

FPConv: Learning Local Flattening for Point Convolution

Yiqun Lin^{1,2}, Zizheng Yan^{1,2}, Haibin Huang⁴, Dong Du^{2,3}, Ligang Liu³,
Shuguang Cui^{1,2}, Xiaoguang Han^{*1,2}

¹The Chinese University of Hong Kong, Shenzhen

²Shenzhen Research Institute of Big Data

³University of Science and Technology of China ⁴Kuaishou Technology

Abstract

We introduce FPConv, a novel surface-style convolution operator designed for 3D point cloud analysis. Unlike previous methods, FPConv doesn't require transforming to intermediate representation like 3D grid or graph and directly works on surface geometry of point cloud. To be more specific, for each point, FPConv performs a local flattening by automatically learning a weight map to softly project surrounding points onto a 2D grid. Regular 2D convolution can thus be applied for efficient feature learning. FPConv can be easily integrated into various network architectures for tasks like 3D object classification and 3D scene segmentation, and achieve comparable performance with existing volumetric-type convolutions. More importantly, our experiments also show that FPConv can be a complementary of volumetric convolutions and jointly training them can further boost overall performance into state-of-the-art results. Code is available at <https://github.com/lyqun/FPConv>

1. Introduction

With the rapid development of 3D scan devices, it is more and more easy to generate and access 3D data in the form of point clouds. This also brings the challenging of robust and efficient 3D point clouds analysis, which serves as an important component in many real world applications like robotics navigation, autonomous driving, augmented reality applications and so on [35, 52, 3, 38].

Despite decades of development in 3D analysing technologies, it is still quite challenging to perform point cloud based semantic analysis, largely due to its sparse and unordered structure. Early methods [7, 8, 11, 28] utilized hand-crafted features with complex rules to tackle this prob-

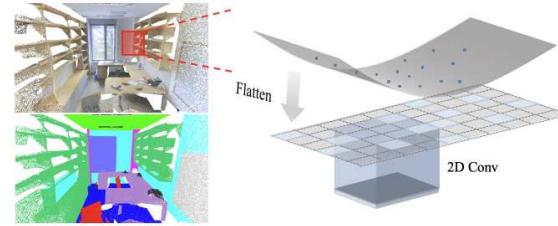


Figure 1: Flattening Projection Convolution: flatten local patch onto a grid plane and then apply 2D convolutions.

lem. Such empirical human-designed features would suffer from limited performance in general scenes. Recently, with the explosive growth of machine learning and deep learning techniques, Deep Neural Network (CNN) based methods have been introduced into this task [36, 37] and reveal promising improvements. However, both PointNet [36] and PointNet++ [37] doesn't support convolution operation which is a key contributing factor in Convolutional Neural Network (CNN) for efficient local processing and handling large-scale data.

A straightforward extension of 2D CNN is treating 3D space as a volumetric grid and using 3D convolution for analysis [49, 39]. Although these approaches have achieved success in tasks like object classification and indoor semantic segmentation [30, 9], they still have limitations like cubic growth rate of memory requirement and high computational cost, leading to insufficient analysis and low prediction accuracy on large-scale scenes. Recently, [48, 44] are proposed to approximate such volumetric convolutions with point-based convolution operations, which greatly improves the efficiency and preserves the output accuracy. However, these methods are still difficult to capture fine details on surface with relatively flat and thin structures.

In reality, data captured by 3D sensors and LiDAR are

*Corresponding Author: hanxiaoguang@cuhk.edu.cn

usually sparse that points fall near scene surfaces and almost no points interior. Hence, surfaces are more natural and compact for 3D data representation. Towards this end, works like [10, 51] establish connections among points and apply graph convolutions in the corresponding spectral domain or focus on the surface represented by the graph [40], which are usually impractical to create and sensitive to local topological structures.

More recently, [43, 33, 18] are proposed to learn convolution on a specified 2D plane. Inspired by these pioneering works, we develop FPCConv, a new convolution operation for point clouds. It works directly on local surface of geometry without any intermediate grid or graph representation. Similar to [43], it works in projection-interpolation manner but more general and implicit. Our key observation is that projection and interpolation can be simplified into a single weight map learning process. Instead of explicitly projecting onto the tangent plane [43] for convolution, FPCConv learns how to diffuse convolution weights of each point along the local surface, which is more robust to various input data and greatly improves the performance of surface-style convolution.

As a local feature learning module, FPCConv can be further integrated with other operations in classical nerual network architectures and works on various analysis tasks. We demonstrate FPCConv on 3D object classification as well as 3D scene semantic segmentation. Networks with FPCConv outperform previous surface-style approaches [43][18][33] and achieves comparable results with current start-of-the-art methods. Moreover, our experiments also shows that FPCConv performs better at regions that are relatively flat thus can be a complementary to volumetric-type works and joint training helps to boost the overall performance into state-of-the-art results.

