Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks

Juho Lee [1 2] **Yoonho Lee** [3] **Jungtaek Kim** [4] **Adam R. Kosiorek** [1 5] **Seungjin Choi** [4] **Yee Whye Teh** [1]

Abstract

Many machine learning tasks such as multiple instance learning, 3D shape recognition and few-shot image classification are defined on sets of instances. Since solutions to such problems do not depend on the order of elements of the set, models used to address them should be *permutation invariant*. We present an attention-based neural network module, the *Set Transformer*, specifically designed to model interactions among elements in the input set. The model consists of an encoder and a decoder, both of which rely on attention mechanisms. In an effort to reduce computational complexity, we introduce an attention scheme inspired by inducing point methods from sparse Gaussian process literature. It reduces computation time of self-attention from quadratic to linear in the number of elements in the set. We show that our model is theoretically attractive and we evaluate it on a range of tasks, demonstrating increased performance compared to recent methods for set-structured data.

1. Introduction

Learning representations has proven to be an essential problem for deep learning and its many success stories. The majority of problems tackled by deep learning are *instance-based* and take the form of mapping a fixed-dimensional input tensor to its corresponding target value (Krizhevsky et al., 2012; Graves et al., 2013).

For some applications, we are required to process *set-structured data*. Multiple instance learning (Dietterich et al., 1997; Maron & Lozano-Pérez, 1998) is an example of such a *set-input* problem, where a set of instances is given as an input and the corresponding target is a label for the entire set. Other problems such as 3D shape recognition (Wu et al., 2015; Shi et al., 2015; Su et al., 2015; Charles et al., 2017), sequence ordering (Vinyals et al., 2016), and various set operations (Muandet et al., 2012; Oliva et al., 2013; Edwards & Storkey, 2017; Zaheer et al., 2017) can also be viewed as the set-input problems. Moreover, many meta-learning (Thrun & Pratt, 1998; Schmidhuber, 1987) problems which learn using different, but related tasks may also be treated as set-input tasks where an input set corresponds to the training dataset of a single task. For example, few-shot image classification (Finn et al., 2017; Snell et al., 2017; Lee & Choi, 2018) operates by building a classifier using a support set of images, which is evaluated with query images.

A model for *set-input* problems should satisfy two critical requirements. First, it should be *permutation invariant* — the output of the model should not change under any permutation of the elements in the input set. Second, such a model should be able to process input sets of any size. While these requirements stem from the definition of a set, they are not easily satisfied in neural-network-based models: classical feed-forward neural networks violate both requirements, and RNNs are sensitive to input order.

Recently, Edwards & Storkey (2017) and Zaheer et al. (2017) propose neural network architectures which meet both criteria, which we call *set pooling* methods. In this model, each element in a set is first independently fed into a feed-forward neural network that takes fixed-size inputs. Resulting feature-space embeddings are then aggregated using a *pooling* operation (mean, sum, max or similar). The final output is obtained by further non-linear processing of the aggregated embedding. This remarkably simple architecture satisfies both aforementioned requirements, and more importantly, is proven to be a universal approximator for any set function (Zaheer et al., 2017). Thanks to this property, it is possible to learn a complex mapping between input sets and their target outputs in a black-box fashion, much like with feed-forward or recurrent neural networks.

Even though this set pooling approach is theoretically attractive, it remains unclear whether we can approximate

[1]Department of Statistics, University of Oxford, United Kingdom [2]AITRICS, Republic of Korea [3]Kakao Corporation, Republic of Korea [4]Department of Computer Science and Engineering, POSTECH, Republic of Korea [5]Oxford Robotics Institute, University of Oxford, United Kingdom. Correspondence to: Juho Lee <juho.lee@stats.ox.ac.uk>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

complex mappings well using only instance-based feature extractors and simple pooling operations. Since every element in a set is processed independently in a set pooling operation, some information regarding interactions between elements has to be necessarily discarded. This can make some problems unnecessarily difficult to solve.

Consider the problem of *amortized clustering*, where we would like to learn a parametric mapping from an input set of points to the centers of clusters of points inside the set. Even for a toy dataset in 2D space, this is not an easy problem. The main difficulty is that the parametric mapping must assign each point to its corresponding cluster while modelling the explaining away pattern such that the resulting clusters do not attempt to explain overlapping subsets of the input set. Due to this innate difficulty, clustering is typically solved via iterative algorithms that refine randomly initialized clusters until convergence. Even though a neural network with a set poling operation can approximate such an amortized mapping by learning to quantize space, a crucial shortcoming is that this quantization cannot depend on the contents of the set. This limits the quality of the solution and also may make optimization of such a model more difficult; we show empirically in Section 5 that such pooling architectures suffer from under-fitting.

In this paper, we propose a novel set-input deep neural network architecture called the *Set Transformer*, (*cf. Transformer*, (Vaswani et al., 2017)). The novelty of the Set Transformer is in three important design choices:

1. We use a self-attention mechanism to process every element in an input set, which allows our approach to naturally encode pairwise- or higher-order interactions between elements in the set.

2. We propose a method to reduce the $\mathcal{O}(n^2)$ computation time of full self-attention (e.g. the Transformer) to $\mathcal{O}(nm)$ where m is a fixed hyperparameter, allowing our method to scale to large input sets.

