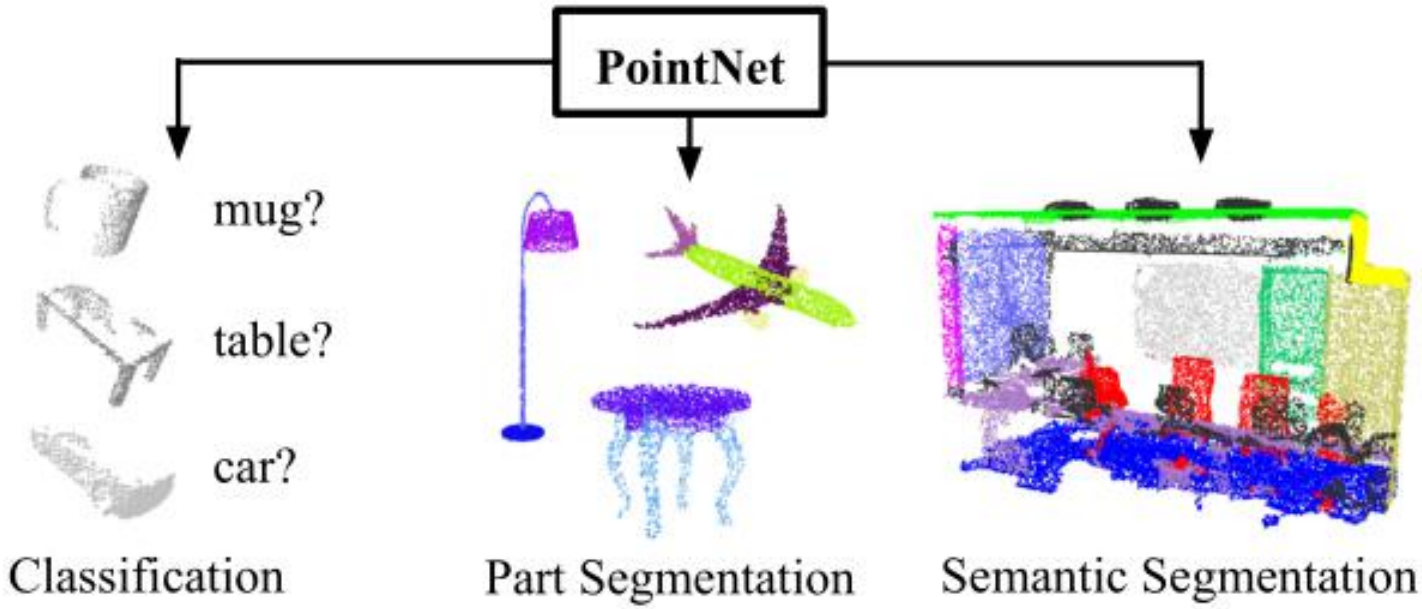
Deep Learning for 3D Point Cloud Analysis



Xu Yan
2020.3.27

Challenges

The model has respect properties of point clouds

Permutation Invariance

- Point cloud is a set of unordered points

Local Feature Representation

- the learned representation should be of discriminative shape awareness

Challenges

The model has respect properties of point clouds

Permutation Invariance

- Point cloud is a set of unordered points

Local Feature Representation

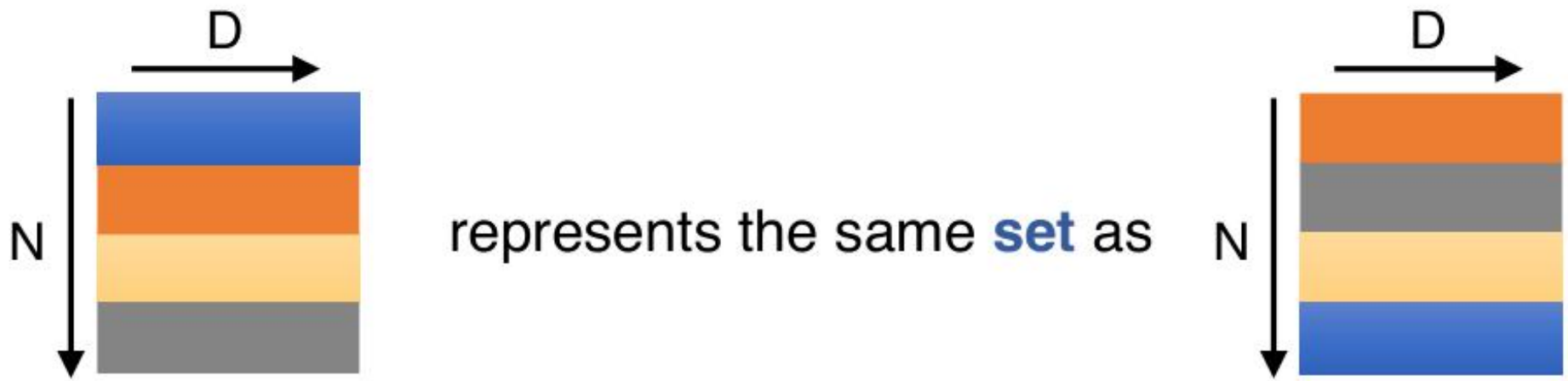
- the learned representation should be of discriminative shape awareness

Challenges

The model has respect properties of point clouds

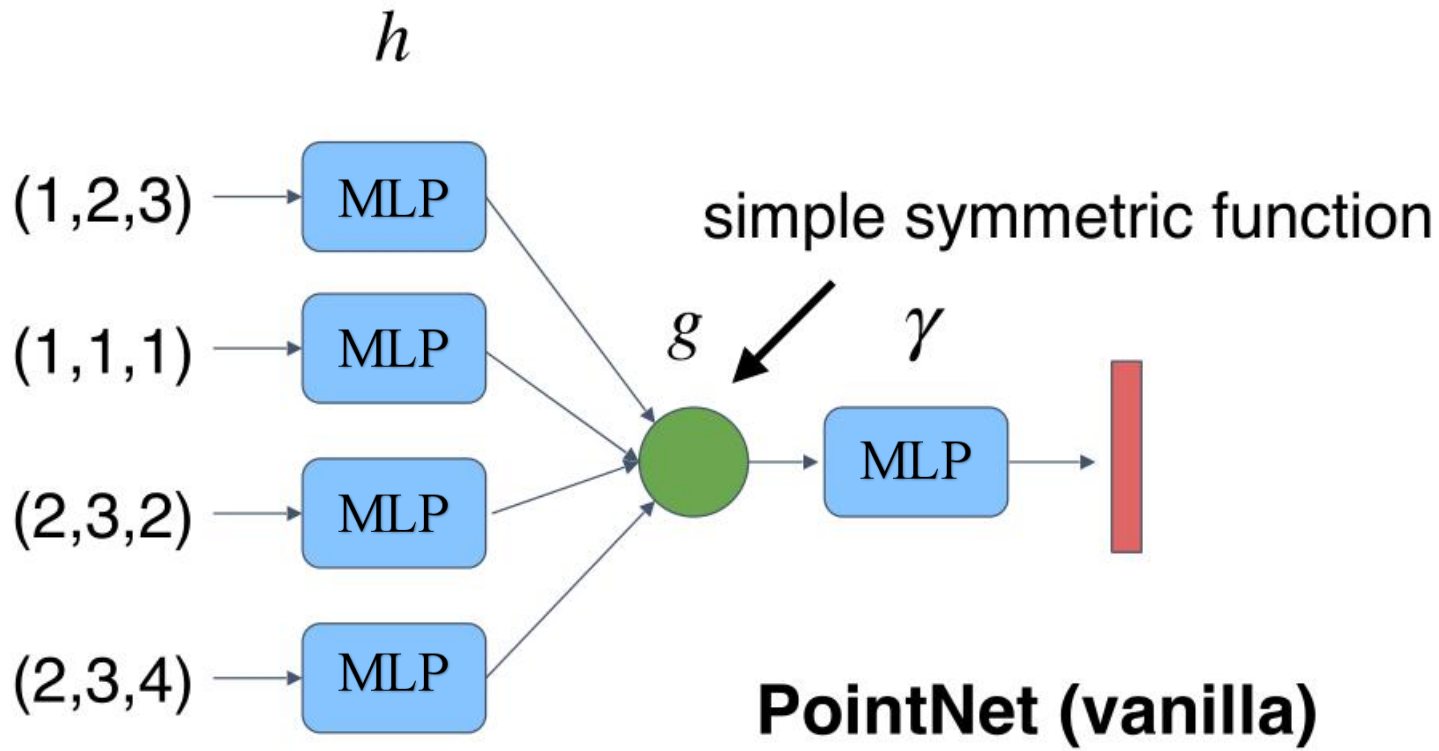
Permutation Invariance

- Point cloud is a set of unordered points



Solution

Symmetric function



Observe: $f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$ is symmetric if g is symmetric

Challenges

The model has respect properties of point clouds

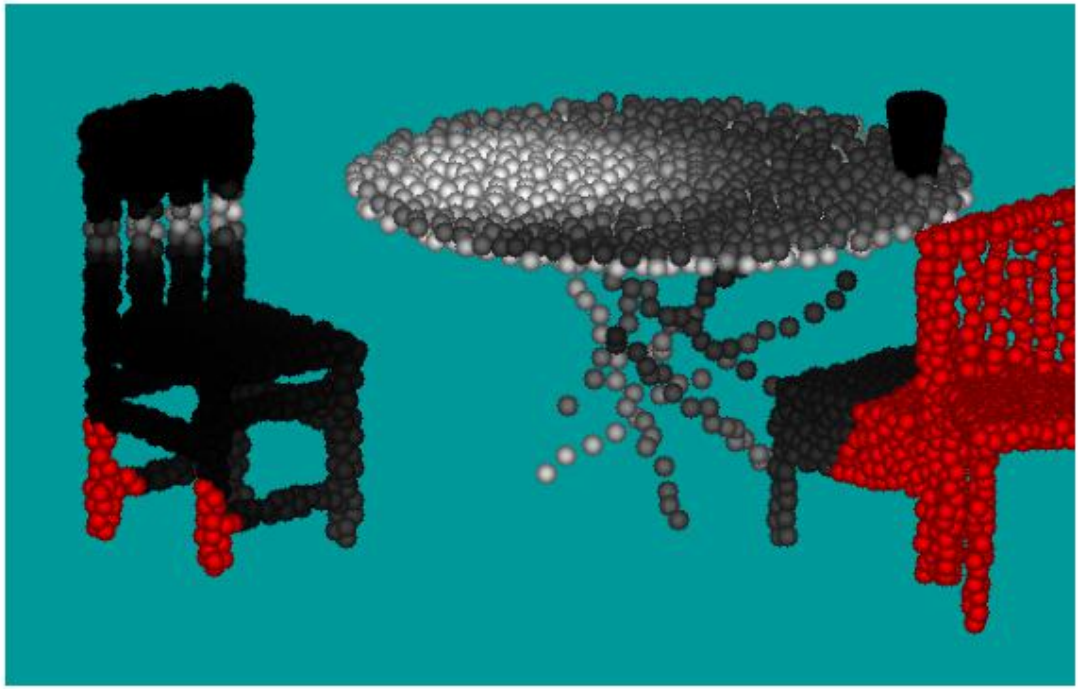
Permutation Invariance

- Point cloud is a set of unordered points

Local Feature Representation

- the learned representation should be of discriminative shape awareness

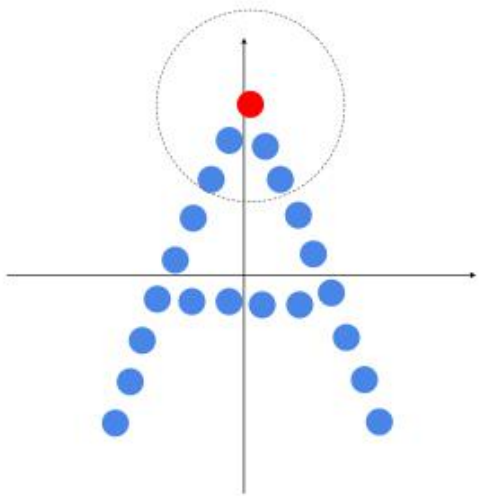
Limitations of PointNet



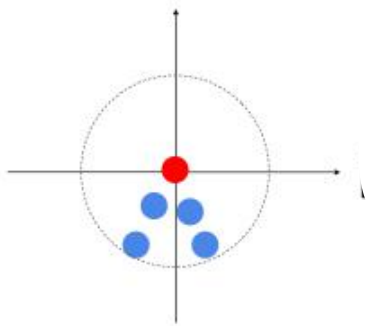
Instance segmentation in table-chair-cup scene

- Global feature depends on absolute XYZ
- No local information

PointNet++

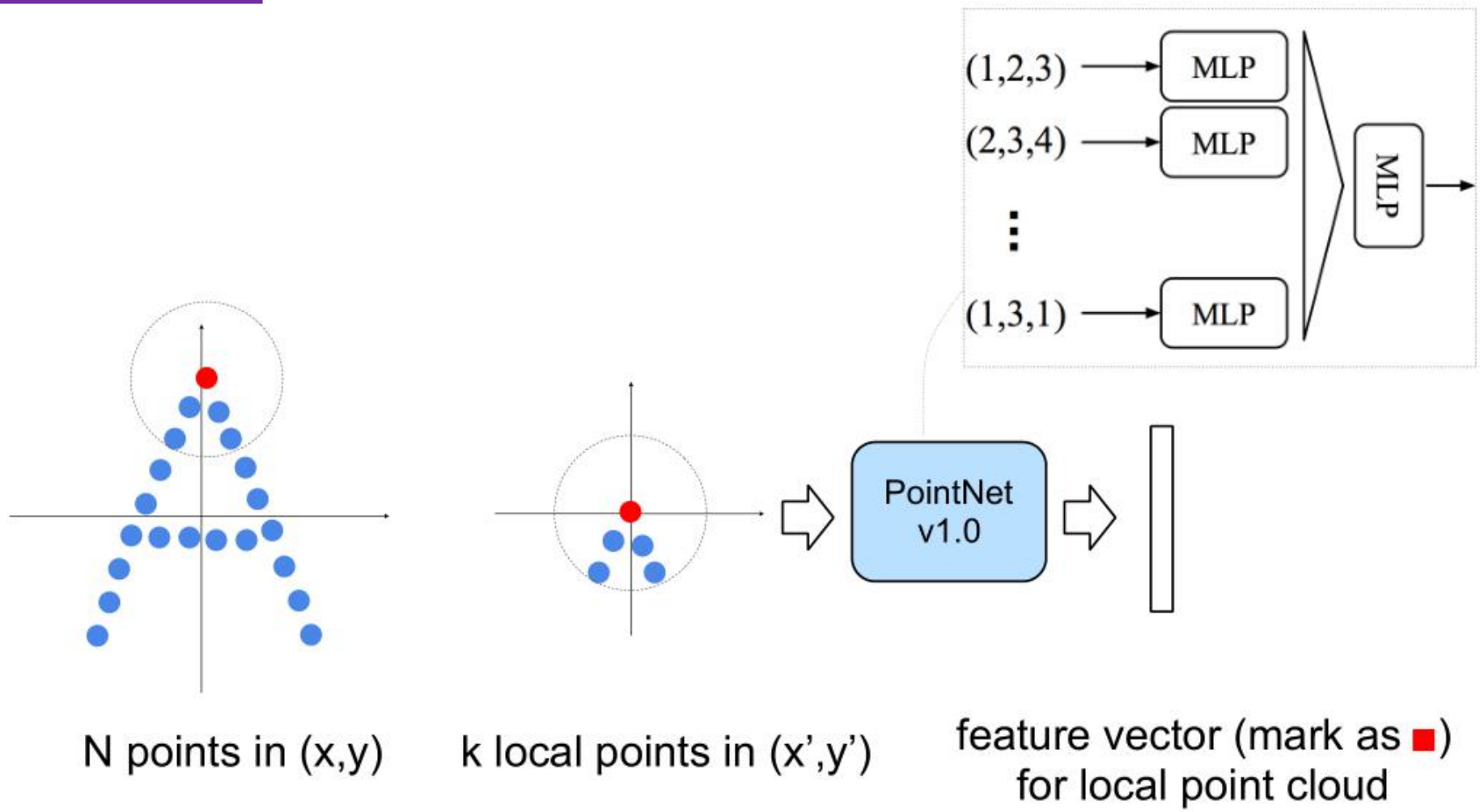


N points in (x,y)

