BoW Pooling: A Plug-and-Play Unit for Feature Aggregation of Point Clouds

Xiang Zhang[1]*, Xiao Sun[2,1]*, Zhouhui Lian[1]†
[1]Wangxuan Institute of Computer Technology, Peking University, Beijing, P.R. China
[2]Meituan
{1801210733, lianzhouhui}@pku.edu.cn, sunxiao10@meituan.com

Abstract

Point cloud provides a compact and flexible representation for 3D shapes and recently attracts more and more attention due to the increasing demands in practical applications. The major challenge of handling such irregular data is how to achieve the permutation invariance of points in the input. Most of existing methods extract local descriptors that encode the geometry of local structure, followed by a symmetric function to form a global representation. The max pooling usually serves as the symmetric function and shows slight superiority compared to the average pooling. We argue that some discrimination information is inevitably missing when applying the max pooling across all local descriptors. In this paper, we propose the BoW pooling, a plug-and-play unit to substitute the max pooling. Our BoW pooling analyzes the set of local descriptors statistically and generates a histogram that reflects how the primitives in the dictionary constitute the overall geometry. Extensive experiments demonstrate that the proposed Bow pooling is efficient to improve the performance in point cloud classification, shape retrieval and segmentation tasks and outperforms other existing symmetric functions.

Introduction

3D shape analysis is one of the most fundamental topics in computer vision and computer graphics fields. With the development of computing resources and the availability of large-scale 3D model repositories (Wu et al. 2015; Yi et al. 2016), extensive attention have been paid in 3D shape analysis. There are several popular data formats to represent 3D shapes, including voxels, multi 2D views, point clouds and meshes. Among them, point cloud analysis has drawn a significantly increasing amount of interests in the past few years both in academic research and industry, motivated by the advancement of autonomous driving, indoor navigation, robotics, and other wide range of applications.

Owing to the development of 3D sensors and computing resources, point cloud becomes easy to access. Despite the fact that point cloud is a light-weight format to represent flexible 3D shapes, it is unsuitable to be directly processed by neural networks because of the irregularity of the arrangement. Different from neural networks that deal with well-organized data, point-based models directly accept point clouds as input. The key to these approaches is to design a way to aggregate features and eliminate the impact of the permutation of points.

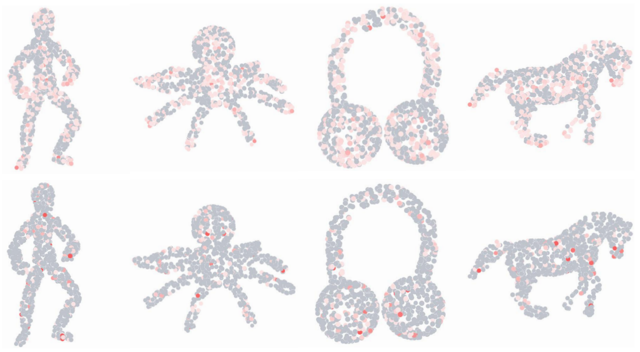


Figure 1: Visualization of the contribution of each point to the global representation. Results in the first row are from the original DGCNN, while results in the second row are from DGCNN with our BoW pooling. Redder denotes greater contribution.

According to the different implementations of such goal, neural networks that deal with point clouds can be roughly categorized into two classes. One idea is to design a symmetric function applied across all points on the point cloud (Qi et al. 2017a; Wang et al. 2018; Sun, Lian, and Xiao 2019). A typical choice of the symmetric function is the average pooling or max pooling. The backbone of these networks attaches the feature of local structure to each point, and uses a symmetric function to automatically aggregate the features of informative points and form a global representation. Despite that the max pooling usually outperforms the average pooling in terms of aggregating point features, we argue that there are still two main drawbacks when directly using it. First, applying the max pooling across features of points may lead to an average contribution of each point to the global representation. Since the global representation is obtained by selecting the biggest value across all points for each dimension, the contribution of a point is measured by finding out how many values of the global represen-

*Denotes equal contribution.
†Corresponding author.

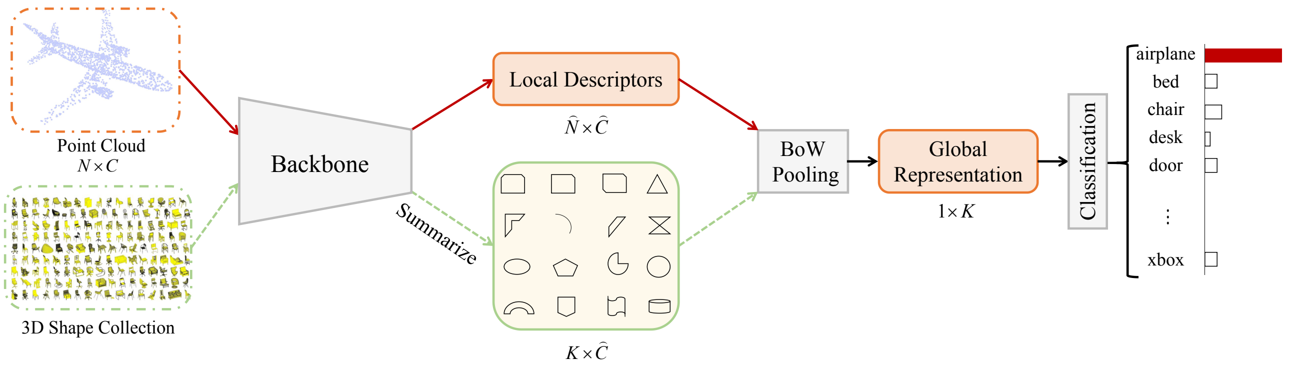


Figure 2: The pipeline of our model that plugs the proposed BoW pooling unit in an existing point-based network. The BoW pooling replaces the original pooling method for aggregating local descriptors.

