A-CNN: Annularly Convolutional Neural Networks on Point Clouds

Artem Komarichev Zichun Zhong Jing Hua
Department of Computer Science, Wayne State University
{artem.komarichev,zichunzhong,jinghua}@wayne.edu

Abstract

Analyzing the geometric and semantic properties of 3D point clouds through the deep networks is still challenging due to the irregularity and sparsity of samplings of their geometric structures. This paper presents a new method to define and compute convolution directly on 3D point clouds by the proposed annular convolution. This new convolution operator can better capture the local neighborhood geometry of each point by specifying the (regular and dilated) ring-shaped structures and directions in the computation. It can adapt to the geometric variability and scalability at the signal processing level. We apply it to the developed hierarchical neural networks for object classification, part segmentation, and semantic segmentation in large-scale scenes. The extensive experiments and comparisons demonstrate that our approach outperforms the state-of-the-art methods on a variety of standard benchmark datasets (e.g., ModelNet10, ModelNet40, ShapeNet-part, S3DIS, and ScanNet).

1. Introduction

Nowadays, the ability to understand and analyze 3D data is becoming increasingly important in computer vision and computer graphics communities. During the past few years, the researchers have applied deep learning methods to analyze 3D objects inspired by the successes of these techniques in 2D images and 1D texts. Traditional low-level handcrafted shape descriptors suffer from not being able to learn the discriminative and sufficient features from 3D shapes [1]. Recently, deep learning techniques have been applied to extract hierarchical and effective information from 3D shape features captured by low-level descriptors [20, 6]. 3D deep learning methods are widely used in shape classification, segmentation, and recognition, etc. But all these methods are still constrained by the representation power of the shape descriptors.

One of the main challenges to directly apply deep learning methods to 3D data is that 3D objects can be represented in different formats, i.e., regular / structured representation

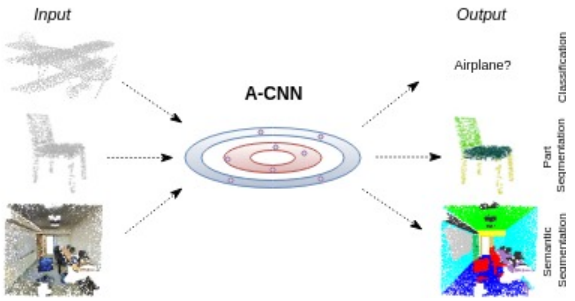


Figure 1: The proposed annularly convolutional neural networks (A-CNN) model on point clouds to perform classification, part segmentation, and semantic segmentation tasks.

(e.g., multi-view images and volumes), and irregular / unstructured representation (e.g., point clouds and meshes). There are extensive approaches based on regular / structured representation, such as multi-view convolutional neural networks (CNNs) [34, 27, 10] and 3D volumetric / grid CNN methods and its variants [40, 27, 29, 37, 38, 16, 9]. These methods can be conveniently developed and implemented in 3D data structure, but they easily suffer from the heavy computation and large memory expense. So it is better to define the deep learning computations based on 3D shapes directly, i.e., irregular / unstructured representation, such as point cloud based methods [26, 28, 13, 32, 3, 18, 19, 35, 17, 44, 36, 8, 42]. However, defining the convolution on the irregular / unstructured representation of 3D objects is not an easy task. Very few methods on point clouds have defined an effective and efficient convolution on each point. Meanwhile, several approaches have been proposed to develop convolutional networks on 2D manifolds [22, 4, 24, 41]. Their representations (e.g., 3D surface meshes) have point positions as well as connectivities, which makes it relatively easier to define the convolution operator on them.

In this work, we present a new method to define and compute convolutions directly on 3D point clouds effectively and efficiently by the proposed *annular convolutions*. This new convolution operator can better capture local neighborhood geometry of each point by specifying the (regular and dilated) ring-shaped structures and directions

in the computation. It can adapt to the geometric variability and scalability at the signal processing level. Then, we apply it along with the developed hierarchical neural networks to object classification, part segmentation, and semantic segmentation in large-scale scene as shown in Fig. 1. The key *contributions* of our work are as follows:

- We propose a new approach to define convolutions on point cloud. The proposed *annular convolutions* can define arbitrary kernel sizes on each local ring-shaped region, and help to capture better geometric representations of 3D shapes;
- We propose a new multi-level hierarchical method based on *dilated rings*, which leads to better capturing and abstracting shape geometric details. The new dilated strategy on point clouds benefits our proposed closed-loop convolutions and poolings;
- Our proposed network models present new state-of-the-art performance on object classification, part segmentation, and semantic segmentation of large-scale scenes using a variety of standard benchmark datasets.

2. Related Work

Due to the scope of our work, we focus only on recently related deep learning methods, which are proposed on different 3D shape representations.

Volumetric Methods. One traditional way to analyze a 3D shape is to convert it into the regular volumetric occupancy grid and then apply 3D CNNs [40, 27]. The major limitation of these approaches is that 3D convolutions are more expensive in computations than 2D cases. In order to make the computation affordable, the volume grid size is usually in a low resolution. However, lower resolution means loosing some shape geometric information, especially in analyzing large-scale 3D shapes / scenes. To overcome these problems, octree-based methods [29, 37, 38] have been proposed to allow applying 3D CNNs on higher / adaptive resolution grids. PointGrid [16] is a 3D CNN that incorporates a constant number of points within each grid cell and allows it to learn better local geometric details. Similarly, Hua et al. [9] presented a 3D convolution operator based on a uniform grid kernel for semantic segmentation and object recognition on point clouds.

Point Cloud based Methods. PointNet [26] is the first attempt of applying deep learning directly on point clouds. PointNet model is invariant to the order of points, but it considers each point independently without including local region information. PointNet++ [28] is a hierarchical extension of PointNet model and learns local structures of point clouds at different scales. But [28] still considers every point in its local region independently. In our work, we address the aforementioned issues by defining the convolution operator that learns the relationship between neighboring points in a local region, which helps to better capture the local geometric properties of the 3D object.

Klokov et al. [13] proposed a new deep learning architecture called Kd-networks, which uses kd-tree structure to construct a computational graph on point clouds. KC-Net [32] improves PointNet model by considering the local neighborhood information. It defines a set of learnable point-set kernels for local neighboring points and presents a pooling method based on a nearest-neighbor graph. PCNN [3] is another method to apply convolutional neural networks to point clouds by defining extension and restriction operators, and mapping point cloud functions to volumetric functions. SO-Net [17] is a permutation invariant network that utilizes spatial distribution of point clouds by building a self-organizing map. There are also some spectral convolution methods on point clouds, such as SyncSpecCNN [44] and spectral graph convolution [36]. Point2Sequence [19] learns the correlation of different areas in a local region by using attention mechanism, but it does not propose a convolution on point clouds. PointCNN [18] is a different method that proposes to transform neighboring points to the canonical order and then apply convolution.

Recently, there are several approaches proposed to process and analyze large-scale point clouds from indoor and outdoor environments. Engelmann et al. [8] extended PointNet model to exploit the large-scale spatial context. Ye et al. [42] proposed a pointwise pyramid pooling to aggregate features at local neighborhoods as well as two-directional hierarchical recurrent neural networks (RNNs) to learn spatial contexts. However, these methods do not define convolutions on large-scale point clouds to learn geometric features in the local neighborhoods. TangentConv [35] is another method that defines the convolution on point clouds by projecting the neighboring points on tangent planes and applying 2D convolutions on them. The orientation of the tangent image is estimated according to the local point / shape curvature, but as we know the curvature computation on the local region of the point clouds is not stable and not robust (see the discussion in Sec. 3.4), which makes it orientation-dependent. Instead, our method proposes an annular convolution, which is invariant to the orientations of local patches. Also, ours does not require additional input features while theirs needs such features (e.g., depth, height, etc.).

Mesh based Methods. Besides point cloud based methods, several approaches have been proposed to develop convolutional networks on 3D meshes for shape analysis. Geodesic CNN [22] is an extension of the Euclidean CNNs to non-Euclidean domains and is based on a local geodesic system of polar coordinates to extract local patches. Anisotropic CNN [4] is another generalization of Euclidean CNNs to non-Euclidean domains, where classical convolutions are replaced by projections over a set of oriented anisotropic diffusion kernels. Mixture Model Net-

works (MoNet) [24] generalizes deep learning methods to non-Euclidean domains (graphs and manifolds) by combining previous methods, e.g., classical Euclidean CNN, Geodesic CNN, and Anisotropic CNN. MoNet proposes a new type of kernel in parametric construction. Directionally Convolutional Networks (DCN) [41] applies convolution operation on the triangular mesh of 3D shapes to address part segmentation problem by combining local and global features. Lastly, Surface Networks [14] propose upgrades to Graph Neural Networks to leverage extrinsic differential geometry properties of 3D surfaces for increasing their modeling power.

3. Method

In this work, we propose a new end-to-end framework named as annularly convolutional neural networks (A-CNN) that leverages the neighborhood information to better capture local geometric features of 3D point clouds. In this section, we introduce main technique components of the A-CNN model on point clouds that include: regular and dilated rings, constraint-based k-nearest neighbors (k-NN) search, ordering neighbors, annular convolution, and pooling on rings.

3.1. Regular and Dilated Rings on Point Clouds

To extract local spatial context of the 3D shape, PointNet++ [28] proposes multi-scale architecture. The major limitation of this approach is that multiple scaled regions may have overlaps (i.e., same neighboring points could be duplicately included in different scaled regions), which reduces the performance of the computational architecture. Overlapped points at different scales lead to redundant information at the local region, which limits a network to learn more discriminative features.