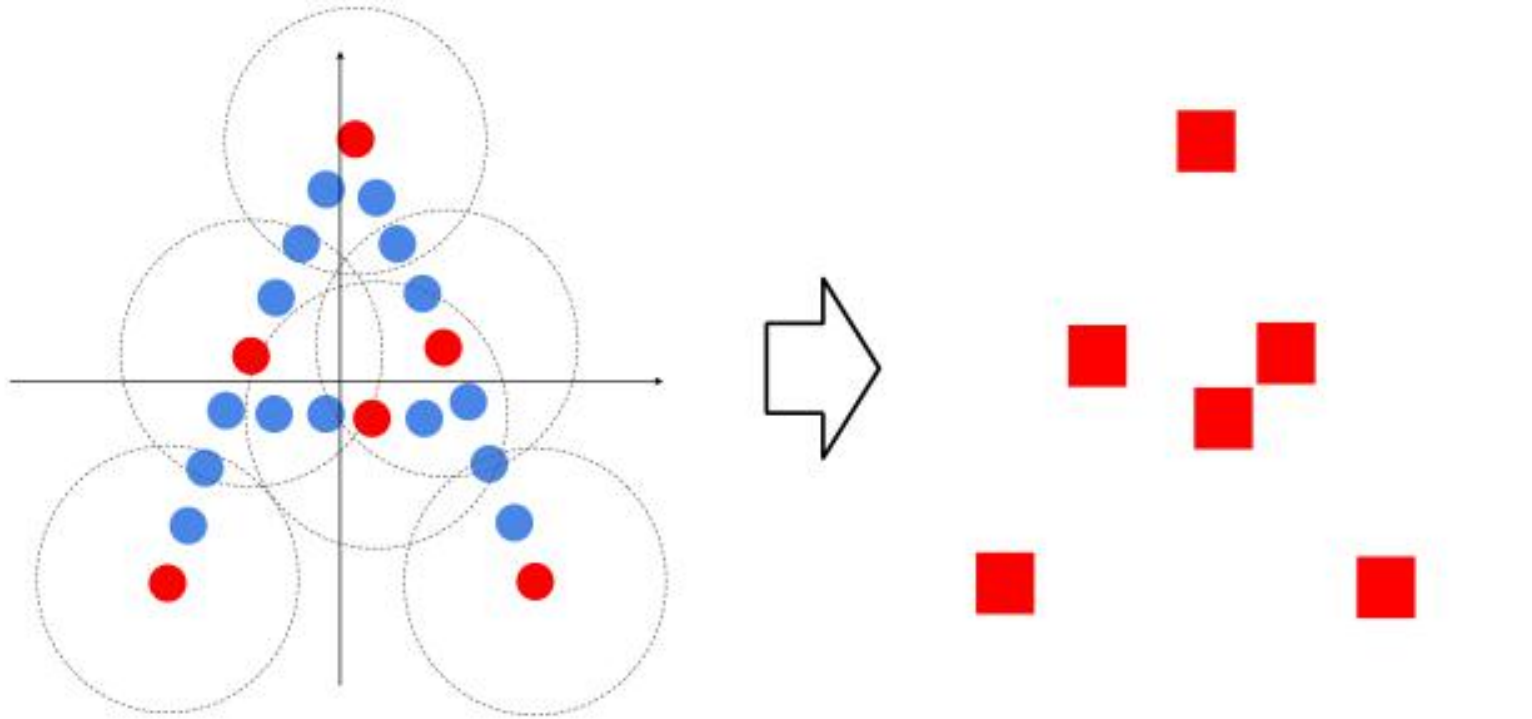


k local points in (x’,y’)

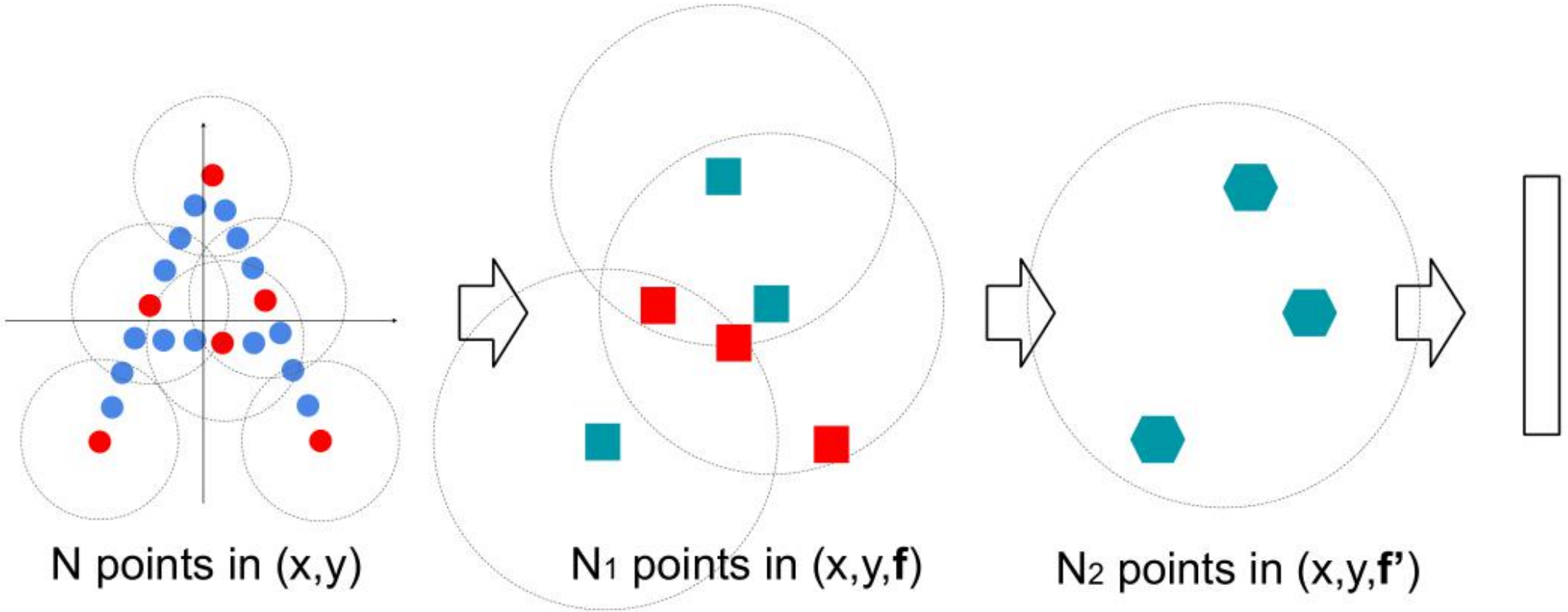
PointNet++


(1,2,3)
MLP
(2,3,4)
MLP
(1,3,1)
MLP
MLP
PointNet
v1.0
N points in (x,y)
k local points in (x',y')
feature vector (mark as ■)
for local point cloud

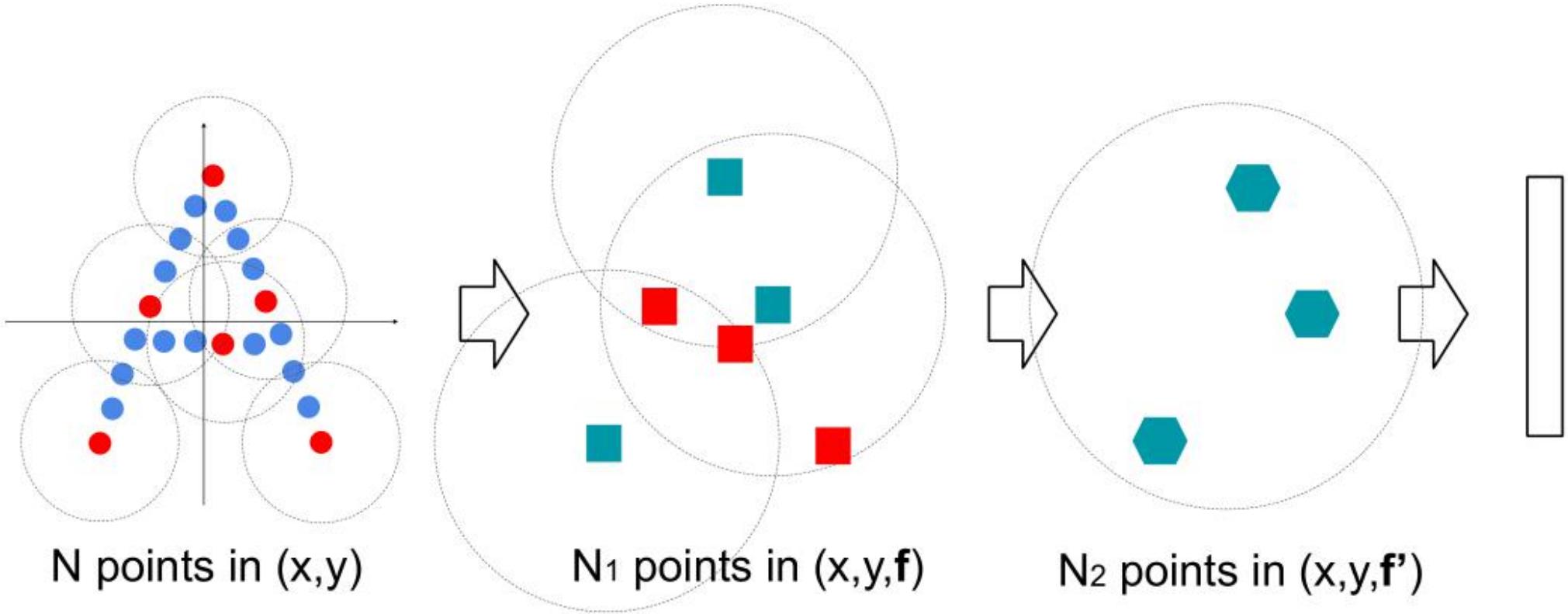
PointNet++



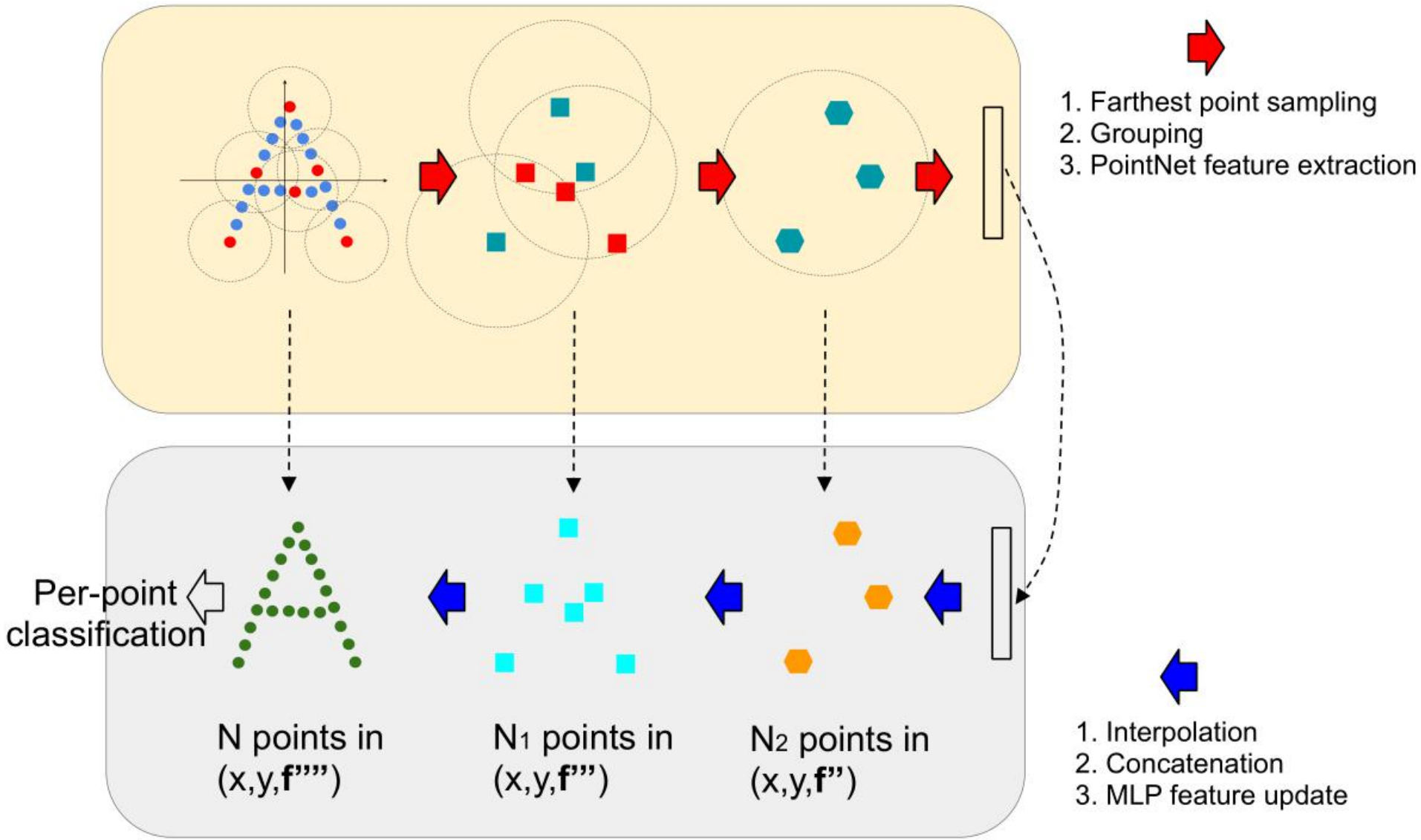
PointNet++



PointNet++



PointNet++


1. Farthest point sampling
2. Grouping
3. PointNet feature extraction
Per-point classification
N points in (x,y,f””)
N1 points in (x,y,f”’)
N2 points in (x,y,f”)
1. Interpolation
2. Concatenation
3. MLP feature update

Drawback of PointNet++

- Simple **Max Pooling** for local feature extraction
- **Structuring** method has O(N^2) complexity when sample large scale point cloud and very sensitive to noise

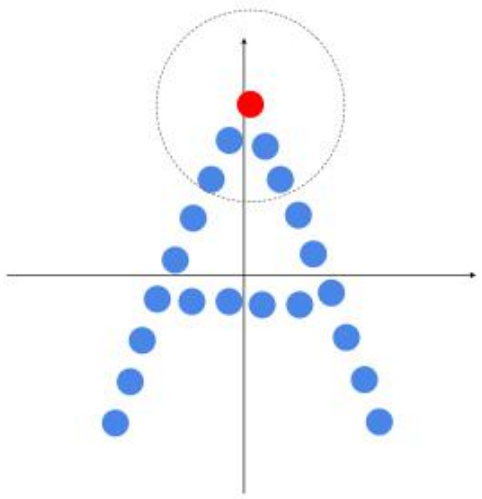
Recent works

- Defining new and more flexible **local feature aggregation** method.
- Exploiting more proper **sampling strategy** to improve the effect of feature aggregation

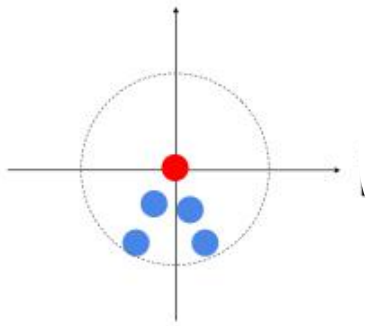
Recent works

- Defining new and more flexible **local feature aggregation** method.
- Exploiting more proper **sampling strategy** to improve the effect of feature aggregation

Recent Works



N points in (x,y)



k local points in (x',y')

Local Feature Aggregation

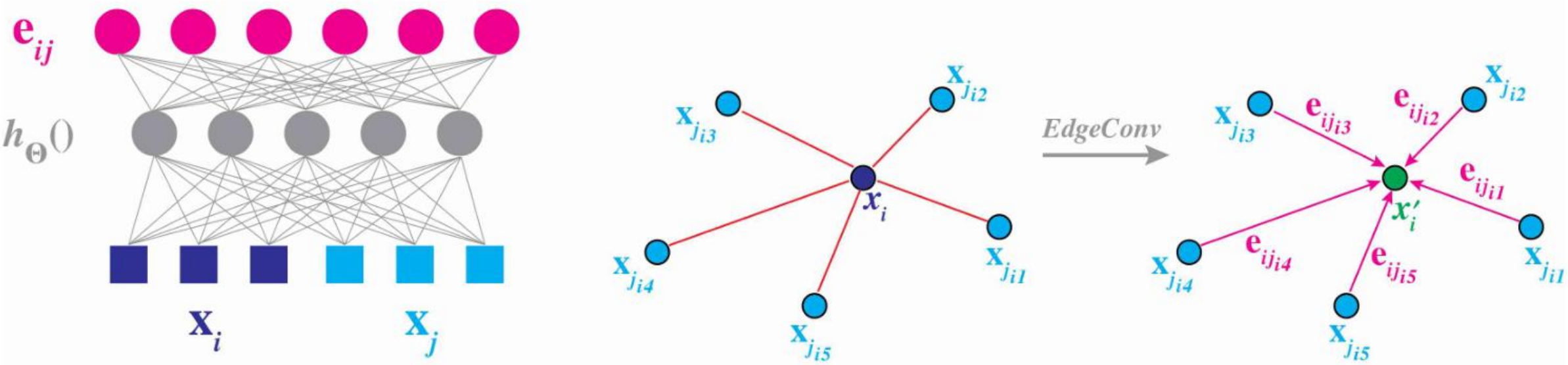
Recent works

Graph Convolution for local neighborhoods

Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs

Martin Simonovsky
Université Paris Est, École des Ponts ParisTech
martin.simonovsky@enpc.fr

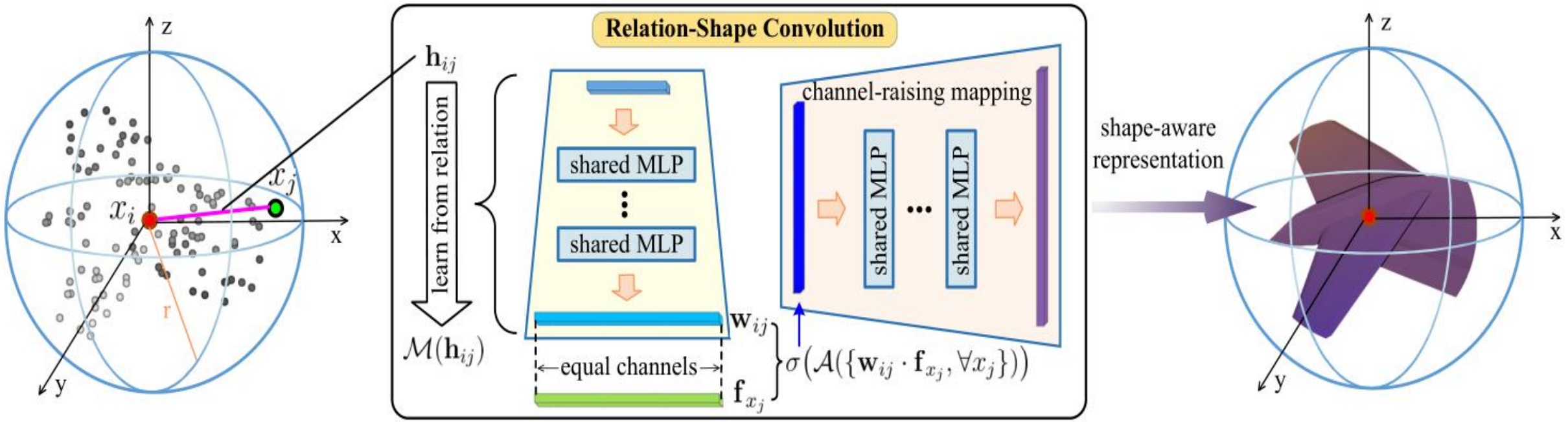
Nikos Komodakis
Université Paris Est, École des Ponts ParisTech
nikos.komodakis@enpc.fr



Recent works

Relation-Shape Convolutional Neural Network for Point Cloud Analysis

Yongcheng Liu[†‡] Bin Fan[*†] Shiming Xiang[†‡] Chunhong Pan[†]
[†] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[‡] School of Artificial Intelligence, University of Chinese Academy of Sciences
Email: {yongcheng.liu, bfan, smxiang, chpan}@nlpr.ia.ac.cn



Recent works

Results with different relations

model	low-level relation **h**	channels	acc.
A	(3D-Ed)	1	92.5
B	(3D-Ed, $x_i - x_j$)	4	93.0
C	(3D-Ed, $x_i - x_j, x_i, x_j$)	10	**93.6**
D	(3D-cosd, $x_i^{\text{nor}}, x_j^{\text{nor}}$)	7	92.8
E	(2D-Ed, $x'_i - x'_j, x'_i, x'_j$)	10	≈ 92.2

Recent works

- Defining new and more flexible **local feature aggregation** method.