tation are from such point. The visualization of the contribution of each point is demonstrated in Figure 1, which shows that a large number of points tend to make similar contributions to form the global descriptor. It has been proved that points in different positions are of unequal importance for the geometric perception (Sun, Lian, and Xiao 2019). Averaging the contributions means that the neural network pays less attention to key local points, which we know are important for tolerating the pose variations in processing non-rigid shapes (Lian et al. 2013). Second, applying the max pooling across all the points lacks reasonable interpretation, although it often brings fine results in point cloud analyzing tasks. Another idea of avoiding permutation problem is to sample some center points from the original data and reduce the resolution of point cloud gradually. The feature of points lying in the neighborhood of central point flows to the center. For instance, ShellConv (Zhang, Hua, and Yeung 2019) partitions the local points into multiple shells according to the distances to the center and the statistic of each shell is summarized by a max pooling over all the points in the shell. Extracting features layer by layer shows slight superiority in terms of performance compared to symmetric function, but usually accompanies with complex network architecture.

As we know, the Bag-of-Words model was initially used in text classification and information retrieval tasks (Boulis and Ostendorf 2005; Croft, Metzler, and Strohman 2010) and has achieved great success in computer vision field (Lian, Godil, and Sun 2010). The main idea is to first quantize local descriptors into visual words and then represent each image as a histogram that counts the points lying near each clustering center. In this paper, we propose a plug-and-play unit, the BoW pooling, to substitute for the max pooling operation and serve as the symmetry function. The basic assumption is that objects from different categories are not independent. Each 3D model can be viewed as a combination of primitives, differing in the type, number, and arrangement of the primitives. We construct a dictionary consisting of the feature of primitives, and then the global representation of an object can be expressed as a histogram, showing how the primitives construct the 3D object. In addition, we introduce the truncated linear unit (TLU) to avoid the average contribution problem. The intuition is that we increase the difficulty of mining local features by applying TLU on the local descriptors, aiming to restrict the representation capacity of local features. The network is trained towards suppressing the expression of local points that are not crucial for perception. As a result, the difference between the discrimination of local features is expanded.

Major contributions of this paper are threefold.

- We propose the BoW pooling, a plug-and-play unit that substitutes for the symmetric functions in existing methods for the feature aggregation of point clouds. A novel dictionary update strategy is explored and discussed.
- The truncated linear unit is introduced to suppress the expression of unimportant local descriptors, so that most of the contributions to form the global representation are from a small number of key points.
- Experiments conducted on four publicly-available datasets demonstrate the superiority of the proposed pooling method compared to other symmetry functions in dealing with point clouds.

Related Work

Deep Learning on Voxel-based Methods

Point clouds and meshes are two most commonly used formats in 3D vision applications due to their compactness and flexibility. They are composed of a list of unordered points or faces, which are hard to be directly processed by neural networks. Pioneer methods (Wu et al. 2015; Maturana and Scherer 2015; Qi et al. 2016) convert meshes or point clouds into voxels. Voxel-based 3D shapes meet the requirements of 3D convolutions and can be directly fed into neural networks. However, the 3D convolutions occupy large amounts of memory, limiting the resolution of 3D shapes and increasing the quantization loss enforced by the 3D grid. Efficient methods (Wang et al. 2017; Tatarchenko, Dosovitskiy, and Brox 2017) have been proposed based on the idea of space partition to alleviate the problem of low resolutions, but still depend on the division of a bounding volume rather than the semantic local geometric structure.

Deep Learning on View-based Methods

View-based methods avoid the above-mentioned problems that the voxel-based methods meet. They project 3D shapes into 2D grids, so that many effective convolutional architectures can be employed. (Guo et al. 2016) represents a 3D shape with a collection of 2D views from multiple perspectives. The convolutional neural network extracts the feature of each image, followed by a view pooling to eliminate the impact of the views' permutation. GVCNN (Feng et al. 2018) introduces a grouping module to estimate the content discrimination of each view, and views of different discriminative levels are combined into shape-level descriptors according to their discriminative weights. SeqViews2SeqLabels (Han et al. 2018) uses an encoder-decoder structure based on RNN to map sequential views to sequential labels step by step. Increasing the number of views may carry more details, but also bring in redundant clues.

Deep Learning on Mesh-based Methods

The mesh representation of 3D shapes is dominant for rendering and storing 3D models in computer graphics. The mesh data consists of vertices, edges and faces, and different types of connections among them are complex to define. Besides, the number of elements in meshes varies dramatically, and the permutation of them are arbitrary, which make it less exploited in the academic research. (Monti et al. 2017) propose a unified framework allowing to generalize CNN architectures non-Euclidean domains (graphs and manifolds). MeshNet (Feng et al. 2019) regards the face as the unit and splits the feature of faces into spatial and structural ones, followed by a mesh convolution block for aggregating neighboring features. MeshSNet (Lian et al. 2019) employs graph-based blocks to extract contextual features of multiple scales, and then integrates local-to-global geometric features to comprehensively characterize mesh cells for the segmentation task. It is difficult for neural networks to deal with meshes due to their property of complexity and irregularity.

