EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Max Planck Institute for Intelligent Systems

Million-Scale Point-Cloud Learning Beyond Grid-Worlds (Flex-Conv)

Fabian Groh[1], Patrick Wieschollek[1,2], Hendrik P.A. Lensch[1]

[1] University of Tübingen, [2] Max Planck Institute for Intelligent Systems, Tübingen

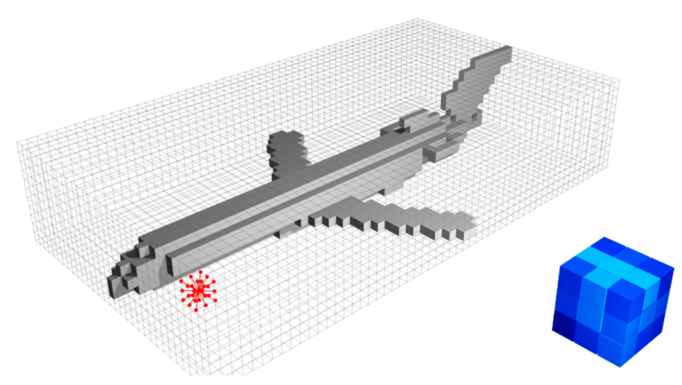
CyberValley

ROBUST VISION SFB 1233

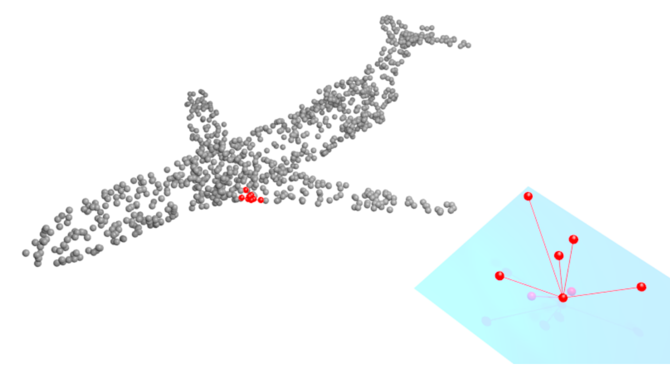
In a nutshell: Learning directly from million-scale unstructured point-cloud data using a novel deep convolutional network.

Why Using Point-Cloud Data?

Point-clouds are the natural representation of 3D data without any loss of information. Note, voxels do not scale to large resolution due to memory constraints.

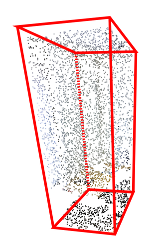


Voxel
(18 816 cells)

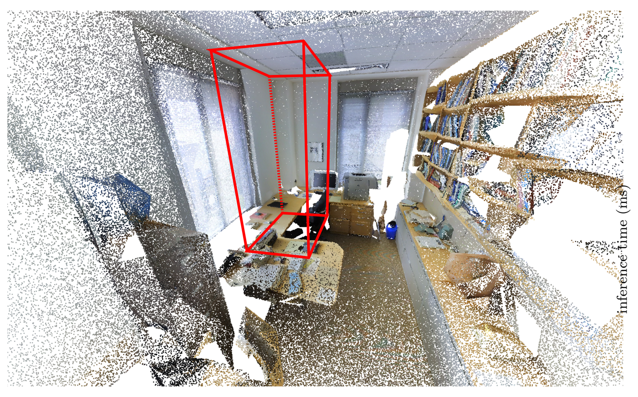


Point-Cloud
(1024 points)

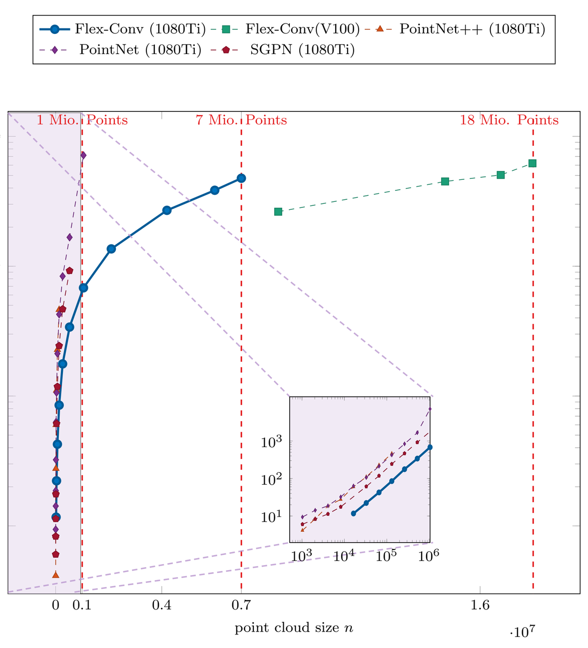
Why Is Large-Scale Important?