In order to address the above issue, our proposed framework is aimed to leverage a neighborhood at different scales more wisely. We propose two ring-based schemes, i.e., *regular rings* and *dilated rings*. Comparing to multi-scale strategy, the ring-based structure does not have overlaps (no duplicated neighboring points) at the query point's neighborhood, so that each ring contains its own unique points, as illustrated in Sec. A of Supplementary Material.

The difference between regular rings and dilated rings is that dilated rings have empty space between rings. The idea of proposed dilated rings is inspired by dilated convolutions on image processing [45], which benefits from aggregating multi-scale contextual information. Although each ring may define the same number of computation / operation parameters (e.g., number of neighboring points), the coverage area of each ring is different (i.e., dilated rings will have larger coverage than the regular rings) as depicted in Fig. 7. Regular rings can be considered as dilated rings with the dilation factor equal to 0.

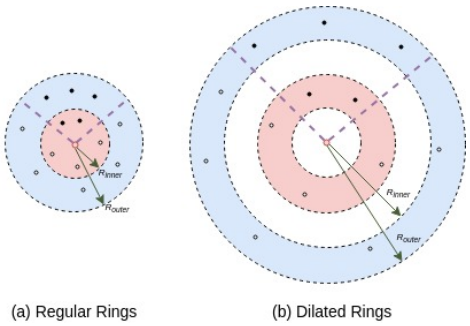


Figure 2: The comparison of the regular and dilated ring-shaped structures (such as with two rings). We can see that comparing two sectors (e.g., black solid points) in the regular and dilated rings, the dilated rings cover larger space by using the same number of neighbors as in regular rings. Moreover, each ring contains unique neighboring points comparing to the other ring.

The proposed regular rings and dilated rings will contribute to neighboring point search, convolution, and pooling in the follow-up processes. First, for k-NN algorithm, we constrain search areas in the local ring-shaped neighborhood to ensure no overlap. Second, the convolutions defined on rings cover larger areas with the same kernel sizes without increasing the number of convolution parameters. Third, the regular / dilated ring architectures will help to aggregate more discriminative features after applying max-pooling at each ring of the local region. We will discuss them in more detail in the following subsections.

To justify the aforementioned statements, we will compare multi-scale approach with our proposed multi-ring scheme on object classification task in the ablation study (Sec. 5.4). The results show that ring-based structure captures better local geometric features than previous multi-scale method, since it achieves higher accuracy.

3.2. Constraint-based K-NN Search

In the original PointNet++ model, the ball query algorithm returns the first K neighbors found inside a search ball specified by a radius R and query point $\mathbf{q}_i$, so that it cannot guarantee that the closest points will always be found. However, our proposed k-NN search algorithm guarantees returning closest points inside the searching area by using the Euclidean metric. Each ring is defined by two parameters: the inner radius R_{inner} and the outer radius R_{outer} (in Fig. 7); therefore, the *constraint-based k-NN search* ensures that the closest and unique points will be found in each ring.

3.3. Ordering Neighbors

In order to learn relationships between neighboring points in a local regions, we need first to order points in a clockwise / counterclockwise manner and then apply annular convolutions. Our proposed ordering operator consists of two main steps: projection and ordering. The importance

of the projection before ordering is that the dot product has its restriction in ordering points. By projecting points on a tangent plane at a query point $\mathbf{q}_i$, we effectively order neighbors in clockwise / counterclockwise direction by taking use of cross product and dot product together. The detailed explanations of normal estimation, orthogonal projection, and ordering are given in the following subsections.

3.3.1 Normal Estimation on Point Clouds

Normal is an important geometric property of a 3D shape. We use it as a tool for projecting and ordering neighboring points at a local domain. The simplest normal estimation method approximates the normal $\mathbf{n}_i$ at the given point $\mathbf{q}_i$ by calculating the normal of the local tangent plane $\mathcal{T}_i$ at that point, which becomes a least-square plane fitting estimation problem [30]. To calculate normal $\mathbf{n}_i$, one needs to compute eigenvalues and eigenvectors of the covariance matrix $\mathbf{C}$ as:

$$\mathbf{C} = \frac{1}{K}\sum_{j=1}^{K}(\mathbf{x}_j - \mathbf{q}_i)\cdot(\mathbf{x}_j - \mathbf{q}_i)^T, \\ \mathbf{C}\cdot\mathbf{v}_\gamma = \lambda_\gamma \cdot \mathbf{v}_\gamma, \gamma \in \{0,1,2\}, \tag{1}$$

where K is the number of neighboring points $\mathbf{x}_j$s around query point $\mathbf{q}_i$ (e.g., K = 10 in our experiments), λ_γ and $\mathbf{v}_\gamma$ are the γth eigenvalue and eigenvector of the covariance matrix $\mathbf{C}$, respectively. The covariance matrix $\mathbf{C}$ is symmetric and positive semi-definite. The eigenvectors $\mathbf{v}_\gamma$ form an orthogonal frame, in respect to the local tangent plane $\mathcal{T}_i$. The eigenvector $\mathbf{v}_0$ that corresponds to the smallest eigenvalue λ_0 is the estimated normal $\mathbf{n}_i$.

3.3.2 Orthogonal Projection

After extracting neighbors $\mathbf{x}_j, j \in \{1,...,K\}$ for a query point $\mathbf{q}_i$, we calculate projections $\mathbf{p}_j$s of these points on a tangent plane $\mathcal{T}_i$ described by a unit normal $\mathbf{n}_i$ (estimated in Sec. 3.3.1) as:

$$\mathbf{p}_j = \mathbf{x}_j - ((\mathbf{x}_j - \mathbf{q}_i)\cdot\mathbf{n}_i)\cdot\mathbf{n}_i, \quad j \in \{1,...,K\}. \tag{2}$$

Fig. 3 (a) illustrates the orthogonal projection of neighboring points on a ring.

3.3.3 Counterclockwise Ordering

Firstly, we use the geometric definition of the dot product to compute the angle between two vectors $\mathbf{c}$ (i.e., starts from the query point $\mathbf{q}_i$ and connects with a randomly starting point, such as $\mathbf{p}_1$) and $\mathbf{p}_j - \mathbf{q}_i$ (i.e., starts from the query point $\mathbf{q}_i$ and connects with other neighboring points $\mathbf{p}_j$):

$$cos(\theta_{\mathbf{p}_j}) = \frac{\mathbf{c}\cdot(\mathbf{p}_j - \mathbf{q}_i)}{||\mathbf{c}||||\mathbf{p}_j - \mathbf{q}_i||}. \tag{3}$$

We know that $cos(\theta_{\mathbf{p}_j})$ lies in $[-1,1]$, which corresponds to angles between $[0°, 180°]$. In order to sort the neighboring points around the query point between

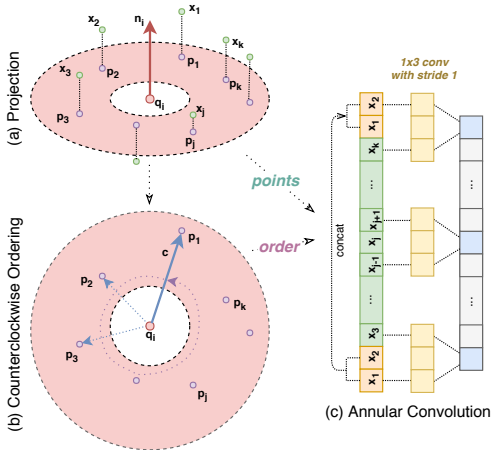


Figure 3: The illustration of the proposed annular convolution on a ring. (a) Projection: $\mathbf{q}_i$ is a query point. After applying the constraint-based k-NN search, neighboring points $\mathbf{X} = \{\mathbf{x}_j | j = 1,...,K\}$ are extracted on a ring. Given the normal $\mathbf{n}_i$ at query point $\mathbf{q}_i$, we project the searched points on the tangent plane $\mathcal{T}_i$. (b) Counterclockwise Ordering: After projection, we randomly pick a starting point as our reference direction $\mathbf{c}$ and order points in counterclockwise. It is worth mentioning that we order original points $[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_j, ..., \mathbf{x}_K]$ based on their projections. (c) Annular Convolution: Depending on the kernel size, we copy several original points from the beginning position and concatenate them to the end of the ordered points. Finally, we apply annular convolution with the given kernel.

$[0°, 360°)$, we must to decide which semicircle the considered point $\mathbf{p}_j$ belongs to as follows:

$$sign_{\mathbf{p}_j} = (\mathbf{c}\times(\mathbf{p}_j - \mathbf{q}_i))\cdot\mathbf{n}_i, \tag{4}$$

where $sign_{\mathbf{p}_j} \geq 0$ is $\theta_{\mathbf{p}_j} \in [0°, 180°]$, and $sign_{\mathbf{p}_j} < 0$ is $\theta_{\mathbf{p}_j} \in (180°, 360°)$.

Then, we can recompute the cosine value of the angle as:

$$\angle_{\mathbf{p}_j} = \begin{cases} -cos(\theta_{\mathbf{p}_j}) - 2 & sign_{\mathbf{p}_j} < 0 \\ cos(\theta_{\mathbf{p}_j}) & sign_{\mathbf{p}_j} \geq 0. \end{cases} \tag{5}$$

Now the values of the angles lie in $(-3, 1]$, which maps angles between $[0°, 360°)$.