3. We use a self-attention mechanism to aggregate features, which is especially beneficial when the problem requires multiple outputs which depend on each other, such as the problem of meta-clustering, where the meaning of each cluster center heavily depends its location relative to the other clusters.

We apply the Set Transformer to several set-input problems and empirically demonstrate the importance and effectiveness of these design choices, and show that we can achieve the state-of-the-art performances for the most of the tasks.

2. Background

2.1. Pooling Architecture for Sets

Problems involving a set of objects have the *permutation invariance* property: the target value for a given set is the same regardless of the order of objects in the set. A simple example of a permutation invariant model is a network that performs pooling over embeddings extracted from the elements of a set. More formally,

$$\text{net}(\{x_1, \ldots, x_n\}) = \rho(\text{pool}(\{\phi(x_1), \ldots, \phi(x_n)\})). \quad (1)$$

Zaheer et al. (2017) have proven that all permutation invariant functions can be represented as (1) when pool is the sum operator and ρ, ϕ any continuous functions, thus justifying the use of this architecture for set-input problems.

Note that we can deconstruct (1) into two parts: an *encoder* (ϕ) which independently acts on each element of a set of n items, and a *decoder* ($\rho(\text{pool}(\cdot))$) which aggregates these encoded features and produces our desired output. Most network architectures for set-structured data follow this encoder-decoder structure.

Zaheer et al. (2017) additionally observed that the model remains permutation invariant even if the encoder is a stack of permutation-equivariant layers:

Definition 1. *Let S_n be the set of all permutations of indices $\{1, \ldots, n\}$. A function $f : X^n \to Y^n$ is permutation equivariant iff for any permutation $\pi \in S_n$, $f(\pi x) = \pi f(x)$.*

An example of a permutation-equivariant layer is

$$f_i(x; \{x_1, \ldots, x_n\}) = \sigma_i(\lambda x + \gamma \text{pool}(\{x_1, \ldots, x_n\})) \quad (2)$$

where pool is the pooling operation, λ, γ are learnable scalar variables, and $\sigma(\cdot)$ is a nonlinear activation function.

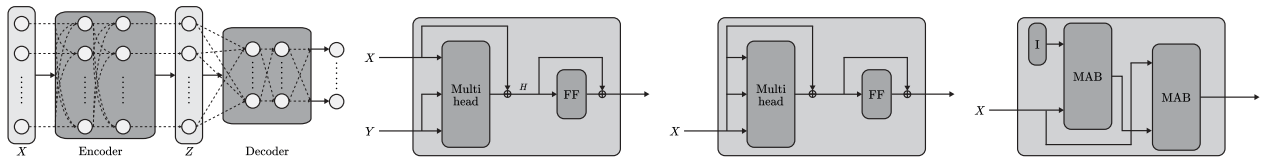
2.2. Attention

Assume we have n query vectors (corresponding to a set with n elements) each with dimension d_q: $Q \in \mathbb{R}^{n \times d_q}$. An attention function $\text{Att}(Q, K, V)$ is a function that maps queries Q to outputs using n_v key-value pairs $K \in \mathbb{R}^{n_v \times d_q}, V \in \mathbb{R}^{n_v \times d_v}$.

$$\text{Att}(Q, K, V; \omega) = \omega\left(QK^\top\right)V. \quad (3)$$

The pairwise dot product $QK^\top \in \mathbb{R}^{n \times n_v}$ measures how similar each pair of query and key vectors is, with weights computed with an activation function ω. The output $\omega(QK^\top)V$ is a weighted sum of V where a value gets more weight if its corresponding key has larger dot product with the query.

Multi-head attention, originally introduced in Vaswani et al. (2017), is an extension of the previous attention



(a) Our model (b) MAB (c) SAB (d) ISAB

Figure 1. Diagrams of our attention-based set operations.

scheme. Instead of computing a single attention function, this method first projects Q, K, V onto h different d_q^M, d_q^M, d_v^M-dimensional vectors, respectively. An attention function ($\mathrm{Att}(\cdot;\omega_j)$) is applied to each of these h projections. The output is a linear transformation of the concatenation of all attention outputs:

$$\mathrm{Multihead}(Q, K, V; \lambda, \omega) = \mathrm{concat}(O_1, \cdots, O_h)W^O, \tag{4}$$

$$\text{where } O_j = \mathrm{Att}(QW_j^Q, KW_j^K, VW_j^V; \omega_j) \tag{5}$$

Note that $\mathrm{Multihead}(\cdot,\cdot,\cdot;\lambda)$ has learnable parameters $\lambda = \{W_j^Q, W_j^K, W_j^V\}_{j=1}^h$, where $W_j^Q, W_j^K \in \mathbb{R}^{d_q \times d_q^M}$, $W_j^V \in \mathbb{R}^{d_v \times d_v^M}$, $W^O \in \mathbb{R}^{hd_v^M \times d}$. A typical choice for the dimension hyperparameters is $d_q^M = d_q/h$, $d_v^M = d_v/h$, $d = d_q$. For brevity, we set $d_q = d_v = d$, $d_q^M = d_v^M = d/h$ throughout the rest of the paper. Unless otherwise specified, we use a scaled softmax $\omega_j(\cdot) = \mathrm{softmax}(\cdot/\sqrt{d})$, which our experiments were worked robustly in most settings.

