Modeling Local Geometric Structure of 3D Point Clouds using Geo-CNN

Shiyi Lan[1] Ruichi Yu[1] Gang Yu[2] Larry S. Davis[1]
[1]University of Maryland, College Park [2]Megvii Inc (Face++)
sylan@cs.umd.edu, {yrcbsg, lsd}@umiacs.umd.edu, yugang@megvii.com

Abstract

Recent advances in deep convolutional neural networks (CNNs) have motivated researchers to adapt CNNs to directly model points in 3D point clouds. Modeling local structure has been proven to be important for the success of convolutional architectures, and researchers exploited the modeling of local point sets in the feature extraction hierarchy. However, limited attention has been paid to explicitly model the geometric structure amongst points in a local region. To address this problem, we propose Geo-CNN, which applies a generic convolution-like operation dubbed as GeoConv to each point and its local neighborhood. Local geometric relationships among points are captured when extracting edge features between the center and its neighboring points. We first decompose the edge feature extraction process onto three orthogonal bases, and then aggregate the extracted features based on the angles between the edge vector and the bases. This encourages the network to preserve the geometric structure in Euclidean space throughout the feature extraction hierarchy. GeoConv is a generic and efficient operation that can be easily integrated into 3D point cloud analysis pipelines for multiple applications. We evaluate Geo-CNN on ModelNet40 and KITTI and achieve state-of-the-art performance.

1. Introduction

With the development of popular sensors such as RGB-D cameras and LIDAR, 3D point clouds can be easily acquired and directly processed in many computer vision tasks [64, 19, 38, 30, 55, 28, 16, 56, 8, 61]. Although hand-crafted features on point clouds have been utilized for many years, recent breakthroughs came with the development of convolutional neural networks (CNNs) inspiring researchers to adapt insights from 2D image analysis with CNNs to point clouds.

One intuitive idea is to convert irregular point clouds into regular 3D grids by voxelization [31, 63, 51, 5], which enables CNN-like operations to be applied. However, volumetric methods suffer from insufficient resolution, due to sparsely-occupied 3D grids and the exponentially increasing memory cost associated with making the grid finer. To learn 3D representation at high resolution, kd-tree and octree based methods hierarchically partition space to exploit input sparsity [24, 39]. But those methods focus more on subdivision of a volume rather than local geometric structure. An important architectural model that directly processes point sets is PointNet [33], which aggregates features from points using a symmetric function. To improve the ability to handle local feature extraction, PointNet++ [35] aggregates features in local regions hierarchically. However, these methods still ignore the geometric structure amongst points by treating points independently in the global or local point sets.

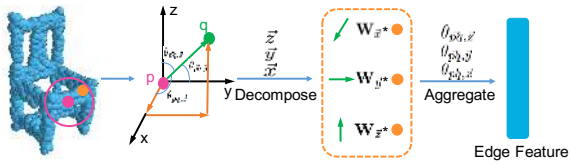

Figure 1. Modeling Geometric Structure between Points via Vector Decomposition. We first decompose the edge features along three orthogonal directions and apply direction-associated weights to extract directional descriptions. Then we aggregate them according to the vector's orientation to construct compact edge features between point p and q.

One recent attempt to model geometric relationships between points is EdgeConv [50], which extracts features from each point and its local k-nearest-neighborhood. EdgeConv extracts *edge features* between a center point and its neighboring points. The geometric structure between two points p and q is represented by the vector $\vec{pq}$. However, EdgeConv only models the distance between points (which is the norm of $\vec{pq}$) when constructing the neighborhood, and it ignores the *direction* of the vector, which leads to loss of local geometric information. Considering that 3D coordinates are given at the input level of most point

cloud analysis pipelines, One might reasonably assume that geometric information might be implicitly learned directly from the coordinates. However, current methods may have the following two challenges for geometric modeling: first, the geometric relationship amongst points may be overwhelmed by the large variance of the 3D coordinates, which makes it difficult to be learned from data directly; second, current methods project the 3D coordinates to some high-dimensional space, which may not preserve the geometric structure of the points in the original Euclidean space, especially when the feature extraction hierarchy is deep.

To address these problems, we propose a novel convolution-like operation *GeoConv* to explicitly model the geometric structure amongst points throughout the hierarchy of feature extraction. GeoConv is applied to each point and its local spherical neighborhood determined by a radius. As shown in Fig.1, the vector $\vec{pq}$ which represents the geometric structure between two points, can be decomposed into three orthogonal bases. Based on this vector decomposition, we project the edge features between the two points into three fixed orthogonal bases (the $\vec{x}, \vec{y}, \vec{z}$ in Fig.1) and apply direction-associated weight matrices ($\mathbf{W}_{\vec{x}}, \mathbf{W}_{\vec{y}}, \mathbf{W}_{\vec{z}}$ in Fig.1) to extract features along each direction; then we aggregate them proportional to the angle between $\vec{pq}$ and the bases ($\theta_{\vec{pq},\vec{x}}, \theta_{\vec{pq},\vec{y}}, \theta_{\vec{pq},\vec{z}}$ in Fig.1). By decomposing the edge feature extraction process into three orthogonal directions, we reduce the variance of the absolute coordinates of the point cloud, and encourage the network to learn edge features along each basis independently; by aggregating the features according to the geometric relationship between the edge vector and the bases, we explicitly model the geometric structure amongst points. Learning in this fashion decomposes the complex geometric structure learning problem into simpler ones while still preserving geometric information. Finally, to extract local features of the center point, we weight the edge features from all points in the local neighborhood based on the norm of $\vec{pq}$. Another advantage of GeoConv is that it enables feature level multi-view augmentation. Our decomposition-aggregation method enables us to approximate the rotation of point clouds at the feature level via re-weighting the features by manipulating the angles.