Finally, we sort neighboring points $\mathbf{x}_j$ by descending the value of $\angle_{\mathbf{p}_j}$ to obtain the counterclockwise order. Fig. 3 (b) illustrates the process of ordering in a local neighborhood. The neighboring points can be ordered in the clockwise manner, if we sort neighboring points $\mathbf{x}_j$ by ascending the value of $\angle_{\mathbf{p}_j}$.

Our experiments show in Sec. 5.4 that ordering points in the local regions is an important step in our framework and our model achieves better classification accuracy with ordered points than without ordering them.

3.4. Annular Convolution on Rings

Through the previous computation, we have the ordered neighbors represented as an array $[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_K]$. In or-

der to develop the *annular convolution*, we need to loop the array of neighbors with respect to the size of the kernel (e.g., 1×3, 1×5, ...) on each ring. For example, if the convolutional kernel size is 1×3, we need to take the first two neighbors and concatenate them with the ending elements in the original array to construct a new circular array $[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_K, \mathbf{x}_1, \mathbf{x}_2]$. Then, we can perform the standard convolutions on this array as shown in Fig. 3 (c).

There are some nice properties of the proposed annular convolutions as follows: (1) The annular convolution is invariant to the orientation of the local patch. That is because the neighbors are organized and ordered in a closed loop in each ring by concatenating the beginning with the end of the neighboring points' sequence. Therefore, we can order neighbors based on any random starting position, which does not negatively affect the convolution results. Compared with some previous convolutions defined on 3D shapes [4, 41, 35], they all need to compute the real principal curvature direction as the reference direction to define the local patch operator, which is not robust and cumbersome. In particular, 3D shapes have large areas of flat and spherical regions, where the curvature directions are arbitrary. (2) As we know, in reality, the normal direction flipping issues are widely existing in point clouds, especially the large-scale scene datasets. Under the annular convolution strategy, no matter the neighboring points are ordered in clockwise or counterclockwise manner, the results are the same. (3) Another advantage of annular convolution is that we can define an arbitrary kernel size, instead of just 1×1 kernels [26, 28]. Therefore, the annular convolution can provide the ability to learn the relationship between ordered points inside each ring as shown in Fig. 3 (c).

Annular convolutions can be applied on both regular and dilated rings. By applying annular convolutions with the same kernel size on different rings, we can cover and convolve larger areas by using the dilated structure, which helps us to learn larger spatial contextual information in the local regions. The importance of annular convolutions is shown in the ablation study in Sec. 5.4.

3.5. Pooling on Rings

After applying a set of annular convolutions sequentially, the resulting convolved features encode information about its closest neighbors in each ring as well as spatial remoteness from a query point. Then we aggregate the convolved features across all neighbors on each ring separately. We apply the max-pooling strategy in our framework. Our proposed ring-based scheme allows us to aggregate more discriminative features. The extracted max-pooled features contain the encoded information about neighbors and the relationship between them in the local region, unlike the pooling scheme in PointNet++ [28], where each neighbor is considered independently from its neighbors. In our pooling process, the non-overlapped regions (rings) will aggregate different types of features in each ring, which can uniquely describe each local region (ring) around the query point. The multi-scale approach in PointNet++ does not guarantee this and might aggregate the same features at different scales, which is redundant information for a network. The (regular and dilated) ring-based scheme helps to avoid extracting duplicate information but rather promotes extracting multi-level information from different regions (rings). This provides a network with more diverse features to learn from. After aggregating features at different rings, we concatenate and feed them to another abstract layer to further learn hierarchical features.

4. A-CNN Architecture

Our proposed A-CNN model follows a design where the hierarchical structure is composed of a set of abstract layers. Each abstract layer consists of several operations performed sequentially and produces a subset of input points with newly learned features. Firstly, we subsample points by using Farthest Point Sampling (FPS) algorithm [23] to extract centroids randomly distributed on the surface of each object. Secondly, our constraint-based k-NN extracts neighbors of a centroid for each local region (i.e., regular / dilated rings) and then we order neighbors in a counterclockwise manner using projection. Finally, we apply sequentially a set of annular convolutions on the ordered points and max-pool features across neighbors to produce new feature vectors, which uniquely describe each local region.

Given the point clouds of 3D shapes, our proposed end-to-end network is able to classify and segment the objects. In the following, we discuss the classification and segmentation network architectures on 3D point clouds.

4.1. Classification Network

The classification network is illustrated at the top of Fig. 4. It consists of two major parts: encoder and classification. The encoder extracts features from each ring independently inside every layer and concatenates them at the end to process further to extract high-level features. The proposed architecture includes both *regular rings* and *dilated rings*. We end up using two rings per layer, because it gives us pretty good experimental results as shown in the Sec. 5. It can be easily extended to more than two rings per layer, if necessary.

We use regular rings in the first layer and dilated rings in the second layer in the encoder. Annular convolutions with the kernel sizes 1×3 and stride 1 are applied in the first two layers, followed by a batch normalization [12] (BN) and a rectified linear unit [25] (ReLU). Different rings of the same query point are processed in parallel. Then, the aggregated features from each ring concatenate together to propagate to the next layer. The last layer in the encoder performs convolutions with kernel sizes 1×1 followed by BN and ReLU layers, where only spatial positions of the sampled points

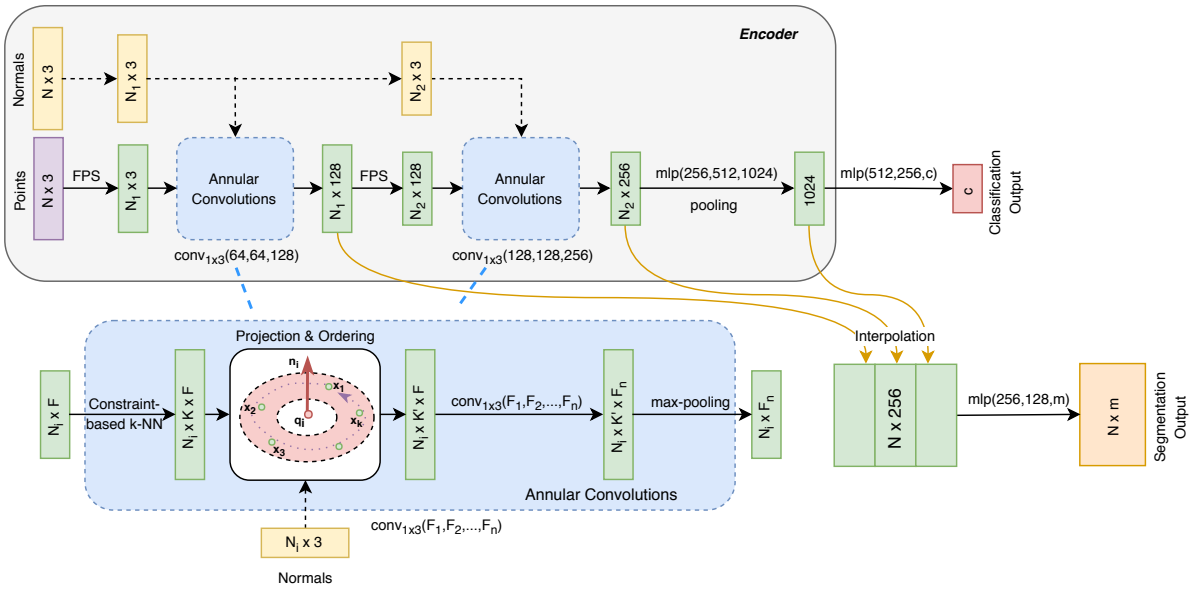


Figure 4: The architecture of A-CNN. Both classification and segmentation networks share encoder part for the feature extraction. Normals are used only to determine the order of neighboring points in the local regions (dashed arrows mean no backpropagation during training) and not used as additional features, unless it is mentioned explicitly in the experiments. N, N_1, N_2 (where $N > N_1 > N_2$) are the numbers of points as input, after the first and second layer, respectively. K and K' are the unordered and ordered points inside the local rings, respectively. c is the number of classification classes. m is the number of segmentation classes. "FPS" stands for Farthest Point Sampling algorithm. "mlp" stands for multi-layer perceptron. $conv_{1\times3}(F_1, F_2, ..., F_n)$ stands for annular convolutions with the kernel size 1×3 applied sequentially with corresponding feature map sizes $F_i, i \in 1, ..., n$.

are considered. After that aggregated high-level features are fed to the set of fully-connected layers with integrated dropout [33] and ReLU layers to calculate probability of each class. The output size of the classification network is equal to the number of classes in the dataset.

4.2. Segmentation Network

The segmentation network shares encoder part with the classification network as shown in Figure 4. In order to predict the segmentation label per point, we need to upsample the sampled points in the encoder back to the original point cloud size. As pointed out by [46], the consecutive feature propagation proposed by [28] is not the most efficient approach. Inspired from [46], we propagate features from different levels from the encoder directly to the original point cloud size, and concatenate them by allowing the network to learn the most important features from different levels as well as to learn the relationship between them.

