Improving the Robustness of Point Convolution on k-Nearest Neighbor Neighborhoods with a Viewpoint-Invariant Coordinate Transform

Xingyi Li, Wenxuan Wu, Xiaoli Z. Fern, and Li Fuxin
School of Electrical Engineering and Computer Science, Oregon State University
{lixin, wuwen, xfern, lif}@oregonstate.edu

Abstract

Recently, there is significant interest in performing convolution over irregularly sampled point clouds. Point clouds are very different from raster images, in that one cannot have a regular sampling grid on point clouds, which makes robustness under irregular neighborhoods an important issue. Especially, the k-nearest neighbor (kNN) neighborhood presents challenges for generalization because the location of the neighbors can be very different between training and testing times. In order to improve the robustness to different neighborhood samplings, this paper proposes a novel viewpoint-invariant coordinate transform as the input to the weight-generating function for point convolution, in addition to the regular 3D coordinates. This allows us to feed the network with non-invariant, scale-invariant and scale+rotation-invariant coordinates simultaneously, so that the network can learn which to include in the convolution function automatically. Empirically, we demonstrate that this effectively improves the performance of point cloud convolutions on the SemanticKITTI and ScanNet datasets, as well as the robustness to significant test-time downsampling, which can substantially change the distance of neighbors in a kNN neighborhood. Experimentally, among pure point-based approaches, we achieve comparable semantic segmentation performance with a comparable point-based convolution framework KPConv on SemanticKITTI and ScanNet, yet is significantly more efficient by virtue of using a kNN neighborhood instead of an ϵ-ball.

1. Introduction

Convolutional neural networks (CNNs) have redefined the state-of-the-art for almost every task in computer vision. In order to transfer such successes from 2D images to the 3D world, there is a significant body of work aiming to develop the convolution operation on 3D point clouds. This is essential to many applications such as autonomous driving and virtual/augmented reality.

One mainstream definition of convolution on point clouds involves discretizing a continuous convolution function on a neighborhood defined by input points[50, 22, 65, 18, 64, 73]. In many point cloud convolution frameworks, a multi-layer perceptron (MLP) is used to learn the convolution weights on each point implicitly. Such an operation is permutation-invariant and translation-invariant, but the initial formulations were memory-intensive due to the need to compute all the convolution weights for all points and all their neighbors. [66] proposed PointConv which decoupled the MLP into two pieces where computations can be shared among multiple convolution kernels, hence greatly reducing the memory requirement and allowing deep CNNs to be built on point clouds. It has one of the best performances as a pure point-based convolutional network in 3D point clouds and can match 2D CNN performance on CIFAR-10 if the image is treated as a (regular) point cloud.

However, so far the best performance in 3D point cloud segmentation benchmarks has still been obtained by fusion algorithms [62] that jointly use point cloud networks with sparse 3D convolution approaches [14, 6], which discretizes the point cloud onto a 3D grid. In principle, both point convolutions and sparse 3D convolutions are convolutions, hence they should have similar behavior and should not require the fusion that complicates the network structure and slows down the inference speed. Hence, we set out to explore the reason underlying the seemingly worse generalization power of the point cloud convolution networks and seek ways to improve it.

One culprit we have identified is the choice of neighborhood in point cloud convolutions. Networks based on point clouds introduce a new complication on the neighborhoods used in convolution. In 2D images, we are accustomed to having fixed-size neighborhoods such as 3×3 or 5×5. PointConv and other point-based networks instead adopt k-nearest neighbors (kNN), which could be significantly more irregular (Fig. 1(a)) and may potentially make it harder for point cloud networks to generalize from training neighborhoods to testing neighborhoods since those could be irregular in very different manners. Usually, point cloud networks

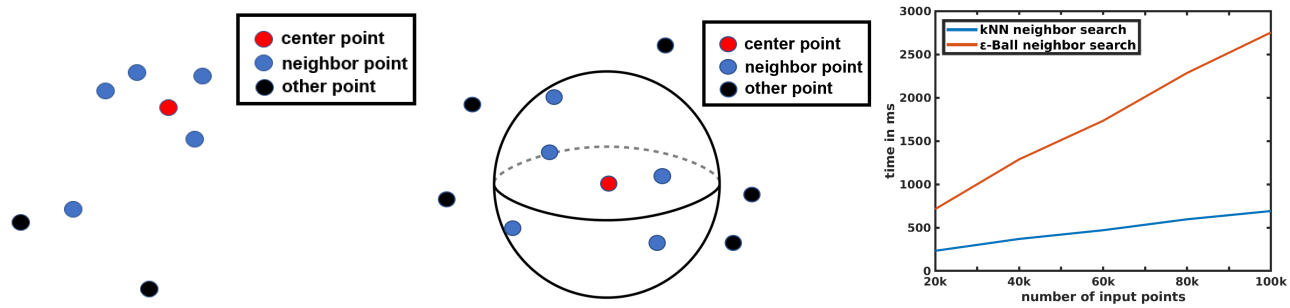

Figure 1. (a) kNN in point clouds; The neighborhood might be very irregular; (b) Epsilon-ball neighborhood controls the distance of neighbors; (c) Neighbor search time comparison between kNN and ϵ-ball in a deep network. Both implemented using `nanoflann` with the default KPConv model architecture on SemanticKITTI. ϵ-ball neighbor search is $2-4$ times slower than kNN (Best Viewed in Color)

augment the data by randomly jittering point locations, but such jittering only provides local generalization to different neighborhood sizes. Such a simple shortcut is, however, unlikely to suffice for point clouds as each kNN neighborhood may be significantly different from others in terms of scale. Indeed, even in 2D images practitioners usually rely on re-scaling of all the images to the same scale to avoid this generalization issue.

Another common approach that is adopted in the state-of-the-art KPConv [56] is to use an ϵ-ball, which ensures that neighboring points do not have too large distances (Fig. 1(b)). However, ϵ-ball is computationally significantly slower than kNN, posing a significant burden to the training and testing time of these networks (Fig. 1(c)). Besides, the resulting uneven number of neighbors (some points may have less neighbors, even 0) may introduce further wastes in computation time and memory (e.g. significant amount of zero-padding to ensure vectorized computations).

