Deformable Filter Convolution for Point Cloud Reasoning

Yuwen Xiong* 1,2 , Mengye Ren*1,2 , Renjie Liao^{1,2} , Kelvin Wong^{1,2} , Raquel Urtasun^{1,2}

¹Uber Advanced Technologies Group

²University of Toronto

{yuwen, mren3, rjliao, kelvin.wong, urtasun}@uber.com

Abstract

Point clouds are the native output of many real-world 3D sensors. To borrow the success of 2D convolutional network architectures, a majority of popular 3D perception models voxelize the points, which can result in a loss of local geometric details that cannot be recovered. In this paper, we propose a novel learnable convolution layer for processing 3D point cloud data directly. Instead of discretizing points into fixed voxels, we deform our learnable 3D filters to match with the point cloud shape. We propose to combine voxelized backbone networks with our deformable filter layer at 1) the network input stream and 2) the output prediction layers to enhance point level reasoning. We obtain state-of-the-art results on LiDAR semantic segmentation and producing a significant gain in performance on LiDAR object detection.

1. Introduction

3D perception is one of the key components of real-world robotic systems. These robots are typically equipped with 3D sensors such as LiDAR and RGBD cameras which produce outputs in the form of point clouds. These point clouds correspond to a set of vectors of location coordinates and associated features. Driven by the success of deep learning on 2D images, there has been a fresh wave of deep network architectures proposed to tackle this new challenge: unlike image grids, point clouds are sampled sparsely and non-uniformly with continuous spatial coordinates, and they are equivalent up to permutations.

A well-studied approach is to discretize the points into voxels [22]. As 3D convolution can be inefficient in terms of both computation and storage, various approximations have been proposed [29, 37] – recent work also finds 2D convolutions almost as competitive but with a much improved efficiency [44, 47, 28]. Despite great strides being made by leveraging existing 2D CNN backbone networks

on voxelized inputs, current approaches face the dilemma of either clashing with a loss of local geometric details or struggling with sensor noise and a smaller field of view.

To complement the weaknesses of voxelized networks, new network architectures have been proposed to process points directly. While simple average and max-pooling operations can preserve permutation invariance [24, 26, 45], this approach lacks interpretability on how spatial features are aggregated. Various attempts to define a general learnable continuous convolution layer have been made, by hammering a learned multi-layer network to predict the filter weights or features [38, 19, 16, 9], or restricting to a simpler family of functions [42, 31]. While the former may lack robustness and require extra supervision for regions with sparse neighborhoods, the latter could be limited by less powerful feature representations.

This paper advocates a simple idea: we learn 3D cuboid filters just like the ones in a voxelized network. However, when performing the convolution operation, instead of discretizing the points into a voxel grid, we deform the 3D filter towards the point clouds. We propose two ways to integrate our proposed convolution operator into popular backbones, by 1) enriching feature representation from an additional point-wise input stream, and by 2) smoothing out point-wise predictions within local neighborhoods. Composing our deformable filter convolution layers with voxelized networks results in a significant gain in performance on semantic segmentation and object detection tasks, comparing to voxel only networks and previous attempts at fusing point-wise features. Moreover, our proposed joint network achieves state-of-the-art performance on the TOR4D large-scale LiDAR semantic segmentation benchmark.

2. Related work

Previous work on processing 3D data can be roughly categorized into multi-viewpoints, voxelization, and point-based representations. Inspired by an agent-centric 2D view of the world, multi-view representations [35, 25, 4] treat 3D data as snapshots of 2D images taken at different view points. Front view representations [14], which considers

^{*}Equal contribution

depth as an additional channel in the input, can leverage 2D image network architectures. However, these approaches bear a significant loss of 3D information, and are unable to reason about 3D rigid transformations. To address this issue, voxelization-based representations instead process the occupancy grids using 3D convolution [22]. As pure 3D convolution suffers from computation and storage inefficiency, OctNet [29] and O-CNN [37] use OctTree to efficiently compute voxel convolutions.

While voxels are intuitive 3D counterparts to 2D images, real world sensors such as LiDAR and depth camera instead produce point clouds as their native output. A popular approach is to discretize points with point statistics in each voxel of a grid and learn a 2D convolutional neural net (CNN) using a bird's eye view representation [4, 21, 44, 28, 43]. As such procedure can potentially result in loss of details, especially when the points are sampled non-uniformly, deep network architectures have hence been designed to directly handle point data as inputs. Qi et al. [24] propose PointNet which applies a fully connected network on each point individually and a permutation-invariant max-pooling operation to aggregate global information. To leverage local neighborhood and hierarchical information passing, PointNet++ [26] adds grouping and sampling layers to perform stagewise aggregation, which is similar to pooling layers in regular CNNs. [23, 16] demonstrate the effectiveness of the PointNet-based architecture on 3D object detectors. Inspired by the SIFT feature extractor [20], Jiang et al. propose PointSIFT [11], which uses local octant directional vectors as feature extraction layers showing good results on point cloud segmentation tasks.

Grouping local neighborhood of points and aggregating information using permutation invariant operators, as done in PointNet++ [26], are special cases of graph neural networks (GNNs) [3, 2, 12, 18], where node interactions are modeled using a neural network. Point clouds can be treated as a sparse graph where edges denote two points which are close. 3D-GNN [27] uses a GNN to approximate messaging passing in point clouds. ECC [33] and EdgeConv [39] propose to generate the edge weights through a neural network. KD-Net [13] recursively processes hierarchical graph structures through a KD-Tree. [32] uses learnable kernel anchor points to smooth out local neighborhoods.