To summarize, the main contributions of this work are as follows:

- FPCConv, a novel surface-style convolution for efficient 3D point cloud analysis.
- Significant improvements over previous surface-style convolution based methods and comparable performance with state-of-the-art volumetric-style methods in classification and segmentation tasks.
- An in-depth analysis and comparison between surface-style and volumetric-style convolution, demonstrating that they are complementary to each other and joint training boosts the performance into state-of-the-art.

2. Related Work

Deep learning based 3D data analysis has been a quite hot research topic in recent years. In this section, we mainly

focus on point cloud analysis and briefly review previous works according to their underling methodologies.

Volumetric-style point convolution Since a point cloud disorderly distributes in a 3D space without any regular structures, pioneer works sample points into grids for conventional 3D convolutions apply, but limited by high computational load and low representation efficiency [30, 49, 39, 41]. PointNet [36] proposes a shared MLP on every point individually followed by a global max-pooling to extract global feature of the input point cloud. [37] extends it with nested partitionings of point set to hierarchically learn more local features, and many works follow that to approximate point convolutions by MLPs [24, 25, 16, 46]. However, adopting such a representation can not capture the local features very well. Recent works define explicit convolution kernels for points, whose weights are directly learned like image convolutions [17, 50, 12, 2, 44]. Among them, KPConv [44] proposes a spatially deformable point convolution with any number of kernel points which alleviates both varying densities and computational cost, outperform all associated methods on point analysis tasks. However, there volumetric-style approaches may not capture uniform areas very well.

Graph-style point convolution When the relationships among points have been established, a Graph-style convolution can be applied to explore and study point cloud more efficiently than volumetric-style. Convolution on a graph can be defined as convolution in its spectral domain. [6, 15, 10]. ChebNet [10] adopts Chebyshev polynomial basis for representing the spectral filters to alleviate the cost of explicitly computing the graph Fourier transform. Furthermore, [20] uses a localized first-order approximation of spectral graph convolutions for semi-supervised learning on graph-structured data which greatly accelerates calculation efficiency and improves classification results. However, these methods are all depending on a specific graph structure. Then [51] introduces a spectral parameterization of dilated convolution kernels and a spectral transformer network, sharing information across related but different shape structures. In the meantime, [29, 5, 40, 32] focus on graph learning on manifold surface representation to avoid the spectral domain operation, while [45, 47] learn filters on edge relationships instead of points relative positions. Although a graph convolution combines features on local surface patches and can be invariant to the deformations in Euclidean space. However, reasonable relationships among distinct points are not easy to establish.

Surface-style point convolution Since data captured by 3D sensors typically represent surfaces, another mainstream approach attempts to operate directly on surface geometry. Most works project a shape surface consist of points to an intermediate grid structure, e.g. multi-view RGB-D images, following with conventional convolutions

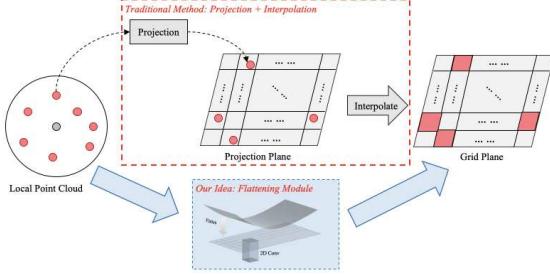


Figure 2: Flattening module compared with traditional method: we design a module to learn local flattening directly instead of learning projection and interpolation separately.

[13, 26, 31, 4, 22]. Such methods often suffer from the redundant representation of multi-view and the ambiguity caused by different viewpoints. [43] proposes projecting local neighborhoods of each point to its local tangent plane and processing them with 2D convolutions, which is efficient for analyzing dense point clouds of large-scale and outdoor environments. However, this method relies heavily on point tangent estimation, and this linear projection is not always optimal for complex areas. [33] optimizes the calculation with parallel tangential frames, while [18] utilizes a 4-rotational symmetric field to define a domain for convolution on surface, which not only increase the robustness, but also make the utmost of detailed information. However, existing surface-style learning algorithms cannot perform very well on challenge datasets such as S3DIS [1] and ScanNet [9], since they lose 1-dimensional information and they cannot estimate the surface accurately.

Our method is inspired by surface-style point convolutions. The network learns a non-linear projection for each local patch, say flattening the local neighborhood points into a 2D grid plane. Then 2D convolutions can be applied for feature extraction. Although learning on surface will lose 1-dimensional information, FPConv still achieves comparable performance with existing volumetric-style convolutions. In addition, our FPConv can be integrated into volumetric-style convolution and achieve state-of-the-art results.

3. FPConv

In this section, we formally introduce FPConv. We first revisit the definition of convolution along point cloud surface and then show it can be simplified into a weight learning problem under discrete setting. All derivations are provided in the form of point clouds.