For the kNN neighborhood to perform well, it would be important to build some invariance over the selection of neighborhoods so that it generalizes from training neighborhoods to testing. Since PointConv is an MLP on the point coordinates, one interesting idea is to directly build those invariances through coordinate transforms. In this paper, we introduce a novel viewpoint-invariant (VI) descriptor on the 3D coordinates by exploiting angles between the surface normal and an invariant orthonormal bases. However, one needs to note that real-life data is seldom fully invariant to rotation/scales, e.g. the scale of the object can be an indicator of many classification categories such as refrigerator or cars, and the placement on ground might be salient features for the network. Our results show that the combination of this viewpoint-invariant coordinate transform with the original 3D coordinates tend to yield significantly improved results from convolution on the kNN neighborhood, so that it becomes comparable to the ϵ-ball based KPConv, the state-of-the-art in point cloud-based convolutional networks. This would lead to significant efficiency gains for the application of point-based networks.

Our VI-transform contains dimensions that are invariant to scale changes, which enables the network to generalize to neighborhoods of different scale. Such generalization can lead to significant performance improvements when the point cloud is heavily downsampled during testing time, since that would make the radius of the kNN neighborhood significantly different from training time and pose a challenge to non-invariant convolution functions.

Furthermore, experiments show that with the VI transform smaller models suffer substantially less performance loss, paving the way for smaller models with greater efficiency and robustness to be deployed in practice. Although we have not fully closed the gap between point cloud convolution approaches and sparse volumetric ones, we believe this research is a useful step towards better understanding the implications of convolutions in irregular neighborhoods.

2. Related Work

Volumetric and Projection-based Approaches A direct extension from convolution in 2D raster images to 3D is to compute convolution on volumetric grids [68, 39, 45, 63]. In densely sampled point clouds, sparse volumetric convolutions[6, 14] currently work better than point-based approaches. However, point cloud convolution have potentially much wider use cases when the sampling density is very low or uneven. They also can be used in convolutions higher than 3-dimensional, such as 4D or 6D cost volume convolutions [67] where discretized convolutions are hard to be applied to. Some other approaches that project point clouds onto multi-view 2D images [53, 43, 34] or lattice space [52] may suffer from the same issue. Currently, the best approaches on LIDAR fuse a point-based head and a volumetric head [38, 49], which showed that both types of approaches have their own benefits. Especially, volumetric approaches often cannot obtain enough details e.g. on thin parts and object boundaries. Improving the point-based network could potentially improve fusion performance as well.
Point-based Approaches PointNet [42] first attempted to directly work on point clouds, and PointNet++ [44] improved it by adding a hierarchical structure. Other studies

also attempted to utilize hierarchical architectures to aggregate information from neighbor points with MLPs [28, 35]. Other point-based methods include [31] [59] [15] [25] [79] [71] [58] [74] use different ideas to improve over PointNet [42] and achieved better performances.

Generally, convolutional approaches on point clouds performed better than the approaches listed above. [50, 22, 65, 18, 64, 73] proposed to learn discretizations of continuous convolutional filters. [22] utilized a side network to generate weights for 2D convolutional kernels. [50] generalized it to 3D point clouds, and [64] proposed an efficient approximation which corresponds to depthwise convolution. EdgeConv [65] encodes pairwise features between a neighbor point and the center point through MLPs. [18] takes densities into account. Pointwise CNN [20] located kernel weights for predefined voxel bins, so it was not flexible. SpiderCNN [73] proposed a polynomial weight function, which we experiment with in this paper. However, they did not utilize regularization to control the smoothness. The main contribution in PointConv [66] is an efficient variant that does not explicitly generate weight functions, but implicitly so, which removed the memory requirement to store the weights, allowing for scaling up to the "modern" deep network size, e.g. dozens of layers with hundreds of filters per layer. It is also the only paper showing results on CIFAR-10 matching those of a 2D CNN.

The main competitive point-based convolutional approach to PointConv is KPConv [56]. In KPConv, convolution weights are generated as kernel functions between each point and anchor points, points in the 3D space that are pre-specified as parameters for each layer separately. KPConv enjoys nice performance due to the smooth and well-regularized kernel formulation, but it introduces significantly more parameters in the specification of anchor points and their ϵ-ball neighborhoods are computationally costly. We show that with the proposed VI coordinate transform, PointConv with kNN is competitive with KPConv in terms of performance while being faster due to the usage of the kNN neighborhood. Similar to KPConv, PCNN [1] also assign anchor points with kernel weights, but it does not take neighbor points into account for convolution.

Scale and rotation invariance in convolution Building rotation and scale invariance into recognition systems has been a long-standing goal for computer vision. A standard approach has been data augmentation [51, 16, 57, 27, 5, 17], where the training set is augmented by including objects with random rescaling or rotations. Other studies attempted to integrate deep CNNs with side-networks [33, 80, 61, 72, 76, 24] or attention modules [60, 48]. [82] convolved the input with several rotated versions of the same CNN filter before feeding to the pooling layer. Some techniques proposed to learn transformations directly [21, 32] on the input or intermediate outputs from convolutional layers in a deep network [11, 46]. There has also been interest in group-theoretical approaches[7, 3, 9]. Most of these work either include significant computational overhead, or only adapts to one type of invariance. [8, 78, 8, 54, 70, 41] aim to tackle the rotation invariance, but their experiments require SO(3) augmentation during training, which could be costly if the training dataset is large.

On point clouds, viewpoint-invariant point descriptors have been proposed for recognition and relocalization [47, 13] before deep learning. In deep learning, [78, 36, 23, 29] approached rotation invariance. [36, 23, 29] proposed to extract rotation invariant features for each neighborhood. [62, 75] build a spatial transformer side network (STN) to learn global transformations on input point clouds. [36] utilized the centroid of the neighborhood, which is sensitive to the number of neighbors. [29] relied on the intersection between the ϵ-ball and line extended from origin to the center point, which is sensitive to the radius. [23] performed principal component analysis (PCA) on a set of local neighbor points to establish the local frame. Instead of focusing solely on invariance, our work is the first to utilize an invariant coordinate transform to study the robustness of convolution operations in **non-invariant** tasks and show tangible improvements. In experiments we compare against some of these invariant approaches and show that they fail short of improving the performance in non-invariant tasks.

