

# VV-NET: Voxel VAE Net with Group Convolutions for Point Cloud Segmentation

Hsien-Yu Meng<sup>1,4</sup>, Lin Gao<sup>2\*</sup>, Yu-Kun Lai<sup>3</sup>, Dinesh Manocha<sup>1</sup>

<sup>1</sup>University of Maryland, College Park

<sup>2</sup>Beijing Key Laboratory of Mobile Computing and Pervasive Device,

Institute of Computing Technology, Chinese Academy of Sciences

<sup>3</sup>School of Computer Science & Informatics, Cardiff University

<sup>4</sup>Tsinghua University

mengxy19@umd.edu, gaolin@ict.ac.cn, LaiY4@cardiff.ac.uk, dm@cs.umd.edu

## Abstract

We present a novel algorithm for point cloud segmentation. Our approach transforms unstructured point clouds into regular voxel grids, and further uses a kernel-based interpolated variational autoencoder (VAE) architecture to encode the local geometry within each voxel. Traditionally, the voxel representation only comprises Boolean occupancy information which fails to capture the sparsely distributed points within voxels in a compact manner. In order to handle sparse distributions of points, we further employ radial basis functions (RBF) to compute a local, continuous representation within each voxel. Our approach results in a good volumetric representation that effectively tackles noisy point cloud datasets and is more robust for learning. Moreover, we further introduce group equivariant CNN to 3D, by defining the convolution operator on a symmetry group acting on  $\mathbb{Z}^3$  and its isomorphic sets. This improves the expressive capacity without increasing parameters, leading to more robust segmentation results. We highlight the performance on standard benchmarks and show that our approach outperforms state-of-the-art segmentation algorithms on the ShapeNet and S3DIS datasets.

## 1. Introduction

3D data processing including classification and segmentation flourishes these days as 3D data can be easily captured using 3D scanners or depth cameras. It is eminent to deal with irregular and unordered data formats such as the point cloud. The processing pipeline must also be robust towards rotation, scaling, translation and permutation on input data as mentioned in [3]. However, previous work fails to capture the internal symmetry within point clouds. We address these issues in this paper by proposing a novel represen-

tation that considers both spatial distribution of points and group symmetry in a unified framework.

In this paper, we address the problem of developing more effective learning methods using regular data structures such as voxel-based representations, to retain and exploit spatial distributions. Typically, each voxel only contains the Boolean occupancy status (i.e. occupied or unoccupied), rather than other detailed point distributions and therefore can only capture limited details. We address this problem by investigating alternative representations, which can effectively encode the distribution of points in a voxel.

**Main Results:** We present a novel learning method for point cloud segmentation. The key idea is to effectively encode point distributions within each voxel. Directly treating the point distribution as a 0-1 signal is highly non-smooth, and cannot be compactly represented as per Mairhuber-Curtis theorem [26]. We instead transform an unstructured point cloud to a voxel grid. Moreover, each voxel is further subdivided into subvoxels that interpolate sparse point samples within the voxel by smooth Radial Basis Functions, which are symmetric around point samples as centers and positive definite. This smooth signal can then be effectively compacted, and to achieve this we train a variational autoencoder (VAE) [11] to map the point distribution within each voxel to a compact latent space. Our combination of RBF and VAE provides an effective approach to representing point distributions within voxels for deep learning.

A key issue with 3D representations is to ensure that the result of point cloud segmentation does not change due to any rotations, scaling or translation with respect to an external coordinate system. In order to capture the intrinsic symmetry of a point cloud, we use group equivariant convolutions [5] and combine the per point feature extracted by an *mlp* function similar to [3]. These group convolutions were originally proposed for 2D images and we generalize them on  $\mathbb{Z}^3$  and its isomorphic sets for 3D point cloud processing. They help detect the co-occurrence in the feature

\*Corresponding Author

space, namely the latent space of our pre-trained RBF-VAE network of voxels, and thereby improve the learning capability of our approach.

Overall, we present VV-Net, a novel Voxel VAE network with group convolutions, and apply this to point cloud segmentation. Our approach is useful for segmenting objects into parts and 3D scenes into individual semantic objects. We have evaluated and compared its performance on standard point-cloud datasets including **ShapeNet** [29] and **S3DIS** [1]. In practice, our method outperforms the state-of-the-art methods on these datasets by 2.7% and 16.12% in terms of mean IoU (intersection over union), respectively. Even when some of the ground truth data from the point cloud is labeled incorrectly, our approach is also able to compute a meaningful segmentation, as shown in Figure 4. The novel contributions of our work include:

- We develop a novel information-rich voxel-based representation for point cloud data. Point distribution within each voxel is captured using a variational auto-encoder taking RBF at the subvoxel level as input. This provides both the benefits of regular structure and capturing the detailed distribution for learning algorithms.
- We introduce group convolutions defined on the 3-dimensional data, which encode the symmetry and increase the expressive capacity of the network without increasing the number of parameters.

## 2. Related Work

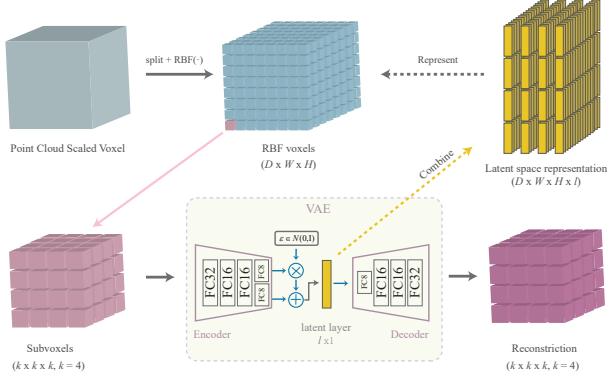
There have been growing interests in 3D data processing algorithms. In this section, we give a brief overview of prior work on point cloud processing and semantic segmentation.

