Set Norm and Equivariant Skip Connections: Putting the Deep in Deep Sets

Lily H. Zhang [*1] **Veronica Tozzo** [*2 3] **John M. Higgins** [2 3] **Rajesh Ranganath** [1 4]

Abstract

Permutation invariant neural networks are a promising tool for making predictions from sets. However, we show that existing permutation invariant architectures, Deep Sets and Set Transformer, can suffer from vanishing or exploding gradients when they are deep. Additionally, layer norm, the normalization of choice in Set Transformer, can hurt performance by removing information useful for prediction. To address these issues, we introduce the "clean path principle" for equivariant residual connections and develop set norm (SN), a normalization tailored for sets. With these, we build Deep Sets++ and Set Transformer++, models that reach high depths with better or comparable performance than their original counterparts on a diverse suite of tasks. We additionally introduce Flow-RBC, a new single-cell dataset and real-world application of permutation invariant prediction. We open-source our data and code here: https://github.com/rajesh-lab/deep_permutation_invariant.

1. Introduction

Many real-world tasks involve predictions on sets as inputs, from point cloud classification (Guo et al., 2020; Wu et al., 2015; Qi et al., 2017a) to the prediction of health outcomes from single-cell data (Regev et al., 2017; Lähnemann et al., 2020; Liu et al., 2021; Yuan et al., 2017).

Models applied to input sets should satisfy *permutation invariance*: for any permutation of the elements in the input set, the model prediction stays the same. Deep Sets (Zaheer et al., 2017) and Set Transformer (Lee et al., 2019) are two general-purpose permutation-invariant neural networks that have been proven to be universal approximators of permutation-invariant functions under the right conditions (Zaheer et al., 2017; Lee et al., 2019; Wagstaff et al., 2019). In practice, however, these architectures are often tailored to specific tasks to achieve good performance (Zaheer et al., 2017; Lee et al., 2019).

In this work, we pursue a general approach to achieve improved performance: making permutation-invariant networks deeper. Whether deeper models benefit performance is often task-dependent, but the strategy of building deeper networks has yielded benefit for a variety of architectures and tasks (He et al., 2016b; Wang et al., 2019; Li et al., 2019). Motivated by these previous results, we investigate whether similar gains can be made of permutation-invariant architectures and prediction tasks on sets.

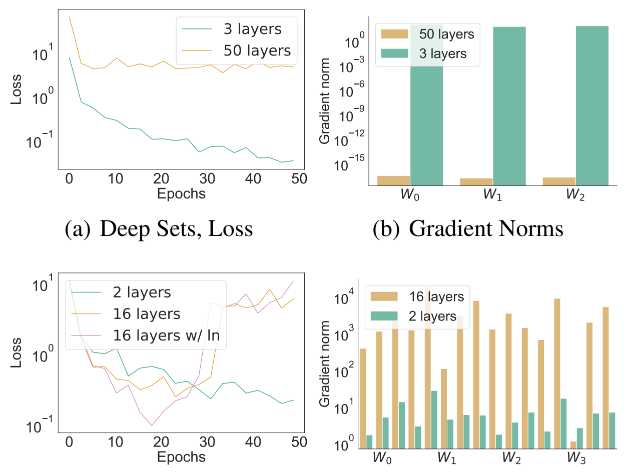
However, naively increasing layers in Deep Sets and Set Transformer can hurt performance (see Figure 1). We show empirical evidence, supported by a gradient analysis, that both models can suffer from vanishing or exploding gradients (Section 3.1, Section 3.2). Moreover, we observe that layer norm, the normalization layer discussed in Set Transformer, can actually hurt performance on tasks with real-valued sets, as its standardization forces potentially unwanted invariance to scalar transformations in set elements (Section 3.3).

To address these failures, we introduce Deep Sets++ and Set Transformer++, new versions of Deep Sets and Set Transformer with carefully designed residual connections and normalization layers (Section 4). First, we propose skip connections that adhere to what we call the "clean path" principle to address potential gradient issues. Next, we propose set norm (SN), an easy-to-implement normalization layer for sets which standardizes each set over the minimal number of dimensions. We consider both residual connections and normalization layers since either alone can still suffer from gradient problems (Zhang et al., 2018; Yang et al., 2019; De & Smith, 2020).

Deep Sets++ and Set Transformer++ are able to train at high depths without suffering from the issues seen in the original models (Section 7). Furthermore, deep versions of these architectures improve upon their shallow counterparts on many tasks, avoiding issues such as exploding or vanishing

[*]Equal contribution [1]Center for Data Science, New York University, New York, NY [2]Massachusetts General Hospital, Harvard Medical School, Cambridge, MA [3]Department of Systems Biology, Harvard Medical School, Boston, MA [4]Department of Computer Science, New York University, New York, NY. Correspondence to: Lily H. Zhang <lily.h.zhang@nyu.edu>, Veronica Tozzo <vtozzo@mgh.harvard.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).



(a) Deep Sets, Loss (b) Gradient Norms

(c) Set Transformer, Train Loss (d) Gradient Norms

Figure 1: At high depths, Deep Sets can suffer from vanishing gradients (top), while Set Transformer can suffer from exploding gradients (bottom). Experiment is MNIST digit variance prediction (see Section 6 for details).

gradients. Among other results, these new architectures yield better accuracy on point cloud classification than the task-specific architectures proposed in the original Deep Sets and Set Transformer papers.

We also introduce a new dataset for permutation-invariant prediction called Flow-RBC (Section 5). The dataset consists of red blood cell (RBC) measurements and hematocrit levels (i.e. the fraction of blood volume occupied by RBCs) for 100,000+ patients. The size and presence of a prediction target (hematocrit) makes this dataset unique, even among single-cell datasets in established repositories like the Human Cell Atlas (Regev et al., 2017). Given growing interest around single-cell data for biomedical science (Lähnemann et al., 2020), Flow-RBC provides machine learning researchers with the opportunity to benchmark their methods on an exciting new real-world application.

2. Permutation invariance

Let M be the number of elements in a set, and let $\mathbf{x}$ denote a single set with samples $\mathbf{x}_1, ..., \mathbf{x}_M, \mathbf{x}_i \in \mathcal{X}$. A function $f : \mathcal{X}^M \to \mathcal{Y}$ is *permutation invariant* if any permutation π of the input set results in the same output: $f(\pi\mathbf{x}) = f(\mathbf{x})$. A function $\sigma : \mathcal{X}^M \to \mathcal{Y}^M$ is *permutation equivariant* if, for any permutation π, the outputs are permuted accordingly: $\sigma(\pi\mathbf{x}) = \pi\sigma(\mathbf{x})$. A function is permutation-invariant if and only if it is sum-decomposable with sufficient conditions on the latent space dimension (Zaheer et al., 2017; Wagstaff et al., 2019). A sum-decomposable function $f : \mathcal{X}^M \to \mathcal{Y}$ is one which can be expressed using a function $\phi : \mathcal{X} \to \mathcal{Z}$ mapping each input element to a latent vector, a sum aggregation over the elements of the resulting output, and an unconstrained decoder $\rho : \mathcal{Z} \to \mathcal{Y}$:

$$f(\mathbf{x}) = \rho\left(\sum_{i=1}^{M} \phi(\mathbf{x}_i)\right). \tag{1}$$

Existing permutation-invariant architectures utilize the above fact to motivate their architectures, which consist of an equivariant encoder, permutation-invariant aggregation, and unrestricted decoder. Equivariant encoders can express $\phi(\mathbf{x}_i)$ for each element $\mathbf{x}_i$ if interactions between elements are zeroed out. For the remainder of the paper, we consider the depth of a permutation-invariant network to be the number of layers in the equivariant encoder. We do not consider decoder changes as the decoder is any unconstrained network, so we expect existing work on increasing depth to directly transfer.

3. Problems with Existing Architectures

Both Deep Sets and Set Transformer are permutation invariant (Zaheer et al., 2017; Lee et al., 2019). However, a gradient analysis of each shows that both architectures can exhibit vanishing or exploding gradients. We present experimental evidence of vanishing and exploding gradients in Deep Sets and Set Transformer respectively (Figure 1).

3.1. Deep Sets gradient analysis

Deep Sets consists of an encoder of equivariant feedforward layers (where each layer is applied independently to each element in the set), a sum or max aggregation, and a decoder also made up of feedforward layers (Zaheer et al., 2017). Each feedforward layer is an affine transform with a ReLU non-linearity: for layer ℓ and element i, we have $\mathbf{z}_{\ell,i} = \mathrm{relu}(\mathbf{z}_{\ell-1,i}W_\ell + b_\ell)$. We denote the output after an L-layer encoder and permutation-invariant sum aggregation as $\mathbf{y} = \sum_i \mathbf{z}_{L,i}$. Then, the gradient of weight matrix W_1 of the first layer is as follows:

$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \sum_i \frac{\partial \mathbf{y}}{\partial \mathbf{z}_{L,i}} \frac{\partial \mathbf{z}_{L,i}}{\partial W_1}. \tag{2}$$

The rightmost term above is a product of terms which can become vanishingly small when the number of layers L is large:

$$\begin{aligned}
\frac{\partial \mathbf{z}_{L,i}}{\partial W_1} &= \frac{\partial \mathbf{z}_{L,i}}{\partial \mathbf{z}_{1,i}} \frac{\partial \mathbf{z}_{1,i}}{\partial W_1} && (3)\\
&= \prod_{\ell=2}^{L} \frac{\partial \mathbf{z}_{\ell,i}}{\partial \mathbf{z}_{\ell-1,i}} \frac{\partial \mathbf{z}_{1,i}}{\partial W_1} && (4)\\
&= \frac{\partial \mathbf{z}_{1,i}}{\partial W_1} \prod_{\ell=2}^{L} \frac{\partial \mathrm{relu}(\mathbf{z}_{\ell,i})}{\partial \mathbf{z}_{\ell,i}} W_\ell. && (5)
\end{aligned}$$

This gradient calculation mirrors that of a vanilla feedforward network, except for the additional summation over each of the elements (or the corresponding operation for max aggregation). Despite the presence of the sum, the effect of a product over many layers of weights still dominates the overall effect on the gradient of earlier weights. We provide experimental evidence in Figure 1.

