SET NORM AND EQUIVARIANT SKIP CONNECTIONS: PUTTING THE DEEP IN DEEP SETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Permutation invariant neural networks are a promising tool for predictive modeling of set data. We show, however, that existing architectures struggle to perform well when they are deep. In this work, we address this issue for the two most widely used permutation invariant networks, Deep Sets and its transformer analogue Set Transformer. We take inspiration from previous efforts to scale neural network architectures by incorporating normalization layers and skip connections that work for sets. First, we motivate and develop set norm (SN), a normalization tailored for sets. Then, we employ equivariant residual connections and introduce the "clean path principle" for their placement. With these changes, our many-layer Deep Sets++ and Set Transformer++ models reach comparable or better performance than their original counterparts on a diverse suite of tasks, from point cloud classification to regression on sets of images. We additionally introduce Flow-RBC, a new single-cell dataset and real-world application of permutation invariant prediction. On this task, our new models outperform existing methods as well as a clinical baseline. We open-source our data and code here: link-omitted-for-anonymity.

1 INTRODUCTION

Many real-world tasks involve predictions on sets or multisets. One such example prevalent in the machine learning literature is point cloud classification (Guo et al., 2020; Wu et al., 2015; Qi et al., 2017a). Another exciting application is 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).

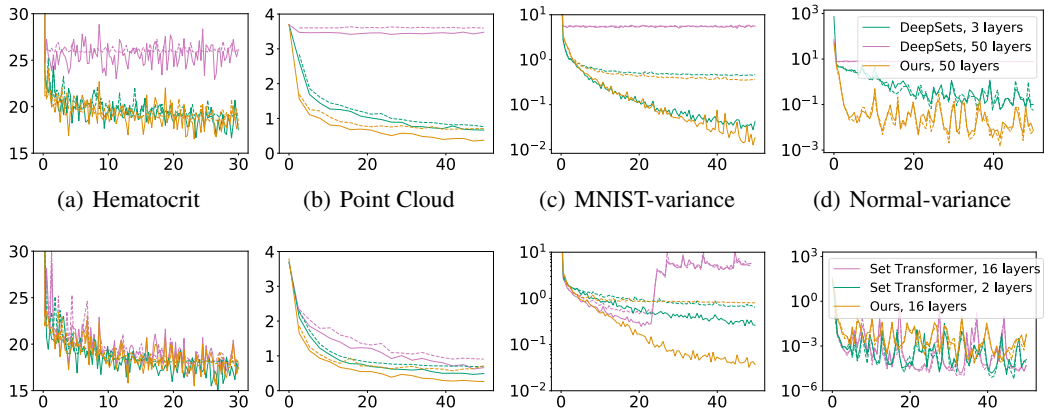
For prediction tasks which operate on sets, the index or ordering of the elements in the set does not matter for the prediction. Accordingly, models applied to this type of data should be *permutation invariant*: for any permutation of the instances in the input set, the model prediction should stay the same. For example, the order in which white blood cells or red blood cells are measured is not informative for predicting leukemia or anemia, respectively.

Examples of general-purpose deep permutation invariant architectures are Deep Sets (Zaheer et al., 2017) and Set Transformer (Lee et al., 2019). Such models are state-of-the-art architectures and have been proven to be universal approximators of permutation invariant functions under the right conditions (Zaheer et al., 2017; Wagstaff et al., 2019). However, we find that for both models, training and test error do not improve when we scale shallow architectures to larger depths (see Figure 1). Thus, the potential performance benefits of deeper models have yet to be realized.

In this work, we tackle the issue of making effective deep permutation invariant models. Guided by previous works which scale existing architectures by employing residual connections and normalization layers (He et al., 2016b;a; Li et al., 2019; Wang et al., 2019), we develop Deep Sets++ and Set Transformer++, two deep permutation invariant models which utilize residual connections and normalization layers specific for sets. Both models perform well across a wide variety of settings without the need for additional hyperparameter tuning or architectural modifications.[1]

Additionally, given rapidly growing interest in single-cell analysis (Regev et al., 2017) and the need for methods which can reliably extract signal from this data (Lähnemann et al., 2020), we introduce a

[1] In contrast, both the DeepSets and Set Transformer papers modify their architectures and experimental set up for each task.



(a) Hematocrit (b) Point Cloud (c) MNIST-variance (d) Normal-variance

(e) Hematocrit (f) Point Cloud (g) MNIST-variance (h) Normal-variance

Figure 1: Existing permutation invariant architectures (Deep Sets, Set Transformer) do not perform well when they are deep (grey). With our changes (orange), we can consistently train deep models to match or outperform existing counterparts (blue, grey). Solid lines are training curves, and dashed lines are test curves.

new single-cell dataset, one of the largest of its kind, to facilitate further research on deep permutation invariant models targeted to single-cell data, an endeavor in turn may yield new insights for biology and human health.