3.1. Learn Local Flattening

Notation: Let p be a point from a point cloud \mathcal{P} and $\mathcal{F}(p)$ be a scalar function defined over points. Here $\mathcal{F}(p)$ can encode signals like color, geometry or features from intermediate network layers. We denote $\mathcal{N}(p)$ as a local point cloud patch centered at p where $\mathcal{N}(p) = \{q_i = q_i - p \mid \|q_i - p\|_2 < \rho, q_i \in \mathcal{P}\}$ with $\rho \in \mathbb{R}$ being the chosen radius.

Convolution on local surface: In order to convolve \mathcal{F} around the surface, we first extend it to a continuous function over a continuous surface. We introduce a virtual 2D plane \mathcal{S} with a continuous signal $\mathcal{S}(u)$ together with a map $\pi(\cdot)$ which maps $\mathcal{N}(p)$ onto \mathcal{S} and

$$\mathcal{S}(\pi(q_i)) = \mathcal{F}(q_i) \quad (1)$$

The convolution at p is defined as:

$$\mathcal{X}(p) = \int_{\mathcal{S}} c(u) \mathcal{S}(u) du \quad (2)$$

where $c(u)$ is a convolution kernel. We now describe how to formulate the above convolution into a weight learning problem.

Local flattening by learning projection weights: As shown in Eq.3, with $\pi(\cdot)$, $\mathcal{N}(p)$ will be mapped as scattered points in \mathcal{S} , thus we need a interpolation method to estimate the full signal function $\mathcal{S}(u)$, as shown in Eq.3.

$$\mathcal{S}(u) = \sum_i w(u, \pi(q_i)) \mathcal{S}(\pi(q_i)) \quad (3)$$

Now if we discretize \mathcal{S} into a grid plane with size of $M_w \times M_h$. For each grid $\mathcal{S}(v_j)$, where j in $\{1, 2, \dots, M_w \times M_h\}$, we can have from Eq.1 and Eq.3:

$$\mathcal{S}(v_j) = \sum_i w_{ji} \mathcal{F}(q_i) \quad (4)$$

Where $w_{ji} = w(v_j, \pi(q_i))$. Furthermore, we can rewrite Eq.2 in an approximate discretized form as:

$$\begin{aligned} \mathcal{X}(p) &= \int_{\mathcal{S}} c(u) \mathcal{S}(u) du \\ &= \sum_j c_j \sum_i w_{ji} \mathcal{F}(q_i) \\ &= M_c * (W_f^T \times F(p)) \end{aligned} \quad (5)$$

Where c_j is discretized convolution kernel weights, and j in $\{1, 2, \dots, M_w \times M_h\}$. Let $L = M_w \times M_h$, $W_f \in \mathbb{R}^{N \times L}$, $W_f(i, j) = w(v_j, \pi(q_i))$, and $F(p) =$

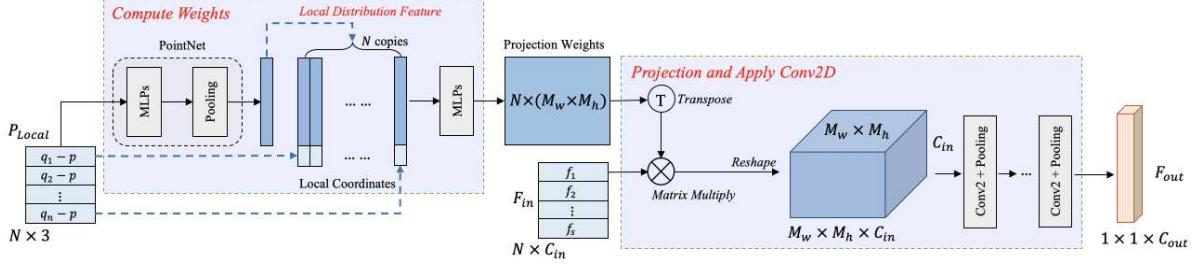


Figure 3: Process of conducting FPConv on local region centered around point p . The input coordinates and features come from N neighbor points randomly picked in a radius range of p . The output is F_{out} at p .

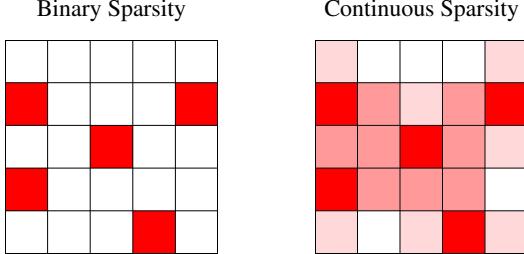


Figure 4: Left: binary sparsity, intensity at each position should be 0 or 1. Right: continuous sparsity, intensity can be in the range of 0 to 1.

$(\mathcal{F}(q_1), \dots, \mathcal{F}(q_N))^T \in \mathbb{R}^{N \times C}$. Now we can see that projection and interpolation can be combined into a single weight matrix W_f where it only depends on the point location w.r.t the center point.

