

Adaptive GMM Convolution for Point Cloud Learning

Fei Yang

yangfei92516@163.com

Huan Wang

wanghuanphd@njust.edu.cn

Zhong Jin

zhongjin@njust.edu.cn

School of Computer Science and Engineering

Nanjing University of Science and Technology

Nanjing 210094, China

Abstract

The success of CNNs (Convolutional Neural Networks) is mainly attributed to the (translation) invariance and local pattern matching effect of convolution kernels. Accordingly, generalizing discrete convolution operation with such two properties (invariance and local pattern matching) to point cloud domain is enlightening for point cloud learning. Inspired by this, we propose an adaptive GMM (Gaussian Mixture Model) convolution (AGMMConv) operation for point cloud learning. Considering the irregularity of point clouds, we propose to represent the kernel points with a GMM, where the mean vectors denote coordinates of kernel points and the covariance matrices determine the shape of each kernel point. Meanwhile, the GMM is a distribution representation of the local geometric surface learned from the local observation, which makes the kernel adaptive to local geometric structures. The proposed convolution is intrinsically invariant to permutation and translation. Besides, potential rotation invariance can be induced from the probability representation, which is an important prior for 3D objects recognition. In convolution, a series of shared weights are associated with each GMM kernel point to match local patterns of point clouds, which allows us to learn rich features with various learnable templates by analogy to the classical image convolution. Experiments on various datasets including object-level and scene-level tasks demonstrate the effectiveness and robustness of the proposed method. Code is available at <https://github.com/yangfei1223/AGMMConv>.

1 Introduction

With the rapid development of modern sensors such as LiDAR (Light Detection and Ranging), 3D point clouds have been widely applied in autonomous driving cars, robotics, as well as some smart devices. Because of the unordered, sparse, and discrete properties of point clouds, traditional network architectures applied for grid structure data cannot be generalized to point cloud domain directly. Therefore, the learning of point cloud data has become urgent necessary in computer vision. In view of the success of CNNs, some previous methods first convert the point cloud to a regular representation (*e.g.* image and voxel representation) to cater to CNN architectures. However, the structure information will be inevitably lost during

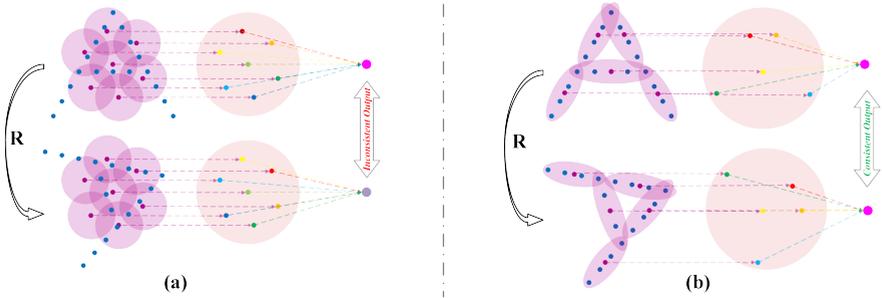


Figure 1: (a) Fixed kernel representation is sensitive to local geometric. (b) GMM kernel representation is adaptive to local geometric thus invariant to rotation.

such conversions. In the literature, there has been a number of works concentrate on devising deep networks for point cloud data. Since Qi *et al.* first propose PointNet[16] in 2017 to directly process the points without data conversions, many point-based methods have emerged out. Thus far, point-based methods have become the mainstream in the community.

Some researchers have tried constructing MLP-based[16, 17], graph-based[20, 29], and attention-based[8, 27] networks for point cloud learning. These methods aggregate local point features with a permutation invariant function. However, the representation ability of the network is largely limited by such simple operators in comparison with image convolutions. Other studies[6, 25, 30] have also attempted to define the convolution operators in point cloud domain. [9] and [30] define continuous convolutions for point cloud from the perspective of Monte Carlo, in which the convolution kernels are parameterized with a continuous function of the distance between the center point and neighbor points. The kernel weights are computed in an attentive way, which still lacks the ability of local pattern matching as image convolutions. The success of CNNs is largely attributed to the local pattern matching mechanism of discrete convolution kernels, which allows the CNN to extract rich features with various learnable kernel templates. Inspired by image convolutions, KPConv[25] defines the convolution kernels as series of fixed kernel points associated with shared kernel weights to match local patterns of a point cloud. However, the fixed kernels are sensitive to local geometric structures. To address this problem, a deformable version is further extended. Even though, the deformable kernel points are still unpredictable and cannot ensure invariant to rotation which is critically important for 3D objects recognition.

