Eurographics Workshop on 3D Object Retrieval (2019)
S. Biasotti, G. Lavoué, B. Falcidieno, and I. Pratikakis (Editors)

Generalizing discrete convolutions for unstructured point clouds

A. Boulch [1]

[1]DTIS, ONERA, Université Paris-Saclay, FR-91123 Palaiseau, France

Abstract

Point clouds are unstructured and unordered data, as opposed to images. Thus, most of machine learning approaches, developed for images, cannot be directly transferred to point clouds. It usually requires data transformation such as voxelization, inducing a possible loss of information. In this paper, we propose a generalization of the discrete convolutional neural networks (CNNs) able to deal with sparse input point cloud. We replace the discrete kernels by continuous ones. The formulation is simple, does not set the input point cloud size and can easily be used for neural network design similarly to 2D CNNs. We present experimental results, competitive with the state of the art, on shape classification, part segmentation and semantic segmentation for large scale clouds.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ *Machine Learning;* • ***Computer graphics*** $\rightarrow$ *Point-based models; Shape analysis;*

1. Introduction

From 3D surface reconstruction for historical heritage preservation to autonomous driving, a large range of applications make use of 3D point clouds. The point clouds are either the direct input, e.g. lidar acquisitions, or intermediary products, e.g. photogrammetry.

These point sets are sparse samples of the underlying surface of the scene or objects. With the exception of structured acquisition (e.g. Lidars), point clouds are generally unordered and not spatially structured. They cannot be sampled on a regular grid and processed as image pixels. Moreover, most of the time, the points do not hold colorimetric features and the observation is consequently only defined by the relative positions of the points.

Due to these considerations, processing point clouds is difficult and methods developed for image processing do not apply. To do so, data transformations and problem reformulation are required. For example, look at convolutional neural networks (CNNs) which have reached the state of the art in many image processing tasks. These neural networks make an extensive use of the grid relations between pixels by using small convolution kernels. This assumption does not hold with point clouds. A first way to adapt this approach is to voxelize the space such that usual convolution applies. However voxelization may induce information loss and directional bias.

In this paper, we propose a generalization of the discrete CNNs for unstructured data. Starting from these networks, we build a new continuous convolutional framework able to ingest sparse point data. The contribution of this paper is two-fold. First, we introduce the convolutional kernels for points. They can be used with points, not necessarily sampled on a grid. It is a simple and straightforward extension of the usual discrete convolutions for grid sampled data. Second, we design neural networks using a hierarchical data representation structure based on a search tree. By using a progressive reduction of the number of observed point in the point cloud space, we end up with a structure very similar to the usual network architectures used with grid-structured input data such as voxels or images. The layer and network architecture are then trained for point cloud classification, part segmentation and semantic segmentation. For each task we show that our method is competitive with the state of the art.

The paper is organized as follow: section 2 presents the related works, section 3 decribes the continuous convolutional layer, section 4 is dedicated to the spatial representation of the data and the description of the networks. Finally, the section 5 shows experiments on different datasets for classification and semantic segmentation.

2. Related Work

Point cloud processing

Point cloud processing is a widely discussed topic. We focus here on machine learning techniques for point cloud classification or local attribute estimation.

Most of the methods use handcrafted features defined using a point and its neighborhood or a local estimate of the underlying surface around the point. These features describe statistical properties of the shape and are designed to be invariant to rigid or non rigid transformation of the shape [JH99, ASC11, BK10, LJ07]. These descriptors are then usually fed to classical machine learning objects such as Support Vector Machine or random forest.

In the last years, the release of large annotated point cloud

DOI: 10.2312/3dor.20191064

databases has allowed the development of deep neural networks methods able to learn both descriptors and decision function.

The direct adaptation to 3D of CNNs developed for image processing is to use 3D convolutions. Methods like [WSK*15, MS15] apply 3D convolutions on voxel grid. Even though recent hardware advances allow us to use these network on larger scenes, they are still time consuming and require a relatively low voxel resolution which may result in a loss of information and undesirable bias due to grid axis alignment. In order to avoid these drawbacks, [GvdM17, Gra15] use sparse convolutional kernels or [LPS*16, WP15] observe the 3D space to focus computation where objects are located.

A second class of algorithms avoid 3D convolutions by creating 2D representations of the shape, applying 2D CNNs and projecting the results back to 3D. This approach have been used for object classification [SMKLM15], jointly with voxels [QSN*16] or for semantic segmentation [BGLSA17]. One the main issues using multi-view frameworks is to define an efficient and robust view generation procedure which can be, depending on the data, very difficult.

The previous methods are based on a 2D and 3D CNN. Somehow, it implies to organize the data (3D grid or 2D image) to process it. A third class of machine learning approaches considers unstructured data: PointNet [QSMG17] and derivatives. The key idea is to construct a transfer function invariant by permutation of the inputs features, obtained by using the symmetric function *max pooling*. The coordinates of the points are given as input features and geometric operations (affine transformation) are obtained with small auxiliary networks. However, it fails to capture local structures. Its improvement, PointNet++ [QYSG17], uses a cascade of PointNet networks from local scale to global scale.

Learning features

Convolutional neural layers are widely used in machine learning for image processing and more generally for data sampled on a regular grid (such as 3D voxels). However, the use of CNNs when data is missing or not sampled on a regular grid is not straightforward. Several works have studied the generalization of such powerful schemes to other data.

