CSC2457 3D & Geometric Deep Learning

KPConv: Flexible and Deformable Convolution for Point Clouds

Hugues Thomas[1] Charles R. Qi[2] Jean-Emmanuel Deschaud[1] Beatriz Marcotegui[1]
François Goulette[1] Leonidas J. Guibas[2,3]

[1]Mines ParisTech [2]Facebook AI Research [3]Stanford University

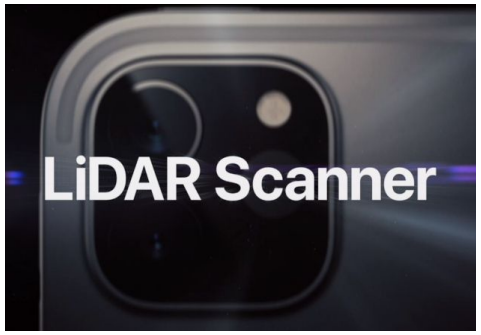
Date: 26 Jan. 2021

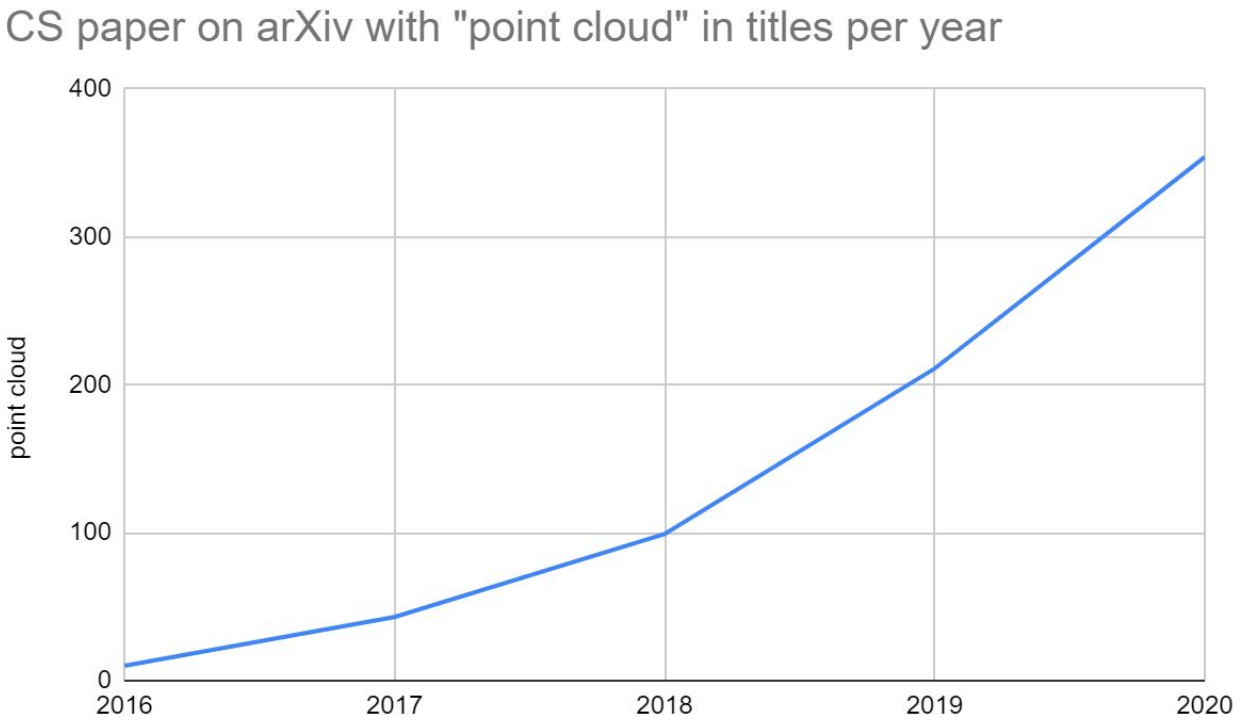
Presenter: Bin Yang

