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*

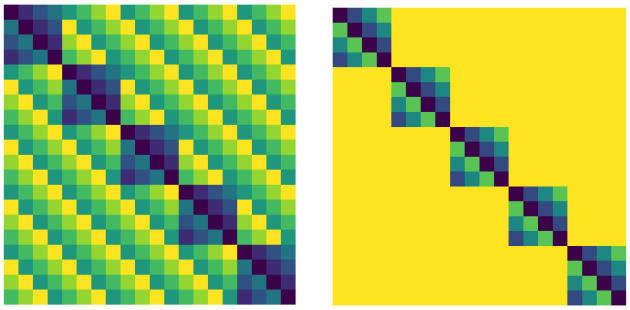
$$T_{i_1,i_2} = L_{i_1}^1 \otimes L_{i_2}^2, \quad i_1 = 1, \ldots, E(H1),\ i_2 = 1, \ldots, E(H_2)$$

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.