

Advanced Feature Learning on Point Clouds using Multi-resolution Features and Learnable Pooling

Kevin Tirta Wijaya^{1*} Dong-Hee Paek^{2*} Seung-Hyun Kong^{2†}

¹Robotics Program ²CCS Graduate School of Mobility
KAIST

{kevin.tirta, donghee.paek, skong}@kaist.ac.kr

Abstract

Existing point cloud feature learning networks often incorporate sequences of sampling, neighborhood grouping, neighborhood-wise feature learning, and feature aggregation to learn high-semantic point features that represent the global context of a point cloud. Unfortunately, such a process may result in a substantial loss of granular information due to the sampling operation. Moreover, the widely-used max-pooling feature aggregation may exacerbate the loss since it completely neglects information from non-maximum point features. Due to the compounded loss of information concerning granularity and non-maximum point features, the resulting high-semantic point features from existing networks could be insufficient to represent the local context of a point cloud, which in turn may hinder the network in distinguishing fine shapes. To cope with this problem, we propose a novel point cloud feature learning network, PointStack, using multi-resolution feature learning and learnable pooling (LP). The multi-resolution feature learning is realized by aggregating point features of various resolutions in the multiple layers, so that the final point features contain both high-semantic and high-resolution information. On the other hand, the LP is used as a generalized pooling function that calculates the weighted sum of multi-resolution point features through the attention mechanism with learnable queries, in order to extract all possible information from all available point features. Consequently, PointStack is capable of extracting high-semantic point features with minimal loss of information concerning granularity and non-maximum point features. Therefore, the final aggregated point features can effectively represent both global and local contexts of a point cloud. In addition, both the global structure and the local shape details of a point cloud can be well comprehended by the network head, which enables PointStack to advance the state-of-the-art of feature learning on point clouds. Specifically, PointStack outperforms various existing feature learning networks for shape classification and part segmentation on the ScanObjectNN and ShapeNetPart datasets. The codes are available at <https://github.com/kaist-avelab/PointStack>.

1 Introduction

Point cloud has been one of the most popular representations of 3D objects in recent years (Qi et al., 2017a; Yang et al., 2020; Yu et al., 2021a). The capability of point cloud data in representing highly-complex 3D objects with low memory requirements enables real-time 3D vision applications for resource-limited agents. This is contrary to the voxel-based representations (Graham et al., 2018; Yan et al., 2018), where the memory requirement is cubically proportional to the spatial resolution. In addition, point cloud is the native data representation of most 3D sensors, thus performing 3D vision

*co-first authors

†corresponding author

directly on point clouds minimizes the pre-processing complexity. The advantages indicate that point cloud can be the prime data representation for fast and accurate neural networks for 3D vision.

Unfortunately, there are several challenges in applying the matured 2D deep learning-based feature learning techniques to the point cloud, for example, the irregular and unordered nature of the point cloud. These issues are addressed in the pioneering works of Qi et al. (2017a) and Qi et al. (2017b), which present the multilayer perceptron-based (MLP-based) PointNet and PointNet++, respectively. In the PointNet++ framework, a sequence of keypoints sampling, neighborhood grouping, neighborhood-wise feature learning, and feature aggregation is repeated several times to produce high-semantic point features. The relatively simple framework of PointNet++ is widely-used in literature. For example, PointMLP (Ma et al., 2022) enhances the framework by incorporating residual connections and construct a 40-layer MLP-based network that achieves state-of-the-art classification performance on several datasets.

Despite the promising results, the final high-semantic point features of the PointNet++ framework lose granular information due to the repeated keypoints sampling, where each of the surviving point features at the deeper layer of the network represents a larger spatial volume in the point cloud. Moreover, the max-pooling function that is used for feature aggregation may exacerbate the loss since it completely neglects information from non-maximum point features. Such a compounded loss of information concerning granularity and non-maximum point features could substantially harm the capability of the point features in delivering local context information such as the detailed shapes of objects in point clouds.

Considering the problem of losing granular and non-maximum point features information, we present two hypotheses. (1) It is advantageous for the task-specific head to have access to point features from all levels of resolutions. This enables the network to extract high semantic information while preserving the granularity to a certain degree. (2) A generalized pooling function that combines information from all point features could improve the representation capacity of the aggregated point features since the loss of information from non-maximum point features is minimized.

