PointNet with Self-Attention Layers

Bringer Leo
Department of Mechanical Engineering
University of Michigan
Ann Arbor, MI
lbringer@umich.edu

Kaifan Yue
Department of Robotics
University of Michigan
Ann Arbor, MI
kaifany@umich.edu

Xiang Li
Department of Electrical Engineering
University of Michigan
Ann Arbor, MI
forkobe@umich.edu

***Abstract*—Point cloud is an important type of geometric data structure that can represent 3D objects and scenes. It has gained popularity since the generalization of laser scanners and time-of-flight scanners in autonomous vehicles, consumer electronics, and industrial machines. While processing point clouds with machine learning methods is inherently challenging due to point cloud's irregularities and complexities, PointNet architecture has been developed to efficiently process point clouds by using multi-layer perceptrons (MLPs) and symmetric function. The architecture can be easily adapted to perform 3D recognition tasks such as classification, part segmentation, and semantic parsing, and achieve state-of-the-art performance. However, PointNet does not capture local structures induced by the metric space because of only applying elementary binary symmetric or aggregation operations, treating each points separately. This work aims to address this limitation by incorporating self-attention mechanism with positional encoding to capture local features. The extended architecture is able to achieve 3 percent increase in accuracy at 3D classification task, and detailed analysis of the architecture is provided.**

I. Introduction

Point cloud is an important type of geometric data structure that can represent 3D objects and scenes. An example of a point cloud is shown in Fig. 2. 3D point cloud processing and understanding are considerably more challenging than 2D images mainly due to the fact that point cloud samples are irregularly structured while 2D image samples (pixels) rely on a structured grid in the image plane with a regular spacing. Indeed, most conventional operators such as Convolutional Neural Network (CNN) require data to be ordered, regular and on a structured grid. Hence, early approaches that use CNN preprocessed the data into structured grid formats such as voxels. However, this method suffers from high memory consumption because it makes the data unnecessarily voluminous and sparse, while limiting its resolution thus also resulting in a loss of information. Moreover, since point sets are unordered they require the neural network to be permutation invariant and robust against rotation and translation to operate in the metric space. Also, since the points usually describe objects, the network needs to capture the spatial relations between them.

A pioneer solution that came up in 2017 by Qi et al. called PointNet [2], only used multi-layer perceptrons (MLPs), max-pooling and rigid transformations to ensure invariance under permutations and rotation by using only elementary binary symmetric operations and aggregation operations treating each

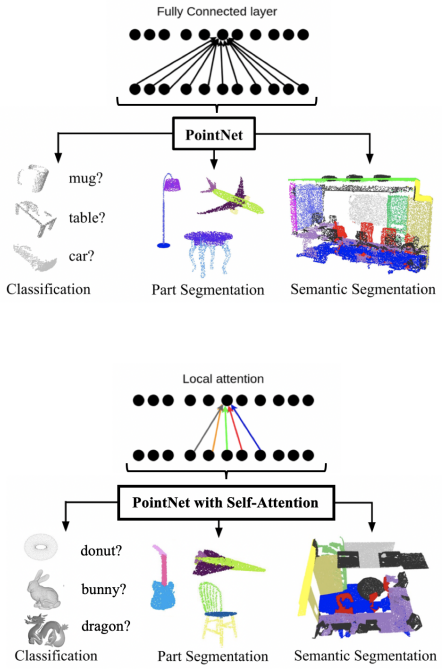


Fig. 1. Adding Self-Attention Layers to PointNet

point separately. This approach obtained a lot of the support from the community and has gathered an increasing interest in addressing the challenge. However, by treating each point separately by design, PointNet does not capture local structures, limiting its ability to recognize fine-grained patterns. As a result, in this specific study our aim is to use the fundamental structure of the original PointNet neural network but with some technical extensions to overcome this issue and try to obtain even better results.

Our point cloud processing neural network tackles these issues through the use of a specific neural network structure inspired by the success of transformers [3] in natural language processing and image analysis. The transformer family of models is particularly appropriate for point cloud processing because the self-attention operator, which is at the core of transformer blocks, is in essence a set operator: it is invariant to permutation and cardinality of the input elements. The application of self-attention to 3D point clouds is therefore quite natural, since point clouds are essentially sets embedded

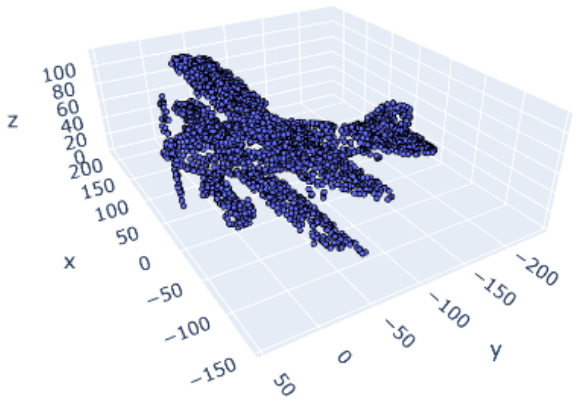


Fig. 2. Point cloud of an airplane from ModelNet10 dataset

in 3D space.