By stacking multiple layers of GeoConv with increasing size of neighborhoods, we construct Geo-CNN, to hierarchically extract features with increasing receptive fields. We aggregate the features from all points by channel-wise max pooling to maintain permutation invariance. GeoConv is a generic module that models local geometric structure of points. It can be easily integrated into different pipelines for 3D point cloud analysis, *e.g.,* 3D shape classification, segmentation and object detection. We evaluate Geo-CNN on ModelNet40 [51] and KITTI [17] and achieve state-of-the-art performance.

2. Related Work

Motivated by the recent development in 3D sensor technology, increasing attention has been drawn to developing efficient and effective representations on 3D point clouds for shape classification, shape synthesis and modeling, indoor navigation, 3D object detection, *etc.*[47, 53, 54, 44, 22, 57, 46, 40, 1, 25, 52, 37, 9, 12]. Some earlier works constructed hand-crafted feature descriptors to capture local geometric structure and model local similarity between shapes [2, 6, 3, 20, 42, 41]. More recently, deep neural networks have been used to learn representations directly from data. One intuitive way to model the unstructured geometric data is voxelization, which represents a point cloud as a regular 3D grid over which 3D ConvNets can be easily applied [63, 34, 51, 31, 5, 4, 10, 29]. However, volumetric methods usually produce 3D grids which are sparsely occupied in the space. Their exponentially growing computational cost associated with making the grid finer limits the resolution in each volumetric grid, and leads to quantization artifacts. Due to its regular structures and scalability compared to uniform grids, some indexing techniques such as kd-tree and octree have also been applied to model point clouds [24, 39], but those methods still focus more on subdivision of a volume rather than modeling local geometric structure.

To directly model each 3D point individually, PointNet [33], PointNet++ [35] and their variations [32, 27] aggregated point features by a symmetric function to construct a global descriptor. Instead of working on individual points, some recent works exploited local structures by constructing a local neighborhood graph and applying convolution-like operations on the edges connecting neighboring pairs of points [50, 11]. However, in contrast to our proposed Geo-CNN, all of the above methods do not explicitly model the geometric structure of 3D points, which is represented by the norm and orientation of the vector between two points. Our proposed GeoConv operation models the geometric structure of points by a decomposition and aggregation method based on vector decomposition, and can be easily integrated into different pipelines for 3D object recognition, segmentation and detection tasks [35, 33, 32, 63].

Instead of modeling the native 3D format of a point cloud, view-based techniques represent a 3D object as a collection of 2D views, which is compatible with standard CNNs used for image analysis tasks [34, 45, 23, 7, 60]. To aggregate information from different orientations of a 3D object, multi-view methods are applied to pool the features extracted from different rendered 2D views, and usually yield better performance than using a single view. Inspired by this, we augment different orientations of the 3D points via approximating rotations of the input point clouds at feature level to further improve the performance of our model.

3. Our Approach

We propose a generic operation *GeoConv* to explicitly model the geometric structure in a local region. By stacking several layers of GeoConv with increasing receptive field, we construct a Geometric-induced Convolution Neural Network (Geo-CNN) to hierarchically extract features that preserve the geometric relationships among points in Euclidean space. We then aggregate the features from each point by channel-wise max-pooling to extract a global feature descriptor of point clouds.

3.1. Hierarchical Feature Extraction with Geo-CNN

With a set of 3D points as input, we exploit local geometric structure by applying a convolutional-like operation (GeoConv) on each point and its local neighborhood. We build the Geo-CNN by stacking multiple GeoConv layers with increasing neighborhood size. We progressively enlarge the receptive field of the convolution and abstract larger and larger local regions, to hierarchically extract features and preserve the geometric structure of points along the hierarchy (as shown in (a) of Fig.2).

Consider a C dimensional point cloud with n points. We denote the feature of point p at the l^{th} layer of Geo-CNN as $\mathbf{X}_{\vec{p}}^{l} \in \mathbb{R}^{C}$. Usually the 3D points at the input level are represented by their 3D coordinates and additional features like appearance, surface normal, *etc.* For each point p, we construct its local neighborhood using a sphere centered at that point with radius r. GeoConv is applied on point p and all points q in the neighborhood $N(\vec{p}, r)$, where $N(\vec{p}, r) = \{\vec{q} \mid \|\vec{p} - \vec{q}\| \leqslant r\}$. The general formula of GeoConv operation applied at the neighborhood of point p at layer $l+1$ is:

$$\begin{aligned}\mathbf{X}_{\vec{p}}^{l+1} =& s(\vec{p}) + \sum_{\vec{q} \in N(\vec{p},r)} h(\vec{p}, \vec{q}, r) \\ =& \mathbf{W}_c \mathbf{X}_{\vec{p}}^{l} + \frac{\sum_{\vec{q} \in N(\vec{p},r)} d(\vec{p}, \vec{q}, r) g(\vec{p}, \vec{q})}{\sum_{\vec{q} \in N(\vec{p},r)} d(\vec{p}, \vec{q}, r)}\end{aligned} \tag{1}$$