3.2. Set Transformer gradient analysis

Set Transformer consists of an encoder, aggregation, and decoder built upon a multihead attention block (MAB) (Lee et al., 2019).[1] The MAB differs from a transformer block in that its skip connection starts at the linearly transformed input $\mathbf{x}W_Q$ rather than $\mathbf{x}$ (see Equation (7)).[2] Let Attn_K be multihead attention with K heads and a scaled softmax, *i.e.* $\text{softmax}(\cdot/\sqrt{D})$ where D is the number of features. Then, MAB can be written as:

$$\text{MAB}_K(\mathbf{x},\mathbf{y}) = f(\mathbf{x},\mathbf{y}) + \text{relu}(f(\mathbf{x},\mathbf{y})W + \mathbf{b}), \quad (6)$$
$$f(\mathbf{x},\mathbf{y}) = \mathbf{x}W_Q + \text{Attn}_K(\mathbf{x},\mathbf{y},\mathbf{y}). \quad (7)$$

The Set Transformer encoder block is a sequence of two MAB blocks, the first between learned inducing points and the input $\mathbf{x}$, and the second between the input $\mathbf{x}$ and the output of the first block. Given D hidden units and M learned inducing points $\mathbf{p}$,[3] the inducing point set attention block (ISAB) can be written as such:

$$\text{ISAB}_M(\mathbf{x}) = \text{MAB}_K(\mathbf{x},\mathbf{h}) \in \mathbb{R}^{S\times D} \quad (8)$$
$$\text{where } \mathbf{h} = \text{MAB}_K(\mathbf{p},\mathbf{x}) \in \mathbb{R}^{M\times D}. \quad (9)$$

The aggregation block is an MAB module between a single inducing point and the output of the previous block ($M = 1$ in Equation (9)), and the decoder blocks are self-attention modules between the previous output and itself. In Lee et al. (2019), layer norm is applied to the outputs of Equation (6) and Equation (7) in the MAB module definition but is turned off in the experiments.

Consider a single ISAB module. We let $\mathbf{z}_1$ denote the output of the previous block, $\mathbf{z}_2$ denote the output after the first MAB module (i.e. $\mathbf{h}$ in Equation (9)), and $\mathbf{z}_3$ denote the output of the second MAB module, or the overall output of the ISAB module. Then,

$$f_1 = f(\mathbf{p},\mathbf{z}_1) = IW_{1Q} + \text{Attn}_K(\mathbf{p},\mathbf{z}_1,\mathbf{z}_1) \quad (10)$$
$$\mathbf{z}_2 = f_1 + \text{relu}(f_1W_1 + \mathbf{b}_1) \quad (11)$$
$$f_2 = f(\mathbf{z}_1,\mathbf{z}_2) = \mathbf{z}_1W_{2Q} + \text{Attn}_K(\mathbf{z}_1,\mathbf{z}_2,\mathbf{z}_2) \quad (12)$$
$$\mathbf{z}_3 = f_2 + \text{relu}(f_2W_2 + \mathbf{b}_2). \quad (13)$$

[1]The MAB module in Lee et al. (2019) should not be confused with multihead attention (Vaswani et al., 2017), which is a component of the module.

[2]The implementation of the MAB module in code differs from the definition in the paper. We follow the former.

[3]Typically, $M << S$ for computational efficiency.

	No norm	Layer norm
Hematocrit	18.7436 ± 0.0148	$\underline{19.0904 \pm 0.1003}$
Point Cloud	0.9217 ± 0.0119	0.9219 ± 0.0052
Normal Var	0.0023 ± 0.0006	$\underline{0.0801 \pm 0.0076}$

Table 1: Set Transformer can perform worse (underlined) with layer norm than with no normalization, particularly when inputs are real-valued. Results are test loss over three seeds (CE for Point Cloud, MSE for rest). Lower is better.

Let I denote the identity matrix Then, the gradient of a single ISAB block output $\mathbf{z}_3$ with respect to its input $\mathbf{z}_1$ can be represented as $\frac{\partial \mathbf{z}_3}{\partial \mathbf{z}_1} = \frac{\partial \mathbf{z}_3}{\partial f_2}\frac{\partial f_2}{\partial \mathbf{z}_1}$, or

$$\Big(I + \frac{\partial \text{relu}(f_2W_2+\mathbf{b}_2)}{\partial(f_2W_2+\mathbf{b}_2)}W_2\Big)\Big(W_{2Q} + \frac{\partial \text{Attn}_K(\mathbf{z}_1,\mathbf{z}_2,\mathbf{z}_2)}{\partial \mathbf{z}_1}\Big).$$

In particular, we notice that even if the elements in $\frac{\partial \text{relu}(f_2W_2+\mathbf{b}_2)}{\partial(f_2W_2+\mathbf{b}_2)}W_2$ and $\frac{\partial \text{Attn}_K(\mathbf{z}_1,\mathbf{z}_2,\mathbf{z}_2)}{\partial \mathbf{z}_1}$ are close to zero, the weights W_{2Q} will affect the partial derivatives of each ISAB output with respect to its input. The gradient of earlier weights will be the product of many terms of the above form, and this product can explode when the magnitude of the weights grows, causing exploding gradients and unstable training (see Figure 1(c) for an example). We find experimentally that even with the addition of layer norm, the problem persists. See Appendix B.1 for an analogous gradient analysis with the inclusion of layer norm.

Based on the gradient analysis provided for both Deep Sets and Set Transformer, both vanishing and exploding gradients are possible for both models. In our experiments, we primarily see evidence of vanishing gradients for Deep Sets and exploding gradients for Set Transformer.

3.3. Layer norm can hurt performance

Layer norm (Ba et al., 2016) was introduced for permutation-invariant prediction tasks in Set Transformer (Lee et al., 2019), mirroring transformer architectures for other tasks. However, while layer norm has been shown to benefit performance in other settings (Ba et al., 2016; Chen et al., 2018), we find that layer norm can in fact hurt performance on certain tasks involving sets (see Table 1).

Let $\overrightarrow{\mu}_{\mathbf{z}}, \overrightarrow{\sigma}_{\mathbf{z}} \in \mathbb{R}^D$ be the statistics used for standardization of a vector $\mathbf{z} \in \mathbb{R}^D$ and $\overrightarrow{\gamma}, \overrightarrow{\beta} \in \mathbb{R}^D$ be transformation parameters acting on each feature independently. Then, given a set with elements $\{\mathbf{x}_i\}_{i=1}^M \in \mathbb{R}^D$, layer norm first standardizes each element independently $\bar{\mathbf{x}}_i = \frac{\mathbf{x}_i - \mu_{\mathbf{x}_i}}{\sigma_{\mathbf{x}_i}}$, and then transforms $\hat{\mathbf{x}}_i = \mathbf{x}_i \odot \overrightarrow{\gamma} + \overrightarrow{\beta}$.

Element-wise standardization forces an invariance where two elements whose activations differ in only a scale yield the same output when processed through layer norm following a linear projection. If we consider layer norm in is typical placement, after a linear projection and before the non-linear activation $f(\mathbf{x}_i) = \text{relu}(\text{LN}(\mathbf{x}_i W))$ (Ba et al., 2016; Ioffe & Szegedy, 2015; Ulyanov et al., 2016; Cai et al., 2021), we have that for $\mathbf{x}_i$ and $\mathbf{x}_{i'} = \alpha \mathbf{x}_i$, $\alpha \in \mathbb{R}$,

$$\text{LN}(\mathbf{x}_{i'} W) = \frac{(\alpha \mathbf{x}_i)W - \overrightarrow{\mu}_{\mathbf{x}_{i'} W}}{\overrightarrow{\sigma}_{\mathbf{x}_{i'} W}} * \overrightarrow{\gamma} + \overrightarrow{\beta} \tag{14}$$

$$= \frac{\alpha \mathbf{x}_i W - \alpha \overrightarrow{\mu}_{\mathbf{x}_i W}}{\alpha \overrightarrow{\sigma}_{\mathbf{x}_i W} W} * \overrightarrow{\gamma} + \overrightarrow{\beta} \tag{15}$$

$$= \frac{\mathbf{x}_i W - \overrightarrow{\mu}_{\mathbf{x}_i W}}{\overrightarrow{\sigma}_{\mathbf{x}_i W}} * \overrightarrow{\gamma} + \overrightarrow{\beta} = \text{LN}(\mathbf{x}_i W). \tag{16}$$

Since $\text{LN}(\mathbf{x}_{i'} W) = \text{LN}(\mathbf{x}_i W)$, $f(\mathbf{x}_{i'}) = f(\mathbf{x}_i)$, meaning the two elements are indistinguishable at this point in the network. This invariance reduces representation power (two such samples cannot be treated differently in the learned function) and removes information which may potentially be useful for prediction (i.e. per-element mean and standard deviation).

An Example in 2D. Consider sets of two-dimensional real-valued elements and a model with 2D activations. Layer norm's standardization will map all elements to either (-1, 1), (0, 0), or (1, -1), corresponding to whether the first coordinate of each element is less than, greater than, or equal to the second coordinate. If the task is classifying 2D point clouds, any two shapes which share the same division of points on either side of the $y = x$ line will be indistinguishable (see Appendix B.2 for a visualization). Generalizing this phenomenon to higher dimensions, layer norm's standardization decreases the degrees of freedom in elements' outputs relative to their inputs, an effect that can be particularly harmful for sets of low-dimensional, real-valued elements.

In contrast, layer norm is commonly used in NLP, where one-hot encoded categorical tokens will not be immediately mapped to the same outputs. Differences such as these ones highlight the need to consider normalization layers tailored to the task and data type at hand.

Our analysis on gradients and layer norm does not suggest that these issues will always be present. However, the possibility of these issues, as well as experimental evidence thereof, raises the need for alternatives which do not exhibit the same problems.

4. Deep Sets++ and Set Transformer++

We propose Deep Sets++ and Set Transformer++, new architectures that differ from the originals only in their encoders, as we fix the decoder and aggregation to their original versions. For simplicity, we let the hidden dimension remain constant throughout the encoder. Based on the analysis of Section 3, we explore alternative residual connections scheme to fix the vanishing and exploding gradients. Moreover, given the potential issues with layer norm for real-valued set inputs, we consider an alternative normalization. Concretely, we propose the clean-path equivariant residual connections and set norm.