**Deep learning on 3D data.** The point cloud is a very general representation for 3D data. A lot of pioneering research works with deep learning technologies are proposed. PointNet [3] applies multi-layer perceptrons to each point in the input point cloud and symmetric operations to eliminate the permutation problem. Furthermore, PointNet is robust to rotations on the input point cloud by explicitly adding a transform net to align the input point cloud. In the 3D object classification and semantic segmentation tasks, PointNet is regarded as a state-of-the-art approach. Yi et al. [28] cluster 3D shapes by their labels in the dataset and then train a model for hierarchical segmentation. Wang et al. [23] present a similarity matrix that measures the similarity between each pair of points in the embedded space to produce the semantic segmentation map. To capture information at different scales, a commonly used approach is to capture the hierarchical information by recursive sampling or recursively applying neural network structures [17]. In particular, the work [9] applies recurrent neural networks to combine slice pooling layers, and the work [20] uses sparse bi-

lateral convolutional layers as building blocks. Some methods work on 3D meshes, and strive to extract information from graph structures generated from a mesh representation. Yu et al. [30] use a spectral CNN method that enables weight sharing by parameterizing kernels in the spectral domain spanned by graph Laplacian eigenbases. Verma et al. [22] use graph convolutions proposed in [2] to design a graph-convolution operator, which aims to establish correspondences between filter weights and graph neighborhoods with arbitrary connectivity. Deep learning based on variational autoencoders is also employed in [21, 8] for mesh generation.

**Point cloud processing using neighborhood mining.** To address lack of connectivity, some methods use K-nearest neighbors in the Euclidean space and exploit information within local regions [24, 14, 13, 19]. In particular, Li et al. [13] model the spatial distribution of point clouds by building a *self-organizing map* and applying PointNet [3] to multiple smaller point clouds. Moreover, the works [24, 12, 13, 14] use graph structures and graph Laplacian to capture the local information in the selected neighborhoods and leverage the spatial information [14]. Remil et al. [18] utilize the shape priors which are defined as point-set neighborhoods sampled from shape surfaces. However, there are many issues that make it challenging to mine the neighborhood information: First, topology information is not easy to capture with LiDAR scans, which makes it more challenging to estimate vertex normals. Second, encoding K-nearest neighborhoods in the Euclidean space may in some cases simultaneously encode two points that do not belong to the same object (especially for the circumstance that two objects are close to each other). In our work, we do not explicitly encode the K-nearest neighborhoods in our architecture. Instead, we aim to encode the symmetry information rather than encoding the neighborhood information.

**Point cloud processing using voxels.** Some works use voxels for processing point data (e.g. [23, 31, 15, 16]). These methods apply neural networks on voxelized data, and cannot be applied to raw point clouds directly due to their irregular and unordered data format. However, the resolution is limited by data sparsity and computational costs. For the purpose of 3D detection, Zhou and Tuzel [31] sample a LiDAR point cloud to reduce the computation overhead and irregularity of point distribution using farthest point sampling. In order to further reduce the imbalance of points between voxels, their method only takes into consideration densely populated voxels. It applies the point-wise feature learning function *mlp* on each point and aggregates the features by a symmetric function. In contrast, our method does not perform sampling to eliminate the unbalanced distribution. Instead we use regular voxels along with RBF to improve the learning capabilities.

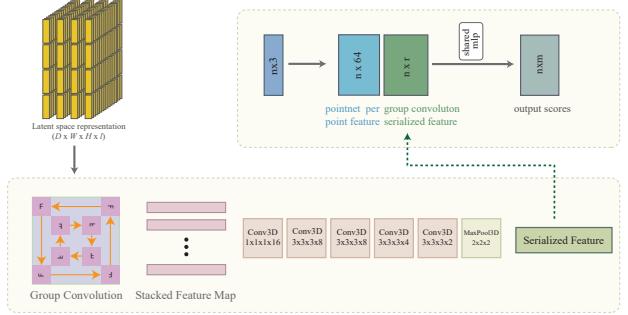


**Figure 1. Radial Basis Function interpolated Variational Auto-Encoder module.** For a given point cloud, we divide it into equally spaced  $D \times H \times W$  voxels, and for each voxel we further divide it into  $k \times k \times k$  subvoxels, where each subvoxel value is defined by the radial basis function in Equation 3 rather than Dirac delta function sampled by *sinc*. The kernel of RBF is set to  $\phi(\|\cdot\|_2^2)$  according to VAE latent distribution. For a voxel with  $k \times k \times k$  subvoxels, we infer the latent space representation using a pre-trained variational auto-encoder. Finally, the point cloud can be presented as a  $D \times H \times W \times l$  voxel data, where  $l$  denotes the dimension of the latent space.

**Convolutions defined on groups, equivariance and transformations.** It is known that the power of CNNs lies in the translation equivariant property, and they exploit translational symmetries by CNN kernel weight sharing [4]. Recently, Cohen and Wellin [5] introduced equivariance to the combinations of  $90^\circ$ -rotations and dihedral flips in CNNs. They extend the theory to a steerable representation which is the composition of elementary feature types although it requires special treatment for anti-aliasing [6]. Cohen et al. [4] further introduce the spherical cross-correlation which satisfies the generalized Fourier transformation although the resulting spherical CNN requires a closed genus-0 manifold as input so that it can be projected as a spherical signal. Similarly, Weiler et al. [25] and Worrall et al. [27] design SO(2) steerable networks, although they are limited by discrete groups and are computationally expensive. All of these methods are either designed for the 2D image domain or the spherical surface domain, and none of them work directly for 3D point data.