The output of each level has different sizes due to the hierarchical feature extractions, so we have to restore hierarchical features from each level back to the original point size by using an interpolation method [28]. The interpolation method is based on the inverse squared Euclidean distance weighted average of the three nearest neighbors as:

$$f^{(l+1)}(\mathbf{x}) = \sum_{j=1}^{3} f^{(l)}(\mathbf{x}_j) \frac{w_j(\mathbf{x})}{\sum_{j=1}^{3} w_j(\mathbf{x})}, \qquad (6)$$

where $w_j(\mathbf{x}) = \frac{1}{d(\mathbf{x},\mathbf{x}_j)^2}$ is an inverse squared Euclidean distance weight.

Then, we concatenate upsampled features from different levels and pass them through 1×1 convolution to reduce feature space and learn the relationship between features from different levels. Finally, the segmentation class distribution for each point is calculated.

5. Experiments

We evaluate our A-CNN model on various tasks such as point cloud classification, part segmentation, and large-scale scene segmentation. In the following subsections, we demonstrate more details on each task. It is noted that for the comparison experiments, best results in the tables are shown in bold font.

All models in this paper are trained on a single NVIDIA Titan Xp GPU with 12 GB GDDR5X. The training time of our model is faster than that of PointNet++ model. More details about the network configurations, training settings and timings in our experiments can be found in Sec. B and Tab. 5 of Supplementary Material. The source code of the framework will be made available later.

5.1. Point Cloud Classification

We evaluate our classification model on two datasets: *ModelNet10* and *ModelNet40* [40]. ModelNet is a large-scale 3D CAD model dataset. *ModelNet10* is a subset of ModelNet dataset that consists of 10 different classes with 3991 training and 908 testing objects. *ModelNet40* includes

Table 1: Classification results on *ModelNet10* and *ModelNet40* datasets. AAC is accuracy average class, OA is overall accuracy.

	ModelNet10		*ModelNet40*	
	AAC	OA	AAC	OA
different methods with additional input or more points				
AO-CNN [38]	-	-	-	90.5
O-CNN [37]	-	-	-	90.6
PointNet++ [28]	-	-	-	91.9
SO-Net [17]	95.5	95.7	90.8	93.4
MVCNN-MultiRes [27]	-	-	91.4	93.8
VRN Ensemble [5]	-	97.1	-	95.5
point cloud based methods with 1024 points				
PointNet [26]	-	-	86.2	89.2
Kd-Net (depth 15) [13]	93.5	94.0	88.5	91.8
Pointwise CNN [9]	-	-	81.4	86.1
KCNet [32]	-	94.4	-	91.0
PointGrid [16]	-	-	88.9	92.0
PCNN [3]	-	94.9	-	92.3
PointCNN [18]	-	-	88.1	92.2
Point2Sequence [19]	95.1	95.3	**90.4**	**92.6**
A-CNN (our)	**95.3**	**95.5**	90.3	**92.6**

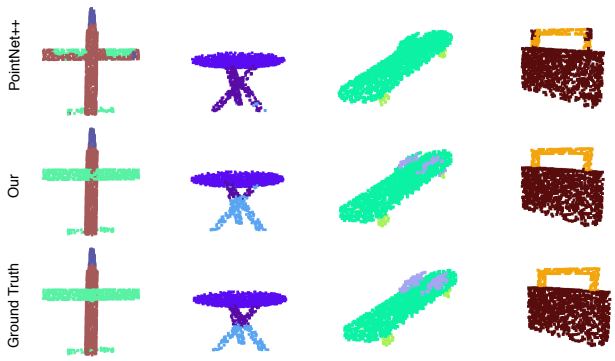
40 different classes with 9843 objects for training and 2468 objects for testing. Point clouds with 10,000 points and normals are sampled from meshes, normalized into a unit sphere, and provided by [28].

For experiments on *ModelNet10* and *ModelNet40*, we sample 1024 points with normals, where normals are only used to order points in the local region. For data augmentation, we randomly scale object sizes, shift object positions, and perturb point locations. For better generalization, we apply point shuffling in order to generate different centroids for the same object at different epochs.

In Tab. 1, we compare our method with several state-of-the-art methods in the shape classification results on both *ModelNet10* and *ModelNet40* datasets. Our model achieves better accuracy among the point cloud based methods (with 1024 points), such as PointNet [26], PointNet++ [28] (5K points + normals), Kd-Net (depth 15) [13], Pointwise CNN [9], KCNet [32], PointGrid [16], PCNN [3], and PointCNN [18]. Our model is slightly better than Point2Sequence [19] on *ModelNet10* and shows comparable performance on *ModelNet40*.

Meanwhile, our model performs better than other volumetric approaches, such as O-CNN [37] and AO-CNN [38]; while we are a little worse than SO-Net [17], which uses denser input points, i.e., 5000 points with normals as the input (1024 points in our A-CNN); MVCNN-MultiRes [27], which uses multi-view 3D volumes to represent an object (i.e., 20 views of $30 \times 30 \times 30$ volume); the VRN Ensemble [5], which involves an ensemble of six models.

We also provide some feature visualization results in Sec. C of Supplementary Material, including global feature (e.g., t-SNE clustering) visualization and local feature (e.g., the magnitude of the gradient per point) visualization.



(a) **Rocket** (b) **Table** (c) **Skateboard** (d) **Bag**

Figure 5: Qualitative results on *ShapeNet-part* dataset. We compare our results with PointNet++ [28] and ground truth.

5.2. Point Cloud Segmentation

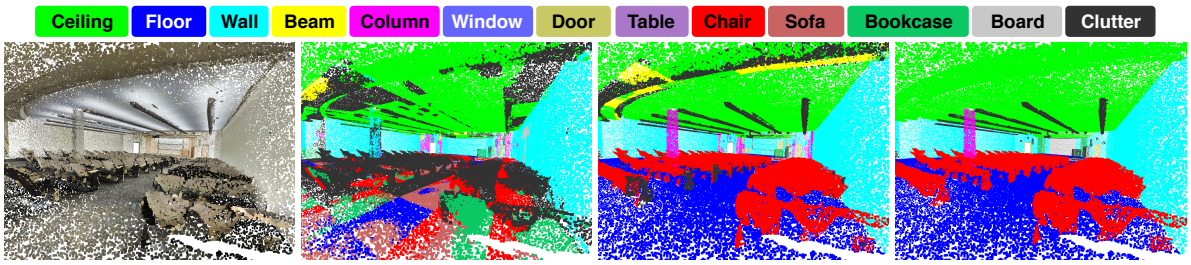
We evaluate our segmentation model on *ShapeNet-part* [43] dataset. The dataset contains 16,881 shapes from 16 different categories with 50 label parts in total. The main challenge of this dataset is that all categories are highly imbalanced. There are 2048 points sampled for each shape from the dataset, where most shapes contain less than six parts. We follow the same training and testing splits provided in [26, 43]. For data augmentation, we perturb point locations with the point shuffling for better generalization.

We evaluate our segmentation model with two different inputs. One of the models is trained without feeding normals as additional features and the other model is trained with normals as additional features. The quantitative results are provided in Tab. 2, where mean IoU (Intersection-over-Union) is reported. The qualitative results are visualized in Fig. 5. Our approach with point locations only as input outperforms PointNet [26], Kd-Net [13], KCNet [32], and PCNN [3]; and shows slightly worse performance comparing to PointGrid [16] (volumetric method) and PointCNN [18]. Meanwhile, our model achieves the best performance with the input of point locations and normals, compared with PointNet++ [28], SyncSpecCNN [44], SO-Net [17], SGPN [39], O-CNN [37], RSNet [11], and Point2Sequence [19]. The more detailed quantitative results (e.g., per-category IoUs) and more visualization results are provided in Sec. E of Supplementary Material.

5.3. Semantic Segmentation in Scenes

We also evaluate our segmentation model on two large-scale indoor datasets *Stanford 3D Large-Scale Indoor Spaces (S3DIS)* [2] and *ScanNet* [7]. *S3DIS* contains 6 large-scale indoor areas with 271 rooms sampled from 3 different buildings, where each point has the semantic label that belongs to one of the 13 categories. *ScanNet* includes 1513 scanned indoor point clouds, where each voxel has been labeled with one of the 21 categories.

We employ the same training and testing strategies as



(a) **Input** (b) **PointNet [26]** (c) **Our** (d) **Ground Truth**

Figure 6: Qualitative results on *S3DIS* dataset. We compare our results with PointNet [26] and ground truth. The auditorium is a challenging room type and appears only in Area 2. Our model produces much better segmentation result, compared with the result of PointNet.

PointNet [26] on *S3DIS*, where we use 6-fold cross validation over all six areas. The evaluation results are reported in Tab. 2, and qualitative results are visualized in Fig. 6. Our model demonstrates better segmentation results compared with PointNet [26], MS+CU (2) [8], G+RCU [8], 3P-RNN [42], SPGraph [15], and TangentConv [35]. However, our model performs slightly worse than PointCNN [18] due to their non-overlapping block sampling strategy with paddings which we do not use. Meanwhile, our approach shows the best segmentation results on *ScanNet* [7] and achieves the state-of-the-art performance, compared with PointNet [26], PointNet++ [28], TangentConv [35], and PointCNN [18] according to Tab. 2.

More qualitative visualization results and data preparation details on both datasets are provided in Sec. D and Sec. E, respectively, of Supplementary Material and Video.

Table 2: Segmentation results on *ShapeNet-part*, *S3DIS*, and *ScanNet*. "mean" is mean IoU (%), OA is overall accuracy.

