Master's Thesis

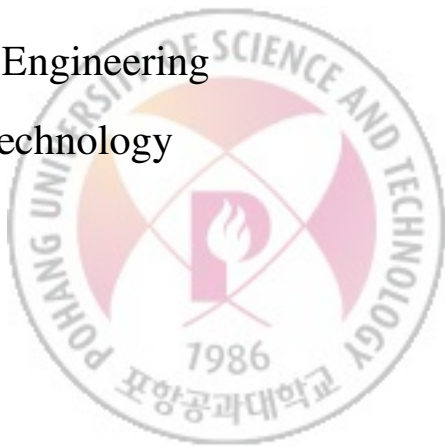
3D Shape Classification with Attention-based Pooling

Weonsuk Lee (이 원 석)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2019



어텐션 기반의 풀링을 이용한 3차원 모양 분류

3D Shape Classification with Attention-based Pooling

3D Shape Classification with Attention-based Pooling

by

Weonsuk Lee
Department of Computer Science and Engineering
Pohang University of Science and Technology

A thesis submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Science and Engineering.

Pohang, Korea
Dec 21, 2018
Approved by

Suha Kwak

Academic Advisor

3D Shape Classification with Attention-based Pooling

Weonsuk Lee

The undersigned have examined this dissertation and hereby certify that it is worthy of acceptance for a Master's degree from POSTECH.

Dec 21, 2018

Committee Chair Suha Kwak

Member Bohyung Han

Member Minsu Cho

MCSE 이 원 석, Weonsuk Lee,
20162526 3D Shape Classification with Attention-based Pooling,
어텐션 기반의 풀링을 이용한 3차원 모양 분류,
Department of Computer Science and Engineering, 2019, 35P,
Advisor: Suha Kwak. Text in English.

ABSTRACT

This work is a 3D shape classification algorithm that directly processes point cloud and outputs a class label. The proposed algorithm employs PointNet to attend to a few representative points of an input point cloud, and extracts local geometric information of attended points with EdgeConv operation. A few representative points can be learned in an end-to-end fashion since they are computed as attentions to an input point cloud. Also, considering local neighborhood demands substantial computation since point cloud is an unordered and irregular representation. Proposed algorithm avoids exhaustive computation of nearest neighbor search and feature extraction by reducing the number of points using attention-based pooling. Due to reduced number, each representative point is able to refer to a greater number of neighbors in original point cloud. Also, intermediate embedding layers can use higher dimensional representation as a result of pooling. Proposed algorithm achieved comparable performance to the state-of-the-art techniques in 3D shape classification benchmark, ModelNet40 dataset.

Contents

List of Figures

List of Tables

1 Introduction

Understanding 3D data is a fundamental task for different kinds of real world applications. It is studied as various 3D recognition tasks including shape classification, object detection and semantic segmentation. Despite the great success of convolutional neural networks (CNNs) in image recognition tasks, it is still challenging to effectively extend CNNs to 3D data representations. Applying 3D convolutional networks to the voxelized 3D data is one straightforward approach. However, a voxelization trade-off between the resolution and the memory consumption makes it impossible to feed high resolution 3D data to the CNNs. Also, the most of computations and memories are redundant since 3D data is inherently sparse. To utilize sparsity of 3D data, several methods [2, 3] used efficient indexing algorithms such as octree and kd-tree. Although these methods have shown encouraging results, they still requires large amount of computational resources to avoid discretization artifacts.

In contrast, 3D point cloud is sparse and concise 3D representation. Point cloud is sparse representation since it only carries the existing coordinates and their features. It is concise and rigid and affine transformations are easily defined and applied to point cloud. Additionally, it can be easily acquired from 3D data collectors such as RGB-D cameras, LiDAR, and Structure-from-Motion (SfM). Despite having several nice properties, point cloud is difficult to be combined with deep learning due to their spatial irregularity and

permutation invariance.

Recently, PointNet[4] proposed a simple network structure that directly processes point cloud. PointNet uses point-wise embedding and global max pooling, and they generate per-point feature and global feature respectively. However, it cannot capture local structures in metric space by its design. PointNet++[5] is introduced to address this problem by applying PointNet recursively on the point hierarchy generated by sampling, but the point sampling and grouping strategy is based on random sampling and does not reflect the semantics of input point cloud. DGCNN[6] builds hierarchical feature by using EdgeConv, which computes edge feature of k nearest neighbor graph and aggregating edge feature for each point. However, stacking multiple EdgeConv layers requires intensive computation since DGCNN maintains entire point cloud through out the feature extraction.

This work is a 3D shape classification algorithm which employs PointNet to attend to a few representative points of an input point cloud, and extracts local geometric information of attended points with EdgeConv operation. By only considering a few representative points rather than maintaining entire point cloud through out the feature extraction, our algorithm efficiently abstracts the input point cloud. The proposed method reduces the number of points by estimating a few different attentions to the entire point cloud. Point attentions are computed as normalized per-point scores and they are trained in an end-to-end fashion. Features of representative points are computed from the neighborhood gathered from the original point cloud. The main contributions of this thesis are summarized below:

- Proposed 3D shape classification attends to a few representative points of an input point cloud, which reduces exhaustive neighborhood computation and local feature computation. Attentions can be learned in an end-to-end fashion.
- The proposed network extracts local feature with a greater number of neighbors and employs more complex intermediate embedding layers as a result of pooling.

- The proposed algorithm presents comparable accuracy in the 3D shape classification benchmark [1].

The rest of this paper is organized as follows. First, Chapter 2 reviews various 3D representations and techniques in 3D recognition, and then the background and architecture of the proposed method is described in Chapter 3. Chapter 4 illustrates experimental results in ModelNet40 dataset.

2 Related Work

2.1 Feature Learning on Voxels and Images

Convolutional neural networks (CNNs) are naturally extended to 3D domain in two ways. One way is to quantize 3D shapes into voxels and feed them to 3D CNNs [1, 7]. However, naive implementation of 3D CNN demands either quantization artifacts from low resolution voxels or substantial computation and memory. Most of its computation and memory consumption is redundant since 3D data is inherently sparse. Indexing techniques such as octree and kdtree are used to efficiently represent and skip convolutions in empty spaces. O-CNN [2] builds a hybrid grid-octree data structure to enable convolutions on the octree. To allow efficient access to the underlying data, the maxiumal depth of an octree is restricted to a small number, and the shallow octrees are encoded using bit-strings. Kd-net [3] builds a balanced kd-tree recursively in a top-down fashion. After that, each node's feature is computed in a bottom-up fashion, where parent node's feature is computed as a result of applying non-linearity and affine transformation on its two child node's feature.

Another way is to render and project the 3D data into 2D images in multiple views and apply standard 2D CNNs. Multi-vew convolutional neural network (MVCNN) [8] generates multi-vew images with different camera settings and aggregates views using

view-pooling layer. View clustering and pooling layer based on dominant sets [9] is used to relax the winner-take-all view pooling behaviour and consider multiple views effectively. Both volumetric and multi-view CNNs are utilized [10] to complement the other's representation. RotationNet [11] treats the view point labels as latent variables, which are learned in an unsupervised manner. Although the multi-view approaches achieve superior performance to the other methods, they cannot be easilty extended to per-point labeling tasks such as semantic segmentation.

2.2 Feature Learning on Point Cloud and Graph

PointNet [4] is the pioneer of deep network that directly processes the point cloud as an input. PointNet generates the global feature invariant to the point order because it is composed of shared point-wise multi-layer perceptrons (MLPs) and a symmetric function, which is a channel-wise max pooling across points. Around the same time, a similar architecture [12] is proposed to process general set data. Despite the success of PointNet, it is lack of hierarchical feature extraction like CNNs and the local geometry of points in ambient space is ignored by design.