Compared with these methods mentioned above, our work is more related to the KPConv. In this paper, we stand with KPConv and propose AGMMConv, an adaptive discrete convolution in point cloud domain. Instead of defining a series of fixed kernel points intuitively, we propose to represent the kernel points with an adaptive GMM. The mean vectors and covariance matrices of the GMM determine the location and shape of each kernel point. There is an assumption that a point cloud is sampled from a continuous 3D surface. The GMM is therefore a probability distribution of the local geometric surface, which can be learned from local observations (i.e. sampled points). Hence, our GMM representation is adaptive to local geometric structures, which makes the kernels more flexible and robust. As illustrated in Figure 1, when the input cloud rotated, fixed kernel representation causes inconsistent output (see Figure 1 (a)), while our GMM kernel representation can be adaptive to rotation, thus resulting in consistent output (see Figure 1 (b)). Once the kernel points are determined, the point features will be softly associated with each kernel point according to

their probability in each Gaussian component. Then a discrete convolution will be conducted on the kernel points with shared kernel weights to extract local patterns of the point cloud. Moreover, the mixture coefficients of the GMM can be viewed as an attention to the kernel points, by which valid kernel points can also be selected adaptively.

The contributions of this paper can be concluded as: 1) We propose the AGMMConv for discrete convolution in point cloud domain, in which the discrete kernel points are represented with an adaptive GMM therefore being adaptive to local geometric structures. 2) The AGMMConv is demonstrated intrinsically invariant to permutation and translation. Besides, potential rotation invariance can be induced from the probability representation, which makes it robust to rigid transformation. 3) Based on the AGMMConv, an encoder and an encoder-decoder networks are devised for object-level and scene-level point cloud learning tasks, respectively. Various experiments demonstrate the effectiveness and scalability of the proposed method.

2 Related Work

Deep learning methods for point clouds have been widely explored in the literature. For the irregularity of point cloud data, researchers have attempted to transform a point cloud to a structured representation then learn it with classical CNN architectures, in view of their success in processing grid-like data, such as images and natural languages. For example, Su *et al.* [21] use efficient multiview representation for 3D objects. Maturana *et al.* [22] propose a volumetric occupancy grid representation for discrete point clouds. SPLATNet [23] represents the point cloud data as a sparse set of samples in a high-dimensional lattice space. However, the structure information of the point cloud will be inevitably lost during such conversions.

To overcome the limitations of structured representation methods, Qi *et al.* [16, 17] propose to directly process the points with MLP-based network architectures for the first time. Since Qi *et al.* pioneered this field, a number of point-based methods emerge out in the community. RandLA-Net [8] implements efficient large-scale point cloud learning by employing random sampling to reduce the cloud resolution and introducing a novel local feature aggregation module to preserve geometric details. Qiu *et al.* [18] augment the local context by fully utilizing both geometric and semantic features in a bilateral structure. With the advent of graph neural networks (GNNs) in recent years, graph-based methods are also extended to process point clouds. Wang *et al.* [24] propose EdgeConv in which the graph is dynamically computed in each layer of the network for high-level point cloud tasks. Considering the success of the Transformer [25] in language processing and computer vision, transformer/attention-based methods have also been applied in point cloud learning. The PCT (Point cloud Transformer) [9] proposed by Gao *et al.* achieves state-of-the-art performance on object-level point cloud tasks.

Some previous methods also attempt to define convolution operators in point cloud domain. [5] and [30] define continuous convolutions for point clouds from the view of Monte Carlo. However, these methods compute the convolution weights in an attentive way. FPCConv [27] performs a local flattening by learning a weight map to project surrounding points onto a 2D grid, on which regular 2D convolutions can thus be applied. PACConv [31] constructs the convolution kernel by dynamically assembling basic weight matrices, where the coefficients are self-adaptively learned from point positions. Thomas *et al.* [28] propose the kernel point convolution (KPCConv) by borrowing concepts from image convolutions,

where the kernel points are defined as the vertexes of a regular polygon. However, fixed kernel representation makes the convolution sensitive to local geometric structures. Inspired by [9], the authors further extend a deformable version dubbed deformable KPConv. Although the deformable KPConv can induce flexible kernel points, the locations of the kernel points are still unpredictable in this way. Nevertheless, KPConv has shown its ability in point cloud learning, which enlightens researchers on devising discrete convolutions in point cloud domain.

