Attention-Based Set Embeddings for Point Cloud Classification

Shiyue Zhang[*]
Department of Computer Science
UNC Chapel Hill
shiyue@cs.unc.edu

Xiang Zhou[*]
Department of Computer Science
UNC Chapel Hill
xzh@cs.unc.edu

Abstract

Set-structured data is involved in many real-world tasks, such as anomaly detection and 3D objects modeling. Due to its less structured and unordered nature, using machine learning methods to represent a set of instances, i.e. *set embedding*, is always challenging. In this study, inspired by the widely-used successful *self-attention* mechanism, we proposed an *Attention-based Set Embedding* method. We expect that such structure can take advantage from the additional modeling capacity of attention mechanism while maintaining the *permutation invariance* as other existing set embedding models, and finally obtain better embedding ability. We test our model and its variants on a notable set embedding application: *point cloud classification*. However, according to the results of our experiments, the learned attention functions didn't demonstrate significant superiority over base functions.

1 Introduction

The majority of machine learning tasks are *instance-based*, taking an input vector and mapping it to the corresponding target. While most of those problems are dealing with *fixed-dimensional data* instances, such as 2D images, some have *set-structured data* involved, such as modeling 3D point clouds and anomaly detection. Unlike the fixed-dimensional data, in which the elements within on instance interact with each other based on their order, set elements do not have any order. In another word, permuting the sets should not affect the mapping results.

Due to the unique nature of set instance, to represent and analysis set-structured data using typical machine learning methods can be challenging, since many techniques in neural networks, such as *convolutional neural network (CNN)* and *recurrent neural network (RNN)* are order-sensitive. To address set modeling problems, the model should be *permutation invariant*, i.e. any permutation of a set input should lead to the same output. Second, the model should have the ability to handle the sets of different sizes, such as the point clouds with 100 or 1000 points. Above all, a good *set embedding* algorithm should be able to represent both the global structure of a set as well as useful local interactions and provide reasonable output results.

Recently, Zaheer et al. [2017] proposed a general framework for learning set embeddings. They use a combination of permutation invariant layers and permutation equivariant layers to capture and model the information in a set. While theoratically their model can approximate any function on sets, this framework is very difficult to learn the interaction between elements because such interaction can only be weakly captured by the global pooling operation.

In our work, we try to propose a better architecture for set embeddings, which not only meets the permutation invariant requirement and have the ability to handle different data size but also captures

[*]indicates equal contributions.

Final report.

complex information interaction within the set instances and provides decent output results. Specifically, we use *Self-Attention* mechanism for the model to learn a better set embedding and test different network structures. We've explored a number of different designs for the attention mechanism and different ways to combine the attention mechanism and other pooling, linear operations. We've tested our method on one of the most well-known set-structured problems: *point cloud classification*. However, the results show that the modeling ability doesn't have significant improvement with self-attention structure over some basic global pooling functions. Therefore, we draw the conclusion that learned attention functions don't show any superiority over base functions.

2 Related Works

Set embeddings A number of works have studied how to better model the permutation invariance of sets. Chen et al. [2014] uses pairwise coupling of features at the previous layer. Su et al. [2015], Shi et al. [2015] uses pooling for set embeddings. Recently, Zaheer et al. [2017] proposes a fundamental structure called *Deep Sets* for learning the set embeddings, they also studied the property of permutation equivariance in their work. However, none of these methods explicitly considers and models the complex interactions among elements in a set.

Attention mechanism The attention mechanism is achieved great performance in tasks such as machine translation (Bahdanau et al. [2014], Vaswani et al. [2017]), machine comprehension (Seo et al. [2016]), etc. Vinyals et al. [2015] first applies the attention mechanism to learn set embeddings by modeling the set as a sequence and use attention to calculate weighted sum over the whole sequence. Attention mechanism has also been applied to other relevant tasks such as multi-view 3D reconstruction (Yang et al. [2018]) and multiple instance learning (Ilse et al. [2018]). Our project come across a concurrent preprint Lee et al. [2018], in which they also apply self-attention for learning better set embeddings.