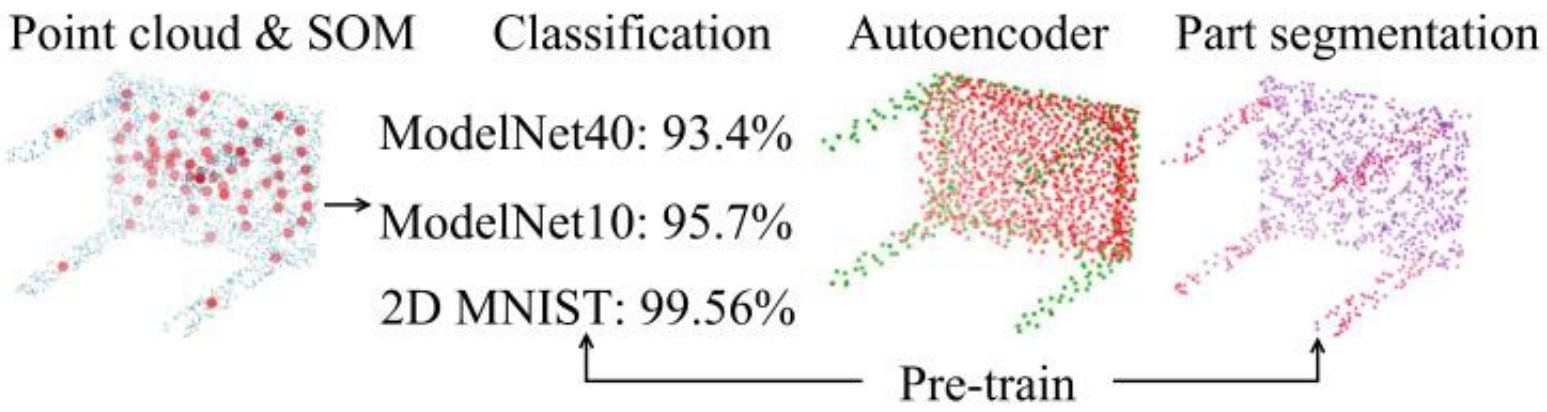
- Exploiting more proper **sampling strategy** to improve the effect of feature aggregation or speed

Recent works

- **Sampled by Self-Organizing Map (SOM)**

SO-Net: Self-Organizing Network for Point Cloud Analysis

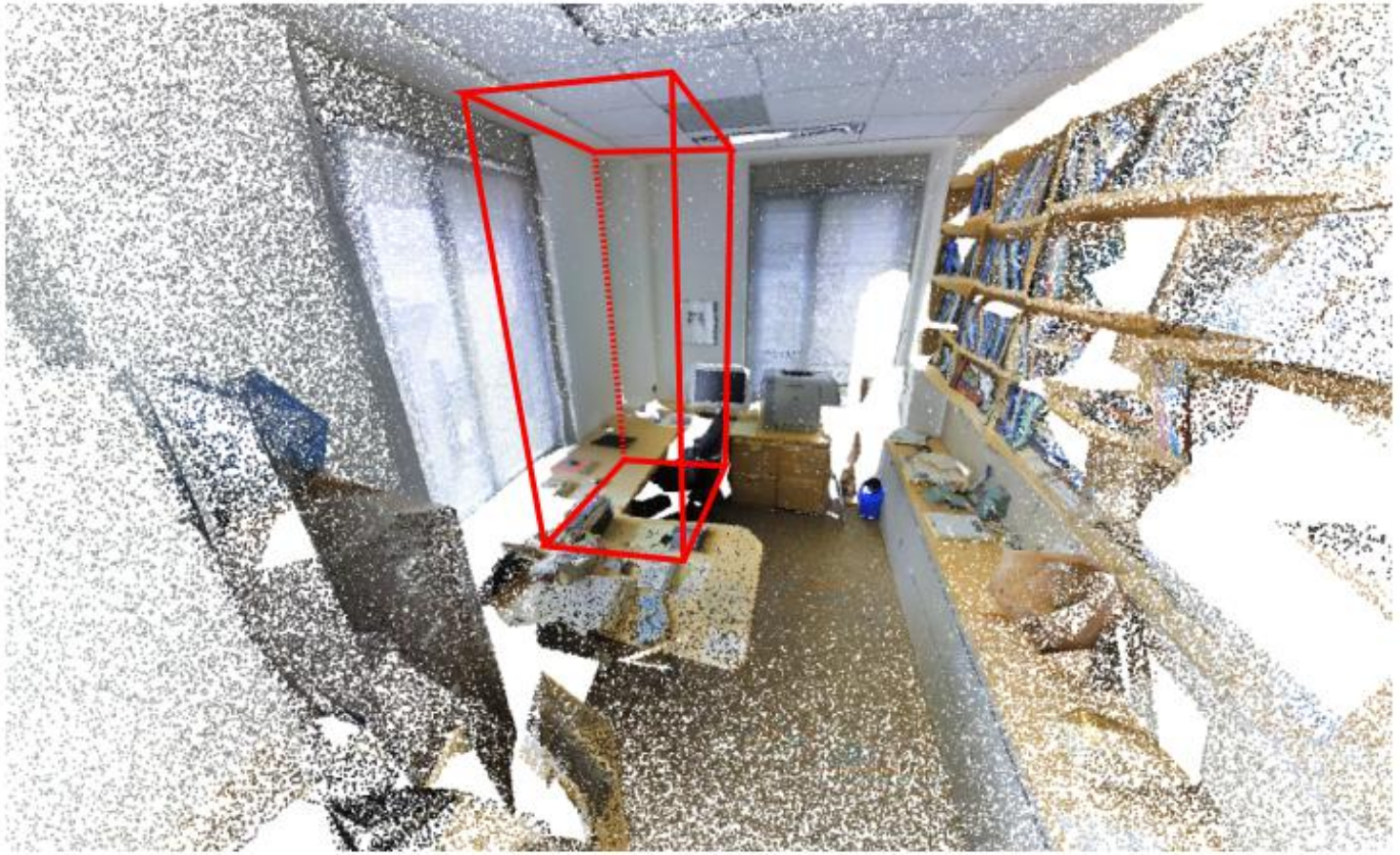
Jiaxin Li Ben M. Chen Gim Hee Lee
National University of Singapore



Recent works

- Still training via chopped cubes

4’096 points



CVPR2020

- RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds (oral)
- Grid-GCN for Fast and Scalable Point Cloud Learning
- FPConv: Learning Local Flattening for Point Convolution
- PointASNL: Robust Point Clouds Processing using Nonlocal Neural Networks with Adaptive Sampling
- PointAugment: an Auto-Augmentation Framework for Point Cloud Classification (oral)

CVPR2020

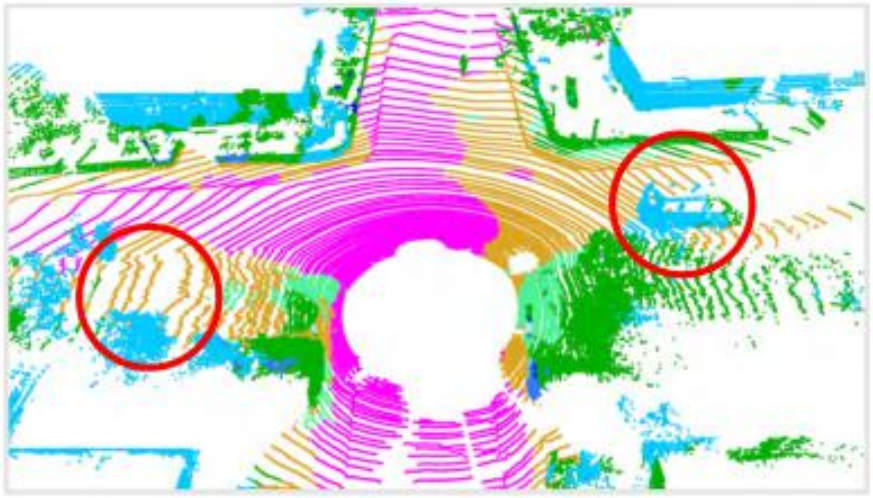
- **RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds**
- Grid-GCN for Fast and Scalable Point Cloud Learning
- FPConv: Learning Local Flattening for Point Convolution
- PointASNL: Robust Point Clouds Processing using Nonlocal Neural Networks with Adaptive Sampling
- PointAugment: an Auto-Augmentation Framework for Point Cloud Classification

CVPR2020

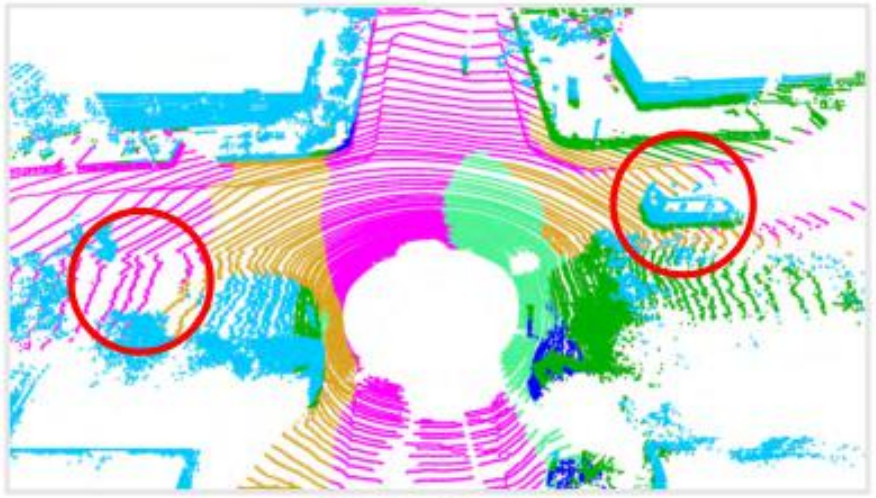
RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds

Qingyong Hu[1], Bo Yang[1*], Linhai Xie[1], Stefano Rosa[1], Yulan Guo[2,3],
Zhihua Wang[1], Niki Trigoni[1], Andrew Markham[1]
[1]University of Oxford, [2]Sun Yat-sen University, [3]National University of Defense Technology
firstname.lastname@cs.ox.ac.uk

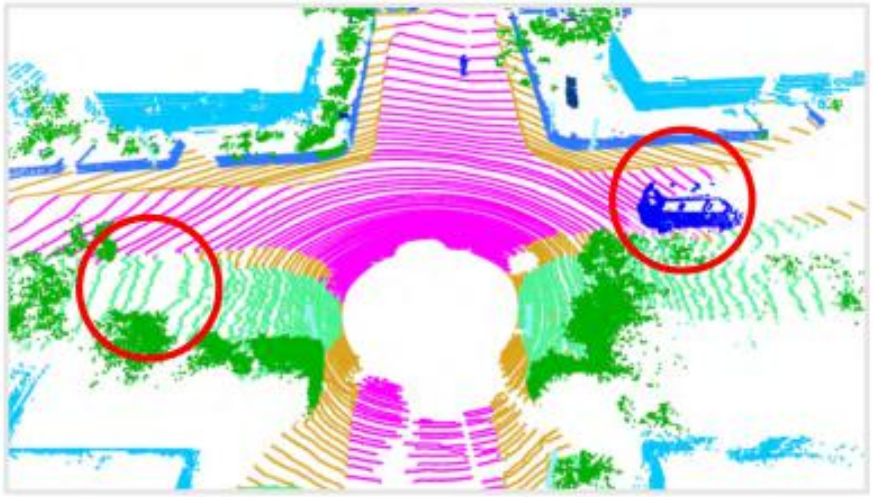
RandLA-Net



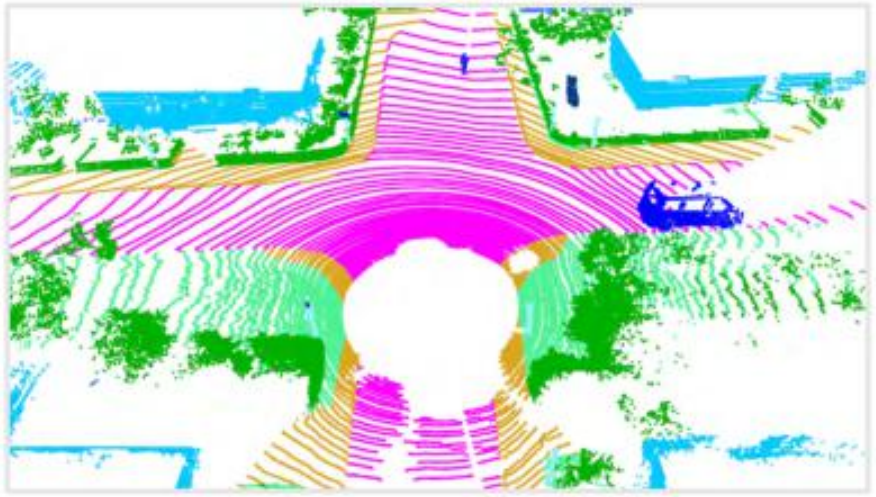
PointNet++ (**2.4s**)



SPG (**10.8s**)

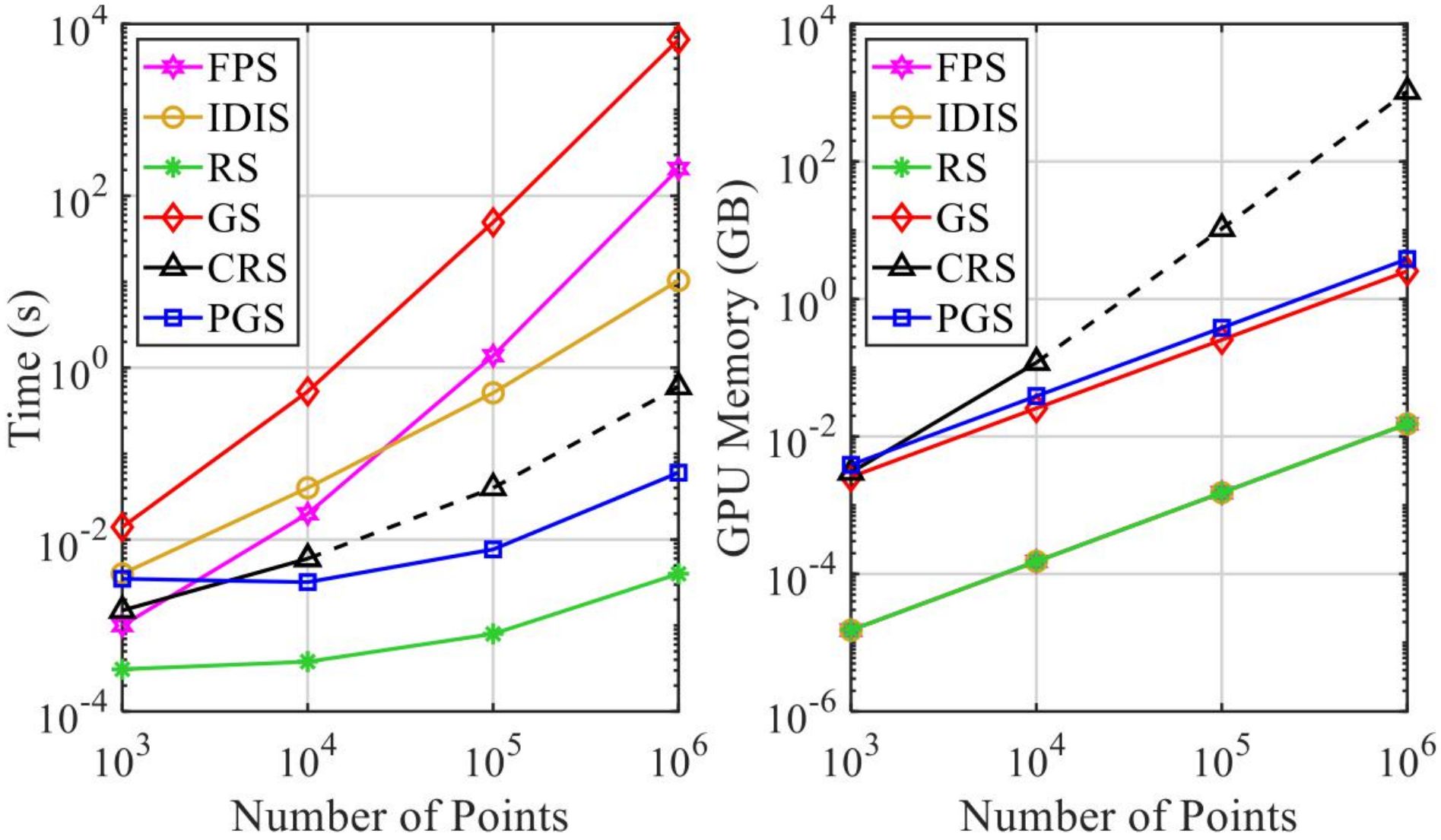


Ours (**0.04s**)