4.1. Clean-path equivariant residual connections

Let f be an equivariant function where $\mathcal{X} = \mathcal{Y} = \mathbb{R}^D$, i.e. $f : \mathbb{R}^{M \times D} \to \mathbb{R}^{M \times D}$. A function g which adds each input to its output after applying any equivariant function f is also equivariant:

$$g(\pi \mathbf{x}) = f(\pi \mathbf{x}) + \pi \mathbf{x} = \pi f(\mathbf{x}) + \pi \mathbf{x} = \pi g(\mathbf{x}).$$

While such residual connections exist in the literature (Weiler & Cesa, 2019; Wang et al., 2020), here we refer to them as *equivariant residual connections* (ERC) to highlight their equivariant property and differentiate them from other possible connections that skip over blocks (see Section 7 for an example). In sets, ERCs act on every element and eliminate the vanishing gradient problem (see Appendix G for a gradient analysis).

ERCs can be placed in different arrangements within an architecture (He et al., 2016b;a; Vaswani et al., 2017; 2018). We consider *non-clean path* and *clean path* arrangements. Let l indicate the layer in the network. Non-clean path blocks include operations before or after the residual connections and must be expressed as either

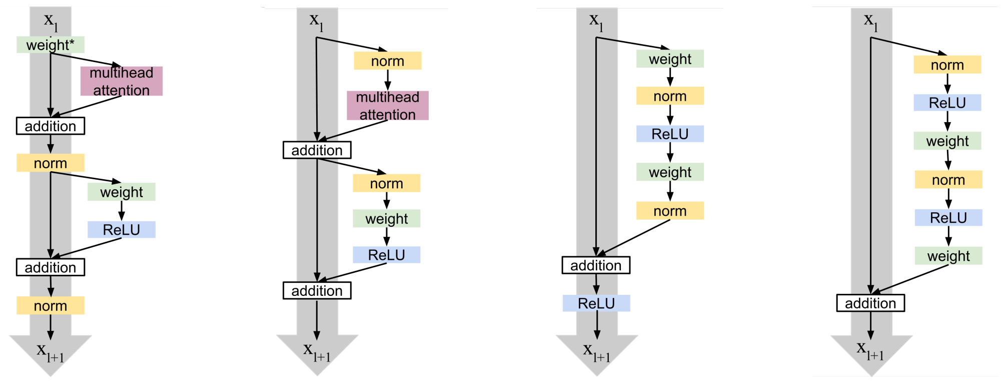
$$\mathbf{x}_{l+1} = g(\mathbf{x}_l) + f(\mathbf{x}_l) \quad \text{or} \quad \mathbf{x}_{l+1} = g(\mathbf{x}_l + f(\mathbf{x}_l)), \tag{17}$$

where g, f cannot be the identity function. This arrangement was used in the MAB module of the Set Transformer architecture (see Figure 2 panel a). Previous literature on non permutation-invariant architectures shows that the presence of certain operations between skip connections could yield undesirable effects (He et al., 2016a;b; Klein et al., 2017; Vaswani et al., 2018; Xiong et al., 2020).

In contrast, *clean path* arrangements add the unmodified input to a function applied on it,

$$\mathbf{x}_{l+1} = \mathbf{x}_l + f(\mathbf{x}_l), \tag{18}$$

resulting in a clean path from input to output (see gray arrows in Figure 2 b and d). The clean path MAB block (Figure 2 panel b) mirrors the operation order of the Pre-LN Transformer (Klein et al., 2017; Vaswani et al., 2018), while the clean path version of Deep Sets mirrors that of the modified ResNet in He et al. (2016a) (Figure 2 panel d).



(a) Non-clean path for Set Transformer (b) Clean path for Set Transformer (c) Non-clean path for Deep Sets (d) Clean path for Deep Sets

Figure 2: Clean path variants have no additional operations on the residual path (denoted by a grey arrow), whereas non-clean path variants do. In (c), weight* is also part of the attention computation.

4.2. Set norm

Designing normalization layers for permutation equivariant encoders requires careful consideration, as not all normalization layers are appropriate to use. To this aim, we analyze normalization layers as a composition of two operations: standardization and transformation. This setting captures most common normalizations (Ioffe & Szegedy, 2015; Ba et al., 2016; Ulyanov et al., 2016).

Let $\mathbf{a} \in \mathbb{R}^{N\times M\times D}$ be the activation before the normalization operation, where N is the size of the batch, M is the number of elements in a set (sets are zero-padded to the largest set size), and D is the feature dimension. First, the activations are standardized based on a setting $\mathcal{S}$ which defines which dimensions utilize separate statistics. For instance, $\mathcal{S} = \{N, M\}$ denotes that each set in a batch and each element in a set calculates its own mean and standard deviation for standardization, e.g. $\mu_{\mathcal{S}}(\mathbf{a})_{b,s} = \frac{1}{D}\sum_{d=1}^{D} \mathbf{a}_{n,i,d}$. Results are repeated over the dimensions not in $\mathcal{S}$ so that $\mu_{\mathcal{S}}(\mathbf{a}), \sigma_{\mathcal{S}}(\mathbf{a}) \in \mathbb{R}^{N\times M\times D}$ match $\mathbf{a}$ in dimensions for elementwise subtraction and division. A standardization operation can be defined as:

$$\bar{\mathbf{a}}_{\mathcal{S}} = \frac{\mathbf{a} - \mu_{\mathcal{S}}(\mathbf{a})}{\sigma_{\mathcal{S}}(\mathbf{a})}, \tag{19}$$

where we assume that the division is well-defined (i.e. non-zero standard deviation).

Next, the standardized activations are transformed through learned parameters which differ only over a setting of dimensions $\mathcal{T}$. For instance, $\mathcal{T} = \{D\}$ denotes that each feature is transformed by a different scale and bias, which are shared across the sets in the batch and elements in the sets. Let $\overrightarrow{\gamma}_{\mathcal{T}}, \overrightarrow{\beta}_{\mathcal{T}} \in \mathbb{R}^{N\times M\times D}$ denote the learned parameters and $\odot$ represent elementwise multiplication. Any transformation operation can be defined as:

$$\hat{\mathbf{a}}_{\mathcal{T}} = \bar{\mathbf{a}} \odot \overrightarrow{\gamma}_{\mathcal{T}} + \overrightarrow{\beta}_{\mathcal{T}}. \tag{20}$$

Proposition 1. *Let $\mathcal{F}$ be the family of transformation functions which can be expressed via Equation (20). Then, for $f \in \mathcal{F}$, $\mathcal{T} = \{D\}$ and $\mathcal{T} = \{\}$ are the only settings satisfying the following properties:*

1. *$f_{\mathcal{T}}(\pi_i \boldsymbol{a}) = \pi_i f_{\mathcal{T}}(\boldsymbol{a})$ where π_i is a permutation function that operates on elements in a set;*
2. *$f_{\mathcal{T}}(\pi_n \boldsymbol{a}) = \pi_n f_{\mathcal{T}}(\boldsymbol{a})$ where π_n is a permutation function that operates on sets.*

See Appendix C for proof. In simpler terms, the settings $\mathcal{T} = \{D\}$ and $\mathcal{T} = \{\}$ are the only ones that maintain permutation invariance and are agnostic to set position in the batch. The setting $\mathcal{T} = \{D\}$ contains $\mathcal{T} = \{\}$ and is more expressive, as $\mathcal{T} = \{\}$ is equivalent to $\mathcal{T} = \{D\}$ where learned parameters $\overrightarrow{\gamma}_{\mathcal{T}}, \overrightarrow{\beta}_{\mathcal{T}}$ each consist of a single unique value. Thus, we choose $\mathcal{T} = \{D\}$ as our choice of transformation.

Standardization will always remove information; certain mean and variance information become unrecoverable. However, it is possible to control what information is lost based on the choice of dimensions over which standardization occurs.

With this in mind, we propose *set norm* (SN), a new normalization layer designed to standardize over the fewest number of dimensions of any standardization which acts on each set separately. Per-set standardizations are a more practical option for sets than standardizations which happen over a batch ($N \notin \mathcal{S}$, batch norm is an example), as the latter introduce issues such as inducing dependence between inputs, requiring different procedures during train and test, and needing tricks such as running statistics to be stable. In addition, any standardization over a batch needs to take into account how to weight differentially-sized sets in calculating the statistics as well as how to deal with small batch sizes caused by large inputs.

Set norm is a normalization defined by a per set standardization and per feature transformation ($\mathcal{L} = \{N\}, \mathcal{T} = \{D\}$):

$$\text{SN}(a_{nid}) = \frac{\mathbf{a}_n - \mu_n}{\sigma_n} \odot \gamma_d + \beta_d,$$
$$\mu_n = \frac{1}{M}\frac{1}{D}\sum_{i=1}^{M}\sum_{d=1}^{D} a_{nid},$$
$$\sigma_n^2 = \frac{1}{M}\frac{1}{D}\sum_{i=1}^{M}\sum_{d=1}^{D}(a_{nid} - \mu_n)^2.$$

Set norm is permutation equivariant (see Appendix C for proof). It also standardizes over the fewest dimensions possible of any per-set standardization, resulting in the least amount of mean and variance information removed (e.g. only the global mean and variance of the set rather than the mean and variance of each sample in the set in the case of layer norm). Note that set norm assumes sets of size greater than one ($M > 1$) or multi-sets in which at least two elements are different.

Table 2: Set norm can improve performance of 50-layer Deep Sets, while layer norm does not (Point Cloud, MNIST Var). In some cases, normalization alone is not enough to overcome vanishing gradients (Hematocrit, Normal Var). Table reports test loss (CE for Point Cloud, MSE otherwise). Lower is better. Uunderlined results are notable failures.

	no norm	layer norm	set norm
Hematocrit	25.879 ± 0.001	25.875 ± 0.002	25.875 ± 0.002
Point Cloud	3.609 ± 0.000	<u>3.619 ± 0.000</u>	**1.542 ± 0.086**
MNIST Var	5.555 ± 0.001	<u>5.565 ± 0.001</u>	**0.259 ± 0.003**
Normal Var	8.4501 ± 0.0031	8.4498 ± 0.0054	8.4433 ± 0.0011

Next, we combine clean-path equivariant residual connections and set norm to build modified permutation-invariant architectures Deep Sets++ and Set Transformer++.