To avoid problems linked to discrete kernels and sparse data. [DQX*17] introduces deformable convolutional kernels able to adapt to the recognition task. In [SJS*18], the authors are able to deal with point clouds. The input signal is interpolated on the convolutional kernel, convolution is applied, and output interpolated back to input shape. The proposed approach shares ideas with these works but is not dependent of a convolution kernel designed on a grid. The kernel elements location are optimized like in [DQX*17] and the input points are weighted according to their distance to kernel elements, like in [SJS*18].

In [LBS*18], a χ-transform is applied on the point coordinates to create geometrical features to be combined with the input point features. It is a major difference with our approach: as in [QYSG17], [LBS*18] makes use of the input geometry as features. We want the geometry to behave as in discrete convolutions, i.e. weighting the relation between kernel and input. In other word, the geometric

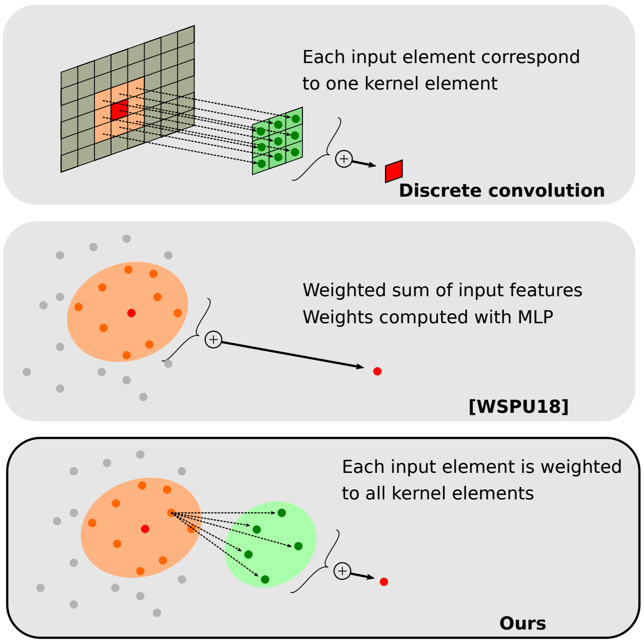

Each input element correspond
to one kernel element
Discrete convolution
Weighted sum of input features
Weights computed with MLP
[WSPU18]
Each input element is weighted
to all kernel elements
Ours

Figure 1: *Convolutions.*

aspects are defined in the network structure (convolution strides, pooling layers) and pixel coordinates are not a network input.

In [WSPU18], the authors propose a convolution layer defined as a weighted sum of the input features, the weights being computed by a multi layer perceptron (MLP). Our approach shares with [WSPU18] both in concept and implementation but differs in two parts. First, our approach computes a dense weighting function that takes into account the whole kernel. Second, the derived kernel is an explicit set of points associated with weights as opposed to an implicit version using a multi layer perceptron (MLP).

3. Convolutional kernel for point clouds

We build our continuous convolutional layer by deriving the discrete convolution formulation, used for grid-sampled data, such as images.

Discrete convolutions

In discrete convolutions, the kernel $K = \{\mathbf{w}_i\}, i \in [\![1, N]\!]$ (where $[\![.,.]\!]$ is an integer sequence and N is the number of elements $\mathbf{w}_i$ in K) is convoluted with an input $X = \{\mathbf{x}_i\}, i \in [\![1, |X|]\!]$ of same size $|X| = N$, $\mathbf{x}_i$ being the feature vector associated with the i-th element of X. Given a bias β, the output $\mathbf{y}$ is:

$$y = \beta + \sum_{i=1}^{N} \mathbf{w}_i \mathbf{x}_i = \beta + \sum_{i=1}^{N} \sum_{j=1}^{|X|} \mathbf{w}_i \mathbf{x}_j \mathbf{1}(i, j) \tag{1}$$

where $\mathbf{1}(.,.)$ is the indicator function such that $\mathbf{1}(a,b) = 1$ if $a = b$, 0 otherwise. This is a one to one relation between kernel elements and input elements. This is illustrated on figure 1 top for a convolution on grid-sampled data.

If we now consider $\mathbf{c}_i$, resp. $\mathbf{p}_i$ the spatial locations of the kernels

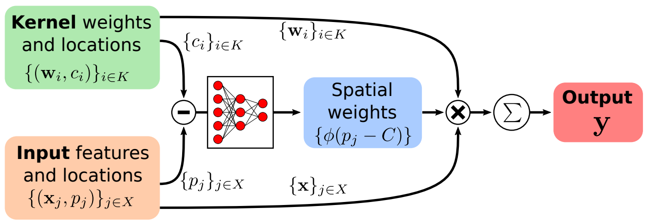

Kernel weights and locations $\{(\mathbf{w}_i, c_i)\}_{i\in K}$
$\{c_i\}_{i\in K}$
$\{\mathbf{w}_i\}_{i\in K}$
Spatial weights $\{\phi(p_j - C)\}$
Output $\mathbf{y}$
Input features and locations $\{(\mathbf{x}_j, p_j)\}_{j\in X}$
$\{p_j\}_{j\in X}$
$\{\mathbf{x}\}_{j\in X}$

Figure 2: *Convolutional layer.*

elements, resp. the input elements (for an image it would be the pixel coordinates of the patch), then $K = \{(\mathbf{c}_i, \mathbf{w_i})\}$ and $X = \{(\mathbf{p}_i, \mathbf{x}_i)\}$. We can rewrite the expression:

$$y = \beta + \frac{1}{|X|}\sum_{j=1}^{|X|}\sum_{i=1}^{N} \mathbf{w}_i \mathbf{x}_j \mathbf{1}(\mathbf{c}_i, \mathbf{p}_j) \tag{2}$$

where we added a normalization according to the input set size, for robustness to input variation in size.

