma 4. *If $f : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ is $S_n \times H$ equivariant, then $f_1 : \mathbb{R}^{n \times d} \to \mathbb{R}^d$ the first element of f is S_{n-1} invariant and H equivariant. We assume S_{n-1} acts by permuting the last n-1 elements.*

Proof. The proof is simple, we can think of f_1 as $\pi_1 \circ f$, ,i.e., f followed by the projection map on the first element. Since the permutations in S_{n-1} leave the first element in place, π_1 is invariant to them and so is f_1 as composition of equivariant and invariant. It is also clear that π_1 is H equivariant making f_1 H equivariant.

□

E. Implementation details

All experiments (unless stated otherwise) were conducted using the PyTorch framework (Paszke et al., 2017), trained with the Adam optimizer (Kingma & Ba, 2014) on NVIDIA V100 GPU. We performed hyper-parameter search for all methods to choose a learning rate in $\{10^{-1}, 10^{-2}, \dots, 10^{-}7\}$. All model architectures use batch normalization (Ioffe & Szegedy, 2015) after each linear layer.

Datasets. The following datasets were used:(1) Places (Zhou et al., 2017), an image dataset with natural scenes such as beach, parking lot or soccer field; (2) UCF101 (Soomro et al., 2012), an action recognition dataset for realistic action videos; (3) Celeba (Liu et al., 2018), a large scale image dataset that contains celebrity faces; (4) Dynamic Faust (Bogo et al., 2017), triangular meshes of real people performing different activities; (5) ImageNet (Deng et al., 2009) large image classification dataset.

E.1. Signal classification experiment

Data preparation. We generated 30,000 training examples and 3,000 test and validation examples. The type of the signal was uniformly sampled from the three possible types (sine, rectangular and saw-tooth). Frequency and amplitude were uniformly sampled from $[1, 10]$, horizontal shift was uniformly sampled from $[0, 2\pi]$, vertical shift was sampled from $[-5, 5]$. From each clean signal, we generate a set of size 25 by replicating the signal and adding independent noise to each copy. The noise is sampled from an i.i.d. Gaussian distribution with zero mean and $3a$ standard deviation where a is the amplitude of the signal.

Network and training. For training we used batch size of 64 and ran for 200 epochs with validation-based early stopping. Training took between 15 minutes for MLP to 5 hours for DSS(Aitalla). For all layer types, we used three layers followed by a fully connected layer with the following number of features: MLP $(840, 420, 420)$, Siamese $(220, 220, 110)$, DSS $(160, 160, 80)$, DSS (max) $(160, 160, 80)$,Siamese+DS(2 Siamese layers + a single DS layer) $(200, 200, 100)$, DS $(1000, 1000, 500)$, DS (max) $(1000, 1000, 500)$, Aittala (Aittala & Durand, 2018) $(160, 160, 80)$, Sridhar (Sridhar et al., 2019) $(220, 220, 110)$. In models that use convolution, we used strided-convolution with stride 2. For all models, we have used sum-pooling on the set and spatial dimensions before the fully connected layer.

E.2. Image selection

Data preparation. The data for the video frame ordering experiment was taken from the UFC101 dataset (Soomro et al., 2012). For the highest quality image selection task we used the Places dataset (Zhou et al., 2017). For the places dataset, we first selected 25 classes that have the largest number of images. We then generated the train and validation sets from the standard train split, and used the standard validation split as test. In both cases, we used 20,000 training examples and 2,000 validation and test examples. The set sizes are $n = 8$ for the frame ordering experiment and $n = 20$ for the image quality assessment experiment. Train Image size was reduced to 80×80 and we used random cropping to 64×64 as well as random flipping as data augmentation. For the image quality task we also used random rotations. For the highest image quality task, we sampled a base blur $\sigma \sim U[0, 1]$ for each image, and another example specific $\sigma' \sim U[0, 1]$ and used $\sigma + \sigma'$ as the Gaussian width for blurring; i.i.d Gaussian noise was added to the result for the Gaussian noise case and i.i.d random pixels where zeroed-out in the Occlusion noise case.

Network and training. For training we used batch size of 16 for 200 epochs with validation-based early stopping. training time was 6.5 (3.75) hours for DS and 4.5 (3.25) hours for DSS for the image quality assessment task (video frame ordering task). The network architecture is based on the image anomaly detection network suggested by (Zaheer et al., 2017) and is composed of convolutional part followed by a DeepSets block. The convolutional part consists of three blocks, each of which consists of the following number of features (32,32,64),(64,64,128),(128,128,256) for *DSS(sum)* and *DSS(max)*,(90,90,100),(100,100,100)(110,110,128) for *DSS(Aittala)* and (50,50,100),(100,100,180),(200,200,256) for DS+Siamese and DSS(Sridhar). All DeepSets blocks have three layers with features(256,128,1).