Several methods have been proposed to address the local feature extraction based on the graph. Point cloud is considered as a graph (usually k-nearest neighbor graph) and graph convolution is conducted on each point in order to encode its local neighbors [6, 13–17]. DGCNN [6] introduced EdgeConv operation which computes edge features that describe the relationships between a point and its k-nearest neighbors. Similarly, message passing neural networks [13] and dynamic edge-conditioned filters are proposed to exploit the edge and its connected vertices' feature. However, each point feature cannot be sufficiently large because the number of vertices in graph based methods is invariant during the process. Spectral CNNS [15–17] transforms point signals to graph Fourier domain by computing eigenvectors of the graph Laplacian to apply convolution in non-Euclidean space, but these methods are mostly limited to manifolds.

Other methods directly deals with absolute and relative point coordinates in ambient Euclidean space without explicit pair-wise relations(edges) [5, 18–21]. PointNet++ [5] builds a hierarchy of point set by sampling and grouping strategy, and applies PointNet recursively on each group. Some methods provide natural extension or replacement of convolution kernels. Point-set kernel [19] is a set of learnable points in 3D space, and for each point neighborhood, the kernel correlation between them and the point-set kernel is computed. The kernel correlation corresponds to the geometric affinities. Pointwise convolution [20] computes the grid convolution for each point where the input for pre-defined grid is averaged feature value inside the kernel support. Flex convolution [21] extends discrete convolution operation to continuous space and the continuous kernel is approximated linearly. Although above methods extend convolution operation in various way, pooling scheme is rather not defined or relies on random sampling.

2.3 Pooling on Irregular domains

In regular domain such as an image and grid space, a proper and scalable sub-sampling is intuitively defined. However, it is not canonically defined in unstructured data such as a point cloud and graph. Uniform random sampling is the most straightforward method but it is not deterministic and reliable. Thus, a few predictable sub-sampling methods are used to reduce the spatial size. VoxelGrid algorithm is adopted by [14], but the algorithm inherits the drawbacks of voxel-based methods. Farthest point sampling is used by [5], but the algorithm has the complexity of $\mathcal{O}(n^2)$. Inverse density importance sub-sampling is adopted by [21] where density is efficiently estimated as the sum of the distances of pre-computed k-nearest neighbors. SONet [22] utilized self-organizing map which reflects the spatial distribution of point cloud. In graph domain, Graculus algorithm [16] is adopted to produce multilevel clustering. Although above algorithms reduce the size of input systematically, their pooling scheme is determined in an unsupervised manner which may leave room for improvement. Recently, [23] pro-

poses a differentiable graph pooling module that can learn a soft cluster assignment for each node.

3

3D Shape Classification with Attention-based Pooling

In this section, we describe our attention-based pooling for 3D shape classification and their background works.

3.1 Overview

Our proposed method consists of an attention module and a feature extraction network. Attention module is built upon PointNet [4] with the differentiable pooling [23]. The module takes a point cloud as an input and outputs a few attentions which are used to sample representative points of the input. Feature extraction network is built based on DGCNN [6], but it only consumes a few representative points rather than the entire point cloud. In section 3.2, three baseline methods will be described, and the proposed method will be described in section 3.3.

3.2 Background

In this section, we present three baseline works of our method; PointNet, DGCNN and DiffPool. Firstly, we introduce PointNet which is a pioneer in deep learnings on point

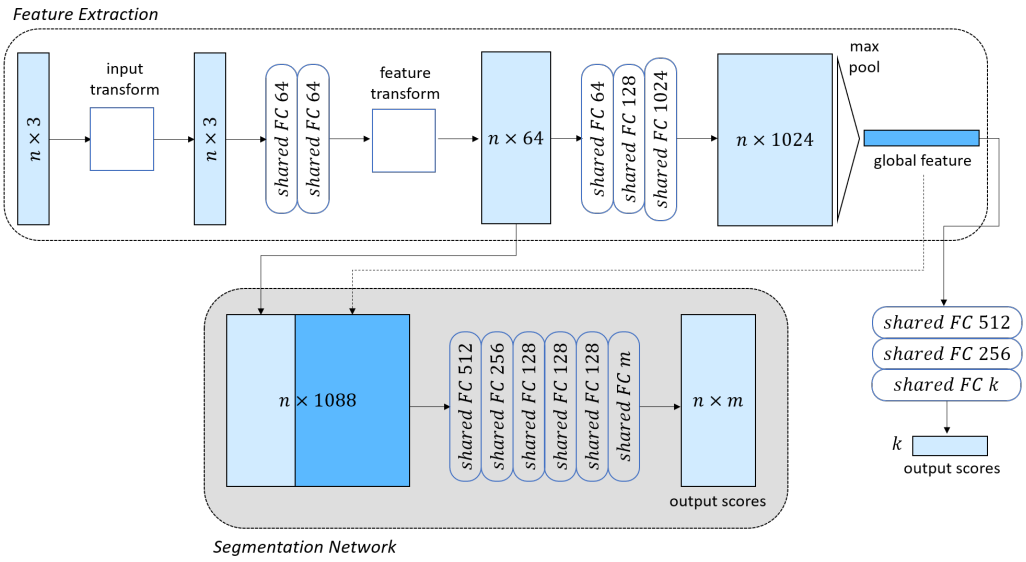


Figure 3.1: **PointNet Architecture**. The feature extraction part takes n points as input, embeds per-point feature with shared fully connected (FC) layers. Input and feature transformations are applied in the middle of embedding and point features are aggregated by max pooling. Global feature is fed to small classifier whose output is classification scores for k classes. Segmentation network is an extension which concatenates global and local features and outputs per-point segmentation scores. BatchNorm is used with ReLU activation, dropout is applied to last layer in classifier.

sets. It consists of two simple components, yet it has shown to be effective. However, PointNet does not capture ambient local structure or neighbor relationship due to its limited design. To overcome this limitation, DGCNN utilizes K-nearest neighbors of each point to encode local structure. Meanwhile, in the field of graph neural networks, the differentiable pooling (DiffPool) is proposed to learn hierarchical representations of graphs. We introduce an architecture and properties of each work, and its limitations.

3.2.1 PointNet

PointNet is a unified framework that directly processes point clouds, learns both global and local point features, and provides a solution to 3D recognition tasks such as

classification, semantic segmentation (Fig. 3.1). An input of the network is a point set where each point is represented as a three coordinates (x, y, z) along with other information such as surface normals or colors if provided. The network produces a permutation invariant global feature as an intermediate result, and outputs classification scores. Segmentation network, which is an extension of feature extraction network, produces permutation equivariant per-point feature. Two key components of PointNet is per-point feature computation and channel-wise max pooling applied across input points.

PointNet consists of common feature extraction network which produces global feature and task-specific part which outputs final scores (Fig. 3.1). At the feature extraction part, each point is processed identically and independently by shared fully connected(FC) layers. Then, a symmetric function (e.g. max pooling) takes the embedded results to aggregate information over all points to a single vector. Additionally, input and feature transformations are applied to the intermediate results to encourage the network to be invariant under certain transformations.

For shape classification, the global feature is fed to a simple classifier and it outputs class scores. Dropout is applied to last layer in classifier. For semantic segmentation, it requires a combination of local and global information. This is achieved by concatenating the intermediate local point feature and the global feature. Then the semantic labels are predicted by applying shared FC in a per-point manner.

Due to its simplicity, a few interesting properties of PointNet is provided theoretically. First, the universal approximation ability of the network to continuous set functions are shown. Also, the network learns to encode shape by a sparse set of key points, thus the network value is unlikely to be changed up to small input corruption.

