# Supplementary Material of 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 Engneering
Nanjing University of Science and Technology
Nanjing 210094, China

## 1 Rotation Invariance of Gaussian

Denote $\mathbf{R} \in \mathbb{R}^{3\times3}$ 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 point cloud $\mathcal{P}' = \{\mathbf{x}'_i = \mathbf{R}\mathbf{x} | i \leq N\}$. The mean vector and covariance matrix of the Gaussian distribution can be estimated from the first and second order moments:

$$\begin{cases} \mu = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_i = \bar{\mathbf{x}} \\ \Sigma = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top \end{cases}. \tag{1}$$

We can obtain the mean vector $\mu'$ and the covariance matrix $\Sigma'$ of the rotated Gaussian by substituting $\mathbf{x}_i$ with $\mathbf{x}'_i = \mathbf{R}\mathbf{x}_i$ in Eq. (1):

$$\begin{cases} \mu' = \mathbf{R}\mu \\ \Sigma' = \mathbf{R}\Sigma\mathbf{R}^\top \end{cases}. \tag{2}$$

We have the fact that the first and second order momentums of Gaussian distribution are rotation equivariant. Knowing this, it is easy to find that:

$$\begin{aligned} \mathcal{N}(\mathbf{x}';\mu',\Sigma') &= \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma'|^{\frac{1}{2}}}\exp\big(-\frac{1}{2}(\mathbf{x}'-\mu')^\top\Sigma'^{-1}(\mathbf{x}'-\mu')\big) \\ &= \frac{1}{(2\pi)^{\frac{d}{2}}|\mathbf{R}\Sigma\mathbf{R}^\top|^{\frac{1}{2}}}\exp\big(-\frac{1}{2}(\mathbf{R}\mathbf{x}-\mathbf{R}\mu)^\top(\mathbf{R}\Sigma\mathbf{R}^\top)^{-1}(\mathbf{R}\mathbf{x}-\mathbf{R}\mu)\big) \\ &= \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}}\exp\big(-\frac{1}{2}(\mathbf{x}-\mu)^\top\Sigma^{-1}(\mathbf{x}-\mu)\big) \\ &= \mathcal{N}(\mathbf{x};\mu,\Sigma) \end{aligned}, \tag{3}$$

which indicates the rotation invariance of Gaussian.

## 2 An EM perspective

The solution process of a GMM can be concluded as an EM (Expectation-Maximization) procedure. In the expectation step, the posterior distribution of the latent variables $\gamma_{ik}$ is updated by the guessed parameters:

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}_k(\mathbf{x}_i|\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}_j(\mathbf{x}_i|\mu_j,\Sigma_j)}. \tag{4}$$

Then the parameters of the GMM can be estimated by maximizing the likelihood under the expectation of posterior distributions of latent variables.

$$\begin{cases} \mu_k = \frac{1}{N_k}\sum_{i=1}^{N} \gamma_{ik}\mathbf{x}_i \\ \Sigma_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{ik}(\mathbf{x}_i-\mu_k)(\mathbf{x}_i-\mu_k)^\top \\ \pi_k = \frac{1}{N}N_k \end{cases}, \tag{5}$$

where $N_k = \sum_{i=1}^{N} \gamma_{ik}$.

From the EM procedure, we can find that:

1. In the M-step, when $\gamma_{ik}$ is fixed, the mean vector $\mu_k$ and covariance matrix $\Sigma_k$ are equivariant to rotation according to Eq. (1) and Eq. (2). And the mixture coefficient $\pi_k$ is independent of rotation.

2. In the E-step, $\gamma_{ik}$ remains the same as that before rotation with the estimated parameters $\{\pi_k,\mu_k,\Sigma_k|k\leq K\}$ in the M-step according to Eq. (3).

That is to say, with proper initialization, the GMM is rotation equivariant in the EM procedure. In other words, if $\Theta = \{\pi_k,\mu_k,\Sigma_k|k\leq K\}$ is a solution of a GMM $p(\mathbf{x})$, we can find a solution $\Theta' = \{\pi_k,\mathbf{R}\mu_k,\mathbf{R}\Sigma_k\mathbf{R}^\top|k\leq K\}$ for the GMM $p(\mathbf{Rx})$ after rotating the random variables by $\mathbf{R}$. In practice, the likelihood function of the GMM may be multimodal, thus we cannot guarantee the parameters always convergent to the global minimal across different views (with different initializations) in optimization. Nevertheless, we can say that the GMM is potentially invariant to rotation at least.