4.3. Deep Sets++ (DS++)

DS++ adopts the building blocks mentioned above, resulting in a residual block of the form

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \text{SetNorm}(W_{l1}(\text{relu}(\text{SetNorm}(W_{l2}\mathbf{x}_l)))).$$

The DS++ encoder starts with a first linear layer and no bias, as is customary before a normalization layer (Ioffe & Szegedy, 2015; Ba et al., 2016) and ends with a normalization-relu-weight operation after the final residual block in the encoder, following He et al. (2016a).

4.4. Set Transformer++ (ST++)

Similarly, ST++ adds a set norm layer and adheres to the clean path principle (see Figure 2 (b)). In practice, we define a variant of the ISAB model, which we call ISAB++, that changes the residual connections and adds normalization off the residual path, analogous to the Pre-LN transformer (Klein et al., 2017; Vaswani et al., 2018; Xiong et al., 2020). We define two multi head attention blocks MAB^1 and MAB^2 with K heads as

$$\text{MAB}_K^1(\mathbf{x}, \mathbf{y}) = \mathbf{h} + \text{fcc}(\text{relu}(\text{SetNorm}(\mathbf{h}))) \quad (21)$$
$$\text{where } \mathbf{h} = \mathbf{x} + \text{Attn}_K(\mathbf{x}, \text{SetNorm}(\mathbf{y}), \mathbf{y}). \quad (22)$$

$$\text{MAB}_K^2(\mathbf{x}, \mathbf{y}) = \mathbf{h} + \text{fcc}(\text{relu}(\text{SetNorm}(\mathbf{h}))) \quad (23)$$
$$\text{where } \mathbf{h} = \mathbf{x} + \text{Attn}_K(\text{SetNorm}(\mathbf{x}), \text{SetNorm}(\mathbf{y}), \mathbf{y}). \quad (24)$$

Then, the ISAB++ block with D hidden units, K heads and M inducing points is defined as

$$\text{ISAB++}_M(\mathbf{x}) = \text{MAB}_K^2(\mathbf{x}, \mathbf{h}) \in \mathbb{R}^{S \times D}, \quad (25)$$
$$\mathbf{h} = \text{MAB}_K^1(\mathbf{p}, \mathbf{x}) \in \mathbb{R}^{M \times D}. \quad (26)$$

The reason why MAB$_K^1$ does not include normalization on the first input is because that inducing points $\mathbf{p}$ are learned.

5. FlowRBC

To complement our technical contributions, we open-source FlowRBC, a prototypical example of a clinically-available single cell blood dataset. In this type of dataset, permutation invariance holds biologically as blood cells move throughout the body. FlowRBC aims to answer an interesting physiological question: can we predict extrinsic properties from intrinsic ones? In practice, the task is to predict a patient's hematocrit levels from individual red blood cell (RBC) volume and hemoglobin measurements. Hematocrit is the fraction of overall blood volume occupied by red blood cells and thus an aggregated measure of RBCs and other blood cell types. See more details in Appendix A. FlowRBC represents an exciting real-world use case for prediction on sets largely overlooked by the machine learning community. It differs from other real-valued datasets (e.g. Point Cloud) in that every absolute measurement carries biological information beyond its relative position with other points. This implies that translations might map to different physiological states. For this reason, careful architectural design is required to preserve useful knowledge about the input.

Table 3: Clean path residual connections outperform non-clean path residual connections both in Deep Sets and Set Transformer. Clean path residuals with set norm perform best overall. Results are test loss for deep architectures (50 layers Deep Set, 16 layers Set Transformer), lower is better.

Path	Residual type	Norm	Hematocrit (MSE)	Point Cloud (CE)	Mnist Var (MSE)	Normal Var (MSE)
Deep Sets	non-clean path	layer norm	19.6649 ± 0.0394	**0.5974** ± **0.0022**	0.3528 ± 0.0063	1.4658 ± 0.7259
		feature norm	19.9801 ± 0.0862	0.6541 ± 0.0022	**0.3371** ± **0.0059**	0.8352 ± 0.3886
		set norm	19.3146 ± 0.0409	**0.6055** ± **0.0007**	**0.3421** ± **0.0022**	0.2094 ± 0.1115
	clean path	layer norm	19.4192 ± 0.0173	0.63682± 0.0067	0.3997 ± 0.0302	0.0384 ± 0.0105
		feature norm	19.3917 ± 0.0685	0.7148 ± 0.0164	**0.3368** ±**0.0049**	0.1195 ± 0.0000
		set norm	**19.2118** ± **0.0762**	0.7096 ± 0.0049	**0.3441** ±**0.0036**	**0.0198** ± **0.0041**
Set Transformer	non-clean path	layer norm	19.1975 ± 0.1395	0.9219 ± 0.0052	2.0663 ± 1.0039	0.0801 ± 0.0076
		feature norm	19.4968 ± 0.1442	0.8251 ±0.0025	**0.4043** ± **0.0078**	0.0691 ± 0.0146
		set norm	19.0521 ±0.0288	1.9167 ± 0.4880	**0.4064** ± **0.0147**	0.0249 ± 0.0112
	clean path	layer norm	**18.5747** ± **0.0263**	0.6656 ± 0.0148	0.6383 ± 0.0020	0.0104 ± 0.0000
		feature norm	19.1967± 0.0330	**0.6188** ± **0.0141**	0.7946 ±0.0065	0.0074 ± 0.0010
		set norm	18.7008 ± 0.0183	**0.6280** ± **0.0098**	0.8023 ± 0.0038	**0.0030** ± **0.0000**

6. Experimental setup

To evaluate the effect of our proposed modifications, we consider tasks with diverse inputs (point cloud, continuous, image) and outputs (regression, classification). We use four main datasets to study the individual components of our solution (Hematocrit, Point Cloud, Mnist Var and Normal Var) and two (CelebA, Anemia) for validation of the models.

- **Hematocrit Regression from Blood Cell Cytometry Data (Hematocrit a.k.a. Flow-RBC).** The dataset consists of measurements from 98240 train and 23104 test patients. We select the first visit for a given patient such that each patient only appears once in the dataset, and there is no patient overlap between train and test. We subsample for each distribution to 1,000 cells.
- **Point Cloud Classification (Point Cloud).** Following (Zaheer et al., 2017; Lee et al., 2019), we use the ModelNet40 dataset (Wu et al., 2015) (9840 train and 2468 test clouds), randomly sample 1,000 points per set, and standardize each object to have mean zero and unit variance along each coordinate axis. We report ablation results as cross entropy loss to facilitate the readability of the tables, i.e. lower is better.
- **Variance Prediction, Image Data (MNIST Var).** We implement empirical variance regression on MNIST digits as a proxy for real-world tasks with sets of images, e.g. prediction on blood smears or histopathology slides. We sample 10 images uniformly from the training set and use the empirical variance of the digits as a label. Test set and training set images are non-overlapping. Training set size is 50,000 sets, and test set size is 1,000 sets. We represent each image as a 1D vector.
- **Empirical Variance Prediction, Real Data (Normal Var).** Each set is a collection of 1000 samples from a univariate normal distribution. Means are drawn uniformly in [-10, 10], and variances are drawn uniformly in [0, 10]. The target for each set is the empirical variance of the samples (regression task) in the set. Training set size is 10,000 sets, and test set size is 1,000 sets.
- **Set anomaly detection, Image Data (CelebA).** Following Lee et al. (2019), we generate sets of images from the CelebA dataset (Liu et al., 2015) where nine images share two attributes in common while one does not. We learn an equivariant function whose output is a 10-dimensional vector that identifies the anomaly in the set. We build a train and test datasets with 18000 sets, each of them containing 10 images (64×64). Train and test do not contain the same individuals.
- **Anemia detection, Blood Cell Cytometry Data.** The dataset consists of 11136 train and 2432 test patients. Inputs are individual red blood cell measurements (volume and hemoglobin) and the outputs are a binary anemic vs. non-anemic diagnosis. A patient was considered anemic if they had a diagnosis for anemia of any type within 3 days of their blood measurements. We sample 1,000 cells for each input distribution.

Unless otherwise specified, results are reported in Mean Squared Error (MSE) for regression experiments and in cross entropy loss (CE) for point cloud classification, averaged over three seeds. We fix all hyperparameters, including epochs, and use the model at the end of training for evaluation. We notice no signs of overfitting from the loss curves. For further experimental details, see Appendix D.

7. Results

Clean path residuals have better performance than non-clean path ones. Table 3 confirms that clean path pipelines

Table 4: Equivariant residual connections perform better than aggregated residual connections in both Deep Sets and Set Transformer. Max aggregation for Set Transformer led to exploding gradient so we do not report result.

Path	Residual type	Hematocrit (MSE)	Point Cloud (CE)	Mnist Var (MSE)	Normal Var (MSE)
Deep Sets	equivariant	**19.2118 ± 0.0762**	**0.7096 ± 0.0049**	**0.3441 ± 0.0036**	**0.0198 ± 0.0041**
	mean	19.3462 ± 0.0260	0.8585 ± 0.0253	1.2808 ± 0.0101	0.8811 ± 0.1824
	max	19.8171 ±0.0266	0.8758 ±0.0196	1.3798 ±0.0162	0.8964 ± 0.1376
Set Transformer	equivariant	**18.6883 ± 0.0238**	**0.6280 ± 0.0098**	**0.7921 ± 0.0006**	**0.0030 ± 0.0000**
	mean	19.6945 ± 0.1067	0.8111 ± 0.0453	1.6273 ± 0.0335	0.0147 ± 0.0028

Table 5: While Deep Sets and Set Transformer exhibit notable failures when deep (underlined), Deep Sets++ and Set Transformer++ do not. The latter also achieve new levels of performance on a several tasks.