Although PointNet provides a simple, efficient neural module, the absence of hierarchical feature learning makes the model sensitive to global transformation including translation. Also, the lack of granularity during feature aggregation prevents the model from learning details of shape.

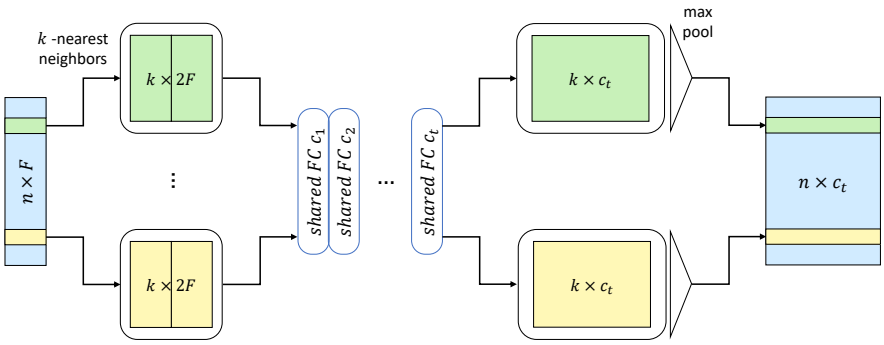


Figure 3.2: **EdgeConv operation**. EdgeConv block takes n features as an input and outputs another n features. For each input feature p_i, the module first computes the difference $p_j - p_i$ between k-nearest neighbors and concatenates them with the original feature. Then the edge features are computed by shared fully connected layers. The edge features are aggregated to associated vertex.

3.2.2 Dynamic Graph CNN for Learning on Point Clouds (DGCNN)

To address the drawbacks of PointNet, a simple operation called EdgeConv is proposed to capture the geometric relationships among points [6]. EdgeConv is differentiable module that can be easily plugged into existing architectures. It is a kind of graph convolution [24] because the algorithm works on general graph. EdgeConv has shown the effectiveness in encoding local feature and that it can learn hierarchical features like CNNs.

Entire structure of DGCNN is similar to PointNet, but EdgeConv is inserted between shared FC layers. EdgeConv consists of three parts, the graph construction, individual and independent embedding of edges, and channel-wise max pooling across point neighborhood. First, it constructs the graph that well represents the geometric relations of the

input point cloud. Here the simple k-nearest neighbor graph is used where the point p_i's *k* closest points $x_{j_1}, x_{j_2}, ..., x_{j_k}$ are pointing to p_i. Then the edge feature is computed by subtraction of two end nodes from source to target, resulting the relative feature. Then, the original point feature p_i is concatenated to them, resulting the edge features of dimension $n \times k \times 2F$, where n is the number of vertex and F is the dimension of input feature. Each input edge feature is given by

$$e_{ij} = concat(x_i, x_j - x_i)$$

Each edge feature is processed by the shared FC layers and aggregated by channel-wise max pooling across neighborhood. The output of EdgeConv at the *i*-th vertex is given by

$$x'_i = \max_{j:(i,j)\in E} h_\theta(e_{ij})$$

where E is edges, h_θ is the function approximated by FC layers.

EdgeConv can be interpreted as applying shared mini-PointNet to each point's neighborhood, after aligning the center to the origin. Although the receptive field is small at first, DGCNN recursively applies EdgeConv to obtain hierarchical feature. Also, the k-nearest neighbor is dynamically recomputed in the feature space, which helps the model to learn how to group points in a point cloud.

Although EdgeConv module learns how to extract local geometric features, the module does not have pooling scheme which makes it inefficient. Since the computation and memory increase linearly in terms of *k*, it is difficult increase *k* to get large receptive field. Also, increasing the channel size of individual point feature costs substantial computation resources.

3.2.3 Hierarchical Graph Representation Learning with Differentiable Pooling

Meanwhile, the differentiable pooling (DiffPool) of graph is proposed to assign each graph vertex to soft clusters [23]. Combined with existing graph neural network architectures, it can generate hierarchical representations of graphs.

They present a graph G as (A, F), where $A \in \{0,1\}^{n \times n}$ is the adjacency matrix of graph, and $F \in \mathbb{R}^{n \times d}$ is the vertex features. Existing graph neural networks take a graph and update vertex features as

$$X^{(k)} = M(A, X^{(k-1)}; \theta^{(k)})$$

where $X^{(k)}$ are the node embeddings, and M is a message passing function that propagates the previous node features according to the adjacency matrix A and trainable parameters $\theta^{(k)}$. The input node embedding $X^{(0)}$ is initialized as F.

Differentiable pooling is achieved by predicting a soft cluster assignment matrix S using graph neural networks. Let n_l be the number of vertex at layer l and GNN is used to reduce the number of node to n_{l+1}. Soft cluster assignment matrix $S^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$ is predicted where each row of the matrix $S^{(l)}$ represents the assignment score of one of n_l nodes at layer l:

$$S^{(l)} = softmax(GNN_l(A^{(l)}, X^{(l)}))$$

where the softmax function is applied row-wise and $X^{(l)}$ is the input vertex feature.

A graph $G^{(l)} = (A^{(l)}, X^{(l)})$ is updated according to computed cluster assignment $S^{(l)}$. New coarsened adjacency matrix $A^{(l+1)}$ and new vertex feature $X^{(l+1)}$ is generated as:

$$X^{(l+1)} = {S^{(l)}}^T Z^{(l)} \in \mathbb{R}^{n_{l+1} \times d}$$

$$A^{(l+1)} = {S^{(l)}}^T A^{(l)} S^{(l)} \in \mathbb{R}^{n_{l+1} \times n_{l+1}}$$

This new pooling method has several advantages over previous methods. It can be inserted to existing graph neural network architecture and the entire system can be trained in an end-to-end fashion using stochastic gradient descent. Unlike other pooling methods like [16], which obtain a pooling scheme in unsupervised manner, it learns to assign cluster according to the global feature and node features. Also, hierarchical cluster structure is easily obtained by recursively using pooling layer.

3.3 3D Shape Classification with Attention-based Pooling

We propose a classification network that can efficiently extract local features and aggregate them. Our proposed network takes n points as input and calculates n_1 attentions, then computes local geometric feature for each n_1 attended points. Computed local features are embedded and aggregated by channel-wise max pooling to produce the global feature. Proposed architecture can be divided into two parts: attention-based pooling and local feature extraction (Fig 3.3).

3.3.1 Attention-based Pooling

Exploiting local geometry has been shown to be effective [5, 6, 19, 21]. For handcrafted descriptors [25, 26], local information is considered in forms of surface normal and covariance matrix. Also, local feature learned by translation invariant convolution kernel is critical to the success of image-based deep learning methods.

Unfortunately, extracting local information in 3D (or higher dimensional) point cloud is computationally expensive. Local neighborhood index should be computed first to obtain local information, unlike grid space where neighbor feature can be easily accessed by simple indexing. Also, computing neighborhood of every point may be redundant because adjacent points would have almost the same feature unless the input point cloud is extremely sparse.

To address this inefficiency, we propose to first attend to a few representative points.