Actually, the EM solution also provides us an alternative GMM implementation different with our *mixture density network*. To be specific, we can let the network learn a posterior for each input point, which can be easily implemented with a softmax layer in the network: $\gamma_i = \text{Softmax}(\mathbf{x}_i) \in \mathbb{R}^K$. Then the parameters of the GMM can be estimated according to Eq. (5) in one iteration without additional likelihood loss. And the rotation invariance can be also ensured in this way. In this formulation, the leaning of the GMM turns to the learning of the posteriors of latent variables. However, we find this strategy will cause unstable training. Because there exist two successive exponential operations in this implementation (one in the softmax layer for computing $\gamma_{ik}$ and another in the Gaussian function), which will causes gradients explosion. Besides, this implementation requires higher space complexity to store the posteriors for each neighbor point. We may address this problem in future work. Anyway, the *mixture density network* used in this paper is more efficient and can avoid such limitations.

# 3 Implementation Details

## 3.1 Network Architectures

Inspired by ResNet[4], we use a residual bottleneck structure for the AGMMConv layer. The convolution layer is illustrated in Figure 1. A BN (Batch Normalization) layer with a momentum of 0.98 is added after each convolution layer. And the LeakyReLU layer with a negative slope of 0.1 is used as an activation layer.

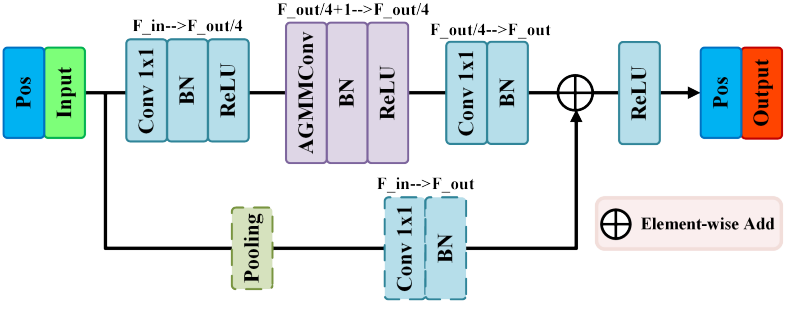


Figure 1: Residual bottleneck AGMMConv layer used in this paper. Note that the pooling layer and convolution layer in the residual path are used to align the cloud resolution and feature dimension if necessary.

Table 1: Details of network architectures used in this paper.

| modules | blocks | layers | |
|---|---|---|---|
| encoder | conv_1 | AGMMConv(in_dims=$F_{\text{in}}$, out_dims=32, num_kernels=8, ratio=1.0)<br>AGMMConv(in_dims=32, out_dims=32, num_kernels=8, ratio=1.0) | |
| | conv_2 | AGMMConv(in_dims=32, out_dims=64, num_kernels=8, ratio=$r_1$)<br>AGMMConv(in_dims=64, out_dims=64, num_kernels=8, ratio=1.0) | |
| | conv_3 | AGMMConv(in_dims=64, out_dims=128, num_kernels=8, ratio=$r_2$)<br>AGMMConv(in_dims=128, out_dims=128, num_kernels=8, ratio=1.0) | |
| | conv_4 | AGMMConv(in_dims=128, out_dims=256, num_kernels=8, ratio=$r_3$)<br>AGMMConv(in_dims=256, out_dims=256, num_kernels=8, ratio=1.0) | |
| | conv_5 | AGMMConv(in_dims=256, out_dims=512, num_kernels=8, ratio=$r_4$)<br>AGMMConv(in_dims=512, out_dims=512, num_kernels=8, ratio=1.0) | |
| decoder | deconv_4 | AGMMConv(in_dims=512, out_dims=256, num_kernels=8, ratio=$1/r_4$) | |
| | deconv_3 | AGMMConv(in_dims=256+256, out_dims=128, num_kernels=8, ratio=$1/r_3$) | |
| | deconv_2 | AGMMConv(in_dims=128+128, out_dims=64, num_kernels=8, ratio=$1/r_2$) | |
| | deconv_1 | AGMMConv(in_dims=64+64, out_dims=32, num_kernels=8, ratio=$1/r_1$) | |
| classifiers | | classification | segmentation |
| | fc_1 | Linear(in_dims=512, out_dims=1024) | Linear(in_dims=512, out_dims=256) |
| | activation | LeakyReLU(negative_slope=0.1) | LeakyReLU(negative_slope=0.1) |
| | dropout | Dropout(p=0.5) | None |
| | fc_2 | Linear(in_dims=1024, out_dims=C) | Linear(in_dims=256, out_dims=C) |
| | normalize | Softmax() | Softmax()/Normalize() |

We devise two network architectures with the proposed AGMMConv for different tasks. The encoder network is used for object-level classification task. And the encoder-decoder network is used for object-level normal estimation and scene segmentation tasks. The encoder is stacked with five convolution blocks, each of which consists of two successive AGMMConv layers. The feature dimensions are increased after each convolution block. They are set to [32, 64, 128, 256, 512] in the five convolution blocks, respectively. Meanwhile, the spatial resolutions of the point cloud are decreased by sampling with a ratio of $r$. Specifically, We employ FPS (Farthest Point Sampling) for the object-level tasks and grid sampling for scene-level tasks. We adopt the same encoder for the two network architectures. The decoding part of the encoder-decoder network contains five corresponding transposed AGMMConv layers to restore the cloud resolution gradually. Two full connected layers are used as the classifier in the network. A drop out layer with the probability of 0.5 is added after the first fully connected layer for classification tasks. For the normal estimation task, we replace the softmax layer with a normalization layer to regress per-point normals. The detailed network architectures are shown in Table 1.