Another line of work views point clouds as discrete samples of a continuous function in space. Various parameterizations of learnable continuous filter functions have thus been proposed. Wang *et al.* [38] and Hermosilla *et al.* [9] propose to use a multi-layer perceptron (MLP) to represent the convolution filter function. The MLP takes in an offset vector towards the center of the local neighborhood, and outputs the value of the filter function at that location. ContFuse [19] is a memory efficient successor of [38] as it directly predicts the output features through an MLP. Other

families of learnable filter functions have also been studied, e.g. radial basis function (RBF) [31] and polynomial function kernels [42]. To prevent the function value from growing unbounded at a large distance, a step function is applied in [42] to make sure the filter function is zero outside a certain radius. In contrast, instead of predicted by a parametric function [19, 38, 9, 31, 42], our 3D filters have learnable weights at well defined 3D positions, which are potentially more robust and sample efficient.

Motivated by 3D convolution with grid structured filters, Atzmon et al. [1] propose "extension" and "restriction" operators. First, the extension operator interpolates point features onto a grid structure; then 3D convolution is applied on the grids; finally, the restriction operator projects convolved features back to point locations. Similarly, SPLATNet [34] extends points onto lattice grids, and PointwiseCNN [10] discretizes points into filter bins. These extension operators could potentially suffer from the loss of local geometric information due to discretization. The design of our deformable filter convolution also takes inspiration from deformable convolution [6], extending model capacity with continuous spatial reasoning. Despite the main difference of applying on 2D images vs. 3D point clouds, our proposed operator does not resample point features, as was done in [6], but interpolates 3D filters at point locations. This particular design addresses the potential discretization issue mentioned above [1, 34]. In concurrent work KPConv [36] also tries to deform filters similarly to our approach.

3. Deformable Filter Convolution

In this section, we first define our deformable filter convolution operator in the context of spatially continuous functions with discrete samples. We then show equivariance properties of the proposed operator.

3.1. Deformable Filters

Let's consider a spatially continuous signal $f: \mathbb{R}^3 \mapsto \mathbb{R}^D$ to represent the features from a 3D world. f takes in a 3-D coordinate and returns the features at that location. Point clouds can be viewed as discrete samples of this continuous function. Convolution in the continuous domain can be written as:

$$h(\mathbf{y}) = (f * g)(\mathbf{y}) \equiv \int_{-\infty}^{\infty} f(\mathbf{x}) \cdot g(\mathbf{y} - \mathbf{x}) d\mathbf{x},$$
 (1)

where f is the original signal, and $g: \mathbb{R}^3 \mapsto \mathbb{R}^D$ is the 3D filter. Computing this integral is, however, difficult in most cases. To make this computation tractable, we discretize the signal and approximate the integral using a Monte Carlo estimate of the local neighborhood, which is valid as long

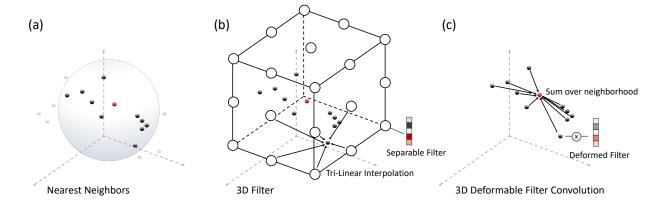


Figure 1. Overview of the proposed deformable filter convolution on 3D point cloud data. (a): At a given centroid point (red), a neighborhood of points are searched. (b): Each point in the neighborhood retrieves its own filter by performing tri-linear interpolation over the $3 \times 3 \times 3$ filter anchors. (c): The filter is multiplied with the point features, and summed towards the centroid point. To speed up computation, we use separable filters, and the spatial filter is of shape $3 \times 3 \times 3 \times D$, where D is the feature dimension.

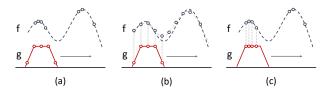


Figure 2. Our proposed deformable filter operator vs. extension operator [1]. (a): Convolution of point cloud signal f (under non-uniform sampling) and filter g. (b): Extension operator [1] resamples signal at grid locations, which can potentially lose local geometric details. (c): Our deformable filter convolution, which resamples the filter at positions of the point clouds, preserving local geometric details.

as the support of g is a subset of the neighborhood:

$$h(\mathbf{y}) \approx \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{y})} \mu(\mathbf{x}) f(\mathbf{x}) \cdot g(\mathbf{y} - \mathbf{x}),$$
 (2)

where $\mathcal{N}(\mathbf{y}) = \{\mathbf{x} : \|\mathbf{x} - \mathbf{y}\| \le r\}$ is the set of points in the neighborhood of \mathbf{y} , and μ is the measure of the volume covered by each neighboring point. For simplicity, we further assume that μ is some constant based on the approximation that points are uniformly sampled in local neighborhoods. Without loss of generality, we assume $\mu = 1$ since the actual value can be merged with the learned filter g.

Previous work proposed to represent the filter g(y - x) using an MLP [38, 9] or a polynomial function [42]. The potential issues with these continuous representations are 1) the filter kernel can be highly non-linear, and 2) depending on the point cloud distribution, the actual filter being used can vary significantly.