where we aggregate features from the center point $\vec{p}$ and the edge features that represent the relationship between the center point and its neighboring points. $\mathbf{W}_c$ is the weight matrix used to extract features from the center point. $g(\vec{p}, \vec{q})$ is the function that models edge features, which will be defined in Section3.2, and we weight the features from different neighboring points according to the distance between point $\vec{p}$ and $\vec{q}$ using $d(\vec{p}, \vec{q}, r)$ as:

$$d(\vec{p}, \vec{q}, r) = (r - \|\vec{p} - \vec{q}\|)^2 \tag{2}$$

$d(\vec{p}, \vec{q}, r)$ satisfies two desired properties: (1) monotonically decreasing with $\|\vec{p} - \vec{q}\|$; (2) as r increases, which means as the receptive field of our operation becomes larger, the difference of the weight function $d(\cdot)$ between points that have similar distance to the center point p will decrease.

After several GeoConv layers, we apply channel-wise max-pooling to aggregate features of each individual point to construct a global feature descriptor of the point cloud. This feature descriptor can be fed into a classifier for 3D shape recognition, segmentation or detection network. GeoConv is a generic operator that can be easily integrated into current 3D point set analysis pipelines to extract local features while preserving geometric structure in Euclidean space.

3.2. GeoConv: Local Geometric Modeling with Basis-based Decomposition and Aggregation

The most important part of the GeoConv operation is the way it models the edge features. A straightforward way would be to apply a neural network or multi-layer perceptron (MLP) to compute its activation against each edge. However, this method could easily suffer overfitting due to the large variance of edge geometry, which is represented by the vector $\vec{pq}$. On the other hand, the above operation may also project the features into some high dimensional space, where the original Euclidean geometric structure among points is not preserved. In 3D Euclidean space, any vector can be represented by its projection on the three orthogonal bases ($\vec{x}, \vec{y}, \vec{z}$), and the projection norm of the vector on each basis represents the "energy" along that direction. So, we decompose the process of edge feature extraction using the three orthogonal bases: we apply direction-associated weight matrices $\mathbf{W}_{\vec{b}}$ to extract edge features along each direction independently. Then, we aggregate the direction-associated features based on the projection of the vector $\vec{pq}$ on each basis to preserve geometric structure. In practice, to differentiate between positive and negative directions of each basis, we consider six bases represented as:

$$\begin{aligned}B = \{&(1,0,0), (-1,0,0), (0,1,0), \\ &(0,-1,0), (0,0,1), (0,0,-1)\}\end{aligned} \tag{3}$$

As shown in (c) of Fig.2, the six bases separate the space into 8 quadrants, and any vector in a specific quadrant can be composed by three bases out of B. Given a neighboring point q, we first localize the quadrant it lies in (we consider a relative coordinate system by setting p as the origin). Then we project the vector $\vec{pq}$ onto the three bases of this quadrant, and compute the angle between $\vec{pq}$ and each basis (shown in Fig.2 (d)). We apply the direction-associated weight matrices represented as $\mathbf{W}_{\vec{b}}$ to extract the component of edge features along each direction, and aggregate them as shown below:

$$g(\vec{p}, \vec{q}) = \sum_{\vec{b} \in B_{\vec{q}}} cos^2(\theta_{\vec{pq},\vec{b}}) \mathbf{W}_{\vec{b}} \mathbf{X}_{\vec{q}}^{l} \tag{4}$$