3. Set Transformer

In this section, we motivate and describe the *Set Transformer*: an attention-based neural network that is designed to process sets of data. Similar to other architectures, a Set Transformer consists of an encoder followed by a decoder (*cf.* Section 2.1), but a distinguishing feature is that each layer in the encoder and decoder attends to their inputs to produce activations. Additionally, instead of a fixed pooling operation such as mean, our aggregating function $\mathrm{pool}(\cdot)$ is parameterized and can thus adapt to the problem at hand.

3.1. Permutation Equivariant (Induced) Set Attention Blocks

We begin by defining our attention-based set operations, which we call SAB and ISAB. While existing pooling methods for sets obtain instance features independently of other instances, we use self-attention to concurrently encode the whole set. This gives the Set Transformer the ability to compute pairwise as well as higher-order interactions among instances during the encoding process. For this purpose, we adapt the multihead attention mechanism used in Transformer. We emphasize that all blocks introduced here are neural network blocks with their own parameters, and not fixed functions.

Given matrices $X, Y \in \mathbb{R}^{n \times d}$ which represent two sets of d-dimensional vectors, we define the Multihead Attention Block (MAB) with parameters ω as follows:

$$\mathrm{MAB}(X, Y) = \mathrm{LayerNorm}(H + \mathrm{rFF}(H)), \tag{6}$$

$$\text{where } H = \mathrm{LayerNorm}(X + \mathrm{Multihead}(X, Y, Y; \omega)), \tag{7}$$

rFF is any row-wise feedforward layer (i.e., it processes each instance independently and identically), and $\mathrm{LayerNorm}$ is layer normalization (Ba et al., 2016). The MAB is an adaptation of the encoder block of the Transformer (Vaswani et al., 2017) without positional encoding and dropout. Using the MAB, we define the Set Attention Block (SAB) as

$$\mathrm{SAB}(X) \coloneqq \mathrm{MAB}(X, X). \tag{8}$$

In other words, an SAB takes a set and performs self-attention between the elements in the set, resulting in a set of equal size. Since the output of SAB contains information about pairwise interactions among the elements in the input set X, we can stack multiple SABs to encode higher order interactions. Note that while the SAB (8) involves a multihead attention operation (7), where $Q = K = V = X$, it could reduce to applying a residual block on X. In practice, it learns more complicated functions due to linear projections of X inside attention heads, (3) and (5).

A potential problem with using SABs for set-structured data is the quadratic time complexity $\mathcal{O}(n^2)$, which may be too expensive for large sets ($n \gg 1$). We thus introduce the *Induced Set Attention Block* (ISAB), which bypasses this problem. Along with the set $X \in \mathbb{R}^{n \times d}$, additionally define m d-dimensional vectors $I \in \mathbb{R}^{m \times d}$, which we call *inducing points*. Inducing points I are part of the ISAB itself, and they are *trainable parameters* which we train along with other parameters of the network. An ISAB with m inducing points I is defined as:

$$\mathrm{ISAB}_m(X) = \mathrm{MAB}(X, H) \in \mathbb{R}^{n \times d}, \tag{9}$$

$$\text{where } H = \mathrm{MAB}(I, X) \in \mathbb{R}^{m \times d}. \tag{10}$$

The ISAB first transforms I into H by attending to the input set. The set of transformed inducing points H, which

contains information about the input set X, is again attended to by the input set X to finally produce a set of n elements. This is analogous to low-rank projection or autoencoder models, where inputs (X) are first projected onto a low-dimensional object (H) and then reconstructed to produce outputs. The difference is that the goal of these methods is reconstruction whereas ISAB aims to obtain good features for the final task. We expect the learned inducing points to encode some global structure which helps explain the inputs X. For example, in the amortized clustering problem on a 2D plane, the inducing points could be appropriately distributed points on the 2D plane so that the encoder can compare elements in the query dataset indirectly through their proximity to these grid points.

Note that in (9) and (10), attention was computed between a set of size m and a set of size n. Therefore, the time complexity of $\mathrm{ISAB}_m(X;\lambda)$ is $\mathcal{O}(nm)$ where m is a (typically small) hyperparameter — an improvement over the quadratic complexity of the SAB. We also emphasize that both of our set operations (SAB and ISAB) are *permutation equivariant* (definition in Section 2.1):

Property 1. *Both* $\mathrm{SAB}(X)$ *and* $\mathrm{ISAB}_m(X)$ *are permutation equivariant.*

3.2. Pooling by Multihead Attention

A common aggregation scheme in permutation invariant networks is a dimension-wise average or maximum of the feature vectors (*cf.* Section 1). We instead propose to aggregate features by applying multihead attention on a learnable set of k seed vectors $S \in \mathbb{R}^{k\times d}$. Let $Z \in \mathbb{R}^{n\times d}$ be the set of features constructed from an encoder. *Pooling by Multihead Attention* (PMA) with k seed vectors is defined as

$$\mathrm{PMA}_k(Z) = \mathrm{MAB}(S, \mathrm{rFF}(Z)). \tag{11}$$

Note that the output of PMA_k is a set of k items. We use one seed vector ($k = 1$) in most cases, but for problems such as amortized clustering which requires k correlated outputs, the natural thing to do is to use k seed vectors. To further model the interactions among the k outputs, we apply an SAB afterwards:

$$H = \mathrm{SAB}(\mathrm{PMA}_k(Z)). \tag{12}$$

We later empirically show that such self-attention after pooling helps in modeling explaining-away (e.g., among clusters in an amortized clustering problem).