Different from previous approaches, we only parameterize the filter at discretized anchors X' that are coherent with the 3D grid structure. In contrast to [1], which "voxelizes"

the points into a grid structure and then projects them back to the original locations, we "deform" the standard 3D filter from the anchors towards the points (y - x). This leads to better preservation of local geometric information compared to feature voxelization as shown in Figure 2 for the 1D case. To deform 3D filters, we use an interpolation kernel $k(\cdot, \cdot)$,

$$\hat{g}(\mathbf{y} - \mathbf{x}) \approx \sum_{\mathbf{x}' \in X'} k(\mathbf{x}', \mathbf{y} - \mathbf{x}) g(\mathbf{x}').$$
 (3)

In practice, we choose to use a tri-linear interpolation kernel

$$k(\mathbf{x}, \mathbf{x}') = \prod_{d=1}^{3} \max \left(1 - \frac{|\mathbf{x}_d - \mathbf{x}'_d|}{a_d}, 0 \right), \quad (4)$$

where a_d is the filter grid unit length on dimension d. This interpolation scheme is continuous everywhere and naturally decays to zero when a point falls far away from the anchors, without using a manually designed step function, as was done in [42]. Tri-linear interpolation is easy to implement and unlike the Gaussian kernel, it does not have the gradient vanishing problem. In summary, our 3D deformable filter convolution operator can be written as:

$$h(\mathbf{y}) = \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{y})} f(\mathbf{x}) \cdot \left[\sum_{\mathbf{x}' \in X'} k(\mathbf{x}', \mathbf{y} - \mathbf{x}) g(\mathbf{x}') \right]. \quad (5)$$

Separable convolution: To make our convolution operator more efficient, in practice we consider using *separable convolution* [5], where we approximate $g(\mathbf{x}) \approx g_1(\mathbf{x})g_2$. $g_1 : \mathbb{R}^3 \mapsto \mathbb{R}$ is a spatial filter, and $g_2 \in \mathbb{R}^D$ is a *D*-dimensional vector. Figure 1 illustrates the overview of applying our deformable filter convolution on 3D point clouds.

3.2. Analysis

In this section we analyze various equivariance properties of the proposed convolution operator on point clouds. First, we show that our operator is translation-equivariant. Second, we show that our operator is permutation-equivariant under discretization of continuous signals. Equivariance could be more useful than invariance in certain scenarios as it preserves the transformation information. We start by defining the equivariance property mathematically.

Definition 1 (Equivariance). Let $\mathcal{T}_g^F: F \mapsto F$ be a transformation operator that produces a group action of g in a transformation group G on a function space F. An operator $L: F \mapsto H$ is said to be \mathcal{T}_G -equivariant if $L(\mathcal{T}_q^F(f)) = \mathcal{T}_q^H(L(f))$ for any $f \in F$, $g \in G$.

Translation equivariance is a desired property as it is an efficient way of sharing parameters that produces consistent outputs regardless of the location of the regions of interest. In short, if the input f is translated by an offset Δx , the effect on the output h is also a translation of Δx . CNNs have translation equivariance on 2D grid locations, which is one of the reasons they are successful in the image domain. Here, we show that our convolution operator is translation equivariant in a continuous domain with a d-dimensional coordinate system. In contrast, popular voxelization based approaches (e.g., [22, 4, 44, 1, 34]) are unfortunately not translation equivariant since the voxel grid is fixed and points can be assigned to different discretization bins.

Definition 2 (Translation operator). $\mathcal{T}_{\Delta \mathbf{x}}^F(\cdot): F \mapsto F$ is a translation operator on F if $\mathcal{T}_{\Delta \mathbf{x}}(f)(\mathbf{x}) = f(\mathbf{x} + \Delta \mathbf{x})$ for all $f \in F$, $\mathbf{x} \in dom(f)$.

Proposition 1 (Translation equivariance). Let $C_g(\cdot): F \mapsto H$ be the deformable filter convolution operator, where $F = \{f : \mathbb{R}^d \mapsto \mathbb{R}^{D'}\}$ is the set of input functions, $g: \mathbb{R}^d \mapsto \mathbb{R}^{D' \times D}$ is the convolution filter, and $H = \{h: \mathbb{R}^d \mapsto \mathbb{R}^D\}$ is the set of output functions. For all $\mathbf{y} \in \mathbb{R}^d$, $C_g(\mathcal{T}_{\mathbf{x}}^F(f))(\mathbf{y}) = \mathcal{T}_{\mathbf{x}}^H(C_g(f))(\mathbf{y})$.

Proof. See Appendix A.
$$\Box$$

Remark. Note that Proposition 1 is generalized to any coordinate system of the input points. In Cartesian coordinates of 3D space, d=3 and translation means the conventional translation, whereas translation in polar coordinates is equivalent to rotation in Cartesian coordinates.

When the inputs are point clouds, the input function is discretized by an input array, where each entry stores D'-dimensional features of the point. The output of the convolution operation is an array with output D-dimensional

features of the point. Permutation equivariance ensures that the ordering of the points in the input does not affect the output. PointNet [24] aggregates information using a global max-pooling, which is permutation-invariant but not equivariant. As a result, it cannot aggregate local neighborhood information.

We first define the permutation operator on the set of functions with integer domain. Note that all arrays can be represented as a function that maps from positive integers to numbers.