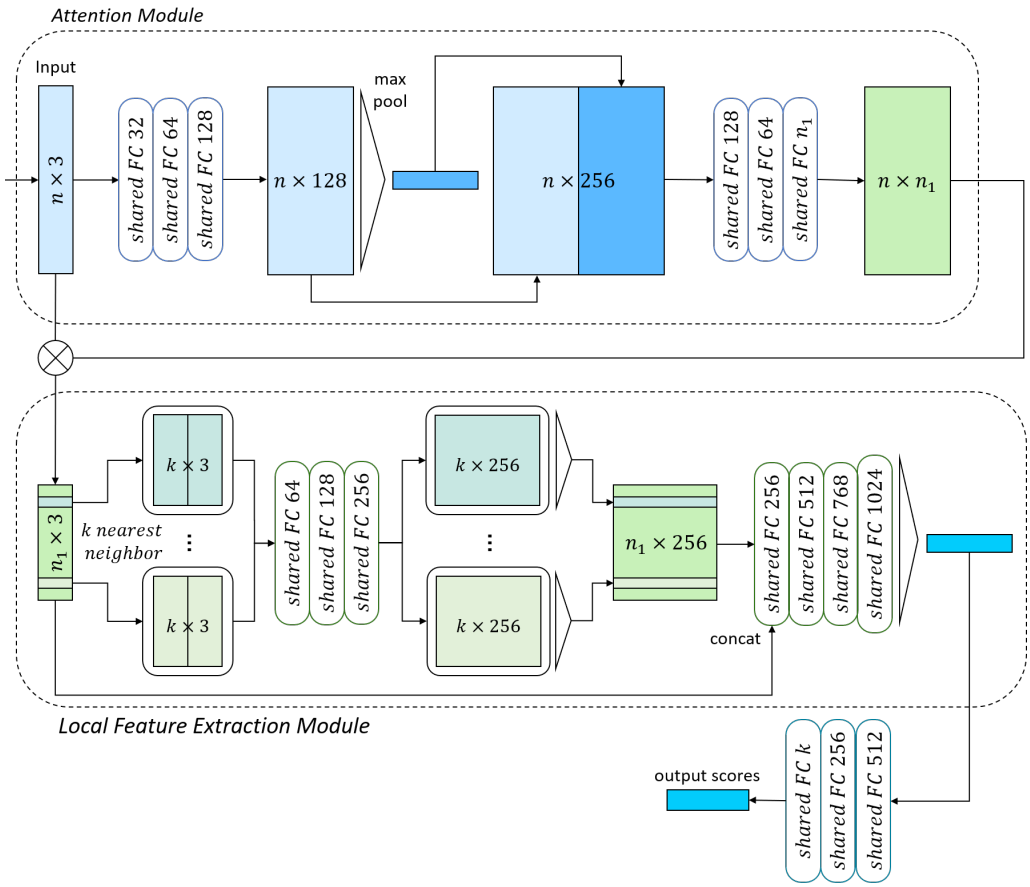


Figure 3.3: **Proposed Architecture**. The proposed method consists of two parts: attention-based pooling and local feature extraction. n_1 points are attended by the differentiable pooling and local neighborhood of each attended point is obtained from the input point cloud. Local feature extraction, which is based on modified EdgeConv operation, produces local features. Finally, they are embedded and aggregated by max pooling.

Given a input point cloud of n points, n_1 critical points are estimated. The network outputs $S \in \mathbb{R}^{n \times n_1}$ matrix where each column is attention map of n points. Attended points are computed as

$$X' = S^T X \in \mathbb{R}^{n_1 \times 3}$$

where $X \in \mathbb{R}^{n \times 3}$ is the input point cloud.

Attention module is analogous to the PointNet segmentation network (Fig. 3.3). Each point is embedded by shared FC layers and max pooling is applied to the embedded feature across points to produce the global feature. Per-point feature and the global feature are concatenated and shared FC layers are applied to generate n_1 scores for each point. Attention map S is given by

$$S = softmax(PointNetSeg(X))$$

which is similar to the DiffPool [23], but the softmax function is applied in a column-wise fashion in our case.

Noticeable difference between the proposed attention-based pooling and DiffPool is the softmax function direction. In case of DiffPool, the normalization is applied row-wise to assign points to pre-defined clusters. Then, cluster feature is computed as the sum of input features assigned to the cluster. However, aggregation by summation cannot generate the representative of the cluster in 3D space. Cluster representative in 3D should be computed by averaging which leads to column-wise normalization.

Proposed attention module contains the global feature extraction to determine representative points considering both local and global information. If the global feature is not included, the attention will be learned to only indicate points in a few positions in context-free manner. By combining the global feature, the attention identifies a few representative points considering the spatial distribution of the input point cloud.

3.3.2 Local Feature Extraction

After the pooling, *k* nearest neighborhood of n_1 attended points is gathered in the original point cloud. Then, the neighbors are normalized by subtraction to the center point. The *i*-th attended point's normalized neighborhood is given by

$$\mathcal{N}_i = \{x_j - x'_i | x_j \in kNN(x'_i, X)\}$$

where $kNN(x'_i, X)$ is the neighborhood set computed in the original point cloud X. Normalized points are processed individually by shared FC layers and aggregated along the neighborhood to produce local feature of attended points:

$$l_i^{(1)} = \max_{x_j \in kNN(x'_i, X)} h_\theta(x_j - x'_i)$$

Local feature computation is similar to EdgeConv, but there are two key difference between EdgeConv and our model. Firstly, original point coordinate x'_i is not concatenated to the relative coordinates. By this, our local feature becomes translation invariant. Although our entire model is not translation invariant, we believe that this translation invariance improves the generalization. Secondly, our model is able to use greater *k* as a result of pooling. The number of attention n_1 and k can be adjusted to determine the overlap of receptive field size. We used $n_1 = 128$ and $k = 64$ for the input size of 2048×3.

After the local features are obtained, we concatenate them with the attended point coordinates. Then, shared FC layers and max pooling is applied to obtain the global feature. Finally, the global feature is fed to classifier and it outputs the classification score.

Proposed classification network is trained with stochastic gradient descent in end-to-end fashion. Our model can be trained with cross-entropy loss, but this may drive the attentions to be focused to the same discriminative area which may cause poor general-

ization. Hence, we adopted an additional loss that separates attended points from each other. Additional loss is a combination of two different terms. One is hinge loss of the pairwise distance of attended points, which induces attended points to repel each other. The other is Chamfer distance of the original point cloud and attended points, which encourages the attended points to resemble the original point cloud. Our total loss function is given by

$$\begin{aligned}\mathcal{L} = \mathcal{L}_c + \alpha(\frac{1}{n_1}\sum_{y_i,y_j\in Y}\max(0, m - \|y_j - y_i\|^2)) \\ + \beta(\frac{1}{n_1}\sum_{y\in Y}\min_{x\in X}\|y-x\|^2 + \frac{1}{n}\sum_{x\in X}\min_{y\in Y}\|y-x\|^2)\end{aligned} \tag{3.1}$$

where $\mathcal{L}_c$ is cross entropy loss, X is a set of original points and Y is a set of attended points, m is the margin of hinge loss and α and β is regularization parameters.

4 Experiment

4.1 Dataset

We evaluated our algorithm on ModelNet40 [1] shape classification benchmark, which consists of CAD models of 40 man-made categories (Fig. 4.1). ModelNet40 contains 12,311 objects, and we used the standard split with 9,843 and 2,468 shapes for training and testing, respectively. Point clouds are sampled uniformly from vertices and faces of CAD models. We used the sampled ModelNet40 dataset from [4], where each model is represented by 2,048 points.

Input point clouds are shifted and resized to fit in zero-mean inside a unit sphere. A number of data augmentations are applied at training phase following [5]. Each point in the point clouds is slightly rotated with an angle sampled from Gaussian noise $\mathcal{N}(0, 0.06)$. Point clouds are scaled by a factor sampled from an uniform distribution $\mathcal{U}(0.8, 1.25)$. Point clouds are shifted by a value sampled from an uniform distribution $\mathcal{U}(-0.1, 0.1)$. Finally, each point in the point clouds is jittered with a value sampled from Gaussina noise $\mathcal{N}(0, 0.01)$. Unlike [5], random rotation is not applied since it does not improve the results.