Intuitively, feature aggregation using attention should be beneficial because the influence of each instance on the target is not necessarily equal. For example, consider a problem where the target value is the maximum value of a set of real numbers. Since the target can be recovered using only a single instance (the largest), finding and attending to that instance during aggregation will be advantageous.

3.3. Overall Architecture

Using the ingredients explained above, we describe how we would construct a set transformer consists of an encoder and a decoder. The encoder $\mathrm{Encoder} : X \mapsto Z \in \mathbb{R}^{n\times d}$ is a stack of SABs or ISABs, for example:

$$\mathrm{Encoder}(X) = \mathrm{SAB}(\mathrm{SAB}(X)) \tag{13}$$

$$\mathrm{Encoder}(X) = \mathrm{ISAB}_m(\mathrm{ISAB}_m(X)). \tag{14}$$

We point out again that the time complexity for ℓ stacks of SABs and ISABs are $\mathcal{O}(\ell n^2)$ and $\mathcal{O}(\ell nm)$, respectively. This can result in much lower processing times when using ISAB (as compared to SAB), while still maintaining high representational power. After the encoder transforms data $X \in \mathbb{R}^{n\times d_x}$ into features $Z \in \mathbb{R}^{n\times d}$, the decoder aggregates them into a single or a set of vectors which is fed into a feed-forward network to get final outputs. Note that PMA with $k > 1$ seed vectors should be followed by SABs to model the correlation between k outputs.

$$\mathrm{Decoder}(Z;\lambda) = \mathrm{rFF}(\mathrm{SAB}(\mathrm{PMA}_k(Z))) \in \mathbb{R}^{k\times d} \tag{15}$$

$$\text{where } \mathrm{PMA}_k(Z) = \mathrm{MAB}(S, \mathrm{rFF}(Z)) \in \mathbb{R}^{k\times d}, \tag{16}$$

3.4. Analysis

Since the blocks used to construct the encoder (i.e., SAB, ISAB) are permutation equivariant, the mapping of the encoder $X \to Z$ is permutation equivariant as well. Combined with the fact that the PMA in the decoder is a permutation invariant transformation, we have the following:

Proposition 1. *The Set Transformer is permutation invariant.*

Being able to approximate any function is a desirable property, especially for black-box models such as deep neural networks. Building on previous results about the universal approximation of permutation invariant functions, we prove the universality of Set Transformers:

Proposition 2. *The Set Transformer is a universal approximator of permutation invariant functions.*

Proof. See supplementary material. □

4. Related Works

Pooling architectures for permutation invariant mappings Pooling architectures for sets have been used in various problems such as 3D shape recognition (Shi et al., 2015; Su et al., 2015), discovering causality (Lopez-Paz et al., 2017), learning the statistics of a set (Edwards & Storkey, 2017), few-shot image classification (Snell et al., 2017), and conditional regression and classification (Garnelo et al., 2018). Zaheer et al. (2017) discuss the structure

in general and provides a partial proof of the universality of the pooling architecture, and Wagstaff et al. (2019) further discuss the limitation of pooling architectures. Bloem-Reddy & Teh (2019) provides a link between probabilistic exchangeability and pooling architectures.

Attention-based approaches for sets Several recent works have highlighted the competency of attention mechanisms in modeling sets. Vinyals et al. (2016) pool elements in a set by a weighted average with weights computed using an attention mechanism. Yang et al. (2018) propose AttSets for multi-view 3D reconstruction, where dot-product attention is applied to compute the weights used to pool the encoded features via weighted sums. Similarly, Ilse et al. (2018) use attention-based weighted sum-pooling for multiple instance learning. Compared to these approaches, ours use multihead attention in aggregation, and more importantly, we propose to apply self-attention after pooling to model correlation among multiple outputs. PMA with $k = 1$ seed vector and single-head attention roughly corresponds to these previous approaches. Although not permutation invariant, Mishra et al. (2018) has attention as one of its core components to meta-learn to solve various tasks using sequences of inputs. Kim et al. (2019) proposed attention-based conditional regression, where self-attention is applied to the query sets.

Modeling interactions between elements in sets An important reason to use the Transformer is to explicitly model higher-order interactions among the elements in a set. Santoro et al. (2017) propose the relational network, a simple architecture that sum-pools all pairwise interactions of elements in a given set, but not higher-order interactions. Similarly to our work, Ma et al. (2018) use the Transformer to model interactions between the objects in a video. They use mean-pooling to obtain aggregated features which they fed into an LSTM.