3. Methodology

3.1. Background: Convolution on Point Clouds

A point cloud can be denoted as a set of points $P = \{p_1, p_2, ..., p_n\}$, where each point p_i contains a position vector (e.g. if 3D, then $(x, y, z) \in R^3$) as well as a feature vector (RGB color, surface normal, etc.). A line of work including PointConv generalizes the convolution operation to point clouds based on discretizations of continuous 3D convolutions [50, 18, 65, 66]. For a center point $p_{xyz} = (x, y, z)$, its PointConv is defined by:

$$PConv(S, \mathbf{W}, \mathbf{F})_{xyz} = \sum_{(\delta_x, \delta_y, \delta_z) \in G} S(\delta_x, \delta_y, \delta_z) \mathbf{W}(\delta_x, \delta_y, \delta_z) \mathbf{F}(x + \delta_x, y + \delta_y, z + \delta_z) \quad (1)$$

where $(\delta_x, \delta_y, \delta_z)$ denote the coordinate offsets for a point in p_{xyz}'s local neighborhood G, usually located by kNN or an ϵ-ball $\mathcal{N} = \{(\delta_x, \delta_y, \delta_z) | \|\delta_x + \delta_y + \delta_z\|^2 \leq \epsilon\}$. $\mathbf{F}(x+\delta_x, y+\delta_y, z+\delta_z)$ represents the feature of the neighbor point, and $\mathbf{W}(\delta_x, \delta_y, \delta_z)$ generates the weights for convolution and is approximated implicitly by an MLP, called *WeightNet* in [66]. Finally, $S(\delta_x, \delta_y, \delta_z)$ represents the inverse local density to balance the impact of non-uniform sampling of the point clouds.

The novelty of PointConv over previous continuous convolution lies in its efficient computation of $W(\delta_x, \delta_y, \delta_z)$ by

re-writing it as the output of an MLP:

$$\mathbf{W}(\delta_x, \delta_y, \delta_z) = \mathbf{W}_2 g(\mathbf{W}_1(\delta_x, \delta_y, \delta_z)) \tag{2}$$

where $\mathbf{W}_1$ is a perceptron network with $1-2$ layers with a vector output, g is an activation function (e.g. ReLU) and $\mathbf{W}_2$ is a matrix of weights of a final linear layer. Because $g(\mathbf{W}_1(\delta_x, \delta_y, \delta_z))$ is shared across all filters, and $\mathbf{W}_2$ is independent of $(\delta_x, \delta_y, \delta_z)$ we can re-write convolution as:

$$\begin{aligned} PConv(S, \mathbf{W}, \mathbf{F})_{xyz} = \mathbf{W}_2 \sum_{(\delta_x, \delta_y, \delta_z) \in G} S(\delta_x, \delta_y, \delta_z) \\ g(\mathbf{W}_1(\delta_x, \delta_y, \delta_z)) \mathbf{F}(x+\delta_x, y+\delta_y, z+\delta_z) \end{aligned} \tag{3}$$

In this equivalent formulation, the summation can be computed only once for all the different filters in one layer as they only differ in the final $\mathbf{W}_2$. This has yielded significant speed and memory savings in the network and allowed a deep network of dozens of layers to be built from PointConv layers similar to 2D convolutions. For stride-2 convolution/pooling, one can just subsample the point clouds [44]. The formulation eq. (3) also allows computing output [66] on (x, y, z) with no features with a neighborhood that does not contain itself. Hence, classification and semantic segmentation tasks can be solved directly with PointConv networks. It is also straightforward to incorporate other commonly used 2D convolution operations, e.g. residual connections. Dilated convolution can be implemented by first sampling a larger kNN neighborhood, and then subsampling from the neighborhood.

3.2. kNN vs ϵ-ball neighborhoods

The neighborhood G in PointConv is usually defined by kNN. Fig. 1 (a) illustrates the potential robustness issue for the kNN neighborhood. Namely, the equivalent receptive field for a sparse point cloud is much larger than the one for a densely distributed point cloud. If trained only on dense (high resolution) point clouds, the learned weight function may not generalize well to much larger (unseen) receptive fields when dealing with sparse point clouds during testing.

An ϵ-ball based neighborhood [56] on the other hand would be robust to different sampling rates (Fig. 1(b)). For a point p_i, denote $N_\epsilon(p_i) = \{p_j \in P | d(p_i, p_j) < \epsilon\}$ as its ϵ-ball neighborhood. To ease the computation burden, we (randomly) select at most K neighbors from $N_\epsilon(p_i)$. The actual chosen neighbors from $N_\epsilon(p_i)$ are denoted as $C_\epsilon(p_i, K)$. Compared with kNN, the ϵ-ball neighborhood constrains the maximal distance of the neighbors w.r.t. the center point. Since different ϵ-balls may contain different number of neighbors, we replace the normalizer $S(\delta_x, \delta_y, \delta_z)$ in equation (**4**) from the main paper with $\frac{1}{|C_\epsilon(p_i, K)|}$. Note that the flexibility of the PointConv framework allows for variable number of neighbors in each neighborhood. However, the ϵ-ball neighborhood search is significantly slower than kNN because of the need to explore more nodes on the kDTree. With the state-of-the-art implementation `nanoflann` our tests on SemanticKITTI with the default KPConv network structure show that the neighborhood search for ϵ-ball is consistently $2-4$ times slower than the corresponding kNN search (Fig. 1(c)), greatly increasing training and testing times.

3.3. A Viewpoint-Invariant Coordinate Transform

PointConv relies on the (x, y, z) coordinates to compute the weights, which are sensitive to the rotation of the object as well as the sampling rate of the point clouds. In this subsection we describe a viewpoint-invariant coordinate transform which may serve as better inputs to the weight-generating function in PointConv. Our coordinate transform is defined in 3D based on the idea that the surface normal vector and the vector from one point to another can span an orthonormal basis of the 3D space that is invariant to rotations, and then angles computed between the surface normal and this basis can be utilized as scale-invariant features as well. Computing surface normal (usually with PCA from a neighborhood) is a standard operation utilized in many previous point cloud networks (e.g. [44]). But our way of using it to build a viewpoint-invariant coordinate transform is novel.