Our Contributions:

- We present desiderata for **normalization layers for set inputs**, motivated by the needs of real-world tasks. We introduce **set norm**, aimed at reducing drawbacks of existing normalizations (Section 3). We empirically demonstrate the benefit of set norm with ablation studies (Section 6).
- We introduce **equivariant residual connections** and the **"clean residual path" principle** for building equivariant residual pipelines. We show that they can not only improve optimization but also alleviate drawbacks incurred by normalization layers (Section 4).
- We introduce a **novel single-cell clinical dataset for prediction on sets** called **Flow-RBC**. The dataset consists of red blood cell (RBC) measurements and hematocrit levels of over 100,000 patients. The size and presence of a prediction target (hematocrit) makes this dataset unique, even among datasets in established repositories like the Human Cell Atlas (Regev et al., 2017). Its relevance for clinical science makes it an exciting tool not only for methods development, but also for scientific inquiry. See Appendix A for details.
- We develop **Deep Sets++ (DS++)** and **Set Transformer++ (ST++)**, leveraging set norm and equivariant residual connections (Section 5). We demonstrate that both models improve performance over their existing counterparts across a variety of tasks: regression on real-valued inputs, regression on sets of images, point cloud classification, and hematocrit prediction from single-cell data (Section 6). On this last task, ST++ reaches state-of-the-art performance with exciting implications for clinical science.

2 Background

Let S be the number of samples in a set. A function $f : \mathcal{X}^S \to \mathcal{Y}$ is *permutation invariant* if any permutation π of the input set results in the same output: $f(\pi x) = f(x)$. A function $\sigma : \mathcal{X}^S \to \mathcal{Y}^S$ *permutation equivariant* if, for any permutation π, the outputs are permuted accordingly: $\sigma(\pi \mathbf{x}) = \pi\sigma(\mathbf{x})$. A sum/max-decomposable function $f : \mathcal{X}^S \to \mathcal{Y}$ is one which can be expressed via an equivariant encoder $\sigma : \mathcal{X}^S \to \mathcal{Z}^S$ that maps each input sample in $\mathcal{X}$ to a latent space $\mathcal{Z}$, a sum or max aggregation over the sample dimension of the encoder output, and an unconstrained decoder $\rho : \mathcal{Z} \to \mathcal{Y}$:

$$f(\mathbf{x}) = \rho\left(\sum \sigma(\mathbf{x})\right). \tag{1}$$

An architecture is a universal approximator of permutation invariant functions if and only if it is sum/max-decomposable with sufficient conditions on the latent space dimension (Wagstaff et al., 2019). Both the Deep Sets and Set Transformer architectures can be written in the form of Equation (1). Concretely, Deep Sets architecture consists of an equivariant encoder made up of feedforward layers, a sum or max aggregation, and a decoder also made up of feedforward layers (Zaheer et al., 2017). The Set Transformer encoder consists of multi-attention blocks called inducing point attention blocks (Lee et al., 2019), a variant of the original transformer block (Vaswani et al., 2017) modified to handle large set sizes. The aggregation is learned via attention, and the decoder consists of transformer blocks with self-attention.

Both architectures are permutation invariant since they consist of a permutation equivariant encoder and a permutation invariant aggregation over the outputs of the encoder. In order to maintain permutation invariance of the architectures upon modification, one needs to ensure that changes to the encoder preserve permutation equivariance, and changes to the aggregation maintain permutation invariance. In the next two sections, we discuss how to incorporate normalization layers and residual connections while maintaining permutation invariance overall.

3 Normalization for Set Data

Normalization layers have been widely used in neural networks to ease optimization and add regularization (Ioffe & Szegedy, 2015; Ba et al., 2016). Normalization layers have been shown to smooth the optimization landscape (Santurkar et al., 2018), suppress outlier eigenvalues of the Hessian (Ghorbani et al., 2019), regularize the model (Luo et al., 2019), and control the variance of the gradients (Xiong et al., 2020).

Incorporating normalization layers into the permutation equivariant encoders of Deep Sets and Set Transformer[2] requires careful consideration, as not all normalization layers are appropriate to use. We formally introduce the desiderata every such layer should meet to be a good candidate for permutation invariant architectures. In particular:

(i) The normalization must be permutation equivariant.

(ii) The normalization should work well with sets of varying sizes.

(iii) The normalization should not remove essential information needed for the prediction.