Based on the hypotheses, we propose a novel MLP-based network for feature learning on point clouds, PointStack, with multi-resolution feature learning and learnable pooling (LP). PointStack collects point features from various resolutions that are already available in the multiple layers of the PointNet++. The collected multi-resolution point features are then aggregated and fed to the task-specific head. Therefore, the task-specific head has access to both high-semantic and high-resolution point features. Moreover, PointStack utilizes the LP that is based on the multi-head attention (MHA) mechanism (Vaswani et al., 2017) with learnable queries for both single-resolution and multi-resolution feature aggregation. The LP is a permutation invariant and generalized pooling function that does not neglect information from non-maximum point features, but calculates the weighted sum of the multi-resolution point features according to their attention scores. Consequently, PointStack is capable of producing high-semantic point features with minimal loss of information concerning granularity and non-maximum point features, such that both global and local contexts of a point cloud can be effectively represented. As a result, the network head can well comprehend the global structure and distinguish fine shapes of a point cloud, enabling PointStack to advance the state-of-the-art of feature learning from point clouds.

Specifically, we observe that PointStack exhibits superior performance compared to various existing feature learning networks on two popular tasks: shape classification that requires global context understanding, and part segmentation that requires both global and local context understanding. On the shape classification task with ScanObjectNN dataset, PointStack outperforms existing feature learning networks by 1.5% and 1.9% in terms of the overall and class mean accuracy, respectively. On the part segmentation task with ShapeNetPart dataset, PointStack outperforms existing feature learning networks by 0.4% in terms of the instance mean intersection over union metric. The two results demonstrate the superiority of the proposed PointStack, not only for tasks that require global context, but also for tasks that require local context.

In a summary, our contributions are as follows,

- In the proposed PointStack, we employ a multi-resolution feature learning framework for point clouds. Leveraging point features of multiple resolutions provides both high-semantic and high-resolution point features to the task-specific heads. Therefore, the task-specific heads can obtain high-semantic information without substantially losing granularity.

- We propose a permutation invariant learnable pooling (LP) for point clouds as an advancement to the widely-used max pooling. LP is a generalized pooling compared to the max pooling, since it combines information from multi-resolution point features through the multi-head attention mechanism, as opposed to only preserving the highest-valued features.
- We demonstrate that PointStack outperforms various existing feature learning networks for point clouds on two popular tasks that includes shape classification on the ScanObjectNN dataset and part segmentation on the ShapeNetPart dataset.

The remaining of this paper is organized as follows. Section 2 discusses existing works that are related to feature learning. Section 3 describes the proposed PointStack in detail. Section 4 shows the experimental results with extensive discussions. Section 5 concludes this work with a summary.

2 Related Work

Feature Learning on Point Clouds. Most of the modern feature learning neural networks for point cloud data originate from the pioneering work PointNet by Qi et al. (2017a). In PointNet, a sequence of point-wise multilayer perceptron (MLP) blocks is applied to the raw point cloud to produce high-dimensional point features. The point features are then aggregated through the max pooling operation, which results in a fixed-length global feature vector. PointNet++ (Qi et al., 2017b) refines the PointNet by considering local structures of the point via sampling, grouping, and local group feature aggregation. First, a collection of keypoints is obtained through farthest-point sampling. Then, neighboring points around each keypoint are grouped, and a PointNet operation is applied to each group, resulting in a neighborhood-wise global feature vector for each keypoint.

Since then, numerous research has been conducted to enable learning the fine-grained local geometrical features of the point clouds. For example, Wang et al. (2019) propose a method to learn the relationships between the points with graph-based EdgeConv. Wu et al. (2019) introduce a convolution-based network that learns the appropriate convolution kernel via MLP networks and kernel density estimation. Hamdi et al. (2021) present a multi-view approach, where the network regresses the optimal view-point of the objects for 3D recognition. Recently, Ma et al. (2022) introduce PointMLP, a relatively deep MLP-based network for point cloud. The network is based on the original PointNet++ with additional residual connections and geometric affine modules. Owing to the residual connections, PointMLP manages to comprise deep layers, where the best-performing variant is composed of 40 layers.

Deep Learning with Multi-resolution Features. Multi-resolution features have been extensively explored in the image-based computer vision. Various traditional image processing techniques, such as the ones introduced by Dalal and Triggs (2005) and Lowe (2004), utilize a feature pyramid that leverages features of various resolutions (scales) from multiple layers for the downstream task prediction. The feature pyramid framework is still widely used in the deep-learning, especially after the introduction of the Feature Pyramid Network (FPN) by Lin et al. (2017). In FPN, feature maps of multiple resolutions are downsampled or upsampled to match the output feature map size, and concatenated together, resulting in an output feature map with both high-resolution and high-semantic information. In the point cloud domain, Hui et al. (2021) propose a transformer-based feature extractor that learns multi-scale feature maps for large-scale place recognition.

3 PointStack: Multi-resolution Feature Learning with Learnable Pooling

In this section, we first introduce an overview of the proposed multi-resolution feature learning implemented on a deep MLP-based network, PointStack. Following the overview, we introduce the learnable pooling that is permutation invariant.