E.3. Shape selection

Data preparation. The data for the shape selection task was taken from Dynamic Fuast (Bogo et al., 2017). This dataset contains 3D videos of 10 human subjects (male and female) performing 15 activities (e.g. jumping jacks, punching, etc.) The data are represented as triangular meshes of the same topology. Importantly, all the shapes are in one-to-one correspondence. For the graph modality we directly use the mesh, see sec. F for more details. For the point-cloud modality we simply use the mesh vertices. We generated sets of 7 frames by randomly cropping sequences and shuffling their order. Note that, since the scans were captured at 60fps, the motion between few consecutive frames is approximately rigid and at constant velocity. To make the problem more challenging we chose to skip every k frames. To choose k we ran the following simple experiment. For each value of k we computed the mean shape of the set by averaging the point coordinates of all the set elements. We then searched for the shape in the set that was closest to the mean shape and evaluated the accuracy on the validation set. The results were $80.95, 43.75, 32.91, 27.73$ for skip sizes of $1, 3, 5, 10$ respectively. We ended up choosing a skip size of 5. For testing we used a held out set. We chose a very challenging split where both the subjects and the activities are not seen at train time.

Network and training. We repeated all experiments with 3 different seeds, trained for 100 epochs using Adam optimizer on an NVIDIA TitanX with validation-based early stopping. For the point-cloud modality, all methods were ran using a batch-size of 16. Training times were roughly 2 hours. The network architecture is based on a PointNet module (Qi et al., 2017) with 1D convolutions of dimensions (64, 256, 128) followed by a DeepSets block of dimensiones (128, 128, 128, 1). For the graph experiment we used batch-sizes of 8 for the architecture of Aittala, and 12 for all the other architectures. Training took about 20 hours. The architecture is based on a pytorch-geometric (Fey & Lenssen, 2019) implementation of Graph Convolutional Networks (GCN) (Kipf & Welling, 2016) with the adjacency matrix: $\hat{A} = A + 2I$. Dimensions of the graph layers were the same as described above for PointNet. We note that CGN, and in general message passing networks on graphs are not universal approximators.

E.4. Color matching

Data preparation. We used the Places (Zhou et al., 2017) and the CelebA (Liu et al., 2018) datasets. For the places dataset, we first selected 25 classes that have the largest number of images. We generated the train and validation sets from the standard train split, and used the standard validation split as test. For the CelebA dataset, we used the standard splits. In both cases we generate 30,000 train examples and 3,000 examples for validation and test with resolution 64×64

Network and training. We used U-net like networks. All architecture are composed of an encoder followed by a DeepSets block and a decoder. The encoder and decoder are composed of convolution blocks (2X(conv,batchnorm,relu)) according to the DSS variant/Siamese+DS architecture with each folowed by a max pooling layer with stride=2 for the first encoding layers and stride=8 for the last encoding layer. The DeepSets block is composed of three DeepSets layers with the same number of features as in its input. each decoding block applies similar convolution blocks, upsamples the signal and concatenates the appropriate features from the encoding phase. We use the following number of features: (50,100,150,200) for *DSS(sum)* and *DSS(max)*, (64,128,200,300) for *DSS(Sridhar)* and Siamese+DS and (75,100,150,160) for *DSS(Aittala)*. Training was done with batch size = 32 for 50 epochs starting with initial learning rate 0.001 and learning rate decay of 0.4 every 10 epochs, and with validation-based early stopping. We use the L_1 loss.

E.5. Burst image deblurring

Data generation. We follow the protocol in (Aittala & Durand, 2018). We generate blurred images by randomizing a (non-centered) blur kernel and noise. We use a loss that penalizes the deviations of the output image and its gradients from the original image. We also added the mean absolute error of the trivial predictor that outputs the median pixel of the images in the burst at each pixel (the mean predictor produced worse results). we used training set of size 100,000 and test/validation sets of size 10,000, randomly chosen from the ImageNet dataset (Deng et al., 2009). We down-sample images to 128×128 for efficiency. Training was done with batch size=32, learning rate of 0.003 and a decay rate of 0.95 every epoch for 35 epochs and validation-based early stopping.