Model	No. Layers	Hematocrit (MSE)	MNIST Var (MSE)	Point Cloud (accuracy)	CelebA (accuracy)	Anemia (accuracy)
DeepSets	3	**19.1257 ± 0.0361**	0.4520 ±0.0111	0.7755 ± 0.0051	0.3808 ± 0.0016	0.5282 ± 0.0018
	25	20.2002 ± 0.0689	1.3492 ± 0.2801	0.3498 ± 0.0340	0.1005 ± 0.0000	0.4856 ± 0.0000
	50	25.8791± 0.0014	5.5545 ± 0.0014	0.0409 ± 0.0000	0.1005 ± 0.0000	0.4856 ± 0.0000
Deep Sets++	3	19.5882 ± 0.0555	0.5895 ± 0.0114	0.7865 ± 0.0093	0.5730 ± 0.0016	0.5256 ± 0.0019
	25	**19.1384 ± 0.1019**	0.3914 ± 0.0100	**0.8030 ± 0.0034**	**0.6021 ± 0.0072**	0.5341 ± 0.0118
	50	**19.2118 ± 0.0762**	**0.3441 ± 0.0036**	**0.8029 ± 0.0005**	**0.5763 ± 0.0134**	**0.5561 ± 0.0202**
Set Transformer	2	18.8750 ± 0.0058	0.6151 ± 0.0072	0.7774 ± 0.0076	0.1292 ± 0.0012	**0.5938 ± 0.0075**
	8	18.9095 ± 0.0271	**0.3271 ± 0.0068**	0.7848 ± 0.0061	0.4299 ± 0.1001	**0.5943 ± 0.0036**
	16	**18.7436 ± 0.0148**	6.2663 ± 0.0036	0.7134 ± 0.0030	0.4570 ± 0.0540	0.5853 ± 0.0049
Set Transformer++	2	18.9223 ± 0.0273	1.1525 ± 0.0158	0.8146 ± 0.0023	0.6533 ± 0.0012	0.5770 ± 0.0223
	8	18.8984 ± 0.0703	0.9437 ± 0.0137	**0.8247 ± 0.0020**	**0.6621 ± 0.0021**	0.5680 ± 0.0110
	16	**18.7008 ± 0.0183**	0.8023 ± 0.0038	**0.8258 ± 0.0046**	0.6587 ± 0.0001	0.5544 ± 0.0113

generally yield the best performance across set tasks both for Deep Sets and Set Transformer, independently of normalization choice. The primary exception to this trend is Deep Sets on Point Cloud, which can be explained by a Point Cloud-specific phenomenon where the repeated addition of positive values in the architecture improves performance (see Appendix E.1 for empirical analysis). Non-clean path Set Transformer has both the worst and best results on Mnist Var among Set Transformer variants, evidence of its unpredictable behavior at high depths. In contrast, ST++ results are more stable, and Table 5 illustrates that ST++ consistently improves on Mnist Var as the network depth increases.

The clean path principle has previously been shown in other applications to improve performance and yield more stable training (He et al., 2016a; Wang et al., 2019). Its benefit for both Deep Sets and Set Transformer provides further proof of the effectiveness of this principle.

Equivariant residual connections are the best choice for set-based skip connections. ERCs generalize residual connections to permutation-equivariant architectures. For further validation of their usefulness, we empirically compare them with another type of residual connection: an *aggregated residual connection* (ARC) which sums an aggregated function of the elements (e.g. sum, mean, max) from the previous layer. Appendix F provides a more detailed discussion. Results in Table 4 show that clean-path ERCs remain the most suitable choice.

Set norm performs better than other norms. Table 2 shows that Deep Sets benefits from the addition of set norm when no residual connections are involved. Hematocrit and Normal Var performances are the same across normalizations, but this is due to a vanishing gradient that cannot be overcome by the presence of normalization layers alone.

We further analyzed normalizations in the presence of residual connections in Table 3. Here, we also consider the normalization layer used in the PointNet and PointNet++ architectures for point cloud classification (Qi et al., 2017a;b), implemented as batch norm on a transposed tensor. We call this norm *feature norm*, which is an example of a normalization that occurs over the batch rather than on a per-set basis ($\mathcal{S} = \{D\}, \mathcal{T} = \{D\}$).

Clean path residuals with set norm generally perform best. The pattern is particularly evident for Normal Var, where clean path is significantly better than non-clean path and the addition of set norm further improves the performance.

We additionally observe in Table 3 that results for layer norm improve with the addition of clean-path residual connections relative to earlier results in Table 1 and Table 2. We hypothesize that skip connections help alleviate information loss from normalization by passing forward values before normalization. For instance, given two elements $\mathbf{x}_l$ and $\mathbf{x}'_l$ that will be mapped to the same output $\hat{\mathbf{x}}$ by layer norm, adding a residual connection enables the samples to have distinct outputs $\mathbf{x}_{l+1} = \mathbf{x}_l + \hat{\mathbf{x}}$ and $\mathbf{x}'_{l+1} = \mathbf{x}'_l + \hat{\mathbf{x}}$.

Deep Sets++ and Set Transformer++ outperform existing architectures. We validate our proposed models DS++ and ST++ on real-world datasets (Table 5). Deep Sets (DS) and Set Transformer (ST) show failures (underlined entries) as depth increases. On the contrary, DS++ and ST++ tend to outperform their original and shallow counterparts at high depths (rows highlighted in gray have the highest number of best results). Deep Sets++ and Set Transformer++ particularly improve performance on point cloud classification and CelebA set anomaly detection. We show in Appendix E that, on an official point cloud benchmark repository (Goyal et al., 2021a), DS++ and ST++ without any modifications outperform versions of Deep Sets and Set Transformer tailored for point cloud classification. On Hematocrit, both deep modified models surpass the clinical baseline (25.85 MSE) while the original Deep Sets at 50 layers does not (more details are provided in Appendix A).

Table 5 highlights that DS++ and ST++ generally improve over DS and ST overall without notable failures as depth increases. Due to their reliability and ease of use, DS++ and ST++ are practical choices for practitioners who wish to avoid extensive model search or task-specific engineering when approaching a new task, particularly one involving sets of measurements or images. We expect this benefit to be increasingly relevant in healthcare or biomedical settings, as new datasets of single cell measurements and cell slides continue to be generated, and new tasks and research questions continue to be posed.

Lastly, while ST and ST++ performance are better than DS++, it is worth noticing that the former models have approximately 3 times more parameters and take more time and memory to run. As an example, on point cloud classification, ST++ took ≈ 2 times longer to train than DS++ for the same number of steps on a NVIDIA Titan RTX.

8. Related Work

Previous efforts to design residual connections (He et al., 2016b; Veit et al., 2016; Yao et al., 2020) or normalization layers (Ioffe & Szegedy, 2015; Ba et al., 2016; Santurkar et al., 2018; Ghorbani et al., 2019; Luo et al., 2019; Xiong et al., 2020; Cai et al., 2021) have often been motivated by particular applications. Our work is motivated by applications of predictions on sets.

The effects of non-clean or clean path residual connections have been studied in various settings. He et al. (2016a) showed that adding a learned scalar weight to the residual connection, i.e. $\mathbf{x}_{\ell+1} = \lambda_\ell \mathbf{x}_\ell + \mathcal{F}(\mathbf{x}_\ell)$, can result in vanishing or exploding gradients if the λ scalars are consistently large (e.g. > 1) or small (< 1). Wang et al. (2019) while see that for deep transformers, only the clean path variant converges during training. Xiong et al. (2020) show that Post-LN transformers (non-clean path) require careful learning rate scheduling unlike their Pre-LN (clean path) counterparts. Our analysis provides further evidence of the benefit of clean-path residuals. While our clean and non-clean path DS architectures mirror those of the clean and non-clean path ResNet architectures (He et al., 2016a;b), the non-clean path Set Transformer differs from non-clean path Post-LN Transformer in that the former also has a linear projection on the residual path.

Many normalization layers have been designed for specific purposes. For instance, batch norm (De & Smith, 2020), layer norm (Ba et al., 2016), instance norm (Ulyanov et al., 2016) and graph norm (Cai et al., 2021) were designed for image, text, stylization, and graphs respectively. In this work, we propose set norm with set inputs in mind, particularly sets of real-valued inputs. The idea in set norm to address different samples being mapped to the same outputs from layer norm is reminiscent of the goal to avoid oversmoothing motivating pair norm (Zhao & Akoglu, 2019), developed for graph neural networks.

Our work offers parallels with work on graph convolutional networks (GCNs). For instance, previous works in the GCN literature have designed architectures that behave well when deep and leverage residual connections (Li et al., 2019; Chen et al., 2020). However, while GCNs and set-based architectures share a lot of common principles, the former relies on external information about the graph structure which is not present in the latter.

9. Conclusion

We illustrate limitations of Deep Sets and Set Transformer when deep and develop Deep Sets++ and Set Transformer++ to overcome these limitations. We introduce set norm to address the unwanted invariance of layer norm for real-valued sets, and we employ clean-path equivariant residual connections to enable identity mappings and help address gradient issues. DS++ and ST++ are general-purpose architectures and the first permutation invariant architectures of their depth that show good performance on a variety of tasks. We also introduce Flow-RBC, a new open-source dataset which provides a real-world application of permutation invariant prediction in clinical science. We believe our new models and dataset have the potential to motivate future work and applications of prediction on sets.

References

Ba, J., Kiros, J., and Hinton, G. E. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Cai, T., Luo, S., Xu, K., He, D., Liu, T.-Y., and Wang, L. Graphnorm: A principled approach to accelerating graph neural network training. In *ICML*, 2021.

Chen, M., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G. F., Jones, L., Parmar, N., Schuster, M., Chen, Z., Wu, Y., and Hughes, M. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*, 2018.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

De, S. and Smith, S. L. Batch normalization biases residual blocks towards the identity function in deep networks. *arXiv: Learning*, 2020.

Ghorbani, B., Krishnan, S., and Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In *ICML*, 2019.

Goyal, A., Law, H., Liu, B., Newell, A., and Deng, J. Revisiting point cloud shape classification with a simple and effective baseline. In *ICML*, 2021a.