As illustrated in Fig. 1, the self-attention module generates refined attention feature for its input based on global context. Whereas PointNet[2] fails to consider local structures, the self-attention layer takes as input the sum of input embedding and positional encoding and the latter allows the operator to adapt to local structure in the data. Finally, all operations of Transformer are parallelizable and order-independent. Therefore, in theory, it should be able to replace the convolution operation while having a better versatility.

To analyze the results of our study, we have made our own implementation of the PointNet classification architecture, tested it on the common dataset ModelNet10[4] and obtained fairly similar accuracy results compared to the original paper on the dataset ModelNet40. Then, we added a Transformer Block with at its core the self-attention layers into our previous implementation of PointNet, tested it on the same dataset and compared the results. The results are conclusive because our technical extensions allowed us to obtain even better results than those from our our implementation of the initial structure of PointNet.

II. Related work

Many works have explored variations of transformer-based architectures for point clouds, such as Point Transformer [6], PCT [1] and Point-Bert[5]. These models have demonstrated further improvements in accuracy on a range of 3D computer vision tasks, suggesting that transformer-based architectures are a promising direction for further research in this area.

One example of incorporating self-attention layer into point cloud processing is the Point Transformer architecture [6]. It leverages the characteristics of the self-attention operator in the transformer networks which is invariant to permutation and cardinality of the input data. The constructed architecture is based purely on self-attention and pointwise operations, and is able to perform a variety of 3D learning tasks. What makes this architecture especially powerful is that it applies self-attention layer locally around each point and encodes positional information in the network, resulting in high scalability and high accuracy. For classification tasks, the network goes through five stages of downsampling to encode the features in a higher-resolution space, each consisting of a point transformer block followed by a transition down block to adjust the cardinality of the point sets to prepare for the next stage. Each transformer block is integrated with their point transformer layer (composed of a vector self-attention operator and position encoding aggregations0, linear projections, and a residual connection. The resulting network is able to set the new state of the art in ModelNet40 classification task with overall accuracy of 93.7% . Our implementation of self-attention layer onto PointNet architecture is inspired by the use of residual block locally applied self-attention layers.

Another noticeable work that utilizes the transformer for improving PointNet is PCT [1]. The goal of PCT is to convert the input points into a higher-dimensional feature space that captures the semantic relationships between points. The encoding process begins by embedding the input coordinates into this new feature space. Next, the embedded features are passed through four stacked attention modules that learn a highly descriptive and distinctive representation for each point. Finally, a linear layer is used to generate the output global feature. For classification, the global feature is fed to a classification decoder, which comprises two cascaded fully connected networks and then finalized by a linear classification layer to predict the class scores. Two implementations of Self-Attention layers are given by the paper. One is the naive Self-Attention layer which is very similar to the original Transformer [3]. Inspired by benefits of using Laplacian matrix in graph convolutional networks, a novel Offset-Attention layer is proposed, which generates features by element-wise subtraction of the input features from the original Self-Attention features. To further improve the algorithm, a neighbor aggregation strategy called neighbor embedding is proposed to capture the local information of the point clouds. The resulting network achieves 93.2% percent classification accuracy on the ModelNet40 classification task.

Most previous efforts to incorporate self-attention layers to PointNet based architectures involved some inductive bias, making them out of the line with standard Transformers. The Point-Bert [5] aims to use a pure Transformer architecture to process point clouds. However, a straight-forward integration of vanilla Transformer usually does not result in satisfying performance. The author claims that the unsatisfying performance is due to the lack of accurate annotations in the point clouds of training models. To solve thia data scarcity issue, the Point-Bert is pre-trained on a new task called Masked Point Modeling (MPM), which is similar to the masked language modeling (MLM) task used in BERT. In MPM, a subset of points in the input point cloud is randomly masked, and the model is trained to predict the masked points based on the remaining points in the cloud. The MPM task encourages the model to learn meaningful representations of the point cloud data, as it must infer missing information from the surrounding points. Once the model is pre-trained, it can be fine-tuned

for various downstream tasks, such as object detection or segmentation, by adding task-specific layers on top of the pre-trained transformer layers. Fine-tuning allows the model to adapt to specific tasks and achieve better performance. For classification task, Point-Bert achieves 87.43% accuracy.