4096



818k

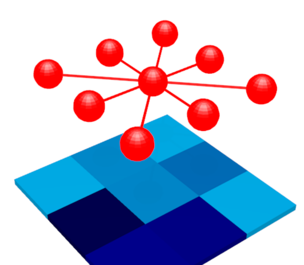


Previous methods [1, 2, 3] subsample small blocks (left) from large point-clouds, instead, ours can handle up to 7 Millions in a single forward pass (linear scaling).

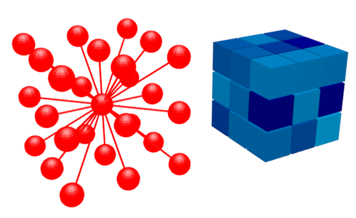
Flex-Convolution

$$(w \circledast f)[\vec{p}_\ell] = \sum_{c\in C} \boxed{\sum_{n\in N}} \boxed{w_c(\vec{p}_\ell - \vec{p}_n)} \; f(c, \vec{p}_n)$$

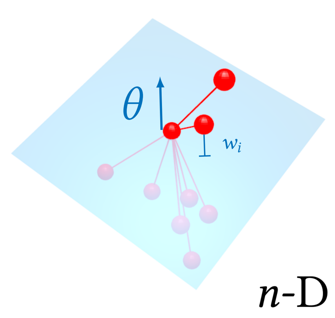
Grid Convolution		Flex Convolution
regular grid (implicit)		irregular grid (NN-search)
$w_c(\vec{p}) = W[p_x, p_y]$	$w_c(\vec{p}) = W[p_x, p_y, p_z]$	$w_c(\vec{p}) = \langle \theta_c, \vec{p} \rangle + \theta_{c,b}$



2D



3D



n-D

This formulation can be considered as a linear approximation of the lookup table, with the advantage of being defined everywhere. As a natural extension it can represent several image operations:

Ground-truth Filter k / Learned Parameters

Prewitt:

$$\begin{array}{c} \\ \theta_y \end{array}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}_{\theta_x} \qquad \begin{aligned} \theta_x &= 0.0000 \\ \theta_y &= 1.0058 \\ \theta_b &= -0.0004 \end{aligned}$$

Blur:

$$\frac{1}{N}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \begin{aligned} \theta_x &= 0.0002 \\ \theta_y &= -0.0022 \\ \theta_b &= 0.1108 \end{aligned}$$

Usage: Our implementation can be used as drop-in replacement:

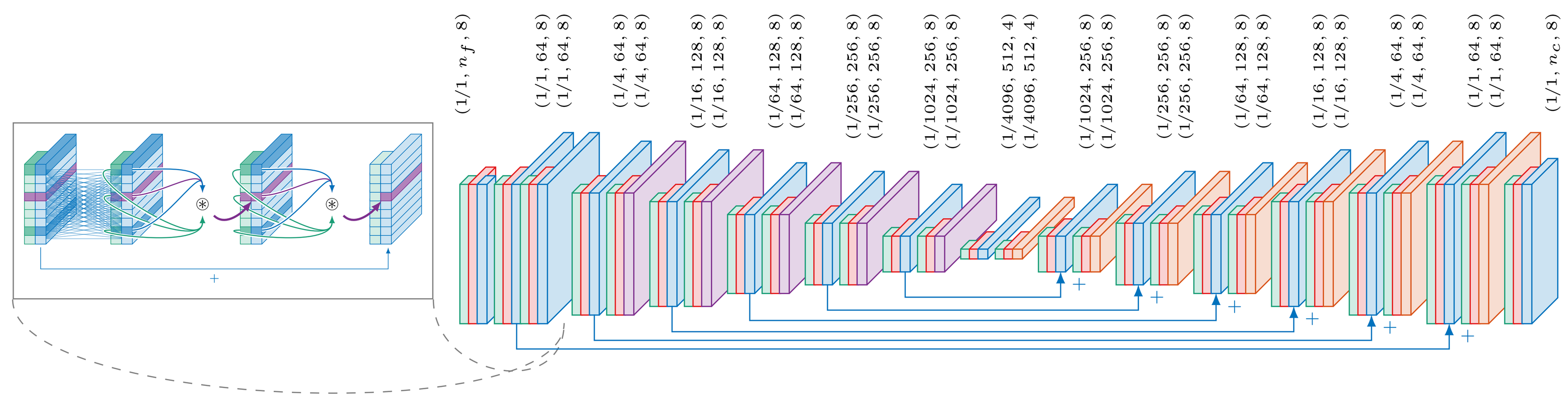
```
# grid-base convolution
net = np.random.rand(B, h, w, Din)
net = tf.layers.convolution_2d(net, Dout, 3, activation=tf.nn.relu)
net = tf.layers.max_pooling2d(net, 2, 2)

# flex-convolution
net, pos = np.random.rand(B, Din, N), np.random.randn(B, Dp, N)
knn = knn_bruteforce(pos, K=8)
net = flex_convolution(net, pos, knn, Dout, activation=tf.nn.relu)
net = flex_pooling(net, knn)
```

[1] Qi etal. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.*, (CVPR 2017).
[2] Qi at al. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.*, (NIPS 2017).
[3] Wang et al. *Sgpn: Similarity group proposal network for 3d point cloud instance segmentation.*, (CVPR 2018).
[4] Armeni etal. *3D Semantic Parsing of Large-Scale Indoor Spaces.*, (CVPR 2016).
[5] Su etal. *SPLATNet: Sparse Lattice Networks for Point Cloud Processing.*, (CVPR 2018).
[6] Klokov etal. *Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models.*, (ICCV 2017).