Definition 3 (Permutation operator). Let $F = \{f : dom(f) = X \subseteq \mathbb{Z}\}$, where \mathbb{Z} is the set of integers. Given a permutation $s \in Sym(X)$, $\mathcal{P}_s^F(\cdot) : F \mapsto F$ is a permutation operator if $\mathcal{P}_s^F(f)(i) = f(s(i))$ for all $i \in X$.

Now consider an array of M input points $p : \mathbb{Z}_M \to \mathbb{R}^O$. The neighborhood function on this discretized domain is defined as $\mathcal{N}(i) = \{j : ||p(j) - p(i)|| \le r\}$.

Proposition 2 (Permutation equivariance). Let $\tilde{C}_g(\cdot,\cdot)$: $P \times F \mapsto H$ be the deformable filter convolution operator on discretized inputs of M points, where $P = \{p : \mathbb{Z}_M \mapsto \mathbb{R}^d\}$ is the set of input coordinate arrays, $F = \{f : \mathbb{Z}_M \mapsto \mathbb{R}^D' \}$ is the set of input feature arrays, $g : \mathbb{R}^d \mapsto \mathbb{R}^{D' \times D}$ is the convolution filter, and $H = \{h : \mathbb{Z}_M \mapsto \mathbb{R}^D \}$ is the set of output feature arrays. For all $i \in \mathbb{Z}_M$, $s \in Sym(M)$, $C_g(\mathcal{P}_s^{P \times F}(p,f))(i) = \mathcal{P}_s^H(C_g(p,f))(i)$.

Proof. See Appendix A.
$$\Box$$

4. Experimental Evaluation

We verify the effectiveness of our proposed operator on a suite of point cloud benchmarks and tasks including semantic segmentation, object detection, and object classification.

4.1. TOR4D point cloud segmentation

First, we verify the usefulness of the proposed deformable filter convolution on the task of LiDAR semantic segmentation. We report results on the TOR4D dataset [21, 38], which consists of 1,239,706 frames in training, 123,975 frames in validation and 123,475 frames in test set. The dataset contains 7 object classes: "vehicle", "bicyclist", "pedestrian", "motorcycle", "background", "animal", and "road". We omit the "animal" class for evaluation due to the small number of examples.

Point cloud convolution using voxel features: We adopt a UNet architecture [30] to extract voxelized features and then add our point convolution layer on top. One baseline approach is to use a voxelized network only, which suffers from the precision of the output, since points belonging to the same voxel will have the same output. To predict on the point level, one needs to fuse the voxelized feature onto

Method	mIOU	Vehicle	Bicyclist	Pedestrian	Motorcycle	Background	Road
PointNet [24]	46.00	76.73	2.85	6.62	8.02	89.83	91.96
3D-FCN [38]	57.33	86.74	22.30	38.26	17.22	86.91	92.56
3D-FCN +PCC [38]	67.53	91.83	40.23	47.74	42.91	89.27	93.18
2D-U-Net [47]	60.73	91.15	27.41	51.44	41.19	92.45	87.82
3D-U-Net [47]	67.15	91.29	43.35	43.91	45.01	92.40	86.96
U-Net [30]	73.69	91.78	52.18	60.32	51.01	91.53	95.32
+ContFuse [19]	76.93	94.81	58.72	78.53	38.01	94.94	96.59
+DeformFilter (Ours)	79.19	94.93	60.86	77.96	50.16	94.71	96.50

Table 1. TOR4D point cloud semantic segmentation results on test set

Method		Car			Pedestrian			Cyclist	
Method	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
HDNet [43]	90.67	86.99	85.32	73.23	69.88	66.41	78.41	64.14	60.80
+PCC [38]	92.67	87.46	84.71	73.49	70.43	66.19	74.18	59.84	57.17
+DeformFilter (Ours)	92.71	87.30	85.48	75.24	71.35	67.31	79.57	65.41	62.86

Table 2. KITTI BEV object detection results on val set

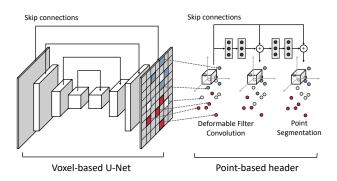


Figure 3. Our point cloud segmentation network contains a voxelized backbone U-Net and a point-based header using our proposed deformable filter convolution layers.

point clouds. We consider the continuous fusing operator (ContFuse) proposed in [19] as a strong baseline. This operator is similar to what has been done in PointNet++ [26], where each point first queries a local neighborhood, then passes the neighboring point features plus coordinate offsets through an MLP, and finally averages the neighborhood features together. Note that our approach is much more memory efficient than the ContFuse operation, since we do not need to tile the neighboring point features into a larger size tensor. PCC [38] also adopts the same setting where they found the best performance when their continuous convolution layers are composed on top of the voxelized features. Figure 3 illustrates the overall network architecture, where the point-based convolution header takes inputs from a voxelized feature extractor.

Implementation details: The voxel feature extractor uses a standard U-Net [30] with [32, 64, 128, 256] channels in the encoder and decorder. Each encoder block contains convolution, batch norm, ReLU, and max pooling layers.

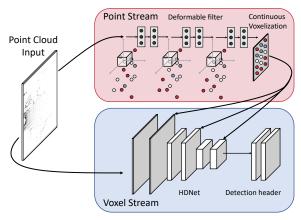


Figure 4. Our point cloud detection network fuses point features into various stages of the voxelized backbone network.