### 3. Voxel VAE Net with Group Convolutions

In this section we describe the overall algorithm and highlight the various stages of the pipeline. First, we illustrate the interpolation of multidimensional scattered samples, and show the intuitive motivation of VAE equipped with RBF kernel, which enjoys several advantages: symmetric and positive definite for any choice of data locations. Our formulation computes a better representation with an encoder-decoder scheme, instead of using the stan-



**Figure 2. Segmentation Network Architecture.** We highlight the various components of our approach. The input of the network is a point cloud containing  $n$  points and the latent space representation is illustrated in Figure 1. The output is the per-class score of each point in the point cloud (for  $m$  classes). We use the group convolutional module to detect the co-occurrence in the feature space (see Equation 5). We highlight the group  $p4m$  for functions  $g(m_x, m_y, m_z, r_x, r_y, r_z, t_x, t_y, t_z)$  in Equation 5 in the bottom left figure (where  $m_*$ ,  $r_*$  and  $t_*$  refer to mirroring, rotation and translation). A  $p4m$  function has 128 planar patches in our formulation, where each is associated with a rotation  $r_x, r_y, r_z$  and mirroring  $m_x, m_y, m_z$ . In this figure, we only illustrate 8 planar patches. Each patch follows the arrow and undergoes a  $90^\circ$  rotation. The patches on the outer square are mirror reflection of the patches on the inner square, and vice-versa.

dard  $\{0, 1\}$  voxels (occupancy). Empirically, the distribution of  $\{0, 1\}$  voxels is discrete and insufficient to fully capture point distributions. Moreover, its discontinuous nature makes it difficult to be learned by a deep neural network. Second, we describe our mathematical framework based on group convolutions defined on  $\mathbb{Z}^3$  and their isomorphic sets to detect the co-occurrence of features in the latent space. This increases the expressive capacity of the CNN without increasing the number of parameters and the number of layers. Third, we concatenate the  $n \times 64$  per-point features extracted by the *mlp* function [3] with the serialized features extracted by our network, where  $n$  is the number of points, and 64 is the dimension of features extracted using PointNet. Finally, after *mlp* layers, we output the score map which indicates the probability of a point belonging to the  $m$  classes as in the upper right of Figure 2, where  $m$  is the number of classes in the segmentation task (e.g. 40 in the **ShapeNet** part segmentation task and 13 in the **S3DIS** semantic segmentation task).

#### 3.1. Symbols and Notation

If  $G$  is a group acting on set  $X$ , and  $f, g : X \rightarrow \mathbb{C}$  are actions on group  $G$ , then the convolution is defined as:

$$(f * g)(u) = \int_G f(uv^{-1})g(v)d\mu(v) \quad (1)$$

where  $\mu$  is Haar-measure. In this paper, we have  $X = \mathbb{Z}^3$ , and  $G$  is the group of integer transformation, which is isomorphic to  $\mathbb{Z}^3$ . Note that this is a special case, and  $G$  and

$X$  are usually two different sets.

In our pipeline, the input is a point cloud, represented using 3D coordinates  $(x, y, z)$  in the Euclidean space. We use the symbols  $(\tilde{x}, \tilde{y}, \tilde{z})$  to represent the coordinates of voxel grid. In particular, for a given point cloud with  $n$  points which encompasses 3D space with ranges  $\tilde{D}$ ,  $\tilde{H}$  and  $\tilde{W}$  in the  $Z$ ,  $Y$ ,  $X$  axes, respectively, we divide the entire point cloud into  $D \times H \times W$  voxels. Therefore, the sizes of a voxel in  $Z$ ,  $Y$  and  $X$  directions are:  $v_D = \tilde{D}/D$ ,  $v_H = \tilde{H}/H$  and  $v_W = \tilde{W}/W$ . The output of our RBF-VAE scheme is a  $(D, H, W, l)$ -size matrix, where  $l$  represents the latent space dimension of the encoder-decoder setting. We use the notion of symmetry groups for group equivariant convolutions. Given a group  $G$ , we can define a  $G$ -CNN by analogy to standard CNNs, by similarly defining the function  $G$ -convolution on the group  $G$ .

### 3.2. RBF-VAE Scheme

The traditional voxel representation can be deemed as a 0-1 signal  $f$  sampled at each grid point with spacing  $v_D, v_H, v_W$  along each dimension by Whittaker - Shannon interpolation formula. Applying Fourier transformation to such signal  $f$  involving a combination of Dirac delta functions produces a dense distribution in the frequency domain, forming a Haar-space (Chebyshev space), which cannot be effectively compacted, according to Mairhuber-Curtis theorem [26]. Instead of Boolean occupancy information, we evaluate grid value at  $p$  as a linear combination of radial basis functions:

$$f(p) = \sum_{j=1}^N w_j \phi(\|p - v_j\|_2^2) \quad (2)$$

where  $N$  is the number of data points,  $w_j$  is a scalar value and  $\phi(\cdot)$  is a symmetry function about each data point and is positive definite according to *Bochner theorem*. We measure the point distribution over  $k \times k \times k$  subvoxels by using a variational auto-encoder, leading to an  $l$ -dimensional latent space for each voxel, which is not only compact but also captures the spatial distribution of points. Overall, the voxel representation size for the entire point cloud is  $D \times H \times W \times l$ , which is more detailed than the standard  $D \times H \times W$  volumetric representation.

#### 3.2.1 Radial Basis Functions

To map discrete points to a continuous distribution, we use radial basis functions to estimate their contributions within each subvoxel:

$$f(p) = \max_{v \in V} \left( \exp \frac{-\|p - v\|_2^2}{2\sigma^2} \right). \quad (3)$$

Here  $V$  represents the set of points,  $p$  is the center of the subvoxel, and  $\sigma$  is a pre-defined parameter, usually is a multiple of the subvoxel size. In principle all the points in  $V$

may affect the value of  $f(p)$ , it is the point closest to  $p$  that is dominant. As a result  $f(p)$  can be evaluated efficiently. The formulation here is based on the commonly used Gaussian RBF kernel. Empirically, the kernel used in RBF, i.e.  $\phi(\|\cdot\|_2^2)$  has the same form as VAE latent variable distribution. Furthermore, we show the comparison results of different kernels in Section 4.4.