3.1 Multi-resolution Feature Learning

The concept of multi-resolution feature learning is widely used for various downstream tasks in the computer vision (Lin et al., 2017; Ghiasi et al., 2019; Kirillov et al., 2019). The principle approach is to construct a feature pyramid from semantic features from all levels of resolutions. As a result, the feature pyramid has both high-semantic and high-resolution information that is often needed to recognize objects of various scales.

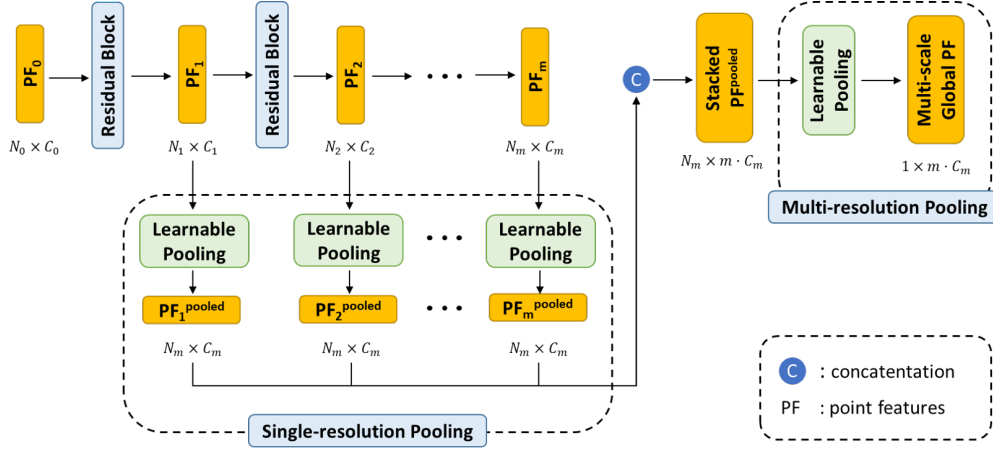


Figure 1: Feature learning backbone of PointStack. The residual block (one stage of PointMLP, Ma et al. (2022)) learns the underlying representation of the input point features and outputs point features of a reduced length. For m repeated residual blocks, the output point features of each block are pooled by the learnable pooling (LP) and concatenated to form stacked point feature. The final LP is then applied to obtain the multi-resolution features, which can be used by the network heads.

In the 3D point cloud domain, the potential benefits of utilizing multi-resolution features arise from the fact that 3D shapes are significantly more complex compared to the 2D images. Important textures or curves of the 3D shapes may be only observable at the highest level of granularity. As constructing high-semantic features comes at a cost of losing granularity in the existing approaches, the finer details of the 3D shapes may be obscured. Therefore, multi-resolution point features can be a solution for both collecting sufficient semantic information and preserving granularity to a certain degree.

Unlike PyramNet (Zhiheng and Ning, 2019) that uses multiple convolutions with different kernel sizes on the same point features to create a multi-scale feature map, we opt to leverage multiple point features from m different resolutions that are already available in the existing MLP-based networks, as shown in Figure 1. PointStack first learns the underlying representations of the points with the m repeated residual blocks, where the output of each block has lower resolution but higher semantic information compared to the corresponding input. We use the residual blocks instead of the transformer blocks as in Hui et al. (2021), since residual blocks are more efficient in terms of the memory requirements. This is because the self-attention mechanism in each of the transformer blocks has $\mathcal{O}(n^2)$ memory complexity with respect to the input size n .

After learning the appropriate representations, PointStack performs single-resolution pooling on each output point features, as shown within the bottom-left dotted box in Figure 1. That is, PointStack pools from each output point features (\mathbf{PF}_i of N_i feature vectors) at the i -th layer to produce $\mathbf{PF}_i^{\text{pooled}}$ of a fixed length N_m , where $\mathbf{PF}^{\text{pooled}}$ contains important features on the specific level of resolution.

Following the single-resolution pooling, PointStack concatenates all $\mathbf{PF}_i^{\text{pooled}}$ to form and process the stacked pooled point features, stacked- $\mathbf{PF}^{\text{pooled}}$, through the multi-resolution pooling (top-right dotted box in Figure 1), to produce a global feature vector. Since the global feature vector is obtained from the features of m resolutions, it contains information from both high-semantic and high-resolution features. Therefore, the task-specific heads have access to high-semantic information with minimal loss of granularity.

Note that the multi-resolution feature learning framework can be realized without fixing the length of the output features of the single-resolution pooling. However, we find empirically that the fixed-length single-resolution pooling substantially improves the classification performance. Such a phenomenon may originate from the fact that point features of m different resolutions have different numbers of entries. That is, the highest-resolution point feature, \mathbf{PF}_1 , has significantly more feature vectors compared to the lowest-resolution point feature, \mathbf{PF}_m . The disparity between the number of feature vectors may adversely affect the multi-resolution LP. Therefore, we incorporate the single-resolution

pooling process to produce the same number of feature vectors from m resolution levels. This explanation is supported by the experimental results shown with the ablation study in Section 4.