Convolution on point set

In this study, we consider points in dimension d, without spatial structure. In the general case, using $\mathbf{1}(\mathbf{c}_i, \mathbf{p}_j)$ will be too restrictive, taking almost all the time 0 value. More generally, we need a continuous function $\phi : \mathbb{R}^{N*d} \to \mathbb{R}$, to establish a relation between the c_is and p_j. ϕ is a geometrical weighting function that takes into account the relations between the elements of X and K. In a more general case ϕ may not only consider $\mathbf{c_i}$ but the whole kernel configuration. To this end, we choose a ϕ_i function, specific to the i-th kernel element, function of the relative position between the kernel elements and the input points, $(\mathbf{c}_i - \mathbf{p}_j)$. We note $p_j - C$ the set $\{\mathbf{c}_i - \mathbf{p}_j, i \in [\![1, N]\!]\}$:

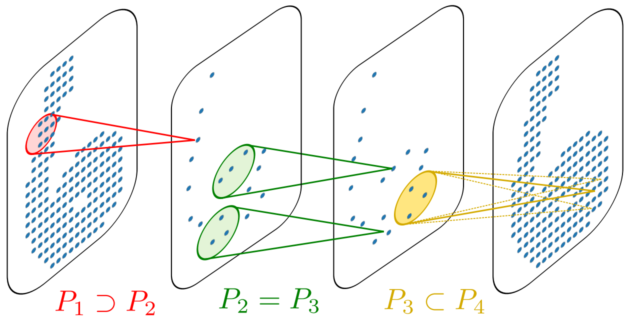
$$y = \beta + \frac{1}{|X|}\sum_{j=1}^{|X|}\sum_{i=1}^{N} \mathbf{w}_i \mathbf{x}_j \phi_i(p_j - C) \tag{3}$$

Our approach and a synthetic view of [WSPU18] are presented on figure 1 bottom and middle (resp.).

The main difference in formulation with [WSPU18] resides in the explicit formulation of the kernel in a way similar to discrete formulation. In [WSPU18], the author directly use a MLP to weight the input features centered around the reference point used for computing the neighborhood. It is particular case of our formulation with $N = 1$ and $\mathbf{c}_1$ is the reference point. Moreover, as opposed to approaches like PointNet [QSMG17] and PointNet++ [QYSG17], the geometrical space and the feature space are separated, i.e. point spatial coordinates are not input features.

Weighting function

In practice designing by hand such ϕ functions is not easy. Intuitively, it would decrease with the norm of $(\mathbf{c}_i - \mathbf{p}_j)$. We tested several functions including Gaussian functions. They require hand crafted parameters, difficult to tune. Instead, we choose to learn this function with a simple multi-layer perceptron (MLP). Such approach does not presume the behavior of the function.



(a) Spatial reduction (b) No spatial reduction (c) Spatial augmentation

Figure 3: *Spatial structure. Possible behavior of convolutional layer depending on the cardinal of input and output clouds.*

The whole convolutional layer is represented on figure 2

Parameters and optimization

At training, both parameters $\mathbf{w}_i$ of K and MLP parameters are optimized using gradient descent.

In addition, even though we can hope for MLP to partly deal with this issue, setting kernel element locations $\mathbf{c}_i$ remains a problem. As an example, setting $\mathbf{c}_i$ positions on a regular grid could induce bias. To this end, we randomly select the locations of $\mathbf{c}_i$ in the unit sphere and also optimize these parameters, thanks to the fact that the formula (3) is differentiable.

Properties

Permutation invariant As stated in [QSMG17], operators on points must be invariant by permutation of the points. In the general case, the points are not ordered. For robustness reasons it is important that permuting two inputs has no influence on the extracted features. y is a sum over the input points and a permutation of the point indices has no effect on the sum results.

Translation invariant As the geometric relations are relative between the points and the kernel elements $(p_j - C)$, applying a global translation on the point cloud and the kernel doesn't change the result.

Insensible to input point cloud scale Many point clouds, such as photogrammetric pointclouds,have no metric scale information. In order to make the convolution robust to the input scale, the input geometric points $\mathbf{p}_j$ are normalized.

Reduced sensibility to input point cloud size Dividing the results by $|X|$ makes the output less sensible to input size. E.g., using $2X$ ans input does not change the result.

4. Spatial structure

Each convolutional layer operates a projection of the input features of point cloud P on the output point cloud Q.