Code+Video

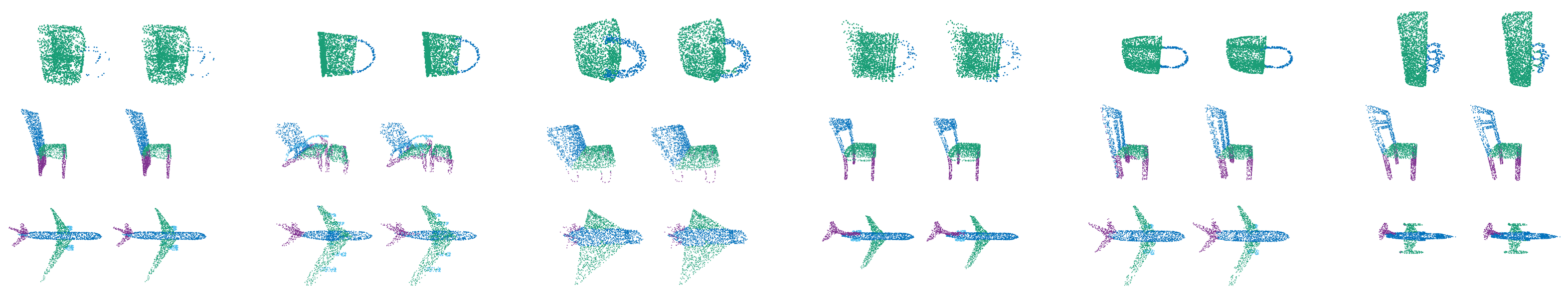
	Performance		Memory	
Method	Forward	Backward	Forward	Backward
Flex-Conv (pure tensorflow)	1 829ms	2 738ms	34 015.2MB	63 270.8MB
Flex-Conv (hand-crafted CUDA)	24ms	265ms	8.4MB	8.7MB
Grid-Conv (cuDNN)	16ms	1.5ms	1574.1MB	153.4MB



Importance of hand-crafted CUDA implementations is shown in a performance test for 32k points. The low memory footprint allows for deep networks as shown in our semantic segmentation architecture.

Results

Synthetic: ShapeNet part segmentation pairs (left: ground-truth & right: prediction) and quantitative results per category and mIoU (%) for different methods and inference speed (on a Nvidia GeForce GTX 1080 Ti).



	Airpl.	Bag	Cap	Car	Chair	Earph.	Guitar	Knife	Lamp	Laptop	Motorb.	Mug	Pistol	Rocket	Skateb.	Table	**mIoU**	**shapes/sec**
Kd-Network [6]	80.1	74.6	74.3	70.3	88.6	73.5	90.2	**87.2**	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3	77.4	n.a.
PointNet [1]	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6	80.4	n.a.
PointNet++ [2]	82.4	79.0	87.7	77.3	**90.8**	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6	81.9	2.7
SPLATNet3D [5]	81.9	83.9	88.6	**79.5**	90.1	73.5	91.3	84.7	**84.5**	96.3	69.7	95.0	81.7	59.2	70.4	81.3	82.0	9.4
SGPN [3]	80.4	78.6	78.8	71.5	88.6	**78.0**	90.9	83.0	78.8	95.8	**77.8**	93.8	**87.4**	60.1	**92.3**	**89.4**	82.8	n.a.
Ours	**83.6**	**91.2**	**96.7**	**79.5**	84.7	71.7	**92.0**	86.5	83.2	**96.6**	71.7	**95.7**	86.1	**74.8**	81.4	84.5	**85.0**	**489.3**

Real-World: Class specific average precision (AP) on the 2D-3D-S dataset (‡uses additional input features and * uses non-trivial post-processing)

	Table	Chair	Sofa	Bookc.	Board	Ceiling	Floor	Wall	Beam	Col.	Wind.	Door	**mAP**
Armenin *et al.* [4]*	46.02	16.15	6.78	54.71	3.91	71.61	88.70	72.86	66.67	**91.77**	25.92	54.11	49.93
Armenin *et al.* [4]‡	39.87	11.43	4.91	**57.76**	3.73	50.74	80.48	65.59	68.53	85.08	21.17	45.39	44.19
PointNet [1]*	46.67	33.80	4.76	n.a.	11.72	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
SGPN [3]*	46.90	40.77	6.38	47.61	11.05	79.44	66.29	**88.77**	**77.98**	60.71	**66.62**	**56.75**	54.35
Ours	**67.02**	**52.75**	**16.61**	39.26	**47.68**	**87.33**	**96.10**	65.52	56.83	55.10	57.66	36.76	**56.55**