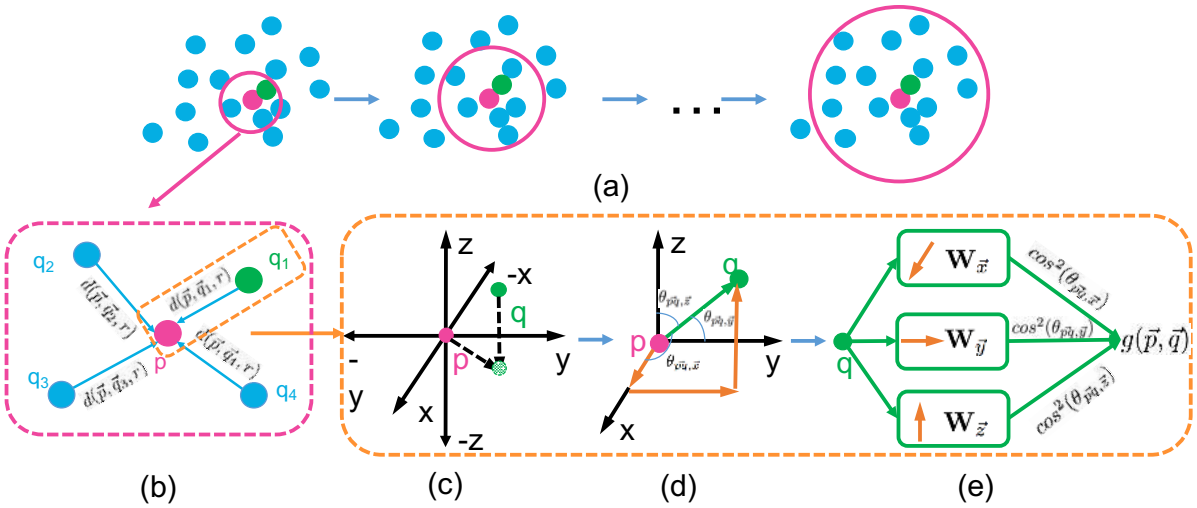

Figure 2. Geo-induced Convolution Neural Network (Geo-CNN). We apply Geo-CNN to hierarchically extract feature representations from a point set. For each point p, GeoConv is applied to its local spherical neighborhood defined by a radius r. We enlarge the receptive field of GeoConv by increasing r at higher levels of the network (shown as the larger circles in (a)). In the local neighborhood of p, we compute the edge features between point p and all neighboring points $q's$, and weight them with a distance measurement function $d(\cdot)$ as shown in (b). To extract the edge features between point p and q, we first localize the quadrant that point q belongs to, in a coordinate system with p as its origin, as illustrated in (c). Then, we compute the edge features along the three bases of that quadrant by direction-associated weight matrices represented as $\mathbf{W}_{\vec{x}}, \mathbf{W}_{\vec{y}}, \mathbf{W}_{\vec{z}}$, and aggregate them according to the angles between vector $\vec{pq}$ and the three bases, shown as $\theta_{\vec{pq},\vec{x}}, \theta_{\vec{pq},\vec{y}}, \theta_{\vec{pq},\vec{z}}$ in (d-e).

where $\mathbf{X}_{\vec{q}}^{l}$ is the feature of point q at the l^{th} layer, and $B_{\vec{q}}$ is a set consisting of three bases selected from B according to the quadrant in which point $\vec{q}$ lies. The feature along each direction is aggregated with the coefficients $cos^2(\theta_{\vec{pq},\vec{b}})$, which corresponds to the square of the ratio between the norm of each projected component of $\vec{pq}$ and the norm of $\vec{pq}$, and they naturally sum to 1.

By modeling edge geometry using the basis-based decomposition, our network learns to extract representations for each direction independently. This reduces the complexity of the learning task, when compared with directly learning from the large variance of input 3D coordinates. By aggregating the features along each basis, we explicitly model geometric structure of the edge vector between each point and its neighbors. By learning geometric modeling using GeoConv, we model and preserve the geometric structure of 3D point clouds at every level of our hierarchical feature extraction framework.

3.3. Approximating 3D Multi-view Augmentation at the Feature Level using Geo-CNN

Inspired by previous works [45, 23] that aggregate information of a 3D object by utilizing rendered 2D images with different virtual camera views, we can also sample from different orientations by rotating 3D points, and then pool the multi-view representations to augment information from different views. In the 3D space, any rotation of the point clouds can be decomposed into the rotation around the $\vec{z}$ axis and around the plane spanned by $\vec{x}$ and $\vec{y}$. For simplicity, "rotation" in this paper refers to rotation around the $\vec{z}$ axis; our analysis can be easily expanded to other cases.

A naive way to incorporate multiple 3D views at training time is to use the rotated point sets as data augmentation, but this method usually leads to even worse performance in our baseline as well as our implementation of some other works (*e.g.,* [35, 33]). A possible reason is that the current methods cannot efficiently learn a compact model from the large variance introduced by multiple 3D views. An alternative is to train a specific model for each 3D view and aggregate the output of multiple networks, which will dramatically increase model complexity.

Instead of input-level multi-view augmentation, we approximate rotation at the feature level in our network using the GeoConv operation. This is done by sharing the computations on edge features along different directions and only changing the aggregation model. Specifically, we approximate multi-view training and testing by manipulating the aggregation step in GeoConv:

$$g_{MV}(\vec{p},\vec{q}) = \sum_{v \in V} w_v \sum_{\vec{b} \in B_{\vec{q}}} cos^2(\theta_{\vec{pq}_v,\vec{b}}) \mathbf{W}_{\vec{b}} \mathbf{X}_{\vec{q}} \tag{5}$$

where w_v are learned weights to fuse multi-view features; $\theta_{\vec{pq}_v,\vec{b}}$ are the re-computed angles between the rotated edge vector and the fixed bases.

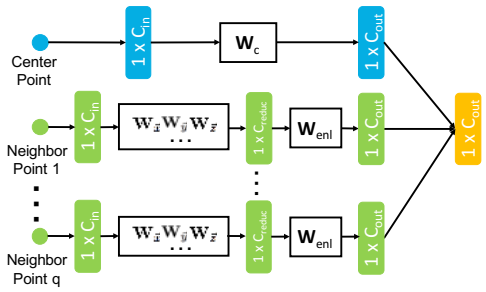

Figure 3. Implementation of GeoConv. The filled boxes show the point features with their dimensionality. The black boxes are the operations. We employ a bottleneck-like structure to first use our decomposition-aggregation method to extract edge features with a lower dimensionality, and then enlarge the dimensionality to match the features extracted from the center point. We aggregate edge features from each point following Eq.(1).

4. Implementation Details

4.1. GeoConv Module

The input/output of a GeoConv layer is $n \times C_{in}$ and $n \times C_{out}$, where n is the number of points, C_{in} and C_{out} are the input/output dimensionality of each point feature. For each point, we construct its local spherical neighborhood defined by a hyper-parameter r. We apply a weight matrix $\mathbf{W}_c$ with size $C_{in} \times C_{out}$ to extract features from the center point. For edge feature extraction, we apply a bottleneck module inspired by ResNet[21] to first extract features with lower-dimension C_{reduc} (we refer to this layer as "*reduction layer*"), and then enlarge their dimensionality as shown in Fig.3. The hyper-parameter of GeoConv is the radius r. In practice, we split the training data into training and validation set and apply cross-validation to choose the radius for each layer.