3.2. Implementation

According to Eq.5, we can design a module to learn projection weights directly instead of learning projection and interpolation separately, as shown in Fig.2. We also want this module to have two properties: first, it should be invariant to input permutation since the local point cloud is unordered; second, it should be adaptive to input geometry, hence the projection should combine local coordinates and global information of local patch. Therefore, we first use pointnet [36] to extract the global feature of local region, namely distribution feature, which is invariant to permutation. Then we concatenate the distribution feature to each of the input points, as shown in Fig.3. After that, a shared MLPs is employed to predict the final projection weights.

After projection, 2D convolution is applied on obtained grid feature plane. To extract a local feature vector, global convolution or pooling can be applied on the last layer of 2D convolution network.

However, feature intensity of pixels in grid plane may be unbalanced when the summation of feature intensities

received from points in local region is varying, which can break the stability of a neural network and make the training hard to converge. In order to balance the feature intensity of grid plane, we further introduce two normalization methods on learned projection weights.

Dense Grid Plane: Let projection weights matrix be $W \in \mathbb{R}^{(N \times L)}$. One possible way to obtain a dense grid plane is normalizing W at the first dimension by dividing their summation to make sure the summation of intensities received at each pixel is equal to 1. This is similar to bilinear interpolation method. In our implementation, we use softmax to avoid being divided by zero, which is shown in Eq.6.

$$W_{ij} = \frac{e^{W_{ij}}}{\sum_{k=1}^N e^{W_{kj}}} \quad (6)$$

Sparse Grid Plane: Due to natural sparsity of point cloud, normalize the projection weights to get a dense grid plane may not be optimal. In this case, we design a 2-step normalization which can preserve the sparsity of projection weights matrix, and then the grid plane. Moreover, we conduct ablation studies on our proposed two normalization techniques.

First step is to normalize at second dimension to balance the intensity given out by local neighbor points. Here, we add a positive ϵ to avoid being divided by zero. As shown in Eq.7, $W_{i \cdot}$ indicates i -th row of W .

$$W_{ij} = \frac{W_{ij}}{\|W_{i \cdot}\|_2 + \epsilon} \quad (7)$$

Second step is to normalize at first dimension to balance the intensity received at each pixel position. It can be implemented similar to first step by dividing by summation of each column. However, we choose another method shown in Eq.8 to maintain a continuous sparsity, where $W_{\cdot j}$ indicates j -th column of W . Examples of continuous sparsity and binary sparsity are shown in Fig.4.

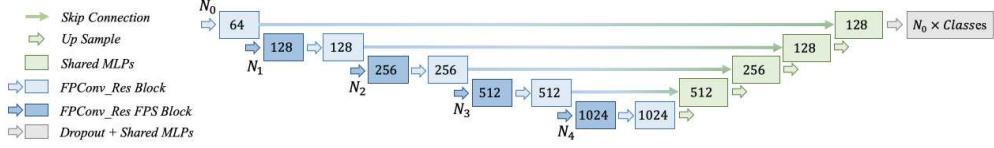


Figure 5: Network Architectures for Large Scene Segmentation: our segmentation architecture is composed of 4 downsampling layers for multi-scale analysis and apply skip connections for combination of features from encoder and decoder.

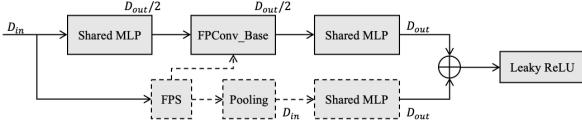


Figure 6: Residual FPCConv Block: operations at the shortcut connection are optional, shared MLP is only required when D_{out} is not equal to D_{in} , which is similar to projection shortcut [14]. FPS (Farthest Point Sampling [37]) and Pooling is needed for downsampling.

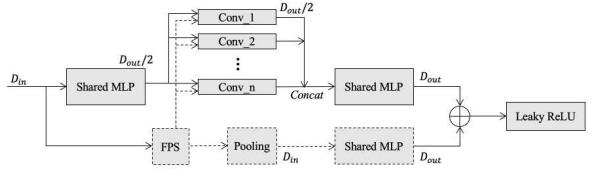


Figure 7: Parallel Residual Block: combine different types (Surface-Conv or Volume-Conv) of convolution kernel for fusion.

$$W_{ij} = \frac{W_{ij}}{\text{MAX}(\|W_{\cdot j}\|_2, 1)} \quad (8)$$

4. Architecture

4.1. Residual FPCConv Block

To build a deep network for segmentation and classification, we develop a bottleneck-design residual FPCConv block inspired by [14], as shown in Fig.6. This block takes a point cloud as input, applying a stack of shared MLP, FPConv, and shared MLP, where shared MLPs are responsible for reducing and then increasing (or restoring) dimensions, similar to 1×1 convolutions in residual convolution block [14].

4.2. Multi-Scale Analysis

As shown in Fig.6 and Fig.5, we design other operations for multi-scale analysis:

Farthest Point Sampling: we use iterative farthest point sampling to downsample the point cloud. As mentioned in PointNet++ [37], FPS has better coverage of the entire point set given the same number of centroids compared with random sampling.