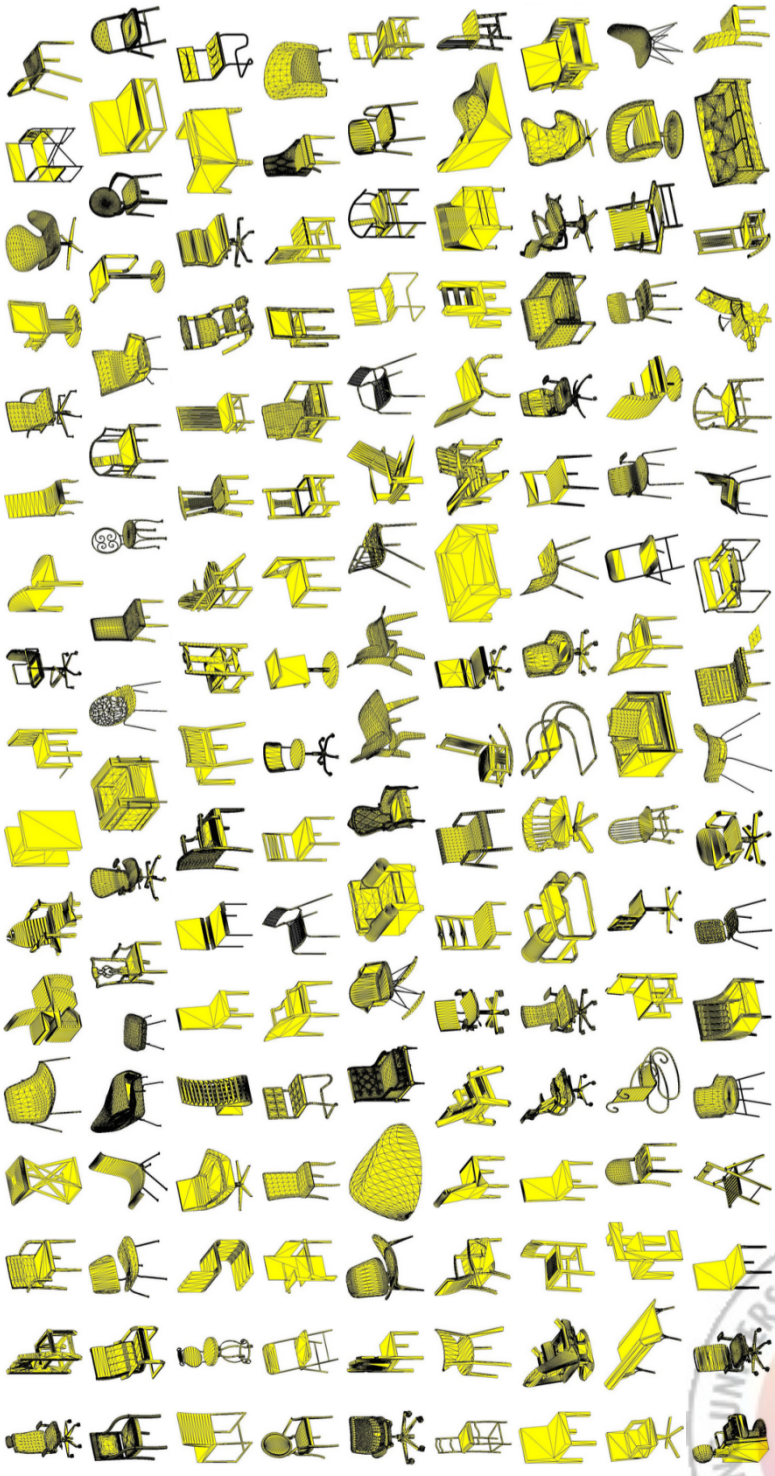


Figure 4.1: **ModelNet Dataset**. ModelNet40 contains 3D CAD models of 40 man-made categories. Examples of 3D chair model is visualized [1]. Point cloud is sampled from the mesh surface of CAD model.

4.2 Evaluation Metric

4.2.1 Instance-wise Classification Accuracy

Instance-wise classification accuracy is most commonly measure in classification tasks. The number of correct objects, whose top-1 prediction and label is the same, divided by the number of total objects:

$$InstAcc = \frac{\#\ of\ correct\ objects}{\#\ of\ total\ objects}$$

4.2.2 Class-wise Classification Accuracy

Class-wise accuracy is an average of accuracies of instance-wise accuracy of classes:

$$ClassAcc = \frac{1}{c}\sum_{i=0}^{c-1}\frac{\#\ of\ correct\ objects\ in\ class\ i}{\#\ of\ total\ objects\ in\ class\ i}$$

where c is the number of classes. Class-wise accuracy is usually lower than instance-wise accuracy because it requires high accuracy in low data classes.

4.3 Result of 3D Shape Classification on ModelNet40

Proposed method is evaluated on ModelNet40 dataset and compared with other works (Table 4.2). Two kinds of accuracy and input shape are shown because the number of input points and additional feature usage (e.g. surface normal) are different. SO-Net [22] achieved the highest performance with the use of rich point cloud with surface normal. However, our model outperforms SO-Net when the input point cloud is smaller, without surface normal. Proposed method achieved comparable performance to DGCNN without exaustive computation of neighborhood features.

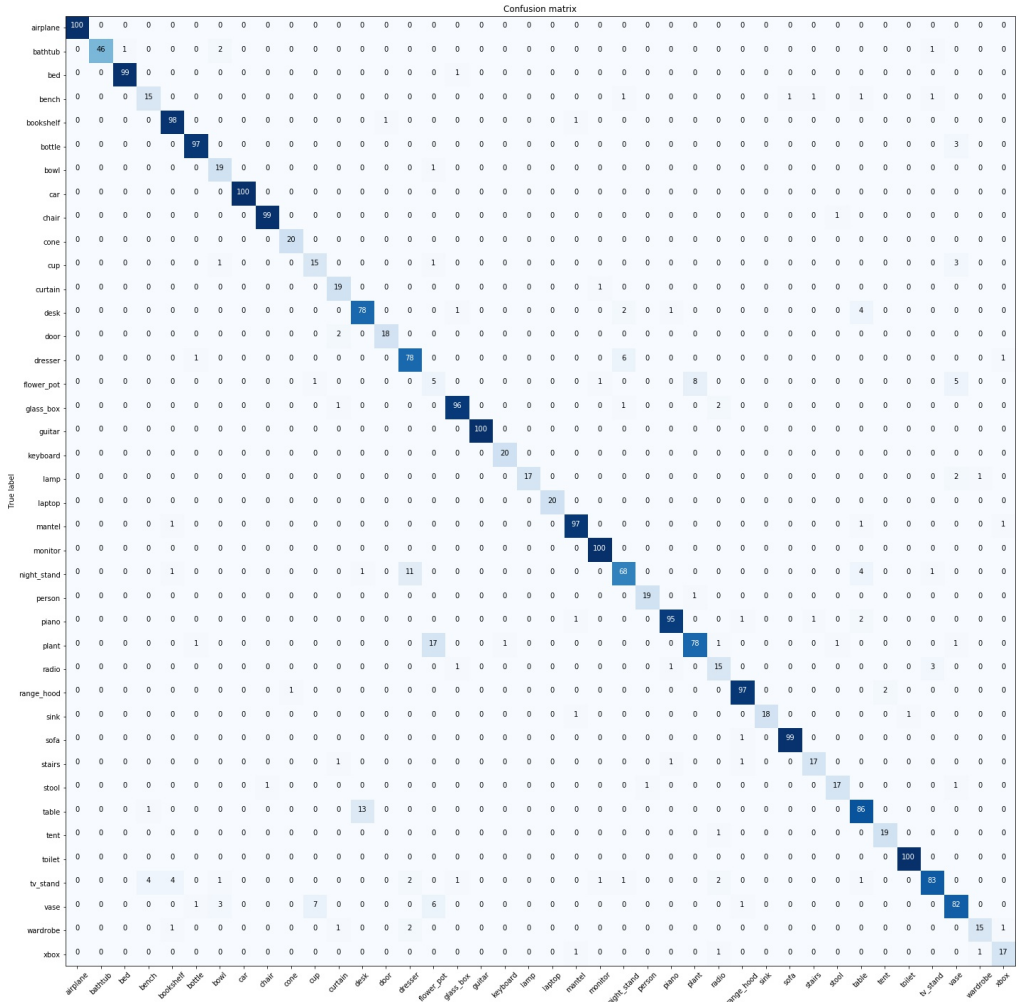


Figure 4.2: **Confusion matrix** of proposed model on ModelNet40 benchmark; row: predicted label; column: true label. Dark blue values outside of diagonal indicate their impact on instance-wise accuracy.