Inducing point methods The idea of letting trainable vectors I directly interact with data points is loosely based on the inducing point methods used in sparse Gaussian processes (Snelson & Ghahramani, 2005) and the Nyström method for matrix decomposition (Fowlkes et al., 2004). m trainable inducing points can also be seen as m independent memory cells accessed with an attention mechanism. The differential neural dictionary (Pritzel et al., 2017) stores previous experience as key-value pairs and uses this to process queries. One can view the ISAB is the inversion of this idea, where queries I are stored and the input features are used as key-value pairs.

5. Experiments

To evaluate the Set Transformer, we apply it to a suite of tasks involving sets of data points. We repeat all experiments five times and report performance metrics evaluated on corresponding test datasets. Along with baselines, we compared various architectures arising from the combination of the choices of having attention in encoders and decoders. Unless specified otherwise, "simple pooling" means average pooling.

Table 1. Mean absolute errors on the max regression task.

Architecture	MAE
rFF + Pooling (mean)	2.133 ± 0.190
rFF + Pooling (sum)	1.902 ± 0.137
rFF + Pooling (max)	$\mathbf{0.1355 \pm 0.0074}$
SAB + PMA (ours)	0.2085 ± 0.0127

- rFF + Pooling (Zaheer et al., 2017): rFF layers in encoder and simple pooling + rFF layers in decoder.
- rFFp-mean/rFFp-max + Pooling (Zaheer et al., 2017): rFF layers with permutation equivariant variants in encoder (Zaheer et al., 2017, (4)) and simple pooling + rFF layers in decoder.
- rFF + Dotprod (Yang et al., 2018; Ilse et al., 2018): rFF layers in encoder and dot product attention based weighted sum pooling + rFF layers in decoder.
- SAB (ISAB) + Pooling (ours): Stack of SABs (ISABs) in encoder and simple pooling + rFF layers in decoder.
- rFF + PMA (ours): rFF layers in encoder and PMA (followed by stack of SABs) in decoder.
- SAB (ISAB) + PMA (ours): Stack of SABs (ISABs) in encoder and PMA (followed by stack of SABs) in decoder.

5.1. Toy Problem: Maximum Value Regression

To demonstrate the advantage of attention-based set aggregation over simple pooling operations, we consider a toy problem: regression to the maximum value of a given set. Given a set of real numbers $\{x_1, \ldots, x_n\}$, the goal is to return $\max(x_1, \cdots, x_n)$. Given prediction p, we use the mean absolute error $|p - \max(x_1, \cdots, x_n)|$ as the loss function. We constructed simple pooling architectures with three different pooling operations: $\max$, mean, and sum. We report loss values after training in Table 1. Mean- and sum-pooling architectures result in a high mean absolute error (MAE). The model with max-pooling can predict the output perfectly by learning its encoder to be an identity function, and thus achieves the highest performance. Notably, the Set Transformer achieves performance comparable to the max-pooling model, which underlines the importance of additional flexibility granted by attention mechanisms — it can learn to find and attend to the maximum element.

Figure 2. Counting unique characters: this is a randomly sampled set of 20 images from the Omniglot dataset. There are 14 different characters inside this set.

Table 2. Accuracy on the unique character counting task.

Architecture	Accuracy
rFF + Pooling	0.4382 ± 0.0072
rFFp-mean + Pooling	0.4617 ± 0.0076
rFFp-max + Pooling	0.4359 ± 0.0077
rFF + Dotprod	0.4471 ± 0.0076
rFF + PMA (ours)	0.4572 ± 0.0076
SAB + Pooling (ours)	0.5659 ± 0.0077
SAB + PMA (ours)	$\mathbf{0.6037 \pm 0.0075}$

5.2. Counting Unique Characters

In order to test the ability of modelling interactions between objects in a set, we introduce a new task of counting unique elements in an input set. We use the Omniglot (Lake et al., 2015) dataset, which consists of 1,623 different handwritten characters from various alphabets, where each character is represented by 20 different images.

We split all characters (and corresponding images) into train, validation, and test sets and only train using images from the train character classes. We generate input sets by sampling between 6 and 10 images and we train the model to predict the number of different characters inside the set. We used a Poisson regression model to predict this number, with the rate λ given as the output of a neural network. We maximized the log likelihood of this model using stochastic gradient ascent.

We evaluated model performance using sets of images sampled from the test set of characters. Table 2 reports accuracy, measured as the frequency at which the mode of the Poisson distribution chosen by the network is equal to the number of characters inside the input set.

We additionally performed experiments to see how the number of incuding points affects performance. We trained $\text{ISAB}_n + \text{PMA}$ on this task while varying the number of inducing points (n). Accuracies are shown in Figure 3, where other architectures are shown as horizontal lines for comparison. Note first that even the accuracy of $\text{ISAB}_1 + \text{PMA}$ surpasses that of both $\text{rFF} + \text{Pooling}$ and $\text{rFF} + \text{PMA}$, and that performance tends to increase as we increase n.