Pooling: we use max-pooling to group local features. Given an input point cloud \mathcal{P}_n and a downsampled point cloud \mathcal{P}_m with their corresponding features \mathcal{F}_n and \mathcal{F}_m , we group neighbors for each point in \mathcal{P}_m with radius of r and apply pooling operator on features of grouped points, as shown in Eq.9, where $\mathcal{P}_{neb} = \{x \mid \|x - y_i\|_2 < r, x \in \mathcal{P}_n\}$

for any $y_i \in \mathcal{P}_m$.

$$\mathcal{F}_{out}(y_i) = \text{Pooling}(\mathcal{F}(\mathcal{P}_{neb})) \quad (9)$$

FPCConv with FPS: similar to pooling operation, this block applies FPCConv on each point of downsampled point cloud and search neighbors over full point cloud, as shown in Eq.10.

$$\mathcal{F}_{out}(y_i) = \text{FPCConv}(\mathcal{F}(\mathcal{P}_{neb})) \quad (10)$$

Upsampling: we use K nearest neighbors interpolation to upsample point cloud by euclidean distance. Given a point cloud \mathcal{P}_m with features \mathcal{F}_m and a target point cloud \mathcal{P}_n , we compute feature for each point in \mathcal{P}_n by interpolating its K neighbor points searched over \mathcal{P}_m .

In the upsampling phase, skip connection and a shared MLPs is used for fusing features from encoder and decoder. K nearest neighbors upsampling and shared MLPs can be replaced by de-convolution, but it does not lead to a significant improvement as mentioned in [44], so we do not employ it in our experiments.

Architecture shown in Fig.5 is designed for large scene segmentation, including four layers of downsampling and upsampling for multi-scale analysis. For classification task, we apply a global pooling on the last layer of downsampling to obtain global feature for representing full point cloud, and then use a fully connected network for classification.

4.3. Fusing Two Convolutions

As one of our main contributions, we also try to answer a question "Can we combine two convolutions for further boosting the performance?" The answer is yes but only

Method	Conv.	Samp.	ScanNet	S3DIS	S3DIS-6
SPGraph [21]	G	-	-	58.0	62.1
ResGCN [23]	G	-	-	53.8	60.0
HPEIN [19]	G	FPS	61.8	61.9	67.8
PointNet [36]	V	FPS	-	41.1	47.6
PointNet++ [37]	V	FPS	33.9	-	-
PointCNN [25]	V	FPS	45.8	57.3	-
PointConv [48]	V	FPS	55.6	58.3 [†]	-
KPConv [44]	V	Grid	68.4	65.4	69.6
TangentConv [43]	S	Grid	43.8	52.6	-
SurfaceConv [33]	S	-	44.2	-	-
TextureNet [18]	S	QF [18]	56.6	-	-
<i>FPConv</i> (ours)	S	FPS	63.9	62.8	68.7
FP \oplus PointConv	S + V	-	-	64.4	-
FP \otimes PointConv	S + V	FPS	-	64.8	-
FP \oplus KPConv	S + V	-	-	66.7	-

Table 1: Mean IoU of large scene segmentation result. The second column is the convolution type (graph, surface or volumetric-style) and third column indicates sampling strategy. S3DIS-6 represents 6-fold cross validation. \oplus is fusion in final feature level while \otimes is fusion in convolutional feature level by applying parallel block. † indicates our implementation.

works when the two convolutions are in different types or complementary (please see Section 6), say surface-style and volumetric-style.

In this section, we propose two convenient and quick fusion strategies, by combining two convolution operators in a single framework. First one is fusing different convolutional features, similar to inception net [42]. As shown in Fig.7, we design a parallel residual block. Given an input feature, apply multiple convolutions in parallel and then concatenate their outputs as fused feature. This strategy is suitable for some compatible methods, such like SA Module of PointNet++ [37], PointConv [48], both using point cloud as input and applying downsampling strategy, which is the same used in our architecture.

While for other incompatible methods, such as TextureNet [18] using mesh as an additional input, and KPConv [44] applying grid downsampling, we have second fusion strategy by concatenating their output features in the last second layer of networks, an then applying a tiny network for fusion.

5. Experiments

To demonstrate the efficacy of our proposed convolution, we conduct experiments on point cloud semantic segmentation and classification tasks. ModelNet40 [49] is used for

Method	Conv.	Samp.	Accuracy
PointNet [36]	V	FPS	89.2
PointNet++ [37]	V	FPS	90.7
PointCNN [25]	V	FPS	92.2
PointConv [48]	V	FPS	92.5
KPConv [44]	V	Grid	92.9
<i>FPConv</i> (ours)	S	FPS	92.5

Table 2: Classification Accuracy on ModelNet40

shape classification. Two large scale datasets named Stanford Large-Scale 3D Indoor Space (S3DIS) [1] and ScanNet [9] are used for 3D point cloud segmentation. We implement our FPConv with PyTorch [34]. Momentum gradient descent optimizer is used to optimize a point-wise cross entropy loss, with a momentum of 0.98, and an initial learning rate of 0.01 scheduled by cosine LR scheduler [27]. Leaky ReLU and batch normalization are applied after each layer except the last fully connected layer. We trained our models 100 epochs for S3DIS, 300 epochs for ScanNet.