## 3.2 Precomputed Multiscale Strategy

Our GMMConv is flexible and scalability to various sampling and neighbors grouping strategy. In KPConv, the authors employ a grid sampling strategy to ensure a fixed spatial resolution at each layer, which allows they to choose appropriate hyperparameters with respect to the grid size of each layer, such as the radius and influence range. Unlike the KPConv, our GMMConv learn an adaptive representation for the convolution kernel points can thus avoid such limitations. In point cloud learning, the data organization, such as sampling and neighbor grouping, costs lots of computations, which hinders the algorithm from generalizing to large-scale scenes. Inspired by RandLA-Net[5], we use a precomputed multiscale strategy to get a multiscale representation for the point cloud. We use kNN to search the neighbors of each point (we use k=16 in our experiments). The kNN searching can be efficiently computed on CPU using KDTree. To reduce the spatial resolution, we apply different strategies for various tasks. Considering computation efficiency, we employ FPS for object-level tasks and grid sampling for scene-level tasks. The multiscale point clouds and neighbor indexes are stored beforehand in the form of a sparse multiscale batch, which will be viewed as a whole to feed the network. The sparse batch representation also makes the network support various input point cloud size. Owing to the precomputed multiscale strategy, our network can handle a million points in a single pass, which is helpful for learning large-scale outdoor point clouds.

## 3.3 Training Approach

The momentum SGD method is used to minimize the total loss with the momentum of 0.98 and weight decay of 1e-4. The total loss is the linear combination of the task loss and the likelihood loss, in which the likelihood loss is regarded as a regularizer to the network and weighted by a weight factor $\lambda = 1e-4$. We apply a decay learning rate strategy in training. For the object-level tasks, the learning rate is set to 1e-3 and multiplied by 0.1 every 100 epochs. The network converges in 200 epochs. We training 50K iterations for the scene-level tasks, the initial learning rate is set to 1e-2 and multiplied by 0.1 every 25K iterations. Neither pretraining nor additional data is used in training.

# 4 More Experimental Results

## 4.1 Datasets

**ModelNet**[21] is a synthetic dataset for 3D objects classification, which contains 12,311 meshed CAD models from 40 categories. This dataset compiled a list of the most common object categories in the world, using the statistics obtained from the SUN database. The full version contains 40 classes, and 10 popular object categories are further chosen as a 10-class subset. The data is original represented with triangle faces. In the experiment, we first sample the point cloud with the grid size of 0.02 from the original face. Then we uniformly sample a fix number of points (1,024) to feed the network. The coordinate is used as the input feature for each point.

**S3DIS**[1] is a large-scale indoor 3D scenes segmentation dataset, which is collected on six large-scale indoor areas from 3 different buildings. The scene is annotated with 13 categories (including clutter). Following [10], we split the dataset by rooms (271 rooms in total). Each point is represented as a 6D vector including the coordinate in 3D Euclidean space and the corresponding RGB color. In our experiments, the training and test sets are split according to areas. That is, five of the six areas are selected as a training set and the remaining one is used for test.

**Semantic3D**[5] is a large-scale outdoor point cloud segmentation benchmark. The data is acquired by a 3D Velodyne LiDAR. Each point is also colorized in a postprocessing step by deploying a high resolution cubemap, which is generated from the corresponding camera images. This benchmark provides a large labeled 3D point cloud data set of natural scenes with over 4 billion points in total, which covers a range of diverse urban scenes. We conduct our experiments on the reduced-8 benchmark that contains 15 scenes for training and 4 reduced scenes for test with 8 semantic categories. Because the test set has been uniformly downsampled with the resolution of $0.01m$. In the preprocessing, we also downsample the training set to maintain the same resolution as the reduced test set.

## 4.2 Normal Estimation

The groundtruth normals are originally computed from the triangle meshes. We show some visualization results compared with the groundtruth in Figure 3. It can be seen that the predicted normals are more reasonable in direction than the groundtruth in some cases.

## 4.3 Indoor Scenes Segmentation

The 6-fold cross validation results on the six areas of the S3DIS dataset are given in Table 2 in detail. And the 6-fold per-class IoUs compared with some current popular methods are shown in Table 3. Because some previous methods only report the performance on Area 5, we also give a comparison on this area in Table 4. Some visualization results on the S3DIS dataset are illustrated in Figure 4.