Suppose we have a center point p_μ with surface normal $\hat{n}_\mu$, for each point p_α with surface normal $\hat{n}_\alpha$, we develop its viewpoint-invariant (VI) coordinate transform w.r.t. p_μ as an 8-dimensional vector. We first denote $\vec{r}_\mu^\alpha = p_\alpha - p_\mu$, as the difference between p_α and p_μ. Using the Gram-Schmidt process from $\{\hat{n}_\mu, \vec{r}_\mu^\alpha\}$, we generate an orthonormal basis $\{\hat{r}, \hat{v}, \hat{w}\}$ where:

$$\hat{r} = \frac{\vec{r}_\mu^\alpha}{||\vec{r}_\mu^\alpha||}, \hat{v} = \frac{\hat{n}_\mu - (\hat{r}^\top \hat{n}_\mu)\hat{r}}{\sqrt{1-(\hat{r}^\top \hat{n}_\mu)^2}}, \hat{w} = \frac{\hat{r} \times \hat{v}}{\|\hat{r} \times \hat{v}\|} \tag{4}$$

where $\times$ denotes outer product, as illustrated in Fig. 2. Note that this basis is seldom degenerate, because it is unlikely for $\hat{n}_\mu$ and $\vec{r}_\mu^\alpha$ to be collinear in 3D surface point clouds.

With a global rotation of the scene, the basis and normal vectors are identically rotated. hence the angles between $\hat{n}_\mu$, $\vec{r}_\mu^\alpha$ and $\hat{n}_\alpha$ remain the same. We also compute the projection lengths of $\hat{n}_\alpha$ and $\hat{n}_\mu$ onto the orthonormal bases. Hence, our viewpoint-invariant descriptor provides a complete characterization of the vectors $\hat{n}_\mu$, $\vec{r}_\mu^\alpha$ and $\hat{n}_\alpha$. Formally, for each point p_α in the neighborhood of p_μ, we extract the following rotation-invariant coordinate transform:

$$\begin{aligned} \beta_\mu^\alpha = [\hat{n}_\alpha \cdot \hat{n}_\mu, \frac{\hat{r}_\alpha \cdot \hat{n}_\mu}{\|\hat{r}_\alpha\|}, \frac{\hat{r}_\alpha \cdot \hat{n}_\alpha}{\|\hat{r}_\alpha\|}, \hat{n}_\alpha \cdot \hat{v}_\alpha, \hat{n}_\alpha \cdot \hat{w}_\alpha, \\ \vec{r}_\mu^\alpha \cdot \hat{n}_\mu, \vec{r}_\mu^\alpha \cdot (\hat{n}_\alpha \times \hat{n}_\mu), ||\vec{r}_\mu^\alpha||] \end{aligned} \tag{5}$$

where $\times$ represents the cross product. We name it as the viewpoint-invariant (VI) descriptor, where the first 5 dimensions are both scale and rotation invariant as they are angles between normalized vectors, and the last 3 dimensions are

rotation invariant only. We believe that generating the convolutional weights with this viewpoint-invariant descriptor will improve robustness to different scale and rotations between neighborhoods in training and the test set.

Note that the weight-generating network in PointConv is an MLP, which in principle has universal approximation properties to be able to approximate any nonlinear function of $(\delta_x, \delta_y, \delta_z)$, hence it is a fair argument that with enough data it is possible that the network could learn a fixed transformation in $\mathbf{W}_1$ that is as effective or more so than eq. (5). However, first it is not clear what amount of data would be needed for this to happen, and second deep networks could easily overfit to less robust descriptors since they would only be seeing the training set and not understanding the generalization problem enough. In our experiments on current datasets, we have always observed significant improvements of the VI transform over the regular approach of inputting $(\delta_x, \delta_y, \delta_z)$ to $\mathbf{W}_1$.

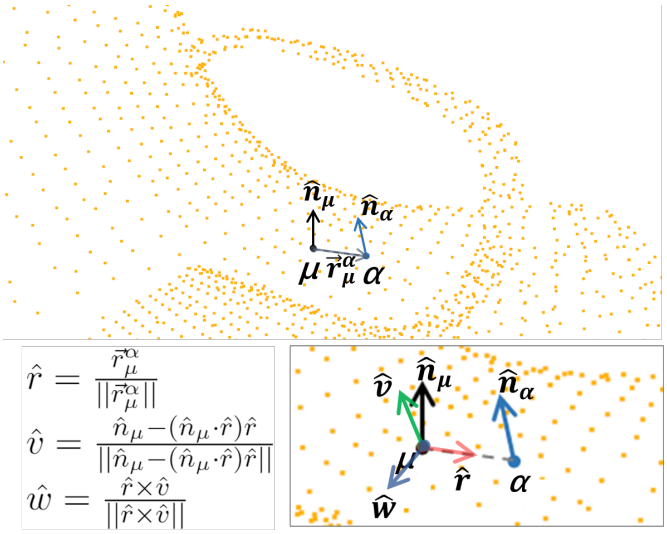

Figure 2. For a given local center point p_μ and $p_\alpha \in N_\epsilon(p_\mu)$ for a pair of points, a set of viewpoint-agnostic basis $(\vec{r}, \vec{w}, \vec{v})$ can be generated from $\vec{r}_\mu^\alpha$ and n_μ with the Gram-Schmidt process, and viewpoint-invariant features such as the angles between $\vec{n_\mu}$ and $\vec{v}$ can be extracted from them

The VI descriptor can either be used standalone or concatenated with the conventional $(\delta_x, \delta_y, \delta_z)$ representation (we use the shorthand VI+XYZ for the contatenated version) as the input to the weight-generating function $W(\cdot)$ in eq.(3). When concatenated, it creates room for the learning algorithm to automatically select from rotation-invariant, scale and rotation-invariant, and non-invariant $(\delta_x, \delta_y, \delta_z)$ convolution weights. In real-life segmentation scenarios, not all kernels should be scale or rotation invariant. For example, usually a bicyclist would be above a bicycle, instead of below it. Such relationship would only be captured with the non-invariant features instead of rotation-invariant ones. Hence it would be important for the network to have the flexibility to choose. Empirically, we have observed significant performance improvements by this concatenation of different levels of invariance.