III. Algorithmic Extension

A. Self-Attention Layer

Originally, attention functions have been introduced as an improvement from Residual Neural Networks (RNN) that struggles to deal with long-range dependencies in sequence to sequence transduction tasks. Indeed, in RNNs, the input sequence is bottlenecked through a fixed-size vector called the context vector which results in a significant loss of information for long-range input sequences and a lack of consideration to local features. So this new idea of attention function introduced in the famous paper "Attention is all you need" [3], uses a new context vector at each step of the output sequence to "look at" different parts of the input sequence and consider local features.

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. Usually both the keys and the values are computed from the input. We compute the alignment between the keys and the query through a compatibility function (usually a weighted dot product) and then we normalize those results to obtain attention weights. Finally, we obtain the output by computing a linear combination of the values and the attention weights. This concept is resumed on Fig. 3.

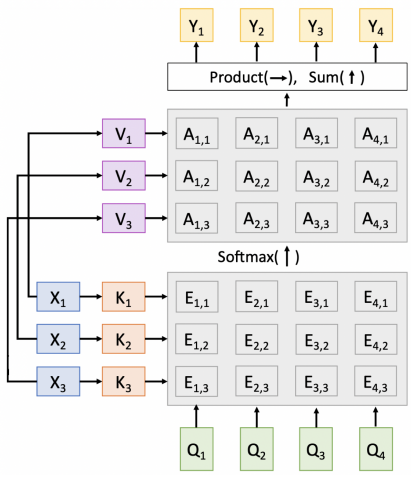


Fig. 3. The concept of attention functions.

In a self-attention layer all of the keys, values and queries come from the same place which is the output of the previous layer. This technique allows to connect each set of the output to all the sets from the previous layer while keeping the permutation invariance. An example of a self-attention module is described on Fig. 4.

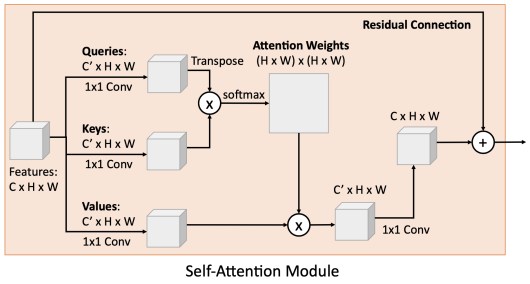


Fig. 4. Self-Attention layers of the Transformer Block.

Although, self-attention operators can be classified into two types: scalar attention and vector attention. The type that we have just presented is closer to the scalar attention operators because the set of input are feature vectors that we apply a standard scalar dot-product on. In vector attention, the computation of attention weights is a little bit different. In particular, attention weights are vectors that can modulate individual feature channels. In this case, the compatibility function is different. Instead of computing the dot-product of the keys and query we apply linear functions on them separately and then apply a symmetric relation function (such as subtraction). Finally, we apply a mapping function such as an MLP (multi-layer perceptron) on the results. However, in both case (scalar or vector) self-attention functions remain set operators.

B. Transformer Block

After trying to stack self-attention layers to the shared MLPs, we realized that we obtained vanishing gradients and thus formed our Transformer block through residual layers and adding normalization layers. The transformer block uses the self-attention layer as its core, as well as the shared MLPs from the original structure (we kept the original dimensions of the initial PointNet structure), normalization layers and a residual connection. Since the multi-layer perceptron increases the dimensions of the features from 64 to 1024 to put them in a higher-resolution latent space, we needed to interpolate the input of the MLP for the residual connection. The structure of our Transformer Block is described in Fig 5. The transformer block facilitates information exchange between these localized feature vectors, producing new feature vectors for all data points as its output considering both local and global features. Moreover, inspired by the PCT network structure [1] we have tried to stack 4 consecutive self-attention layers to try to learn representation for each point that is even richer semantically and more discriminative.