## 4.4 Outdoor Scenes Segmentation

The per-class IoUs on the Semantic 3D dataset (reduced-8) compared with some current popular methods are given in Table 5. The proposed method performs comparably with

RandLA-Net and surpasses the other methods. Some visualization results on the validation scenes are also shown in Figure 2.

Table 2: Cross validation results of the proposed method on the S3DIS dataset.

| Areas | **OA** | **mACC** | **mIoU** | ceil. | floor | wall | beam | col. | wind. | door | table | chair | sofa | book. | board | clut. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 89.2 | 86.3 | 75.5 | 96.3 | 95.7 | 78.8 | 61.2 | 61.5 | 82.3 | 83.7 | 69.9 | 81.4 | 73.5 | 64.3 | 64.7 | 68.0 |
| 2 | 85.5 | 72.4 | 60.7 | 90.8 | 96.4 | 82.8 | 27.1 | 60.6 | 75.4 | 71.5 | 62.8 | 42.4 | 66.2 | 50.5 | 15.8 | 46.6 |
| 3 | 92.0 | 90.1 | 80.2 | 95.5 | 98.3 | 83.9 | 64.7 | 34.3 | 87.4 | 88.8 | 73.2 | 86.3 | 89.1 | 77.4 | 86.7 | 76.4 |
| 4 | 88.8 | 82.1 | 68.0 | 95.4 | 97.9 | 82.4 | 43.4 | 70.8 | 47.6 | 70.2 | 62.4 | 81.8 | 60.9 | 63.1 | 43.9 | 64.1 |
| 5 | 89.4 | 74.7 | 66.8 | 93.7 | 97.7 | 83.4 | 0.0 | 35.7 | 61.3 | 66.3 | 80.7 | 88.2 | 67.6 | 74.3 | 61.4 | 58.5 |
| 6 | 92.5 | 92.4 | 81.7 | 96.5 | 97.7 | 85.5 | 81.9 | 76.4 | 82.7 | 86.1 | 78.1 | 86.8 | 71.9 | 74.1 | 72.1 | 72.6 |
| 6-fold | 89.3 | 82.8 | 72.3 | 94.3 | 97.2 | 82.7 | 60.8 | 57.6 | 68.6 | 75.7 | 73.5 | 68.9 | 70.7 | 68.9 | 59.2 | 61.4 |

Table 3: Performance of the proposed method on the S3DIS dataset (6-fold).

| Methods | **OA** | **mACC** | **mIoU** | ceil. | floor | wall | beam | col. | wind. | door | table | chair | sofa | book. | board | clut. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PointNet[10] | 78.6 | 66.2 | 47.6 | 88.0 | 88.7 | 69.3 | 42.4 | 23.1 | 47.5 | 51.6 | 54.1 | 42.0 | 9.6 | 38.2 | 29.4 | 35.2 |
| PointCNN[8] | 88.1 | 75.6 | 65.4 | 94.8 | 97.3 | 75.8 | 63.3 | 51.7 | 58.4 | 57.2 | 71.6 | 69.1 | 39.1 | 61.2 | 52.2 | 58.6 |
| SPGraph[15] | 86.4 | 73.0 | 62.1 | 89.9 | 95.1 | 76.4 | 62.8 | 47.1 | 55.3 | 68.4 | 73.5 | 69.2 | 63.2 | 45.9 | 8.7 | 52.9 |
| DGCNN[20] | 84.1 | - | 56.1 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| DeepGCN[9] | 85.9 | - | 60.0 | 93.1 | 95.3 | 78.2 | 33.9 | 37.4 | 56.1 | 68.2 | 64.9 | 61.0 | 34.6 | 51.5 | 51.1 | 54.4 |
| ShellNet[24] | 87.1 | - | 66.8 | 90.2 | 93.6 | 79.9 | 60.4 | 44.1 | 64.9 | 52.9 | 71.6 | **84.7** | 53.8 | 64.6 | 48.6 | 59.4 |
| PointWeb[25] | 87.3 | 76.2 | 66.7 | 93.5 | 94.2 | 80.8 | 52.4 | 41.3 | 64.9 | 68.1 | 71.4 | 67.1 | 50.3 | 62.7 | 62.2 | 58.5 |
| SegGCN[6] | 87.8 | 77.1 | 68.5 | 92.5 | 97.6 | 78.9 | 44.6 | 58.2 | 53.7 | 67.3 | **74.6** | 83.9 | 68.0 | 65.7 | 46.8 | 58.8 |
| PointASNL[23] | 88.8 | 79.0 | 68.7 | **95.3** | **97.9** | 81.9 | 47.0 | 48.0 | 67.3 | 70.5 | 71.3 | 77.8 | 50.7 | 60.4 | 63.0 | 62.8 |
| RandLA-Net[8] | 88.0 | 82.0 | 70.0 | 93.1 | 96.1 | 80.6 | 62.4 | 48.0 | 64.4 | 69.4 | 69.4 | 76.4 | 60.0 | 64.2 | 65.9 | 60.1 |
| FPConv[9] | - | - | 68.7 | 94.8 | 97.5 | 82.6 | 42.8 | 41.8 | 58.6 | 73.4 | 71.0 | 81.0 | 59.8 | 61.9 | 64.2 | **64.2** |
| KPConv[16] | - | 79.1 | 70.6 | 93.6 | 92.4 | **83.1** | **63.9** | 54.3 | 66.1 | **76.6** | 57.8 | 64.0 | 69.3 | **74.9** | 61.3 | 60.3 |
| PAConv[22] | - | 78.7 | 69.3 | 94.3 | 93.5 | 82.8 | 56.9 | 45.7 | 65.2 | 74.9 | **74.6** | 59.7 | 61.8 | 67.4 | 65.8 | 58.4 |
| BCM+AFM[12] | 88.9 | **83.1** | 72.2 | 93.3 | 96.8 | 81.6 | 61.9 | 49.5 | 65.4 | 73.3 | 72.0 | 83.7 | 67.5 | 64.3 | **67.0** | 62.4 |
| Ours | **89.3** | 82.8 | **72.3** | 94.3 | 97.2 | 82.7 | 60.8 | **57.6** | **68.6** | 75.7 | 73.5 | 68.9 | **70.7** | 68.9 | 59.2 | 61.4 |