#### 3.2.2 Variational Auto-Encoder

Our approach uses the approach highlighted in [11] to model the probabilistic encoder and the probabilistic decoder. The encoder aims to map the posterior distribution from datapoint  $X_{(D_i, H_i, W_i)}$  to the latent vector  $Z_{(D_i, H_i, W_i)}$ , where  $(D_i, H_i, W_i)$  represents  $k \times k \times k$  subvoxels and is denoted as  $K_i$ . And the decoder produces a plausible corresponding datapoint  $\hat{X}_{K_i}$  from a latent vector  $Z_{K_i}$ . In our setting, the datapoint  $X_{K_i}$  is represented by RBF kernel subvoxels as formulated in Equation 3. The total loss function of our model can be evaluated as :

$$\begin{aligned} Loss = & \sum_{K_i \in (D, H, W)} E_{Z_{K_i}} [\log P(X_{K_i}^{(i)} | Z_{K_i})] \\ & - D_{KL}(q_\phi(Z_{K_i} | X_{K_i}^{(i)}) || P_\theta(Z_{K_i})) \\ & + D_{KL}(q_\phi(Z_{K_i} | X_{K_i}^{(i)}) || P_\theta(Z_{K_i} | X_{K_i}^{(i)})) \end{aligned} \quad (4)$$

where we sample  $Z_{K_i} | X_{K_i}$  from  $Z_{K_i} | X_{K_i} \sim \mathcal{N}(\mu_{Z_{K_i} | X_{K_i}}, \Sigma_{Z_{K_i} | X_{K_i}})$  and sample  $X_{K_i} | Z_{K_i}$  from  $X_{K_i} | Z_{K_i} \sim \mathcal{N}(\mu_{X_{K_i} | Z_{K_i}}, \Sigma_{X_{K_i} | Z_{K_i}})$ ,  $q_\phi(Z_{K_i} | X_{K_i})$  indicates the encoder network and  $P_\theta(X_{K_i} | Z_{K_i})$  indicates the decoder network. Note that the latent variable  $Z_{K_i}$  only captures the spatial information within a single voxel by the variational auto-encoder scheme. For a pre-trained VAE module, we infer each voxel from the fixed-parameter VAE and compute the final point cloud representation of size  $D \times H \times W \times l$ , where  $l$  is the latent space size of the pre-trained VAE module. The variational auto-encoder captures point data distribution within a voxel in a more compact manner. This not only reduces memory footprint, but also makes our learning algorithm more efficient. The VAE has significantly better generalizability than AE due to the prior distribution assumption, and avoids potential overfitting to the training set.

### 3.3. Symmetry Group and Equivariant Representations

In this section, we present our algorithm to compute the equivariant representations using the symmetry groups. The goal is to build on the VAE based voxel representation and detect the co-occurrence in features with filters in the CNN. The ultimate goal is to enhance the network expressive capacity without increasing the number of layers or the filter sizes in the standard CNN. The work [5] illustrates these issues in the current generation of neural networks, where

Table 1. **ShapeNet experiment settings to test the performance of each module:** Our VAE module is illustrated in Figure 1, and the group convolutional module is highlighted in Figure 2. We present the parameters used for our approach (group-conv + RBF-VAE) and with one module disabled, namely only RBF-VAE without group-conv and group-conv with  $\{0, 1\}$  voxels. The input subvoxel (for VAE-based) or voxel (for non-VAE based) resolutions are fixed to  $64 \times 64 \times 64$ .

Experiment	input of VAE	output of VAE	input of group conv
RBF-VAE	$64 \times 64 \times 64$ RBF voxel	$16 \times 16 \times 16 \times 8$ latent voxel	None
group-conv + $\{0, 1\}$ voxel	None	None	$64 \times 64 \times 64 \{0, 1\}$ voxel
(Our)group-conv + RBF-VAE	$64 \times 64 \times 64$ RBF voxel	$16 \times 16 \times 16 \times 8$ latent voxel	$16 \times 16 \times 16 \times 8$ latent voxel

Table 2. **Results on ShapeNet part segmentation:** We highlight the instance average mIoU and mIoU scores for all the categories on point cloud labeling using prior algorithms and our method. Note that the comparison performances listed below are reported by PointNet [3], RSN [9], SO-Net [13], SyncSpecCNN [30] and SPLATNET [20], respectively. The numbers in bold show the best performances for different object categories. Furthermore, in our experiments, we highlight the results that outperform the state-of-the-art method. The 3 experiments listed in the bottom correspond to the experiment settings in Table 1. Overall, with both the RBF-VAE module and the group convolutional module, our method outperforms the state-of-the-art method by 2.5% in terms of mean IoU. If we replace the RBF-VAE module with the standard  $\{0, 1\}$  voxel VAE module, the training does not converge because the point data is too sparse. Moreover, if we remove the group convolutional module or the RBF-VAE module from our complete pipeline, mIoU would drop by 1.3% or 1.4% respectively. Note that the *Motor* and *Car* categories are challenging as they each contain 4 or more parts. Nonetheless, our method shows significantly better performances.

	Mean IoU	Aero	Bag	Cap	Car	Chair	Ear	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate	Table
PointNet [3]	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
RSN [9]	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
SO-Net [13]	84.6	81.9	83.5	84.8	78.1	<b>90.8</b>	72.2	90.1	83.6	82.3	95.2	69.3	94.2	80.0	51.6	72.1	82.6
SyncSpecCNN [30]	84.74	81.55	81.74	81.94	75.16	90.24	74.88	<b>92.97</b>	86.10	<b>84.65</b>	95.61	66.66	92.73	81.61	60.61	82.86	82.13
SPLATNET-3D [20]	84.6	81.9	83.9	<b>88.6</b>	79.5	90.1	73.5	91.3	84.7	84.5	<b>96.3</b>	69.7	95.0	81.7	59.2	70.4	81.3
RBF-VAE	<b>86.1</b>	82.3	<b>86.6</b>	82.4	<b>81.7</b>	87.7	<b>77.1</b>	91.2	83.7	77.5	94.0	<b>71.0</b>	<b>96.1</b>	<b>86.6</b>	56.1	<b>87.8</b>	<b>89.5</b>
group-conv + $\{0, 1\}$ voxel	<b>86.0</b>	82.1	68.9	83.8	<b>80.9</b>	87.8	<b>81.2</b>	91.2	78.4	77.4	94.5	<b>72.8</b>	<b>98.0</b>	<b>86.0</b>	53.8	<b>83.9</b>	<b>90.0</b>
group-conv + RBF-VAE	<b>87.4</b>	<b>84.2</b>	<b>90.2</b>	72.4	<b>83.9</b>	88.7	<b>75.7</b>	92.6	<b>87.2</b>	79.8	94.9	<b>73.4</b>	94.4	<b>86.4</b>	<b>65.2</b>	<b>87.2</b>	<b>90.4</b>