5.1. 3D Shape Classification

ModelNet40 [49] contains 12311 3D meshed models from 40 categories, with 9843 for training and 2468 for testing. Normal is used as additional input feature in our model. Moreover, randomly rotation among the z -axis and jittering are also used for data augmentation. As shown in Table.2, our model achieves state-of-the-art performance among surface-style methods.

5.2. Large Scene Semantic Segmentation

Data. S3DIS [1] contains 3D point clouds of 6 areas, totally 272 rooms. Each point in the scan is annotated with one of the semantic labels from 13 categories (chair, table, floor, wall etc. plus clutter). To prepare the training data, 14k points are randomly sampled from a randomly picked block of 2m by 2m. Both sampling are on-the-fly during training. While for testing, all points are covered. Each point is represented by a 9-dim vector of XYZ, RGB, and normalized location w.r.t to the room (from 0 to 1). In particular, the sampling rate for each point is 0.5 in every training epoch.

ScanNet [9] contains 1513 3D indoor scene scans, split into 1201 for training and 312 for testing. There are 21 classes in total and 20 classes are used for evaluation while 1 class for free space. Similar to S3DIS, we randomly sample the raw data in blocks then sample points on-the-fly during training. Each block is of size 2m by 2m, containing 11k points represented by a 6-dim vector, XYZ and RGB.

Pipeline for fusion. As mentioned in Section 4.3, we propose two fusion strategies for fusing conv-kernels of different types. In our experiment, we select PointConv [48]

Method	oA	mAcc	mIoU	ceil.	floor	wall	beam	col.	wind.	door	table	chair	sofa	book.	board	clut.
PointConv [†] [48]	85.4	64.7	58.3	92.8	96.3	77.0	0.0	18.2	47.7	54.3	87.9	72.8	61.6	65.9	33.9	49.3
KPConv [44]	-	70.9	65.4	92.6	97.3	81.4	0.0	16.5	54.4	69.5	80.2	90.1	66.4	74.6	63.7	58.1
<i>FPCConv</i> (ours)	88.3	68.9	62.8	94.6	98.5	80.9	0.0	19.1	60.1	48.9	80.6	88.0	53.2	68.4	68.2	54.9
FP \otimes PointConv	88.2	70.2	64.8	92.8	98.4	81.6	0.0	24.2	59.1	63.0	79.5	88.6	68.1	67.9	67.2	52.4
FP \oplus PointConv	88.6	71.5	64.4	94.2	98.5	82.4	0.0	25.5	62.9	63.1	79.8	87.9	53.5	68.3	67.1	54.5
KP \oplus PointConv	89.4	71.5	65.5	94.6	98.4	81.4	0.0	17.8	56.0	71.7	78.9	90.1	66.8	72.6	65.0	58.7
FP \oplus KPConv	89.9	72.8	66.7	94.5	98.6	83.9	0.0	24.5	61.1	70.9	81.6	89.4	60.3	73.5	70.8	57.8

Table 3: Detailed semantic segmentation scores on S3DIS Area-5. \oplus represents fusion in final feature level while \otimes represents fusion in convolutional feature level. Note that PointConv[†] indicates our implementation on S3DIS.

and KPConv [44] *rigid* for comparison on S3DIS. We apply both two fusion strategies on PointConv with FPCConv, and the second strategy, fusion on final feature level on FPCConv with KPConv and PointConv with KPConv. In our experiments, KPConv *rigid* is used for fusion, while its deformable version is ignored for missing released pre-trained model and hyper-parameters setting. Thus, in the latter part, we use KPConv to represent KPConv *rigid*.

Results. Following [36], we report the results on two settings for S3DIS, the first one is evaluation on Area 5, and another one is 6-fold cross validation (calculating the metrics with results from different folds merged). We report the mean of class-wise intersection over union (mIoU), overall point-wise accuracy (oA) and the mean of class-wise accuracy (mAcc). For Scannet [9], we report the mIoU score tested on ScanNet bencemark.

Results (mIoU) are shown in Table.1. Detailed results of S3DIS including mIoU of each class are shown in Table.3. As we can see, FPCConv outperforms all the existing surface-style learning methods with large margins. Specifically, the mIoU of FPCConv on Scannet [9] benchmark reaches 63.9%, which outperforms the previous best surface-style method by 7.3%. In addition, our FPCConv fused with KPConv achieves state-of-the-art performance on S3DIS.

Even though mIoU of S3DIS of our FPCConv is lower than KPConv, there are still IoUs of some classes outperform the ones of KPConv, such as ceiling, floor, board, etc. Particularly, we find that all of these classes are flat objects, which should have small curvatures. Based on this discovery, we further conduct several ablation studies to explore the relationship between segmentation performance of FPCConv and objects curvatures, as shown in next section.