Deep Learning on Point-based Methods

Altering the permutation of the points does not change the geometry, but may result in different representations when feeding point cloud into neural networks. The pioneer work, PointNet (Qi et al. 2017a), adopts point-wise convolutions to map the original 3D coordinates to a high-dimensional feature space, followed by a max pooling operation across all the points to eliminate the impact of the permutation of points. PointNet neglects to mine the local shape structure, making it hard to distinguish the tiny diversity between similar 3D shapes. KCNet (Shen et al. 2018) utilizes the Kernel Correlation layer on the K-NN Graph to explore the local geometry. SpiderCNN (Xu et al. 2018) and DGCNN (Wang et al. 2018) take the point-pair relation into account and employ Edge-related operations to incorporate neighboring knowledge. The above methods all use a so-called symmetric function applied on the whole points to achieve the permutation invariance. Specifically, the max pooling or average pooling serves as the symmetric function and compresses the information of the whole points into a single point that encodes the global geometry feature, in which some important information is unavoidable missing. PointNet++ (Qi et al. 2017b) subdivides each point cloud into several subsets and applies a simplified PointNet on each subset. These local features are grouped together to obtain a global feature. PointCNN (Li et al. 2018) samples points from original ones to reduce resolution and learns a transform matrix using X-conv that pre-multiplies a local feature matrix to alleviate the impact of permutation. A-CNN (Komarichev, Zhong, and Hua 2019) divides the space into several rings and arranges the points of a ring in the counterclockwise ordering, then annular convolutions are applied. ShellConv (Zhang, Hua, and Yeung 2019) partitions the neighboring points into multiple shells and the statistic of each shell is obtained by implementing the max pooling over all the points in the shell. Reducing resolution layer by layer decreases the information loss compared to the symmetric function at a cost of increasing expenditure of inference time and more complex architecture. This paper proposes a symmetric function, the BoW pooling, that integrates local descriptors into a global representation. Compared to other symmetric functions, the BoW pooling is capable of obtaining more informative representations to boost the accuracy in multiple tasks.

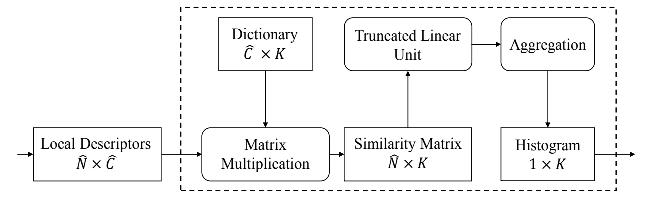


Figure 3: The architecture of the proposed BoW pooling. The dictionary is composed of the K clustering centers, and the distance from each point to every clustering center is calculated. The histogram is generated by aggregating the points that lie close to each center.

Method

In this section, we introduce the proposed BoW pooling. The goal of our BoW pooling is to integrate the features of all points and form a global representation for downstream tasks, and the detailed operations are demonstrated in Figure 3. We present the design of the proposed BoW pooling in the following subsections.

BoW Pooling: A Generalized BoW Layer

Given a set of C-dimensional local descriptors of a 3D shape $\{x_i\}$ as input, and K clustering centers $\{c_k\}$ as BoW parameters, the output of our BoW pooling is a K-dimensional vector V, encoding the global information of point cloud. The $j-th$ element of V is formulated as follows

$$V_j = \sum_{i=1}^{n} I(dist(x_i, c_j) < dist(x_i, c_{k \neq j})), \tag{1}$$

where I denotes the $0-1$ indicator function, i.e., it equals to 1 if c_j is the closest clustering center to x_i, and 0 otherwise.

Intuitively, each element in V represents the number of points that lie in the neighborhood of the corresponding center. The clustering centers are grouped as a dictionary that consists of the features of primitives. The cosine distance is chosen to measure the similarity between local descriptors and clustering centers. Thus, the similarity matrix can be easily obtained by the matrix multiplication between local descriptors and the dictionary matrix.

Soft Point Assignment to the Points

Finding the closest center for every point and assigning it to the point using the indicator function is non-differentiable. To embed the BoW pooling into the pipeline of neural networks and make it trainable, we propose two alternatives that replace the hard assignment method.

Softmax-like function. The first one is a softmax-like function that normalizes the similarity

$$V_{ij} = \frac{e^{\alpha x_i^T c_j}}{\sum_{k=1}^{K} e^{\alpha x_i^T c_k}}, \tag{2}$$

where the scalar α is a hyper parameter that controls the increase of response with the magnitude of similarity. The softmax operation is conducted along the dimension of different centers for every point, showing how much the point belongs to different centers. Then, the weight of the descriptor x_i is assigned to the center c_k exponentially proportional to their proximity. Note that the softmax-like function degrades into the hard assignment if α is set to ∞.