the representation spaces have minimal internal structure. To address this issue, we use symmetry groups and equivariance CNN to perform efficient data processing. In this case, *G-CNN* is defined in the linear *G*-space, where each vector in the *G*-space has a *pose*, and can be transformed by an element from a group of transformations *G*. Particularly, *G-convolution* corresponds to an operation that helps a filter in *G-CNN* to detect the co-occurrence in features. The transformation in *G-space* is structure preserving. We extend the formulation of *G-space* presented in [5] which was defined for 2D images to 3D. In particular, we define and use *p4* and *p4m* as symmetry groups on  $\mathbb{Z}^3$ . Furthermore, we show the group equivariant convolution on  $\mathbb{Z}^3$  and the underlying CNN is a function on the group. When we apply  $90^\circ$  rotations on a function on *p4*, the simplified results of this operation are shown in Figure 2.

### 3.3.1 Group *p4*

The group *p4* is comprised of all compositions of translations and rotations by  $90^\circ$  about any center of rotation in a 3D grid. We can parameterize the group *p4* in terms of  $r_x, r_y, r_z, t_x, t_y, t_z$  where  $r_*$  and  $t_*$  are rotations and translations w.r.t. axis  $*$ , respectively. Here  $*$  refers to either *X*, *Y* or *Z*. This can be formulated as  $g(r_x, r_y, r_z, T) = R_x \times R_y \times R_z \times T$ , where  $R_*$  is the rotation matrix which rotates around the axis  $*$  by  $\frac{\pi \cdot r_*}{2}$ , and  $T$  is the translation matrix which translates along the *X*, *Y*, *Z* axes by  $t_x, t_y, t_z$ , respectively. Here,  $0 \leq r_x \leq 4$ ,  $0 \leq r_y \leq 4$ ,  $0 \leq r_z \leq 4$  and  $(t_x, t_y, t_z) \in \mathbb{Z}^3$ . The group operation is performed

using matrix multiplication. As mentioned above, the composition of two functions and the inverse function can be easily formulated in terms of  $(r_x, r_y, r_z, t_x, t_y, t_z)$ , hence the operation defines a symmetry group. The group *p4* acts on points in  $\mathbb{Z}^3$  (voxel coordinates) by multiplying the matrix  $g(r_x, r_y, r_z, t_x, t_y, t_z)$  by homogeneous coordinates of a point.

### 3.3.2 Group *p4m*

Here, we extend the group *p4* and construct a symmetry group *p4m* defined on  $\mathbb{Z}^3$ , which also includes mirroring (reflection) along axis aligned planes. More formally, we have the following lemma:

**Lemma 3.1** *The group *p4m* is comprised of all compositions of transformations, rotations by  $90^\circ$  about any center of rotation in the grid, and mirror reflections (i.e. *p4* plus mirroring). As the group *p4* formulated above, we can parameterize the group *p4m* in terms of integers  $(m_x, m_y, m_z, r_x, r_y, r_z, t_x, t_y, t_z)$  as  $R_{mx} \times R_{my} \times R_{mz} \times T$ , where  $R_{mx}$  is formulated as below:*

$$R_{mx} = \begin{bmatrix} (-1)^{m_x} \cos(r_x \frac{\pi}{2}) & -(-1)^{m_x} \sin(r_x \frac{\pi}{2}) & 0 & 0 \\ \sin(r_x \frac{\pi}{2}) & \cos(r_x \frac{\pi}{2}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (5)$$

and  $m_*$  indicates mirroring,  $m_x \in \{0, 1\}$ ,  $m_y \in \{0, 1\}$ ,  $m_z \in \{0, 1\}$ ,  $0 \leq r_x \leq 4$ ,  $0 \leq r_y \leq 4$ ,  $0 \leq r_z \leq 4$  and  $(t_x, t_y, t_z) \in \mathbb{Z}^3$ . The group *p4m* is a symmetry group.

**Table 3. Results of Semantic Segmentation on the S3DIS Dataset.** Our underlying metric is Intersection over Union (IoU) calculated on the points, evaluated on the benchmark [1]. One metric is different between Table 3 and Table 4, IoU in Table 3 and  $AP_{0.5}$  in Table 4, following the practice of existing papers. We report both metrics while most previous works choose to report one or the other. The numbers in bold face fonts indicate the best performances and we highlight the numbers in our experiments if the results outperform the state-of-the-art methods. Notice that the full pipeline (last experiment) outperforms only using RBF-VAE by 1.8% and only using group-conv by 6.33%. Note that the performances of PointNet [3], Engelmann [7] and SPG [12] are reported in [12]. The RSN [9] performance is reported in their paper.