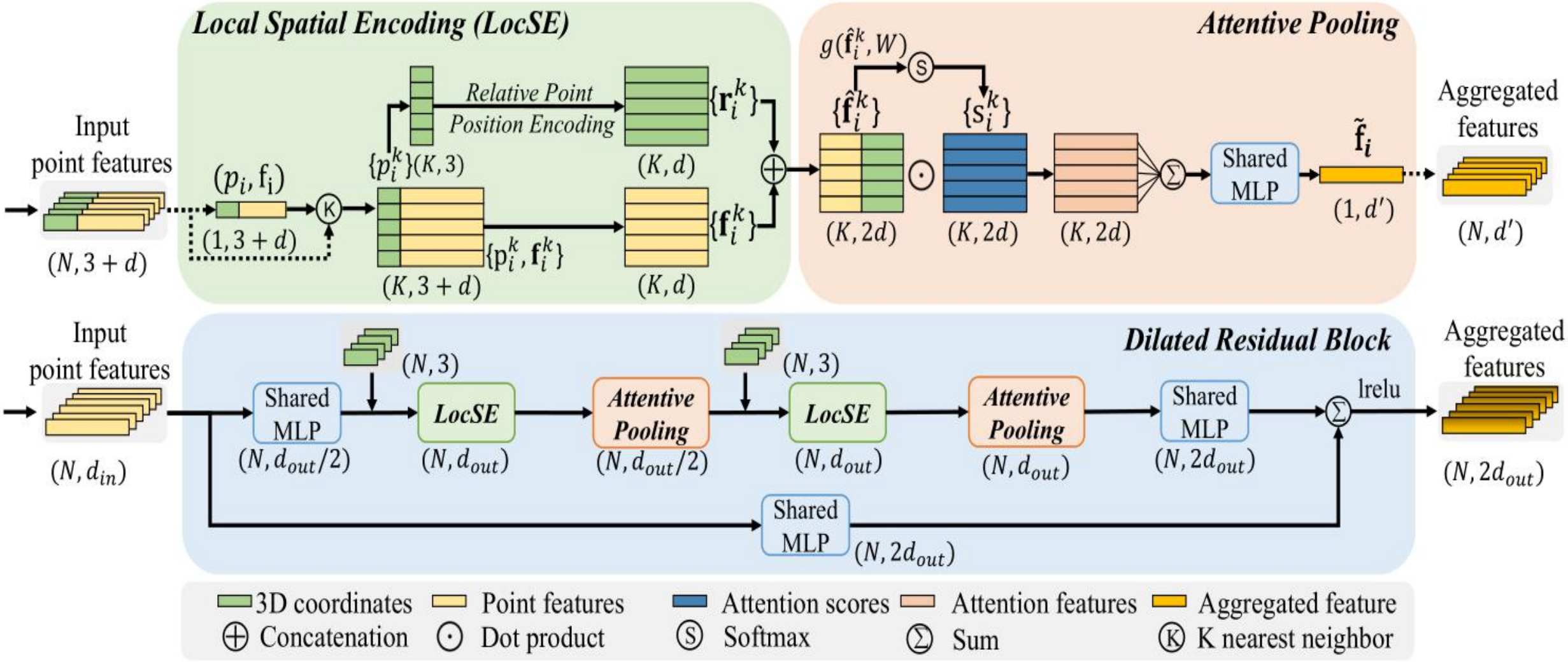


Ground Truth

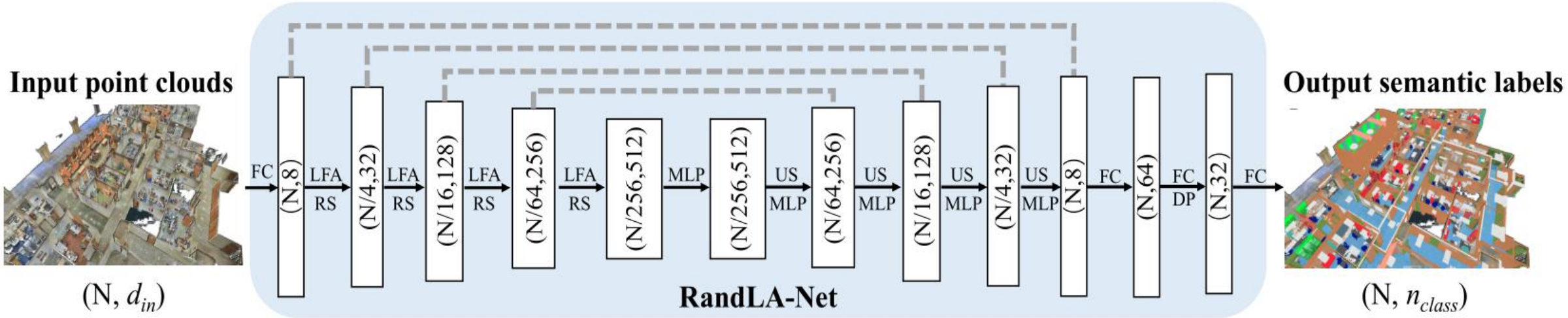
RandLA-Net

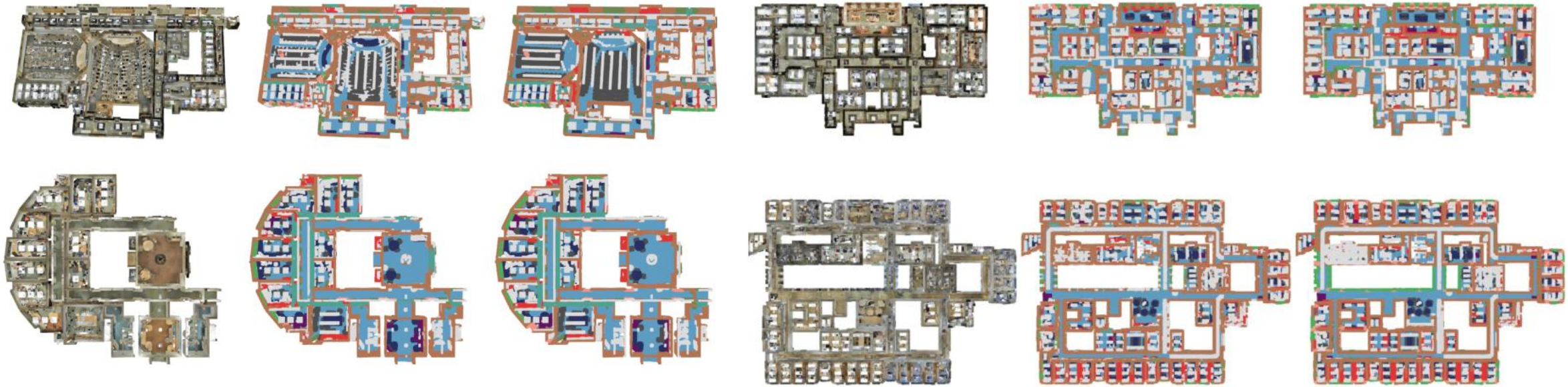

FPS
IDIS
RS
GS
CRS
PGS
Time (s)
Number of Points
GPU Memory (GB)
Number of Points

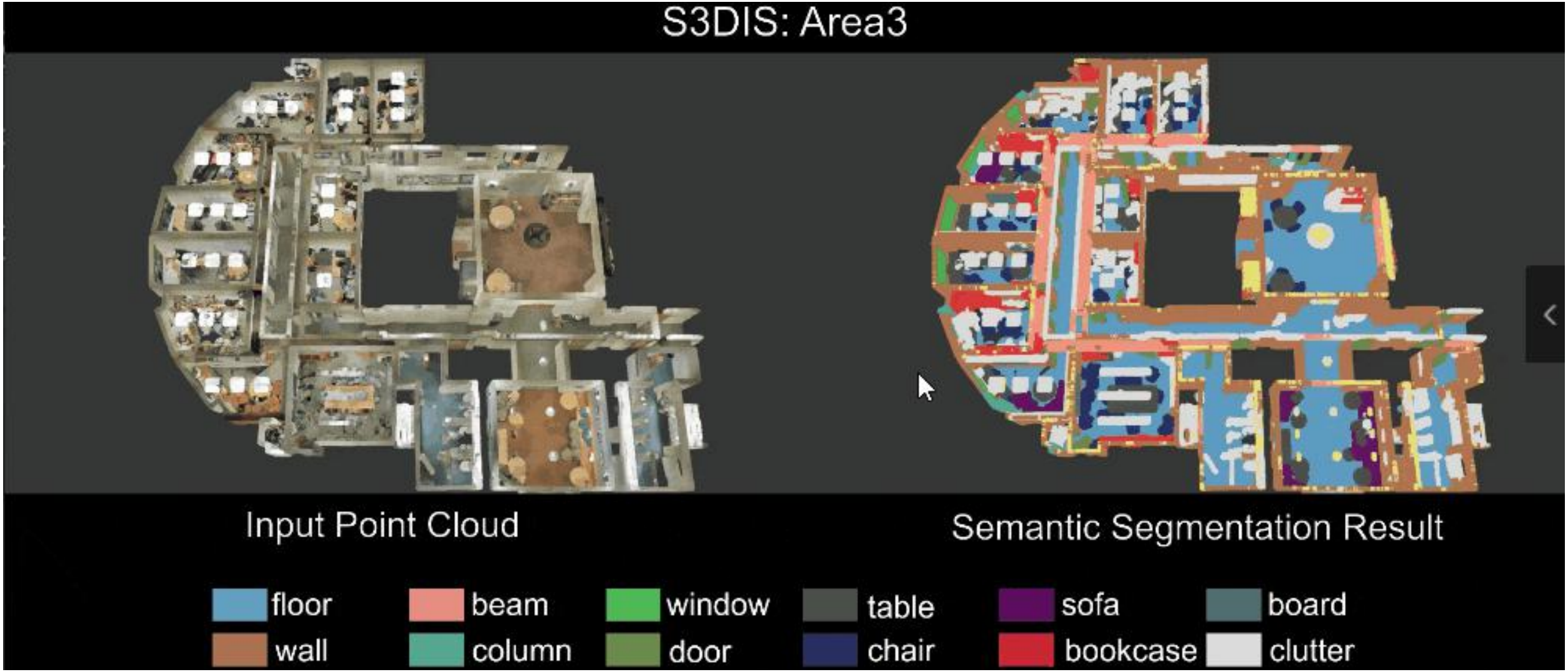
RandLA-Net

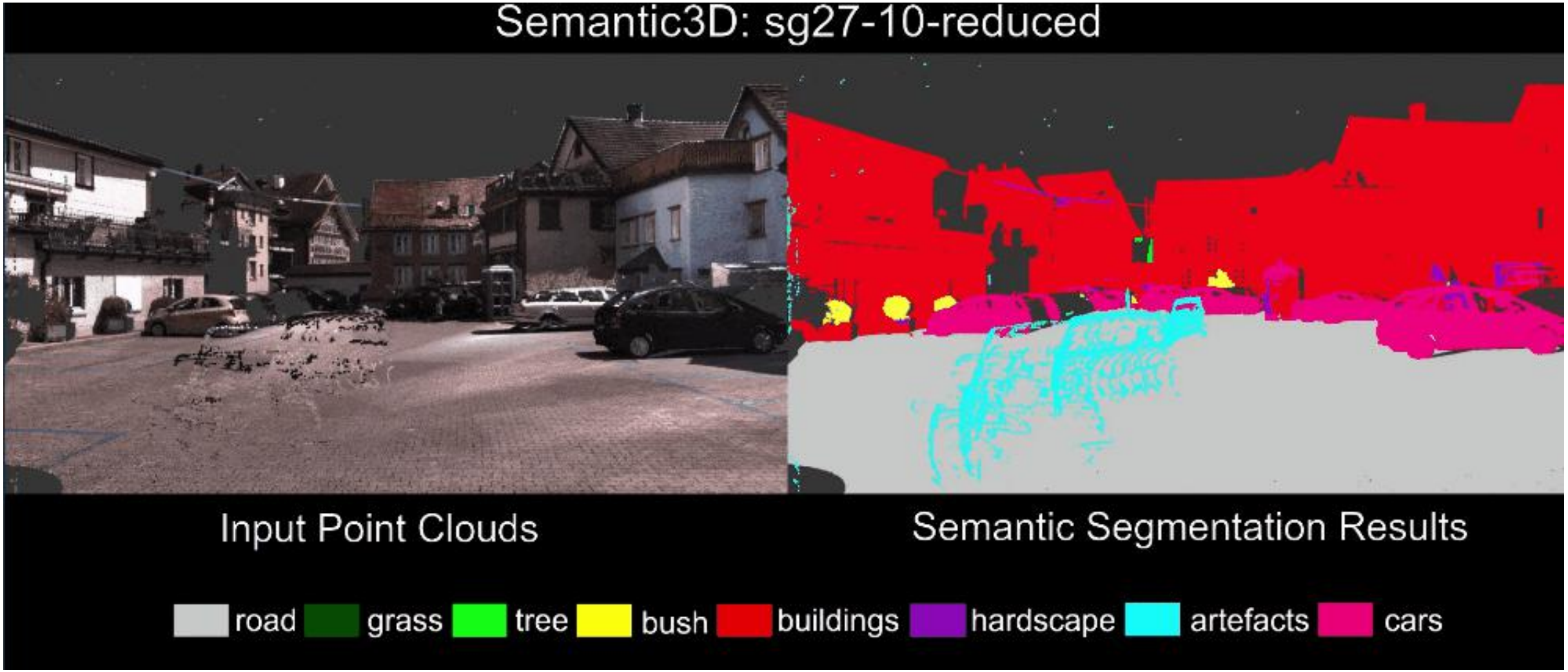

Local Spatial Encoding (LocSE)
Attentive Pooling
Input point features
$(N, 3+d)$
(p_i, f_i)
$(1, 3+d)$
K
$\{p_i^k\}(K,3)$
Relative Point Position Encoding
$\{r_i^k\}$
(K,d)
$\{p_i^k, f_i^k\}$
$(K, 3+d)$
$\{f_i^k\}$
(K,d)
$g(\hat{f}_i^k, W)$
S
$\{\hat{f}_i^k\}$
$\{s_i^k\}$
$(K,2d)$
$(K,2d)$
$(K,2d)$
Shared MLP
$\tilde{f}_i$
$(1,d')$
Aggregated features
(N,d')
Input point features
(N, d_{in})
Shared MLP
$(N, d_{out}/2)$
$(N,3)$
LocSE
(N, d_{out})
Attentive Pooling
$(N, d_{out}/2)$
$(N,3)$
LocSE
(N, d_{out})
Attentive Pooling
(N, d_{out})
Shared MLP
$(N, 2d_{out})$
lrelu
Aggregated features
$(N, 2d_{out})$
Shared MLP
$(N, 2d_{out})$
Dilated Residual Block
3D coordinates
Point features
Attention scores
Attention features
Aggregated feature
Concatenation
Dot product
Softmax
Sum
K nearest neighbor

RandLA-Net


Input point clouds
(N, d_{in})
FC
(N,8)
LFA
RS
(N/4,32)
LFA
RS
(N/16,128)
LFA
RS
(N/64,256)
LFA
RS
(N/256,512)
MLP
(N/256,512)
US
MLP
(N/64,256)
US
MLP
(N/16,128)
US
MLP
(N/4,32)
US
MLP
(N,8)
FC
(N,64)
FC
DP
(N,32)
FC
RandLA-Net
Output semantic labels
(N, n_{class})




S3DIS: Area3
Input Point Cloud
Semantic Segmentation Result
floor
beam
window
table
sofa
board
wall
column
door
chair
bookcase
clutter


Semantic3D: sg27-10-reduced
Input Point Clouds
Semantic Segmentation Results
road
grass
tree
bush
buildings
hardscape
artefacts
cars

RandLA-Net

	mIoU (%)	OA (%)	man-made.	natural.	high veg.	low veg.	buildings	hard scape	scanning art.	cars
SnapNet_ [4]	59.1	88.6	82.0	77.3	79.7	22.9	91.1	18.4	37.3	64.4
SEGCloud [46]	61.3	88.1	83.9	66.0	86.0	40.5	91.1	30.9	27.5	64.3
RF_MSSF [47]	62.7	90.3	87.6	80.3	81.8	36.4	92.2	24.1	42.6	56.6
MSDeepVoxNet [40]	65.3	88.4	83.0	67.2	83.8	36.7	92.4	31.3	50.0	78.2
ShellNet [63]	69.3	93.2	96.3	90.4	83.9	41.0	94.2	34.7	43.9	70.2
GACNet [50]	70.8	91.9	86.4	77.7	**88.5**	**60.6**	94.2	37.3	43.5	77.8
SPG [23]	73.2	94.0	**97.4**	**92.6**	87.9	44.0	83.2	31.0	63.5	76.2
KPConv [48]	74.6	92.9	90.9	82.2	84.2	47.9	94.9	40.0	**77.3**	**79.7**
RandLA-Net (Ours)	**76.0**	**94.4**	96.5	92.0	85.1	50.3	**95.0**	**41.1**	68.2	79.4