3 Methodology

3.1 Discrete Convolution on Point Cloud

The success of CNNs is mainly attributed to the employment of discrete convolution filters. On the one hand, the spatial locality of these convolution filters introduces translation invariance to the network. The intrinsic invariance of the network can reduce the scale of the hypothesis space of the model, thus facilitating the learning process. In addition, better generalization performance can be achieved by introducing some known priors of invariance. On the other hand, the discrete convolution filter sliding on the image grid can be regarded as a template to match local patterns of the image, which allows the network to extract rich features with various learnable filters. Analogous to image convolution, one can define spatially localized discrete convolution in point cloud domain as:

$$f \star g(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} \langle f(\mathbf{x}_i), g(\mathbf{x}_i - \mathbf{x}) \rangle, \quad (1)$$

where $\mathbf{x} \in \mathbb{R}^3$ denotes the coordinate of a point in 3D Euclidean space. $f(\mathbf{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}^d$ represents the feature of each point. $\mathcal{N}(\mathbf{x})$ is the set of local neighbors of \mathbf{x} in the point cloud. The key point of Eq. (1) is the form of the convolution kernel function g . An alternative choice is to define g as a continuous function with respect to the local coordinate $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{x}$, which results in continuous convolution methods. For example, $g := \text{MLP}(\tilde{\mathbf{x}}_i)$ is adopted in [9, 30]. A discrete version can be obtained by defining a series of discrete kernel points $\{\mathbf{y}_k \in \mathbb{R}^3 | k \leq K\}$ with the associated learnable kernel weights $\{\mathbf{w}_k \in \mathbb{R}^d | k \leq K\}$, where K is the number of kernel points. Subsequently, the convolution kernel g can be defined as:

$$g(\tilde{\mathbf{x}}_i) = \sum_{k=1}^K h(\mathbf{y}_k, \tilde{\mathbf{x}}_i) \mathbf{w}_k, \quad (2)$$

where h is a correlation function that associates each kernel point with corresponding point features $f(\mathbf{x}_i)$. Knowing this, Eq.(1) can be rewritten as:

$$f \star g(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} \langle f(\mathbf{x}_i), \sum_{k=1}^K h(\mathbf{y}_k, \tilde{\mathbf{x}}_i) \mathbf{w}_k \rangle = \sum_{k=1}^K \langle \mathbf{w}_k, \sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} h(\mathbf{y}_k, \tilde{\mathbf{x}}_i) f(\mathbf{x}_i) \rangle. \quad (3)$$

3.2 Mixture Density Network for Kernel Representation

The key to discrete convolutions for point cloud is the representation of kernel points. A fixed kernel representation impedes its generality to different geometric structures. Exploring a flexible and universal kernel representation is necessary for point cloud learning with discrete

convolutions. To address this problem, we propose to represent the discrete kernel points with a GMM. In our representation, the locations of the kernel points can be represented with the mean vectors of the GMM. And the covariance matrices determine the shape of each kernel point, which controls the influence range of each kernel point. To be specific, we use an anisotropic Gaussian to represent each kernel point. In this case, the kernel points can be defined as $\{\mathbf{y}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) | k \leq K\}$, where $\boldsymbol{\mu}_k \in \mathbb{R}^3$, $\boldsymbol{\Sigma}_k \in \mathbb{R}^{3 \times 3}$ are the coordinate and shape of each kernel point, respectively. Because we have the fact that a point cloud can be viewed as a discrete sampling on a continuous 3D surface, the GMM is also a probability distribution of the local geometric surface, which can be estimated from the observations of a local geometric surface. The mixture distribution of a local geometric surface can be denoted as a mixture model:

$$p(\tilde{\mathbf{x}}|\Theta) = \sum_{k=1}^K \pi_k \mathcal{N}_k(\tilde{\mathbf{x}}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad s.t. \quad \sum_{k=1}^K \pi_k = 1, \quad (4)$$

where $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k | k \leq K\}$ is the parameter set and π_k is the mixture coefficient of each Gaussian component. \mathcal{N}_k is a 3D anisotropic normal distribution:

$$\mathcal{N}_k(\tilde{\mathbf{x}}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(\sqrt{2\pi})^3 (\boldsymbol{\sigma}_k^{(1)} \boldsymbol{\sigma}_k^{(2)} \boldsymbol{\sigma}_k^{(3)})} \exp\left(-\frac{1}{2}(\tilde{\mathbf{x}} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\tilde{\mathbf{x}} - \boldsymbol{\mu}_k)\right), \quad (5)$$

where we let $\boldsymbol{\Sigma}_k = \text{diag}((\boldsymbol{\sigma}_k^{(1)})^2, (\boldsymbol{\sigma}_k^{(2)})^2, (\boldsymbol{\sigma}_k^{(3)})^2)$ to reduce complexity.