	overall ACC	Mean IoU	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
PointNet ([3])	78.5	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	42.0	54.1	38.2	9.6	29.4	35.2
Engelmann ([7])	81.1	49.7	90.3	92.1	67.9	44.7	24.2	52.3	51.2	47.4	58.1	39.0	6.9	30.0	41.9
SPG ([12])	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
RSN ([9])	59.42	51.93	<b>93.34</b>	<b>98.36</b>	<b>79.18</b>	0.00	15.75	45.37	50.10	65.52	67.87	22.45	52.45	41.02	43.64
RBF-VAE	<b>85.98</b>	<b>75.40</b>	85.01	95.52	71.58	<b>73.81</b>	<b>60.91</b>	<b>61.54</b>	<b>74.38</b>	65.67	67.59	61.47	26.11	38.72	<b>56.16</b>
group-conv + {0,1} voxel	81.45	<b>68.70</b>	83.27	93.95	59.37	<b>64.35</b>	40.23	54.06	66.48	65.20	63.52	41.48	20.37	16.21	47.41
group-conv + RBF-VAE	<b>87.78</b>	<b>78.22</b>	87.64	95.36	74.80	<b>75.04</b>	<b>68.03</b>	<b>71.33</b>	<b>76.87</b>	72.67	<b>70.08</b>	61.97	33.56	<b>49.81</b>	<b>60.00</b>

**Table 4. Results of Semantic Segmentation on the S3DIS Dataset with  $AP_{0.5}$ .** The metric is average precision ( $AP(\%)$ ) with IoU threshold 0.5. Note that the complete pipeline (group-conv+RBF-VAE) achieves the best performance, outperforming both state-of-the-art work and with one of our modules disabled. The result of Armeni [1] is for 3D object detection and IoU is calculated on 3D bounding boxes, while SGPN and ours are based on point cloud datasets. Note that the comparison performances listed below are reported in SGPN [23].

	Mean IoU( $AP_{0.5}$ )	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board
Armeni ([1])	49.93	71.61	88.70	72.86	66.67	<b>91.77</b>	25.92	54.11	16.15	46.02	54.71	6.78	3.91
SGPN ([23])	54.35	79.44	66.29	<b>88.77</b>	77.98	60.71	66.62	56.75	40.77	46.90	47.61	6.38	11.05
RBF-VAE	<b>79.00</b>	<b>88.73</b>	<b>97.43</b>	77.20	<b>79.91</b>	67.27	62.39	<b>81.36</b>	<b>67.08</b>	<b>74.68</b>	<b>55.38</b>	<b>37.08</b>	<b>33.05</b>
group-conv + {0,1} voxel	<b>72.66</b>	<b>88.98</b>	<b>95.32</b>	64.13	67.74	49.21	55.35	<b>74.02</b>	<b>64.34</b>	<b>68.38</b>	29.11	<b>22.58</b>	13.33
group-conv + RBF-VAE	<b>82.17</b>	<b>91.68</b>	<b>96.54</b>	80.38	<b>80.68</b>	71.87	<b>72.94</b>	<b>85.81</b>	<b>73.86</b>	<b>76.76</b>	<b>57.68</b>	<b>43.82</b>	<b>46.35</b>

As illustrated in the bottom left of Figure 2, there are 128 3D patches that undergo rotation and translation transformations. The rich transformation structure arises from the group operation  $p4m$ . Our group operation holds the property of a symmetry group. For implementation, the group convolution with  $90^\circ$ -rotations is employed by copying the transformed filters with different rotation-flip combinations ( $R_{mx} \times R_{my} \times R_{mz}$ ). For  $R_{mx}$  we have  $4 \times 2$  combinations (4 choices for  $90^\circ$  rotation, and whether reflection is applied, along the  $X$  axis). As illustrated in Figure 2, patches are stacked to form a 5D tensor ( $B \times (D - K_D) \times (H - K_H) \times (W - K_W) \times (P \cdot C)$ ), where  $B$  represents batch size,  $D, H, W$  are the voxel sizes along  $X, Y, Z$  axes mentioned in Section 3.2.  $K = (K_D, K_H, K_W, K_{C_{in}}, K_{C_{out}})$  is the kernel size used in the 3D CNN,  $P$  is the total patch number, and  $C_{in}$  and  $C_{out}$  are the numbers of input and output channels of the 3D CNN. We also developed an efficient approach to reducing the memory footprint, where 3D rotation-flip combinations are constructed based on applying 2D rotation-flip operations along arbitrary axes.

## 4. Implementation and Performance

Our network architecture is shown in Figure 2. The RBF-VAE module and the segmentation module are trained separately and the RBF-VAE module is trained firstly. There are two reasons of training separately: the loss of RBF-VAE module is voxel-wise and thereby the whole network could benefit little from it; the memory consumption is saved to 1/8 of its joint training size to make it perform

on a typical Nvidia 1080Ti GPU. Our network is trained with 100 epochs and batch size 24. The inference time of our network is about 210 ms per frame on the S3DIS dataset. We have evaluated our segmentation method VV-Net on two datasets: **ShapeNet** [29] and **S3DIS** [1], respectively. Moreover, we demonstrate the effectiveness of each module used in our approach. First, we highlight the performance difference by alternating between the standard {0,1} voxel VAE module and our novel RBF voxel VAE module. Second, we evaluate the expressive capacity of the group convolutional network module. All these results and comparisons are highlighted in Table 2, Table 3 and Table 4 with the parameter settings of our approach for part segmentation given in Table 1. Our code repository is released on Github.

### 4.1. Part segmentation

Part segmentation is a challenging 3D analysis task, which aims to segment a given 3D scan into meaningful segments. We evaluate our algorithm and highlight the performance in Table 2 on a large-scale **ShapeNet** dataset, which contains 16,881 shapes from 16 categories, and annotated with 50 parts in total. Some examples of the results of our approach are shown in Figure 3. Figure 4(top) demonstrates the ground truth of the dataset, and we can notice that each category is labeled with two to five parts. As described in [3], we also formulate our problem as per-point multi-label classification. The loss function is cross entropy function defined as below:

$$Loss = -\sum_l^L g_l \log p_l, \quad (6)$$

where  $L$  is the number of labels,  $g$  is the probability of ground truth label and  $p$  is the probability of each label. The evaluation metric is mIoU (mean IoU) on points, following the formula in [3]: if the union of groundtruth and prediction points is empty, then we count the corresponding label IoU as 1, since we have 50 parts and 16 shape categories, we compute the category IoU as the average instance IoU on the category.