3.2 Learnable Pooling

Recent works for feature learning on point clouds often utilize pooling functions. The pooling function is an important trick to produce fixed-length global features from input points of an arbitrary size. Since a 3D shape can be represented by the same set of points of different order, the pooling function should be permutation invariant. A natural choice for such requirements is the max pooling function. Unfortunately, the max pooling function only preserves the highest-valued point features and completely neglects non-maximum point features, which results in a substantial amount of information loss.

To prevent this problem, we propose a generalized pooling function, learnable pooling (LP), that aggregates by calculating the weighted sum of all point features according to the correlation between the point features and learnable parameters. Since the LP does not neglect information from non-maximum point features, it can be used for aggregating both single-resolution and multi-resolution point features without loss of information.

Structurally, LP utilizes the multi-head attention (MHA) (Vaswani et al., 2017) that can be seen as an information retrieval process, where a set of *queries* is used to retrieve information from the *values* based on the correlation between the *queries* and the *keys*. We set both *keys* and *values* to originate from the same point feature tensor, while the *queries* are learnable parameters. In this setting, we can consider the network to learn the appropriate *queries* so that the retrieved point features (*values*) are highly relevant to the learning objectives. As the *queries* are directly supervised by the learning objectives, and the *values* are obtained through the weighted sum of all point features, the proposed LP is capable of producing representative aggregated point features with minimal loss of information compared to the max pooling function that completely neglects non-maximum point features.

The structure of the proposed LP is shown in Figure 2. The module architecture of the proposed LP is inspired by the Pooling by Multihead Attention (PMA) module introduced by Lee et al. (2019), but designed for a more compact form. That is, we only utilize linear transforms to match the channel size of the input point features to the desired output channel size and the multi-head attention mechanism. Note that, in this setting, the LP is a symmetric function so that the function is permutation invariant to the points of the point cloud.

Property 1. *The proposed learnable pooling is a symmetric function that is invariant to the permutation of the points of the point cloud. The proof can be found in Appendix A.1.*

The key to the permutation invariant property of the LP is the use of the point-wise shared-MLP and the fact that both *keys* and *values* originate from the same row-permuted feature matrix. Since both *keys* and *values* are row-permuted by the same permutation matrix, and the permutation matrices are orthogonal, the scaled dot-product attention mechanism becomes permutation invariant. In addition to the theoretical proof in Appendix A.1, we also show the empirical results in Section 4 to demonstrate the similarity between the standard deviations of the PointStack with LP and PointStack with max pooling outputs for various permutations of the input points.

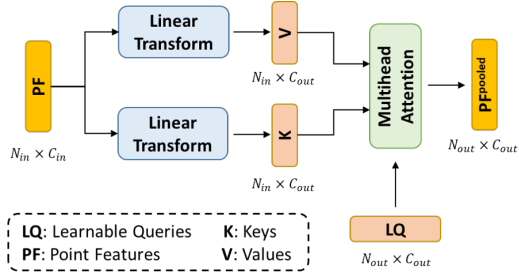


Figure 2: The structure of the learnable pooling (LP) module. Given an input of point features, LP transforms the features such that the channel size of the features match the channel size of the learnable queries (LQ). A multi-head attention (MHA) mechanism is then used to produce the fixed-length pooled point features. For the MHA, we set the input point features as the source of the *keys* and *values*, and LQ as the *queries*.

4 Experiment and Discussion

In this section, we describe the dataset, network details, and training setup used for the experiments. Then we show and discuss the experimental results.

4.1 Implementation Details

Dataset We evaluate the proposed PointStack on two tasks, 3D shape classification and point-wise part segmentation, with three different datasets: ModelNet40 (Wu et al., 2015), ScanObjectNN (Uy et al., 2019), and ShapeNetPart (Yi et al., 2016). We choose the two tasks, because they represent the two extremes of the downstream tasks widely studied for point cloud data. That is, classification requires learning the global context of the overall point cloud, while segmentation additionally requires learning the local context of each point. In the following experiments, the number of input points is set to 1024 for the classification task and 2048 for the segmentation task. Note that the hardest variant of ScanObjectNN (PB_T50_RS) is used in the experiments, where objects are perturbed with translation, rotation, and scale transformations.