The commonly used method to estimate Θ is the EM (Expectation Maximization) algorithm, which alternately solves the latent variables and the parameters in an iterative process. However, the EM estimation in the network may cause unstable training, which will be further discussed in detail in the supplementary material. Instead, we propose to use a *mixture density network* to estimate the GMM in this paper. So that the parameters of the GMM can be easily optimized by the gradient descent algorithm in the back propagation of the network. Specifically, we employ a mini PointNet-like network, denoted as f_{GMM} , as the *mixture density network*. For a local neighbor set $\mathcal{N}(\mathbf{x})$, f_{GMM} is a set function $f_{\text{GMM}}: \mathcal{N}(\mathbf{x}) \rightarrow \Theta$ that consists of two point-wise MLPs, an aggregation function \mathcal{A} , and a softmax layer:

$$f_{\text{GMM}}(\{\tilde{\mathbf{x}}_i | \mathbf{x}_i \in \mathcal{N}(\mathbf{x})\}) := \begin{cases} \mathcal{F} = \{\mathbf{f}_i = \text{MLP}_1(\tilde{\mathbf{x}}_i; \boldsymbol{\omega}_1) | \mathbf{x}_i \in \mathcal{N}(\mathbf{x})\} \\ \mathbf{f}_g = \mathcal{A}(\mathcal{F} = \{\mathbf{f}_i | \mathbf{x}_i \in \mathcal{N}(\mathbf{x})\}) \\ \{\hat{\boldsymbol{\mu}}_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k | k \leq K\} = \text{MLP}_g(\mathbf{f}_g; \boldsymbol{\omega}_g) \\ \{\pi_k | k \leq K\} = \text{Softmax}(\{\hat{\boldsymbol{\mu}}_k | k \leq K\}) \end{cases}, \quad (6)$$

where $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_g$ are the parameters of the MLPs. The mixture coefficient π_k is restricted to $\sum_{k=1}^K \pi_k = 1$, which can be implemented with a softmax layer in the network. By the way, we correct the variance $\boldsymbol{\sigma}_k^{(i)} = \sqrt{(\boldsymbol{\sigma}_k^{(i)})^2 + \varepsilon}$ to ensure positive-definite.

The *mixture density network* can be optimized by minimizing the negative log likelihood of the observations $\mathcal{N}(\mathbf{x})$. The likelihood loss for each point is computed as:

$$\mathcal{L}_{\text{likelihood}}(\boldsymbol{\omega}_1, \boldsymbol{\omega}_g) = - \sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} \ln \left\{ \sum_{k=1}^K \pi_k(\boldsymbol{\omega}_1, \boldsymbol{\omega}_g) \mathcal{N}_k(\tilde{\mathbf{x}}_i; \boldsymbol{\mu}_k(\boldsymbol{\omega}_1, \boldsymbol{\omega}_g), \boldsymbol{\Sigma}_k(\boldsymbol{\omega}_1, \boldsymbol{\omega}_g)) \right\}. \quad (7)$$

In training, the likelihood loss is regarded as a regularizer to the network. Consequently, the total loss can be denoted as the linear combination of the task loss and the *averaged*

likelihood loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda \frac{1}{L} \sum_{l=1}^L \frac{1}{N_l} \sum_{n=1}^{N_l} \mathcal{L}_{\text{likelihood}}^{(n)}, \quad (8)$$

where λ is a weight factor, L is the number of AGMMConv layers in the network, and N_l denotes the number of points in the l_{th} layer. The likelihood loss ensures a probability distribution representation for the local geometric surface. The learned kernel points can thus be adaptive to local geometric structures.

3.3 Adaptive GMM Convolution

With the learned GMM representation for the discrete kernel points, we can define the discrete convolution based on the GMM kernel points, which we dubbed *AGMMConv* (Adaptive GMM Convolution). The AGMMConv can be described as following three steps according to Eq. (3). Figure 2 shows the pipeline of the proposed AGMMConv overall.

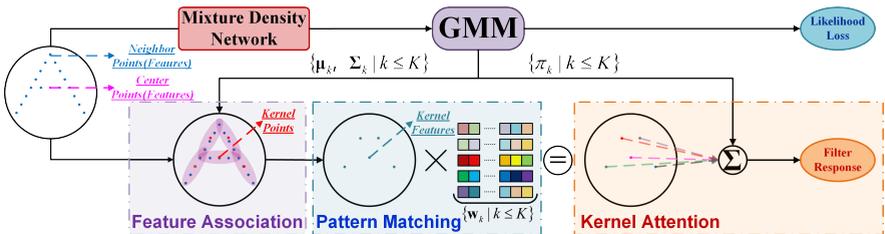


Figure 2: The diagrammatic sketch of the proposed AGMMConv.