6. Ablation Study

Two ablation studies are conducted, the first one is exploring fusion of surface-style and volumetric-style convolutions. Another one is the effect of detailed configurations, normalization methods and plane size on FPCConv.

6.1. On Fusion of S.Conv and V.Conv

We firstly study the performance for different combination methods of the two convolutions. Before that, we show an experimental finding that they are complementary and good at analyzing different specific scenes.

Performance vs. Curvature As experiments mentioned in Section 5.2, we claim that FPCConv can perform better on area with small curvature. To be more convincing, we analyzed the relationship between overall accuracy and curvatures, which is shown in the left of Fig.9. We can see that FPCConv outperforms PointConv [48] and KPConv [44] when curvatures are small, and FPCConv cannot perform very well on structures which have large curvatures. Moreover, the histogram of distribution of points curvatures shown in the right of Fig.9 implies almost all points have either large curvatures or small curvatures. This explains why there is a huge performance degradation when curvature increases. Furthermore, as shown in Fig.10, we highlight points (in red) with incorrect prediction, and points (in red) with large curvature. It is oblivious that incorrect prediction is concentrated on area with large curvature and FPCConv performs well in flat area.

Ablation analysis on fusion method As mentioned above, FPCConv which is a surface-style convolution performs better in flat area, worse in rough area and KPConv, as a volumetric-style convolution performs oppositely. We believe that they can be complementary to each other and conduct 4 fusion experiments, FPCConv \otimes PointConv, FPCConv \oplus PointConv, KPConv \oplus PointConv, and FPCConv \oplus KPConv, where \oplus represents fusion in final feature level and \otimes represents fusion in conv level. We don't conduct fusion of FPCConv and KPConv in conv level for their incompatible downsampling strategies. As shown in Table.3, fusion of FPCConv with PointConv or KPConv brings a great improvement, while fusion of PointConv with KPConv brings little improvement. Therefore, we can claim that our FPCConv can be complementary to volumetric-style



Figure 8: Visualization of semantic segmentation results of S3DIS area 5. Images shown in second row is roomed version of first row images. The two red bounding boxes show that two structures that KPConv [44] and FPConv cannot handle both of them very well while $FP \oplus KPConv$ can do it much better.

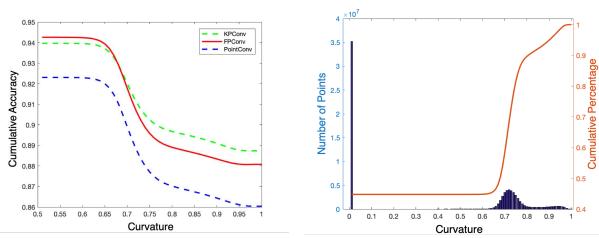


Figure 9: Left: curvature versus cumulative accuracy. Right: histogram of curvatures.

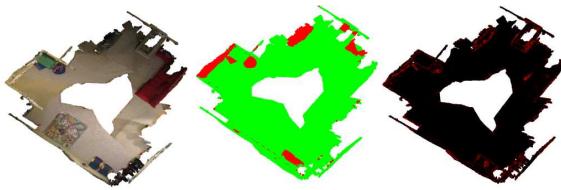


Figure 10: Relationship between accuracy and curvature. Left: raw point cloud. Middle: prediction of FPConv with incorrect points highlighted in red, and correct in green. Right: points with large curvature are highlighted in red. We can see that distribution of incorrect points is consistent with large curvature points.

convolutions, which may direct the convolution design for point cloud in the future.

Visualization of results are shown in Fig.8. Our FPConv can capture better flat structures than KPConv, such as the class column that does not shown in KPConv. While KPConv can capture better complex structures, such as the door. Moreover, the fusion of KPConv and FPConv can achieve better results than both KPConv and FPConv.

6.2. On FPConv Architecture Design

We conduct 4 experiments as shown in Table.4, to study influence of normalization method and the size of grid plane on performance of FPConv. It tells us that, sparse-norm which indicates 2-step normalization method mentioned in Section 3.2 performs better than dense-norm. In addition,

Method	mIoU	mAcc	oAcc
w sparse norm + 6x6	62.8	69.0	88.3
w dense norm + 6x6	61.6	68.5	87.6
w/o norm + 6x6	59.8	67.1	86.2
w sparse norm + 5x5	61.8	68.1	88.4

Table 4: Different normalization results on S3DIS area 5. 6x6 and 5x5 represent different plane sizes.

higher resolution of grid plane may achieve better performance, while bring higher memory cost as well.

7. Conclusion

In this work, we propose FPConv, a novel surface-style convolution operator on 3D point cloud. FPConv takes a local region of point cloud as input, and flattens it onto a 2D grid plane by predicting projection weights, followed by regular 2D convolutions. Our experiments demonstrate that FPConv significantly improved the performance of surface-style convolution methods. Furthermore, we discover that surface-style convolution can be a complementary to volumetric-style convolution and jointly training can boost the performance into state-of-the-art. We believe that surface-style convolutions can play an important role in feature learning of 3D data and is a promising direction to explore.