Let $\mathbf{a} \in \mathbb{R}^{B \times S \times D}$ be the activation before the normalization operation, where B is the size of the batch, S is the number of samples in a set (the tensor is zero-padded to the largest sample size), and D is the feature dimension. We study normalization layers as two step operations. First, the activations are standardized based on a setting L which defines which dimensions utilize separate statistics. The statistics are repeated over the remaining dimensions such that $\mu_L(\mathbf{a}), \sigma_L(\mathbf{a}) \in \mathbb{R}^{B \times S \times D}$ match the original activation tensor for elementwise subtraction and division. For instance, $L = \{B, S\}$ denotes that each set in a batch and each sample in a set gets its own mean and standard deviation for standardization, which are repeated over the D dimension. A standardization operation can be defined as:

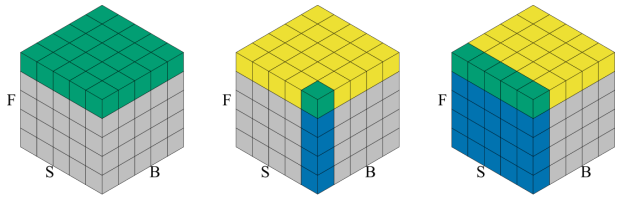
$$\bar{\mathbf{a}}_L = \frac{\mathbf{a} - \mu_L(\mathbf{a})}{\sigma_L(\mathbf{a})}. \tag{2}$$

Next, the standardized activations are transformed through learned parameters which differ only over a setting of dimensions M. Let $\overrightarrow{\gamma}_M, \overrightarrow{\beta}_M \in \mathbb{R}^{B \times S \times D}$ denote the learned parameters which are unique over the dimensions in M and shared over the remaining dimensions. For instance, $M = \{D\}$ denotes that each feature is transformed by a different scale and bias, which are shared across the sets in the batch and samples in the sets. With $\odot$ representing elementwise multiplication, any transformation operation can be defined as:

$$\hat{\mathbf{a}}_M = \bar{\mathbf{a}} \odot \overrightarrow{\gamma}_M + \overrightarrow{\beta}_M. \tag{3}$$

Normalization layers can be summarized by their choice of L and M.

[2]The Set Transformer paper (Lee et al., 2019) mentions layer norm as a component of their equivariant block but turns it off in actual implementation.



(i) Feature Norm (ii) Layer Norm (iii) Set Norm

Figure 2: We can represent normalization layers by the dimensions they perform standardization (blue) and the dimensions they perform transformation (yellow). Green indicates overlap of yellow and blue.

The only suitable transformation for sets is on the feature dimension. Only normalization layers adopting a per feature transformation, i.e. $M = \{D\}$, meet the permutation equivariance desiderata (i), as per sample transformations break permutation equivariance and per set transformations cause the prediction output to change when an element's position in the batch changes (see Appendix B for proofs).

3.1 Existing Normalizations

With the desiderata in mind, we identified two suitable existing normalizations in the literature. The first, which we call *feature norm*, is used in the permutation invariant PointNet and PointNet++ architectures for point cloud classification (Qi et al., 2017a;b). The second is layer norm, proposed as the normalization layer in Set Transformer and found in the transformer architecture for natural language processing (Vaswani et al., 2017; Lee et al., 2019). These normalizations (and any appropriate ones for prediction on sets) use the same per feature transformation setting. Thus, we focus on the differences in their standardization operation for comparison.

Feature norm. Feature norm performs a per feature standardization followed by a per feature transformation, i.e. $L = \{D\}, M = \{D\}$, see Figure 2 (i):

$$\mathrm{FN}(a_{bsd}) = \frac{a_{bsd} - \mu_d}{\sigma_d} \odot \gamma_d + \beta_d, \qquad \mu_d = \frac{1}{B}\frac{1}{S}\sum_{b=1}^{B}\sum_{s=1}^{S} a_{bsd}, \quad \sigma_d^2 = \frac{1}{B}\frac{1}{S}\sum_{b=1}^{B}\sum_{s=1}^{S}(a_{bsd} - \mu_d)^2$$

Since feature norm computes batch statistics, it may exhibit batch effects: sets in the batch are no longer independent, and performance can be heavily impacted by batch size (Hoffer et al., 2017). Additionally, like in batch norm, in feature norm sets are standardized differently at training and test time.[3] With respect to the desiderata, feature norm is permutation equivariant but does not cleanly handle sets of varying sizes, as batch statistics will be skewed towards the statistics of the larger sets. An alternative approach is to weight sets equally, first computing the statistics within each set and then averaging over each set in the batch. In this situation, however, samples are no longer weighted equally, as samples from smaller sets will contribute more to the batch statistics.