The additional computational cost of using the VI descriptor is small. It only affects the first layer in the WeightNet of PointConv computation. With an output dimensionality of 8 or 16, this only adds negligible cost to the network. Yet the better generalization it provides is fascinating. We adopt the `estimate_normals` function in the Open3D library [81] with radius = 0.1 and maximal neighbors of 30. For $100K$ points, it takes about 0.0782 seconds on CPU for the computation, which is less than $1/3$ of the time of the forward pass of the network. The hardware we used include a single RTX 2080Ti, and a AMD 3600 CPU.

3.4. Complete Rotation-Invariance

VI-PointConv achieves rotation invariance on the weights. However, for a completely rotation-invariant network, one needs both rotation-invariant convolutional weights and rotation-invariant features. The conventionally used (x, y, z) feature is not invariant. In order to obtain complete invariance from the network, we propose to utilize the rotation-invariant principle curvatures as features when complete invariance is needed. To be more specific, we compute the principle curvatures (k_1 and k_2), the Gaussian curvature (G), and the mean curvature (H) based on [40]. The final feature input vector is $[k_1, k_2, G, H]$.

Method	SO3/SO3	none/SO3
PointNet(with T-Net) [42]	79.9	-
PointNet++ [44]	80.6	-
DGCNN(with T-Net) [65]	84.4	-
RS-CNN [37]	82.6	-
SpiderCNN(with T-Net) [73]	78.7	-
RIConv [78]	86.4	-
SphericalCNN [8]	86.9	-
SRI-Net [54]	87.0	-
Triangle-Net [70]	86.7	-
SPH-Net [41]	87.6	-
RTN+DGCNN [12]	86.5	-
ClusterNet [4]	87.1	-
PointConv [66]	-	61.0
VI-PointConv (ours)	-	**88.2**

Table 1. Comparison on ModelNet40 with Data SO(3) for 3D point cloud classification. SO3/SO3 indicates the model is trained & tested with SO(3) rotations, and none/SO3 indicate the model is trained without SO3 augmentations & tested with SO(3) rotations.

4. Experiments

4.1. ModelNet40

We first experiment on the ModelNet40 dataset with the main goal to show the capability of VI-PointConv to produce rotation-invariant convolutions in SO(3) on the ModelNet40 [69] dataset. Here we utilize the curvatures as input features for complete invariance.

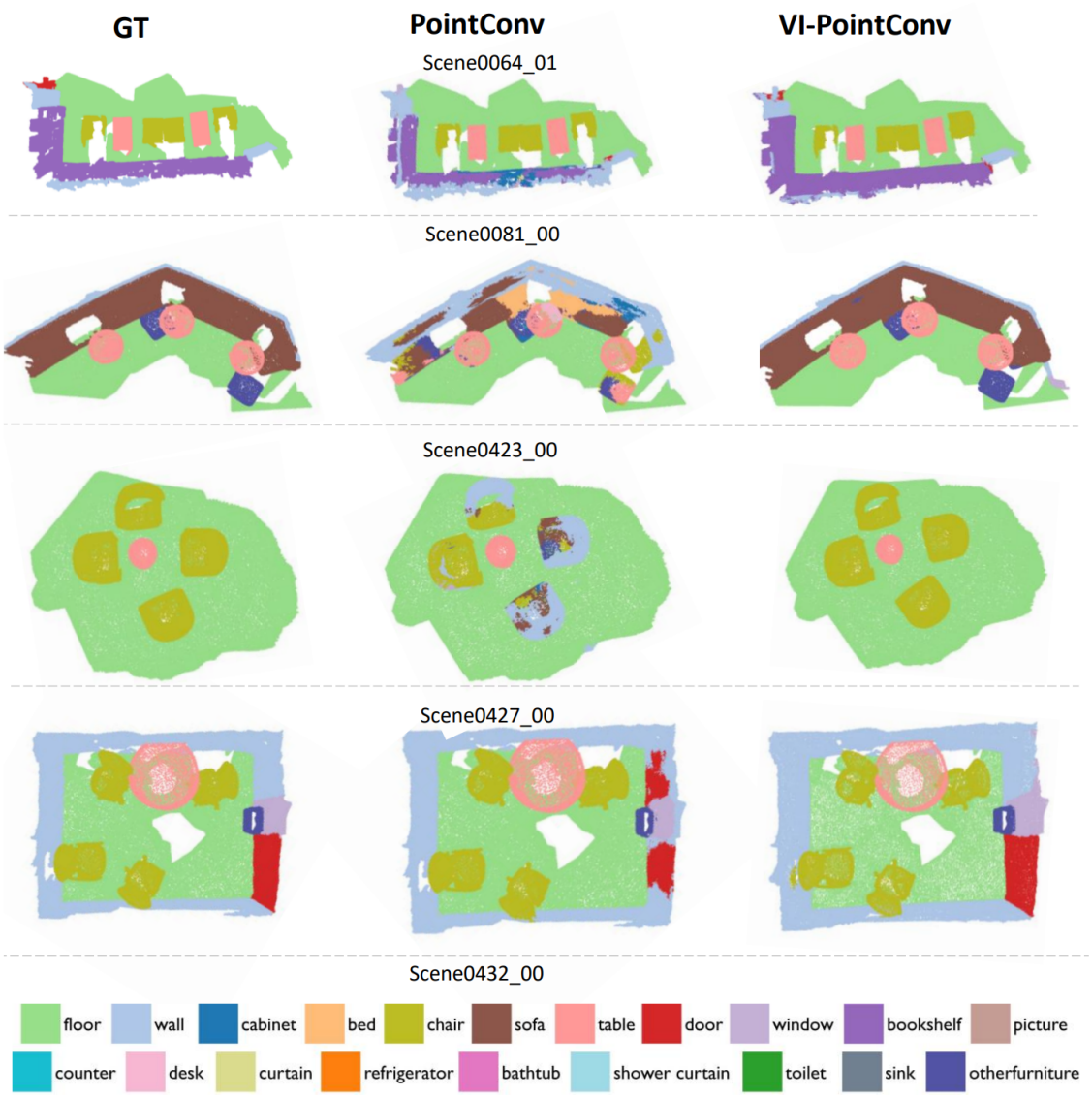

Figure 3. Examples of semantic segmentation results of ScanNet dataset. The images from left to right are the ground truth segmentation, the prediction from PointConv, and the prediction from VI-PointConv. (Best viewed in color)

To evaluate the performance in SO(3), we randomly rotate each point cloud with any arbitrary angle and perform the experiments five times independently and use the mean results as the final results. 1024 points are used as input for all the methods. Our framework is fully rotation-invariant hence do not need any rotation augmentation during training, where all the baselines require rotation augmentation at training time. In Table 1, VI-PointConv outperforms other baselines significantly, without requiring additional training time augmentations.