Network For all experiments, we employ four residual blocks for the feature learning backbone of the PointStack, followed by an additional task-specific head. We set the hyperparameters for the residual blocks according to Ma et al. (2022). The task-specific head is made up of only MLP blocks, where each block consists of an affine transformation, batch normalization (Ioffe and Szegedy, 2015), ReLU non-linearity, and dropout (Srivastava et al., 2014) layers. Each head has a final affine transformation layer to match the shape of the output tensors with the task-specific requirements. For the learnable pooling, we set the size of learnable queries to 64×1024 and 1×4096 in the single-resolution pooling and multi-resolution pooling, respectively. Since there are four residual blocks, we use four separate learnable queries for the four level of resolutions in the single-resolution pooling.

Training Setup We train the networks using the PyTorch library (Paszke et al., 2019) on RTX 3090 GPUs. Networks are optimized using the SGD optimizer with a cosine annealing scheduler (Loshchilov and Hutter, 2016) without the warm restart. The initial learning rate and minimum learning rate are set to be 0.01 and 0.0001, respectively, and we incorporate label smoothing (Szegedy et al., 2016) to the cross-entropy loss. We perform data augmentation by applying random translation to all datasets, and random rotation to the ScanObjectNN dataset. For the shape classification task on ModelNet40 and ScanObjectNN, we set the maximum epoch to 300 and 200, respectively, and the batch size to 48. For the part segmentation task on ShapeNetPart, we set the batch size to 24, and the maximum epoch to 400.

4.2 Shape Classification

We evaluate the proposed PointStack on the shape classification task with ModelNet40 and ScanObjectNN datasets. ModelNet40 is a synthetic dataset of 40 different shape categories in the 12,311 point clouds sampled from computer-aided design (CAD) meshes. ScanObjectNN, on the other hand, acquires point clouds from real-world object scans, thus, the samples contain background points and occlusions. There are about 15,000 point clouds of 15 different shape categories.

Experimental results in Table 1 show that PointStack well outperforms the previous MLP-based network, PointMLP (Ma et al., 2022), on the real-world dataset (i.e., ScanObjectNN dataset) by 1.5% and 1.9% for the mean OA and mean mAcc, respectively. PointStack also outperforms other existing works such as the multiview projection-based MVTN (Hamdi et al., 2021) by 4.1%, and the transformer-based Point-TnT (Berg et al., 2022) by 3.4%. Notice that, PointStack reduces the gap between the OA and mAcc performance, proving that PointStack is less biased towards certain classes compared to existing works. The shape classification results prove that minimizing the loss of information concerning granularity and non-maximum point features through the multi-resolution feature learning and LP is beneficial for tasks that rely on the global context of the point cloud.

We notice that the overall accuracy performance of the PointStack on the synthetic dataset (i.e., ModelNet40) stands competitively at 93.3%, which is not superior to existing works. We speculate that the underlying cause of this issue is the significantly smaller number of training samples available in ModelNet40. To support this speculation, we train PointStack and PointMLP on a small subset of ScanObjectNN dataset, which we discuss in more detail in Subsection 4.6.

Table 1: Comparison of various models on ModelNet40, ScanObjectNN, and ShapeNetPart. We show the overall accuracy (OA), class mean accuracy (mAcc), and instance mIoU (Inst. mIoU). The notation $x \pm y$ represents the mean and standard deviation of the results after multiple runs of training.

Model	ModelNet40		ScanObjectNN		ShapeNetPart
	OA (%)	mAcc (%)	OA (%)	mAcc (%)	Inst. mIoU (%)
PointCNN (Li et al., 2018)	92.5	88.1	78.5	75.1	86.1
SpiderCNN (Xu et al., 2018)	92.4	-	-	-	85.3
DGCNN (Wang et al., 2019)	92.9	90.2	78.1	73.6	85.2
KPConv (Thomas et al., 2019)	92.9	-	-	-	86.4
DRNet (Qiu et al., 2021a)	93.1	-	80.3	78.0	86.4
GBNet (Qiu et al., 2021b)	93.8	91.0	80.5	77.8	85.9
Simpleview (Goyal et al., 2021)	93.9	91.8	80.5	-	-
MVTN (Hamdi et al., 2021)	93.8	92.2	82.8	-	-
CurveNet (Xiang et al., 2021)	93.8	-	-	-	86.8
PointBERT (Yu et al., 2021b)	93.8	-	83.1	-	85.6
PRA-Net (Cheng et al., 2021)	93.7	91.2	82.1	79.1	86.3
Point-MAE (Pang et al., 2022)	94.0	-	85.2	-	86.1
Point-TnT (Berg et al., 2022)	92.6	-	83.5	81.0	-
PointNet (Qi et al., 2017a)	89.2	86.0	68.2	63.4	83.7
PointNet++ (Qi et al., 2017b)	90.7	-	77.9	75.4	85.1
PointMLP (Ma et al., 2022)	94.1	91.5	85.4 \pm 0.3	83.9 \pm 0.5	86.1
PointStack	93.3	89.6	86.9 \pm 0.3 best = 87.2	85.8 \pm 0.3 best = 86.2	87.2