For each point $q \in Q$ of the output point cloud, the convolutional

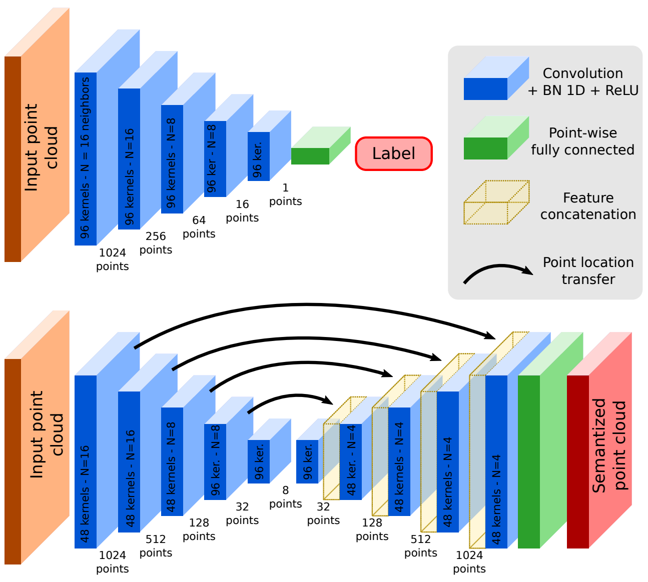


Figure 4: *Networks used in this paper, classification (top) and segmentation (bottom).*

kernels are applied on the K nearest neighbors of q in P. Doing that ensures the spatial locality of the convolution operation.

In practice, chosing $Q \subset P$ (or more generally $|Q| < |P|$) leads to a dimension reduction by the convolutional layer (figure 3(a)). It is similar to the convolution with stride for discrete convolutions.

Using $Q = P$ (or more generally $|Q| = |P|$) do not change the point cloud size (figure 3(b)). It can be compared with the discrete convolution with stride 1.

And Finally, using $Q \supset P$ or more generally $|Q| > |P|$) is a dimension augmentation, similar to the up-convolutional layer in discrete pipelines (figure 3(c)).

4.1. Network design

The similarity between our definition of the convolutional layer and the discrete convolution allows network design comparable to image analysis networks. Our architectures are widely inspired from the literature on CNN for 2D computer vision tasks and particularly LeNet-like models [LBBH98] for classification and U-net [RFB15] for segmentation and regression.

The **classification network** is presented on figure 4 (top). It is a stack of five convolutional layers followed by a linear layer with a number of outputs corresponding to the number of classes. After the first convolutional layer, the point cloud is reduced to 1024 points, and then downsized by a factor 4 at each layer, except the last one where information is compressed to one output with 96 features. This output is then used by a linear layer with output number equal to the number of classes.

The **segmentation network** (figure 4) has an encoder-decoder structure, similar to U-net. The encoder has the same structure as the classification network and the decoder is composed of a stack of five convolutions and a linear layer. In the decoder the points used for upsampling are the same as the points at corresponding size in the encoder. The features from the encoder and the decoder are concatenated at the input of the convolutional layers of the decoder. Finally, the last layer is a point-wise linear layer, with the same purpose as in the previous layer.

For all the convolutions, we choose 27 kernel elements. This number has been chosen with reference to the number of elements of $3 \times 3 \times 3$ 3D convolution kernel. Future works will include an extensive study of the influence of the number of kernel elements. The number of neighbors used for convolution is noted N on figure 4. It varies from 4 for deconvolution to 16 for the first convolutions.

5. Experiments

5.1. Training

We train the networks using a stochastic gradient descent with Adam optimizer. In order to make training compatible with mini-batch training, we use the same input point cloud size for each sample and the neighborhood sizes are fixed for each convolutional layer. This ensures that even tough the spatial structure is randomly generated (random selection of the points in layers with dimension reduction), the global structure is similar for each sample.

5.2. Classification

The first task for experimentation is shape classification. We use the network defined in figure 4 (top).

In order to show the flexibity of the framework, we experiment in both 2D and 3D.

The 2D experiment is done on the MNIST dataset. We consider the input image as 2D points (pixel coordinates) associated with color features (grey value). Results are presented on table 1(left). For comparison, we have also reported the results from two usual 2D CNN, LeNet [LBBH98] and Network in Network [LCY13] and two methods working directly on points, PointNet++ [QYSG17] and PointCNN [LBS*18].

The classification task is also experimented in the 3D, on the ModelNet40 dataset. This dataset is a set of meshes from 40 various classes (plane, cars, chairs, tables...). We generated point clouds by sampling point on the triangular faces of the meshes. In our experiments, we use an input size of 1024 points. Table 1(right) presents the results.

In both 2D and 3D, our approach is competitive with the state of the art methods. With this experiment, we show that implicit coding of the geometry (in the spatial representation) is as efficient as using point positions as features like in [QYSG17] or [LBS*18].