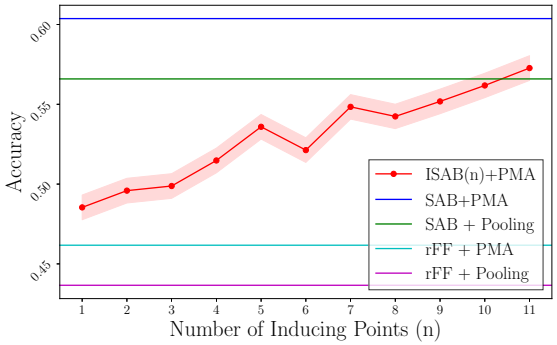


Figure 3. Accuracy of $\text{ISAB}_n + \text{PMA}$ on the unique character counting task. x-axis is n and y-axis is accuracy.

5.3. Amortized Clustering with Mixture of Gaussians

We applied the set-input networks to the task of maximum likelihood of mixture of Gaussians (MoGs). The log-likelihood of a dataset $X = \{x_1, \ldots, x_n\}$ generated from an MoG with k components is

$$\log p(X;\theta) = \sum_{i=1}^{n} \log \sum_{j=1}^{k} \pi_j \mathcal{N}(x_i; \mu_j, \text{diag}(\sigma_j^2)). \quad (17)$$

The goal is to learn the optimal parameters $\theta^*(X) = \arg\max_\theta \log p(X;\theta)$. The typical approach to this problem is to run an iterative algorithm such as Expectation-Maximisation (EM) until convergence. Instead, we aim to learn a generic meta-algorithm that directly maps the input set X to $\theta^*(X)$. One can also view this as amortized maximum likelihood learning. Specifically, given a dataset X, we train a neural network to output parameters $f(X;\lambda) = \{\pi(X), \{\mu_j(X), \sigma_j(X)\}_{j=1}^k\}$ which maximize

$$\mathbb{E}_X \left[\sum_{i=1}^{|X|} \log \sum_{j=1}^{k} \pi_j(X) \mathcal{N}(x_i; \mu_j(X), \text{diag}(\sigma_j^2(X))) \right]. \quad (18)$$

We structured $f(\cdot;\lambda)$ as a set-input neural network and learned its parameters λ using stochastic gradient ascent, where we approximate gradients using minibatches of *datasets*.

We tested Set Transformers along with other set-input networks on two datasets. We used four seed vectors for the PMA ($S \in \mathbb{R}^{4\times d}$) so that each seed vector generates the parameters of a cluster.

Synthetic 2D mixtures of Gaussians: Each dataset contains $n \in [100, 500]$ points on a 2D plane, each sampled from one of four Gaussians.

CIFAR-100: Each dataset contains $n \in [100, 500]$ images sampled from four random classes in the CIFAR-100 dataset. Each image is represented by a 512-dim vector obtained from a pretrained VGG network (Simonyan & Zisserman, 2014).

Table 3. Meta clustering results. The number inside parenthesis indicates the number of inducing points used in ISABs of encoders. We show average likelihood per data for the synthetic dataset and the adjusted rand index (ARI) for the CIFAR-100 experiment. LL1/data, ARI1 are the evaluation metrics after a single EM update step. The oracle for the synthetic dataset is the log likelihood of the actual parameters used to generate the set, and the CIFAR oracle was computed by running EM until convergence.

	Synthetic		CIFAR-100	
Architecture	LL0/data	LL1/data	ARI0	ARI1
Oracle	-1.4726		0.9150	
rFF + Pooling	-2.0006 $\pm$ 0.0123	-1.6186 $\pm$ 0.0042	0.5593 $\pm$ 0.0149	0.5693 $\pm$ 0.0171
rFFp-mean + Pooling	-1.7606 $\pm$ 0.0213	-1.5191 $\pm$ 0.0026	0.5673 $\pm$ 0.0053	0.5798 $\pm$ 0.0058
rFFp-max + Pooling	-1.7692 $\pm$ 0.0130	-1.5103 $\pm$ 0.0035	0.5369 $\pm$ 0.0154	0.5536 $\pm$ 0.0186
rFF + Dotprod	-1.8549 $\pm$ 0.0128	-1.5621 $\pm$ 0.0046	0.5666 $\pm$ 0.0221	0.5763 $\pm$ 0.0212
SAB + Pooling (ours)	-1.6772 $\pm$ 0.0066	-1.5070 $\pm$ 0.0115	0.5831 $\pm$ 0.0341	0.5943 $\pm$ 0.0337
ISAB (16) + Pooling (ours)	-1.6955 $\pm$ 0.0730	-1.4742 $\pm$ 0.0158	0.5672 $\pm$ 0.0124	0.5805 $\pm$ 0.0122
rFF + PMA (ours)	-1.6680 $\pm$ 0.0040	-1.5409 $\pm$ 0.0037	0.7612 $\pm$ 0.0237	0.7670 $\pm$ 0.0231
SAB + PMA (ours)	-1.5145 $\pm$ 0.0046	-1.4619 $\pm$ 0.0048	0.9015 $\pm$ 0.0097	0.9024 $\pm$ 0.0097
ISAB (16) + PMA (ours)	**-1.5009 $\pm$ 0.0068**	**-1.4530 $\pm$ 0.0037**	**0.9210 $\pm$ 0.0055**	**0.9223 $\pm$ 0.0056**