	ShapeNet-part		*S3DIS*	*ScanNet*
	without normals	*with normals*	OA	OA
	mean	mean		
PointNet [26]	83.7	-	78.5	73.9
PointNet++ [28]	-	85.1	-	84.5
SyncSpecCNN [44]	-	84.7	-	-
O-CNN [37]	-	85.9	-	-
Kd-Net [13]	82.3	-	-	-
KCNet [32]	84.7	-	-	-
SO-Net [17]	-	84.9	-	-
SGPN [39]	-	85.8	-	-
MS+CU (2) [8]	-	-	79.2	-
G+RCU [8]	-	-	81.1	-
RSNet [11]	-	84.9	-	-
3P-RNN [42]	-	-	86.9	-
SPGraph [15]	-	-	85.5	-
TangentConv [35]	-	-	*	80.9
PCNN [3]	85.1	-	-	-
Point2Sequence [19]	-	85.2	-	-
PointGrid [16]	**86.4**	-	-	-
PointCNN [18]	86.1	-	**88.1**	85.1
A-CNN (our)	85.9	**86.1**	87.3	**85.4**

Note: * TangentConv [35] OA on *S3DIS* Area 5 is 82.5% (as reported in their paper), which is worse compared with our OA of 85.5%.

5.4. Ablation Study

The goal of our ablation study is to show the importance of the proposed technique components (in Sec. 3) in our A-CNN model. We evaluate three proposed components, such as rings without overlaps (Sec. 3.1), ordering (Sec. 3.3), and annular convolution (Sec. 3.4) on the classification task of *ModelNet40* dataset as shown in Tab. 4. In the first experiment, we replace our proposed constraint-based k-NN on ring regions with ball query in [28], but keep ordering and annular convolutions on. In the second and third experiments, we turn off either annular convolutions or ordering, respectively; and keep the rest two components on. Our experiments show that the proposed ring-shaped scheme contributes the most to our model. It is because multi-level rings positively affect annular convolutions. Finally, A-CNN model with all three components (i.e., rings without overlaps, ordering, and annular convolutions) achieves the best results. We also discover that reducing overlap / redundancy in multi-scale scheme can improve existing methods. We evaluate the original PointNet++ [28] with and without overlap as shown in Sec. A of Supplementary Material.

Table 3: Ablation experiments on *ModelNet40* dataset. AAC is accuracy average class, OA is overall accuracy.

	AAC	OA
A-CNN (without rings / with overlap)	89.2	91.7
A-CNN (without annular conv.)	89.2	91.8
A-CNN (without ordering)	89.6	92.0
A-CNN (with all components)	**90.3**	**92.6**

6. Conclusion

In this work, we propose a new A-CNN framework on point clouds, which can better capture local geometric information of 3D shapes. Through extensive experiments on several benchmark datasets, our method has achieved the state-of-the-art performance on point cloud classification, part segmentation, and large-scale semantic segmentation tasks. Since our work does not solely focus on large-scale scene datasets, we will explore some new deep learning architectures to improve the current results. We will also investigate to apply the proposed framework on large-scale outdoor datasets in our future work.

Acknowledgment. We would like to thank the reviewers for their valuable comments. This work was partially supported by the NSF IIS-1816511, CNS-1647200, OAC-1657364, OAC-1845962, Wayne State University Sub-award 4207299A of CNS-1821962, NIH 1R56AG060822-01A1, and ZJNSF LZ16F020002.

References

[1] E. Ahmed, A. Saint, A. Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada, and B. Ottersten. Deep learning advances on different 3D data representations: A survey. *arXiv preprint arXiv:1808.01462*, 2018.

[2] I. Armeni, O. Sener, A. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3D semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[3] M. Atzmon, H. Maron, and Y. Lipman. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics*, 37(4):71:1–71:12, 2018.

[4] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 3189–3197, 2016.

[5] A. Brock, T. Lim, J. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016.

[6] S. Bu, P. Han, Z. Liu, J. Han, and H. Lin. Local deep feature learning framework for 3D shape. *Computers & Graphics*, 46:117–129, 2015.

[7] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[8] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe. Exploring spatial context for 3D semantic segmentation of point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–724, 2017.

[9] B.-S. Hua, M.-K. Tran, and S.-K. Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018.

[10] H. Huang, E. Kalogerakis, S. Chaudhuri, D. Ceylan, V. G. Kim, and E. Yumer. Learning local shape descriptors from part correspondences with multiview convolutional networks. *ACM Transactions on Graphics*, 37(1):6, 2018.

[11] Q. Huang, W. Wang, and U. Neumann. Recurrent slice networks for 3D segmentation of point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2635, 2018.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[13] R. Klokov and V. 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.

[14] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and J. Bruna. Surface networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2540–2548, 2018.

[15] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[16] T. Le and Y. Duan. PointGrid: A deep network for 3D shape understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9204–9214, 2018.

[17] J. Li, B. M. Chen, and G. H. Lee. SO-Net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9397–9406, 2018.

[18] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. PointCNN: Convolution on X-transformed points. In *Advances in Neural Information Processing Systems*, pages 828–838, 2018.

[19] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker. Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network. In *Association for the Advancement of Artificial Intelligence*, 2019.

[20] Z. Liu, S. Chen, S. Bu, and K. Li. High-level semantic feature for 3D shape based on deep belief networks. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1–6, 2014.

[21] L. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[22] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 37–45, 2015.

[23] C. Moenning and N. A. Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory, 2003.

[24] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–1543, 2017.

[25] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning*, pages 807–814, 2010.

[26] C. Qi, H. Su, K. Mo, and L. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[27] C. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656, 2016.

[28] C. Qi, L. Yi, H. Su, and L. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017.

[29] G. Riegler, A. Ulusoy, and A. Geiger. OctNet: Learning deep 3D representations at high resolutions. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, pages 3577–3586, 2017.

[30] R. Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[31] R. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.

[32] Y. Shen, C. Feng, Y. Yang, and D. Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4548–4557, 2018.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[34] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 945–953, 2015.

[35] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou. Tangent convolutions for dense prediction in 3D. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018.

[36] C. Wang, B. Samari, and K. Siddiqi. Local spectral graph convolution for point set feature learning. In *Proceedings of The European Conference on Computer Vision*, September 2018.

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

[38] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong. Adaptive O-CNN: A patch-based deep representation of 3D shapes. *ACM Transactions on Graphics*, 37(6), 2018.

[39] W. Wang, R. Yu, Q. Huang, and U. Neumann. 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.

[40] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[41] H. Xu, M. Dong, and Z. Zhong. Directionally convolutional networks for 3D shape segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2698–2707, 2017.

[42] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang. 3D recurrent neural networks with context fusion for point cloud semantic segmentation. In *Proceedings of The European Conference on Computer Vision*, September 2018.

[43] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, L. Guibas, et al. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics*, 35(6):210, 2016.

[44] L. Yi, H. Su, X. Guo, and L. Guibas. SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6584–6592, 2017.

[45] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016.

[46] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. PU-Net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2018.

Supplementary Material:
A-CNN: Annularly Convolutional Neural Networks on Point Clouds

A. Ball Query *vs* Ring-based Scheme

The comparison of multi-scale method proposed in [28] and our ring-based scheme is depicted in Fig. 7. It is noted that comparing to multi-scale regions, the ring-based structure does not have overlaps (no neighboring point duplication) at the query point's neighborhood. It means that each ring contains its own unique points.

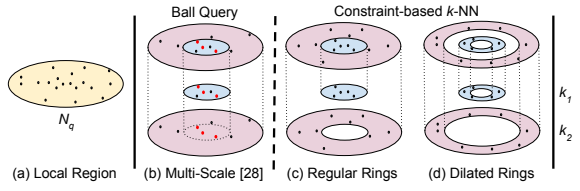


Figure 7: A schematic comparison for searching neighbors in a local region with N_q points between multi-scale approach from [28] and our proposed approaches with regular and dilated rings. The number of neighboring points per region (e.g., k_1 and k_2) is the same between different methods. Regions in multi-scale architecture have neighboring overlaps (red points belong to different regions near the same query point $\mathbf{q}$), while regular and dilated rings have the unique neighbors.

Table 4: Experiments on redundancy on *ModelNet40* dataset. AAC is accuracy average class, OA is overall accuracy.

	AAC	OA
PointNet++ (multi-scale / with overlap)	86.5	90.2
PointNet++ (multi-ring / without overlap)	87.3	90.6
A-CNN (with all components)	**90.3**	**92.6**

We have discovered that reducing redundancy can improve the existing multi-scale approach in [28]. We test redundancy issue on original PointNet++ model [28] with and without overlap / redundancy. We compare the original PointNet++ multi-scale model with ball queries (with redundant points) against PointNet++ with our proposed regular rings (without redundant points). Our experiments show that the proposed multi-ring (i.e., without redundant points) outperforms the multi-scale scheme (i.e., with redundant points) on *ModelNet40* according to Tab. 4.

B. Training Details

We use *A-CNN-3L* network configuration in Tab. 5 for all experiments on point cloud classification tasks and *A-CNN-4L* network configuration in Tab. 5 for both part segmentation and semantic segmentation tasks. We use regular rings in L_1 and dilated rings in L_2 in our *A-CNN-3L* architecture. Similarly, we use regular rings in L_1 and dilated rings in L_2 and L_3 in our *A-CNN-4L* architecture.