Layer norm. Layer norm applies standardization individually to each set and each sample and transformation separately per feature, i.e. $L = \{B, S\}, M = \{D\}$, see Figure 2(ii):

$$\mathrm{LN}(a_{bsd}) = \frac{a_{bsd} - \mu_{bs}}{\sigma_{bs}} \odot \gamma_d + \beta_d, \quad \mu_{bs} = \frac{1}{D}\sum_{d=1}^{D} a_{bsd}, \quad \sigma_{bs}^2 = \frac{1}{D}\sum_{d=1}^{D}(a_{bsd} - \mu_{bs})^2$$

Layer norm is permutation equivariant and naturally handles set of different sizes, since statistics are specific to each set. However, layer norm removes information which may be particularly useful for set prediction tasks. Indeed, by performing standardization on each sample in each set separately, layer norm maps two samples whose activations differ in only a scale and bias to the same output (see Appendix B for proof). The inability to distinguish such samples after layer norm can hurt prediction.

[3] Feature norm is implemented as batch norm on a transposed tensor. Thus, test batches are standardized by a running mean and variance while training batches are standardized by their own batch statistics.

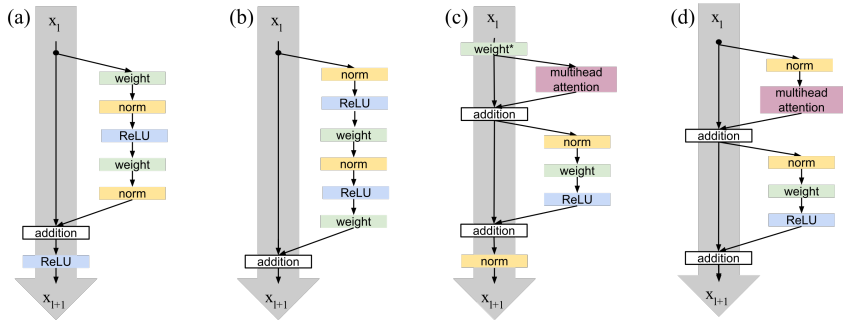


Figure 3: We compare the effect of a ResNet (a) vs. He (b) residual pipeline on Deep Sets, as well as the effect of the original Set Transformer pipeline (c) with a pipeline for sets analogous to that of the Pre-LN Transformer (d). Figures (b) and (d) have clean residual paths (grey arrow), unlike their counterparts in (a) and (c). In (c), weight* is also part of the attention computation.

3.2 Set norm

We introduce a new normalization layer with the intent of alleviating the drawbacks introduced by feature norm and layer norm. Our proposed normalization, set norm (SN), is a per set standardization, per feature transformation, i.e. $L = \{B\}, M = \{D\}$, see Figure 2(iii):

$$\text{SN}(a_{bsd}) = \frac{\mathbf{a}_b - \mu_b}{\sigma_b} \odot \gamma_d + \beta_d, \quad \mu_b = \frac{1}{S}\frac{1}{D}\sum_{s=1}^{S}\sum_{d=1}^{D} a_{bsd}, \quad \sigma_b^2 = \frac{1}{S}\frac{1}{D}\sum_{s=1}^{S}\sum_{d=1}^{D}(a_{bsd} - \mu_b)^2$$

SN is permutation equivariant (see Appendix B for proof) and able to handle sets of different sizes, since it is a per set standardization. It also avoids batch effects that may arise when using feature norm while resulting in less information loss than layer norm. In fact, the per set standardization of set norm, i.e. $L = \{B\}$, removes the least information of any standardization which acts on each set separately (see Appendix B for proof). It is for this reason that we propose set norm over another per set standardization, e.g. per set per feature.

Experimental results (Section 6) confirm our hypothesis that set norm is overall preferable to feature norm and layer norm. However, set norm does remove some information, namely the global mean and variance of each set (as does any standardization with $B \in M$). In the next section, we discuss how we can alleviate this information loss through the use of residual connections.

4 Residual connections for set data

Residual connections have been widely used to facilitate the training of deeper models (He et al., 2016b; Li et al., 2019). They have been shown to address the vanishing gradient problem (Veit et al., 2016) and smooth the loss landscape (Yao et al., 2020).

Residual connections can be incorporated into an equivariant encoder by adding each sample's input to its corresponding output. Such residual connections already exist in Set Transformer, as transformer blocks include residual connections. Here, we formalize equivariant residual connections more generally for any architecture.

An **equivariant residual connection** is a residual connection that acts on every sample. 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 equivariant residual connections to any equivariant function f is also permutation equivariant:

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

There are many existing arrangements of residual pipelines (He et al., 2016b;a; Vaswani et al., 2017; 2018). We hypothesize that the best way to implement residual connections for general-purpose use across data modalities is to leave a clean path from input to output without other operations in

between; we call this the *clean residual path* principle, where the residual path is the shortest path from input to output of a block (see Figure 3 for illustration). Indeed, having certain operations between skip connections could yield undesirable effects.