Table 4: Performance of the proposed method on the S3DIS dataset (Area 5).

| Methods | **OA** | **mACC** | **mIoU** | ceil. | floor | wall | beam | col. | wind. | door | table | chair | sofa | book. | board | clut. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PointNet[10] | - | 49.0 | 41.1 | 88.8 | 97.3 | 69.8 | 0.1 | 3.9 | 46.3 | 10.8 | 52.6 | 58.9 | 40.3 | 5.9 | 26.4 | 33.2 |
| PointNet++[11] | - | 63.5 | 57.3 | 91.3 | 96.9 | 78.7 | 0.0 | 16.0 | 54.9 | 31.2 | 74.6 | 83.5 | 49.3 | 67.2 | 54.2 | 45.9 |
| PointCNN[8] | 85.9 | 63.9 | 57.3 | 92.3 | 98.2 | 79.4 | 0.0 | 17.6 | 22.8 | 62.1 | 74.4 | 80.6 | 31.7 | 66.7 | 62.1 | 56.7 |
| SPGraph[15] | 86.4 | 66.5 | 58.0 | 89.4 | 96.9 | 78.1 | 0.0 | **42.8** | 48.9 | 61.6 | **84.7** | 75.4 | 69.8 | 52.6 | 2.1 | 52.2 |
| SegCloud[14] | - | 57.4 | 48.9 | 90.1 | 96.1 | 69.9 | 0.0 | 18.4 | 38.4 | 23.1 | 70.4 | 75.9 | 40.9 | 58.4 | 13.0 | 41.6 |
| GACNet[17] | 87.8 | - | 62.9 | 92.3 | 98.3 | 81.9 | 0.0 | 20.4 | 59.1 | 40.9 | 78.5 | 85.8 | 61.7 | 70.8 | 74.7 | 52.8 |
| ParamConv[18] | - | 67.0 | 58.3 | 92.3 | 96.2 | 75.9 | **0.3** | 6.0 | **69.5** | 63.5 | 65.6 | 66.9 | 68.9 | 47.3 | 59.1 | 46.2 |
| PointWeb[25] | 87.0 | 66.6 | 60.3 | 92.0 | 98.5 | 79.4 | 0.0 | 21.1 | 59.7 | 34.8 | 76.3 | 88.3 | 46.9 | 69.3 | 64.9 | 52.5 |
| SegGCN[6] | 88.2 | 70.4 | 63.6 | 93.7 | **98.6** | 80.6 | 0.0 | 28.5 | 42.6 | **74.5** | 80.9 | 88.7 | 69.0 | 71.3 | 44.4 | 54.3 |
| PointASNL[23] | 87.7 | 68.5 | 62.6 | 94.3 | 98.4 | 79.1 | 0.0 | 26.7 | 55.2 | 66.2 | 83.3 | 86.8 | 47.6 | 68.3 | 56.4 | 52.1 |
| FPConv[9] | - | - | 62.8 | **94.6** | 98.5 | 80.9 | 0.0 | 19.1 | 60.1 | 48.9 | 80.6 | 88.0 | 53.2 | 68.4 | 68.2 | 54.9 |
| KPConv[16] | - | 72.8 | **67.1** | 92.8 | 97.3 | 82.4 | 0.0 | 23.9 | 58.0 | 69.0 | 81.5 | **91.0** | **75.4** | **75.3** | 66.7 | **58.9** |
| PAConv[22] | - | 72.3 | 65.6 | 93.1 | 98.4 | 82.6 | 0.0 | 22.6 | 61.3 | 63.3 | 78.5 | 88.0 | 64.5 | 73.5 | **70.1** | 57.3 |
| BCM+AFM[12] | 88.9 | 73.1 | 65.4 | 92.9 | 97.9 | 82.3 | 0.0 | 23.1 | 65.5 | 64.9 | 78.5 | 87.5 | 61.4 | 70.7 | 68.7 | 57.2 |
| Ours | **89.4** | **74.7** | 66.8 | 93.7 | 97.7 | **83.4** | 0.0 | 35.7 | 61.3 | 66.3 | 80.7 | 88.2 | 67.6 | 74.3 | 61.4 | 58.5 |