We use Adam optimization method with learning rate 0.001 and decay rate 0.7 in classification and decay 0.5 in segmentation tasks. We have trained our classification model for 250 epochs, our part segmentation model for 200 epochs, and our large-scale semantic segmentation models for 50 epochs on each area of *S3DIS* and for 200 epochs on *ScanNet*. The training time of our model is faster than that of PointNet++ model, since we use ring-based neighboring search, which is more efficient and effective than ball query in PointNet++ model. For instance, the training time on the segmentation model for 200 epochs is about 19 hours on a single NVIDIA Titan Xp GPU with 12 GB GDDR5X, and PointNet++ model needs about 32 hours for the same task. The size of our trained model is 22.3 MB and the size of PointNet++ model is 22.1 MB.

C. Feature Visualization

Local Feature Visualization. Fig. 8 and Fig. 9 visualize the magnitude of the gradient per point in the classification task on *ModelNet10* and *ModelNet40* datasets. Blue color represents low magnitude of the gradients and red color represents high magnitude of the gradients. The points with higher magnitudes get greater updates during training and the learning contribution of them is higher. Therefore, this feature visualization could be thought as the object saliency. For example, in *ModelNet40* dataset our model considers wings and tails as important regions to classify an object as an airplane; bottle neck is important for a bottle; the flowers and leaves are important for a plant; tube or middle part (usually narrow parts) is important for a lamp; legs are important to classify an object as a stool.

Global Feature Visualization. Fig. 10 and Fig. 11 shows the t-SNE clustering visualization [21] of the learned global shape features from the proposed A-CNN model for the shape classification tasks in *ModelNet10* and *ModelNet40* test splits. We reduce 1024-dim feature vectors to 2-dim features. We can see that similar shapes are well clustered together according to their semantic categories. For example, in *ModelNet10* dataset the clusters of desk,

Table 5: Network configurations.

		L_1	L_2	L_3	L_4
A-CNN-3L (classification)	C	512	128	1	-
	rings	[[0.0, 0.1], [0.1, 0.2]]	[[0.1, 0.2], [0.3, 0.4]]	-	-
	k	[16, 48]	[16, 48]	128	-
	F	[[32,32,64], [64,64,128]]	[[64,64,128], [128,128,256]]	[256,512,1024]	-
A-CNN-4L (segmentation)	C	512	128	32	1
	rings	[[0.0, 0.1], [0.1, 0.2]]	[[0.1, 0.2], [0.3, 0.4]]	[[0.2, 0.4], [0.6, 0.8]]	-
	k	[16, 48]	[16, 48]	[16, 48]	32
	F	[[32,32,64], [64,64,128]]	[[64,64,128], [128,128,256]]	[[128,128,256], [256,256,512]]	[512,768,1024]

Note: Both of the models represent encoder part. *A-CNN-3L* model consists of three layers. A-CNN-4L model consists of four layers. For each layer, C is the number of centroids, $rings$ is the inner and outer radiuses of a ring: $[R_{inner}, R_{outer}]$, k is number of neighbors, F is feature map size. For example, our *A-CNN-4L* model at the first layer L_1 has 512 centroids; two regular rings where first ring constrained by radiuses of 0.0 and 0.1 and the second ring has radiuses of 0.1 and 0.2; k-NN search returns 16 points in the first ring, and 48 points in the second ring; the feature map size in the first ring is equal to $[32, 32, 64]$ and in the second ring is $[64, 64, 128]$. Convolutional kernel size across different rings and layers is the same and equal to 1×3. Also, we have to double the number of centroids in each layer in model *A-CNN-4L* on *ScanNet* as the number of points in each block is twice more than that in *S3DIS*.

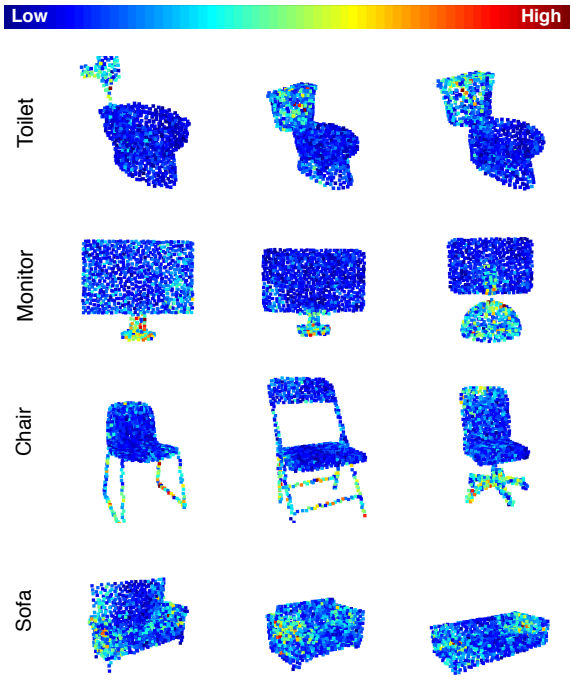


Figure 8: The magnitude of the gradient per point in the classification task on *ModelNet10* dataset.

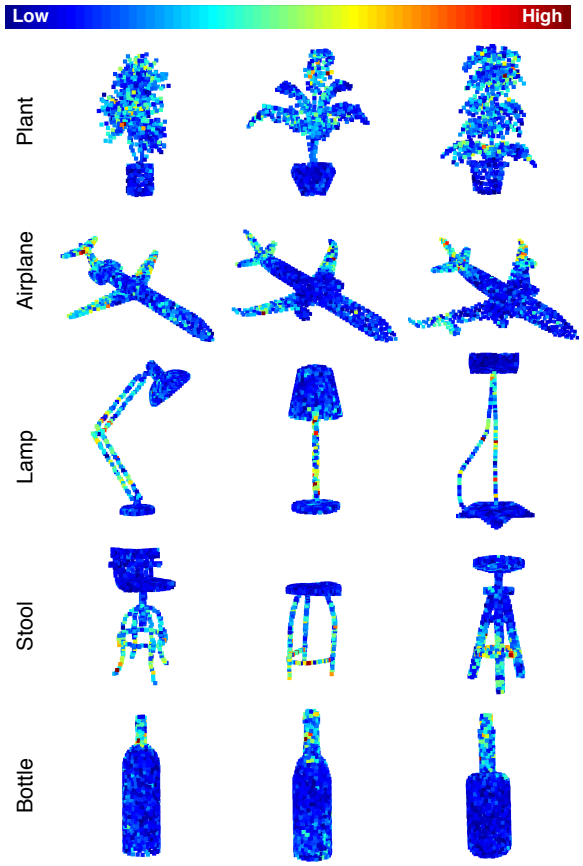


Figure 9: The magnitude of the gradient per point in the classification task on *ModelNet40* dataset.

dresser, night stand, and table classes are closer and even intersect with each other, because the objects from these classes look similar. The perplexity parameters for *ModelNet10* and *ModelNet40* datasets are set as 15 and 50, respectively, to reduce spare space between clusters.

D. Data Preparation Details

***S3DIS* data preparation.** To prepare training and testing datasets, we divide every room into blocks with a size of 1 m $\times$ 1 m $\times$ 2 m and with a stride of 0.5 m. We has sampled 4096 points from each block. The height of each block is scaled to 2 m to ensure that our constraint-based k-NN search works optimally with the provided radiuses. In total, the prepared dataset contains 23,585 blocks across all six areas. Each point is represented as a 6D vector (XYZ: normalized global point coordinates and centered at origin, RGB: colors). We do not use the relative position of the block in the room scaled between 0 and 1 as used in [26], because our model already achieves better results without using this additional information. We calculate point normals for each room by using the Point Cloud Library (PCL) library [31]. The calculated normals are only used to order points in the local region. For data augmentation, we use the same data augmentation strategy as used in the point cloud segmentation on *ShapeNet-part* dataset which is point per-

Table 6: Segmentation results on *ShapeNet-part* dataset (input is *XYZ* only). Per-category and mean IoUs (%) are reported.

	mean	areo	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate board	table
# shapes		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
PointNet [26]	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
Kd-Net [13]	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
KCNet [32]	84.7	82.8	81.5	**86.4**	77.6	90.3	76.8	91.0	87.2	84.5	95.5	69.2	94.4	81.6	60.1	75.2	81.3
PCNN [3]	85.1	82.4	80.1	85.5	79.5	90.8	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
PointGrid [16]	**86.4**	**85.7**	82.5	81.8	77.9	**92.1**	**82.4**	**92.7**	85.8	84.2	95.3	65.2	93.4	81.7	56.9	73.5	**84.6**
PointCNN [18]	86.1	84.1	86.5	86.0	**80.8**	90.6	79.7	92.3	**88.4**	**85.3**	**96.1**	**77.2**	**95.2**	**84.2**	**64.2**	**80.0**	83.0
A-CNN (our)	85.9	83.9	**86.7**	83.5	79.5	91.3	77.0	91.5	86.0	85.0	95.5	72.6	94.9	83.8	57.8	76.6	83.0

Table 7: Segmentation results on *ShapeNet-part* dataset (input is *XYZ* + *normals*). Per-category and mean IoUs (%) are reported.

	mean	areo	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate board	table
# shapes		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
PointNet++ [28]	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
SyncSpecCNN [44]	84.7	81.6	81.7	81.9	75.2	90.2	74.9	**93.0**	86.1	84.7	95.6	66.7	92.7	81.6	60.6	82.9	82.1
SO-Net [17]	84.9	82.8	77.8	**88.0**	77.3	90.6	73.5	90.7	83.9	82.8	94.8	69.1	94.2	80.9	53.1	72.9	83.0
SGPN [39]	85.8	80.4	78.6	78.8	71.5	88.6	78.0	90.9	83.0	78.8	**95.8**	**77.8**	93.8	**87.4**	60.1	**92.3**	**89.4**
RSNet [11]	84.9	82.7	86.4	84.1	78.2	90.4	69.3	91.4	87.0	83.5	95.4	66.0	92.6	81.8	56.1	75.8	82.2
O-CNN (+ CRF) [37]	85.9	**85.5**	**87.1**	84.7	77.0	91.1	**85.1**	91.9	**87.4**	83.3	95.4	56.9	**96.2**	81.6	53.5	74.1	84.4
Point2Sequence [19]	85.2	82.6	81.8	87.5	77.3	90.8	77.1	91.1	86.9	83.9	95.7	70.8	94.6	79.3	58.1	75.2	82.8
A-CNN (our)	**86.1**	84.2	84.0	**88.0**	**79.6**	**91.3**	75.2	91.6	87.1	**85.5**	95.4	75.3	94.9	82.5	**67.8**	77.5	83.3

Note: "CRF" stands for conditional random field method for final result refinement in O-CNN method.