4.3 Part Segmentation

We evaluate the proposed PointStack on the part segmentation task with the ShapeNetPart dataset, a synthetic dataset derived from the ShapeNet dataset. **It contains 16,881 pre-aligned point cloud shapes that can be categorized into 16 shape classes and a total of 50 segmentation classes.**

From experimental results shown in Table 1, we observe that PointStack outperforms existing feature learning networks by at least 0.4%. Note that PointStack achieves such a high performance without using the voting strategy used by Xiang et al. (2021), where each input point cloud is randomly scaled multiple times, and the predicted logits are averaged to produce the final class prediction. It is worth to notice that PointStack achieves such performance with a simple MLP-based network, and the best performance of existing MLP-based network (Ma et al., 2022) is 1.1% lower than the PointStack. The part segmentation result, especially the significant improvement from the existing MLP-based network, testifies that minimizing the loss of information concerning granularity and non-maximum point features is crucial for tasks that require both global and local contexts. We visualize the part segmentation results in Figure 3 to demonstrate the high performance of the PointStack.

4.4 Ablation Study

We conduct an ablation study with the ScanObjectNN dataset to investigate the effect of three major components of PointStack on the classification performance. The three major components are multi-resolution features, LP-based single-resolution pooling, and LP-based multi-resolution pooling.

First, we investigate the effect of multi-resolution features. We apply the max pooling function to the point features of four different resolution levels, which results in four single-resolution global feature vectors. As in PointStack, we concatenate the four single-resolution global feature vectors and then apply another max pooling operation, which produces the multi-resolution global feature vector. From Table 2, we observe that incorporating multi-resolution features improve the OA and mAcc performances by 0.4% and 0.5%, respectively. This result proves that preserving granularity through multi-resolution feature learning is beneficial for the classification performance of the network.

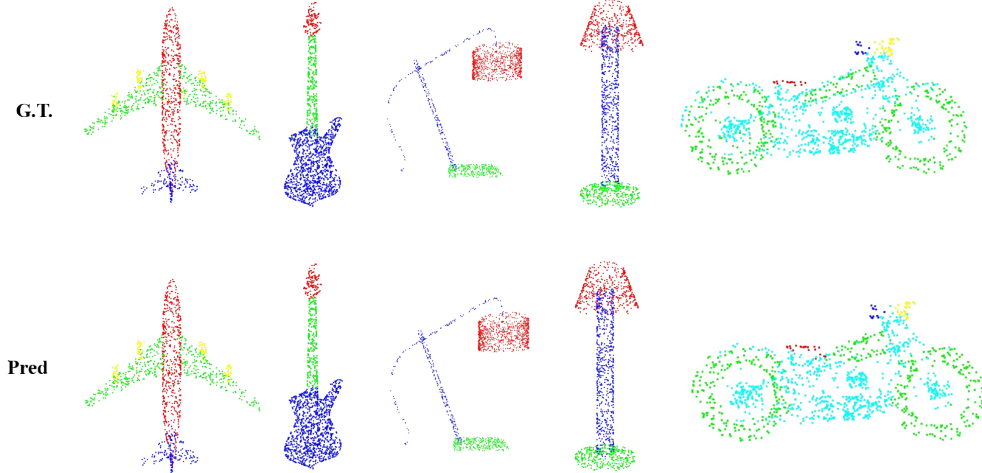


Figure 3: Visualization of the PointStack part segmentation ground truths (G.T.) and predictions (Pred). Qualitatively, the predictions are nearly identical to the ground truths.

Table 2: Ablation study on the PointStack major components on the ScanObjectNN dataset. The notation $x \pm y$ represents the mean and standard deviation of the results after several runs of training.

Multi-resolution Features	Single-resolution LPs	Multi-resolution LP	OA (%)	mAcc (%)
-	-	-	85.4 ± 0.3	83.9 ± 0.5
✓	-	-	85.8 ± 0.1	84.4 ± 0.1
✓	✓	-	86.5 ± 0.4	85.2 ± 0.2
✓	✓	✓	86.9 ± 0.3	85.8 ± 0.3
✓	-	✓	86.0 ± 0.7	84.9 ± 0.4

Second, we examine the effect of the Learnable Pooling (LP). We replace the max pooling function in the single-resolution pooling process with the LP. When the LP is used for pooling feature vectors from each level of the four resolutions, the OA and mAcc scores are further improved by 0.7% and 0.8%, respectively. Subsequently, when the LP is used for the multi-resolution pooling, PointStack gains additional 0.4% and 0.6% performance improvements for OA and mAcc, respectively. This result demonstrates that appropriately utilizing information from all point features in both single-resolution and multi-resolution poolings are crucial for producing relevant representations that benefit the classification performance of the network.