Table 5: Per-class IoU of the proposed method on the Semantic3D dataset (reduced-8).

| Methods | OA | mIoU | man-made terrain | natural terrain | high vegetation | low vegetation | buildings | hard scape | scanning artefacts | cars |
|---|---|---|---|---|---|---|---|---|---|---|
| SnapNet[2] | 88.6 | 59.1 | 82.0 | 77.3 | 79.7 | 22.9 | 91.9 | 18.4 | 37.3 | 64.4 |
| SegCloud[14] | 88.1 | 61.3 | 83.9 | 66.0 | 86.0 | 40.5 | 91.9 | 30.9 | 27.5 | 64.3 |
| RF_MSSF[15] | 90.3 | 62.7 | 87.6 | 80.3 | 81.8 | 36.4 | 92.2 | 24.1 | 42.6 | 56.6 |
| MSDVN[13] | 88.4 | 65.3 | 83.0 | 67.2 | 83.8 | 36.7 | 92.4 | 31.3 | 50.0 | 78.2 |
| SPGraph[19] | 94.0 | 73.2 | 97.4 | 92.6 | 87.9 | 44.0 | 93.2 | 31.0 | 63.5 | 76.2 |
| ShellNet[22] | 93.2 | 69.3 | 96.3 | 90.4 | 83.9 | 41.0 | 94.2 | 34.7 | 43.9 | 70.2 |
| GACNet[12] | 91.9 | 70.8 | 86.4 | 77.7 | **88.5** | **60.6** | 94.2 | 37.3 | 43.5 | 77.8 |
| RandLA-Net[5] | 94.8 | **77.4** | 95.6 | 91.4 | 86.6 | 51.5 | 95.7 | **51.5** | 69.8 | 76.8 |
| KPConv[16] | 92.9 | 74.6 | 90.9 | 82.2 | 84.2 | 47.9 | 94.9 | 40.0 | **77.3** | 79.7 |
| BCM+AFM[17] | 94.3 | 75.3 | 96.3 | 93.7 | 87.7 | 48.1 | 94.6 | 43.8 | 58.2 | 79.5 |
| Ours | **95.0** | 76.1 | **97.7** | **93.9** | 83.9 | 50.0 | **95.8** | 49.8 | 52.9 | **84.8** |