Table 2. Quantitative results of different approaches on Semantic3D (reduced-8) [16]. Only the recent published approaches are compared. Accessed on 15 November 2019

	Total time (seconds)	Parameters (millions)	Maximum inference points (millions)
PointNet (Vanilla) [37]	192	0.8	0.49
PointNet++ (SSG) [38]	9831	0.97	0.98
PointCNN [29]	8142	11	0.05
SPG [23]	43584	**0.25**	-
KPConv [48]	717	14.9	0.54
RandLA-Net (Ours)	**176**	0.95	**1.15**

CVPR2020

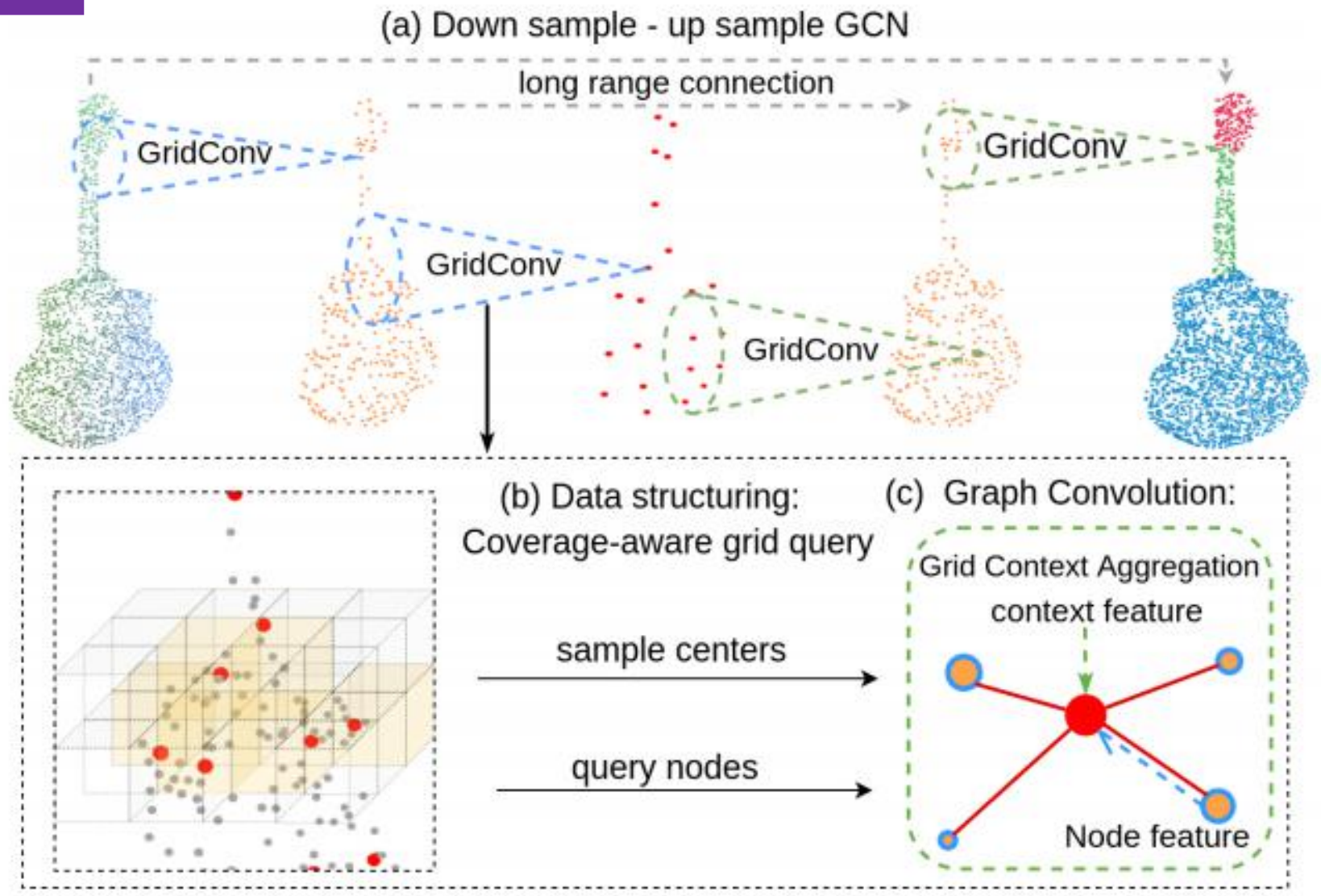
- RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds
- **Grid-GCN for Fast and Scalable Point Cloud Learning**
- FPConv: Learning Local Flattening for Point Convolution
- PointASNL: Robust Point Clouds Processing using Nonlocal Neural Networks with Adaptive Sampling
- PointAugment: an Auto-Augmentation Framework for Point Cloud Classification

CVPR2020

Grid-GCN for Fast and Scalable Point Cloud Learning

Qiangeng Xu[1] Xudong Sun[2] Cho-Ying Wu[1] Panqu Wang[2] Ulrich Neumann[1]
[1]University of Southern California
Los Angeles, California
[2]Tusimple, Inc
San Diego, California

Grid-GCN


(a) Down sample - up sample GCN
long range connection
GridConv
GridConv
GridConv
GridConv
(b) Data structuring:
Coverage-aware grid query
sample centers
query nodes
(c) Graph Convolution:
Grid Context Aggregation
context feature
Node feature

Grid-GCN

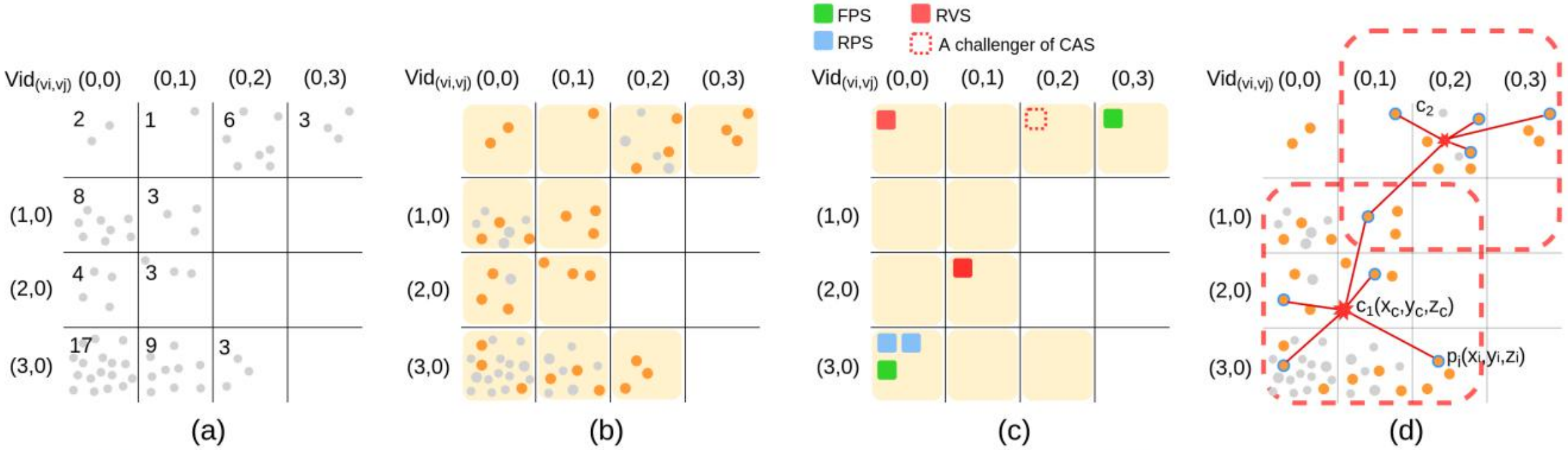


Figure 2: Illustration of Coverage-Aware Grid Query (CAGQ). Assume we want to sample $M = 2$ point groups and query $K = 5$ node points for each group. **(a)** The input is N points (grey). The voxel id and number of points is listed for each occupied voxel. **(b)** We build voxel-point index and store up to $n_v = 3$ points (yellow) in each voxel. **(c)** Comparison of different sampling methods: FPS and RPS prefer the two centers inside the marked voxels. Our RVS could randomly pick any two occupied voxels (e.g. (2,0) and (0,0)) as center voxels. If our CAS is used, voxel (0,2) will replace (0,0). **(d)** Context points of center voxel (2,1) are the yellow points in its neighborhood (we use 3×3 as an example). CAGQ queries 5 points (yellow points with blue ring) from these context points, then calculate the locations of the group centers.

Grid-GCN

1. Random Voxel Sampling (RVS):

2. Coverage-Aware Sampling (CAS):

$$\delta(x) = \begin{cases} 1, & \text{if } x = 0. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$H_{add} = \sum_{V \in \pi(V_C)} \delta(C_V) - \beta \cdot \frac{C_V}{\lambda} \tag{2}$$

$$H_{rmv} = \sum_{V \in \pi(V_I)} \delta(C_V - 1) \tag{3}$$

where λ is the amount of neighbors of a voxel and CV is the number of incumbents covering voxel V .

Grid-GCN

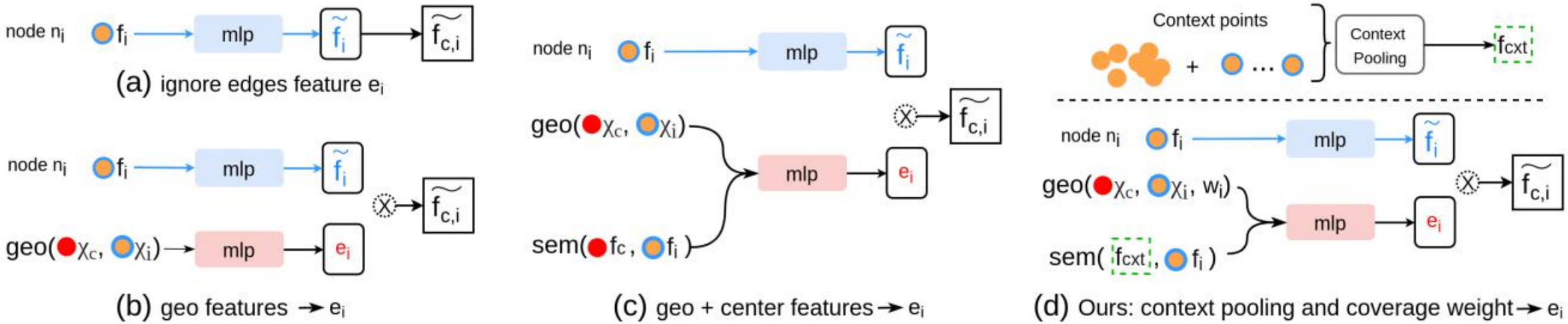
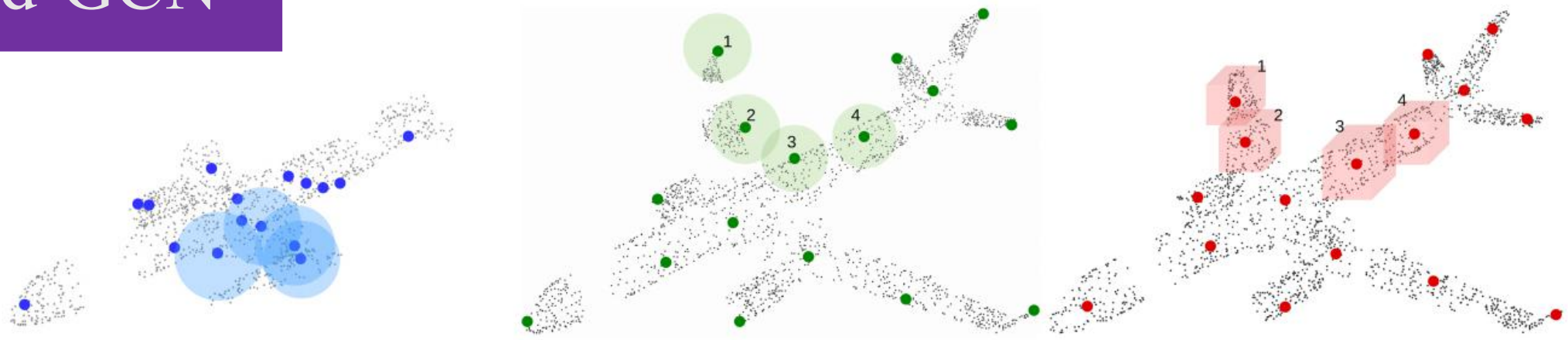


Figure 4: Different strategies to compute the contribution $\tilde{f}_{c,i}$ from a node n_i to its center c. f_i, χ_i are the feature maps and the location of n_i. e_i is the edge feature between n_i and c calculated from the edge attention function. **(a)** Pointnet++ [29] ignores e_i. **(b)** computes e_i based on low dimensional geometric relation between n_i and c. **(c)** also consider semantic relation between the center and the node point, but c has to be sampled on one of the points from the previous layer. **(d)**. Grid-GCN's geo-relation also includes the coverage weight. It pools a context feature f_{cxt} from all stored neighbors to provide a semantic reference in e_i computing.

With that in mind, we introduce the concept of **coverage weight** wi, which is defined as the number of points that have been aggregated to a node in previous layers

Grid-GCN



(a) Random Point Sampling (b) Farthest Point Sampling (c) Coverage-Aware Sampling

Center sampling			RPS	FPS	RVS*	CVS*	RPS	FPS	RVS*	CVS*	RPS	FPS	RVS*	CVS*
Neighbor querying			Ball	Ball	Cube	Cube	Ball	Ball	Cube	Cube	k-NN	k-NN	k-NN	k-NN
N	K	M	Occupied space coverage(%)				Latency (ms) with batch size = 1							
1024	8	8	12.3	12.9	13.1	**14.9**	**0.29**	0.50	0.51	0.74	0.84	0.85	**0.51**	0.77
	8	128	64.0	72.5	82.3	**85.6**	**0.32**	0.78	0.44	0.68	1.47	1.74	**0.52**	0.72
	128	32	60.0	70.1	61.0	**74.7**	**0.37**	0.53	0.96	1.18	22.23	21.08	**2.24**	2.74
	128	128	93.6	99.5	95.8	**99.7**	**0.38**	0.69	1.03	1.17	32.48	32.54	**6.85**	7.24
8192	8	64	19.2	22.9	22.1	**25.1**	**0.64**	1.16	0.66	0.82	1.58	1.80	**0.65**	0.76
	8	1024	82.9	**96.8**	92.4	94.4	0.81	4.90	**0.54**	0.87	1.53	5.36	**0.93**	0.97
	128	256	79.9	90.7	80.0	**93.5**	1.19	1.19	**1.17**	1.41	21.5	21.5	**15.19**	17.68
	128	1024	98.8	99.9	99.5	**100.0**	**1.22**	5.25	1.40	1.76	111.4	111.7	**24.18**	27.65
81920	32	1024	70.6	86.3	78.3	**91.6**	8.30	33.52	**3.34**	6.02	19.49	43.69	**8.76**	10.05
	32	10240	98.8	99.2	**100.0**	**100.0**	8.93	260.48	**4.22**	9.35	20.38	272.48	**9.65**	17.44
	128	1024	72.7	88.2	79.1	**92.6**	9.68	34.72	**4.32**	8.71	71.99	93.02	**50.7**	61.94
	128	10240	99.7	**100.0**	**100.0**	**100.0**	10.73	258.49	**5.83**	11.72	234.19	442.87	**69.02**	83.32

Grid-GCN

Input (xyz as default)		ModelNet40 OA	ModelNet40 mAcc	ModelNet10 OA	ModelNet10 mAcc	latency (ms)
OA $\leqslant$ 91.5						
PointNet[28]	16×1024	89.2	86.2	-	-	**15.0**
SCNet[43]	16×1024	90.0	87.6	-		-
SpiderCNN[45]	8 × 1024	90.5	-	-	-	85.0
O-CNN[39]	octree	90.6	-	-	-	90.0
SO-net[21]	8 × 2048	90.8	87.3	**94.1**	**93.9**	-
Grid-GCN[1]	16×1024	**91.5**	**88.6**	93.4	92.1	15.9
OA $\leqslant$ 92.0						
3DmFVNet[3]	16×1024	91.6	-	95.2	-	39.0
PAT[46]	8 × 1024	91.7	-		-	88.6
Kd-net[14]	kd-tree	91.8	88.5	94.0	93.5	-
PointNet++[29]	16×1024	91.9	**90.7**		-	26.8
Grid-GCN[2]	16×1024	**92.0**	89.7	**95.8**	**95.3**	**21.8**
OA > 92.0						
DGCNN[40]	16×1024	92.2	90.2		-	89.7
PCNN[2]	16×1024	92.3	-	94.9	-	226.0
Point2Seq[23]	16×1024	92.6	-			-
A-CNN[15]	16×1024	92.6	90.3	95.5	95.3	68.0
KPConv[35]	16×6500	92.7	-	-	-	125.0
Grid-GCN[3]	16×1024	92.7	90.6	96.5	95.7	**26.2**
Grid-GCN^{full}	16×1024	**93.1**	**91.3**	**97.5**	**97.4**	42.2

Num. of points (N)	2048	4096	16384	40960	81920
Num. of clusters (M)	512	1024	2048	4096	8192
PointNet++	4.7	8.6	19.9	64.6	218.9
Grid-GCN	4.3	4.7	8.1	12.3	19.8

Table 7: Inference time (ms) on ScanNet[7] under different scales. We compare Grid-GCN with PoinNet++[29] on different numbers of input points per scene. The batch size is 1. M is the number of point groups on the first network layer.