Point cloud classification A point cloud is a set of points that represents a 3D object. The target of point cloud classification is to identify the shape of a point cloud, which is frequently encountered in various applications like robotics, vision, and cosmology. Qi et al. [2017a] and Qi et al. [2017b] proposed a hierarchical feature learning network that contains multiple sampling and grouping layers to capture the local and global information of point clouds. To enable the permutation invariant feature, they augmented the data with all kinds of permutations. Fan et al. [2017] and Lin et al. [2017] studied the latent set-related issues occurred during point cloud generating via variational models. These works achieved good performance on point cloud classification. However, most of them used well-designed and relatively complex deep neural networks, which is difficult to be generalized to other set embedding tasks. Zaheer et al. [2017] also tested their *Deep Sets* model on the Point cloud classification task. Using a simple three-layer neural network, they gave a very decent performance on this task. In our work, we mainly follow the "Deep Sets" idea and leverage the attention mechanism to fuse more global and local information for learning set embeddings.

3 Background

3.1 Set embedding baselines

Deep Sets The state-of-the-art model of set embedding problems is Deep Sets proposed by Zaheer et al. [2017]. This model contains two main sub-structures. The first part is the *permutation equivariant* layer, written as Eq. 1, where γ is a feed-forward neural network and $pool$ is any pooling function, such as max, mean, sum, etc. This layer is equivariant to any permutation of instances in the input X, i.e. change the order of instances in X, the order of outputs will change accordingly. By stacking multiple *permutation equivariant* layers, the model learns the set context information and embeds each instance in X.

$$f_{equ}(x_i) = \gamma(x_i - pool_j(x_j)) \tag{1}$$

The second part is the *permutation invariant* layer, given by Eq. 2, which aggregates instance embeddings $g(x_i)$ to a set embedding $f(X)$ via a pooling function, and ρ is another feed-forward neural network.

$$f_{inv}(X) = \rho(pool_i(f_{equ}(x_i))) \tag{2}$$

The set embedding $f(X)$ can be easily used for downstream tasks by using additional neural networks. In this work, we take Deep Sets as our main baseline model.

Set Transformer Set Transformer, proposed by Lee et al. [2018], is the first work that tried to improve Deep Sets architecture with self-attention mechanism. For the *permutation equivariant* layer, they both investigated Set Attention Block (SAB) that uses the same self-attention structure proposed by Vaswani et al. [2017] and proposed a novel Introduced Set Attention Block (ISAB) to reduce computational complexity by applying a low-rank projection. For the *permutation invariant* layer, instead of using basic pooling fuctions, they proposed a Pooling by Multihead Attention (PMA) layer. Our main difference with this work is that we used a simpler self-attention structure which can keep the attributes of the inputs, see detailed descriptions in Section 4.3. We will also compare with this model in our experiments.

3.2 Self-Attention

The self-attention mechanism was first proposed by Vaswani et al. [2017] for Neural Machine Translation (NMT) task. It was proved to be a successful technique for aggregating context information and boomed the performance of many other tasks in NLP and CV areas, such as question answering (Hu et al. [2017]), image recognition (Wang et al. [2018]), speech recognition and sentence parsing Kaiser et al. [2017]), etc.

The general attention mechanism can be expressed by Eq. 3. Given n query vectors $Q \in \mathbb{R}^{n*d_q}$, keys $K \in \mathbb{R}^{n*d_q}$ and values $V \in \mathbb{R}^{n*d_v}$ matrices. It first computes the dot product of the query and with all keys, divided each by $\sqrt{d_q}$, which can be seen as the "similarity" between query and keys; then applies a softmax function to normalize the similarity scores to get the weights on values; finally, aggregates the values by the attention weights.

$$Att(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_q}})V \tag{3}$$

Self-attention mechanism is just a special case of the general attention mechanism. Instead of taking Q, K, V from different sources, in Self-attention, Q, K, V are all linearly transformed from input X, i.e. $Q = W_q X$.