A Residual Pipeline Separated by Linear Layers Can Suffer From Vanishing or Exploding Gradients. While identity residual connections have been shown to prevent gradient vanishing, He et al. (2016a) note 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). We note that the same is true of a weight matrix in between residual connections, i.e. $\mathbf{x}_{\ell+1} = \mathbf{x}_\ell W + \mathcal{F}((\mathbf{x}_\ell)W)$. By replacing a product of ones inside the gradient expression with a product of learned weights, the addition of weight matrices along the residual path reintroduces the potential for the gradients of early layers in a network to be very large or very small.

Normalization Along The Residual Path Can Also Cause Vanishing or Exploding Gradients. Wang et al. (2019) note that layer norm along the residual path results in a product of its derivatives collecting on that path in the gradient computation (i.e. $\prod_{k=\ell}^{L-1} \frac{\partial \mathrm{LN}(y_k)}{y_k}$), with the number of items scaling with depth of the network. The same is true replacing layer norm with another normalization layer along the path. On the contrary, moving the normalization off the path removes this product term from the gradient. Indeed, Xiong et al. (2020) show that transformers without a clean path require careful learning rate scheduling unlike their clean path counterparts, and Wang et al. (2019) see that for deep transformers, only the clean path variants converge during training.

ReLu Activations Between Connections Force the Output to Grow. Placing a ReLU activation between residual connections, i.e. $\mathbf{x}_{\ell+1} = \mathrm{relu}(\mathbf{x}_\ell) + \mathcal{F}(\mathrm{relu}(\mathbf{x}_\ell))$ ensures that the first additive term is always positive. Such a constraint can cause output explosion for deep models, causing instability.

For the above reasons, we propose to utilize the clean residual path principle in our architectures, i.e. no operations are performed on the residual path from input to output. Concretely, we suggest using the pipeline proposed in He et al. (2016a) over the original ResNet pipeline (He et al., 2016b), as well as the Pre-LN Transformer (Klein et al., 2017; Vaswani et al., 2018; Xiong et al., 2020) residual structure over that of Set Transformer[4] (Lee et al., 2019). See Figure 3 for a visual comparison.

4.1 Residuals Can Alleviate Loss of Information from Normalization

Beyonds their benefits for optimization, residual connections can also help reintroduce information lost by normalization. Specifically, if a normalization layer maps two inputs to the same output, adding back in the input via a skip connection ensures that the outputs of these two distinct sets remain different. Consider set norm placed before the nonlinearity in Deep Sets, e.g. $f(\mathbf{x}) = \mathrm{relu}(\mathrm{SN}(\mathbf{x}W))$ for a single layer. If two sets are scaled and translated versions of each other, i.e. $\mathbf{y} = a\mathbf{x} + b$, then without residual connections, the network with set norm will map both inputs to the same output, i.e. $f(\mathbf{x}) = f(\mathbf{y})$. On the other hand, Deep Sets with a residual connection $g(\mathbf{x}) = \mathbf{x} + f(\mathbf{x}) \neq g(\mathbf{y})$ will result in different outputs. This difference can have large implications for performance on some tasks, as shown in Section 6.1.

5 Making State of the Art Architectures Deeper

We develop Deep Sets++ and Set Transformer++ by adding set norm and clean path equivariant residual connections to the DeepSets and Set Transformer encoders respectively. We fixed the decoder and aggregation to their original versions, focusing on the effect of the encoder changes. Since the decoder is an unconstrained network, we expect general architectural principles non-specific to sets to be applicable, and because of our focus is scaling architectures to large depths, we do not modify the aggregation, which itself is a single block.

DeepSets++ adds a set norm layer before the nonlinearity and a clean residual path to the encoder (See Figure 3b). Following (He et al., 2016a), we precede the first residual block with a linear layer and no bias, as is customary before a normalization layer (Ioffe & Szegedy, 2015; Ba et al., 2016) and place a normalization-relu-weight operation after the final residual block. Set Transformer++

[4]The residual pipeline of Set Transformer almost matches that that of the original Transformer architecture except that, for each attention block, the query matrix $\mathbf{a}W_q + b_q$ rather than the input $\mathbf{a}$ is added to the output.

Table 1: Deep Sets++ and Set Transformer++ outperform their Deep Sets and Set Transformer counterparts at 50 and 16 layers respectively.