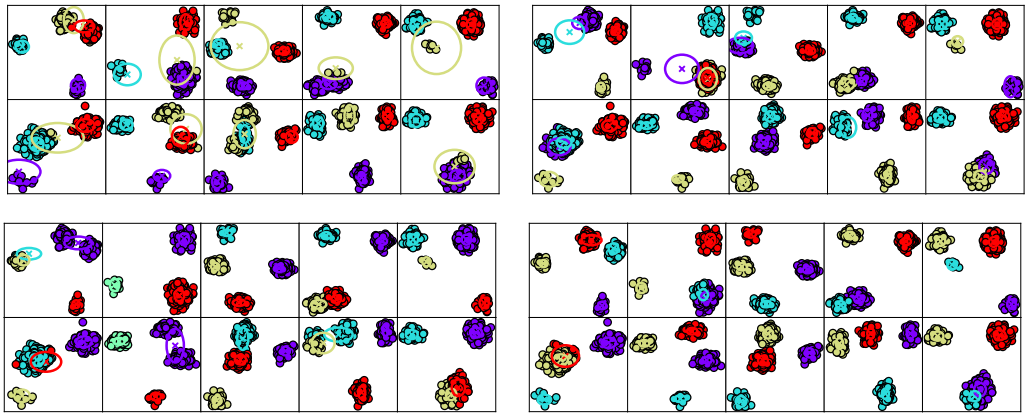


Figure 4. Clustering results for 10 test datasets, along with centers and covariance matrices. rFF+Pooling (top-left), SAB+Pooling (top-right), rFF+PMA (bottom-left), Set Transformer (bottom-right). Best viewed magnified in color.

We report the performance of the oracle along with the set-input neural networks in Table 3. We additionally report scores of all models after a single EM update. Overall, the Set Transformer found accurate parameters and even outperformed the oracles after a single EM update. This may be due to the relatively small size of the input sets; some clusters have fewer than 10 points. In this regime, sample statistics can differ substantially from population statistics, which limits the performance of the oracle while the Set Transformer can adapt accordingly. Notably, the Set Transformer with only 16 inducing points showed the best performance, even outperforming the full Set Transformer. We believe this is due to the knowledge transfer and regularization via inducing points, helping the network to learn global structures. Our results also imply that the improvement from using the PMA is more significant than that of the SAB, supporting our claim of the importance of attention-based decoders. We provide detailed generative processes, network architectures, and training schemes along with additional experiments with various numbers of inducing points in the supplementary material.

5.4. Set Anomaly Detection

We evaluate our methods on the task of meta-anomaly detection within a set using the CelebA dataset. The dataset consists of 202,599 images with the total of 40 attributes. We randomly sample 1,000 sets of images. For every set, we select two attributes at random and construct the set by selecting seven images containing both attributes and one image with neither. The goal of this task is to find the image that does not belong to the set. We give a detailed description of the experimental setup in the supplementary material. We report the area under receiver operating characteristic curve (AUROC) and area under precision-recall curve (AUPR) in Table 5. Set Transformers outperformed all other methods by a significant margin.

Table 4. Test accuracy for the point cloud classification task using $100, 1000, 5000$ points.

Architecture	100 pts	1000 pts	5000 pts
rFF + Pooling (Zaheer et al., 2017)	-	0.83 ± 0.01	-
rFFp-max + Pooling (Zaheer et al., 2017)	0.82 ± 0.02	0.87 ± 0.01	$\mathbf{0.90 \pm 0.003}$
rFF + Pooling	0.7951 ± 0.0166	0.8551 ± 0.0142	0.8933 ± 0.0156
rFF + PMA (ours)	0.8076 ± 0.0160	0.8534 ± 0.0152	0.8628 ± 0.0136
ISAB (16) + Pooling (ours)	0.8273 ± 0.0159	$\mathbf{0.8915 \pm 0.0144}$	$\mathbf{0.9040 \pm 0.0173}$
ISAB (16) + PMA (ours)	$\mathbf{0.8454 \pm 0.0144}$	0.8662 ± 0.0149	0.8779 ± 0.0122

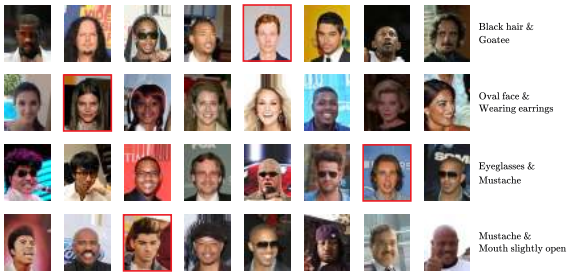


Figure 5. Sampled datasets. Each row is a dataset, consisting of 7 normal images and 1 anomaly (red box). In each subsampled dataset, a normal image has two attributes (rightmost column) which anomalies do not.

Table 5. Meta set anomaly results. Each architecture is evaluated using average of test AUROC and test AUPR.