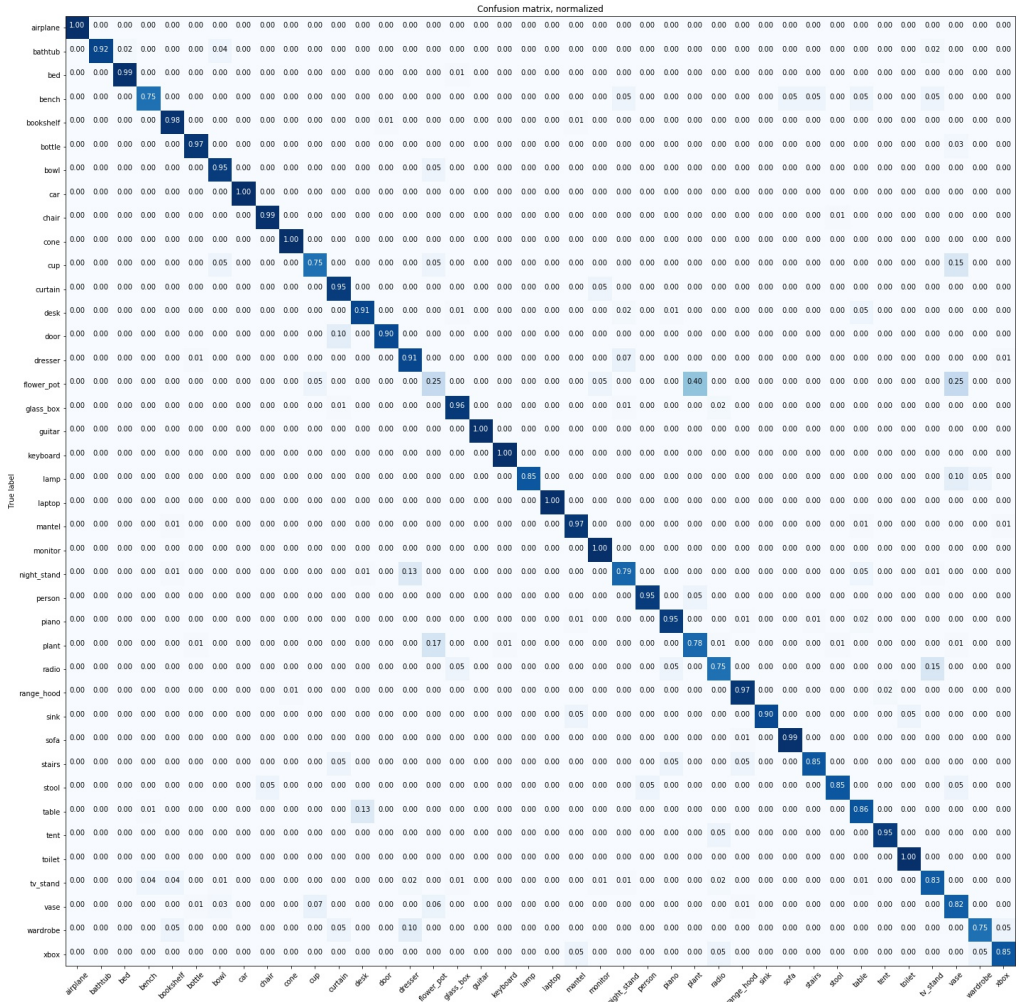


Figure 4.3: **Normalized confusion matrix** of proposed model on ModelNet40 benchmark; row: predicted label; column: true label. Dark blue values outside of diagonal indicate their impact on class-wise accuracy.

4.3.1 Error Analysis

Confusion matrix and normalized confusion matrix are computed and shown (Fig. 4.2, Fig. 4.3). Confusion matrix shows frequently mistaken classes that harm the instance-wise accuracy. Similary, normalized confusion matrix shows the confused classes that harm the class-wise accuracy. Confusion of plant and flower pot and confusion of table and desk are shown to damage both accuracy. Those classes have unique shape from the other categories, but confused two classes are very similar. Normalized confusion matrix shows that the accuracy of flower-pot class is only 25%. This is caused by training data imbalance: the number of flower pot is 149 in training set, where it is 239 and 475 in case of plant and vase.

Table 4.1: **Ablation study of our model**. 'Point Only' model predicts a class only by attended point coordinates without the local feature. Per-point features are aggregated to effectively use our attentions. 'Random Sampling' model chooses representative points of an input point cloud randomly. 'Local Feature Only' model doesn't concatenate the attended coordinates to local feature. Each component of our model is shown to contribute to learning.

	Instance	Class
Point Only (w/o attended feature)	85.0	80.3
PointNet baseline [4]	87.1	-
Point Only (w/ attended feature)	88.0	84.4
Random Sampling	91.8	88.8
Local Feature Only	90.6	86.9
Full Usage	**92.1**	**89.3**

4.3.2 Module Contribution Analysis

In this section, we investigate the contribution of our modules by experiments (Table 4.1). Attention module is compared with the random sampling strategy, and it shows that appropriate selection of points encourages the model to learn better feature. We also compared our method without local feature extraction, to show that local feature plays important role in recognition. Furthermore, we found that applying attentions to

both coordinates and per-point feature helps learning of global feature. It outperforms the baseline model of PointNet [4], which doesn't use transformer networks. Finally, we also present the effectiveness of concatenation of local feature and their associated coordinates by using only local features to predict the class.

4.3.3 Attention Results

Representative points computed by the attention module are shown (Fig. 4.4. Attended points are well separated spatially as a result of using hinge loss of pairwise distances. Still, it preserves the original point cloud well due to the Chamfer distance. Effect of two loss is visualized in Fig. 4.6.

Few attention maps are visualized in Fig. 4.4. Visualization shows that the important local structures are well captured by proposed attention module. While some attentions highlight very few points, the other attentions assign values to the local region. Our model only used attention for a representative point selection, but the results suggest that this attention can be used to aggregate the features as well.