(1) Feature Association. The point features in $\mathcal{N}(\mathbf{x})$ are first associated with each kernel point for the subsequent convolution via the correlation function h . In this paper, we intuitively define the correlation function h as the unnormalized density in each Gaussian:

$$h(\mathbf{y}_k, \tilde{\mathbf{x}}_i) = h(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \tilde{\mathbf{x}}_i) = \exp\left(-\frac{1}{2}(\tilde{\mathbf{x}}_i - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\tilde{\mathbf{x}}_i - \boldsymbol{\mu}_k)\right). \quad (9)$$

On the one hand, the Mahalanobis distance in h is more robust to noises and outliers. On the other hand, h can be obtained from the off-the-shelf *mixture density network*, which saves lots of additional computations. The point feature $f(\mathbf{x}_i)$ is then associated with each kernel point \mathbf{y}_k with the correlation function h to generate the kernel feature:

$$f(\mathbf{y}_k) = \sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} h(\mathbf{y}_k, \tilde{\mathbf{x}}_i) f(\mathbf{x}_i). \quad (10)$$

The feature association can be viewed as an interpolation process, where the interpolation weights are computed by the unnormalized density of the point in each Gaussian component of the GMM. The Mahalanobis distance in h encodes both the length/scale and angle/orientation information of neighbor points, thus can effectively capture the local geometric structure. To emphasize this information in convolution, we concatenate each point feature with an additional "1" (ones feature) before interpolation: $f(\tilde{\mathbf{x}}_i) = f(\mathbf{x}_i) \oplus 1$, where \oplus is a concatenating operation. Accordingly, each kernel point can collect the local geometric information of neighbor points with the help of the "ones feature" during interpolation.

(2) Pattern Matching. The pattern matching step is the same as the standard convolution operator in regular domain. The kernel features $\{f(\mathbf{y}_k) \in \mathbb{R}^d | k \leq K\}$ are transformed by a series of shared kernel weights $\{\mathbf{w}_k \in \mathbb{R}^d | k \leq K\}$. It can be understood as a kind of template to match the local patterns of the signal. This process can be denoted as a weighted summation operation:

$$f'(\mathbf{x}) = \sum_{k=1}^K \langle \mathbf{w}_k, f(\mathbf{y}_k) \rangle. \quad (11)$$

A vector version of $f'(\mathbf{x})$ can be obtained by multiple kernel weights (templates).

(3) Kernel Attention. An alternative strategy is to aggregate the kernel features solely with the mixture coefficients $\{\pi_k | k \leq K\}$ without using the kernel weights: $f'(\mathbf{x}) = \sum_{k=1}^K \pi_k f(\mathbf{y}_k)$. However, this simple aggregation strategy will limit the representation ability of the convolution, which will be further discussed in the experiments. Interestingly, we propose that the mixture coefficients can be used to re-weight the kernel points in an attentive way. Consequently, our discrete convolution can be rewritten as:

$$f'(\mathbf{x}) = \sum_{k=1}^K \pi_k \langle \mathbf{w}_k, f(\mathbf{y}_k) \rangle, \quad (12)$$

where π_k can be viewed as an attention to each kernel point. It is feasible to learn the number of kernel points in convolution in this way. By setting a maximum number of kernel points, the attention mechanism allows the convolution to attend to some important kernel points while ignoring the others, which makes our AGMMConv more flexible and scalable.

3.4 Analysis

The AGMMConv consists of two parts, that is, the *mixture density network* and the *discrete convolution*. For the one, the *mixture density network* is obviously permutation invariant. For the other, the convolution operator only depends on the distance between the neighbor points and kernel points, so it is permutation invariant, too. The composition of the two parts remains permutation invariant. Besides, because of the spatial locality of the AGMMConv, it is intrinsically invariant to translation as image convolution. Denote $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ as a rotation matrix in 3D Euclidean space. When a point cloud $\mathcal{P} = \{\mathbf{x}_i | i \leq N\}$ is rotated by \mathbf{R} , we have the rotated cloud $\mathcal{P}' = \{\mathbf{x}'_i = \mathbf{R}\mathbf{x}_i | i \leq N\}$. We start from a special case when $K = 1$. Let $\mathcal{N}(\mathbf{x}; \mu, \Sigma)$ and $\mathcal{N}(\mathbf{x}'; \mu', \Sigma')$ be the distribution of \mathcal{P} and \mathcal{P}' , respectively. Since $\mathcal{P}' = \mathbf{R}\mathcal{P}$, we have a conclusion that $\mu' = \mathbf{R}\mu$ and $\Sigma' = \mathbf{R}\Sigma\mathbf{R}^\top$. The derivation is omitted here for layout reasons, please refer to the supplementary material for details. Knowing this, it is easy to find that $\mathcal{N}(\mathbf{x}'; \mu', \Sigma') = \mathcal{N}(\mathbf{x}; \mu, \Sigma)$. Therefore, the correlation function h defined in Eq. (9) is rotation invariant, which induces rotation invariant convolution. Consider the GMM version when $K > 1$. There is no closed-form solution for a GMM because of the existence of the latent variables. Since the solution of a single Gaussian is rotation equivariant, it is easy to verify that the solution of a GMM is also rotation equivariant. Specifically, let $\Theta = \{\pi_k, \mu_k, \Sigma_k | k \leq K\}$ be a solution of the GMM $p(\mathbf{x})$ before rotation. The solution of the GMM $p(\mathbf{R}\mathbf{x})$, after rotating by \mathbf{R} , becomes $\Theta' = \{\pi_k, \mathbf{R}\mu_k, \mathbf{R}\Sigma_k\mathbf{R}^\top | k \leq K\}$. In other words, we can get the GMM of a point cloud after a rotation by rotating each Gaussian component respectively. Meanwhile, the mixture coefficients $\{\pi_k | k \leq K\}$ remain the same. In practice, the likelihood function of the GMM may be multimodal, thus we cannot guarantee that it will converge to the global minimum in optimizing. Nevertheless, we can say it is potentially invariant to rotation at least, which will be further validated in the experiments.