Goyal, A., Law, H., Liu, B., Newell, A., and Deng, J. Revisiting point cloud shape classification with a simple and effective baseline. *International Conference on Machine Learning*, 2021b.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2020.

Harris, N., Kunicka, J., and Kratz, A. The advia 2120 hematology system: flow cytometry-based analysis of blood and body fluids in the routine hematology laboratory. *Laboratory Hematology*, 11(1):47–61, 2005.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. *ArXiv*, abs/1603.05027, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016b.

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

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. Opennmt: Open-source toolkit for neural machine translation. *ArXiv*, abs/1701.02810, 2017.

Lähnemann, D., Köster, J., Szczurek, E., McCarthy, D. J., Hicks, S. C., Robinson, M. D., Vallejos, C. A., Campbell, K. R., Beerenwinkel, N., Mahfouz, A., Pinello, L., Skums, P., Stamatakis, A., Attolini, C. S.-O., Aparicio, S., Baaijens, J. A., Balvert, M., de Barbanson, B., Cappuccio, A., Corleone, G., Dutilh, B. E., Florescu, M., Guryev, V., Holmer, R., Jahn, K., Lobo, T. J., Keizer, E. M., Khatri, I., Kiełbasa, S. M., Korbel, J. O., Kozlov, A. M., Kuo, T.-H., Lelieveldt, B. P. F., Măndoiu, I. I., Marioni, J. C., Marschall, T., Mölder, F., Niknejad, A., Raczkowski, L., Reinders, M. J. T., de Ridder, J., Saliba, A.-E., Somarakis, A., Stegle, O., Theis, F. J., Yang, H., Zelikovsky, A., McHardy, A. C., Raphael, B. J., Shah, S. P., and Schönhuth, A. Eleven grand challenges in single-cell data science. *Genome Biology*, 21, 2020.

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

Li, G., Muller, M., Thabet, A., and Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9267–9276, 2019.

Liu, J., Fan, Z., Zhao, W., and Zhou, X. Machine intelligence in single-cell data analysis: Advances and new challenges. *Frontiers in Genetics*, 12, 2021.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Luo, P., Wang, X., Shao, W., and Peng, Z. Towards understanding regularization in batch normalization. *ArXiv*, abs/1809.00846, 2019.

McPherson, R. A., Msc, M., and Pincus, M. R. *Henry's clinical diagnosis and management by laboratory methods E-book*. Elsevier Health Sciences, 2021.

Qi, C., Su, H., Mo, K., and Guibas, L. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017a.

Qi, C., Yi, L., Su, H., and Guibas, L. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017b.

Regev, A., Teichmann, S. A., Lander, E. S., Amit, I., Benoist, C., Birney, E., Bodenmiller, B., Campbell, P., Carninci, P., Clatworthy, M., et al. Science forum: the human cell atlas. *elife*, 6:e27041, 2017.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In *NeurIPS*, 2018.

Tycko, D., Metz, M., Epstein, E., and Grinbaum, A. Flow-cytometric light scattering measurement of red blood cell volume and hemoglobin concentration. *Applied optics*, 24(9):1355–1365, 1985.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *ArXiv*, abs/1607.08022, 2016.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A. N., Gouws, S., Jones, L., Kaiser, L., Kalchbrenner, N., Parmar, N., Sepassi, R., Shazeer, N. M., and Uszkoreit, J. Tensor2tensor for neural machine translation. In *AMTA*, 2018.

Veit, A., Wilber, M. J., and Belongie, S. J. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, 2016.

Wagstaff, E., Fuchs, F., Engelcke, M., Posner, I., and Osborne, M. A. On the limitations of representing functions on sets. In *ICML*, 2019.

Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D. F., and Chao, L. S. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.

Wang, R., Walters, R., and Yu, R. Incorporating symmetry into deep dynamics models for improved generalization. *arXiv preprint arXiv:2002.03061*, 2020.

Weiler, M. and Cesa, G. General e (2)-equivariant steerable cnns. *Advances in Neural Information Processing Systems*, 32, 2019.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, 2015.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. On layer normalization in the transformer architecture. *ArXiv*, abs/2002.04745, 2020.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. A mean field theory of batch normalization. *ArXiv*, abs/1902.08129, 2019.

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Pyhessian: Neural networks through the lens of the hessian. *2020 IEEE International Conference on Big Data (Big Data)*, pp. 581–590, 2020.

Yuan, G., Cai, L., Elowitz, M. B., Enver, T., Fan, G., Guo, G., Irizarry, R. A., Kharchenko, P. V., Kim, J., Orkin, S. H., Quackenbush, J., Saadatpour, A., Schroeder, T., Shivdasani, R. A., and Tirosh, I. Challenges and emerging directions in single-cell analysis. *Genome Biology*, 18, 2017.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. Deep sets. In *NeurIPS*, 2017.

Zhang, H., Dauphin, Y. N., and Ma, T. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*, 2018.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2019.

A. Flow-RBC

The analysis of and prediction from single-cell data is an area of rapid growth (Lähnemann et al., 2020). Even so, Flow-RBC constitutes a dataset unique for its kind, consisting of more than 100,000 measurements taken on different patients paired with a clinical label. Even established projects like the Human Cell Atlas (Regev et al., 2017) or Flow Repository[4] do not include single-cell datasets of this size. For instance, to our knowledge, the second largest open-source dataset of single-cell blood samples contains data from 2,000 individuals and does not include external clinical outcomes for all patients to be used as a target.

Flow-RBC consists of 98,240 train and 23,104 test examples. Each input set is a red blood cell (RBC) distribution of 1,000 cells. Each cell consists of a volume and hemoglobin content measurement (see Figure 3 for a visual representation). The regression task consists of predicting the corresponding hematocrit level measured on the same blood sample. Blood consists of different components: red blood cells, white blood cells, platelets and plasma. The hematocrit level measures the percentage of volume taken up by red blood cells in a blood sample.

Since we only have information about the volume and hemoglobin of individual RBCs and no information about other blood cells, this task aims to answer an interesting clinical question: is there information present in individual RBC volume and hemoglobin measurements about the overall volume of RBCs in the blood? As this question has not been definitively answered in the literature, there is no known expected performance achievable; instead, increases in performance are an exciting scientific signal, suggesting a stronger relationship between single cell RBC and aggregate population properties of the human blood than previously known.

The existing scientific literature notes that in the presence of diseases like anemia, there exists a negative correlation between hematocrit and the red cell distribution width (RDW), also known as the coefficient of variation of the volume i.e. SD(Volume) / Mean(Volume) $\times$ 100 (McPherson et al., 2021, Chapter 9). To represent the current state of medical knowledge on this topic, we use as a baseline a linear regression model with RDW as covariate. Additionally, we build a regression model on hand-crafted distribution statistics (up to the fourth moment on both marginal distributions as well as .1, .25, .5, .75, .9 quantiles). This model improves over simple prediction with RDW, further confirming the hypothesis that more information lies in the single-cell measurements of RBCs. ST++ further improves performance, resulting in an MSE reduction of 28% over the RDW model. See Table 6 for results.

[4]https://flowrepository.org

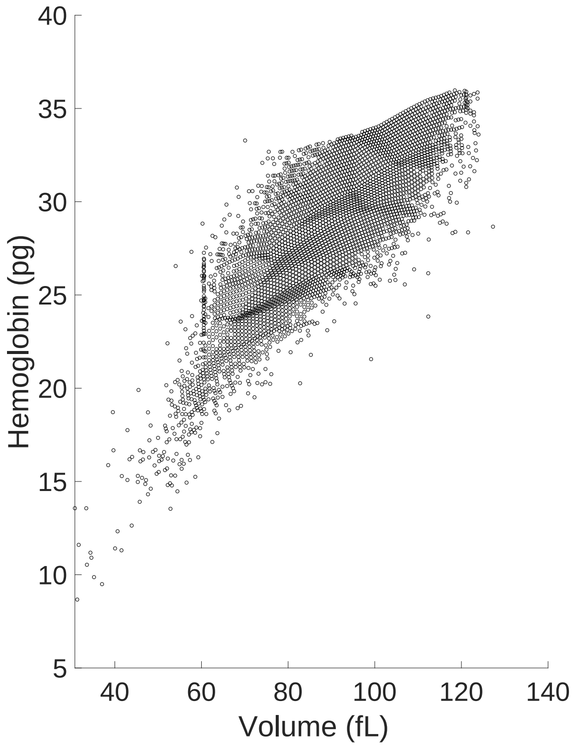


Figure 3: Example of RBC distribution given in input for the prediction of hematocrit level.

Procedure to Obtain RBC distribution measurements All Flow-RBC data is collected retrospectively at Massachusetts General Hospital under an existing IRB-approved research protocol and is available at this link. Each RBC distribution consists of volume and hemoglobin mass measurements collected using the Advia 2120 (Harris et al., 2005), a flow-cytometry based system that measures thousands of cells. The volume and hemoglobin information are retrieved through Mie (or Lorenz-Mie) theory equations for the analysis of light scattering from a homogeneous spherical particle (Tycko et al., 1985). An example of one input distribution is provided in Figure 3. The Advia machine returns an average of 55,000 cells. For this dataset, we downsampled each distribution to 1,000 cells, a number high enough to maintain sample estimates of "population" (i.e. all 55,000 cells) statistics with minimal variance while imposing reasonable memory requirements on consumer gpus. Each distribution is normalized and re-scaled by the training set mean and standard deviation.

Table 6: Baseline regression performances for the prediction of hematocrit from RBC distributions. Our proposed Set Transformer++ currently has the best performance on this task.