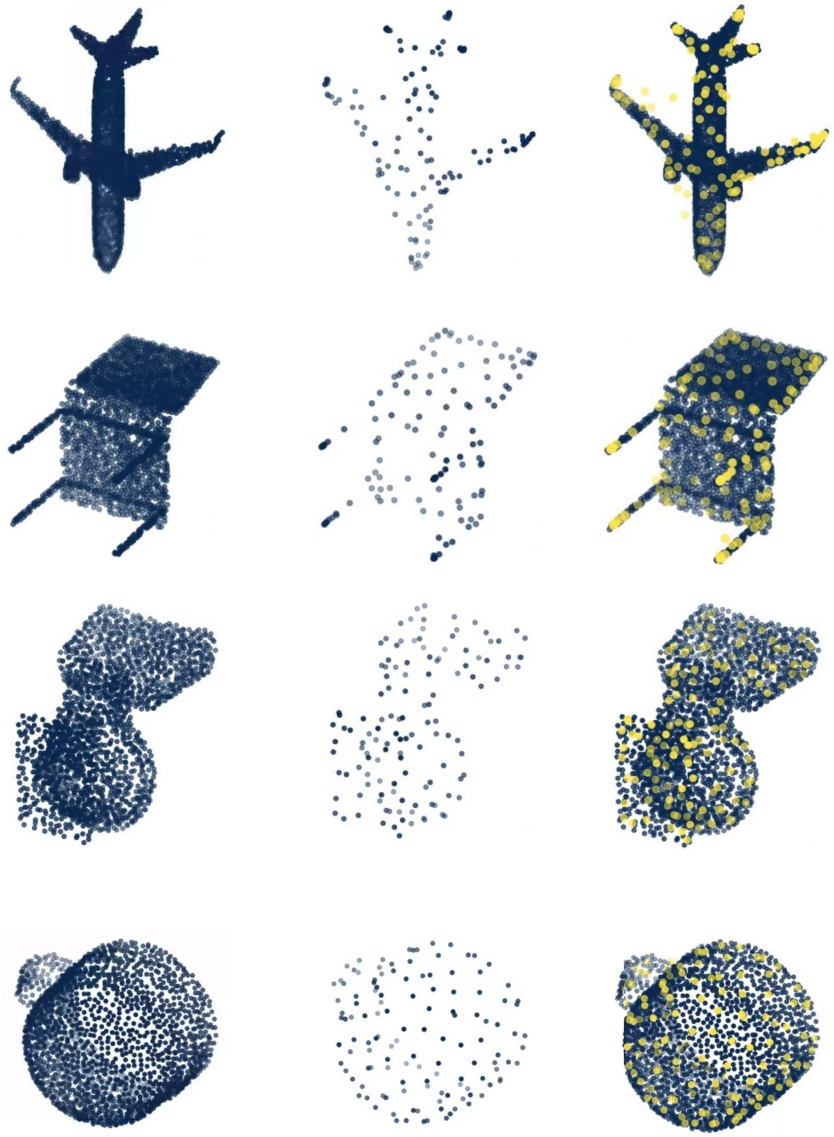


Figure 4.4: **Attention results** of proposed model. Shapes are randomly selected from the test set and visualized. Left column is the original input points, middle column is attended points, and right column visualizes both. Blue points are the original input points and yellow points are attended points computed by multiplying attention map to the input.

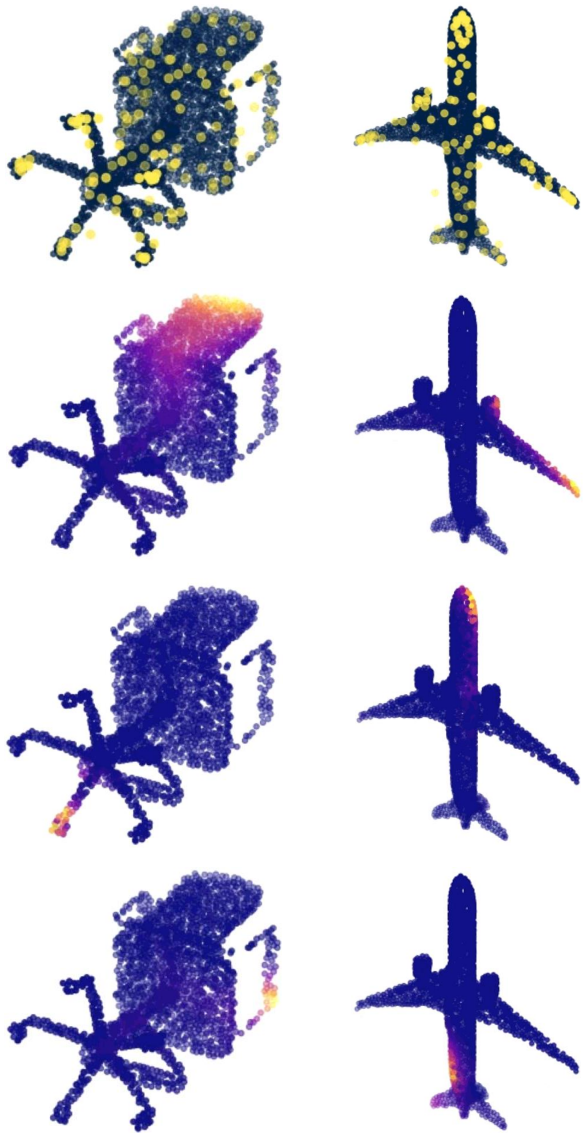


Figure 4.5: **Attention maps**. Input point cloud and their attended points are visualized on the top row. Blue points are the original input points and yellow points are attended points. Below the top column, randomly selected attention maps are visualized by color. Attention value is high if the color of point is closer to yellow.

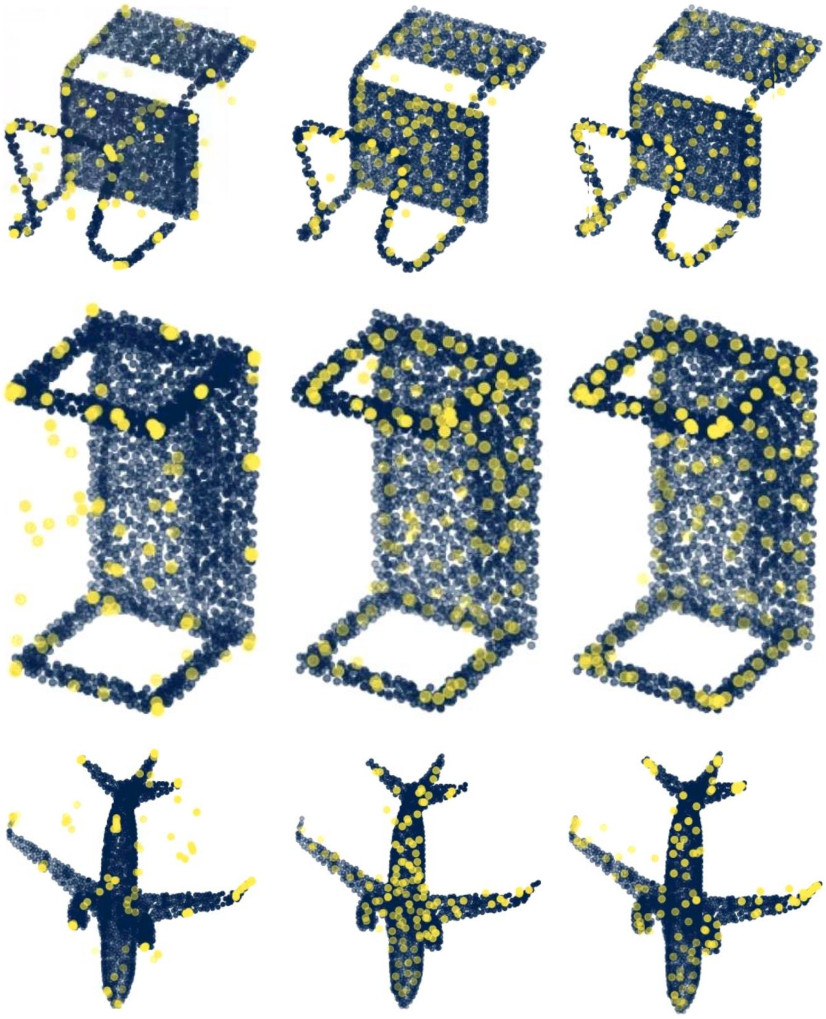


Figure 4.6: **Attention differences by loss**. Attended points generated by three different models trained with different losses are visualized. Left: margin loss, middle: Chamfer distance, right: both. Utilizing both loss term encourages the network to attend to different regions and the attended points resemble the original point cloud.