4.2. Geo-CNN for 3D Shape Classification

For 3D shape classification on ModelNet40 [51], we randomly sample 1,000 points from the 3D model of an object. The input features are the 3D coordinates and the surface normal (6 input channels in total). Geo-CNN has two branches: (1) similar to PointNet++[35], we sample 16 nearest-neighbors from each of the 1,000 points and apply three fully-connected (FC) layers with output dimentionality as 64-128-384 on each group of 16 points. The output size is 1×384 for each of the 1000 point. (2) For the second branch, we feed the same input points into an FC layer to project them into a 64-dim feature space. Then we apply the first GeoConv layer with $C_{in} = 64$, $C_{reduc} = 64$ and $C_{out} = 128$. An FC layer with 256 output channels and a GeoConv layer with $C_{in} = 256$, $C_{reduc} = 64$ and $C_{out} = 512$ follow. At this point, we channel-wisely concatenate the features extracted from the two branches to obtain an 896-dim feature vector for each point. Next, we apply the third GeoConv with $C_{in} = 896$, $C_{reduc} = 64$ and $C_{out} = 768$ followed by the last FC layer with 2048-dim output. Channel-wise max-pooling is then applied to aggregate features from all points. We conduct shape classification on the pooled global feature descriptor. The radius for constructing local neighborhoods for the three GeoConv layers are 0.15, 0.3 and 0.6 (the 3D coordinates are normalized in ModelNet40). Batch-normalization and ReLU are applied after every FC layer and each reduction layer in the GeoConv module.

4.3. Geo-CNN for 3D Object Detection

As a generic feature extraction module, Geo-CNN can be easily applied in any pipeline for point-based 3D object detection. We follow Frustum PointNet V1 pipeline and replace some layers in the segmentation network with GeoConv layers. There are 3 MLP modules in the 3D Instance Segmentation PointNet of Frustum V1 [32] with 9 FC layers in total for feature extraction. We directly replace all of the FC layers with the GeoConv layers. For fair comparison, the output dimensionalities of the GeoConv layers are exactly the same as the replaced FC layers. The radii of the 2 layers in the first MLP block are 0.15-0.2; the radii for the 3 layers in the second block are 0.3-0.4-0.4; for the 4 layers in the third block, the radii are 0.3-0.2-0.1-0.1. We also explored replacing the FC layers in the box estimation module of Frustum PointNet, but obtained slightly worse results. One possible reason is that when comparing with segmentation, bounding box regression depends more on modeling global information of an object, rather than modeling geometric structures of local point sets.

For the 3D object detection pipeline, we construct frustums based on 2D box proposals generated by the object detection pipeline (similar to 2D object detection methods [36, 18, 15, 14]) in [32]. Then, the Point Segmentation Network with GeoConv is applied to categorize the points on the object in each frustum and eliminate the noise caused by background point clouds. Finally, we use the same Box Estimation Network as [32] to obtain the orientation, size and centroid of the 3D bounding box. The implementation of GeoConv is the same as ModelNet.

4.4. Baselines

Our baselines have very similar architecture with Geo-CNN, with two difference in the edge feature extraction process: first, baselines fuse the edge features from different points by simply averaging them, unlike GeoConv which weights the features based on the distance measurement $d(\cdot)$; second, at the reduction layer, GeoConv utilizes three separate weights along each direction, while the baseline applies a single weight matrix to extract edge features.

5. Experiments

We evaluate GeoConv on 3D shape classification using ModelNet40[51] and 3D object detection datasets using KITTI[17]. The improvement made on both datasets shows the ability of GeoConv to model synthetic and real point cloud.

5.1. 3D Shape Classification on CAD-Generated 3D Point Clouds

5.1.1 Dataset

We first evaluate our model on the ModelNet40 [51] data with 3D point clouds generated from CAD. There are 12,311 CAD models from 40 object categories, with 9,843 training and 2,468 testing samples. For fair comparisons with previous works, we use the prepared ModelNet40 dataset from [35], where each model is represented by 10,000 points. One can also sample various sizes of point clouds, e.g., 1000 or 5,000, from the point set.

5.1.2 Comparison with other Methods

Table 1 shows the comparison between our Geo-CNN and prior methods. Geo-CNN achieves state-of-the-art performance on the object classification task with both evaluation metrics of ModelNet40[1]. Our baseline achieves similar performance with the state-of-the-art PointNet++[35]. By simply changing the operation on modeling the edge features in a local point set from a fully-connected layer to GeoConv, we obtain a gain of 1.6%, which demonstrates the effectiveness of our geometrical modeling method. By further approximating 3D multi-view at the feature level, we obtain a further 0.5% performance gain. We implement the multi-view approximation by virtually rotating the point clouds around the z-axis from 0 to 360 degrees. We uniformly approximate 30 views [2] following Eq.(5). Our method, which is applied directly on point clouds, even outperforms single networks with multi-view images, *e.g.,* [34] (92%) and [45](90.1%), and achieves comparable performance with the method integrated multiple networks in [34] (93.8%). However, our single model Geo-CNN with approximated multi-view learning at feature level is more scalable and flexible compare to multi-view representations using multiple networks.

Table 1. ModelNet40 Shape Classification Results. We sort the previous methods by time.