In our experiment,  $(D, H, W) = (16, 16, 16)$ ,  $k = 4$ ,  $\sigma = \min(v_W, v_H, v_D)$  and  $l = 8$ , where we capture the  $4 \times 4 \times 4$  subvoxels with 8 latent variables inferred from the variational auto-encoder. We highlight the performance of various combinations of different modules. The results corresponding to *group-conv + RBF-VAE* highlights VV-Net’s performance based on combining RBF kernel with VAE scheme and the group convolutional neural network module. This version of our algorithm outperforms state-of-the-art RSN [9] by 2.5% (mIoU) and it is better than RSN in 12 out of 16 categories.

In order to demonstrate the benefits of individual components, we perform an ablation study. For fair comparison, the same  $64 \times 64 \times 64$  resolution of subvoxels (for VAE-based) or voxels (for non-VAE based) is used. The implementation of *group-conv+RBF-VAE* (our method) outperforms only using *RBF-VAE* by 1.3% (mIoU) and is better in 11 out of 16 categories. Our method also outperforms only using *group-conv* by 1.4% (mIoU) and is better in 13 out of 16 categories. We also compare *RBF-VAE* with VAE on the  $\{0, 1\}$  occupancy grid. Since the point data is sparse, training on the  $\{0, 1\}$  VAE does not converge. This shows the necessity and benefits of *RBF-VAE*.

## 4.2. Semantic segmentation of scenes

We also evaluate the performance on Stanford 3D semantic parsing dataset [1], which consists of 6 types of benchmarks. Each point in the data scan is annotated with one of the semantic labels from 13 categories. In our experiment,  $(D, H, W) = (16, 16, 32)$ ,  $k = 4$ ,  $\sigma = 5 \cdot \min(v_W, v_H, v_D)$  and  $l = 8$ . Table 3 highlights the results (category IoU, overall accuracy and mean IoU) of semantic segmentation on the **S3DIS** dataset. Furthermore, Table 4 indicates the results of AP (average precision) metric with IoU threshold 0.5. Our implementation of *group-conv + RBF-VAE* outperforms state-of-the-art SPG [12] by 16.12% of *Mean IoU* metric. Our method (*group-conv + RBF-VAE*) also achieves better performance than either only using *group-conv* or only using *RBF-VAE*, as reported in the bottom rows of Tables 3 and 4. Table 5 compares our method with methods reporting mean IoU and also shows the superior performance of our method.

	Mean IoU
PointCNN [14]	62.74
PointSIFT [10]	70.23
Ours	78.22

Table 5. **Results on Semantic Segmentation in S3DIS Dataset.** We compare the results with [14] and [10] using the mean IoU metric.



Figure 3. **Part Segmentation Results on ShapeNet.** Note that *Car* and *Motor* have lower performance than most other categories in Table 2. This is partly because there are more parts in these categories: 4 labels for *Car* and 5 labels for *Motor*.

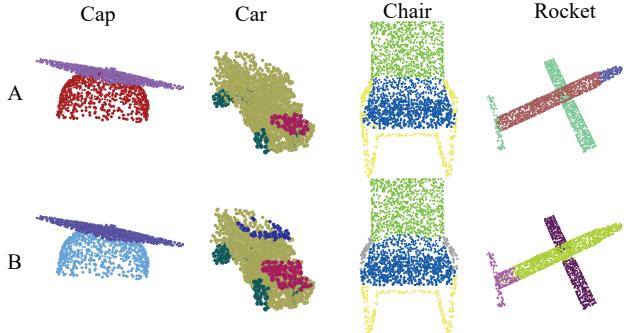


Figure 4. **Failure Cases of Our Algorithm on ShapeNet Part Segmentation.** The top row shows the ground truth and the bottom row is our segmentation results. Our network predicts the *Cap* to be a *Table*, where dark blue indicates the table top and the light blue indicates the table legs. In the second column, the dark blue indicates the top of the car. In the third column, our network segments the chair armrest while the ground truth does not. In the last column, our network predicts the *Rocket* to be an *Airplane*. Notice that in the last column, even a person would find it difficult to distinguish the *Rocket* from the *Airplane*.

## 4.3. Robustness test

We have also evaluated the performance and robustness of our method by removing some of the points in the original data. In particular, we sample the **ShapeNet** by far-

Missing Data Ratio	Accuracy
0%	92.47
75%	92.48
87.5%	91.70

**Table 6. Robustness Test on ShapeNet Part Segmentation Task.** In this evaluation, the point clouds are sampled by farthest point sampling. We test the robustness of our VV-Net network towards missing points. We report the mean accuracy for different missing data ratios. Our approach only has 0.77% accuracy loss, even missing 87.5% of the point cloud data.

	Overall Acc	mean IoU
Gaussian	87.78	78.22
inverse quadratic	78.82	65.04

**Table 7. RBF Kernel Function Comparison on S3DIS Semantic Segmentation Task.** We compare the Gaussian kernel with the inverse quadratic function.

test point sampling and use different missing data ratios. We evaluate the performance and accuracy of the resulting datasets. Table 6 shows the result of our robustness test. This indicates that our approach is not sensitive to missing samples.

#### 4.4. Comparison of Different RBF Kernels

Our RBF function is used to map the distance to each point to its influence. We compare the Gaussian kernel in our method with the inverse quadratic function kernel. With this kernel, the subvoxel function value at position  $p$  is defined as:

$$f(p) = \max_{v \in V} \left( \frac{1}{1 + \sigma^2 \cdot \|p - v\|_2^2} \right). \quad (7)$$

Here  $V$  represents the set of points,  $p$  is the center of the subvoxel, and  $\sigma$  is a pre-defined parameter, usually a multiple of the subvoxel size. The results are shown in Table 7 where using Gaussian kernel achieves better performance.