Input (xyz as default)		OA	latency (ms)
OA < 84.0			
PointNet[28]	8 × 4096	73.9	20.3
OctNet[30]	volume	76.6	-
PointNet++[29]	8 × 4096	83.7	72.3
Grid-GCN$_{(0.5\times K)}$	4 × 8192	**83.9**	**16.6**
OA $\geqslant$ 84.0			
SpecGCN[36]	-	84.8	-
PointCNN[22]	12×2048	85.1	250.0
Shellnet[48]	-	85.2	-
Grid-GCN$_{(1\times K)}$	4 × 8192	**85.4**	**20.8**
A-CNN[15]	1 × 8192	**85.4**	92.0
Grid-GCN$_{(1\times K)}$	1 × 8192	**85.4**	**7.48**

Table 4: Results on ScanNet[7]. Grid-GCN achieves 10× speed-up on average over other models. Under batch size of 4 and 1, we test our model with $1\times K$ neighbor nodes. A compact model with $0.5 \times K$ is also reported.

Input (xyzrgb as default)		mIOU	OA	latency(ms)
mIOU< 54.0				
PointNet[28]	8 × 4096	41.09	-	20.9
DGCNN[40]	8 × 4096	47.94	83.64	178.1
SegCloud[34]	-	48.92	-	-
RSNet[13]	8 × 4096	51.93	-	111.5
PointNet++[29]	8 × 4096	52.28	-	
DeepGCNs[20]	1 × 4096	52.49	-	45.63
TanConv[33]	8 × 4096	52.8	85.5	-
Grid-GCN$_{(0.5\times Ch)}$	8 × 4096	**53.21**	**85.61**	**12.9**
mIOU> 54.0				
3D-UNet[5]	8 × 96^3 volume	54.93	86.12	574.7
PointCNN[22]	-	57.26	85.91	-
PVCNN++[26]	8 × 4096	57.63	86.87	41.1
Grid-GCN$_{(1\times Ch)}$	8 × 4096	**57.75**	**86.94**	**25.9**

Table 5: Results on S3DIS[1] area 5. Grid-GCN is on average 8× faster than other models. We halve the output channels of GridConv for Grid-GCN$_{(0.5\times Ch)}$.

CVPR2020

- RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds
- Grid-GCN for Fast and Scalable Point Cloud Learning
- **FPConv: Learning Local Flattening for Point Convolution**
- PointASNL: Robust Point Clouds Processing using Nonlocal Neural Networks with Adaptive Sampling
- PointAugment: an Auto-Augmentation Framework for Point Cloud Classification

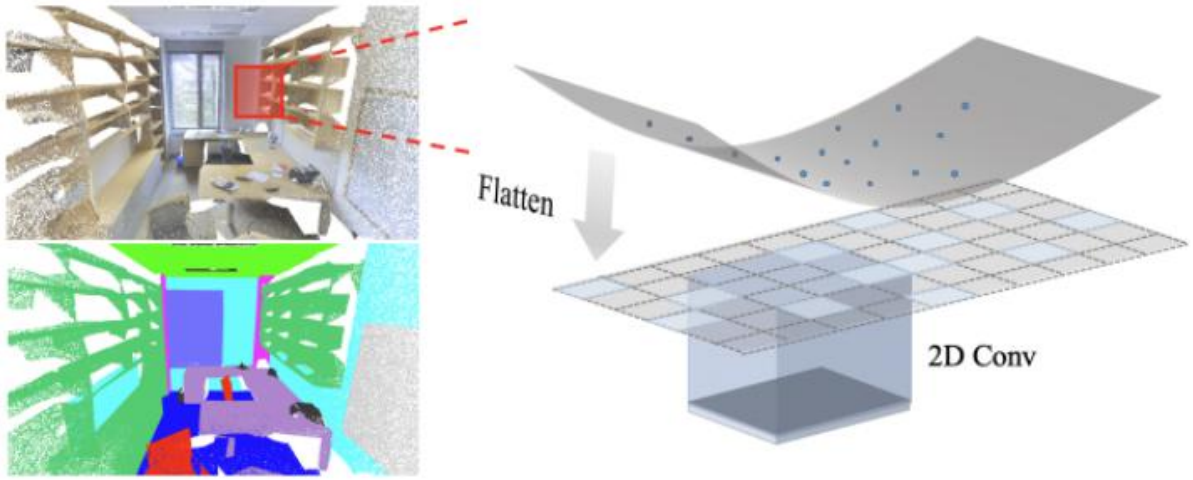
CVPR2020

FPConv: Learning Local Flattening for Point Convolution

Yiqun Lin[1,2], Zizheng Yan[1,2], Haibin Huang[4], Dong Du[2,3], Ligang Liu[3],
Shuguang Cui[1,2], Xiaoguang Han*[1,2]

[1]The Chinese University of Hong Kong, Shenzhen
[2]Shenzhen Research Institute of Big Data
[3]University of Science and Technology of China [4]Kuaishou Technology

FPConv



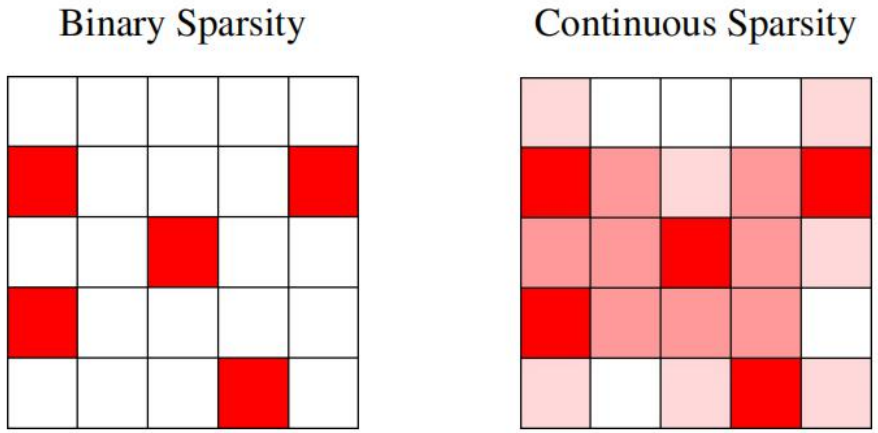
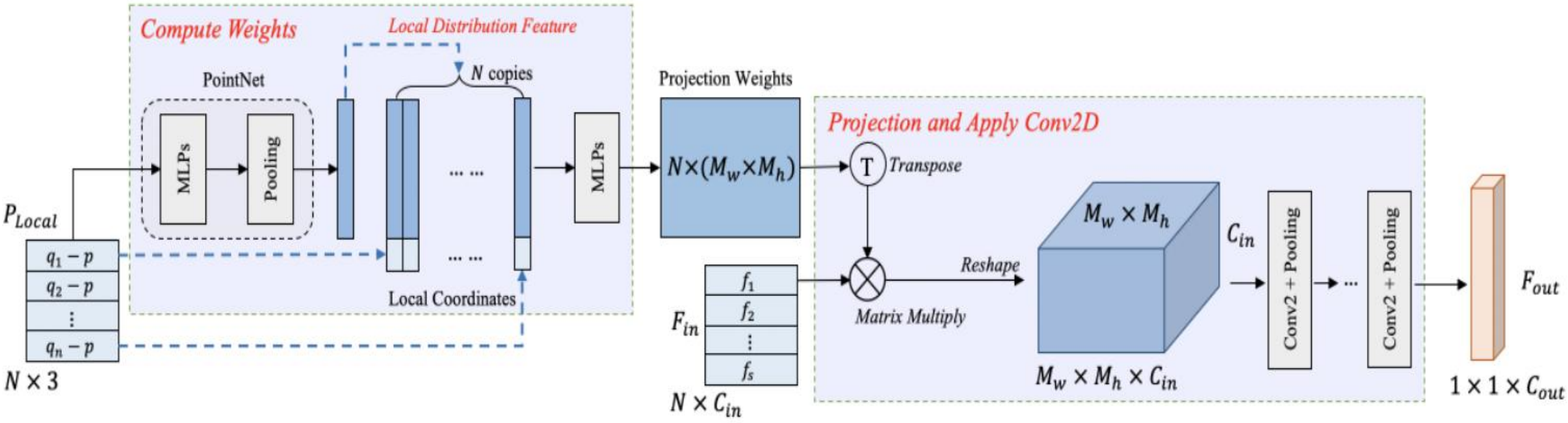


Figure 4: Left: binary sparsity, intensity at each position should be 0 or 1. Right: continuous sparsity, intensity can be in the range of 0 to 1.

FPConv


Compute Weights
Local Distribution Feature
PointNet
MLPs
Pooling
N copies
... ...
... ...
Local Coordinates
MLPs
Projection Weights
$N \times (M_w \times M_h)$
Projection and Apply Conv2D
T
Transpose
Reshape
Matrix Multiply
$M_w \times M_h$
C_{in}
Conv2 + Pooling
...
Conv2 + Pooling
F_{out}
P_{Local}
$q_1 - p$
$q_2 - p$
$q_n - p$
$N \times 3$
F_{in}
f_1
f_2
f_s
$N \times C_{in}$
$M_w \times M_h \times C_{in}$
$1 \times 1 \times C_{out}$

FPConv

Method	Conv.	Samp.	ScanNet	S3DIS	S3DIS-6
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†	-
HPEIN [20]	V	FPS	61.8	61.9	67.8
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	-	-
FPConv (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**	-

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
FPConv (ours)	S	FPS	92.5

Table 2: Classification Accuracy on ModelNet40

CVPR2020

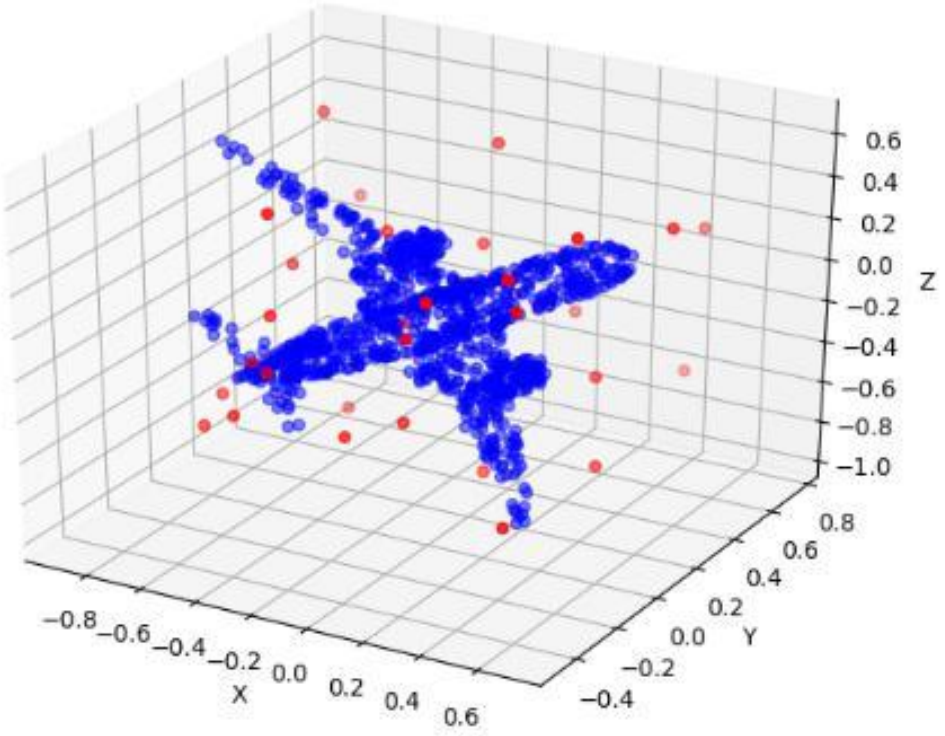
- RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds
- Grid-GCN for Fast and Scalable Point Cloud Learning
- FPConv: Learning Local Flattening for Point Convolution
- **PointASNL: Robust Point Clouds Processing using Nonlocal Neural Networks with Adaptive Sampling**
- PointAugment: an Auto-Augmentation Framework for Point Cloud Classification

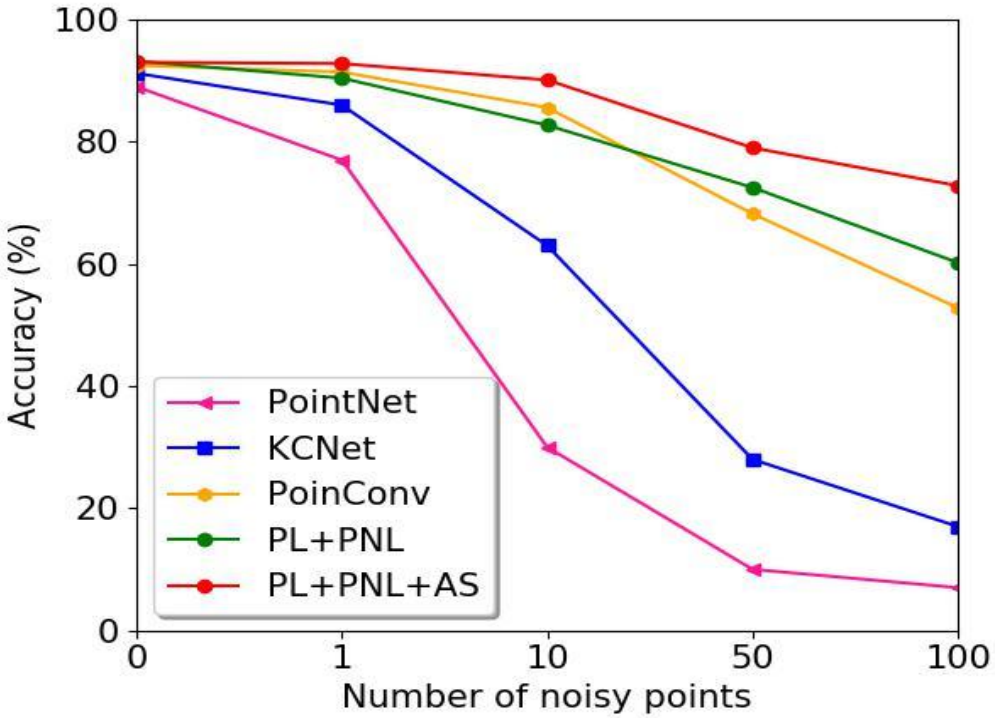
CVPR2020

PointASNL: Robust Point Clouds Processing using Nonlocal Neural Networks with Adaptive Sampling

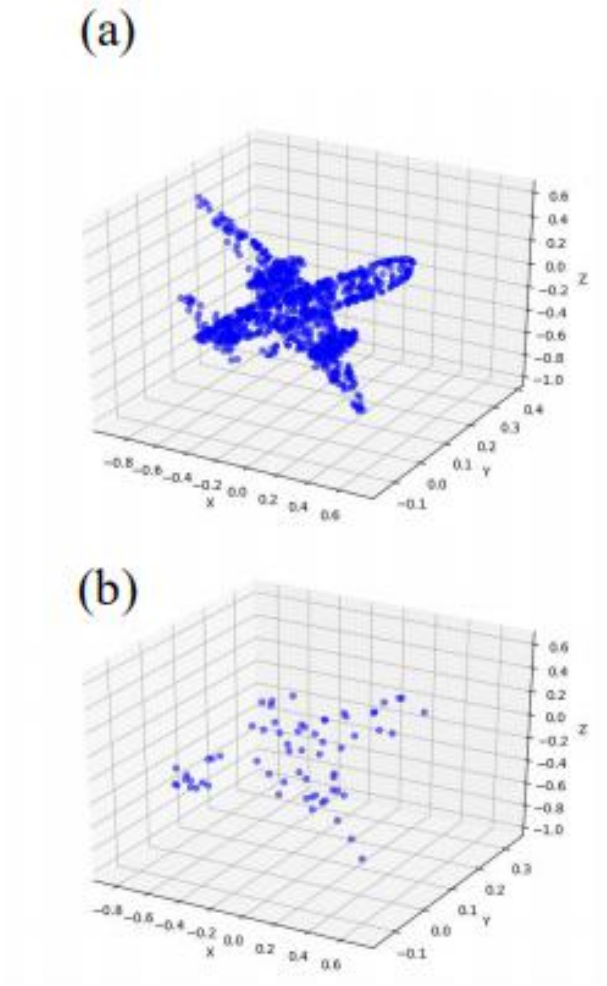
Xu Yan[1,2] Chaoda Zheng[2,3] Zhen Li[1,2,*] Sheng Wang[4] Shuguang Cui[1,2]
[1] The Chinese University of Hong Kong (Shenzhen), [2]Shenzhen Research Institute of Big Data
[3]South China University of Technology, [4]Tencent AI Lab
{xuyan1@link., lizhen@, shuguangcui@}cuhk.edu.cn