Additionally, we emphasize the importance of the single-resolution LP process. As mentioned in Section 3, the single-resolution LP enables PointStack to pool an equal number of feature vectors from each level of the m resolutions. In Table 2, the performance of PointStack without single-resolution LP becomes 0.9% lower for both OA and mAcc, in addition to the higher variance. This result indicates that standardizing the number of feature vectors from each level of the m resolutions is indeed crucial for the multi-resolution LP to achieve high performance.

4.5 Permutation Invariant Property of the Learnable Pooling

As mentioned in Section 3, the pooling function for any point cloud feature learning network should be permutation invariant. That is, the pooling function should be capable of producing the same output even if the order of the input points is changed.

To evaluate the permutation invariant property of the learnable pooling, we compare two variants of the PointStack: one with max pooling and another with the proposed learnable pooling. Specifically, we train the two variants and evaluate the standard deviation of the OA for ten random permutations of the input points.

Table 3: Comparison of the standard deviation of the OA (σ OA) for shape classification on the ScanObjectNN dataset. We use ten random permutations to calculate the σ OA values.

Pooling Function	σ OA (%)
Max Pooling	0.22
Learnable Pooling	0.26

Table 4: Performance comparison on the ScanObjectNN dataset. OA_F and OA_S are the classification performances when trained on the full and subset of ScanObjectNN dataset, respectively.

Model	OA_F (%) \rightarrow OA_S (%)
PointMLP	$85.4 \pm 0.3 \rightarrow 73.9 \pm 0.3$
PointStack	$86.9 \pm 0.3 \rightarrow 71.7 \pm 0.3$

From Table 3 we see that the network with learnable pooling has a similar standard deviation to the network with max pooling, where the standard deviation difference is only 0.04%. As the standard deviations are both small and similar, we confirm that the learnable pooling has the permutation invariant property similar to the max pooling.

4.6 Limitations on the Number of Training Samples

We observe that although PointStack achieves state-of-the-art performance on the ScanObjectNN dataset, it does not outperform existing works on the ModelNet40 dataset. From this observation, we speculate that a potential cause of lower performance of the PointStack can be the insufficient training samples available in the ModelNet40 dataset. The ModelNet40 dataset has 9,843 point clouds for training 40 different classes. In comparison, the main-PB_T50_RS variant of the ScanObjectNN dataset has over 11,000 point clouds for training just 15 classes.

To validate the necessity of a large number of training samples, we train PointStack and PointMLP (state-of-the-art network for ModelNet40) on a small subset of the ScanObjectNN dataset. The subset is constructed such that the number of training samples for each class matches the average number of training samples for each class in the ModelNet40. This translates roughly to 246 samples per class. During training, no augmentation method is applied.

As shown in Table 4, the overall accuracy of PointStack is lower than the existing MLP-based network performance, PointMLP, when the number of training samples is insufficient. And PointStack and PointMLP show 15.2% and 11.5%, respectively, lower performance than when they are trained with the full ScanObjectNN dataset. The results show the importance of sufficient training data size for PointStack to achieve the state-of-the-art performance. One possible explanation on such a requirement is that PointStack has a larger number of trainable parameters than the existing MLP-based networks due to the multiple learnable pooling. However, we emphasize that PointStack still achieves a competitive performance when trained with a limited number of training samples, and that modern datasets such as ScanObjectNN have sufficiently large training samples.

5 Conclusion

Recent point cloud feature learning networks often use aggregated point features originating from the deepest layer when performing downstream tasks. The aggregated point features may contain high-semantic information, but there is a cost of losing information concerning granularity and non-maximum point features due to the sampling operation and max-pooling, respectively. In this work, we have proposed a novel MLP-based feature learning network, PointStack, where the task-specific heads are given inputs of aggregated multi-resolution point features by a generalized pooling function, learnable pooling (LP). As a result, the aggregated point features could effectively represent both global and local contexts, and enable the network head to well comprehend the global structure and local shape details of objects in the point cloud. Empirically, we observe that PointStack outperforms various existing feature learning networks for the shape classification and part segmentation tasks. In the future, it is worthwhile to investigate the effectiveness of PointStack as the feature learning backbone network for other downstream tasks such as 3D object detection and shape completion.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C3008370).