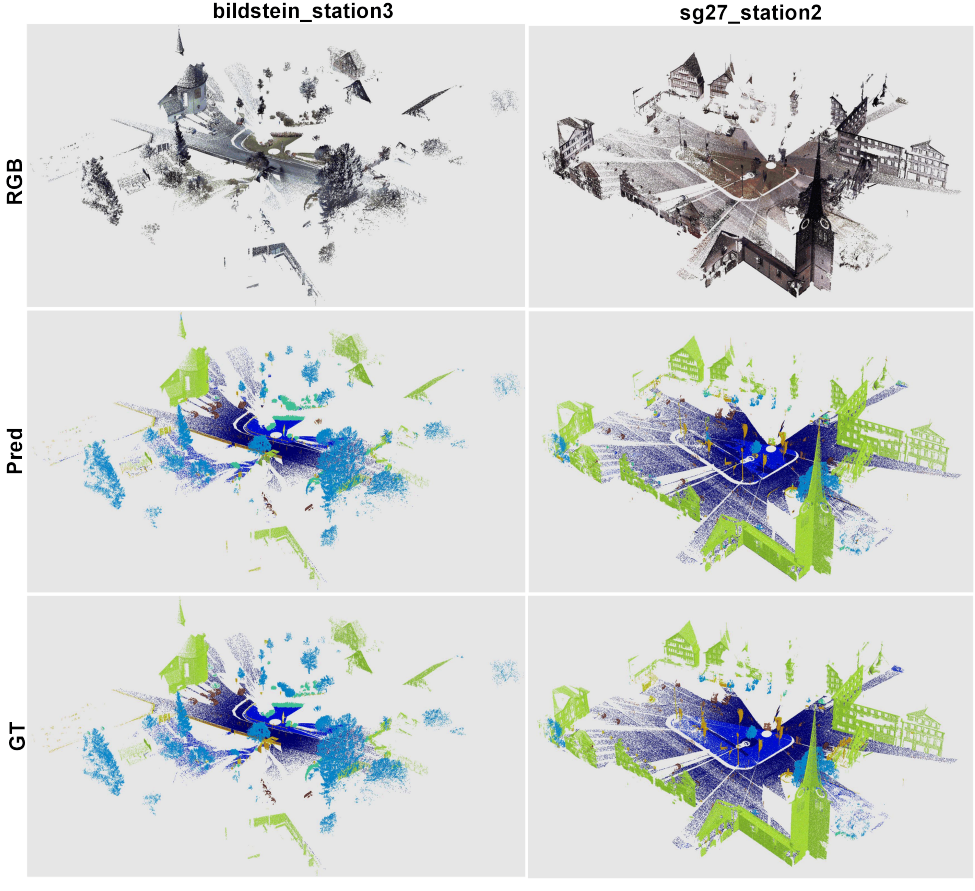


Figure 2: Validation results compared with the groundtruth on the Semantic3D dataset. Note that, the "unkown" class is ignored for better visualizations.

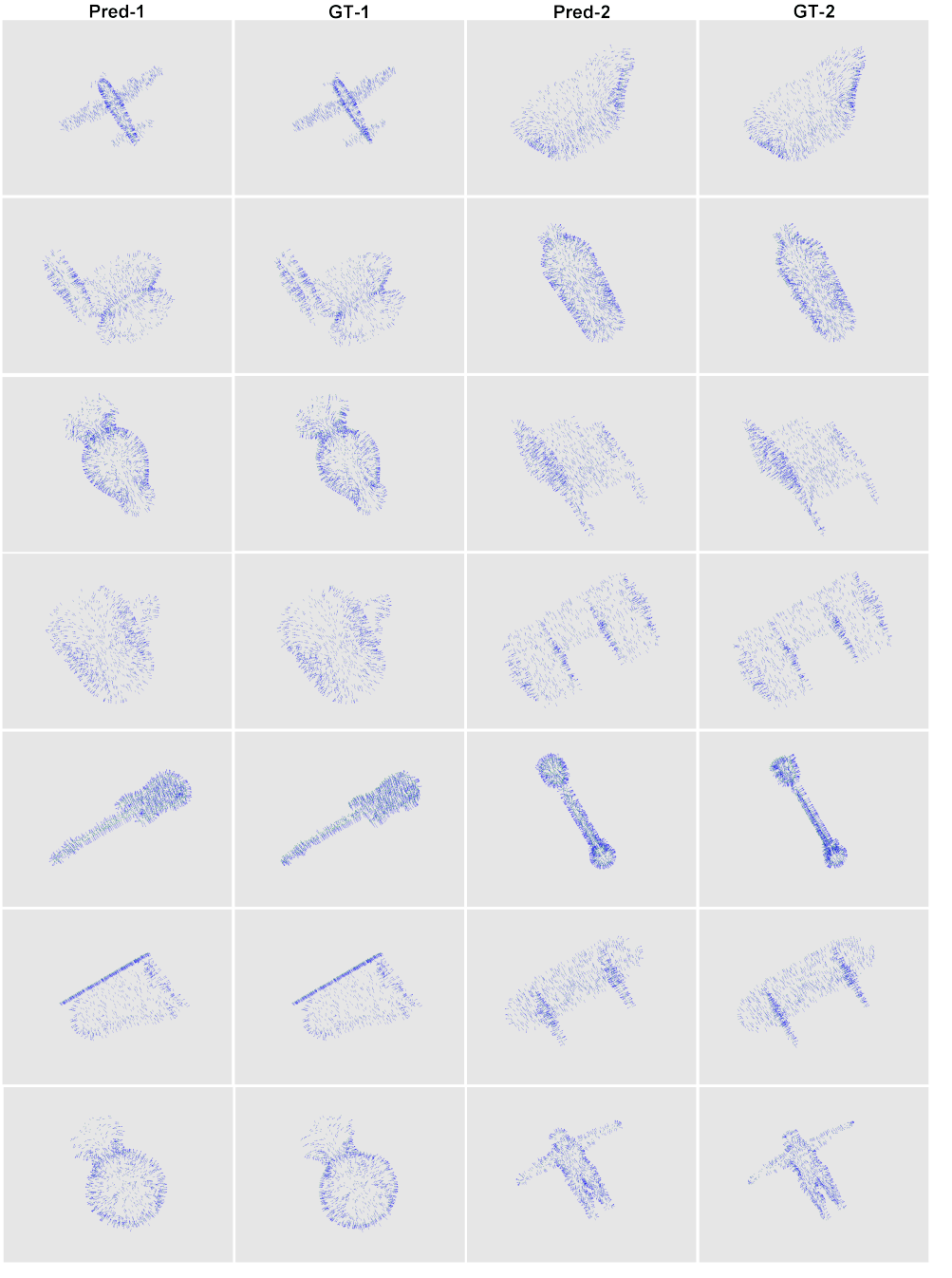


Figure 3: Normal estimation results compared with the groundtruth on the ModelNet 40 dataset.