4.2. ScanNet

We conduct 3D semantic scene segmentation on the ScanNet v2 [10] dataset. We use the official split with $1,201$ scenes for training and 312 for validation. We implemented the state-of-the-art 16-layer PointConv architecture that achieved 66.6% on the ScanNet testing set, provided by the authors [66]. Results are reported on the ScanNet validation set as the benchmark organizers do not allow ablation studies on the testing set. More details on experimental setup is provided in the supplementary material.

To study the robustness to scales and neighborhood size, we subsampled at testing time each validation point cloud from the original $100k$ points to substantially less — $\{60k, 40k, 20k, 10k\}$. This is equivalent to downsampling the image in the 2D space as it increases the size of kNN neighborhoods with a fixed K. (Note that 2D regular CNNs would usually not work with such aggressive test-

MLP input	σ	$100k$	$60k$	$40k$	$20k$	$10k$
VI+XYZ	ReLU	**70.1**	**64.4**	**63.0**	**57.6**	45.4
VI	ReLU	63.3	60.7	59.7	55.0	44.7
VI	SeLU	63.7	61.5	57.7	53.1	40.2
surface normal + XYZ	ReLU	60.2	57.5	56.8	54.8	**50.9**
surface normal	ReLU	53.1	50.6	50.2	47.6	43.3
XYZ-Only	ReLU	61.7	58.7	53.4	34.6	17.8
VI(first layer only)	ReLU	63.1	60.5	57.8	45.8	28.2

Table 2. Performance results (mIoU,%) on the ScanNet validation dataset with a 16-layer PointConv network. The first column shows the configurations being tested, the σ column shows the activation function, $100k, \ldots, 10k$ refers to the number of subsampled points. The default number of neighbors is 8. It can be seen that VI+XYZ significantly outperforms all other variants, including surface normal + $(\delta_x, \delta_y, \delta_z)$ which contains the same amount of information. Besides, the VI input alone outperforms surface normals and $(\delta_x, \delta_y, \delta_z)$ inputs.

MLP input	NBR	$100k$	$60k$	$40k$	$20k$	$10k$
VI	kNN	63.3	60.7	59.7	55.0	44.7
VI	ϵ-ball	61.6	58.6	58.0	52.3	40.6
XYZ	kNN	61.7	58.7	53.4	34.6	17.8
XYZ	ϵ-ball	48.9	43.3	39.7	30.6	20.7

Table 3. mIOU results on a 16-layer PointConv network on the ScanNet dataset, NBR stands for neighboorhood type. $100k, \ldots, 10k$ refer to the number of subsampled points. It can be seen that kNN outperforms ϵ-ball in both cases.

MLP input	NBR	$100k$	$60k$	$40k$	$20k$	$10k$
VI+XYZ	kNN	**64.5**	**61.3**	**60.6**	**57.3**	**51.2**
VI	kNN	61.0	58.8	57.5	50.8	39.4
XYZ	kNN	55.3	53.3	47.0	30.7	16.1
VI	ϵ-ball	59.2	57.5	55.1	44.8	31.1
[77]	kNN	53.0	48.6	44.4	31.4	16.2
[29]	ϵ-ball	44.8	43.0	39.3	26.5	17.7

Table 4. mIoU results on a 4-layer PointConv network on the ScanNet dataset, NBR stands for neighboorhood type. $100k, \ldots, 10k$ refer to the number of subsampled points at test time

Method	mIoU(%) (test)	mIoU(%) (val)
PointNet++ [44]	33.9	-
SPLATNet [52]	39.3	-
TangentConv [55]	40.9	-
PointCNN [31]	45.8	-
PointASNL [74]	63.0	63.5
PointConv [66]	66.6	61.0
KPConv [56]	**68.4**	69.2
VI-PointConv (ours)	67.6	**71.2**

Table 5. Semantic Segmentation results for point-based approaches on the ScanNet test set

time downsampling[30]). Also, each sub-sampled point cloud is further rotated with 4 different predefined angles around z-axis — $\{0°, 90°, 180°, 270°\}$. Such operations could significantly change both local scales and rotations. We also evaluate the performance when the rotation augmentation is not applied during training. We found performance variation between different rotation angles to be less than 1% (see supplementary), hence the mIoUs averaged from all angles are reported.

Validation Set Performance Several different result tables are shown. In Table 2, we evaluate the 16-layer PointConv model among different variants VI+XYZ, VI only, XYZ-only (input only the delta coordinates $(\delta_x, \delta_y, \delta_z)$ to the $W(\cdot)$ function as in eq. (1)), as well as surface normal and surface normal + XYZ. For the last two settings, we directly input surface normals as additional input dimensions to the $W(\cdot)$ function in eq. (1), which contains the same information content as VI, but without the invariance.

The results showed that the proposed VI descriptor significantly improved the performance as well as robustness. Especially, it is significantly more robust to test-time downsampling than the $(\delta_x, \delta_y, \delta_z)$ coordinates as input. For example, at $20k$ testing points (reflecting $5x$ downsampling from the training), the VI descriptors still maintain a 57.6% accuracy while the performance of the $(\delta_x, \delta_y, \delta_z)$ coordinates version dropped to 34.6%, marking a relative improvement of $\mathbf{66.4}\%$. The improvement of VI over surface normals is also very significant – an mIOU improvement of $8\% - 10\%$, which indicates that VI is a better representation of local geometry than the commonly used surface normal.

To further explore the potentials of the VI descriptor, we replace $(\delta_x, \delta_y, \delta_z)$ with the VI descriptor for the first layer only and report the performance at the last row. Compared with the second last row, the VI descriptor significantly improves the performance as well as the robustness. However, if we compare the last row and the second row (where VI descriptors are applied to each layer), the robustness (e.g. at 10K points) dropped significantly, which indicates that inputting VI descriptor in latter layers helps significantly on robustness.