In this paper, we stand with KPConv and define a novel discrete convolution for point cloud learning. Though both KPConv and our AGMMConv belong to discrete convolution, they are essentially different in model construction. First and foremost, we learn the kernel points with a GMM, which makes them adaptive to local geometric and robust to rigid transformation. Second, KPConv simply associates the point features with linear interpolation. Instead, we model the correlation function with a Gaussian function, which can well capture the local geometric structure. Moreover, in network building, KPConv requires employing grid samplings to determine some hyperparameters, while our adaptive kernel representation is flexible and can avoid such limitations.

4 Experiments

4.1 Experimental Settings

The proposed method is evaluated on three public datasets, that is, ModelNet for object-level tasks, S3DIS and Semantic3D for scene-level tasks. We devise an encoder and an encoder-decoder architectures for classification and segmentation tasks, respectively. The encoder is stacked with five convolution blocks, each of which consists of two successive AGMMConv layers. The decoder contains five corresponding transposed AGMMConv layers to decoding the point features. The number of kernels in each AGMMConv is set to $K = 8$ as default. The weight factor of the likelihood loss is set to $\lambda = 1e - 4$. Codes are implemented with *PyTorch* [19] and experiments are conducted on a single Titan RTX GPU (24GB). More implementation details can be found in the supplementary material.

4.2 Evaluation

4.2.1 Object-Level Tasks

Classification. We evaluate the classification performance of the proposed method on the ModelNet dataset. To reduce the complexity, we uniformly sample 1K points from each object mesh to feed the network in both training and test. The FPS (farthest point sampling) method is employed to reduce the cloud resolution with a sampling ratio of 0.5 after each convolution block. The experiment results on the 10-class and 40-class subsets are shown in Table 1 in terms of OA (Overall Accuracy).

Normal Estimation. The proposed method is further evaluated on ModelNet 40 for normal estimation. Normal estimation is a more challenging task compared to classification, which requires understanding local geometric structures of a point cloud. The experimental strategy follows [19]. We employ the encoder-decoder network for this task. The absolute value of cosine distance is taken as the loss and metric to evaluate the performance. The average cosine distance on the test set is given in Table 1.

4.2.2 Scene-Level Tasks

Indoor Segmentation. We evaluate the segmentation performance of indoor scenes on the S3DIS dataset. In data preparation, each area is first split by rooms, then each room is initially sampled with a grid size of 0.04m. If the number of points in a room is larger than 40k, we randomly select a center point and sample 40k nearest points in this room for input. Otherwise, all the points in the room are fed to the network. This strategy avoids

replacement sampling and ensures a uniform cloud resolution for each room. In test, each room is evaluated multiple times to ensure the reduced cloud is tested completely. Finally, per-point labels of the original point cloud are obtained by nearest projection. We report the cross-validation mIoU (mean Intersection over Union) on all six areas in Table 2.

Outdoor Segmentation. To prove the scalability and robustness of the proposed method, we further conduct experiments on the Semantic3D dataset which is an online benchmark for outdoor scenes segmentation. The training set contains 15 scenes and the (reduced) test set consists of 4 scenes. In training, we divide 2 scenes from the training set as a validation set. The data is prepared similarly to indoor scenes. Each scene is first sampled with a grid size of 0.06m. We use the same sampling strategy to sample 60K points from a scene to feed the network. The test strategy is the same as that for the S3DIS dataset. We evaluate the performance on the reduced-8 test set and submit the results to the benchmark website¹. The comparison results (mIoU) with some current popular methods are shown in Table 2.

Table 1: Experimental results on the Table 2: Experimental results on the S3DIS and Semantic3D datasets.