Obviously, the self-attention operation is also *permutation equivariant*. Therefore, the self-attention structure can be used as an advanced alternative of the basic pooling function in the Deep Sets architecture.

4 Attention-based Set Embedding

4.1 Network structure

Similar to Deep Sets (Zaheer et al. [2017]), our attention-based model consists of two different types of layers: *permutation equivariant layers* and *permutation invariant layers*. Our model differs from their approach in that in the equivariant layers, besides using *maxpool* to fuse global information like Deep Sets, we introduce attention-based permutation equivariant layers, aiming to capture subtle interactions between each elements in a set. For the invariant layers, we also explore different types of pooling, aggregating global features. These two submodules will be discussed in detail in 4.2.

As is shown in Figure 1, we first use an element-wise feature extractor to map each input representing one element in the set to a feature vector. Then, the feature vectors go through several equivariant layers. Finally, the invariant layer aggregates all the information from every element and sends the resulted set embedding to a regressor specifically designed upon downstream tasks.

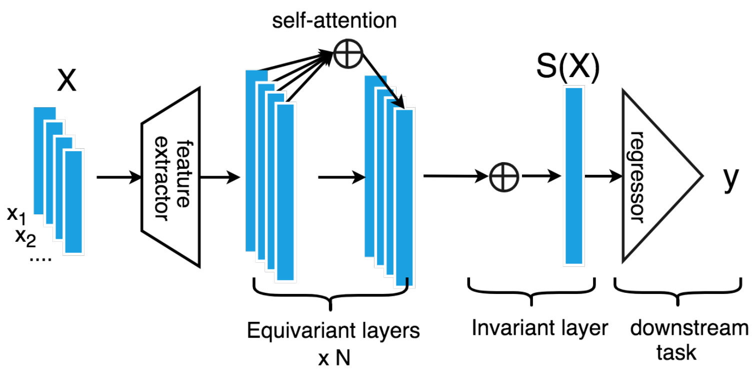

Figure 1: Overall architecture of attention-based set embedding

4.2 Sub-Modules

4.2.1 Equivariant layers

An attention-based equivariant layer can be expressed by Eq. 4. For each d-dimension element vector $x_i \in \mathbb{R}^d$ in the set $X = \{x_1, x_2, \ldots, x_M\}$, a pooling function $pool_i\phi$ is applied to obtain its *correlation vector* with other elements x_j of the set. Besides basic mean and max pooling, here we've also tried to use attention for pooling. A combination function g is introduced to combine correlation vector $pool_i\phi(x_i, x_j)$ with element vector x_i and produces a new feature vector f_{equ} for each element. It is easy to show that this layer is permutation equivariant, since the object is element individuals, and the pooling operation makes the pooling function invariant.

$$f_{equ}(x_i) = g(x_i, pool_j\phi(x_i, x_j)) \tag{4}$$

We choose to use the same size for the input and output vectors of our attention-based equivariant layer so that a number of these layers can be stacked together for better modeling capacity. There are multiple options for the pooling function $pool_i\phi(x_i, x_j)$ and combination function g. Here we list our choices in this project.

Pooling function For the pooling function, we tried both basic pooling and pooling by attention. Our choices are demonstrated below:

- $\max_j x_j$
- $\operatorname{mean}_j x_j$
- $attn(k, X)$
- $attn(x_i, X)$

For the first two options, the $\max$ and mean operations are basic pooling operations that can obtain global information among the set. For the last two options, we introduce a *pooling by attention* mechanism. Here we adopt the self-attention function in Section 3.3, using attention function in Eq. 3. For $attn(k, X)$, we introduce another trainable parameter $k \in \mathbb{R}^d$, which attends to all the elements and we use the weighted sum of those elements as the learned correlation vector. In $attn(x_i, X)$, we first calculate a standard self-attention which outputs another $M \times d$ matrix, and then we use a pooling operation to aggregate these information into one single correlation vector. The *pooling by attention* mechanism is expected to have the ability to capture the elements' interaction depending on the feature of objective element x_i. All the functions above provide a $\mathbb{R}^d$ dimension correlation vector.