Architecture	Test AUROC	Test AUPR
Random guess	0.5	0.125
rFF + Pooling	0.5643 ± 0.0139	0.4126 ± 0.0108
rFFp-mean + Pooling	0.5687 ± 0.0061	0.4125 ± 0.0127
rFFp-max + Pooling	0.5717 ± 0.0117	0.4135 ± 0.0162
rFF + Dotprod	0.5671 ± 0.0139	0.4155 ± 0.0115
SAB + Pooling (ours)	0.5757 ± 0.0143	0.4189 ± 0.0167
rFF + PMA (ours)	0.5756 ± 0.0130	0.4227 ± 0.0127
SAB + PMA (ours)	$\mathbf{0.5941 \pm 0.0170}$	$\mathbf{0.4386 \pm 0.0089}$

5.5. Point Cloud Classification

We evaluated Set Transformers on a classification task using the ModelNet40 (Chang et al., 2015) dataset[1], which contains three-dimensional objects in 40 different categories. Each object is represented as a point cloud, which we treat as a set of n vectors in $\mathbb{R}^3$. We performed experiments with input sets of size $n \in \{100, 1000, 5000\}$. Because of the large set sizes, MABs are prohibitively time-consuming due to their $\mathcal{O}(n^2)$ time complexity.

Table 4 shows classification accuracies. We point out that Zaheer et al. (2017) used significantly more engineering for the 5000 point experiment. For this experiment only, they augmented data (scaling, rotation) and used a different optimizer (Adamax) and learning rate schedule. Set Transformers were superior when given small sets, but were outperformed by ISAB (16) + Pooling on larger sets. First note that classification is harder when given fewer points. We think Set Transformers were outperformed in the problems with large sets because such sets already had sufficient information for classification, diminishing the need to model complex interactions among points. We point out that PMA outperformed simple pooling in all other experiments.

[1]The point-cloud dataset used in this experiment was obtained directly from the authors of Zaheer et al. (2017).

6. Conclusion

In this paper, we introduced the Set Transformer, an attention-based set-input neural network architecture. Our proposed method uses attention mechanisms for both encoding and aggregating features, and we have empirically validated that both of them are necessary for modelling complicated interactions among elements of a set. We also proposed an inducing point method for self-attention, which makes our approach scalable to large sets. We also showed useful theoretical properties of our model, including the fact that it is a universal approximator for permutation invariant functions. An interesting future work would be to apply Set Transformers to meta-learning problems. In particular, using Set Transformers to meta-learn posterior inference in Bayesian models seems like a promising line of research. Another exciting extension of our work would be to model the uncertainty in set functions by injecting noise variables into Set Transformers in a principled way.

Acknowledgments JL and YWT's research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 617071. JL has also received funding from EPSRC under grant EP/P026753/1. JL acknowledges support from IITP grant funded by the Korea government(MSIT) (No.2017-0-01779, XAI) and Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT1702-15.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv e-prints*, arXiv:1607.06450, 2016.

Bloem-Reddy, B. and Teh, Y.-W. Probabilistic symmetry and invariant neural networks. *arXiv e-prints*, arXiv:1901.06082, 2019.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. ShapeNet: An information-rich 3D model repository. *arXiv e-prints*, arXiv:1512.03012, 2015.

Charles, R. Q., Su, H., Kaichun, M., and Guibas, L. J. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Dietterich, T. G., Lathrop Richard, H., and Lozano-Pérez, T. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.

Edwards, H. and Storkey, A. Towards a neural statistician. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Fowlkes, C., Belongie, S., Chung, F., and Malik, J. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):215–225, 2004.

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. M. A. Conditional neural processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Graves, A., Mohamed, A.-r., and Hinton, G. E. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

Ilse, M., Tomczak, J. M., and Welling, M. Attention-based deep multiple instance learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive neural processes. In *Proceedings of International Conference on Learning Representations*, 2019.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Lee, Y. and Choi, S. Gradient-based meta-learning with learned layerwise metric and subspace. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Lopez-Paz, D., Nishihara, R., Chintala, S., Schölkopf, B., and Bottou, L. Discovering causal signals in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Ma, C.-Y., Kadav, A., Melvin, I., Kira, Z., AlRegib, G., and Peter Graf, H. Attend and interact: higher-order object interactions for video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Maron, O. and Lozano-Pérez, T. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1998.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Muandet, K., Fukumizu, K., Dinuzzo, F., and Schölkopf, B. Learning from distributions via support measure machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Oliva, J., Póczos, B., and Schneider, J. Distribution to distribution regression. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

Pritzel, A., Uria, B., Srinivasan, S., Puigdomenech, A., Vinyals, O., Hassabis, D., Wierstra, D., and Blundell, C. Neural episodic control. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Santoro, A., Raposo, D., Barret, D. G. T., Malinowski, M., Pascanu, R., and Battaglia, P. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Schmidhuber, J. *Evolutionary Principles in Self-Referential Learning*. PhD thesis, Technical University of Munich, 1987.

Shi, B., Bai, S., Zhou, Z., and Bai, X. DeepPano: deep panoramic representation for 3-D shape recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343, 2015.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv e-prints*, arXiv:1409.1556, 2014.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2005.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

Thrun, S. and Pratt, L. *Learning to Learn*. Kluwer Academic Publishers, 1998.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Vinyals, O., Bengio, S., and Kudlur, M. Order matters: sequence to sequence for sets. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

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

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3D ShapeNets: a deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Yang, B., Wang, S., Markham, A., and Trigoni, N. Attentional aggregation of deep feature sets for multi-view 3D reconstruction. *arXiv e-prints*, arXiv:1808.00758, 2018.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.