Number of spatial sampling Due to stochastic process for spatial downsampling, different runs on the same shape may result in different output: point picked may be different at each level, which implies that neighborhoods and resulting features may be diferent. To increase robustness, we run several times the spatial sampling, predict and average the output scores. This is refered as the number of sampling (1,8 or 16) in table 1. The performances increase with

MNIST dataset

Methods	OA
Image methods	
LeNet [LBBH98]	99.20
NiN [LCY13]	99.53
Points	
PointNet++ [QYSG17]	99.49
PointCNN [LBS*18]	99.54
Ours	
1 sampling	99.55
8 samplings	99.59
16 samplings	**99.61**

ModelNet40 dataset

Methods	OA	AA
Voxels		
Subvolume [QSN*16]	89.2	
Images		
MVCNN [SMKLM15]	90.1	
Graph		
DGCNN [WSL*18]	92.2	**90.2**
Points		
PointNet [QSMG17]	89.2	86.2
PointNet++ [QYSG17]	90.7	
PointCNN	**92.2**	88.1
Ours		
1 sampling	90.1	86.8
8 samplings	91.2	87.8
16 samplings	91.6	88.1

Table 1: *Shape classification*

Method	pIoU	mpIoU
SyncSpecCNN [YSGG17]	84.74	82.0
Pd-Network [KL17]	85.49	82.7
SSCN [GEvdM18]	85.98	83.3
SPLATNet [SJS*18]	85.4	83.7
SpiderCNN [XFX*18]	85.3	81.7
SO-Net [LCHL18]	84.9	81.0
PCNN [AML18]	85.1	81.8
KCNet [SFYT18]	83.7	82.2
SpecGCN [WSS18]	85.4	-
3DmFV-Net [BSLF17]	84.3	81.0
RSNet [HWN18]	84.9	81.4
DGCNN [WSL*18]	85.1	82.3
PointNet [QSMG17]	83.7	80.4
PointNet++ [QYSG17]	85.1	81.9
SGPN [WYHN18]	85.8	82.8
PointCNN [LBS*18]	86.14	**84.6**
Ours 1024 pts		
1 tree	91.9	80.1
8 tress	93.0	82.5
16 trees	**93.1**	82.6

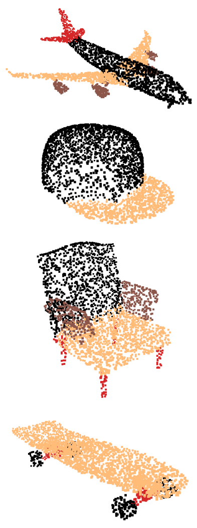


Table 2: *ShapeNet*

the number of samplings. In practice, we only presented up to 16 samplings because we observed that a larger number would not increase score significantly.

5.3. Semantic segmentation

Part segmentation. Given a point cloud, the part segmentation objective is to recognize the different part of the underlying shape. In practice, it is a semantic segmentation at shape level. We use the Shapenet [YKC*16] dataset. It is composed of 16680 models splitted in train/test sets, belonging to 16 shape categories. Each category is annotated with 2 to 6 part labels. The total is 50 part classes. As in [LBS*18], we consider the part annotation problem as a 50 classes semantic segmentation problem. The scores are then computed at shape level.

We use the semantic segmentation network from figure 4(bottom). The provided sample have various sizes. We randomly subsample the point clouds to 1024 points and predict the labels for each input point. As the points do not come with particular features, we set the input features to one. As all points may not have been selected for labelling, the final labels are obtained by the 16 nearest labeled neighbors and their scores are summed.

The results are preseted in table 2. The scores are the part intesection over union (pIoU) and the mean part intersection over union (mpIoU). The first is the global IoU computed over all points, the second compute the IoU at shape level and average the scores over the shapes.

Our approach outperforms the state of the art by 6% looking at the pIoU score and is among the best methods according to the mpIoU. The difference may be explained by the fact that when the model recognises a shape, it produces really accurate segmentation. While on the contrary, some other shapes more poorly segmented due to a confusion in the main classes, producing a mpIoU at 82.6%.

S3DIS. We then experiment on the Standord 2D-3D-Semantics dataset [ASZS17]. It is an indoor point cloud dataset for semantic segmentation. It is composed of six scenes, each corresponding to an office floor. The points are labeled according to 13 classes, 6 building elements classes (floor, ceiling...), 6 office equipment classes (tables, chairs...) and a stuff class regrouping all the small equipment (computers, screens...) and rare items.

For each scene considered as a test set, we train on the other five. At training and testing time, we randomly select a point and all the points in a box of given side size (a scale) for the horizontal axes and unbounded in vertical axis. From these points, we sample the 4096 points, input of the network. The input features are the RGB colors. At test time, we follow the same procedure as for part segmentation. For a given point, the final labeled is inferred from the predictions of the neighboring labeled points.

Table 3 regroups the results of our method and state of the art methods. We trained the network on a two meter scale (column selection size is 2m), scores are presented on the first line of results. We obtain competitive results on most of the classes, getting even the top score en walls. However, the training mostly fails on a few classes, e.g. columns that are mistaken with walls.

One explication could be that looking at the data with a 2m scale could be too coarse. And details such as frontiers between walls and columns are mis-estimated. The direct training with a finer scale (1m) did not give good results, indicating a loss of performance on walls and ceiling.

To refine the results, we trained a second model at 1m scale. The inputs of this second model are the output scores of the first model concatenated with the RGB features. By doing that we include knowledge of the 2m scale in this training. It sequential training, the first network is trained, we freeze the parameters, and use the predictions as input features for the next network. The scores are increased in each category, we can even see an amelioration of 13% on the column category.