Combination function For the combination function $g(a, b)$, we tried three different options to combine the correlation vector with the element feature vector x_i. From the inspiration of the residual connections, we use subtraction. We also use dot product to mimic the gate mechanism. In addition, we've also tried to concatenate these two vectors. The choices for $g(a, b)$ are listed below:

- subtraction $a - b$
- concatenation $[a, b]$
- dot production $a * b$

4.2.2 Invariant layer

The invariant layer can be formalized as Eq. 5.

$$f_{inv}(X) = pool_i f_{equ}(x_i) \tag{5}$$

The invariant layer is designed to aggregate global information and output the set embedding for downstream task. Thus, the pooling operation is introduced among all the feature vectors x_i in set X. For the invariant layer, we choose to use the following three implementations for $pool_i$:

- $\max\limits_i f_{equ}(x_i)$
- $\operatorname*{mean}\limits_i f_{equ}(x_i)$
- $attn(k, f_{equ}(x_i))$

4.3 Different Designs for Self-Attention Layers

The attention mechanism we use in our model follows the pipeline introduced in Section 3.3. In our work, we also explore different implementations of the attention mechanism as well as different attention spans. Apart from the standard transformer version self-attention in Eq. 3 which uses the dot production of the linear projection of the query and key vectors as the similarity score, we've tried some different parameterization as is shown in Eq. 6.

$$Att_{extension}(Q, K, V) = w(\rho(Q, K))V \tag{6}$$

In practice, the attention function usually takes two input feature vectors a and b. Here we've tried to either keep or remove the linear projection before a and b. Several variants to compute attention weights for $\rho(a, b)$ are listed below, where W, U and v are trainable parameters. We've also tried the original attention used in Bahdanau et al. [2014]. All the structures we used are listed as follows:

- $a^T b$
- $a^T W b$
- $(Ua)^T W b$
- $v * \tanh(Ua + Wb)$

We've also tried to change the span of the attention. Besides attending to all the elements in the set, the attention can also be calculated only on a local subset. We use *local attention* to refer to this variation. For the *local attention*, we first try to only uses the top k attention vectors and mask out the other vectors, which is called *top-K% local attention.*. We've also tried to first calculated the distance between each points in the original Euclidean space and only attend to the top $K\%$ closest points, which is called *localized K% nearest attention*

5 Experiments

Following the settings in Deep Sets, here we use the ModelNet40 dataset (Wu et al. [2015]) which contains 3D mesh models from 40 shape classes, and each class is divided into training and testing sets. Point clouds with 100 and 1000 points in each object are sampled from the surface of mesh models. In experiment, we run the full training and testing circle for five times, using the average classification accuracy and standard deviation as evaluation metrics.

Table 1: Point cloud classification Accuracy (%) of Deep Sets.

		Num of points	
Equ layer	Inv layer	100	1000
max	max	85 ± 0.5	88 ± 0.4
mean	max	84 ± 0.4	89 ± 0.1
max	mean	83 ± 0.4	87 ± 0.3
max	sum	77 ± 0.5	66 ± 0.4

Table 2: Point cloud classification Accuracy (%) of attention-based models.

		Num of points	
	Architecture	100	1000
Deep Sets	max pooling for all layers	85 ± 0.5	88 ± 0.4
Set Transformer[2]	SAB	67 ± 1.4	\
	ISAB	60 ± 1.7	\
Pooling Function	$attn(k, x_j)$	84 ± 0.3	88 ± 0.3
	$attn(x_i, X)$	**85 ± 0.2**	**88 ± 0.4**
Combination Function	subtraction $a - b$	**85 ± 0.2**	**88 ± 0.4**
	concatenation $[a, b]$	82 ± 0.4	86 ± 0.3
	dot production $a * b$	67 ± 0.5	72 ± 0.6
Attention Type	$a^T b$	78 ± 0.2	86 ± 0.2
	$a^T W b$	84 ± 0.5	88 ± 0.4
	$(Ua)^T W b$	**85 ± 0.2**	**88 ± 0.4**
	$v * \tanh(Ua + Wb)$	73 ± 0.6	\
Local Attention	max	72 ± 0.8	75 ± 0.9
	top 10%	74 ± 1.2	77 ± 1.2
	top 20%	73 ± 1.0	78 ± 0.5
	localized 10% nearest attention	49 ± 5.1	\
Invariant Layer	max	**85 ± 0.2**	**88 ± 0.4**
	$attn(k, f_{equ}(x_i))$	83 ± 0.4	86 ± 0.4
Hybrid	$x' = x - max_i, attn(x'_i, X')$	84 ± 0.3	88 ± 0.3