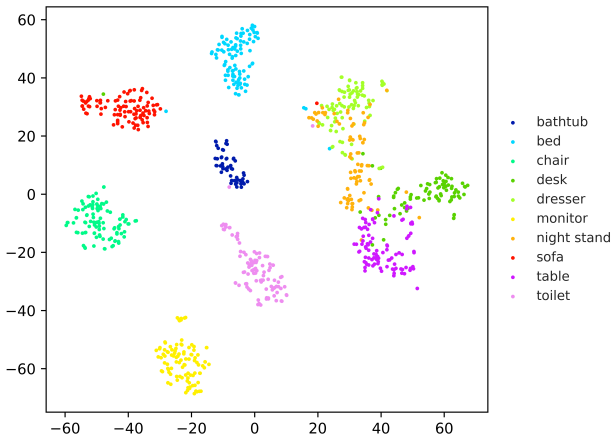


Figure 10: The t-SNE clustering visualization of the learned global shape features from the proposed A-CNN model for the shapes in *ModelNet10* test split.

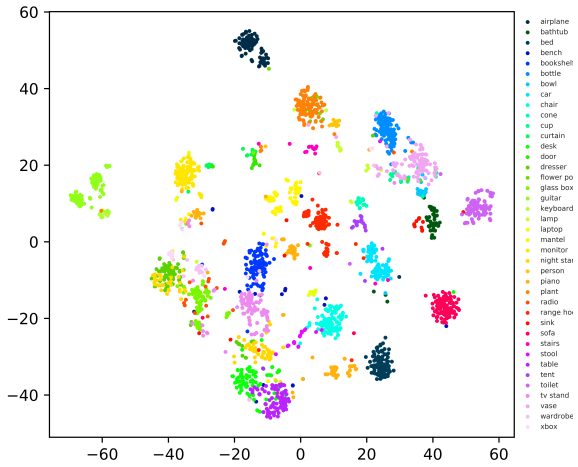


Figure 11: The t-SNE clustering visualization of the learned global shape features from the proposed A-CNN model for the shapes in *ModelNet40* test split.

turbation with point shuffling.

***ScanNet* data preparation**. ScanNet divides original 1513 scanned scenes in 1201 and 312 for training and testing, respectively. We sample blocks from the scenes following the same procedure as in [28], where every block has a size of 1.5 m $\times$ 1.5 m with 8192 points. We estimate point normals using the PCL library [31]. Each point is represented as a 6D vector (XYZ: coordinates of the block centered at origin, $N_xN_yN_z$: normals) without RGB information. For data augmentation, we use the point perturbation with point shuffling.

E. More Experimental Results

Point Cloud Segmentation. Tab. 6 and Tab. 7 show the quantitative results of part segmentation on *ShapeNet-part* dataset with two different inputs. Tab. 6 reports results when the input is point position only. Tab. 7 reports results when the input is point position with its normals.

For *ShapeNet-part* dataset, we visualize more results (besides the segmentation results shown in the paper) in Fig. 12. We compare our results with PointNet++ [28], and our A-CNN model can produce better segmentation results than PointNet++ model.

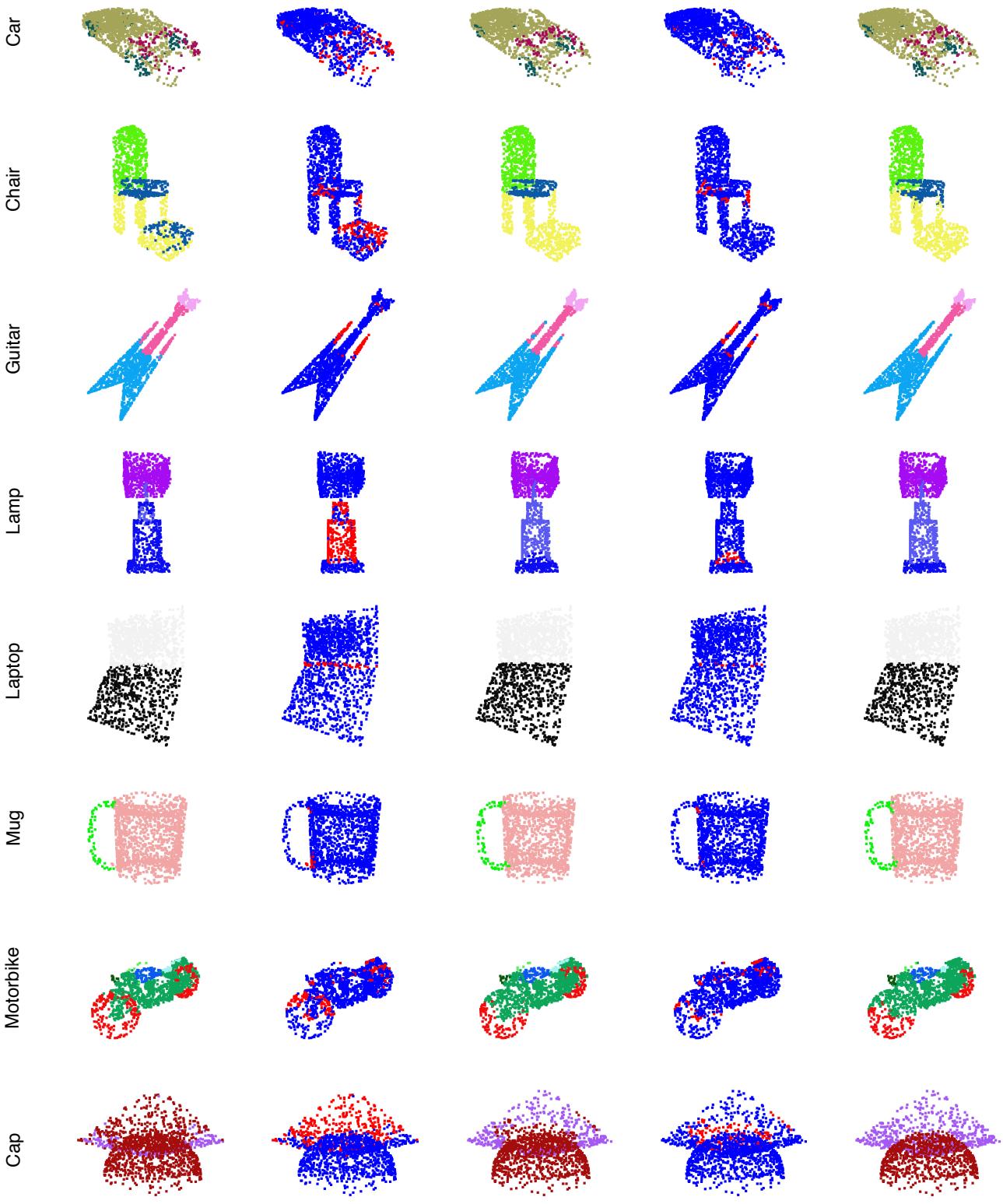
Semantic Segmentation in Scenes. For *S3DIS* dataset, we pick rooms from all six areas: area 1 (*row 1*), area 2

Table 8: Segmentation results on *S3DIS* dataset. "acc" is overall accuracy and "mean" is average IoU over 13 classes.

	acc	mean	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [26]	78.5	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
MS+CU (2) [8]	79.2	47.8	88.6	95.8	67.3	36.9	24.9	48.6	52.3	51.9	45.1	10.6	36.8	24.7	37.5
G+RCU [8]	81.1	49.7	90.3	92.1	67.9	44.7	24.2	52.3	51.2	58.1	47.4	6.9	39.0	30.0	41.9
RSNet [11]	-	56.5	92.5	92.8	78.6	32.8	34.4	51.6	68.1	59.7	60.1	16.4	50.2	44.9	52.0
3P-RNN [42]	86.9	56.3	92.9	93.8	73.1	42.5	25.9	47.6	59.2	60.4	66.7	24.8	57.0	36.7	51.6
SPGraph [15]	85.5	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
PointCNN [18]	**88.1**	**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**
A-CNN (our)	87.3	62.9	92.4	96.4	**79.2**	59.5	34.2	56.3	65.0	66.5	**78.0**	28.5	56.9	48.0	56.8