Instructor: Animesh Garg

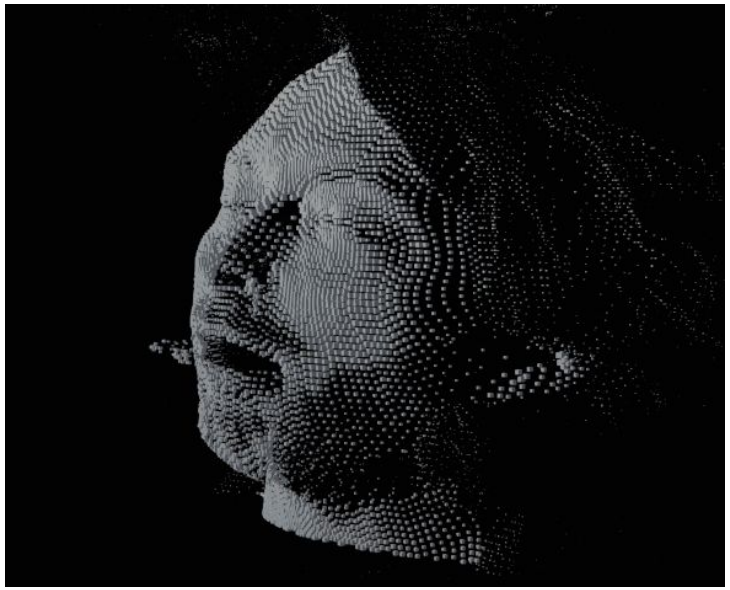
UNIVERSITY OF TORONTO

Point cloud is becoming the new trend!



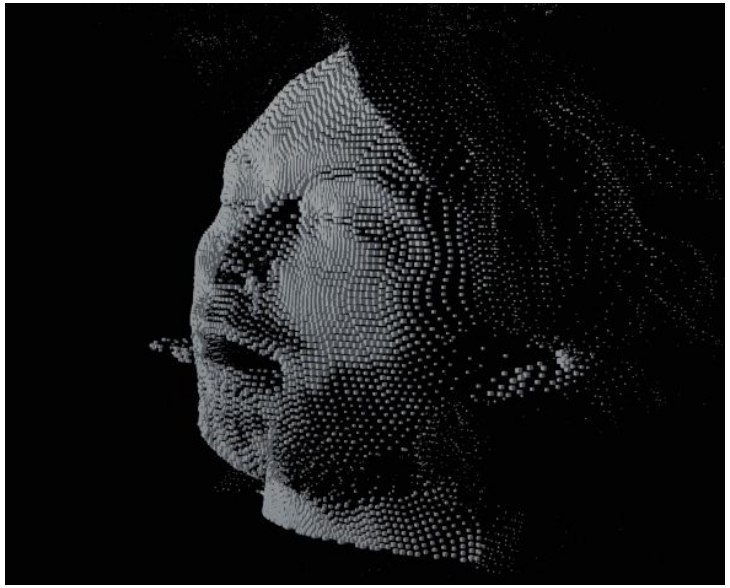


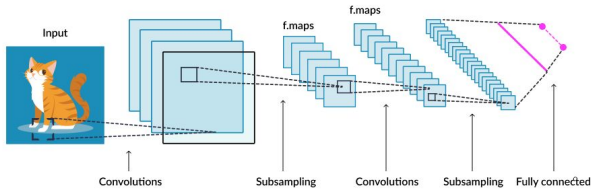
Processing point clouds is challenging!



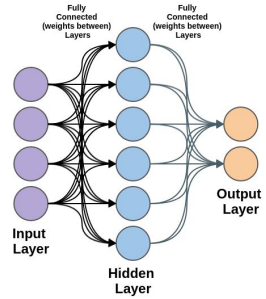
- *Non-grid* structured data
- *Unordered*
- In *continuous* space
- *Sparse*
- *Varying* densities

Processing point clouds is challenging!