Methods	10-class	40-class	normal	Methods	S3DIS	Semantic3D
PointNet[14]	-	89.2	0.47	PoinNet[14]	47.6	-
PointNet++[15]	-	91.9	0.29	PointCNN[16]	65.4	-
ECC[17]	90.8	87.4	-	SnapNet[18]	-	59.1
SO-Net[19]	94.1	90.9	-	SEGCloud[20]	-	61.3
Kd-Network[21]	93.3	90.6	-	RF_MSSF[22]	-	62.7
SpecGCN[23]	-	91.5	-	MSDVN[24]	-	65.3
SpiderCNN[25]	-	92.4	-	SPGraph[26]	62.1	73.2
PCNN[27]	94.9	92.3	0.19	DGCNN[28]	56.1	-
PointCNN[29]	-	92.2	-	DeepGCN[30]	60.0	-
PointWeb[31]	-	92.3	-	ShellNet[32]	66.8	69.3
MC[33]	-	90.9	0.16	GACNet[34]	-	70.8
PointConv[35]	-	92.5	-	PointWeb[36]	66.7	-
DGCNN[37]	-	92.9	-	SegGCN[38]	68.5	-
RS-CNN[39]	-	93.6	0.15	PointASNL[40]	68.7	-
FPCConv[41]	-	92.5	-	RandLA-Net[42]	70.0	77.4
KPCConv[43]	-	92.9	-	FPCConv[44]	68.7	-
PACConv[45]	-	93.9	-	KPCConv[46]	70.6	74.6
PCT[47]	-	93.2	0.13	PACConv[48]	69.3	-
PointASNL[49]	95.9	93.2	-	BCM+AFM[50]	72.2	75.3
Ours	94.4	92.5	0.10	Ours	72.3	76.1

4.2.3 Discussion

From Table 1, our method can obtain competitive results on classification tasks and achieve best performance on normal estimation. In Table 2, the proposed method achieves state-of-the-art performance on the S3DIS dataset with the mIoU of 72.3%. Furthermore, our method significantly surpasses KPConv in both indoor and outdoor scene segmentation, which proves the effectiveness of our adaptive kernel representations. Besides, compared to the simple classification tasks, we find that the proposed method has great advantages in more complex tasks, such as normal estimation and scene segmentation. It can be explained that our AGMMConv facilitates the network learning by introducing various priors of invariance. This superiority would manifest significantly in more complex tasks. Moreover, it can be seen that the proposed method performs better in indoor scenes. Because the distribution of point clouds in outdoor scenes is usually more biased than that in indoor scenes, this may cause the learning of the GMM unstable. We will try to address this problem in future work. More experimental results and visualizations can be found in the supplementary material.

¹http://www.semantic3d.net/view_results.php?chl=2

4.3 Ablation Study

Kernel Numbers. To evaluate the influence of the number of kernel points on the algorithm, we make a comparison of the performance with various number of kernels on the ModelNet 40 dataset. The training and test OA can be found in Table 3. Owing to the kernel attention mechanism, the performance does not significantly degenerate with the kernel numbers increasing. The model performs better when the number of kernels is 8 or 16. We choose $K = 8$ in our experiments considering the balance of performance and computations.

Robustness Test. In order to further prove the robustness of the AGMMConv, we test the 3D objects recognition performance on the ModelNet 40 dataset under different test augmentations including random permutation, translation, and rotation. We compare the performance of fixed kernels (polygon) and the proposed GMM kernels. The results are shown in Table 4. It can be seen that the proposed method is robust to rigid transformation.

Kernel Weights. As we stated in the previous section, the kernel weights that act as templates are important for the AGMMConv. We compare the performance with/without the kernel weights on classification (ModelNet 40) and segmentation (S3DIS (Area 5)) tasks. The results are given in Table 5. The kernel weights increase the representation ability of the network especially for more complex scene segmentation task.

Runtime Analysis. We analyze the runtime of the proposed method on the classification and segmentation tasks. The running time including data preparation in each training and test iteration is counted in Table 6. From the table, the proposed method is very efficient on object-level classification tasks. Even for scene-level segmentation tasks with far larger input size, the proposed method can also achieve promising time complexity.

Table 3: Performance with different num- Table 4: Test performance with different augmentation on ModelNet 40.

#num_kernels	Training OA	Test OA
1	96.24	90.88
2	96.32	91.25
4	96.59	91.25
8	97.98	92.50
16	98.03	92.26
32	97.83	91.53

Test Augmentations	Kernel Types	
	Polygon	GMM
None	91.21	92.50
Permutation	91.21	92.50
Translation [-0.2, +0.2]	91.21	92.50
Rotation [-180, +180]	46.96	91.65
Rotation + Translation	42.75	91.82

Table 5: Performance with/without the Table 6: Runtime of the proposed method with different input sizes.