The decoder has a symmetric structure, with max-pooling replaced by bilinear interpolation to upsample the feature map. Skip connections are added between each pair of corresponding layers connecting the encoder and decoder. Similar to [38], the ContFuse baseline has 7 blocks in the point-based header, each containing an MLP of size [11, 8, 8, 8], where the inputs are concatenated with the offset coordinates of the neighborhood (3-dimensional) and the output classes (7 plus none of the above). There are skip connections combining the outputs of these blocks. Our deformable version has 2 blocks, each with [8, 16, 32] channels, and skip connections are also applied on the output channels. We used $3\times3\times3$ convolution filters with filter grid unit length 0.2m and neighborbood size 16. We used the Adam optimizer with initial learning rate 1e-4, weight decay 5e-4, batch size 16 and 0.1× learning rate decay at 50k and 100k iterations. The baseline U-Net was trained for 450k iterations, and ContFuse and our deformable filter version were trained for 115k iterations.

Results and discussion: Results on TOR4D test set are shown in Table 1 and qualitative visualization of the output results shown in Figure 7. Our joint network using deformable filter layers significantly surpasses the baseline, achieving state-of-the-art performance. Notably, our deformable filter model outperforms the ContFuse baseline, which uses an MLP to aggregate point cloud neighborhood information. To understand filter activations, we also plot the most activated region of each filter channel using guided backprop [46], shown in Figure 6. It is clear that the regions that activate individual neurons the most are roughly corresponding to the semantic classes.

4.2. KITTI BEV object detection

We evaluate our deformable filter convolution on KITTI BEV object detection benchmark [7], consisting of 7,481 training and 7,518 testing frames of LiDAR point clouds. Half of the training data is split for validation. Detectors on "car", "pedestrian" and "cyclist" categories are trained and evaluated individually.

We adopt the HDNet architecture proposed in [43], one of the top-performing object detectors on this benchmark. HDNet is a voxelized network that performs regular 2D convolution on BEV with the z-axis to be the channel dimension. It also predicts the map information with a pretrained module. To preserve local geometric details and to provide extra information to the voxelized backbone network, we add a deformable filter input branch that processes raw LiDAR points and the output of the branch is fused with the backbone network. Shown in Figure 4, we add 3 layers of deformable filter convolution to process the point cloud, with channel dimension [4, 16, 32] respectively. The last deformable filter layer samples the output points at the voxel centers so that we can concatenate the feature with the voxel input branch. We compare our proposed operator with parametric continuous convolution (PCC) [38], which uses an MLP to predict the convolution filter weights. The MLP takes 3-d coordinate inputs and outputs the elementwise seperable filter weight.

Implementation details: The backbone network consists of five residual blocks with [2, 4, 8, 12, 12] convolution layers with channel dimensions [32, 64, 128, 192, 256] respectively. The initial convolution for each residual block has stride 2 to downsample the feature map. The grid unit lengths are 0.15m for car, 0.5m/0.2m/0.1m/0.05m for pedestrian, and 0.33m for cyclist. We use $3\times3\times3$ filters for car and pedestrian, and $7\times7\times7$ filters for cyclist. NMS thresholds are 0.1, 0.3, and 0.5 for car, pedestrian, cyclist respectively. For data augmentation we apply random scaling of $0.9\times$ to $1.1\times$ random rotation of -5 to 5 degrees along z-axis and random translation of -5 to 5 meters for x and y

Method	# points	Acc.
DeepSets [45]	1000	87.1
ECC [33]	1000	87.4
PointNet [24]	1024	89.2
FlexConv [8]	1024	90.2
SpiderCNN [42]	1024	90.5
kd-net [13]	1024	90.6
PointNet++ [26]	1024	90.7
SO-Net [15]	2048	90.9
MCConv [9]	1024	90.9
PointCNN [17]	1024	92.2
DGCNN [40]	1024	92.2
KPConv [36]	6800	92.9
Ours Best	1024	91.7

Table 3. ModelNet-40: Classification results

Method	Acc.
PointNet [24]	89.2
+DeformFilters (Ours)	+1.8
PointNet++ [26]	90.7
+DeformFilters (Ours)	+1.0

Table 4. ModelNet-40: Gain in accuracy

axes. Models are trained using SGD with momentum for 50 epochs with mini-batch size 16. The initial learning rate is 0.01 with $0.1 \times$ learning rate decay at the 30th and 45th epoch; weight decay constant is 2e-4.

Results and discussion: In Table 2 we show results on KITTI validation set where we compare our proposed operator with the HDNet baseline and PCC [38]. Overall, our method has the best performance across all three categories, especially on pedestrian and cyclist. Notably, PCC has a negative impact on the cyclist detector, possibly due to less training data and sparser point clouds for positive examples, whereas our method delivers a significant gain over the baseline network. In Figure 5, we visualize the 3D filters learned on our cyclist detector, compared to the ones learned by PCC. Our filters learn more expressive 3D shapes that are not linearly separable.

4.3. ModelNet 40 classification

To further verify the effectiveness of our proposed operator, we evaluate on ModelNet [41], a standard point cloud classification benchmark. ModelNet contains CAD models of 40 categories, 9,843 shapes for training and 2,468 for testing.