C. Network Architecture

The original PointNet architecture has a relatively simple structure as it does not rely on convolutions. Instead, PointNet directly processes each point in the point cloud independently and learns to extract features from each point using only shared multi-layer perceptron (MLP), a max pooling layer and some rigid transformations. The main idea is that by only

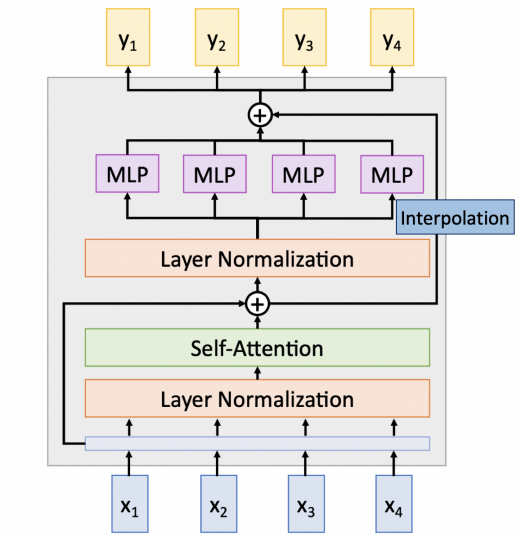


Fig. 5. The Transformer block.

using elementary binary symmetric operations and aggregation operations the network preserves the permutation invariance of the point cloud. As illustrated in Fig. 6, our implementation replaces the second MLP layer with our proposed transformer block which retains the input and output dimension for easier integration and since the self-attention layer is permutation equivariant it also maintains the proprieties of the initial architecture. Inside the transformer block, a single self-attention layer and a stack of self-attention layers are both experimented and the results are shown in the next section.

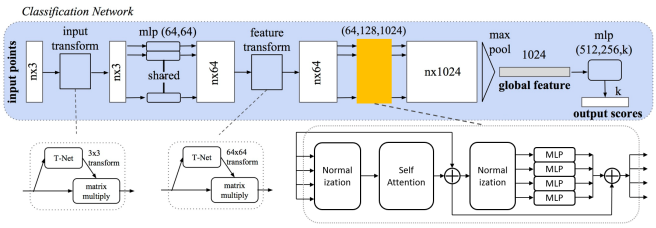


Fig. 6. PointNet with transformer architecture

IV. Experiments and Results

A. Procedures

While the original PointNet paper used ModelNet40 dataset to evaluate the effectiveness of its performances, our reproduced model is trained on ModelNet10 dataset containing 4,899 pre-aligned shapes from 10 categories for faster training and modification [4]. We implemented both the reimplementation PointNet model and the extended PointNet model in PyTorch using the Adam optimizer and a learning rate of 0.001. The dataset follows an 80:20 split for training and testing. Each shape is represented by 1024 uniformly sampled points as inputs to the model. Both versions of the neural network are trained for 15 epochs. The classification results are summarized in confusion matrices as shown in Fig. 7, which give an idea about how well the predictions are for each class. X-axis represents the predicted label and y-axis represents the true label. Each number in the grid represents the number of shapes in the training sets that are classified as the corresponding true label and predicted as the corresponding predicted label. As a result, high values along the diagonal of the matrix means good classification accuracy. The matrix can be normalized along each row to show the prediction accuracy for each class, which is what will be used to compare the performance of the two models.

B. Classification results of our PointNet implementation

The original PointNet architecture was reproduced to be evaluated on 3D shape classification task. The reimplementation follows the exact same structure as outlined in Fig. 6, including both input and feature transform composed of a T-net and a matrix multiply. Fig. 7 shows the normalized confusion matrix on the test dataset. Our implementation of PointNet achieved an overall 81.9% accuracy on the test dataset and was able to attain more than 90% accuracy at majority of the classes. Note that the model confuses desk with table and night-stand with dresser, implying the inability to capture local features to differentiate similarly shaped objects.

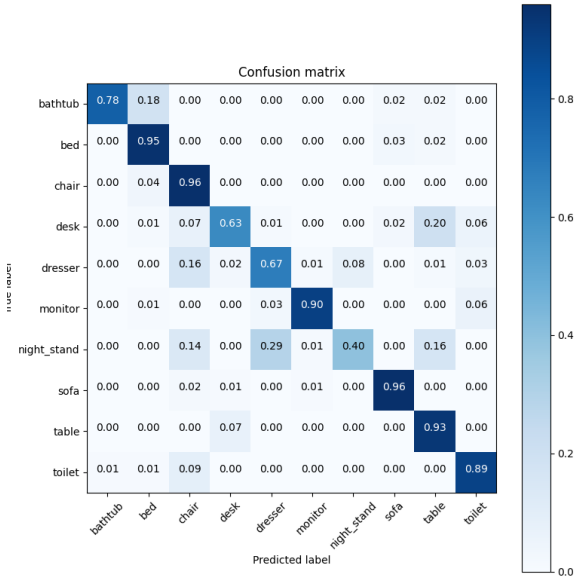


Fig. 7. Normalized Confusion matrix of our implementation of PointNet on the test Dataset.