Method	Accuracy Overall	Accuracy Class
PointNet[33]	89.2	86.2
PointNet++[35]	91.9	-
DeepSets[62]	90.3	-
ECC[43]	87.4	83.2
OctNet[39]	86.5	83.8
O-CNN[49]	90.6	-
Kd-Net[24]	91.8	88.5
EdgeConv[50]	92.2	90.2
SO-Net[27]	93.4	90.8
SpiderCNN[53]	92.4	-
SCN[52]	90.0	87.6
MRTNet[13]	91.7	-
SpecGCNN[48]	92.1	-
Our baseline	91.8	88.2
Geo-CNN	93.4	91.1
Geo-CNN+ MV-Approx.	**93.9**	**91.6**

5.2. 3D Object Detection on LIDAR Points

The distribution of real point clouds could vary significantly from generated points. For instance, generated data contains dense points from various orientations while point clouds obtained by sensors, *e.g.,* LIDAR, only contain points from frontal surfaces due to occlusion. Moreover, LIDAR point clouds are noisier and contain a large amount of background, while generated point clouds contain pure on-object points. Evaluation on real data such as point clouds collected by LIDAR is very important to show the robustness and practicality of a 3D point cloud analysis method. To illustrate the effectiveness of Geo-CNN on real-world 3D points, we evaluate on 3D object detection using the KITTI dataset [17].

5.2.1 Dataset

The KITTI 3D object detection benchmark contains 7,481 training and 7,518 testing images/point clouds with three object categories: *Car*, *Pedestrian*, *Cyclist*. For each class, detection outcomes are evaluated based on three difficulty levels: easy, moderate, and hard. The level of difficulty is based on object size, occlusion state, and truncation level. For fair comparison with the state-of-the-art detection methods, we directly replace the PointNet feature extraction module in the Frustum PointNet v1 [32] detection pipeline with Geo-CNN, and use the 2D bounding box proposals released by [32] in our experiments. Since only train/val proposals of frustum pointnet are published, we conduct evaluation using the protocol described in [32, 63] and use their training/testing split.

5.2.2 Comparison with other Methods

Table 5.2 shows the evaluation results on KITTI 3D object detection. Our implementation of the detection pipeline is based on Frustum PointNet v1, which involves object pro-

[1]Since the two metrics are very similar, "performance" on this dataset refers "Accuracy Overall" metric.

[2]The performance gain of multi-view approximation is robust to the number of views from 10 to 40, with less than ± 0.1 changes.

Method	Cars			Pedestrians			Cyclists		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
VoxelNet[63]	81.97	65.46	62.85	57.86	53.42	48.87	67.17	47.65	45.11
Frustum PointNet v1[32]	83.33	69.00	61.97	67.29	56.74	49.84	71.65	53.43	49.20
Frustum PointNet v2[32]	83.42	70.40	63.37	64.30	57.33	50.43	70.51	55.31	52.11
Baseline	84.56	69.16	62.50	64.68	55.59	48.45	72.32	51.36	47.70
Frustum Geo-CNN	**85.09**	**71.02**	**63.38**	**69.64**	**60.50**	**52.88**	**75.64**	**56.25**	**52.54**

Table 2. Performance Comparison in 3D Object Detection: average precision (in %) on KITTI validation set. Geo-CNN achieves significantly better performance when compared with the baseline, which demonstrates the effectiveness of our decomposition-aggregation method on modeling local geometry. Our Frustum Geo-CNN is implemented based on Frustum PointNet v1, and it outperforms both Frustum PointNet v1 and v2.

posals in 2D object detection [36, 18, 59, 26, 58]. The performance of v1 was surpassed by Frustum PointNet v2, which has a more complex architecture. However, by replacing the PointNet feature extraction module in the segmentation network of v1 with GeoConv, Frustum with Geo-CNN outperforms both Frustum PointNet v1 and v2. The performance of Frustum v1 and v2 on the validation set is evaluated based on the released code of [32], and it is very similar with the performance reported in [32]. We visualize the detection results of Frustum with Geo-CNN on 2D and 3D images in Fig.4.

5.3. Ablation Study

Can we model local geometry by directly learning from 3D coordinates? We study different ways to model local geometry between points in the feature extraction hierarchy. Since the geometric structure is encoded in the 3D coordinates of points, one straightforward way to learn geometric structure is to apply an FC layer to directly learn from the coordinates. However, previous hierarchical feature extraction methods project the 3D coordinates input to some high-dimensional feature space at the first layer of their networks, which may lead to the loss of the Euclidean geometry amongst points. Our baseline method takes the 3D coordinates at the input level to directly learn the geometric structure implicitly. To preserve the Euclidean geometry throughout the feature extraction hierarchy, we apply an FC layer to learn the geometric structure between points directly from the 3D coordinates of point p and q at *every layer* of our baseline model, and concatenate the extracted features with the original ones channelwisely. We refer to this method as "Baseline + 3D Coords". We also investigated alternative approaches to model the angle of the vector $\vec{pq}$ in GeoConv. Instead of using $g(\cdot)$ function as proposed, we directly learn these aggregation coefficients using an FC layer with the 3D coordinates of point p and q as input. We refer to this method as "GeoConv - Learned-Agg". As shown in Table 3, directly learning geometric structure between points or the coefficients to aggregate the decomposed features from the 3D coordinates does not help. This reveals that modeling the local geometry is non-trivial, and GeoConv effectively captures geometric structures amongst points to improve the feature extraction framework.