Truncated linear unit. TLU is designed based on the idea that a point may be negligible for the accumulation of a far-away center. Thus, we calculate the distances between a point and all centers, and assign the point to the top-$\mathcal{L}$ nearest centers

$$V_{ij} = \begin{cases} x_i^T c_j, & \text{if } j \in I_i \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where I_i is a set containing the indexes of the top-$\mathcal{L}$ nearest centers to the point x_i. We further discuss the application of TLU from the perspective of the activation function. Suppose q_i is a $\mathcal{D}$-dimensional feature vector of the point p_i, and $\mathcal{T}$ is short for the TLU operation, then

$$\mathcal{T}(q_i) = (q_i^1 \cdot \beta^1, ..., q_i^{\mathcal{D}} \cdot \beta^{\mathcal{D}}), \tag{4}$$

where β^n equals to 1 if q_i^n is within the Top-$\mathcal{L}$ greatest values and 0 otherwise. It suppresses the expressions of local descriptors when aggregating them into the global representation

$$f_{global} = \mathcal{A}(\mathcal{T}(q_i), ..., \mathcal{T}(q_n)), \tag{5}$$

where $\mathcal{A}$ is the aggregation operation, which sums up the values assigned to each center and generates a histogram as the global representation. Through training, TLU increases the difficulty of mining local features and expands the gap between the discrimination capacities of different regions. As a result, a small number of important local descriptors provide most of the contributions to the global representation, which helps the network to tolerate shape variations within a class, especially for non-rigid shapes.

Mixing Dictionary Update Strategy

The most direct way to update the dictionary is applying the back-propagation algorithm, the same as other parts of the network. The dictionary is updated based on the gradient with respect to the loss function. In this way, the dictionary acts the same as the conventional fully-connected layer and evolves towards a better affine transformation function that maps the input features to more discriminative ones. Though it may obtain an overview of all training samples after one epoch of training, it fails to summarize intrinsic primitives shared by objects from different categories.

From the perspective of statistics, the observed data X, i.e., the local descriptors, are connected to the elements in the dictionary by latent variables Z. The latent variables are actually indicators that show the proportion of the observed data endowed with the elements $\{c_i\}$ in the dictionary. Thus, we formulate the problem of finding the representative primitives $\{c_i\}$ as the parameter estimation of the complete log likelihood $\ln p(X, Z|C)$. As we know, it is efficient to resort to the expectation-maximization algorithm when unobserved variables exist. The E step on the original EM algorithm is to calculate the expected log likelihood, while in the case when the latent variables Z are indicator variables which follow the multinomial distribution, it simply reduces to calculate the posterior distribution of Z

$$z_i^k = \frac{\mathcal{K}(x_i, c_k)}{\sum_{j=1} \mathcal{K}(x_i, c_j)}, \tag{6}$$

where $\mathcal{K}$ is a kernel of many choices, such as the Euclidean distance, RBF kernel and so on. In the M step, the elements of dictionary are updated by

$$c_k = \frac{\sum_i z_i^k x_i}{\sum_j z_j^k}. \tag{7}$$

Note that the K-means algorithm is a special case of such EM updates if Z is defined as

$$z_i^k = \begin{cases} 1, & \text{if } k = \arg\min_j ||x_i - c_j|| \\ 0, & \text{otherwise}. \end{cases} \tag{8}$$

For a given batch of samples, the dictionary is adjusted iteratively, which is hard to balance the consumed time in training and the efficiency of convergence. Alternatively, we provide a heuristic method to solve the problem. The basic idea is to reduce the frequency of dictionary updating, for the requirement of a highly-accurate dictionary is not necessary at a certain step during training. The strategy is that we carry out the EM algorithm for every T-epochs update of BP. More specifically, the dictionary is re-initialized using the results obtained by applying the EM algorithm on the latest observed data, and the initial c_i are set to the elements of the dictionary learned by back propagations. Such mixing update steps make up one round of training, and the whole training procedure possesses several rounds. In this way, the dictionary comes from the local features themselves, and the elements in the dictionary serve as the basis vectors that construct the feature space.

Method	Mean Class Accuracy	Overall Accuracy
PointNet (Qi et al. 2017a)	86.0	89.2
PointNet++ (Qi et al. 2017b)	87.6	90.7
DGCNN (Wang et al. 2018)	90.2	92.9
3D-GCN (Lin, Huang, and Wang 2020)	89.1	92.1
PointNet (+NetVLAD) (Ali C 2019)	-	87.1
DGCNN (+NetVLAD)(Ali C 2019)	-	91.2
PointNet (+BoW)	86.4	90.0
PointNet++ (+BoW)	88.6	91.1
DGCNN (+BoW)	91.0	93.4
3D-GCN (+BoW)	89.6	92.4

Table 1: Classification accuracies (%) of different methods evaluated on ModelNet40.

Method	Avg	Max	Avg+Max	BoW
PointNet	86.9	89.2	89.2	90.0
PointNet++	89.6	90.7	90.5	91.1
DGCNN	93.0	92.7	92.9	93.4

Table 2: Classification accuracies (%) of three methods with different symmetric functions evaluated on ModelNet40.

Experiments

In this section, we evaluate the proposed BoW pooling unit in different tasks: point cloud classification, shape retrieval and segmentation. In order to keep similar number of parameters to the counterpart, we first remove the last fully-connected layer before the global pooling layer from the original architecture, which usually expands the feature dimension to 1024. Then, we plug our BoW pooling unit in several mainstream networks and substitute for original pooling functions. Without loss of generality, three representative mainstream networks are chosen as the backbone, i.e., PointNet (Qi et al. 2017a), PointNet++ (Qi et al. 2017b) and DGCNN (Wang et al. 2018).

Point Cloud Classification

The classification experiment is conducted on ModelNet40 (Wu et al. 2015), which is composed of 12311 CAD models in 40 classes. We use the same train-test split as PointNet. The network architecture used for the classification task is shown in Figure 2. The input feature of a point cloud is composed of 3-dimensional coordinates. The experimental settings keep the same as the original approach in terms of hyper parameters and the optimizer. The adapted networks with our BoW unit are trained for 400 epochs in total using the mixing dictionary update strategy. Table 1 shows the classification results of four methods on ModelNet40. The performance of methods with the BoW pooling unit gets improved compared to those without the BoW unit and those with NetVLAD. Note that the major difference between them is to replace the original symmetric function with the BoW pooling, which indicates that our BoW pooling discards less information when aggregating local descriptors into the global one. The comparison of different symmetric functions are listed in Table 2.