#### 4.5. Ablation study

Table 8 shows ablation study results on the S3DIS dataset. First, when replacing the *G-CNN* with a traditional CNN, the mean IoU decreases by 7.79%, which indicates that symmetry information is useful. We further verify the usefulness of RBF-VAE. We found that without RBF,  $\{0,1\}$ -VAE often fails to produce reasonable results. This is because point clouds are sparse in 3D space. For example, in S3DIS, each point cloud contains 4096 points. Over  $64 \times 64 \times 128$  subvoxels, the average point density per subvoxel is only 0.008. In this example, Our original grid size is  $(64, 64, 128)$  and the input  $\{0, 1\}$  volume is of size  $16MB$ . With the RBF-VAE, the input size is reduced to  $2MB$ . Compared with taking  $\{0, 1\}$  subvoxels as input, our RGB-VAE scheme significantly reduces the memory consumption and computational cost. As shown in the table, it further helps improve the performance.

**Table 8. Ablation study on S3DIS dataset.** First row: original results; Second row: replacing G-CNN with traditional CNN; Third row: replacing RBF-VAE with RBF grids; Fourth row: replacing RBF-VAE voxels with  $\{0,1\}$  grids. Note that the VAE latent variable distribution is designed for incorporation with RBF. We also considered directly applying G-CNN on RBF subvoxels, but that was not useful due to the compact representation of VAE encoding and lowers the performance.

	Overall Acc	mean IoU	mean IoU threshold 0.5
Ori.(G-CNN + $16 \times 16 \times 32$ RBF-VAE)	85.98	75.40	79.00
Trad. CNN + $16 \times 16 \times 32$ RBF-VAE	80.67	67.61	71.43
G-CNN + $32 \times 32 \times 64$ RBF	78.15	64.13	68.11
G-CNN + $64 \times 64 \times 128$ finer grid	82.36	70.00	74.14

**Table 9. Ablation Study on VAE.** First row: our original results; second row: The VAE function is replaced with AE function. The same parameter settings are used ( $l = 8, k = 4$ ). We observe better accuracy with our original VAE.

	Overall Acc.	mean IoU	mean IoU threshold 0.5
VAE (original algorithm)	85.98	75.40	79.00
RBF-AE+GCNN (modified algorithm)	82.07	69.60	73.38

We replaced our VAE with AE and highlight the performance of this modified approach on the S3DIS dataset in Table 9. The average reconstruction losses for both AE and VAE are close on the *training* set, while average reconstruction loss for AE is about  $2.2 \times$  higher than that for VAE on the *test* set. The VAE has significantly better generalizability than AE due to the prior distribution assumption and avoids potential overfitting to the training set.

## 5. Conclusions, Limitations and Future Work

In this paper we introduced a novel Voxel VAE network (VV-Net) for robust point segmentation. Our approach uses a radial basis function based variational auto-encoder and combines it with group convolutions. We have compared its performance with state-of-the-art point segmentation algorithms and demonstrate improved accuracy and robustness on well-known datasets. While we observe improved performance in most categories, occasionally our approach may not perform well for some input shapes. As in Figure 4, the network suggests that the *Cap* is a *Table*, which may be caused by the group convolutional module because the module encodes  $90^\circ$  symmetry. As future work, we would like to further improve the accuracy and evaluate the performance on other complex point cloud datasets. The VV-Net architecture can also be used for other point cloud processing tasks such as normal estimation, which we will investigate in future.

## Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 61828204 and No. 61872440), Beijing Natural Science Foundation (No. L182016), CCF-Tencent Open Fund, Youth Innovation Promotion Association CAS and NVIDIA Corporation with the GPU donation.

## References

- [1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1534–1543, 2016.
- [2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.
- [3] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [4] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *CoRR*, abs/1801.10130, 2018.
- [5] Taco S. Cohen and Max Welling. Group equivariant convolutional networks, 2016.
- [6] Taco S. Cohen and Max Welling. Steerable CNNs. *arXiv preprint arXiv:1612.08498*, 2016.
- [7] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. Exploring spatial context for 3D semantic segmentation of point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 716–724, 2017.
- [8] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. SDM-NET: Deep generative network for structured deformable mesh. *arXiv preprint arXiv:1908.04520*, 2019.
- [9] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3D segmentation of point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2626–2635, 2018.
- [10] Mingyang Jiang, Yiran Wu, and Cewu Lu. PointSIFT: A SIFT-like network module for 3D point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018.
- [11] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.
- [12] Loïc Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. *CoRR*, abs/1711.09869, 2017.
- [13] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. SO-Net: Self-organizing network for point cloud analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9397–9406, 2018.
- [14] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn. *CoRR*, abs/1801.07791, 2018.
- [15] Daniel Maturana and Sebastian Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.
- [16] Charles R. Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *CVPR*, pages 5648–5656, 2016.
- [17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.
- [18] Oussama Remil, Qian Xie, Xingyu Xie, Kai Xu, and Jun Wang. Data-driven sparse priors of 3D shapes. In *Computer Graphics Forum*, pages 63–72, 2017.
- [19] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 4, 2018.
- [20] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse lattice networks for point cloud processing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.
- [21] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3D mesh models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [22] Nitika Verma, Edmond Boyer, and Jakob Verbeek. FeaStNet: Feature-steered graph convolutions for 3D shape analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [23] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. SGPN: Similarity group proposal network for 3D point cloud instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2569–2578, 2018.
- [24] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.
- [25] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. *arXiv preprint arXiv:1711.07289*, 2017.
- [26] Holger Wendland. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004.
- [27] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhametov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.
- [28] Li Yi, Leonidas Guibas, Aaron Hertzmann, Vladimir G. Kim, Hao Su, and Ersin Yumer. Learning hierarchical shape segmentation and labeling from online repositories. *ACM Transactions on Graphics*, 36(4):70, 2017.
- [29] Li Yi, Vladimir G. Kim, Duygu Ceylan, I. Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, Leonidas Guibas, et al. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics*, 35(6):210, 2016.
- [30] Li Yi, Hao Su, Xingwen Guo, and Leonidas J. Guibas. Syncspeccnn: Synchronized spectral cnn for 3D shape segmentation. In *CVPR*, pages 6584–6592, 2017.
- [31] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.