	Hematocrit	Point Cloud	MNIST Var	Normal Var
Deep Sets	25.9971	3.6041	5.6388	8.0486
Deep Sets++	**18.7653**	**0.6980**	**0.3531**	**0.0317**
Set Transformer	18.7777	0.8808	6.2708	**0.0001**
Set Transformer++	**18.6853**	**0.5858**	**0.7950**	0.0079

Table 2: Adding set norm and feature norm improves performance of Deep Sets (50 layers), while adding layer norm does not.

	Hematocrit	Point Cloud	MNIST Var	Normal Var
no norm	25.9971	3.6041	5.6388	8.0486
layer norm	26.0818	3.6050	5.6294	8.2455
feature norm	23.0795	**0.9309**	0.9072	**0.5089**
set norm	**20.6127**	1.7334	**0.3514**	5.4233

adds a set norm layer and modifies the residual connections in the encoder to adhere to the clean path principle (see Figure 3d). This architecture mirrors that of the Pre-LN Transformer (Klein et al., 2017; Vaswani et al., 2018). See Appendix C for full details.

6 Experiments

We run our main experiments and ablation studies on four tasks with diverse inputs (point cloud, continuous, images) and outputs (regression, classification).

- **Hematocrit Prediction from Blood Cell Cytometry Data (Hematocrit).** The dataset we release (Flow-RBC) consists of patients' red blood cell samples (two measurements per cell) and their hematocrit levels measured at a particular visit. 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 sample 1,000 cells per patient and predict hematocrit levels. See Appendix A for complete details about the dataset.
- **Point Cloud Classification (Point Cloud).** We use the ModelNet40 dataset (Wu et al., 2015) and randomly sample 1000 points per set, following Zaheer et al. (2017) and Lee et al. (2019). We standardize each object to have mean zero and unit variance following Zaheer et al. (2017).
- **Variance Prediction, Image Data (MNIST Var).** We implement empirical variance prediction on MNIST digits as a proxy for real-world tasks with sets of images, e.g. prediction on sets of blood smear slides, histopathology slides. We sample 10 images uniformly from the training set and use as a label the empirical variance of the digits in the set. We construct the test set in a similar way, where the 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 pass a flattened 1D vector as input.
- **Empirical Variance Prediction, Real Data (Normal Var).** Each set is a collection of samples (sample size 1000) 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 in the set. Training set size is 10,000 sets, and test set size is 1,000 sets.

Results are reported in Mean Squared Error (MSE) on the test set for the regression experiments and in cross entropy loss for point cloud classification. 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 training and evaluation details, see Appendix C.[5]

[5] All the code and experiments are available at link-omitted-for-anonymity.

Table 3: Adding equivariant residual connections to Deep Sets improves performance across tasks. All ablations surpass the baseline Deep Sets model (none-none) across tasks.

Residuals	Norm	Hematocrit	Point Cloud	Mnist Var	Normal Var
none	none	25.9971	3.6041	5.6388	8.0486
non clean path	layer norm	20.9208	0.5928	0.3822	0.3749
	feature norm	19.2851	0.6131	0.3459	2.0213
	set norm	19.2559	**0.6003**	0.3713	0.1476
clean path	layer norm	19.4076	0.6742	0.3681	0.0335
	feature norm	19.2516	0.7211	**0.3064**	0.0667
	set norm	**18.7653**	0.6980	0.3531	**0.0317**

Table 4: The setting of Set Transformer which performs best overall uses set norm with a clean residual path. Rightmost columns indicate how often the setting outperforms the original Set Transformer (Original) or a no normalization model (No Norm).

Residuals	Norm	Hematocrit	Point Cloud	MNIST Var	Normal Var	Original	No Norm
non-clean path	none	18.7777	0.8808	6.2708	**0.0001**	-	-
	layer norm	19.0882	0.9654	**0.3835**	0.0133	1/4	1/4
	feature norm	19.8088	0.8463	0.5318	0.0921	2/4	2/4
	set norm	19.0590	0.9717	0.4176	0.0014	1/4	1/4
clean path	none	19.0501	0.6075	24.3831	0.0027	1/4	-
	layer norm	18.9370	0.7424	0.6435	0.0003	2/4	2/4
	feature norm	19.0328	0.7029	0.8096	0.0114	2/4	2/4
	set norm	**18.6853**	**0.5858**	0.7950	0.0079	**3/4**	**3/4**

6.1 Main Results

Deep Sets++ and Set Transformer++ outperform existing architectures. Figure 1 and Table 1 show that DS++ and ST++ yield comparable or better performance than their existing counterparts, shallow or deep, across all tasks. Deep Sets++ consistently improves over Deep Sets (50 encoder layers), sometimes with orders of magnitude improvement in test loss. Set Tranformer++ improves performance on Point Cloud and MNIST Var and reaches comparable performance on Hematocrit and Normal Var. Furthermore, in Appendix D, we show on an official point cloud benchmark repo (Goyal et al., 2021a) that DS++ and ST++ without any modifications outperform versions of Deep Sets and Set Transformer tailored for point cloud classification.