Tasks	Weights	#params	OA/mIoU
cls.	No	4.7MB	91.64
	Yes	6.2MB	92.50
seg.	No	2.9MB	64.25
	Yes	4.5MB	66.83

Tasks	Size (B×N)	Training (ms)	Test (ms)
cls.	16×1K	107	45
	32×1K	172	77
	16×4K	339	181
seg.	8×40K	1250	588
	4×60K	1562	940

5 Conclusion

In this paper, we propose a novel discrete convolution for point cloud learning dubbed AGMMConv. We first propose to represent the kernel points with a GMM. The GMM can efficiently represent the local geometric of the point cloud, thus being adaptive to local geometric structures. Then we define the AGMMConv based on the learned kernel representations. Besides, we further demonstrate the AGMMConv is robust to rigid transformation, which is crucial for 3D objects recognition. Experiments on various datasets including object-level and scene-level tasks show the effectiveness of the proposed method.

Acknowledgment

This work is partially supported by *National Natural Science Foundation of China* under Grant Nos. 61872188, 61703209, U1713208, 61972204, 61672287, 61861136011, and 61773215.

References

- [1] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics*, 37(4):1–12, 2018.
- [2] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. In *Proceedings of the Eurographics Workshop on 3D Object Retrieval*, 2017.
- [3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, et al. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.
- [4] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, et al. Pct: Point cloud transformer. *arXiv preprint arXiv:2012.09688*, 2020.
- [5] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, et al. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics*, 37(6):1–12, 2018.
- [6] Qingyong Hu, Bo Yang, Linhai Xie, et al. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.
- [7] Roman Klokov and Victor Lempitsky. Escape from cells: deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017.
- [8] Huan Lei, Naveed Akhtar, and Ajmal Mian. Seggcn: Efficient 3d point cloud segmentation with fuzzy spherical kernel. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11611–11620, 2020.
- [9] Guohao Li, Matthias Muller, Ali Thabet, et al. Deepgcns: can gcns go as deep as cnns? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9267–9276, 2019.
- [10] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: self-organizing network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9397–9406, 2018.
- [11] Yangyan Li, Rui Bu, Mingchao Sun, et al. Pointcnn: convolution on χ -transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018.
- [12] Yiqun Lin, Zizheng Yan, Haibin Huang, et al. Fpconv: Learning local flattening for point convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2020.

- [13] Yongcheng Liu, Bin Fan, Shiming Xiang, et al. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.
- [14] Daniel Maturana and Sebastian Scherer. Voxnet: a 3d convolutional neural network for real-time object recognition. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 922–928, 2015.
- [15] Adam Paszke, Sam Gross, Francisco Massa, et al. Pytorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019.
- [16] Charles R Qi, Hao Su, Kaichun Mo, et al. Pointnet: deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.
- [17] Charles R Qi, Li Yi, Hao Su, et al. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.
- [18] Shi Qiu, Saeed Anwar, and Nick Barnes. Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1757–1767, 2021.
- [19] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. Classification of point cloud for road scene understanding with multiscale voxel deep network. In *Proceedings of the 10th Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, 2018.
- [20] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3693–3702, 2017.
- [21] Hang Su, Subhransu Maji, Evangelos Kalogerakis, et al. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 945–953, 2015.
- [22] Hang Su, Varun Jampani, Deqing Sun, et al. Splatnet: sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.
- [23] Lyne Tchapmi, Christopher Choy, Iro Armeni, et al. Segcloud: semantic segmentation of 3d point clouds. In *Proceedings of the 2017 International Conference on 3D vision*, pages 537–547, 2017.
- [24] Hugues Thomas, François Goulette, Jean-Emmanuel Deschaud, et al. Semantic classification of 3d point clouds with multiscale spherical neighborhoods. In *Proceedings of the 2018 International Conference on 3D Vision*, pages 390–398, 2018.
- [25] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, et al. Kpconv: flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019.

- [26] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.
- [27] Lei Wang, Yuchun Huang, Yaolin Hou, et al. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10296–10305, 2019.
- [28] Weiyue Wang, Ronald Yu, Qiangui Huang, et al. Sgpn: similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018.
- [29] Yue Wang, Yongbin Sun, Ziwei Liu, et al. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5):1–12, 2019.
- [30] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.
- [31] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3173–3182, 2021.
- [32] Yifan Xu, Tianqi Fan, Mingye Xu, et al. Spidercnn: deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision*, pages 87–102, 2018.
- [33] Xu Yan, Chaoda Zheng, Zhen Li, et al. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5589–5598, 2020.
- [34] Li Yi, Hao Su, Xingwen Guo, et al. Syncspecnn: synchronized spectral cnn for 3d shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2282–2290, 2017.
- [35] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1607–1616, 2019.
- [36] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, et al. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019.