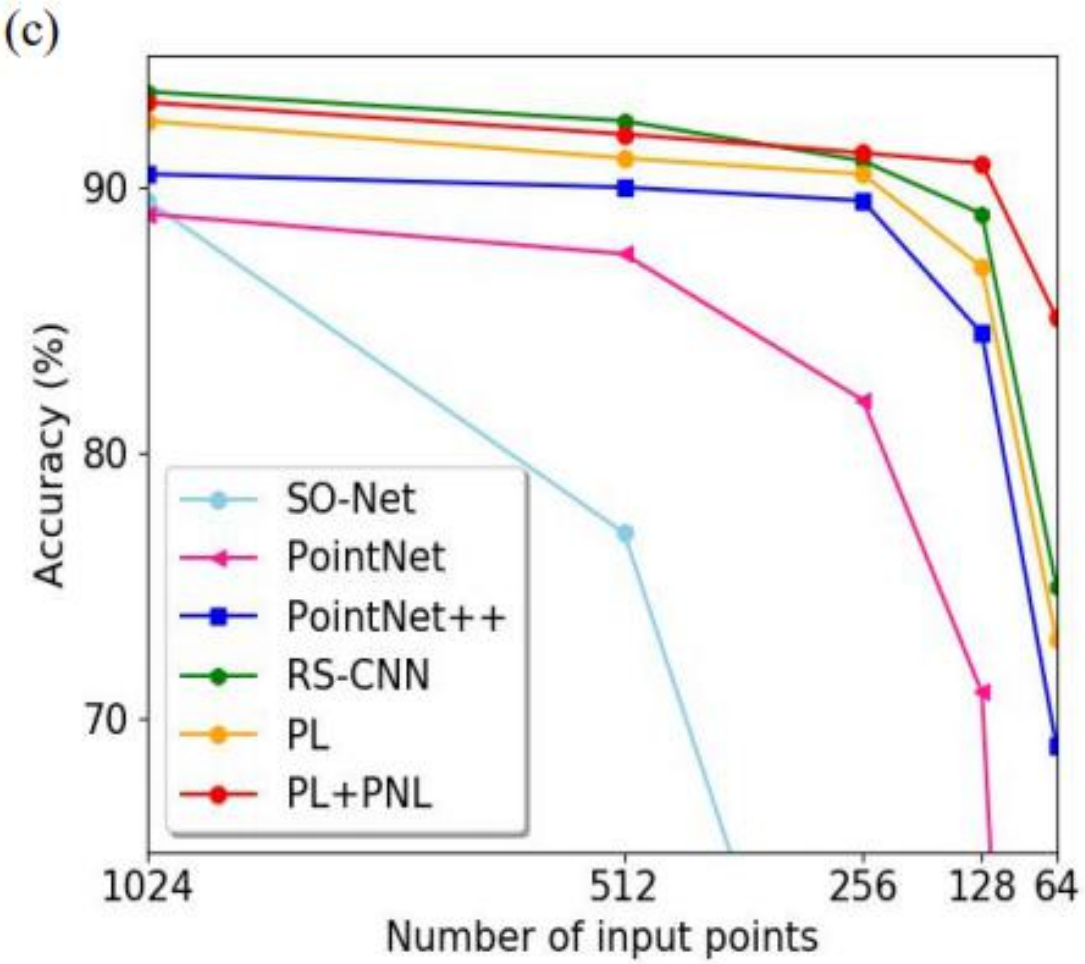
PointASNL


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
Z
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
Y
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
X

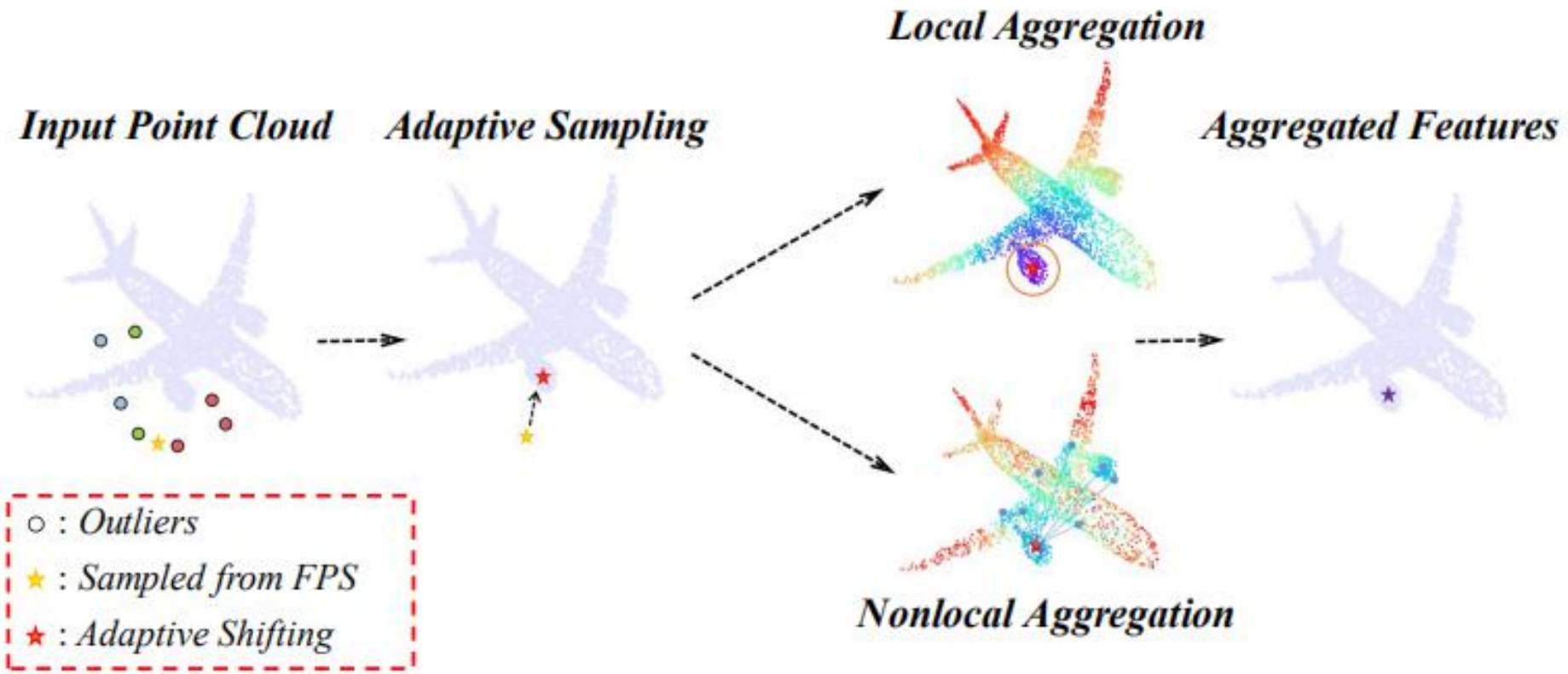

100
80
60
40
20
0
Accuracy (%)
PointNet
KCNet
PoinConv
PL+PNL
PL+PNL+AS
0
1
10
50
100
Number of noisy points

PointASNL


(a)
(b)

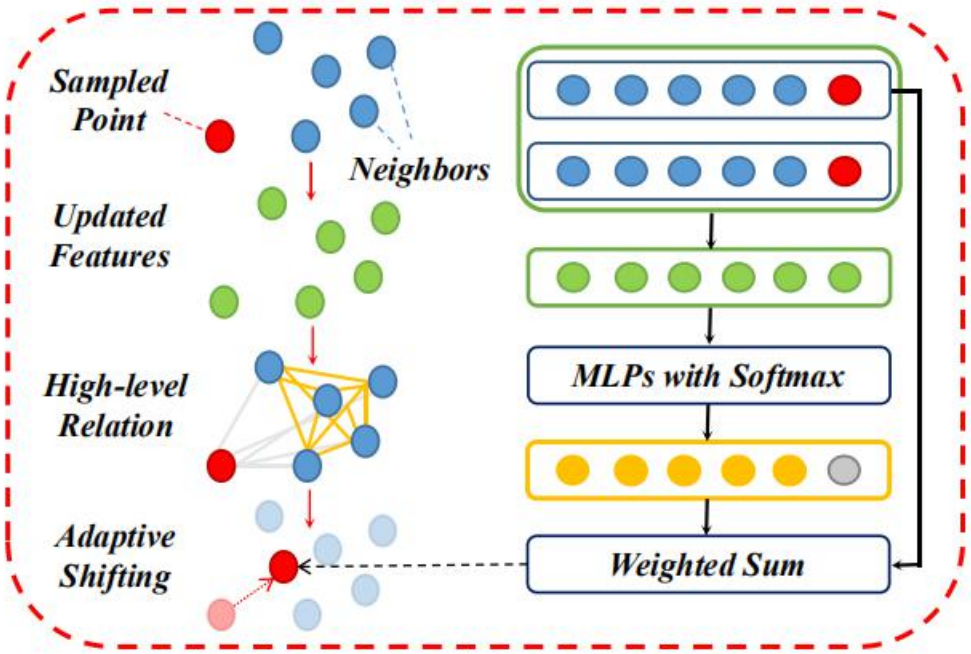

(c)
Accuracy (%)
90
80
70
1024
512
256
128
64
Number of input points
SO-Net
PointNet
PointNet++
RS-CNN
PL
PL+PNL

PointASNL

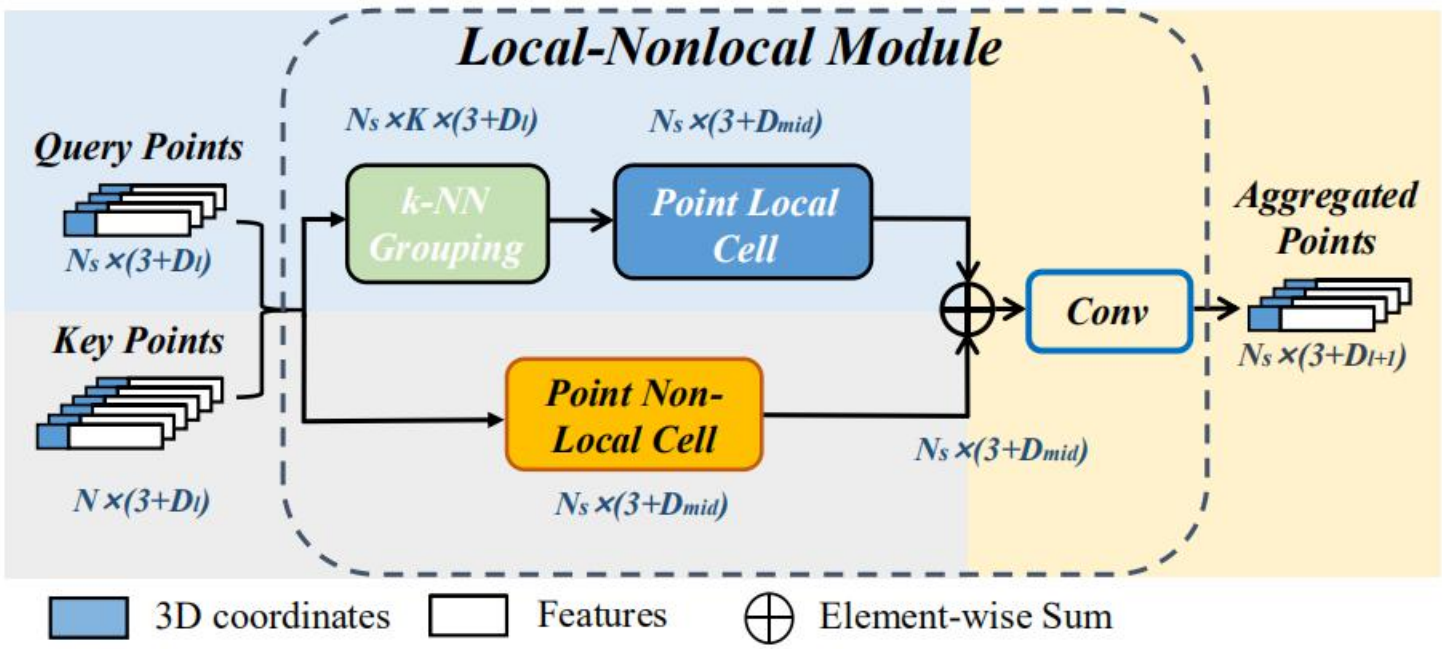


https://github.com/yanx27/PointASNL

PointASNL



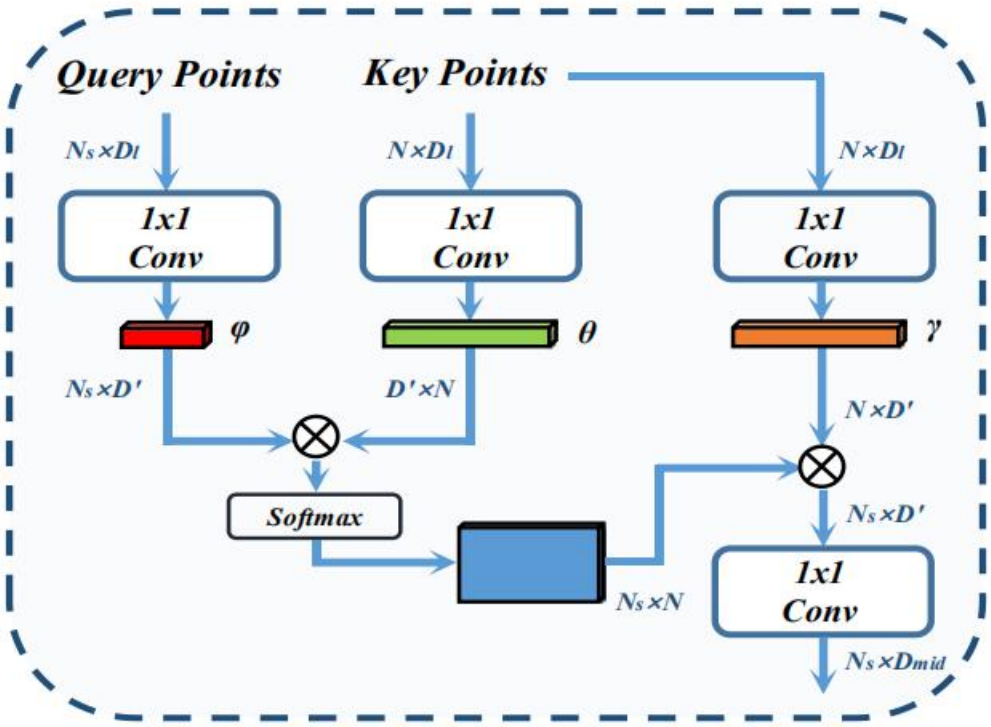
(a) Adaptive Sampling (AS) Module



(b) Local-NonLocal Module

Figure 2. Part (a) shows adaptive sampling (AS) module, which firstly updates features of grouping point by reasoning group relationship, then normalized weighs re-weight initial sampled points to achieve new sampled points. Part (b) illustrates the construction of local-nonlocal (L-NL) module, which consists of point local cell and point nonlocal cell. N_s stands for sampled point number, N stands for point number of entire point clouds, D_l, D_{mid}, and D_{l+1} stand for channel numbers.

PointASNL



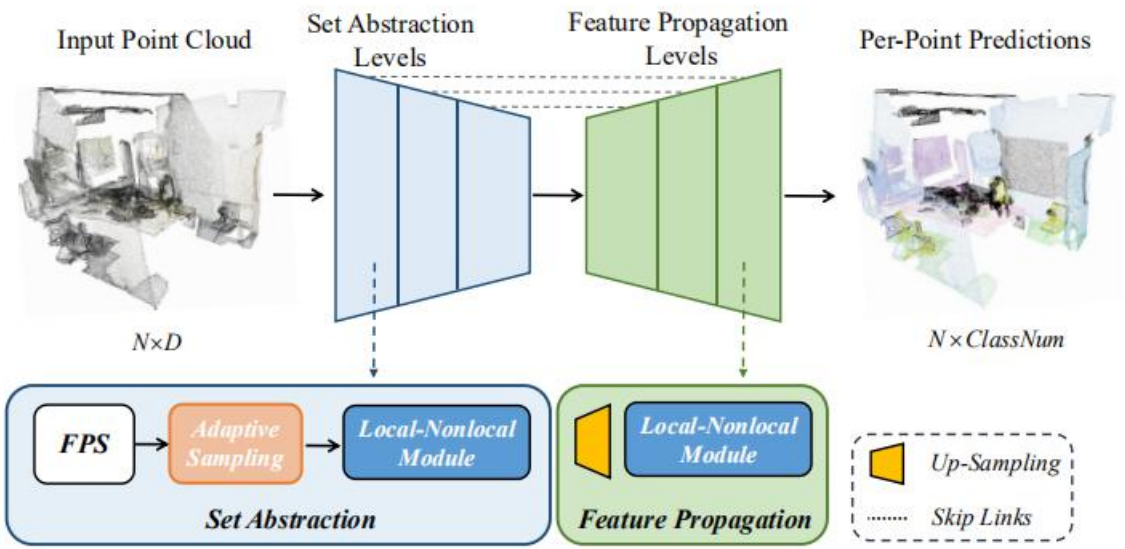


Figure 3. **Inner structure of point nonlocal (PNL) cell.** For the notations N_s, N, D_l, D_{mid} please refer to the caption of Fig. 2, D' is the intermediate channel numbers.

PointASNL

Table 2. Segmentation results on indoor S3DIS and ScanNet datasets in mean per-class IoU (mIoU,%).

Method	S3DIS	ScanNet
methods use unspecific number of points as input		
TangentConv [33]	52.8	40.9
SPGraph [17]	62.1	-
KPConv [34]	**70.6**	**68.4**
methods use fixed number of points as input		
PointNet++ [25]	53.4	33.9
DGCNN [42]	56.1	-
RSNet [13]	56.5	-
PAT [48]	64.3	-
PointCNN [20]	65.4	45.8
PointWeb [51]	66.7	-
PointConv [44]	-	55.6
HPEIN [14]	67.8	61.8
PointASNL	**68.7**	**63.0**

Table 3. Semantic Segmentation results on SemanticKITTI, where "pnt" stands for coordinates of point and SPGraph [17] uses all of points as input.