Semantic8 The last segmentation experiment is on the Semantic8 dataset [HSL*17]. In this experiment we explore the ability of the

Method	OA	mAcc	mIoU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [QSMG17]	78.5	66.2	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	54.1	42.0	9.6	38.2	29.4	35.2
SPGraph [LS18]	85.5	73.0	62.1	89.9	95.1	76.4	62.8	47.1	55.3	**68.4**	**73.5**	69.2	**63.2**	45.9	8.7	52.9
RSNet [HWN18]	-	66.45	56.47	92.48	92.83	78.56	32.75	34.37	51.62	68.11	60.13	59.72	50.22	16.42	44.85	52.03
PCCN [WSPU18]	-	67.01	58.27	92.26	96.20	75.89	0.27	5.98	**69.49**	63.45	66.87	65.63	47.28	68.91	**59.10**	46.22
PointCNN [LBS*18]	**88.14**	**75.61**	**65.39**	**94.78**	**97.3**	75.82	**63.25**	**51.71**	58.38	57.18	71.63	69.12	39.08	**61.15**	**52.19**	**58.59**
Ours																
1 scale 2m	84.05	-	55.36	90.49	92.19	75.84	35.90	20.53	60.99	45.86	59.35	68.27	15.68	52.48	48.94	53.21
2 scales 2m,1m	84.93	-	58.54	90.93	92.99	**76.72**	40.17	33.05	62.13	48.34	61.53	**72.95**	23.53	53.20	50.62	54.85

Table 3: *S3DIS*

Method	AvIoU	OA	Man made	Natural	High veg.	Low veg.	Buildings	Hard scape	Artefacts	Cars
TML-PC [MZWLS14]	0.391	0.745	0.804	0.661	0.423	0.412	0.647	0.124	0.000	0.058
TMLC-MS [HWS16]	0.494	0.850	0.911	0.695	0.328	0.216	0.876	0.259	0.113	0.553
PointNet++ [QYSG17]	0.631	0.857	0.819	0.781	0.643	0.517	0.759	0.364	0.437	0.726
SnapNet [BGLSA17]	0.674	0.910	0.896	**0.795**	0.748	0.561	0.909	0.365	0.343	0.772
SPGraph [LS18]	**0.762**	**0.929**	**0.915**	0.756	**0.783**	**0.717**	**0.944**	**0.568**	**0.529**	**0.884**
Ours	0.666	0.898	0.911	0.778	0.633	0.565	0.888	0.268	0.448	0.839
ranking	*3*	*3*	*2*	*3*	*6*	*4*	*5*	*8*	*2*	*2*

Table 4: *Semantic8.*

training process to scale to very large scenes (up to hundred of millions of points) with high variation in point density accross the scenes. The Semantic8 dataset is composed of 30 ground lidar scenes, 15 for training and 15 for evaluation. The test labels are unknown and evaluated on an online server. Here, we train the network with a neighborhood column of 8 meters.

The results of evaluation are presented in table 4 and figure 6. We have reported the state of the benchmark leaderboard at the time of article writing (for entries that are not anonymous). The PointNet++ has two entries in the benchmark, we only reported the best one. Our convolutional network for segmentation places at the third position behind Super Point Graph (SPGraph) [LS18] and SnapNet [BGLSA17]. It is the first among the direct point processing methods as SPGraph relies on a pre-segmentation of the point cloud and SnapNet uses 2D segmentation network on virtual pictures of the scene to produce segmentation. We surpass the PointNet++ by 3% on the average IoU. We perform particularly well on car detection where other methods except for SPGraph get relatively low results. On the contrary, we obtain poor performances on the vegetation classes (high and low). When looking at the results, this is due to a confusion between these two classes. In our understanding, this is a consequence of the absence of absolute scale in the pipeline. Basically, as all neighborhhods are rescaled to the unit sphere, making the process robust to scale variation of models (efficient for mesh classification without scale), a small tree or an hedge (low vegetation) may be confused with a large one (high vegetation).

5.4. Robustness to point cloud size

One of the main advantages of the convolution using a dense formulation and averaging the final weights by the cardinal of the input size is that can accept any input point cloud size. To test the robustness to point cloud size variation, we use the classification network trained with 1024 points. We run the predictions on the test set (with one spatial sample) for different input sizes.

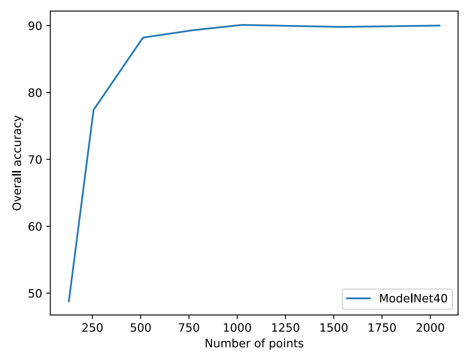


Figure 5: *Robustness to test input size for a model trained with 1024 points.*

We observe that overall accuracy on ModelNet40 is maintained even if we test with a number of points greater than the size used for training. In our opinion, this is related to the size needed to estimate local features. The first layer of the network compute local features (e.g. normals or curvatures) and these do not vary a lot once a given point density is reached (upper than 1024). Starting from second layer, the number of points is fixed and the behavior of the network is the same.

On the opposite, with small point clouds (smaller than 1024 points), there are two phenomena that tend to lower classification scores. First, a small point cloud induces loss of information on the surface compared to large ones. Second, the local features may not be well estimated as neighborhoods becomes larger. Moreover, the 1024 point sampling of the second convolutional layer implies to use several times the same points, which could magnify bad feature estimation of the first convolution. Nevertheless, even with 128 points we have 48% of good classification over 40 classes. This is up to 88% with 512 points, half of the training size.