Implementation details: We modified the PointNet [24] and PointNet++ [26] architectures to incorporate our deformable filter layers. The inputs to these networks are xyz coordinates (3 channels). In the original PointNet,

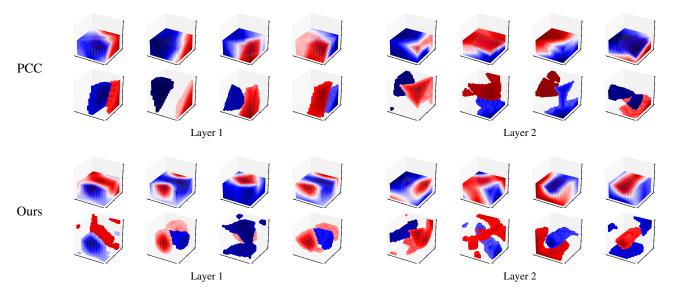


Figure 5. Visualization of our learned 3D filters $(7 \times 7 \times 7)$ trained on KITTI cyclist detection, compared to filters output by an MLP (PCC) [38]. The upper row is the full filter dissected by half across z-axis; the lower row is the top and bottom 10% quantile of the filter weights. Red denotes positive filter weights and blue negative.

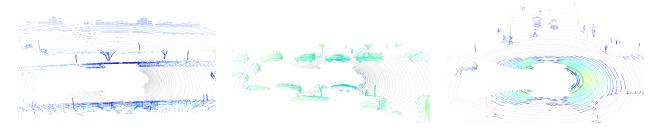


Figure 6. Visualization of different learned filter channels using guided backprop

we replace the pointwise fully connected layers with deformable filter convolution layers. The resulting architecture has the same number of channels ([3, 64, 64, 64, 128, 1024]). In PointNet++, since our convolution layer operates on a different neighborhood compared to the sampling and grouping layers, we augment the original architecture with a residual branch that contains the deformable filter layers with the same number of hidden units compared to the MLP network in the original network. We use $3\times3\times3$ convolution filters with filter grid unit length 0.2 for the PointNet architecture, and [0.2, 0.4, 0.8] for each downsample stage of the PointNet++ architecture. We fix the neighborhood size to be 8. We use mini-batch size 16, and base learning rate 1e-3 with an exponential decay of 0.7× every 20 epochs. In both experiments, we use the standard 1,024 points with furthest point samples as inputs to the network, and for fair comparison with baselines we do not aggregate the final prediction from multiple votes.

Results and discussion: As shown in Table 3 and 4, by simply adding our deformable filter layers in standard PointNet-based architecture, we observe a reasonable in-

crease in performance, comparable to other competitive approaches using point cloud inputs.

5. Conclusion

This paper presents deformable filter convolution, a learnable convolution layer that combines voxel-like filtering and spatially continuous reasoning. It naturally augments existing top-performing voxel-based network architectures by fusing point features into input and output branches. We show significant gain in performance of the joint network, compared to voxel only baselines and other point-based convolution approaches. The proposed convolution operator enables us to achieve state-of-the-art performance on LiDAR semantic segmentation. As future work, we plan to integrate our proposed convolution layer as a fundamental building block of end-to-end point-based networks, without resorting to voxelized feature maps.