We use the same baseline setting as Deep Sets: pooling function in permutation equivariant layers is $\max$ or mean pooling, denoted as "Equ layer", with combination function $g(a, b) = a - b$; invariant layer is $\max$, mean pooling or sum, denoted as "Inv layer". Three equivariant layers are stacked. The baseline results are shown in Table 1.

For our attention-based embedding model, we test several combinations of equivariant and invariant layers as well as different attention modules mentioned in Section 4. The results are summarized in Table 2. The basic structure of attention-based model follows the structure combination: pooling function $pool_i \phi(x_i, x_j) = attn(x_i, X)$, combination function $g(a, b) = a - b$, invariant layer $pool_i = \max_i$, attention type $\rho(a, b) = (Ua)^T W b$, and three equivariant layers are stacked. For each section in result Table 2, only one part of the structure is changed, e.g. in *combination function* section different types of combination functions are tested, and the rest parts of the model follow the structure above. Specifically, in *hybrid* section, we combine max pooling and self-attention.

As shown in Table 2, although some of the variants give lower standard deviation, none of our attention-based model can beat the average classification accuracy of Deep Sets. We also observed that applying local attention in equivariant layer will impair the modeling capacity significantly, which suggests that the classification task mostly relies on the information that can represent the global structure of the set rather that subtle local structure of the point cloud object. Since our model can't outperform the baseline model, we draw the conclusion that the attention mechanism, whose strength is to capture the subtle context information, doesn't help the set classification task.

6 Conclusion

In this work, we explore different architectures for *Attention-based Set Embedding* models. Inspired by previous work on modeling order-invariant set-structured data, our model consists of two main sub-structure: *equivariant layers* and *invariant layer*, capturing context information and aggregating global feature, with generalization ability for different downstream tasks. We introduce *self-attention mechanism* across our architecture and design various of network structure for better modeling ability. We test our baseline model and variants on *Point Cloud Classification* task. Although ideally, our model has stronger capability in modeling the interaction between different elements in the set, the performance of our best proposed structure is still very similar to the baseline model. According to these experiment results, we conclude that for learning set embeddings, the self-attention mechanism doesn't bring any significant improvements over the basic pooling functions used in Deep Sets.

[2] Note that our reproduction of Set Transformer baseline presents very poor performance which is not consistent with the scores reported in their paper. Since they did not open their source code, this inconsistency may result from some different implementation details.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Xu Chen, Xiuyuan Cheng, and Stéphane Mallat. Unsupervised deep haar scattering on graphs. In *Advances in Neural Information Processing Systems*, pages 1709–1717, 2014.

Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, volume 2, page 6, 2017.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. Reinforced mnemonic reader for machine reading comprehension. *arXiv preprint arXiv:1705.02798*, 2017.

Maximilian Ilse, Jakub M Tomczak, and Max Welling. Attention-based deep multiple instance learning. *arXiv preprint arXiv:1802.04712*, 2018.

Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer. *arXiv preprint arXiv:1810.00825*, 2018.

Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. *arXiv preprint arXiv:1706.07036*, 2017.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017b.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343, 2015.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 4, 2018.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

Bo Yang, Sen Wang, Andrew Markham, and Niki Trigoni. Attentional aggregation of deep feature sets for multi-view 3d reconstruction. *arXiv preprint arXiv:1808.00758*, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.