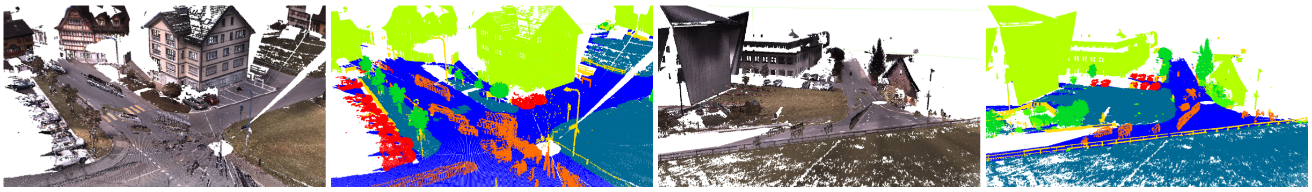


Figure 6: *Semantic segmentation on Semantic8 dataset*

5.5. Computation speed

For timing computation at inference time, we use a Nvidia GTX1070 8Gb. On S3DIS, the segmentation network, with a batch size of 2, processes around 31 point clouds per second. The point cloud size is 4096 and this timing includes the spatial sampling (operated on 4 separated threads). One point cloud is processed in 0.032s.

At training, on the Semantic8 dataset, with batchsize 2 and Adam optimizer, we get 0.41s for a batch (forward and backward). The input features have size 3 (RGB only).

6. Limitations and perspectives

One of the main limitations is that our networks are agnostic to the object scales. It is of interest when dealing with non metric data such as in the ModelNet40 or ShapeNet datasets, where objects are CAD objects designed without scale. It would also be interesting to experiment on photogrammetric point clouds where scales are not always avalaible. On the opposite, in metric scans such as Semantic8 or S3DIS the object sizes are valuable informations that can be used for accurate segmentation. For example, a beam or column may be considered as wall or ceiling if its size is not taken its size into account. It would be the extension of the proposed framework along with using fixed radius neighborhood instead of fixed neighbors number. As pointed out, this would forbid batch training but the network would have to deal at each layer with object at prefined size.

Another perspective is to explore the use of precomputed features as inputs. In this study, we only use raw data for network inputs: RGB colors when available, features set to one otherwise. In the future, we will work on feeding the networks with features such as normals or curvatures.

We will also deepen the multiscale approach. The use of two scales for the 3DIS is only a scratch over the use for multiscale approaches for 3D semantic segmentation. Even though we restricted the multiscale approach to sequential learning of two models in order to stick to moderate GPU usage (see implementation details section for hardware information) We will explore joint learning of the scales and explore which features are the most relevant for passing to the next scale.

Finally, we proposed two networks architectures widely inspired from computer vision models. It is a interesting to explore various network configurations. As the formulation generalizes the discrete convolution, it is possible to explore more recent architectures such as residual networks.

7. Conclusion

In this paper, we presented a new CNN framework for point cloud processing. The objective was to provide an extension of the discrete convolution for sparse, unstructured data. From this convolutional layer, we built two networks, for shape classification and one for semantic segmentation. Through several experiments on various benchmark datasets, real and simulated, we have shown the method to be efficient and flexible. We also proved that even without particular extension, the approach scales well to very large scale datasets such as Semantic8.

Implementation details

We implemented our method using Pytorch framework. All operations are implemented using the native autograd functions and runs with Nvidia CUDA. For the experiments, we used a Nvidia Titan Xp for training, and a GTX1070 for inference.

Code

The code is available under open source licence in the following repository `https://github.com/aboulch/ConvPoint`.

Acknowledgements

This work is supported by the ONERA project Delta. This project aims at developing innovative machine learning approaches for aerospace applications.

References

[AML18] Atzmon M., Maron H., Lipman Y.: Point Convolutional Neural Networks by Extension Operators. *arXiv preprint arXiv:1803.10091* (2018). 5

[ASC11] Aubry M., Schlickewei U., Cremers D.: The wave kernel signature: A quantum mechanical approach to shape analysis. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on* (2011), IEEE, pp. 1626–1633. 1

[ASZS17] Armeni I., Sax A., Zamir A. R., Savarese S.: Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints* (Feb. 2017). 5

[BGLSA17] Boulch A., Guerry J., Le Saux B., Audebert N.: SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. *Computers & Graphics* (2017). 2, 6

[BK10] Bronstein M. M., Kokkinos I.: Scale-invariant heat kernel signatures for non-rigid shape recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 1704–1711. 1

[BSLF17] BEN-SHABAT Y., LINDENBAUM M., FISCHER A.: 3D Point Cloud Classification and Segmentation using 3D Modified Fisher Vector Representation for Convolutional Neural Networks. *arXiv preprint arXiv:1711.08241* (2017). 5

[DQX*17] DAI J., QI H., XIONG Y., LI Y., ZHANG G., HU H., WEI Y.: Deformable convolutional networks. *CoRR, abs/1703.06211 1*, 2 (2017), 3. 2

[GEvdM18] GRAHAM B., ENGELCKE M., VAN DER MAATEN L.: 3D semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 9224–9232. 5

[Gra15] GRAHAM B.: Sparse 3D convolutional neural networks. *arXiv preprint arXiv:1505.02890* (2015). 2