References

- Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017a.
- Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020.
- Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12498–12507, 2021a.
- Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.
- Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.
- Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017b.
- Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.
- Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.
- Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1–11, 2021.
- Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, volume 1, pages 886–893. Ieee, 2005.
- David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
- Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
- Le Hui, Hang Yang, Mingmei Cheng, Jin Xie, and Jian Yang. Pyramid point cloud transformer for large-scale place recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6098–6107, 2021.
- Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019.
- Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

- Kang Zhiheng and Li Ning. Pyramnet: Point cloud pyramid attention network and graph embedding module for classification and segmentation. *arXiv preprint arXiv:1906.03299*, 2019.
- Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.
- 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.
- Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019.
- Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016.
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.
- Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
- Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018.
- Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.
- Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019.
- Shi Qiu, Saeed Anwar, and Nick Barnes. Dense-resolution network for point cloud classification and segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3813–3822, 2021a.
- Shi Qiu, Saeed Anwar, and Nick Barnes. Geometric back-projection network for point cloud classification. *IEEE Transactions on Multimedia*, 2021b.
- 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*, 2021.
- Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 915–924, 2021.
- Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. *arXiv preprint arXiv:2111.14819*, 2021b.
- Silin Cheng, Xiwu Chen, Xinwei He, Zhe Liu, and Xiang Bai. Pra-net: Point relation-aware network for 3d point cloud analysis. *IEEE Transactions on Image Processing*, 30:4436–4448, 2021.
- Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. *arXiv preprint arXiv:2203.06604*, 2022.
- Axel Berg, Magnus Oskarsson, and Mark O’Connor. Points to patches: Enabling the use of self-attention for 3d shape recognition. *arXiv preprint arXiv:2204.03957*, 2022.

A Appendix

A.1 Proof for Property 1

Let \mathbf{F} be the input point features matrix to the learnable pooling function Ψ . Furthermore, suppose that \mathbf{Q} , \mathbf{K} , \mathbf{V} , are the *query*, *key*, and *value* matrices respectively, for a scaled dot-product attention mechanism Φ .

The learnable pooling Ψ can be formally defined as

$$\Psi(\mathbf{Q}, \mathbf{F}) = \Phi(\mathbf{Q}\mathbf{W}_q, \mathbf{F}\mathbf{W}_k, \mathbf{F}\mathbf{W}_v), \quad (1)$$

where \mathbf{W}_q , \mathbf{W}_k and \mathbf{W}_v are the learnable weight matrices of linear transformations for the *query*, *key*, and *value*, respectively. Following the definition of scaled dot-product attention mechanism (Vaswani et al., 2017), equation (1) becomes

$$\Phi(\mathbf{Q}\mathbf{W}_q, \mathbf{F}\mathbf{W}_k, \mathbf{F}\mathbf{W}_v) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{W}_q (\mathbf{F}\mathbf{W}_k)^T}{\sqrt{d_k}} \right) \mathbf{F}\mathbf{W}_v, \quad (2)$$

where d_k is a scaling factor proportional to the feature dimension. Consider a case where \mathbf{F} is row-permuted by a permutation matrix \mathbf{P} . Then, the learnable pooling function becomes

$$\begin{aligned} \Psi(\mathbf{Q}, \mathbf{P}\mathbf{F}) &= \Phi(\mathbf{Q}\mathbf{W}_q, \mathbf{P}\mathbf{F}\mathbf{W}_k, \mathbf{P}\mathbf{F}\mathbf{W}_v) \\ &= \text{softmax} \left(\frac{\mathbf{Q}\mathbf{W}_q (\mathbf{P}\mathbf{F}\mathbf{W}_k)^T}{\sqrt{d_k}} \right) \mathbf{P}\mathbf{F}\mathbf{W}_v. \end{aligned} \quad (3)$$

Expanding the multiplications, and considering that permutation matrix does not scale the values such that performing the permutation before or after the softmax result in the same values, we obtain

$$\text{softmax} \left(\frac{\mathbf{Q}\mathbf{W}_q (\mathbf{P}\mathbf{F}\mathbf{W}_k)^T}{\sqrt{d_k}} \right) \mathbf{P}\mathbf{F}\mathbf{W}_v = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{W}_q \mathbf{W}_k^T \mathbf{F}^T}{\sqrt{d_k}} \right) \mathbf{P}^T \mathbf{P}\mathbf{F}\mathbf{W}_v. \quad (4)$$

Since permutation matrices are orthogonal, i.e., $\mathbf{P}\mathbf{P}^T = \mathbf{P}^T \mathbf{P} = \mathbf{I}$, where \mathbf{I} is an identity matrix, equation (4) becomes

$$\text{softmax} \left(\frac{\mathbf{Q}\mathbf{W}_q \mathbf{W}_k^T \mathbf{F}^T}{\sqrt{d_k}} \right) \mathbf{P}^T \mathbf{P}\mathbf{F}\mathbf{W}_v = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{W}_q (\mathbf{F}\mathbf{W}_k)^T}{\sqrt{d_k}} \right) \mathbf{F}\mathbf{W}_v. \quad (5)$$

Since the right hand side of equation (5) is equal to the right hand side of equation (2), we prove that $\Psi(\mathbf{Q}, \mathbf{F}) = \Psi(\mathbf{Q}, \mathbf{P}\mathbf{F})$ and Property 1 in subsection 3.2 holds.