Acknowledge This work was supported in part by grants No.2018YFB1800800, No.ZDSYS201707251409055, NSFC-61902334, NSFC-61629101, No.2018B030338001, and No.2017ZT07X152.

References

- [1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016. 3, 6

[2] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018. 2

[3] Aseem Behl, Omid Hosseini Jafari, Siva Karthik Mustikovela, Hassan Abu Alhaija, Carsten Rother, and Andreas Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2574–2583, 2017. 1

[4] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. *3DOR*, 2:7, 2017. 3

[5] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 2

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 2

[7] Amin P Charaniya, Roberto Manduchi, and Suresh K Lodha. Supervised parametric classification of aerial lidar data. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 30–30. IEEE, 2004. 1

[8] Nesrine Chehata, Li Guo, and Clément Mallet. Contribution of airborne full-waveform lidar and image data for urban scene classification. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 1669–1672. IEEE, 2009. 1

[9] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 1, 3, 6, 7

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016. 2

[11] Aleksey Golovinskiy, Vladimir G Kim, and Thomas Funkhouser. Shape-based recognition of 3d point clouds in urban environments. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2154–2161. IEEE, 2009. 1

[12] Fabian Groh, Patrick Wieschollek, and Hendrik PA Lensch. Flex-convolution. In *Asian Conference on Computer Vision*, pages 105–122. Springer, 2018. 2

[13] Saurabh Gupta, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation. *International Journal of Computer Vision*, 112(2):133–149, 2015. 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 5

[15] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 2

[16] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. In *SIGGRAPH Asia 2018 Technical Papers*, page 235. ACM, 2018. 2

[17] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018. 2

[18] Jingwei Huang, Haotian Zhang, Li Yi, Thomas Funkhouser, Matthias Nießner, and Leonidas J Guibas. Texturenet: Consistent local parametrizations for learning from high-resolution signals on meshes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4440–4449, 2019. 2, 3, 6

[19] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10433–10441, 2019. 6

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2

[21] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. 6

[22] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer, 2017. 3

[23] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcn: Can gcn go as deep as cnns? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9267–9276, 2019. 6

[24] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018. 2

[25] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn. *arXiv preprint arXiv:1801.07791*, 2018. 2, 6

[26] Zhen Li, Yukang Gan, Xiaodan Liang, Yizhou Yu, Hui Cheng, and Liang Lin. Lstm-cf: Unifying context modeling and fusion with lstms for rgb-d scene labeling. In *European conference on computer vision*, pages 541–557. Springer, 2016. 3

[27] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[28] Andelo Martinovic, Jan Knopp, Hayko Riemenschneider, and Luc Van Gool. 3d all the way: Semantic segmentation of urban scenes from start to end in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4456–4465, 2015. 1

[29] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the*

IEEE international conference on computer vision workshops, pages 37–45, 2015. 2

[30] 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. 1, 2

[31] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and automation (ICRA)*, pages 4628–4635. IEEE, 2017. 3

[32] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017. 2

[33] Hao Pan, Shilin Liu, Yang Liu, and Xin Tong. Convolutional neural networks on 3d surfaces using parallel frames. *arXiv preprint arXiv:1808.04952*, 2018. 2, 3, 6

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[35] Antonio Pomares, Jorge L Martínez, Anthony Madow, María A Martínez, Mariano Morán, and Jesús Morales. Ground extraction from 3d lidar point clouds with the classification learner app. In *2018 26th Mediterranean Conference on Control and Automation (MED)*, pages 1–9. IEEE, 2018. 1

[36] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, pages 652–660, 2017. 1, 2, 4, 6, 7

[37] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 1, 2, 5, 6

[38] Jason Rambach, Alain Pagani, and Didier Stricker. [poster] augmented things: Enhancing ar applications leveraging the internet of things and universal 3d object tracking. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pages 103–108. IEEE, 2017. 1

[39] 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*, pages 3577–3586, 2017. 1, 2

[40] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017. 2

[41] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017. 2

[42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 6

[43] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 2, 3, 6

[44] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. *arXiv preprint arXiv:1904.08889*, 2019. 1, 2, 5, 6, 7, 8

[45] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2598–2606, 2018. 2

[46] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018. 2

[47] 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)*, 38(5):146, 2019. 2

[48] Wenzuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 1, 6, 7

[49] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Lin-guang 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. 1, 2, 6

[50] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018. 2

[51] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2282–2290, 2017. 2

[52] Xiangyu Yue, Bichen Wu, Sanjit A Seshia, Kurt Keutzer, and Alberto L Sangiovanni-Vincentelli. A lidar point cloud generator: from a virtual world to autonomous driving. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 458–464. ACM, 2018. 1