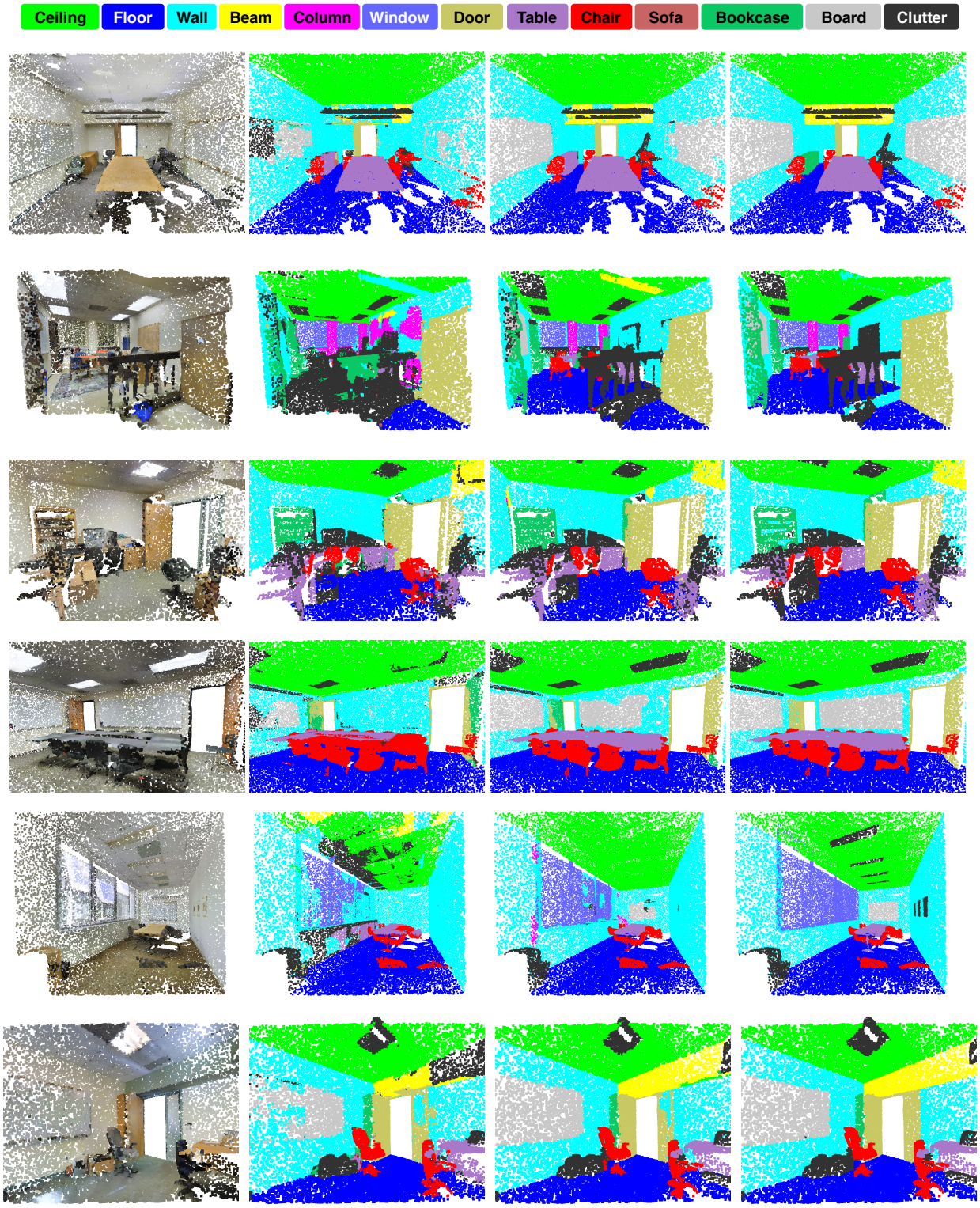
(*row 2*), area 3 (*row 3*), area 4 (*row 4*), area 5 (*row 5*), and area 6 (*row 6*); and compare them with PointNet [26] results and ground truth. The results are shown in Fig. 13. The detailed quantitative evaluation results for each shape class are reported in Tab. 8. Our model demonstrates good semantic segmentation results and achieves the state-of-the-art performance on segmenting *walls* and *chairs*. Meanwhile, our model performs slightly worse than PointCNN [18] on other categories due to their non-overlapping block sampling strategy with paddings which we do not use. Supplementary Video is included for dynamically visualizing each area in detail.

For *ScanNet* dataset, we pick six challenging scenes and visualize the results of our A-CNN model, PointNet++ [28], and ground truth side by side. The visualization results are provided in Fig. 14. Our approach outperforms PointNet++ [28] and other baseline methods, such as PointNet [26], TangentConv [35], and PointCNN [18] according to Tab. 2 in the main paper.



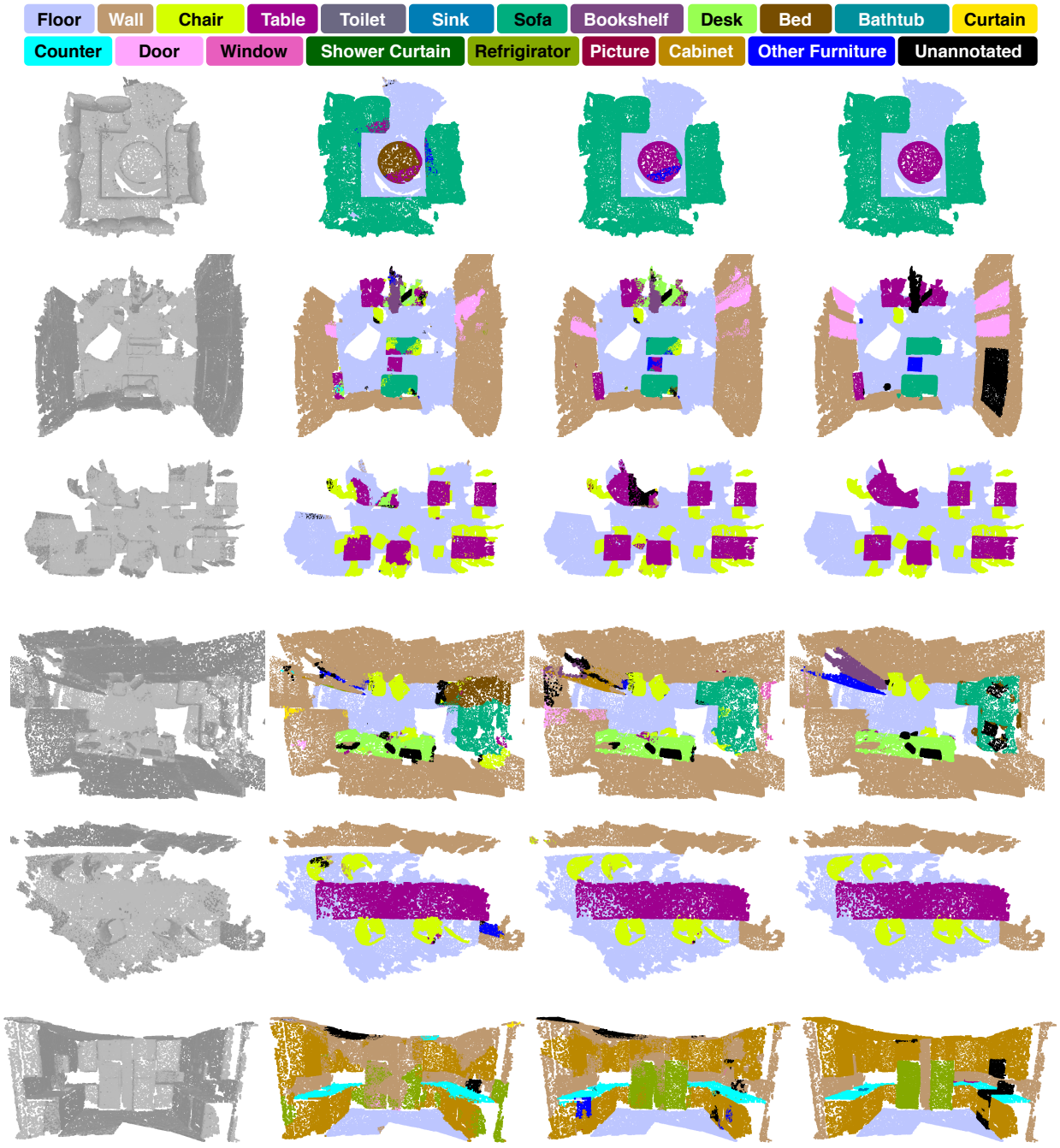
(a) **PointNet++ [28]** (b) **PointNet++ (diff)** (c) **Our** (d) **Our (diff)** (e) **Ground Truth**

Figure 12: More segmentation results on *ShapeNet-part* dataset. Second and fourth columns show the differences between ground truth and prediction (red points are mislabeled points) of PointNet++ and our method.



(a) **Input** (b) **PointNet [26]** (c) **Our** (d) **Ground Truth**

Figure 13: The visualization results on *S3DIS* dataset. We compare our model with PointNet [26] and the ground truth. The challenging sample rooms have been picked from the all six areas: area 1 (*row 1*), area 2 (*row 2*) area 3 (*row 3*), area 4 (*row 4*), area 5 (*row 5*), and area 6 (*row 6*).



(a) **Input** (b) **PointNet++ [28]** (c) **Our** (d) **Ground Truth**

Figure 14: The visualization results on *ScanNet* dataset. We compare our model with PointNet++ [28] and the ground truth. The challenging sample rooms have been picked from the *ScanNet* dataset.