	MSE
RDW	25.85
Moments	22.31
Set Transformer++	18.69

B. Layer Norm Analyses

B.1. Gradient Analysis for Set Transformer with Layer Norm

The addition of Layer norm to Equation (7) does not preclude the possibility of exploding or vanishing gradients. Let Attn_K be multihead attention with K heads and a scaled softmax, *i.e.* $\text{softmax}(\cdot/\sqrt{D})$, and let LN be layer norm. We consider the following definition of a MAB module, with layer norm placement that matches what was described in the original paper (Lee et al., 2019):

$$\text{MAB}_K(\mathbf{x}, \mathbf{y}) = \text{LN}(f(\mathbf{x}, \mathbf{y}) + \text{relu}(f(\mathbf{x}, \mathbf{y})W + \mathbf{b})), \tag{27}$$

$$f(\mathbf{x}, \mathbf{y}) = \text{LN}(\mathbf{x}W_Q + \text{Attn}_K(\mathbf{x}, \mathbf{y}, \mathbf{y})). \tag{28}$$

The inducing point set attention block (ISAB) is then

$$ISAB_M(\mathbf{x}) = \text{MAB}_K(\mathbf{x}, \mathbf{h}) \in \mathbb{R}^{S\times D} \tag{29}$$

$$\text{where } \mathbf{h} = \text{MAB}_K(\mathbf{p}, \mathbf{x}) \in \mathbb{R}^{M\times D}. \tag{30}$$

Consider a single ISAB module. We let $\mathbf{z}_1$ denote the output of the previous block, $\mathbf{z}_2$ denote the output after the first MAB module (i.e. $\mathbf{h}$ in Equation (9)), and $\mathbf{z}_3$ denote the output of the second MAB module, or the overall output of the ISAB module. Then,

$$f_1 = f(\mathbf{p}, \mathbf{z}_1) = \text{LN}(IW_{1Q} + \text{Attn}_K(\mathbf{p}, \mathbf{z}_1, \mathbf{z}_1)) \tag{31}$$

$$\mathbf{z}_2 = \text{LN}(f_1 + \text{relu}(f_1W_1 + \mathbf{b}_1)) \tag{32}$$

$$f_2 = f(\mathbf{z}_1, \mathbf{z}_2) = \mathbf{z}_1W_{2Q} + \text{Attn}_K(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_2)) \tag{33}$$

$$f_3 = \text{LN}(f_2) \tag{34}$$

$$f_4 = f_3 + \text{relu}(f_3W_2 + \mathbf{b}_2) \tag{35}$$

$$\mathbf{z}_3 = \text{LN}(f_4). \tag{36}$$

The gradient of a single ISAB block output $\mathbf{z}_3$ with respect to its input $\mathbf{z}_1$ can be represented as $\frac{\partial \mathbf{z}_3}{\partial \mathbf{z}_1} = \frac{\partial \mathbf{z}_3}{\partial f_4}\frac{\partial f_4}{\partial f_3}\frac{\partial f_3}{\partial f_2}\frac{\partial f_2}{\partial \mathbf{z}_1}$, or

$$\frac{\partial \text{LN}(f_4)}{\partial f_4}\Big(I + \frac{\partial \text{relu}(f_2W_2 + \mathbf{b}_2)}{\partial (f_2W_2 + \mathbf{b}_2)}W_2\Big)$$
$$\frac{\partial \text{LN}(f_2)}{\partial f_2}\Big(W_{2Q} + \frac{\partial \text{Attn}_K(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_2)}{\partial \mathbf{z}_1}\Big).$$

The gradient expression is analogous to the one in Section 3.2, with the exception of additional $\frac{\partial \text{LN}(f_4)}{\partial f_4}$ and $\frac{\partial \text{LN}(f_2)}{\partial f_2}$ per ISAB block. With many ISAB blocks, it is still possible for a product of the weights W_{2Q} to accumulate.

B.2. Visualizing Layer Norm Example in 2D

In Section 3.3, we discussed how layer norm removes two degrees of freedom from each sample in a set, which can make certain prediction difficult or impossible. In particular, we discussed a simple toy example in 2D, that of classifying shapes based on 2D point clouds. We utilize hidden layers of size 2, which means the resulting activations can be visualized. In this setup, different shapes yield the same resulting activations as long as their points are equally distributed above and below the $y = x$ line. See Figure 4.

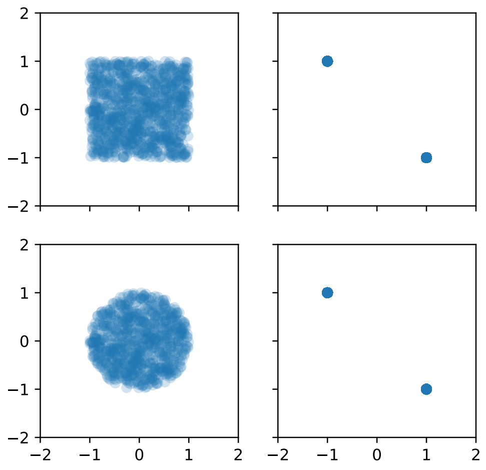


Figure 4: Layer norm performs per-sample standardization, which in 2D point cloud classification can result in shapes (left) whose 2D activations (right) are indistinguishable from each other.

C. Normalization proofs

Proposition 1. *Let $\mathcal{F}$ be the family of transformation functions which can be expressed via Equation (20). Then, for $f \in \mathcal{F}$, $\mathcal{T} = \{D\}$ and $\mathcal{T} = \{\}$ are the only settings satisfying the following properties:*

1. *$f_{\mathcal{T}}(\pi_i \boldsymbol{a}) = \pi_i f_{\mathcal{T}}(\boldsymbol{a})$ where π_i is a permutation function that operates on elements in the set;*

2. *$f_{\mathcal{T}}(\pi_n \boldsymbol{a}) = \pi_n f_{\mathcal{T}}(\boldsymbol{a})$ where π_n is a permutation function that operates on sets.*

Proof. For transformation tensors in $\mathbb{R}^{N\times M\times D}$, the parameters can be distinct over the batch ($N \in \mathcal{T}$), over the elements ($M \in \mathcal{T}$), over the features ($D \in \mathcal{T}$), or any combination of the three. We show that $N \in \mathcal{T}$ and $M \in \mathcal{T}$ are unsuitable, leaving only $D \in \mathcal{T}$.

Having distinct parameters over the samples breaks permutation equivariance, making $M \in \mathcal{T}$ an untenable option. Let $f : \mathbb{R}^{M\times D} \to \mathbb{R}^{M\times D}$ be the transformation function, and $\overrightarrow{\gamma}_{\{M\}}$, $\overrightarrow{\beta}_{\{M\}}$ represent tensors in $\mathbb{R}^{N\times M\times D}$ where the values along dimension M can be unique, while the values along N, D are repeated. We denote an indexing into the batch dimension as $\overrightarrow{\gamma}_{\{M\},n}$, $\overrightarrow{\beta}_{\{M\},n}$. Then, f breaks permutation equivariance:

$$\begin{aligned} f(\pi_i \mathbf{a}) &= \pi_i \mathbf{a} \odot \overrightarrow{\gamma}_{\{M\},n} + \overrightarrow{\beta}_{\{M\},n} && (37)\\ &\neq \pi_i(\mathbf{a} \odot \overrightarrow{\gamma}_{\{M\},n} + \overrightarrow{\beta}_{\{M\},n}) && (38)\\ &= \pi_i f(\mathbf{a}). && (39)\end{aligned}$$

Having distinct parameters over the batch means that the position of a set in the batch changes its ordering, making $N \in \mathcal{T}$ an untenable option. Let $\overrightarrow{\gamma}_{\{N\}}$, $\overrightarrow{\beta}_{\{N\}}$ represent tensors which can differ over the batch, e.g. $\overrightarrow{\gamma}_{\{N\},n} \neq \overrightarrow{\gamma}_{\{N\},n'}, n \neq n'$. Then, the prediction function f_n for batch index n will yield a different output than the prediction function $f_{n'}$ for batch index n':

$$\begin{aligned} f_b(\mathbf{a}) &= \mathbf{a} \odot \overrightarrow{\gamma}_{\{N\},n} + \overrightarrow{\beta}_{\{N\},n} && (40)\\ &\neq \mathbf{a} \odot \overrightarrow{\gamma}_{\{N\},n'} + \overrightarrow{\beta}_{\{N\},n'} && (41)\\ &= f_{n'}(\mathbf{a}). && (42)\end{aligned}$$

As neither M nor N can be in $\mathcal{T}$, the remaining options are $\mathcal{T} = \{D\}$ or $\mathcal{T} = \{\}$, i.e. $\overrightarrow{\gamma}, \overrightarrow{\beta}$ each repeat a single value across the tensor. Note that $\mathcal{T} = \{\}$ is strictly contained in $\mathcal{T} = \{D\}$: if the per feature parameters are set to be equal in the $\mathcal{T} = \{D\}$ setting, the result is equivalent to $\mathcal{T} = \{\}$. Therefore, $\mathcal{T} = \{D\}$ sufficiently describes the only suitable setting of parameters for transformation.

□

Proposition 2. *Set norm is permutation equivariant.*

Proof. Let $\mu, \sigma \in \mathbb{R}$ be the elements mean and variance over all features in the set, $\overrightarrow{\gamma}, \overrightarrow{\beta} \in \mathbb{R}^{M\times D}$ refer to the appropriate repetition of per-feature parameters in the M dimension. Then,

$$\begin{aligned} \mathrm{SN}(\pi\mathbf{x}) &= \frac{\pi\mathbf{x} - \mu}{\sigma} * \overrightarrow{\gamma} + \overrightarrow{\beta}] && (43)\\ &= \pi\Big[\frac{\mathbf{x} - \mu}{\sigma} * \overrightarrow{\gamma} + \overrightarrow{\beta}\Big] && (44)\\ &= \pi\mathrm{SN}(\mathbf{x}). && (45)\end{aligned}$$

Equation (44) follows from the fact that μ, σ are scalars and $\overrightarrow{\gamma}, \overrightarrow{\beta}$ are equivalent for every sample in the set. □

D. Experimental configuration

Across experiments and models we purposefully keep hyperparameters consistent to illustrate the easy-to-use nature of our proposed models. All experiments and models are implemented in PyTorch. The code is available at https://github.com/rajesh-lab/deep_permutation_invariant.

D.1. Experimental Setup

Hematocrit, Point Cloud and Normal Var use a fixed sample size of 1000. MNIST Var and CelebA use a sample size of 10 due to the high-dimensionality of the images in input. The only architectural difference across these experiments is the choice of permutation-invariant aggregation for the Deep Sets architecture: we use sum aggregation for all experiments except Point Cloud, where we use max aggregation, following (Zaheer et al., 2017). We additionally use a featurizer of convolutional layers for the architectures on CelebA given the larger image sizes in this task (see Appendix D.2 section for details).