[GvdM17] GRAHAM B., VAN DER MAATEN L.: Submanifold Sparse Convolutional Networks. *arXiv preprint arXiv:1706.01307* (2017). 2

[HSL*17] HACKEL T., SAVINOV N., LADICKY L., WEGNER J., SCHINDLER K., POLLEFEYS M.: Smeantic3D.net: A new large scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2017), vol. IV-1-W1, pp. 91–98. 5

[HWN18] HUANG Q., WANG W., NEUMANN U.: Recurrent Slice Networks for 3D Segmentation of Point Clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2626–2635. 5, 6

[HWS16] HACKEL T., WEGNER J. D., SCHINDLER K.: Fast semantic segmentation of 3D point clouds with strongly varying density. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences 3*, 3 (2016). 6

[JH99] JOHNSON A. E., HEBERT M.: Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 5 (1999), 433–449. 1

[KL17] KLOKOV R., LEMPITSKY V.: Escape from cells: Deep kd-networks for the recognition of 3D point cloud models. In *IEEE International Conference on Computer Vision (ICCV)* (2017), IEEE, pp. 863–872. 5

[LBBH98] LECUN Y., BOTTOU L., BENGIO Y., HAFFNER P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324. 4, 5

[LBS*18] LI Y., BU R., SUN M., WU W., DI X., CHEN B.: PointCNN: Convolution On X-Transformed Points. In *Advances in Neural Information Processing Systems 31*, Bengio S., Wallach H., Larochelle H., Grauman K., Cesa-Bianchi N., Garnett R., (Eds.). Curran Associates, Inc., 2018, pp. 828–838. 2, 4, 5, 6

[LCHL18] LI J., CHEN B. M., HEE LEE G.: So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 9397–9406. 5

[LCY13] LIN M., CHEN Q., YAN S.: Network in network. *arXiv preprint arXiv:1312.4400* (2013). 4, 5

[LJ07] LING H., JACOBS D. W.: Shape classification using the inner-distance. *IEEE transactions on pattern analysis and machine intelligence 29*, 2 (2007), 286–299. 1

[LPS*16] LI Y., PIRK S., SU H., QI C. R., GUIBAS L. J.: FPNN: Field probing neural networks for 3D data. In *Advances in Neural Information Processing Systems* (2016), pp. 307–315. 2

[LS18] LANDRIEU L., SIMONOVSKY M.: Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 4558–4567. 6

[MS15] MATURANA D., SCHERER S.: Voxnet: A 3D convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on* (2015), IEEE, pp. 922–928. 2

[MZWLS14] MONTOYA-ZEGARRA J. A., WEGNER J. D., LADICKỲ L., SCHINDLER K.: Mind the gap: modeling local and global context in (road) networks. In *German Conference on Pattern Recognition* (2014), Springer, pp. 212–223. 6

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: PointNet: Deep learning on point sets for 3D classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE 1*, 2 (2017), 4. 2, 3, 5, 6

[QSN*16] QI C. R., SU H., NIESSNER M., DAI A., YAN M., GUIBAS L. J.: Volumetric and multi-view CNNs for object classification on 3D data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 5648–5656. 2, 5

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems* (2017), pp. 5105–5114. 2, 3, 4, 5, 6

[RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241. 4

[SFYT18] SHEN Y., FENG C., YANG Y., TIAN D.: Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), vol. 4. 5

[SJS*18] SU H., JAMPANI V., SUN D., MAJI S., KALOGERAKIS V., YANG M.-H., KAUTZ J.: SPLATNet: Sparse Lattice Networks for Point Cloud Processing. *arXiv preprint arXiv:1802.08275* (2018). 2, 5

[SMKLM15] SU H., MAJI S., KALOGERAKIS E., LEARNED-MILLER E.: Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 945–953. 2, 5

[WP15] WANG D. Z., POSNER I.: Voting for Voting in Online Point Cloud Object Detection. In *Robotics: Science and Systems* (2015), vol. 1, p. 5. 2

[WSK*15] WU Z., SONG S., KHOSLA A., YU F., ZHANG L., TANG X., XIAO J.: 3D shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1912–1920. 2

[WSL*18] WANG Y., SUN Y., LIU Z., SARMA S. E., BRONSTEIN M. M., SOLOMON J. M.: Dynamic graph CNN for learning on point clouds. *arXiv preprint arXiv:1801.07829* (2018). 5

[WSPU18] WANG S., SUO S., POKROVSKY W.-C. M. A., URTASUN R.: Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2589–2597. 2, 3, 6

[WSS18] WANG C., SAMARI B., SIDDIQI K.: Local Spectral Graph Convolution for Point Set Feature Learning. *arXiv preprint arXiv:1803.05827* (2018). 5

[WYHN18] WANG W., YU R., HUANG Q., NEUMANN U.: SGPN: Similarity group proposal network for 3D point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2569–2578. 5

[XFX*18] XU Y., FAN T., XU M., ZENG L., QIAO Y.: SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. *arXiv preprint arXiv:1803.11527* (2018). 5

[YKC*16] YI L., KIM V. G., CEYLAN D., SHEN I., YAN M., SU H., LU C., HUANG Q., SHEFFER A., GUIBAS L., ET AL.: A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 210. 5

[YSGG17] YI L., SU H., GUO X., GUIBAS L. J.: SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2282–2290. 5