Method	input	mIoU(%)
PointNet [24]	50k pnt	14.6
SPGraph [17]	-	17.4
SPLATNet [31]	50k pnt	18.4
PointNet++ [25]	45k pnt	20.1
TangentConv [33]	120k pnt	40.9
PointASNL	8k pnt	**46.8**

Table 1. Overall accuracy on ModelNet10 (M10) and ModelNet40 (M40) datasets. "pnt" stands for coordinates of point and "nor" stands for normal vector.

Method	input	#points	*M10*	*M40*
O-CNN [39]	pnt, nor	-	-	90.6
SO-Net [19]	pnt, nor	2k	94.1	90.9
Kd-Net [15]	pnt	32k	94.0	91.8
PointNet++ [25]	pnt, nor	5k	-	91.9
SpiderCNN [47]	pnt, nor	5k	-	92.4
KPConv [34]	pnt	7k	-	92.9
SO-Net [19]	pnt, nor	5k	**95.7**	**93.4**
Pointwise CNN [12]	pnt	1k	-	86.1
ECC [30]	graphs	1k	90.8	87.4
PointNet [24]	pnt	1k	-	89.2
PAT [48]	pnt, nor	1k	-	91.7
Spec-GCN [36]	pnt	1k	-	91.8
PointGrid [18]	pnt	1k	-	92.0
PointCNN [20]	pnt	1k	-	92.2
DGCNN [42]	pnt	1k	-	92.2
PCNN [3]	pnt	1k	94.9	92.3
PointConv [44]	pnt, nor	1k	-	92.5
A-CNN [16]	pnt, nor	1k	95.5	92.6
Point2Sequence [21]	pnt	1k	95.3	92.6
RS-CNN [22]	pnt	1k	-	93.6
PointASNL	pnt	1k	95.7	92.9
PointASNL	pnt, nor	1k	**95.9**	**93.2**

PointASNL

Model	Ablation	*ModelNet40*	*ScanNet*
A	PNL only	90.1	45.7
B	PL only	92.0	56.1
C	PL+PNL	93.2	60.8
D	PL+PNL+AS	**93.2**	**63.5**
E	PointNet2 [24]	90.9	48.9
F	PointNet2+PNL	93.0	54.6
G	PointNet2+PNL+AS	92.8	55.4
H	DGCNN [42]	92.2	52.7
I	DGCNN+PNL	92.9	56.7
J	DGCNN+PNL+AS	93.1	58.3

Table 2. The results (mIoU, (%)) on *ScanNet* v2 validation set with other model setting.

Model	input	grid sample	deeper	mIoU
A	8192 pnt			63.5
B	8192 pnt	✓		64.5
C	10240 pnt	✓		64.8
D	10240 pnt	✓	✓	**66.4**

PointASNL

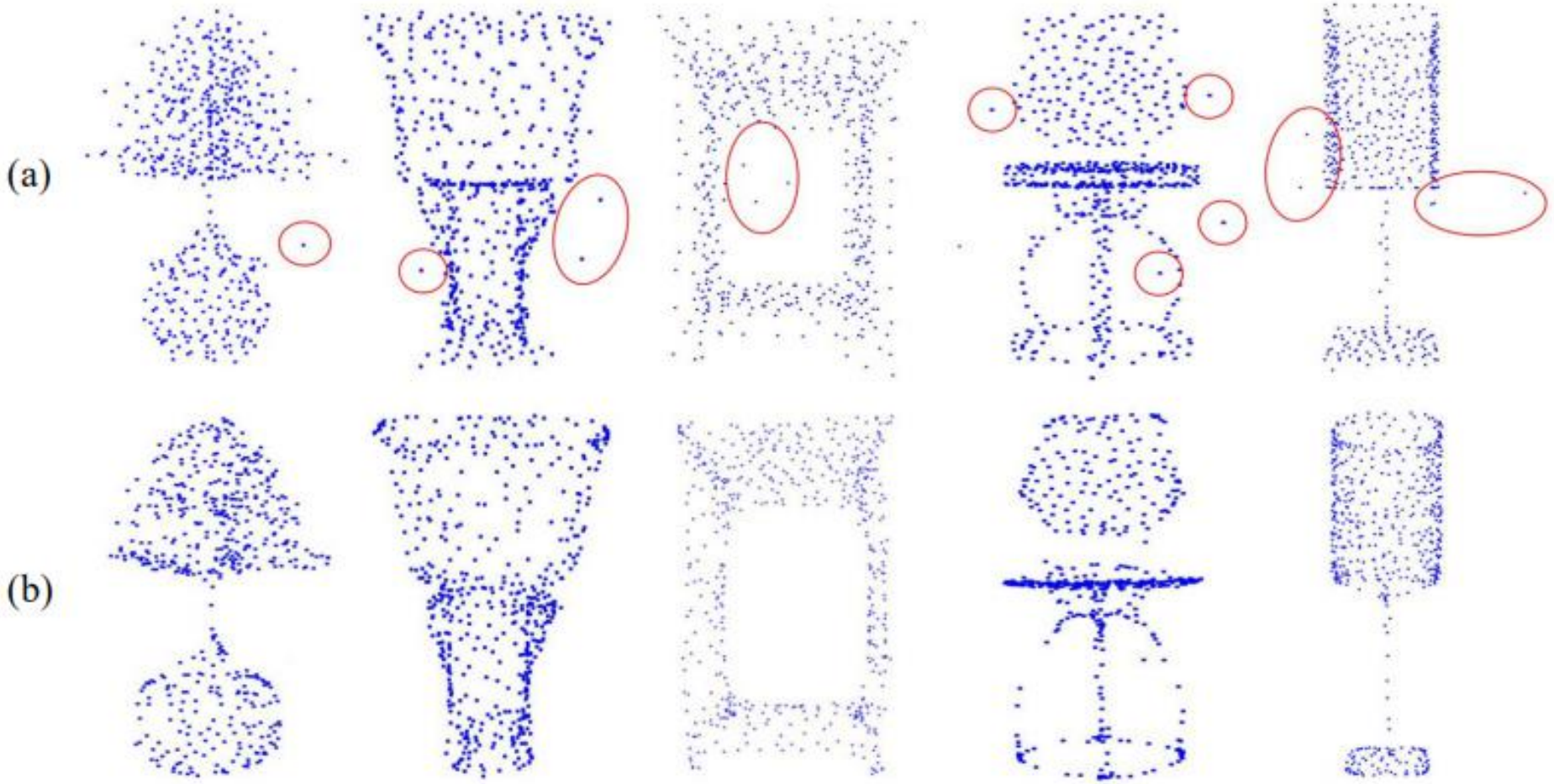


Figure 6. Visualized results of AS module. (a) Sampled points via farthest point sampling (FPS). (b) Sampled points ajusted by AS module.

CVPR2020

- RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds
- Grid-GCN for Fast and Scalable Point Cloud Learning
- FPConv: Learning Local Flattening for Point Convolution
- PointASNL: Robust Point Clouds Processing using Nonlocal Neural Networks with Adaptive Sampling
- **PointAugment: an Auto-Augmentation Framework for Point Cloud Classification**

CVPR2020

PointAugment: an Auto-Augmentation Framework for Point Cloud Classification

Ruihui Li[1] Xianzhi Li[1,2] Pheng-Ann Heng[1,2] Chi-Wing Fu[1,2]
[1]The Chinese University of Hong Kong
[2]Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
{lirh,xzli,pheng,cwfu}@cse.cuhk.edu.hk

PointAugment

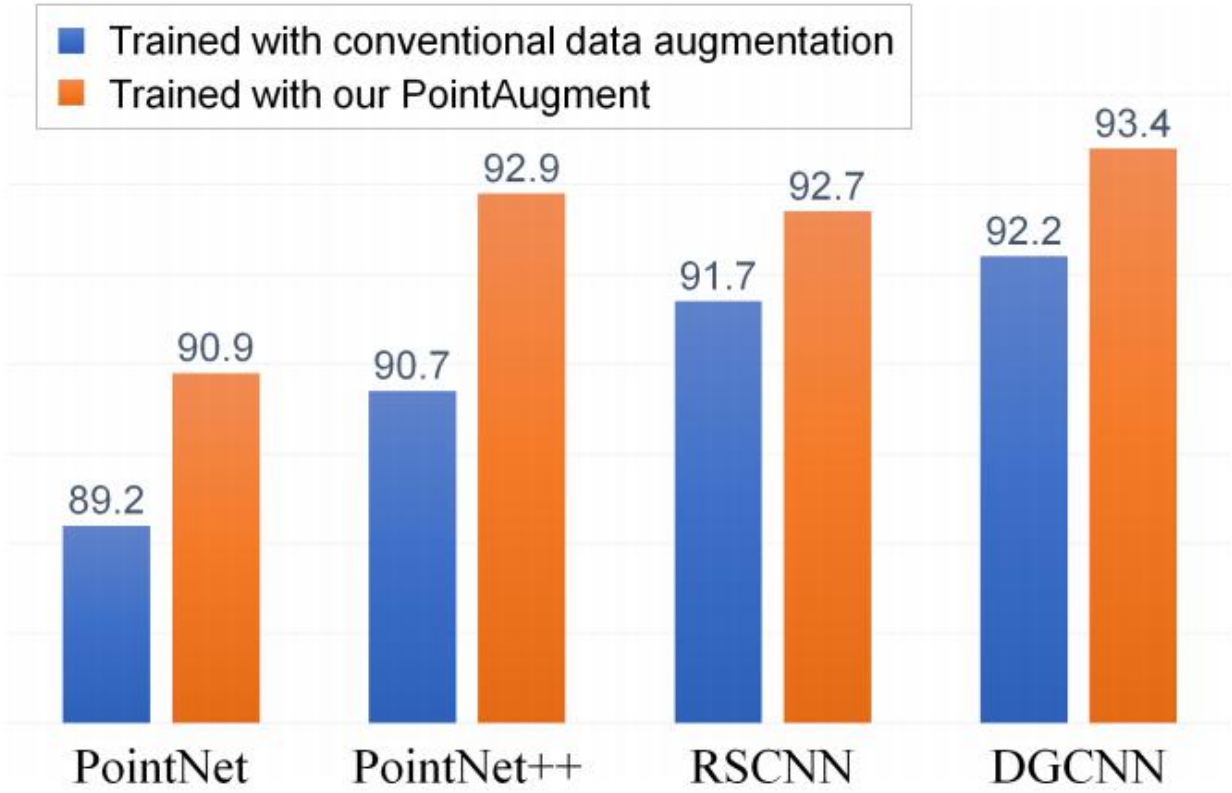


Figure 1: Classification accuracy (%) on ModelNet40 with or without training the networks with our PointAugment. We can see clear improvements on four representative networks. More comparison results are presented in Section 5.

PointAugment

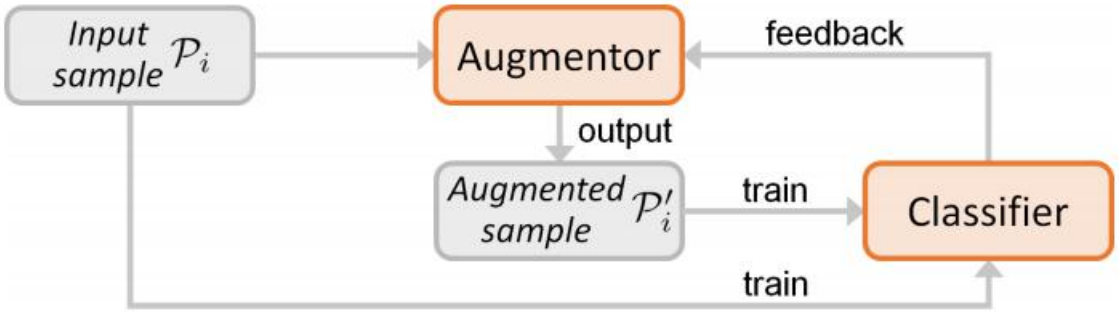


Figure 2: An overview of our PointAugment framework. We jointly optimize the augmentor and classifier in an end-to-end manner with an adversarial learning strategy.

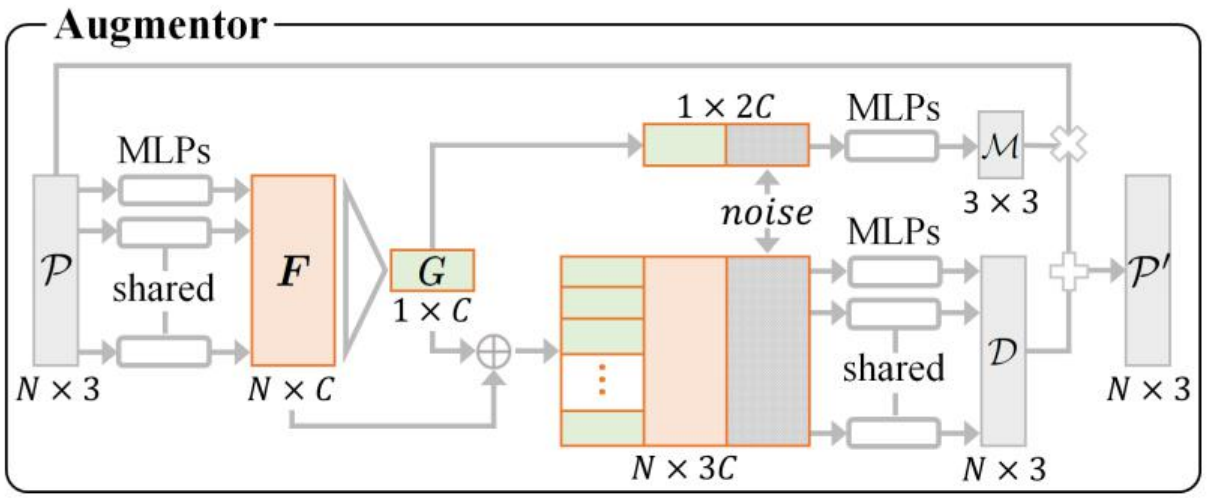


Figure 4: Our implementation of the augmentor.

PointAugment

To maximize the network learning, the augmented sample P0 generated by the augmentor should satisfy two requirements:

(i) **P0 should be more challenging than P;**

$$\mathcal{L}_{\mathcal{A}} = \exp[-(L(\mathcal{P}') - L(\mathcal{P}))], \quad L(\mathcal{P}) = -\sum_{c=1}^{K} \hat{y}_c \log(y_c))$$

Note also that, for P0 to be more challenging than P, we assume that L(P0)≥ L(P) and a larger L(P0) indicates a larger magnitude of augmentation

(ii) **P0 should not lose its shape distinctiveness,** meaning that it should describe a shape that is not too far from P.

$$\mathcal{L}_{\mathcal{A}} = |1.0 - \exp[L(\mathcal{P}') - \rho L(\mathcal{P})]|. \quad \xi = L(\mathcal{P}') - L(\mathcal{P}) \leq (\rho - 1)L(\mathcal{P}), ,$$

PointAugment

Lastly, to further ensure the augmented sample $\mathcal{P}'$ to be shape distinctive (for requirement (ii)), we add $L(\mathcal{P}')$, as a fidelity term, to Eq. (2) to construct the final loss $\mathcal{L}_{\mathcal{A}}$:

Augment

$$\mathcal{L}_{\mathcal{A}} = L(\mathcal{P}') + \lambda|1.0 - \exp(L(\mathcal{P}') - \rho L(\mathcal{P}))|, \quad (5)$$

where λ is a fixed hyper-parameter to control the relative importance of each term. A small λ encourages the augmentor to focus more on the classification with less augmentation on $\mathcal{P}$, and vice versa. In our implementation, we set $\lambda = 1$ to treat the two terms equally.

Classifier

The goal of the classifier $\mathcal{C}$ is to correctly predict both $\mathcal{P}$ and $\mathcal{P}'$. Additionally, $\mathcal{C}$ should also have the ability to learn stable per-shape global features, no matter given $\mathcal{P}$ or $\mathcal{P}'$ as input. We thus formulate the classifier loss $\mathcal{L}_{\mathcal{C}}$ as

$$\mathcal{L}_{\mathcal{C}} = L(\mathcal{P}') + L(\mathcal{P}) + \gamma\|\boldsymbol{F}_g - \boldsymbol{F}_{g'}\|_2, \quad (6)$$

PointAugment

Table 2: Comparing the overall shape classification accuracy (%) on MN40, MN10, and SR16, for various classifiers equipped with conventional DA (first four rows) and with our PA (last four rows); PA denotes PointAugment. We can observe improvements for all datasets and all classifiers.

Method	MN40	MN10	SR16
PointNet [20]	89.2	91.9	83.1
PointNet++ [21]	90.7	93.3	85.1
RSCNN [15]	91.7	94.2	86.6
DGCNN [33]	92.2	94.8	87.0
PointNet (+PA)	90.9 (1.7↑)	94.1 (2.2↑)	87.2 (4.1↑)
PointNet++ (+PA)	92.9 (2.2↑)	95.8 (2.5↑)	89.5 (4.4↑)
RSCNN (+PA)	92.7 (1.0↑)	96.0 (1.8↑)	90.1 (3.5↑)
DGCNN (+PA)	93.4 (1.2↑)	96.7 (1.9↑)	90.6 (3.6↑)

Table 5: Ablation study of PointAugment. $\mathcal{D}$: point-wise displacement, $\mathcal{M}$: shape-wise transformation, DP: dropout, and Mix: mixed training samples (see Section 4.4).

Model	#point	$\mathcal{D}$	$\mathcal{M}$	DP	Mix	Acc.	Inc.↑
A	1k					90.7	-
B	1k	✓				91.7	1.0
C	1k		✓			91.9	1.2
D	1k	✓	✓			92.5	1.8
E	1k	✓	✓	✓		92.8	2.1
F	1k	✓	✓	✓	✓	**92.9**	**2.2**
G	2k	✓	✓	✓	✓	**92.9**	**2.2**

Thank you