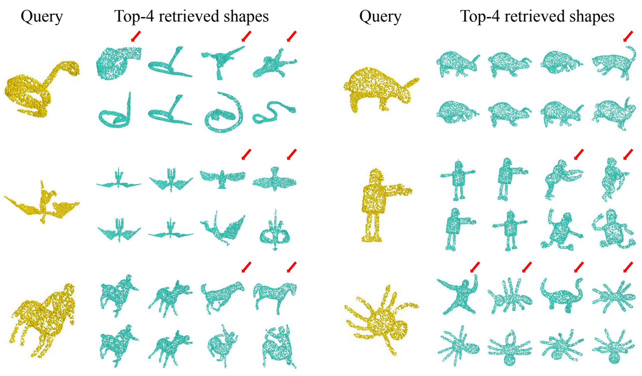


Figure 4: Shape retrieval results on SHREC15 Non-rigid. For each query shape on the left column, we present two rows of Top-4 retrieval results: the top row shows results obtained by the original DGCNN and the bottom row shows results obtained by DGCNN with the BoW pooling. The obviously wrong retrieval results are marked with red arrows.

Shape Retrieval

To evaluate whether the BoW pooling helps us learn a better global descriptor, we evaluate the effectiveness of the BoW pooling in the shape retrieval task on the SHREC15 Non-rigid dataset (Lian and et al 2015). It includes 1200 non-rigid 3D shapes in 50 classes. Each class contains 24 shapes with various poses which derive from the same object. We randomly split 19 shapes into the training set and 5 shapes into the test set for each category. We uniformly sample 1024 points and normalize them into a unit sphere. The features used for retrieval are extracted from the models that are trained for the classification task. Specifically, we use the global representations in the layer right behind the symmetric function. We follow (Lian et al. 2013) to evaluate the performance of retrieval approaches using the precision-recall (PR) curve and five quantitative metrics, including nearest neighbor (NN), first tier (FT), second tier (FT), E-measure (E) and discounted cumulative gain (DCG). The quantitative results are shown in Table 3, respectively, and the PR curves are shown in Figure 5. With the help of the BoW pooling, the approaches obtain steady gains. Since the representations from different shapes are compared using the L_1 distance, the first conclusion can be drawn that the feature space, composed of the representations after the BoW pooling unit, owns better linear property. The distance between different classes becomes larger, while the variance within a class goes smaller. In addition, the approaches with our BoW pooling get more improvements when dealing with non-rigid shapes, which indicates that the representation after the BoW pooling is capable of tolerating the pose variations of an object. We visualize how the global representation attends to each point in Figure 1. The representation with the max pooling layer pays equal attention to the whole geometry, thus most local points provide similar contribution. The proposed BoW pooling suppresses the expression

Method	BoW	NN(%)	FT(%)	ST(%)	E(%)	DCG(%)
PointNet	-	44.76	26.92	36.66	12.44	49.29
PointNet++	-	82.42	63.75	74.90	20.13	80.35
DGCNN	-	62.40	41.40	48.70	14.87	61.16
PointNet	√	46.37	30.75	36.90	12.36	51.47
PointNet++	√	93.36	80.31	89.84	22.27	91.57
DGCNN	√	90.00	68.10	80.20	20.76	84.70

Table 3: Retrieval performance of different methods evaluated on SHREC15.

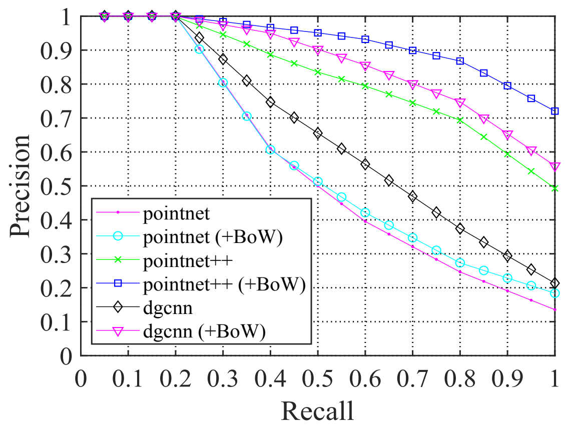


Figure 5: Precision-recall curves of representative methods evaluated on SHREC15 Non-rigid.

Dataset	ShapeNet part		S3DIS	
	Acc(%)	mIoU(%)	Acc(%)	mIoU(%)
PointNet	93.47	83.72	78.62	47.71
PointNet++	93.92	85.11	82.98	56.36
DGCNN	94.26	85.20	84.11	56.12
PointNet(+BOW)	93.69	84.06	79.78	48.17
PointNet++(+BOW)	94.17	85.33	83.75	56.90
DGCNN(+BOW)	94.48	85.39	84.28	56.65

Table 4: Segmentation results on two datasets.

of most local points and leads the global representation to focus on a small number of key points. As a result, such discriminative local descriptors are not sensitive to the deformation of geometry. The retrieved shapes for a query are visualized in Figure 4.