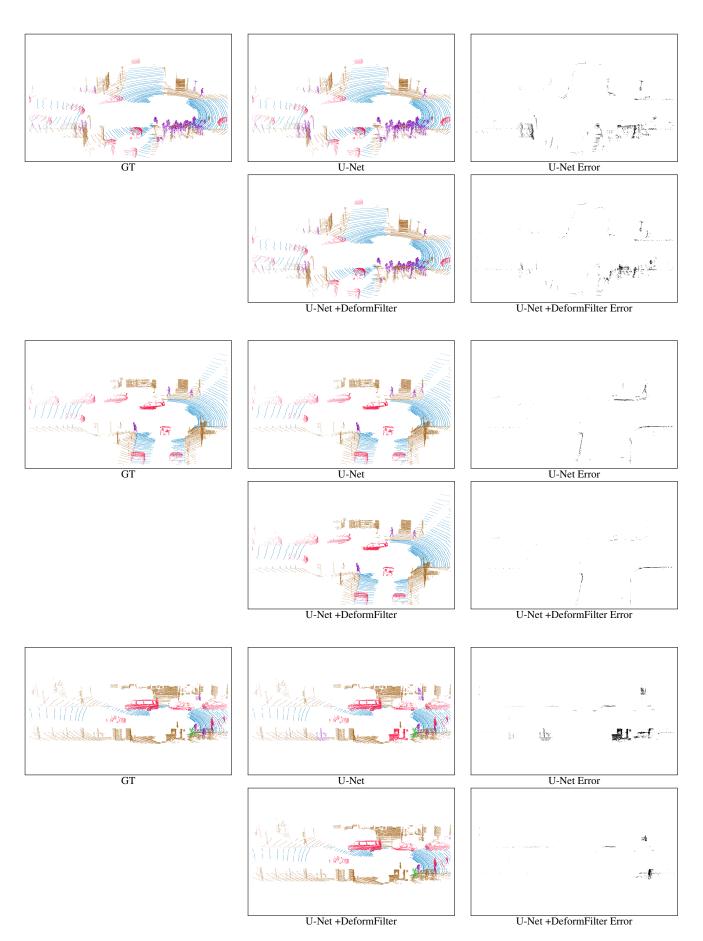


Figure 7. Visualization of results on TOR4D LiDAR semantic segmentation

References

- [1] M. Atzmon, H. Maron, and Y. Lipman. Point convolutional neural networks by extension operators. *ACM Trans. Graph.*, 37(4):71:1–71:12, 2018. 2, 3, 4
- [2] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. *Comput. Graph. Forum*, 34(5):13–23, 2015.
- [3] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In Proceedings of the 2nd International Conference on Learning Representations (ICLR), 2014. 2
- [4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6526–6534, 2017. 1, 2, 4
- [5] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017. 3
- [6] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 764–773, 2017.
- [7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3354–3361, 2012.
- [8] F. Groh, P. Wieschollek, and H. P. A. Lensch. Flex-convolution (million-scale point-cloud learning beyond grid-worlds). In *Asian Conference on Computer Vision (ACCV)*, Dezember 2018. 6
- [9] P. Hermosilla, T. Ritschel, P. Vázquez, A. Vinacua, and T. Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Trans. Graph.*, 37(6):235:1–235:12, 2018. 1, 2, 3, 6
- [10] B. Hua, M. Tran, and S. Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference* on Computer Vision and Pattern Recognition (CVPR), pages 984–993, 2018. 2
- [11] M. Jiang, Y. Wu, and C. Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *CoRR*, abs/1807.00652, 2018. 2
- [12] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. 2017. 2
- [13] R. Klokov and V. S. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 863–872, 2017. 2, 6
- [14] B. Li, T. Zhang, and T. Xia. Vehicle detection from 3d lidar using fully convolutional network. In *Robotics: Science and Systems XII*, 2016. 1

- [15] J. Li, B. M. Chen, and G. H. Lee. So-net: Selforganizing network for point cloud analysis. arXiv preprint arXiv:1803.04249, 2018. 6
- [16] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS), pages 828–838, 2018. 1, 2
- [17] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In Advances in Neural Information Processing Systems, pages 820–830, 2018. 6
- [18] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel. Gated graph sequence neural networks. In *Proceedings of the* 4th International Conference on Learning Representations (ICLR), 2016. 2
- [19] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the 15th European Conference on Computer Vision (ECCV)*, pages 663–678, 2018. 1, 2, 5
- [20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 2
- [21] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3569–3577, 2018. 2, 4
- [22] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. 1, 2, 4
- [23] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 918–927, 2018. 2
- [24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision* and Pattern Recognition (CVPR), pages 77–85, 2017. 1, 2, 4, 5, 6
- [25] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5648–5656, 2016. 1
- [26] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS), pages 5105–5114, 2017. 1, 2, 5, 6
- [27] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun. 3d graph neural networks for rgbd semantic segmentation. In *Proceedings* of the IEEE International Conference on Computer Vision (ICCV), pages 5199–5208, 2017. 2
- [28] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun. Sbnet: Sparse blocks network for fast inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (CVPR), pages 8711–8720, 2018. 1, 2

- [29] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6620–6629, 2017. 1, 2
- [30] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the 18th International Conference Medical Image Computing and Computer-Assisted Intervention (MIC-CAI)*, pages 234–241, 2015. 4, 5
- [31] K. Schütt, P. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS), pages 992–1002, 2017. 1, 2
- [32] Y. Shen, C. Feng, Y. Yang, and D. Tian. Mining point cloud local structures by kernel correlation and graph pooling. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4548–4557, 2018.
- [33] M. Simonovsky and N. Komodakis. Dynamic edgeconditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Com*puter Vision and Pattern Recognition (CVPR), pages 29–38, 2017. 2, 6
- [34] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M. Yang, and J. Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2530–2539. 2, 4
- [35] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Con*ference on Computer Vision (ICCV), pages 945–953, 2015.
- [36] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *arXiv preprint arXiv:1904.08889*, 2019. 2, 6
- [37] P. Wang, Y. Liu, Y. Guo, C. Sun, and X. Tong. O-CNN: octree-based convolutional neural networks for 3d shape analysis. ACM Trans. Graph., 36(4):72:1–72:11, 2017. 1,
- [38] S. Wang, S. Suo, W. Ma, A. Pokrovsky, and R. Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, pages 2589–2597, 2018. 1, 2, 3, 4, 5, 6, 7
- [39] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.
- [40] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019. 6
- [41] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer*

- Vision and Pattern Recognition (CVPR), pages 1912–1920, 2015. 6
- [42] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the 15th European Conference on Computer Vision (ECCV)*, pages 90–105, 2018. 1, 2, 3, 6
- [43] B. Yang, M. Liang, and R. Urtasun. HDNET: exploiting HD maps for 3d object detection. In *Proceedings of the 2nd Annual Conference on Robot Learning (CoRL)*, pages 146–155, 2018. 2, 5, 6
- [44] B. Yang, W. Luo, and R. Urtasun. PIXOR: real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7652–7660, 2018. 1, 2, 4
- [45] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS), pages 3394–3404, 2017. 1, 6
- [46] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proceedings of the 13th Euro*pean Conference Computer Vision (ECCV), pages 818–833, 2014. 6
- [47] C. Zhang, W. Luo, and R. Urtasun. Efficient convolutions for real-time semantic segmentation of 3d point clouds. In *Pro*ceedings of the 6th International Conference on 3D Vision (3DV), pages 399–408, 2018. 1, 5

A. Technical proofs

Proposition 1 (Translation equivariance).