Finally, when we combined the VI coordinates with $(\delta_x, \delta_y, \delta_z)$ inputs, it generated the best performance of all – 71.2% on the original validation set, and better on almost all subsampled scenarios. This shows that a combination of scale-invariant, rotation-invariant and non-invariant coordinates is beneficial, potentially offering the network the flexibility to choose the invariance it requires.

Table 3 shows a comparison between kNN and ϵ-ball with PointConv. Interestingly, adopting ϵ-ball with $(\delta_x, \delta_y, \delta_z)$ actually decreased the performance significantly. With VI, the performance decrease is less severe but still exists. We are not sure about the reason ϵ-ball would not work on 3D point cloud with PointConv, but we suspect that PointConv might be more sensitive to having the same number of neighbors in each neighborhood, in order for the learned weight functions to be comparable with each other.

Method	**mIoUs(%)**	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic-sign
PointNet[42]	14.6	61.6	35.7	15.8	1.4	41.4	46.3	0.1	1.3	0.3	0.8	31.0	4.6	17.6	0.2	0.2	0.0	12.9	2.4	3.7
SPG[26]	17.4	45.0	28.5	0.6	0.6	64.3	49.3	0.1	0.2	0.2	0.8	48.9	27.2	24.6	0.3	2.7	0.1	20.8	15.9	0.8
SPLATNet[52]	18.4	64.6	39.1	0.4	0.0	58.3	58.2	0.0	0.0	0.0	0.0	71.1	9.9	19.3	0.0	0.0	0.0	23.1	5.6	0.0
PointNet++[44]	20.1	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9
TangentConv[55]	40.9	83.9	63.9	33.4	15.4	83.4	90.8	15.2	2.7	16.5	12.1	79.5	49.3	58.1	23.0	28.4	8.1	49.0	35.8	28.5
PointConv[66]	53.0	86.2	68.6	57.7	16.0	89.9	94.2	30.2	29.5	33.9	30.5	78.9	60.8	63.7	48.8	45.7	20.4	59.9	53.4	38.6
RandLA-Net[19]	53.9	**90.7**	**73.7**	60.3	20.4	86.9	94.2	40.1	26.0	25.8	38.9	81.4	61.3	66.8	49.2	48.2	7.2	56.3	49.2	47.7
KPConv[56]	58.8	88.8	72.7	61.3	31.6	90.5	**96.0**	33.4	30.2	**42.5**	44.3	**84.8**	**69.2**	**69.1**	**61.5**	**61.6**	11.8	64.2	56.5	47.4
VI-PointConv (ours)	**59.6**	88.8	72.5	**63.5**	**32.7**	**91.4**	95.9	**41.8**	**38.6**	35.0	**45.7**	83.9	68.0	66.9	51.2	50.1	**27.6**	**66.6**	**57.4**	**54.8**

Table 6. Semantic Scene Segmentation results for point-based approaches on the SemanticKITTI test set

In Table 4, we showed the result of a simpler 4-layer model (see supplementary for more details). It can be seen that VI with kNN still significantly outperform $(\delta_x, \delta_y, \delta_z)$. Especially, VI + XYZ in this 4-layer model has reached performances quite close to that of the 16-layer one, especially at more significant test-time downsampling rates such as $40k, 20k$ and $10k$ points. On $10k$ points, the 4-layer model even outperforms the 16-layer model, showing the potential of deploying such lightweight models in practice if the need arises. In addition, we also compared against the rotation invariant descriptors from [29, 77]. The results were demonstrated at the last two rows which showed that our VI coordinates significantly outperform those rotation-invariant descriptors in real-life data. In Fig. 3 we showed some qualitative results comparing PointConv and VI-PointConv. It can be seen that with VI-PointConv a lot more regions that are uncertain under PointConv are now segmented correctly, which drove the significant improvement.

Test Set Performance On the test set, we achieved comparable mIoU with KPConv [56], the state-of-the-art among point-based approaches (Table 5). However, our framework significantly outperforms KPConv [56] on the validation set. Note that the kNN used in PointConv is still significantly more efficient than the ϵ-ball in KPConv (see the supplementary for more details). Besides, it is not possible to apply VI to KPConv, because the anchor points they choose are likely to not lie on surfaces and do not have surface normals attached with them. Hence we argue VI-PointConv provided more flexibility than KPConv.

4.3. SemanticKITTI

We also evaluate the semantic segmentation performance on SemanticKITTI [2] (single scan), which consists of $43,552$ point clouds sampled from 22 sequences in driving scenes. Each point cloud contains $10 - 13k$ points, collected by a single Velodyne HDL-64E laser scanner, spanning up to $160 \times 160 \times 20$ meters in 3D space. The officially training set includes $19,130$ scans (sequences $00 - 07$ and $09 - 10$), and there are $4,071$ scans (sequence 08) for validation. For each 3D point, only (x, y, z) coordinate is given without any color information. It is a challenging dataset because faraway points are sparser in LIDAR scans. We adopt the exact same 16-layer architecture as in ScanNet. The mini-batch size of 16. The initial learning rate is 10^{-3}, and it is decayed by half every 6 epochs. We do not integrate with any subsampling preprocessing. As reported in Table 6, we achieve the state-of-the-art semantic segmentation performance among point-based baselines, improving by 0.8% over KPConv and 6.6% over standard PointConv.

5. Conclusion

In this paper, we propose a novel viewpoint-invariant transformation for 3D point coordinates, as the input to the weight generation network for PointConv. This coordinate transformation allows us to line up rotation, scale-invariant as well as noninvariant descriptors for the relative coordinates so that the network could learn to choose the amount of invariance it needs in generating the convolution function. Experiments have shown that this could significantly improve the performance of PointConv, bringing it up to be comparable with ϵ-ball neighborhood based KPConv, as well as being more robust to different kNN neighborhoods, test-time downsampling of the point cloud and a substantially smaller model. Our approach adds minimal computational cost to PointConv and we believe there are many applications that can benefit from it. In the future, we would like to further explore its application in problems where viewpoint-invariance is extremely important, such as relocalization problems in Simultaneous Localization and Mapping (SLAM).

Acknowledgments

This work is partially supported by the National Science Foundation grants CBET-1920945, IIS-1751402 and IIS-1911232.