3D Semantic Segmentation

We also evaluate the effectiveness of the BoW pooling through point cloud segmentation experiments on ShapeNet part (Yi et al. 2016) and S3DIS (Armeni et al. 2016) datasets. The former contains 16881 shapes from 16 categories, while the latter is composed of 3D scanned point clouds of 6 indoor scenes. The quantitative results are listed in Table 4. The methods with our BoW pooling achieves stable improvements compared to those without it. Some visualization results of segmentation are shown in Figure 6. The BoW pooling is better at handling details in the joint area of two different parts and gets more precise segmentation results.

Update Strategy	Mean Class Accuracy	Overall Accuracy
BP	90.22	93.07
mixing($T = 30$)	90.52	93.27
mixing($T = 50$)	91.01	93.44
mixing($T = 70$)	90.08	93.23

Table 5: Results of our methods using different dictionary updating strategies.

Ablation Studies

In this section, ablation studies are conducted to assess the effectiveness of each part. Without loss of generality, we conduct experiments on ModelNet40 to evaluate the settings of the BoW pooling based on DGCNN.

The effect of dictionary update strategies. In Table 5, we show the results of different update strategies for dictionary. Though the dictionary learned by BP acquires a holistic perception of the whole training samples after one epoch of BP, it is not good at discovering intrinsic primitives that make up the dataset. We also find that setting T to 50 is better than 30 or 70. Figure 7 shows the loss curves of DGCNN and DGCNN (+BoW pooling) trained with the mixing update strategy. The training loss increases sharply when the dictionary is re-initialized and then goes down slowly. Though we get a higher training loss compared to DGCNN, the test loss is lower at the end of training. A proper explanation is that the re-initialization of dictionary based on the updated whole picture of training data considerably alleviates the overfitting problem.

The effect of different assignments. We compare the effects of different methods to assign the feature points to clustering centers. The results are shown in Table 6. The softmax assignment shows disadvantages compared to the truncated linear unit. One possible reason is that the softmax function keeps all small values that act like noise, and the accumulation of thousands of small values may affect the resulting histogram markedly. For TLU, the percentage of points that are kept is set to 0.25, 0.5 or 0.75. The parameter of percentage markedly influences the result, and a too large or too small value may result in poor performance.

The effect of dictionary size. We conduct experiments to investigate the effect of dictionary size, and the results

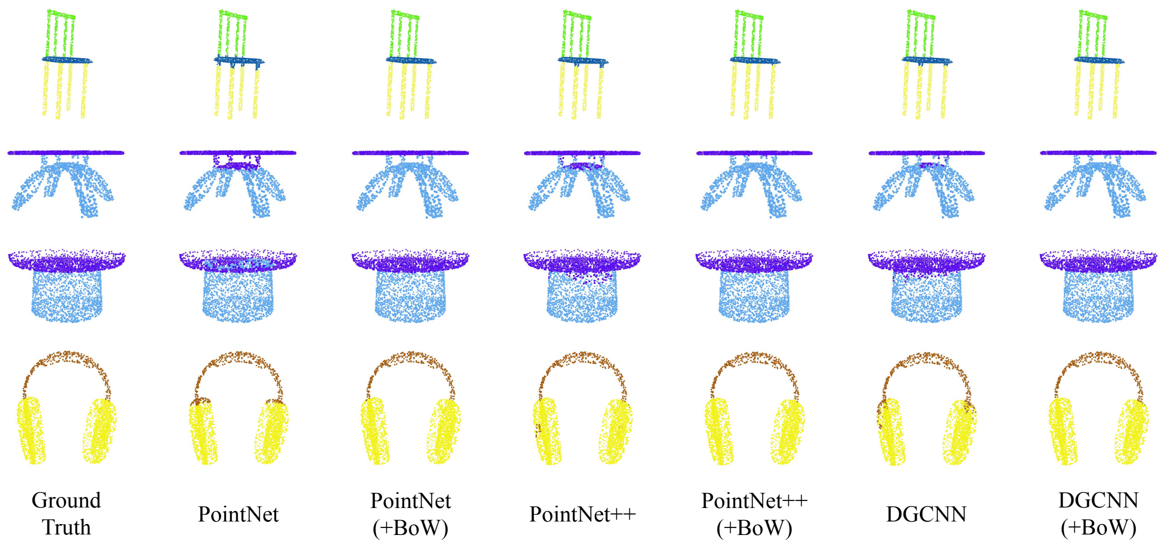


Figure 6: Qualitative results of point cloud segmentation methods run on the ShapeNet part dataset.

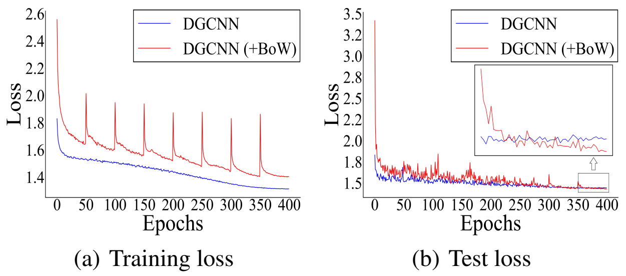


Figure 7: Loss curves in the training process.