Proof. Let $\tilde{\mathbf{x}} = \mathbf{x} - \Delta \mathbf{x}$, we first show that the neighborhood can be translated:

$$\mathcal{N}(\mathbf{y} + \Delta \mathbf{x}) = \{\mathbf{x} : \mathbf{x} \in N(\mathbf{y} + \Delta \mathbf{x})\}$$

$$= \{\mathbf{x} : ||\mathbf{x} - \mathbf{y} - \Delta \mathbf{x}|| \le r\}$$

$$= \{\tilde{\mathbf{x}} + \Delta \mathbf{x} : ||\tilde{\mathbf{x}} - \mathbf{y}|| \le r\}$$

$$= \{\tilde{\mathbf{x}} + \Delta \mathbf{x} : \tilde{\mathbf{x}} \in \mathcal{N}(\mathbf{y})\}.$$
(6)

For all $\mathbf{y} \in \mathbb{R}^d$,

$$\mathcal{T}_{\Delta\mathbf{x}}^{H}(C_{g}(f))(\mathbf{y}) = h(\mathbf{y} + \Delta\mathbf{x})$$

$$= \sum_{\mathbf{x} \in N(\mathbf{y} + \Delta\mathbf{x})} f(\mathbf{x}) \cdot \left[\sum_{\mathbf{x}' \in X'} k(\mathbf{x}', \mathbf{y} + \Delta\mathbf{x} - \mathbf{x}) g(\mathbf{x}') \right]$$

$$= \sum_{\tilde{\mathbf{x}} \in N(\mathbf{y})} f(\tilde{\mathbf{x}} + \Delta\mathbf{x}) \cdot \left[\sum_{\mathbf{x}' \in X'} k(\mathbf{x}', \mathbf{y} - \tilde{\mathbf{x}}) g(\mathbf{x}') \right]$$

$$= \sum_{\tilde{\mathbf{x}} \in N(\mathbf{y})} \mathcal{T}_{\Delta\mathbf{x}}^{F}(f)(\tilde{\mathbf{x}}) \cdot \left[\sum_{\mathbf{x}' \in X'} k(\mathbf{x}', \mathbf{y} - \tilde{\mathbf{x}}) g(\mathbf{x}') \right]$$

$$= C_{g}(\mathcal{T}_{\Delta\mathbf{x}}^{F}(f))(\mathbf{y}). \tag{7}$$

Proposition 2 (Permutation equivariance).

Proof. Let s be any element in $\mathrm{Sym}(M)$. Since s is bijective, let $\tilde{j}=s^{-1}(j), s(N(i))\equiv\{s(j):j\in N(i)\},\ \tilde{N}=s^{-1}(N(s(i))).$ For all $i\in\mathbb{Z}_M$,

$$\mathcal{P}_{s}^{H}(C_{g}(p,f))(i) = h(s(i))$$

$$= \sum_{j \in N(s(i))} f(j) \cdot \left[\sum_{\mathbf{x}' \in X'} k(\mathbf{x}', p(s(i)) - p(j)) g(\mathbf{x}') \right]$$

$$= \sum_{\tilde{j} \in \tilde{N}} f(s(\tilde{j})) \cdot \left[\sum_{\mathbf{x}' \in X'} k(\mathbf{x}', p(s(i)) - p(s(\tilde{j})) g(\mathbf{x}') \right]$$

$$= \sum_{\tilde{j} \in \tilde{N}} \mathcal{P}_{s}^{F}(f)(\tilde{j}) \cdot \left[\sum_{\mathbf{x}' \in X'} k(\mathbf{x}', \mathcal{P}_{s}^{P}(p)(i) - \mathcal{P}_{s}^{P}(p)(\tilde{j}) g(\mathbf{x}') \right]$$

$$= C_{g}(\mathcal{P}_{s}^{P \times F}(p, f))(i). \tag{8}$$

B. KITTI detection results

Figure 8 shows vehicle detection results on the KITTI dataset.

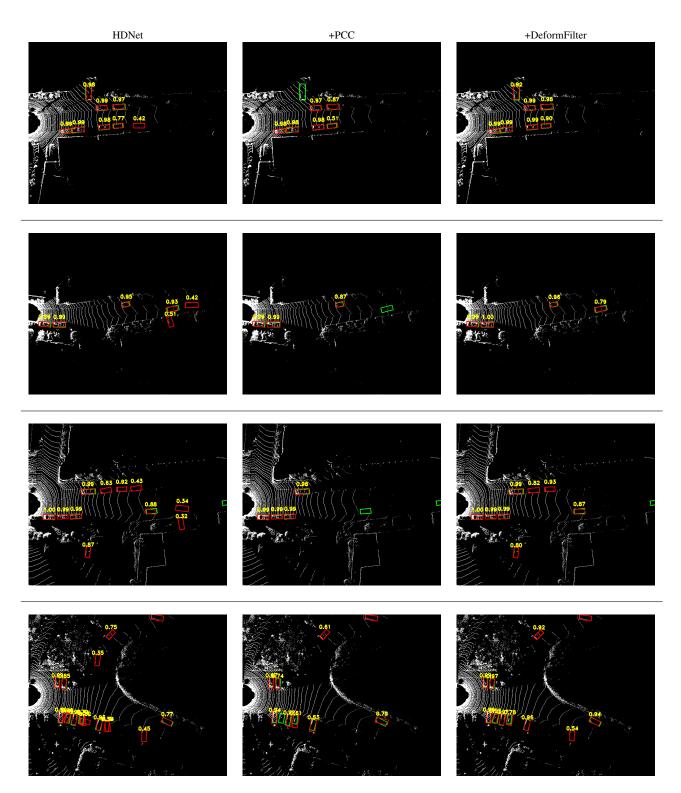


Figure 8. Visualization of results on KITTI BEV LiDAR detection on cars