Adding feature norm or set norm to Deep Sets improves performance. We compare how all suitable norms (feature norm, layer norm, and set norm) affect performance of a deep Deep Sets model (50 encoder layers). Results in Table 2 show that layer norm leads to worse performance than no normalization on Hematocrit, Point Cloud, and Normal Var. We hypothesize that the problem is primarily information loss and explore in Appendix D.3 why layer norm may be suitable for categorical text data and not for real-valued sets. On the contrary, set norm and feature norm both improve results compared to no normalization. Set norm has better performance on Hematocrit and MNIST Var, while feature norm performs better on Point Cloud and Normal Var. For Normal-Var, set norm performs poorly as it specifically removes the scale information that is needed to compute variance. For point cloud, we speculate that feature norm's benefit comes from specific invariances that hold for clouds of points. Indeed, in Appendix D, we show that a hybrid of set norm and feature norm performs similarly to feature norm, suggesting that the difference in results is due to the benefit of per feature standardization for the specific task, rather than a failure of set norm.

Residual connections following the clean path principle improve performance. We confirm through experiments that clean residual path pipelines generally aid in performance across various set tasks (Table 3 and Table 4). For Deep Sets (Table 3), the clean path outperforms the non-clean path on three out of four tasks across all normalizations. With set norm on Normal Var, the clean path residual (3e-2) especially improves performance over the non-clean path residual (1e-1) and no residual (5.4), a testament to the ability of a clean path residual pipeline to make up for the loss of global scale information by the set norm operation. The clean path underperforms the non clean path only on Point Cloud; in Appendix D, we show that this specific task benefits from positive numbers

in the relu addition, as the addition of more positive numbers via more frequent residual additions further improves results. Results on Set Transformer (Table 4) show that in general, the combination of clean path and set norm outperforms all the other combinations and improves over the original Set Transformer architecture (original-no norm). The positive effect of a clean residual pipeline is particularly evident in Hematocrit and Point Cloud. For Set Transformer and Set Transformer++ across tasks, normalization layers improve results more often when they are off the residual path, i.e. under a clean path residual.

Set Transformer++ sets state-of-the-art performance on hematocrit prediction. The prediction of hematocrit (the fraction of blood volume occupied be RBCs) from a set of individual RBC measurements (volume and hemoglobin) aims to answer an open biological and clinical question: are characteristics of individual cells, an intensive property, predictive of their prevalence in the blood, an extensive property? On this task, increases in performance provide exciting scientific signal which can help lead to a deeper understanding of RBC production and regulation. Clinical literature reports a correlation between hematocrit levels and red blood cell distribution width (RDW), i.e. SD(volume)/mean(volume) in anemic patients (McPherson et al., 2021, Chapter 9). Results obtained with our architectures suggest that there is further predictive information beyond what has been documented in the medical literature, as performances surpass a model using RDW as well as prediction from moments computed on the distribution (see Appendix A). The best performing model is ST++ with 18.69 MSE, a 28% reduction in error over the clinical baseline RDW model (25.85 MSE). Set Transformer and DS++ perform similarly while Deep Sets-50 layers performs worse than the clinical baseline.

7 Related Work

There exist several permutation invariant neural architectures in the literature: Deep Sets (Zaheer et al., 2017) and Set Transformer (Lee et al., 2019) for general-purpose sets, and PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b) specialized for point cloud data. The architectures introduced in this paper are general-purpose architectures and the first permutation invariant architectures of their depth that show good performance on a variety of tasks without task-specific architectural changes. Making neural architectures deeper takes inspiration from previous efforts in computer vision (He et al., 2016b), natural language (Wang et al., 2019), and prediction on graphs (Li et al., 2019). Here, we consider both distributional real-valued data as well as sets of high-dimensional inputs. Lastly, the development of normalization layers suited for particular application is not a new concept. For instance, batch norm (Ioffe & Szegedy, 2015) is the default choice for computer vision tasks, layer norm Ba et al. (2016) for text inputs, instance norm Ulyanov et al. (2016) for stylization, and graph norm (Cai et al., 2021) for graphs.