Method	Mean Class Accuracy	Overall Accuracy
None	90.12	92.83
Softmax	86.68	91.05
TLU(0.25)	90.77	93.44
TLU(0.5)	91.01	93.44
TLU(0.75)	89.88	93.11

Table 6: Results of different assignment methods. The percentage of points kept in TLU is shown in the bracket.

are shown in Table 7. The feature dimension of elements in the dictionary is set to 512, while the number of the elements differs. The overall accuracy does not show a great difference by setting the dictionary size from 256 to 3072, and reaches the highest value at the size of 1024. Note that the dictionary is a collection of typical patterns in the training samples, the dictionary of size 1024 is enough to cover the information that the dataset contains. A larger dictionary contains too many redundant patterns, which are useless and even cause the problem of overfitting. When setting the dictionary size to 1024, the parameter number of the network with the BoW pooling is the same as that of the original one, indicating that the improvement comes from the design of the proposed BoW pooling layer.

Dictionary Size	Mean Class Accuracy	Overall Accuracy	Parameters
256	90.35	93.07	0.50M
512	90.86	93.15	0.76M
1024	91.01	93.44	1.28M
2048	90.98	93.43	2.33M
3072	90.89	93.31	3.38M

Table 7: Results of methods with different dictionary sizes.

Conclusion

In this paper, we proposed a novel BoW pooling unit to aggregate the local descriptors. The BoW pooling unit can be plugged in existing networks and replace the original pooling methods. We designed the truncated linear unit to suppress the expression of unimportant local points and lead the network to focus on a small number of key points. A novel update method of the dictionary was also explored and analyzed. We formulated the dictionary learning problem as the parameter estimation task of complete log likelihood, and introduced the mixing update strategy that incorporates the idea of the EM algorithm. Experiments on point cloud classification, shape retrieval and semantic segmentation tasks verified the superiority of the proposed BoW pooling against other symmetric functions in terms of the feature aggregation of point clouds. The applications on both rigid and non-rigid shapes show the effectiveness and universality of our proposed BoW pooling. In the future work, how to generalize the BoW pooling unit into other tasks in computer vision is worth to explore.

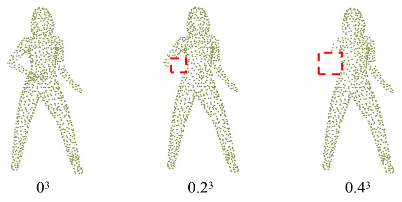
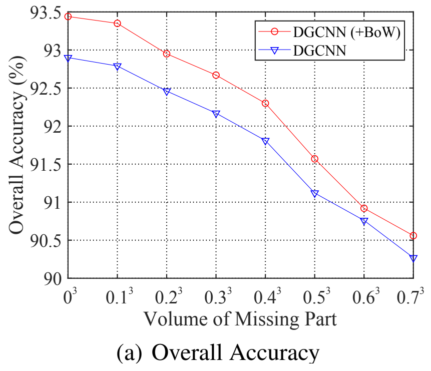
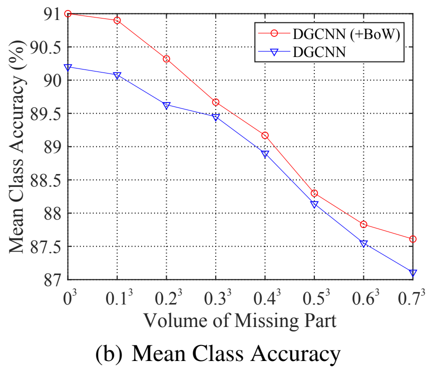


Figure 8: Illustration of removing part of points from 3D shapes in different volumes.



(a) Overall Accuracy



(b) Mean Class Accuracy

Figure 9: The performance of the BoW pooling in the presence of partiality artifacts.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No.: 61672043) and Beijing Nova Program of Science and Technology (Grant No.: Z191100001119077).

Appendix

Here, we evaluate the robustness of the proposed BoW pooling method and provide additional results of classification and retrieval experiments.

Robustness Test

The robustness of the proposed BoW pooling is tested based on DGCNN and DGCNN (+BoW pooling). The main idea is that we train the model on the complete 3D shapes, but test it on the incomplete ones. To generate the test data, we first normalize each point cloud into a unit sphere, and then drop part of 3D shape whose volume ranges from 0.2^3 to 0.4^3, as is shown in Figure 8. The percentage of missing points ranges from 1.0% to 11.2%. Figure 9 show the effect of the volume of the missing part on the performance.

Method	PointNet	PointNet++	DGCNN
NN	83.85	86.18	87.48
FT	51.56	56.16	57.52
ST	64.25	68.19	69.61
E	35.22	37.91	38.05
DCG	81.91	84.52	84.86
Method+BoW	PointNet	PointNet++	DGCNN
NN	84.64	87.06	89.42
FT	53.41	61.52	65.13
ST	65.94	74.31	76.63
E	36.39	39.95	41.46
DCG	83.01	86.96	88.08

Table 8: Retrieval performance of different methods evaluated on ModelNet40.

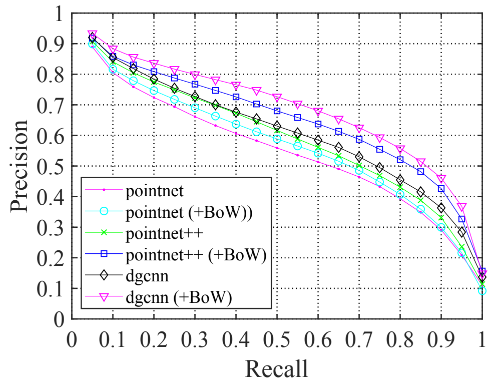


Figure 10: Precision-recall curves of some representative methods evaluated on ModelNet40.

Method	Acc	mAP
MeshNet	91.9	81.9
MeshNet (+BoW)	92.6	85.3

Table 9: The effect of our BoW pooling on a mesh-based method.