	Accuracy Overall
Baseline	91.8
Baseline + 3D Coords	91.7
GeoConv - Learned-Agg	91.5
GeoConv	**93.4**

Table 3. **Ablation Study: Different Geometric Modeling Methods**. We study different ways to model local geometry amongst points using ModelNet40 dataset. "Baseline + 3D Coords" directly learns the geometric structure with 3D coordinates of the two points at *every layer* of the network; "GeoConv - Learned-Agg" aggregates the direction-associated features by learned weights.

Does the performance gain of GeoConv come from increasing model complexity? By decomposing the edge feature extraction process into three directions using separate neural networks, GeoConv increases the number of parameters in the reduction layer of edge feature extraction. The number of parameters for the edge feature extraction operation is $C_{in} * n_{bases} * C_{reduc} + C_{reduc} * C_{out}$, where C_{in} and C_{out} are the input/output channel numbers. c_{reduc} is the output channel number of the channel reduction step, and the difference between GeoConv and the baseline is n_{bases} ($n_{bases} = 6$ for GeoConv and $n_{bases} = 1$ for the baseline). We increase c_{reduc} from 64-64-64 to 192-192-256 for the three reduction layers in the baseline model to roughly match the number of parameters for edge feature extraction of GeoConv operation. The enlarged baseline is referred to as "Baseline-Large" shown in Table 5.3, which is evaluated on the ModelNet40 classification task. It is worth noting that the number of parameters in the reduction layers accounts for a very small portion of the total number of parameters, and the experiment setting of the other components of the networks except for the reduction layers, are exactly the same for both baseline and Geo-CNN. It is clear that simply enlarging the number of channels does not improve performance, and the gain of GeoConv is not due to the increasing number of parameters.

Figure 4. We visualize the detection results on KITTI with 2D and 3D images. The red boxes are the groundtruth boxes and the blue boxes are the prediction results. Some of the false positive detection results are because of missing annotation.

Method	Baseline	Baseline-Large	GeoConv
Accuracy Overall	91.8	91.7	**93.4**
#. of parameters for edge feature extraction	167.9K	610.8K	557.1K

Table 4. **Ablation Study: Model Complexity**. We add channels to the weight matrix of the reduction layer of the baseline method (Baseline-Large) to match the number of parameters of GeoConv. We show the sum of number of parameters of the 3 reduction layers in each model. The results on ModelNet40 shows that simply increasing model complexity does not help.

Method	Accuracy Overall
Baseline	91.8
Baseline + Data Aug.	91.6
Geo-CNN	93.4
Geo-CNN + Data Aug.	92.6
Geo-CNN + MV-Approx.	**93.9**

Table 5. Overall Accuracy on ModelNet40 with Different Multi-view Augmentations. "Data Aug." and "MV-Approx." refer to input-level augmentation and our feature-level multi-view approximation.

3D Multi-view Augmentation. We evaluate the effect of our feature-level multi-view augmentation. As a straightforward way to incorporate multi-view information to the network learning process, one can simply rotate the input point clouds randomly as data augmentation at training time. On the other hand, our proposed decomposition-aggregation method in GeoConv enables us to approximate 3D multi-view augmentation at the feature level. Table 5.3 shows the performance of input-level multi-view augmentation and feature-level approximation on ModelNet40 dataset. We observe that input-level multi-view data augmentation leads to performance degradation of both the baseline method and Geo-CNN. One possible reason is that the input-level data augmentation brings in large variance between different views, which cannot be properly learned with a single compact model. Another possible solution is to learn separate models with different views and then aggregate them. However, the models with multiple networks are less flexible and scalable due to their high complexity.

6. Conclusion

We address the problem of modeling local geometric structure amongst points with GeoConv operation and a hierarchical feature extraction framework dubbed Geo-CNN. Inspired by the success of exploiting local structure using CNNs on 2D image analysis task, we propose to extract features from each point and its local neighborhood with a convolutional-like operation. GeoConv explicitly models the geometric structure between two points by decomposing the feature extraction process onto three orthogonal directions, and aggregating the features based on the angles between the edge vector and the bases. The Geo-CNN with GeoConv operation achieves state-of-the-art performance on the challenging ModelNet40 and KITTI datasets.

7. Acknowledgement

This research was partially supported by National Key R&D Program of China (No. 2017YFA0700800).

The research was partially supported by the Office of Naval Research under Grant N000141612713: Visual Common Sense Reasoning for Multi-agent Activity Prediction and Recognition.

References

[1] M. Angelina Uy and G. Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[2] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *ICCV Workshops*, pages 1626–1633. IEEE, 2011. 2

[3] R. Beserra Gomes, B. M. Ferreira da Silva, L. K. d. M. Rocha, R. V. Aroca, L. C. P. R. Velho, and L. M. G. Gonçalves. Efficient 3d object recognition using foveated point clouds. *Comput. Graph.*, 37(5):496–508, Aug. 2013. 2

[4] A. Brock, T. Lim, J. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. pages 1–9, 12 2016. Workshop contribution; Neural Inofrmation Processing Conference : 3D Deep Learning, NIPS ; Conference date: 05-12-2016 Through 10-12-2016. 2

[5] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks, 2016. cite arxiv:1608.04236Comment: 9 pages, 5 figures, 2 tables. 1, 2

[6] M. M. Bronstein and I. Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *CVPR*, pages 1704–1711. IEEE Computer Society, 2010. 2

[7] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE CVPR*, 2017. 2

[8] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 1

[9] H. Deng, T. Birdal, and S. Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[10] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *ICRA*, pages 1355–1361. IEEE, 2017. 2