Network and training. we have used the same network architecture as in the color channel matching experiment followed by a set max pooling layer and two additional 2D convolutions. We have used the following number of features: (48,50,100,150,200) for *DSS(sum)* and *DSS(max)*, (48,64,128,200,300) for *DSS(Sridhar)* and *Siamese+DSS* and (75,100,110,125,125) for *DSS(Aittala)*.

F. Examples

In this subsection, we discuss how our general results can be used in three specific scenarios: learning sets of images, sets of sets, and sets of graphs.

Learning sets of images. In this case, we write $d = h \cdot w$ for $h, w \in \mathbb{N}$, that is, each x_i is a vector in $\mathbb{R}^{hw}$, and we H to be the group of $2D$ circular translations. According to Theorem 1, a general linear equivariant layer for this setup can be written as $L(x_i) = L_1(x_i) + L_2\left(\sum_{j \neq i} x_j\right)$. In other words, the layer consists of two different convolutional layers, where the first layer L_1 is applied to each image independently and the second layer L_2 is applied to the sum of all images. This layer is easy to implement and we make an extensive use of it in the experiment section (section 6). We note that certain temporal or periodic signals can be handled in a similar fashion. In this case $x_i \in \mathbb{R}^k$ and is the group of $1D$ circular translations.

Learning sets of sets. Another useful application of our theory is for learning sets of consistently-ordered sets. See Section 6 for an example on sets of point-clouds. Here, we set $d = m \times k$ where each item x_i is an $m \times k$ matrix representing a set of k-dimensional points. The group H in this case is S_m which acts by permuting rows. We note that equivariance to this type of group action was first considered by (Hartford et al., 2018) for learning interactions between sets. In their paper, Hartford et al. (2018) also characterized the maximal linear equivariant basis (a special case of Theorem 1) which (as they nicely show) can be easily implemented by simple summation operations.

Learning sets of graphs. Our layers can also be used to learn sets of graphs for tasks such as graph anomaly detection and graph classification. See Section 6 for an example on graphs that represent 3D shapes. In this case, $d = k^2$ and each

item x_i is a $k \times k$ tensor (possibly with another feature dimension) representing the interactions between the k vertices in the graph (e.g., an adjacency of affinity matrix). The group H in this case is S_k, acting on x_i by permuting its rows and columns.

Revisiting Deep Sets As our final example we note that both the characterization of equivariant layers and the universal approximation results in (Zaheer et al., 2017) are special cases of our theoretical results (Theorems 1, 2) where we set $H = I_d$, that is, the symmetry group of the elements x_i is trivial.

G. Multi-view reconstruction

We tested DSS on a multi-view reconstruction task. Here, the input is a set of images of a 3D object and the task is to predict its 3D structure. We closely follow Sridhar et al. (2019), that pose this task as a learning problem in which a network is trained to "lift" image pixels in each view to their 3D normalized coordinate space (NOCS). NOCS is unique in that it canonicalizes shape pose and scale and thus makes the view-aggregation as simple as a union operation. In addition to predicting the NOCS representation of each foreground pixel, the network also predicts the coordinates of the occluded part of the object as if the camera was an x-ray.

The architecture proposed by (Sridhar et al., 2019) advocates a mean-subtraction aggregation scheme. After each convolutional block, the mean of all set elements is subtracted. This aggregation scheme can be seen as a specific case of DSS, in which the sum of all elements is further processed by a different convolution layer and only then added to the elements. This raises up an interesting question of whether a simple modification to the architecture of (Sridhar et al., 2019), in the form of changing the aggregation step to apply a convolution block to the sum of all set elements, can improve performance.

Following the same experimental settings prescribed by the authors, we tested the modified architecture (named Sridhar+DSS) in the case of a fixed-sized input of 3 views per model on three different classes of 3d objects as in (Sridhar et al., 2019): cars, Airplanes and chairs. The results are summarized in table 4. As can be seen, our proposed modification gives a significant boost in performance on 2 out of 3 object classes. While we lack a good explanation for why the performance on chairs is decreased, this result suggests that it is worth to further explore the potential benefit of DSS for this task. We leave the full exploration of this specific task to future work.

Category	Sridhar	Sridhar+DSS
Cars	0.1645	**0.1273**
Airplanes	0.1571	**0.1163**
Chairs	**0.1845**	0.2345
Average	0.1687	**0.1593**

Table 4. Reconstruction error for the Multi-view 3D object reconstruction task. We compare the performance reported in (Sridhar et al., 2019) and our suggested modification (Sridhar+DSS). Reported errors are 2-way Chamfer distance between the ground truth shape and its reconstruction, multiplied by 100.

explora

↓

model learnig

↓

Group theomry