References

[1] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *international conference on computer graphics and interactive techniques*, 2018.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[3] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *International Conference on Robot and Automation*, 2017.

[4] Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4989–4997, 2019.

[5] Gong Cheng, Peicheng Zhou, and Junwei Han. Rifd-cnn: Rotation-invariant and fisher discriminative convolutional neural networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[7] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 2990–2999, 2016.

[8] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

[9] Taco S Cohen and Max Welling. Steerable cnns. In *ICLR*, 2017.

[10] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[11] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

[12] Shuang Deng, Bo Liu, Qiulei Dong, and Zhanyi Hu. Rotation transformation network: Learning view-invariant point cloud for classification and segmentation. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2021.

[13] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *CVPR*, pages 998–1005. IEEE Computer Society, 2010.

[14] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018.

[15] Fabian Groh, Patrick Wieschollek, and Hendrik P. A. Lensch. Flex-convolution (million-scale point-cloud learning beyond grid-worlds). In *Asian Conference on Computer Vision (ACCV)*, Dezember 2018.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*. 2014.

[17] João F. Henriques and Andrea Vedaldi. Warped convolutions: Efficient invariance to spatial transformations. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[18] P. Hermosilla, T. Ritschel, P-P Vazquez, A. Vinacua, and T. Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2018)*, 37(6), 2018.

[19] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[20] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, pages 2017–2025, 2015.

[22] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 667–675. Curran Associates, Inc., 2016.

[23] Park Jaeyoo Kim, Seohyun and Bohyoung Han. Rotation-invariant local-to-global representation learning for 3d point cloud. In *Advances in Neural Information Processing Systems*, 2020.

[24] Yonghyun Kim, Bong-Nam Kang, and Daijin Kim. San: Learning relationship between convolutional features for multi-scale object detection. In *ECCV*, 2018.

[25] Artem Komarichev, Zichun Zhong, and Jing Hua. A-cnn: Annularly convolutional neural networks on point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

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

[27] Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 289–297, 2016.

[28] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[29] Xianzhi Li, Ruihui Li, Guangyong Chen, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. A rotation-invariant framework for deep point cloud analysis. *IEEE transactions on visualization and computer graphics*, PP, 2021.

[30] Xingyi Li, Zhongang Qi, Xiaoli Z. Fern, and Fuxin Li. Scalenet - improve cnns through recursively rescaling objects. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 11426–11433. AAAI Press, 2020.

[31] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 820–830. Curran Associates, Inc., 2018.

[32] Chen-Hsuan Lin and Simon Lucey. Inverse compositional spatial transformer networks. In *CVPR*, 2017.

[33] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[34] Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. Fpconv: Learning local flattening for point convolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[35] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[36] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8895–8904, 2019.

[37] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis (cvpr 2019 oral best paper finalist). 06 2019.

[38] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *arXiv preprint arXiv:1907.03739*, 2019.

[39] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[40] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, 2003.

[41] Adrien Poulenard, Marie-Julie Rakotosaona, Yann Ponty, and Maks Ovsjanikov. Effective rotation-invariant point cnn with spherical harmonics kernels. In *2019 International Conference on 3D Vision (3DV)*, pages 47–56, 2019.

[42] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[43] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016.

[44] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[45] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[46] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *NIPS*. 2017.

[47] Rahul Sawhney, Fuxin Li, Henrik I. Christensen, and Charles L. Isbell Jr. Purely geometric scene association and retrieval - A case for macro scale 3d geometry. *CoRR*, abs/1808.01343, 2018.

[48] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. Action recognition using visual attention. In *NIPS Time Series Workshop*. 2015.

[49] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[50] M. Simonovsky and N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29–38, 2017.

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[52] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.

[53] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015.

[54] Xiao Sun, Zhouhui Lian, and Jianguo Xiao. Srinet: Learning strictly rotation-invariant representations for point cloud classification and segmentation. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 980–988, 2019.

[55] Maxim Tatarchenko*, Jaesik Park*, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3D. *CVPR*, 2018.

[56] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[57] David A van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. 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 5998–6008. Curran Associates, Inc., 2017.

[59] Nitika Verma, Edmond Boyer, and Jakob Verbeek. FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis. In *CVPR - IEEE Conference on Computer Vision & Pattern Recognition*, pages 2598–2606, Salt Lake City, United States, June 2018. IEEE.

[60] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *CVPR*, 2017.

[61] Hao Wang, Qilong Wang, Mingqi Gao, Peihua Li, and Wangmeng Zuo. Multi-scale location-aware kernel representation for object detection. In *CVPR*, 2018.

[62] Jiayun Wang, Rudrasis Chakraborty, and Stella X. Yu. Spatial transformer for 3d points. *CoRR*, abs/1906.10887, 2019.

[63] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Transactions on Graphics (SIGGRAPH)*, 36(4), 2017.

[64] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[65] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[66] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[67] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *European Conference on Computer Vision*, pages 88–107. Springer, 2020.

[68] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[69] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[70] Chenxi Xiao and Juan Wachs. Triangle-net: Towards robustness in point cloud learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 826–835, 2021.

[71] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[72] Hongyu Xu, Xutao Lv, Xiaoyu Wang, Zhou Ren, Navaneeth Bodla, and Rama Chellappa. Deep regionlets for object detection. In *ECCV*, 2018.

[73] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[74] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[75] Wentao Yuan, David Held, Christoph Mertz, and Martial Hebert. Iterative transformer network for 3d point cloud. *arXiv preprint arXiv:1811.11209*, 2018.

[76] Rui Zhang, Sheng Tang, Yongdong Zhang, Jintao Li, and Shuicheng Yan. Scale-adaptive convolutions for scene parsing. *ICCV*, 2017.

[77] Zhiyuan Zhang, Binh-Son Hua, David Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. pages 204–213, 09 2019.

[78] Zhiyuan Zhang, Binh-Son Hua, David W. Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International Conference on 3D Vision (3DV)*, pages 204–213, 2019.

[79] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. PointWeb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019.

[80] Peng Zhou, Bingbing Ni, Cong Geng, Jianguo Hu, and Yi Xu. Scale-transferrable object detection. In *CVPR*, June 2018.

[81] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[82] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. In *CVPR*, 2017.