All models are trained with a batch size of 64 for 50 epochs, except for Hematocrit where we train for 30 given the much larger size of the training dataset (i.e. 90k vs. $\leq$ 10k). All results are reported as test MSE (or cross entropy for point cloud) at the last epoch. We did not use early stopping and simply took the model at the end of training. There was no sign of overfitting. Results are reported setting seeds 0, 1, and 2 for initialization weights. We use the Adam optimizer with learning rate 1e-4 throughout.

D.2. Convolutional blocks for set anomaly

For our set anomaly task on CelebA, similarly to Zaheer et al. (2017), we add at the beginning of all the considered architectures 9 convolutional layers with 3×3 filters. Specifically, we start with 2D convolutional layers with 32, 32, 64 feature-maps followed by max pooling; we follow those with 2D convolutional layers with 64, 64, 128 feature maps followed by another max pooling; and weend with 128, 128, 256 2D convolutional layers followed by a max-pooling layer with size 5. The output of the featurizer (and input to the rest of the permutation invariant model) is 255 features. The architecture is otherwise the same as those used on all other tasks considered in this work.

Table 7: Detailed DeepSets more residuals architecture.

Encoder		Aggregation	Decoder
Residual block $\times 51$			

FC(128)
SetNorm(128)
Addition
ReLU
	FC(128)	Sum/Max	FC(128)
			ReLU
			FC(128)
			ReLU
			FC(128)
			ReLU
			FC(n_outputs)

E. Additional results

E.1. Understanding ResNet vs. He Pipeline for Deep Sets on Point Cloud.

We explore why Deep Sets with the non-clean ResNet residual pipeline performs better on Point Cloud than Deep Sets with the clean He residual pipeline. Specifically, to test whether the difference is due to the ReLU activation in between connections, we design another residual pipeline where the connections (i.e. additions) are more frequent and also separated by a ReLU nonlinearity. We call this pipeline FreqAdd. This new architecture is shown in Table 7 and comparison of loss curves is in Figure 5 where we can observe that the architecture with more residual connection FreqAdd has even better performances than the non-clean pipeline. We speculate that this might be due to peculiarities of Point Cloud which benefit from continual addition positive values. Indeed, in the original Deep Sets paper (Zaheer et al., 2017), the authors add a ReLU to the end of the encoder for the architecture tailored to point cloud classification, and such a nonlinearity is noticeably missing from the model used for any other task.

E.2. Comparing Point Cloud classification with Task-Specific Models

Here, we compare the performances of DS++ and ST++ unmodified with those of models built specifically for point cloud classification. For a fair comparison, we use the experimental setup and the code provided in SimpleView (Goyal et al., 2021b). In practice, we use their DGCNN-smooth protocol and record the test accuracy at 160 epochs. The sample size for this experiment is the default in the SimpleView repository, 1024. We compared Deep Sets++, Set Transformer++, PointNet++ (Qi et al., 2017b), and SimpleView (Goyal et al., 2021a), as well as the models proposed in the original Deep Sets and Set Transformer papers tailored to point cloud classification, which differ than from the baseline architectures used in our main results. We describe these tailored Deep Sets and Set Transformer models in Table 8 and Table 9.

Results are reported in Table 10 and Figure 6. Deep Sets++ and Set Transformer++ without any modifications both achieve a higher test accuracy than the Deep Sets and Set Transformer models tailor designed for the task. PointNet++ and SimpleView perform best, but both architectures are designed specifically for point cloud classification rather than tasks on sets in general. Concretely, PointNet++ hierarchically assigns each point to centroids using Euclidean distance which is not an informative metric for high-dimensional inputs, e.g. sets of images. SimpleView is a non-permutation invariant architecture that represents each point cloud by 2D projections at various angles; such a procedure is ill-suited for sets where samples do not represent points in space.

Table 8: Customized Deep Sets architecture for PointCloud.

Encoder	Aggregation	Decoder

x - max(x)
FC(256)
Tanh
x-max(x)
FC(256)
Tanh
x-max(x)
FC(256)
Tanh
	Max	Dropout(0.5)
		FC(256)
		Tanh
		Dropout(0.5)
		FC(n_outputs)

Table 9: Customized Set Transformer architecture for PointCloud.

Encoder	Aggregation	Decoder

FC(128)
ISAB(128, 4, 32)
ISAB(128, 4, 32)
	Dropout(0.5)
	PMA(128, 4)
		Dropout(0.5)
		FC(n_outputs)

Table 10: Point cloud test accuracy

Model	Accuracy
Deep Sets	0.86
Deep Sets++	0.87
Set Transformer	0.86
Set Transformer++	0.87
SimpleView	0.92
PointNet++	0.92

F. Aggregated residual connections (ARCs)

A function g which adds aggregated equivariant residual connections to any equivariant function f is also permutation equivariant:

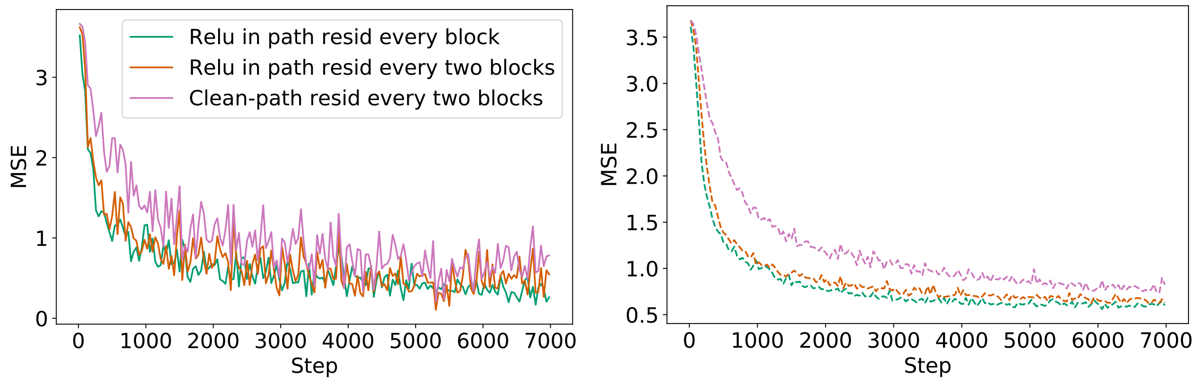


Figure 5: Loss curves for train (left) and test (right) comparing the residual pipelines ResNet (orange), He (magenta) and FreqAdd (green). Adding a positive number more frequently (green $>$ orange $>$ magenta) results in better performance for Point Cloud.

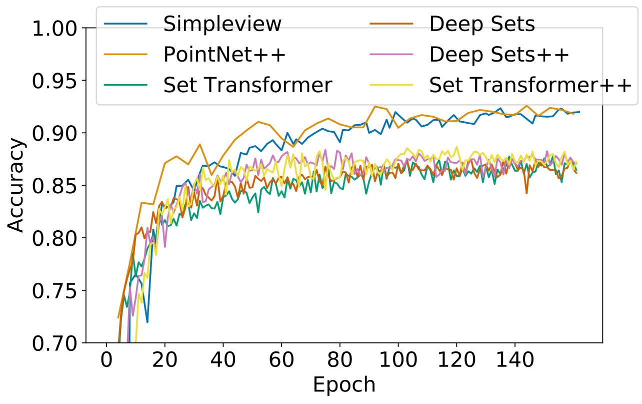


Figure 6: Test accuracy curves of different architectures on Point Cloud classification, as implemented in the SimpleView codebase. Unmodified DS++ and ST++ outperform DS and ST tailored to the task.

$$\begin{aligned} g(\pi \mathbf{x}) &= f(\pi \mathbf{x}) + \text{pool}(\mathbf{x}_1, \dots, \mathbf{x}_S) \\ &= \pi f(\mathbf{x}) + \text{pool}(\mathbf{x}_1, \dots, \mathbf{x}_S) = \pi g(\mathbf{x}). \end{aligned}$$

Results in Table 4 clearly show that clean path ARCs perform worse than clean path ERCs (Table 3).

G. Gradient Computation for Equivariant Residual Connections

We compute the gradients for early weights in a Deep Sets network with equivariant residual connections below.

We denote a single set $\mathbf{x}$ with its samples $\mathbf{x}_1, ...\mathbf{x}_M$. We denote hidden layer activations as $\mathbf{z}_{\ell,i}$ for layer ℓ and sample s. In the case of no residual connection, $\mathbf{z}_{\ell,i} = \text{ReLU}(\mathbf{z}_{\ell-1,m} W_\ell + b_\ell)$. We denote the output after an L-layer encoder and permutation invariant aggregation as $\mathbf{y} = \sum_i \mathbf{z}_{L,i}$ (we use sum for illustration but note that our conclusions are the same also for max). For simplicity let the hidden dimension remain constant throughout the encoder.

Now, we can write the gradient of weight matrix of the first layer W_1 as follows:

$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial W_1} \sum_i \frac{\partial \mathbf{y}}{\partial \mathbf{z}_{L,i}} \frac{\partial \mathbf{z}_{L,i}}{\partial W_1}. \quad (46)$$

Equivariant residual connections prevent vanishing gradients by passing forward the result of the previous computation along that sample's path, i.e. $\mathbf{z}_{\ell,i} = \text{ReLu}(\mathbf{z}_{\ell-1,i} W_\ell + b_\ell) + \mathbf{z}_{\ell-1,i}$:

$$\begin{aligned} \frac{\partial \mathbf{z}_{L,i}}{\partial \mathbf{z}_{1,i}} &= \prod_{\ell=2}^{L} \frac{\partial \mathbf{z}_{\ell,i}}{\partial \mathbf{z}_{\ell-1,i}} && (47) \\ &= \prod_{\ell=2}^{L} \frac{\partial \text{ReLU}(\mathbf{z}_{\ell,i})}{\partial \mathbf{z}_{\ell,i}} (1 + W_\ell). && (48) \end{aligned}$$