[11] M. J. et al. Pointsift: a sift-like network module for 3d point cloud semantic segmentation. *CoRR*, abs/1807.00652, 2018. 2

[12] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2463–2471, 2017. 2

[13] M. Gadelha, R. Wang, and S. Maji. Multiresolution tree networks for 3d point cloud processing. In *The European Conference on Computer Vision (ECCV)*, September 2018. 6

[14] M. Gao, A. Li, R. Yu, V. I. Morariu, and L. S. Davis. C-wsl: Count-guided weakly supervised localization. *arXiv preprint arXiv:1711.05282*, 2017. 5

[15] M. Gao, R. Yu, A. Li, V. I. Morariu, and L. S. Davis. Dynamic zoom-in network for fast object detection in large images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[16] L. Ge, Z. Ren, and J. Yuan. Point-to-point regression pointnet for 3d hand pose estimation. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1

[17] A. Geiger. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3354–3361, Washington, DC, USA, 2012. IEEE Computer Society. 2, 6

[18] R. Girshick. Fast R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 5, 7

[19] A. Golovinskiy, V. G. Kim, and T. Funkhouser. Shape-based recognition of 3D point clouds in urban environments. *International Conference on Computer Vision (ICCV)*, Sept. 2009. 1

[20] Y. Guo, M. Bennamoun, F. A. Sohel, M. Lu, and J. Wan. 3d object recognition in cluttered scenes with local surface features: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(11):2270–2287, 2014. 2

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016. 5

[22] Q. Huang, W. Wang, and U. Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[23] A. Kanezaki, Y. Matsushita, and Y. Nishida. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. CVPR*, 2018. 2, 4

[24] R. Klokov and V. S. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *ICCV*, 2017. 1, 2, 6

[25] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[26] A. Li, J. Sun, J. Y.-H. Ng, R. Yu, V. I. Morariu, and L. S. Davis. Generating holistic 3d scene abstractions for text-based image retrieval. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 7

[27] J. Li, B. M. Chen, and G. Hee Lee. So-net: Self-organizing network for point cloud analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 6

[28] K. Li, T. Pham, H. Zhan, and I. Reid. Efficient dense point cloud object reconstruction using deformation vector fields. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1

[29] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas. Fpnn: Field probing neural networks for 3d data. *CoRR*, abs/1605.06240, 2016. 2

[30] Y. Liu, C. Wang, Z. Song, and M. Wang. Efficient global point cloud registration by matching rotation invariant features through translation search. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1

[31] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 922 – 928, September 2015. 1, 2

[32] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 5, 6, 7

[33] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 77–85. IEEE Computer Society, 2017. 1, 2, 4, 6

[34] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016. 2, 6

[35] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5099–5108. Curran Associates, Inc., 2017. 1, 2, 4, 5, 6

[36] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015. 5, 7

[37] Z. Ren and E. B. Sudderth. 3d object detection with latent support surfaces. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[38] D. Rethage, J. Wald, J. Sturm, N. Navab, and F. Tombari. Fully-convolutional point networks for large-scale point clouds. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1

[39] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017. 1, 2, 6

[40] R. Roveri, L. Rahmann, C. Oztireli, and M. Gross. A network architecture for point cloud classification via automatic depth images generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[41] R. M. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 225–233, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. 2

[42] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, ICRA'09, pages 1848–1853, Piscataway, NJ, USA, 2009. IEEE Press. 2

[43] M. Simonovsky and N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR*, abs/1704.02901, 2017. 6

[44] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[45] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015. 2, 4, 6

[46] J. Vongkulbhisal, B. Irastorza Ugalde, F. De la Torre, and J. P. Costeira. Inverse composition discriminative optimization for point cloud registration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[47] C. Wang, B. Samari, and K. Siddiqi. Local spectral graph convolution for point set feature learning. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[48] C. Wang, B. Samari, and K. Siddiqi. Local spectral graph convolution for point set feature learning. In *The European Conference on Computer Vision (ECCV)*, September 2018. 6

[49] 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 Trans. Graph.*, 36(4):72:1–72:11, July 2017. 6

[50] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018. 1, 2, 6

[51] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920. IEEE Computer Society, 2015. 1, 2, 5, 6

[52] S. Xie, S. Liu, Z. Chen, and Z. Tu. Attentional shapecontextnet for point cloud recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 6

[53] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2, 6

[54] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[55] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang. 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1

[56] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. Ec-net: an edge-aware point set consolidation network. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1

[57] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. Pu-net: Point cloud upsampling network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[58] R. Yu, X. Chen, V. I. Morariu, and L. S. Davis. The role of context selection in object detection. In *British Machine Vision Conference (BMVC)*, 2016. 7

[59] R. Yu, A. Li, V. I. Morariu, and L. S. Davis. Visual relationship detection with internal and external linguistic knowledge distillation. *IEEE International Conference on Computer Vision (ICCV)*, 2017. 7

[60] T. Yu, J. Meng, and J. Yuan. Multi-view harmonized bilinear network for 3d object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[61] M. E. Yumer and N. J. Mitra. Learning semantic deformation flows with 3d convolutional networks. In *European Conference on Computer Vision (ECCV 2016)*, pages –. Springer, 2016. 1

[62] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3391–3401. Curran Associates, Inc., 2017. 6

[63] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2, 6, 7

[64] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. In *IEEE International Conference on Robotics and Automation*, 2017. 1