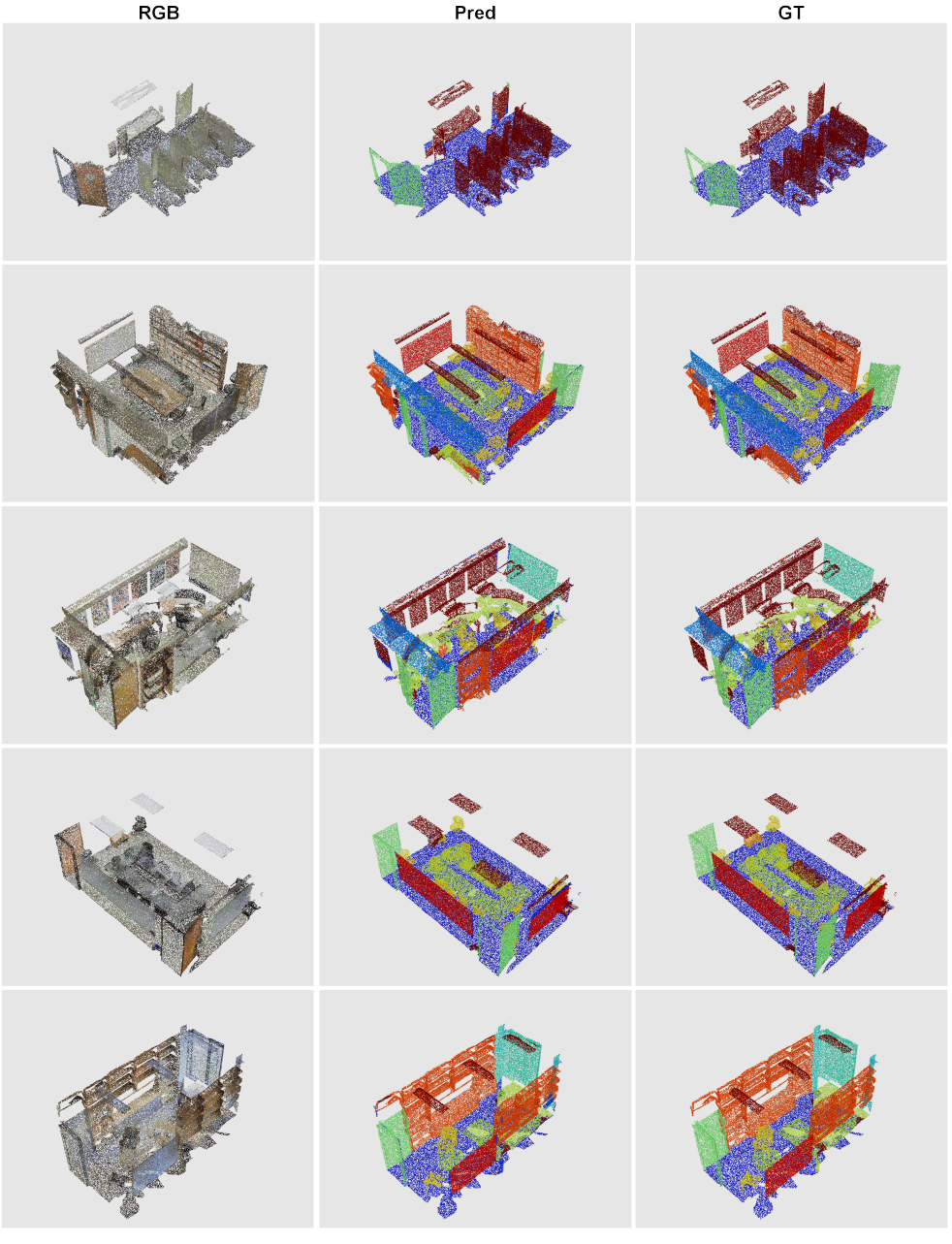


Figure 4: Validation results compared with the groundtruth on the S3DIS dataset. Note that, the "ceiling" and "wall" categories are manually removed in each room for better visualizations

# References


[1] Iro Armeni, Ozan Sener, Amir R Zamir, et al. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[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] Timo Hackel, Nikolay Savinov, Lubor Ladicky, et al. Semantic3d.net: a new large-scale point cloud classification benchmark. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1/W1:91–98, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[5] 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.

[6] 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.

[7] 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.

[8] Yangyan Li, Rui Bu, Mingchao Sun, et al. Pointcnn: convolution on $\chi$-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Shenlong Wang, Simon Suo, Wei-Chiu Ma, et al. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018.

[19] 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.

[20] 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.

[21] Zhirong Wu, Shuran Song, Aditya Khosla, et al. 3d shapenets: a deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[22] 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.

[23] 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.

[24] 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.

[25] 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.