8 Conclusion

We introduced set norm and equivariant residual connections, illustrated the benefits of both, and compared them to other choices of normalization layers and residual pipelines. Using set norm and equivariant residual connections, we developed Deep Sets++ and Set Transformer++, deep permutation invariant architectures which improve upon the current state-of-the-art models. Such architectures are general purpose and can achieve good performance on a variety of tasks without hyperparameter tuning. Interestingly, between Deep Sets++ and Set Transformer++, there is not a best choice in architecture across tasks, so the choice for which to use may often come down to computational resources and time (Set Transformer++ with 16 layers has more expensive computational and memory requirements than Deep Sets++ with 50 layers, so Deep Sets++ may be preferred under limited resources). Lastly, we introduced a new open-source dataset, Flow-RBC, which provides a real-world application of permutation invariant prediction in clinical science. With data from over 100,000 patients, including an external clinical measurement (i.e. hematocrit), Flow-RBC is uniquely suited among single-cell datasets to be a benchmark for deep predictive modeling on sets. We believe our new models and dataset have the potential to motivate future work and impactful applications of neural set-based prediction.

REFERENCES

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

Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training. In *ICML*, 2021.

M. Chen, Orhan Firat, Ankur Bapna, M. Johnson, Wolfgang Macherey, George F. Foster, Llion Jones, Niki Parmar, M. Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*, 2018.

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

Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *ICML*, 2021a.

Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. *International Conference on Machine Learning*, 2021b.

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

Neil Harris, Jolanta Kunicka, and Alexander Kratz. 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.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *ArXiv*, abs/1603.05027, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016b.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *ArXiv*, abs/1705.08741, 2017.

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. *ArXiv*, abs/1701.02810, 2017.

David Lähnemann, Johannes Köster, Ewa Szczurek, Davis J. McCarthy, Stephanie C. Hicks, Mark D. Robinson, Catalina A. Vallejos, Kieran R. Campbell, Niko Beerenwinkel, Ahmed Mahfouz, Luca Pinello, Pavel Skums, Alexandros Stamatakis, Camille Stephan-Otto Attolini, Samuel Aparicio, Jasmijn A. Baaijens, Marleen Balvert, Buys de Barbanson, Antonio Cappuccio, Giacomo Corleone, Bas E. Dutilh, Maria Florescu, Victor Guryev, Rens Holmer, Katharina Jahn, Thamar Jessurun Lobo, Emma M. Keizer, Indu Khatri, Szymon M. Kiełbasa, Jan O. Korbel, Alexey M. Kozlov, Tzu-Hao Kuo, Boudewijn P. F. Lelieveldt, Ion I. Măndoiu, John C. Marioni, Tobias Marschall, Felix Mölder, Amir Niknejad, Lukasz Raczkowski, Marcel J. T. Reinders, Jeroen de Ridder, Antoine-Emmanuel Saliba, Antonios Somarakis, Oliver Stegle, Fabian Joachim Theis, Huan Yang, Alex Zelikovsky, Alice Carolyn McHardy, Benjamin J. Raphael, Sohrab P. Shah, and Alexander Schönhuth. Eleven grand challenges in single-cell data science. *Genome Biology*, 21, 2020.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Y. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

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

Jiajia Liu, Zhiwei Fan, Weiling Zhao, and Xiaobo Zhou. Machine intelligence in single-cell data analysis: Advances and new challenges. *Frontiers in Genetics*, 12, 2021.

Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization. *ArXiv*, abs/1809.00846, 2019.

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

C. Qi, Hao Su, Kaichun Mo, and L. Guibas. 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.

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

Aviv Regev, Sarah A Teichmann, Eric S Lander, Ido Amit, Christophe Benoist, Ewan Birney, Bernd Bodenmiller, Peter Campbell, Piero Carninci, Menna Clatworthy, et al. Science forum: the human cell atlas. *elife*, 6:e27041, 2017.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *NeurIPS*, 2018.

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

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

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, François Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Lukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam M. Shazeer, and Jakob Uszkoreit. Tensor2tensor for neural machine translation. In *AMTA*, 2018.

Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, 2016.

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

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.

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

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. *ArXiv*, abs/2002.04745, 2020.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W. Mahoney. Pyhessian: Neural networks through the lens of the hessian. *2020 IEEE International Conference on Big Data (Big Data)*, pp. 581–590, 2020.

Guocheng Yuan, Long Cai, Michael B. Elowitz, Tariq Enver, Guoping Fan, Guoji Guo, Rafael A. Irizarry, Peter V. Kharchenko, Junhyong Kim, Stuart H. Orkin, John Quackenbush, Assieh Saadatpour, Timm Schroeder, Ramesh A. Shivdasani, and Itay Tirosh. Challenges and emerging directions in single-cell analysis. *Genome Biology*, 18, 2017.

M. Zaheer, Satwik Kottur, Siamak Ravanbakhsh, B. Póczos, R. Salakhutdinov, and Alex Smola. Deep sets. In *NeurIPS*, 2017.