Table 4.2: Comparison of Classification Accuracy in ModelNet40 dataset. Accuracies are measured based on the reported input size on the left, but performance on the other setting is provided with the number of input points.

	Input	Instance	Class
PointNet [4]	1024×3	89.2	86.2
PointNet++ [5]	5000×6	91.7(90.7, #1024)	-
DeepSets [12]	5000×3	90.0	-
Kd-net [3]	$2^{15} \times 3$	91.8	88.5
ECC [14]	1000×3	87.4	83.2
OctNet [27]	128^3	86.5	83.8
SO-Net [22]	5000×6	**93.4**(90.7, #2048)	**90.8**(87.3, #2048)
PointCNN [28]	1024×3	92.2	88.1
DGCNN [6]	1024×3	92.2	90.2
Ours	2048×3	**92.4**	89.6

5 Conclusion

This work is an efficient 3D shape classification algorithm based on PointNet, DGCNN and the differentiable pooling, which first attends to a few representative points and extracts local geometric information of those points. Representative points are computed by multiplying input point cloud with the attention maps which are learned in an end-to-end fashion. Also, considering local neighborhood demands substantial computation since point cloud is an unordered and irregular representation. Proposed algorithm computes less than existing local feature computation since it only computes features of attended points. Additionally, high dimensional embedding can be used to learn more complex structures. Proposed algorithm achieved comparable performance to the state-of-the-art techniques in ModelNet40 dataset.

요 약 문

본 연구는 점 구름을 직접 입력으로 받아 물체의 종류를 출력하는 3차원 물체 분류 알고리즘이다. 제안된 알고리즘은 기존에 제안되었던 포인트넷을 기반으로 입력 점 구름에서 소수의 대표점들을 추측하고, 대표점들 주변의 지역적, 기하학적 정보를 그래프 컨벌루션의 한 종류로 추출한다. 소수의 대표점들은 입력 점 구름에서 어텐션을 계산하는 네트워크에 의해 얻어질 수 있으므로 전체 구조를 손실 함수와 확률적 기울기 강하 방법으로 학습하는 과정중에 같이 학습된다. 한편, 점 구름은 순서가 없고 비정형적인 표현으로, 이미지와 같은 정형 데이터와 달리 특정 점 근처의 정보를 계산하려면 전체 점에서 근처 점을 탐색하는 등의 작업이 필요하여 상당한 계산을 요구한다. 제안한 알고리즘은 어텐션 기반의 풀링을 이용하여 먼저 점의 수를 줄인 후에, 줄인 점들의 특징벡터를 계산하므로 모든 점에 대해 비효율적으로 수행되는 주변 점 탐색과 특징 추출에 사용되는 계산을 줄일 수 있다. 그리고 점의 수를 줄였기 때문에 각각의 대표점들은 더 많은 이웃들을 참고하여 특징을 계산할 수 있다. 또한, 중간 임베딩 레이어들도 더 높은 차원의 벡터 표현을 사용할 수 있게 되어 복잡한 구조를 학습할 수 있게 된다. 3차원 물체 분류 벤치마크인 모델넷40에서 분류기의 정확도를 측정한 결과, 본 알고리즘이 현재 최고 수준의 기술과 견줄만한 성능을 달성했다.

Bibliography

[1] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

[2] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 36, no. 4, 2017.

[3] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 863–872, IEEE, 2017.

[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, p. 4, 2017.

[5] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, pp. 5099–5108, 2017.

[6] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *arXiv preprint arXiv:1801.07829*, 2018.

[7] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition," in *IROS*, 2015.

[8] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

[9] C. Wang, M. Pelillo, and K. Siddiqi, "Dominant set clustering and pooling for multi-view 3d object recognition," in *Proceedings of British Machine Vision Conference (BMVC)*, vol. 12, 2017.

[10] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2016.

[11] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in Neural Information Processing Systems*, pp. 3391–3401, 2017.

[13] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International Conference on Machine Learning*, pp. 1263–1272, 2017.

[14] M. Simonovsky and N. Komodakis, "Dynamic edgeconditioned filters in convolutional neural networks on graphs," in *Proc. CVPR*, 2017.

[15] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[16] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

[17] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proceedings of the IEEE international conference on computer vision workshops*, pp. 37–45, 2015.

[18] H. Su, V. Jampani, S. Deqing Sun, E. Maji, M.-H. Yang, J. Kautz, *et al.*, "Splatnet: Sparse lattice networks for point cloud processing," 2018.

[19] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 4, 2018.

[20] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] F. Groh, P. Wieschollek, and H. Lensch, "Flex-convolution (deep learning beyond grid-worlds)," *arXiv preprint arXiv:1803.07289*, 2018.

[22] J. Li, B. M. Chen, and G. H. Lee, "So-net: Self-organizing network for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9397–9406, 2018.

[23] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical

graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems*, pp. 4801–4811, 2018.

[24] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[25] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 3212–3217, Citeseer, 2009.

[26] L. Landrieu, H. Raguet, B. Vallet, C. Mallet, and M. Weinmann, "A structured regularization framework for spatially smoothing semantic labelings of 3d point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 132, pp. 102–118, 2017.

[27] G. Riegler, A. O. Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 3, 2017.

[28] Y. Li, R. Bu, M. Sun, and B. Chen, "Pointcnn," *arXiv preprint arXiv:1801.07791*, 2018.

ACKNOWLEDGMENTS

대학원에 입학하고 얼마 지나지 않은 것 같은데 졸업을 하게 되었습니다. 지난 석사 기간을 돌아보면 감사의 말을 전하고 싶은 분들이 정말 많습니다. 먼저, 연구뿐만 아니라 많은 일에 조언을 아끼지 않으시고, 열정적으로 지도해주신 한보형 교수님께 감사드립니다. 또, 포항에서 지속적으로 저를 지도해주시고 도움을 주신 곽수하 교수님, 조민수 교수님께도 감사드립니다.

컴퓨터비전연구실 규모가 커지면서 저와 길게, 또 짧게 만난 동료분들에게 모두 고맙습니다. 먼저 부족한 제 의견을 들어주시고 또 고민이 있으면 들어주시고 궁금한 것을 항상 해결해주셨던 연구실 선배분들에게 진심으로 감사드립니다. 제 말을 항상 경청해주시는 진희누나, 동훈이형, 현우형, 승훈이형, 무열이형, 찬미누나, 현섭이형, 종환이형, 택근이형, 일채형, 홍석이형, 웅기형 고맙습니다. 제 4공학관 연구실에서 오랜시간 함께했던 솔애누나, 선우누나, 단일이 정말 고맙습니다. 정통연 새로운 연구실에서 즐거운 시간을 보낸 든솔이형, 원표형, 동주형, 승관이, 아현이, 우현이, 주원이형, 희승이형, 주홍이형, 남엽이형, 성연이, 보승이, 세현이, 지철이형, 민교형, 정빈누나, 종민이, 기현이 모두 정말 감사합니다. 제 짧은 석사 기간에 이렇게나 많은, 좋은 사람들을 만나서 유쾌한 시간을 보낼 수 있었던 것은 축복이라고 생각합니다. 포항의 두 연구실에서 지낸 시간들을 돌이켜보면 정말 활기차고 즐거웠던 것 같고 떠나려니 아쉬운 마음이 드네요. 또, 서울대 연구실에 있는 성욱이형, 민수형, 서현이, 고려대학교의 장훈씨에게도 감사의 인사를 드립니다. 모두 있는 장소는 다르지만 한결같이 열정적이셔서 원하는 바를 이룰 수 있을 것 이라고 생각합니다.

어려울 때 힘이 되어주었던 친구들, 그리고 항상 응원해주고 지지해줬던 부모님과 누나 정말 고맙고 감사합니다.