Shape Retrieval Results on ModelNet40

We also conduct shape retrieval experiments on the ModelNet40 dataset. The quantitative results and PR curves are shown in Table 8 and Figure 10.

Generalizing to a Mesh-based Method

MeshNet (Feng et al. 2019) is a mesh-based method which takes mesh data as input. To evaluate the generalization capacity of the BoW pooling unit, we replace the original final pooling unit with ours. The ratio of TLU is set to 0.5 and the dictionary size is 1024. The classification results (Acc) and retrieval results (mAP) are listed in Table 9.

References

Ali C, Shafin Rahman, L. P. 2019. Zero-shot Learning of 3D Point Cloud Objects. In *MVA*, 1–6.

Armeni, I.; Sener, O.; Zamir, A. R.; Jiang, H.; Brilakis, I.; Fischer, M.; and Savarese, S. 2016. 3D Semantic Parsing of Large-Scale Indoor Spaces. In *CVPR*, 1534–1543.

Boulis, C.; and Ostendorf, M. 2005. Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams. In *Proc. of the International Workshop in Feature Selection in Data Mining*, 9–16. Citeseer.

Croft, W. B.; Metzler, D.; and Strohman, T. 2010. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading.

Feng, Y.; Feng, Y.; You, H.; Zhao, X.; and Gao, Y. 2019. MeshNet: mesh neural network for 3D shape representation. In *AAAI*, volume 33, 8279–8286.

Feng, Y.; Zhang, Z.; Zhao, X.; Ji, R.; and Gao, Y. 2018. GVCNN: Group-view convolutional neural networks for 3D shape recognition. In *CVPR*, 264–272.

Guo, H.; Wang, J.; Gao, Y.; Li, J.; and Lu, H. 2016. Multi-view 3D object retrieval with deep embedding network. *IEEE Transactions on Image Processing* 25(12): 5526–5537.

Han, Z.; Shang, M.; Liu, Z.; Vong, C.-M.; Liu, Y.-S.; Zwicker, M.; Han, J.; and Chen, C. P. 2018. Seqviews2seqlabels: Learning 3d global features via aggregating sequential views by rnn with attention. *IEEE Transactions on Image Processing* 28(2): 658–672.

Komarichev, A.; Zhong, Z.; and Hua, J. 2019. A-CNN: Annularly convolutional neural networks on point clouds. In *CVPR*, 7421–7430.

Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; and Chen, B. 2018. PointCNN: Convolution On X-Transformed Points. In *NIPS*, 820–830.

Lian, C.; Wang, L.; Wu, T.-H.; Liu, M.; Durán, F.; Ko, C.-C.; and Shen, D. 2019. MeshSNet: Deep Multi-scale Mesh Feature Learning for End-to-End Tooth Labeling on 3D Dental Surfaces. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 837–845.

Lian, Z.; and et al. 2015. SHREC'15 Track: Non-rigid 3D shape retrieval. In *8th Eurographics Workshop on 3D Object Retrieva*, 257–266. EG 3DOR.

Lian, Z.; Godil, A.; Bustos, B.; Daoudi, M.; Hermans, J.; Kawamura, S.; Kurita, Y.; Lavoué, G.; Nguyen, H. V.; and Ohbuchi, R. 2013. A comparison of methods for non-rigid 3D shape retrieval. *Pattern Recognition* 46(1): 449–461.

Lian, Z.; Godil, A.; and Sun, X. 2010. Visual similarity based 3D shape retrieval using bag-of-features. In *2010 Shape Modeling International Conference*, 25–36. IEEE.

Lin, Z.-H.; Huang, S.-Y.; and Wang, Y.-C. F. 2020. Convolution in the Cloud: Learning Deformable Kernels in 3D Graph Convolution Networks for Point Cloud Analysis. In *CVPR*, 1797–1806.

Maturana, D.; and Scherer, S. 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 922–928. IEEE.

Monti, F.; Boscaini, D.; Masci, J.; Rodolà, E.; Svoboda, J.; and Bronstein, M. 2017. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *CVPR*, 5425–5434.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 652–660.

Qi, C. R.; Su, H.; Nießner, M.; Dai, A.; Yan, M.; and Guibas, L. J. 2016. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, 5648–5656.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 5099–5108.

Shen, Y.; Feng, C.; Yang, Y.; and Tian, D. 2018. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*, 4548–4557.

Sun, X.; Lian, Z.; and Xiao, J. 2019. SRINet: Learning Strictly Rotation-Invariant Representations for Point Cloud Classification and Segmentation. In *MM*, 980–988.

Tatarchenko, M.; Dosovitskiy, A.; and Brox, T. 2017. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *ICCV*, 2088–2096.

Wang, P.-S.; Liu, Y.; Guo, Y.-X.; Sun, C.-Y.; and Tong, X. 2017. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)* 36(4): 72.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.; Bronstein, M.; and Solomon, J. 2018. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics* 38(5).

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 1912–1920.

Xu, Y.; Fan, T.; Xu, M.; Zeng, L.; and Qiao, Y. 2018. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, 87–102.

Yi, L.; Kim, V. G.; Ceylan, D.; Shen, I.-C.; Yan, M.; Su, H.; Lu, C.; Huang, Q.; Sheffer, A.; and Guibas, L. 2016. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)* 35(6): 1–12.

Zhang, Z.; Hua, B.-S.; and Yeung, S.-K. 2019. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *ICCV*, 1607–1616.