C. Classification results of PointNet with Attention Layers

Our extension of the PointNet with transformer blocks is also tested on the same ModelNet10 dataset. The resulting confusion matrix is normalized and outlined in Fig. 8. We see an increase in overall accuracy from 81.9% to 84.3% by replacing the second MLP with a transformer block. More specifically, the extended model shows significant improvement at identifying dresser and monitor. However, a decrease

in accuracy is seen for recognizing desk and night-stand. Especially for night-stand, there is more false positives than true positives.

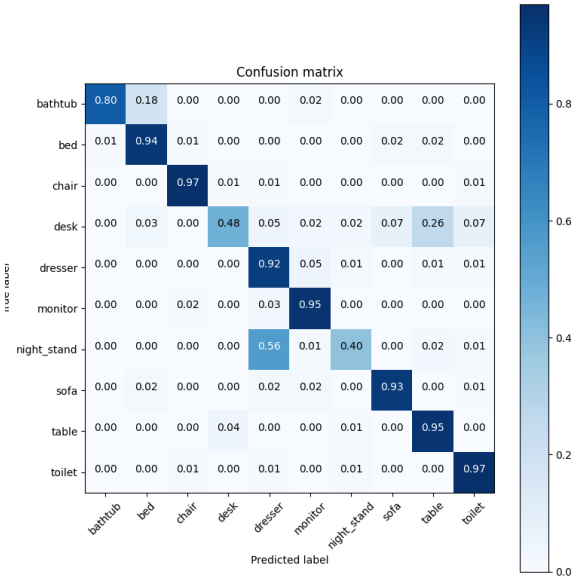


Fig. 8. Normalized Confusion matrix of our own version of PointNet with Transformer Blocks on the test Dataset.

D. Classification results of PointNet with Stacked Attention Layers

Four self-attention layers were stacked together in the transformer block and the resulting classification result is shown in Fig. 9. We can see a slight increase in overall accuracy at 84.8% on the test dataset. With stacked layers, the model is able to show increase in accuracy at detecting desks and nigh-stands which were not detected well with a single self-attention layer. The increase in performance is not as high as we expected and this can probably be a result of putting the transformer block too far into the network and using the interpolation operation.

V. CONCLUSION

This work examines the effect of integrating self-attention mechanism into point cloud processing. The original PointNet architecture is reproduced using pytorch and achieve an overall accuracy of 81.9% at ModelNet10 classification. A transformer block is proposed with variable number of self-attention layer implementation to replace the MLP layer in the PointNet architecture. The modified architecture achieved 84.3% overall accuracy with a single self-attention layer and 84.8% overall accuracy with a stack of four self-attention layers. A three percent increase in accuracy is easily achieved by incorporating self-attention layer and the model is able to differentiate very similar shapes by learning local features. Future work will systematically study the effect of stacking multiple self-attention layers and the effect of placement of transformer block to optimize the model performance.

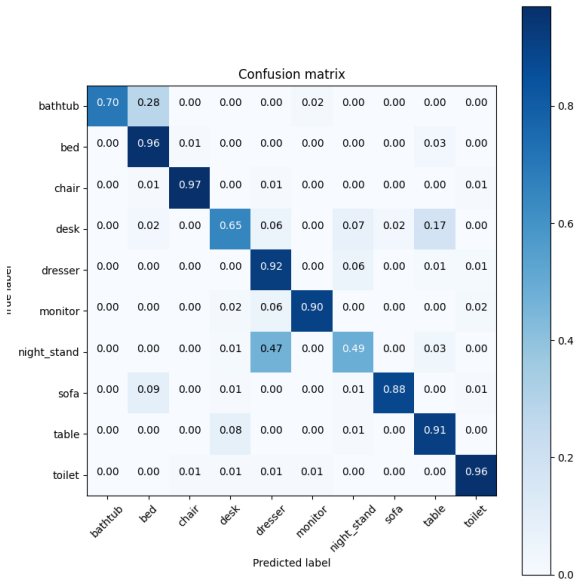


Fig. 9. Normalized Confusion matrix of PointNet with Transformer Blocks containing stacked self-attention layers on the test Dataset.

REFERENCES

[1] Meng-Hao Guo et al. "PCT: Point cloud transformer". In: *Computational Visual Media* 7 (2021), pp. 187–199.

[2] Charles R Qi et al. "PointNet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[3] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[4] Zhirong Wu et al. "3d shapenets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920.

[5] Xumin Yu et al. "Point-bert: Pre-training 3d point cloud transformers with masked point modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 19313–19322.

[6] Hengshuang Zhao et al. "Point Transformer". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 16259–16268.