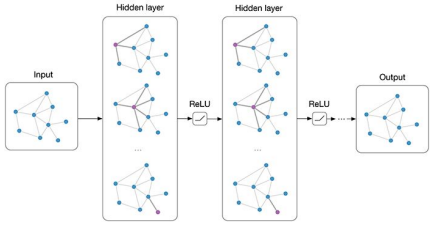




CNN



MLP



GCN

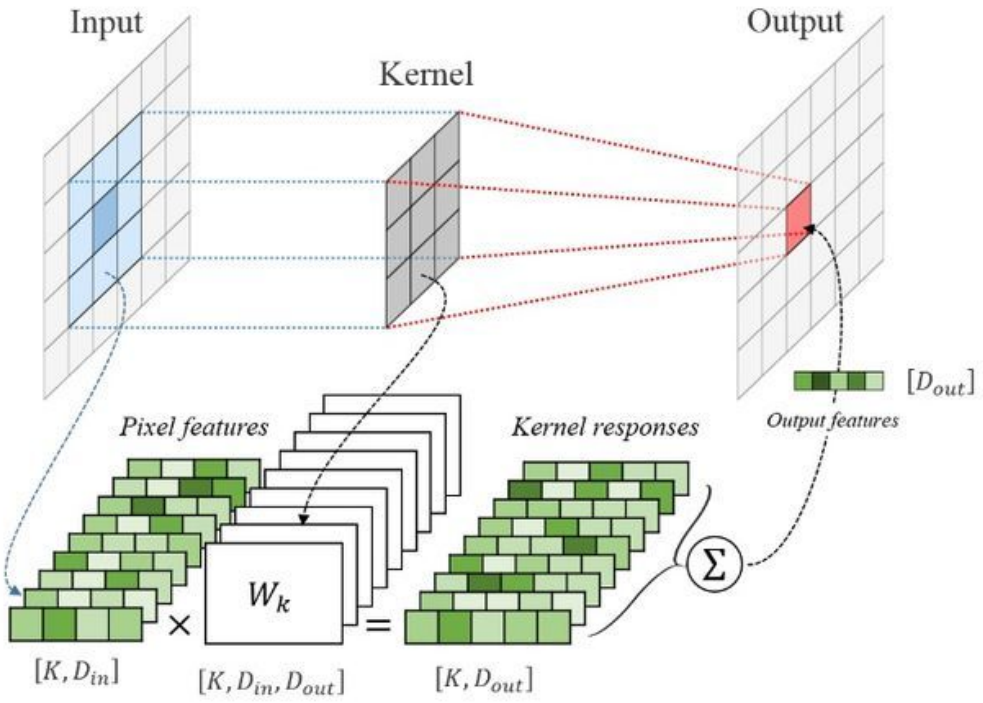
- Limited receptive field with computation overhead
- Lacks the capability to capture local shape
- Limited flexibility

Contributions

- Prior works
  - **Projection networks:** intermediate structure with limited receptive field
  - **Graph convolution:** feature aggregation over edges ignores local shape
  - **Pointwise MLP:** hard to capture local shape in an efficient manner
  - **Point convolution:** limited flexibility or representative power
- This paper proposes a new type of point convolution that
  - Learns *kernel* function to compute *point-wise* filters
  - Increases the discriminative power with *deformable* kernels
- This paper shows
  - State-of-the-art performance in a large variety of tasks (object classification, segmentation of small or large scenes)
  - Fast training and inference time for deep architectures

General Background

2D Convolution



Kernel Trick

Kernel function $\boldsymbol{x} \rightarrow \phi(\boldsymbol{x})$.

Kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$

Map onto high-dimensional space (non-linear combinations)

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \qquad \boldsymbol{x} \in \mathbb{R}^d$$

$$\Downarrow \phi$$

$$\boldsymbol{x}' = \begin{bmatrix} x_1 & x_2 & x_1 x_2 & x_1^2 & x_1 x_2^3 & \dots \end{bmatrix} \qquad \boldsymbol{x} \in \mathbb{R}^k \quad (k >> d)$$

Input Data Notation

We always consider a point cloud as two elements:

- The points $\mathcal{P} \in \mathbb{R}^{N \times 3}$
- The features $\mathcal{F} \in \mathbb{R}^{N \times D}$

Kernel Point Convolution

Point convolution of ***F*** by a kernel ***g*** at a point ***x*** is defined as follows:

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i$$

where neighborhood $\mathcal{N}_x = \{ x_i \in \mathcal{P} \mid \|x_i - x\| \leqslant r \}$

Kernel Point Convolution

Point convolution of ***F*** by a kernel ***g*** at a point ***x*** is defined as follows:

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i$$

We want ***g*** to apply different weights to different areas inside the neighborhood

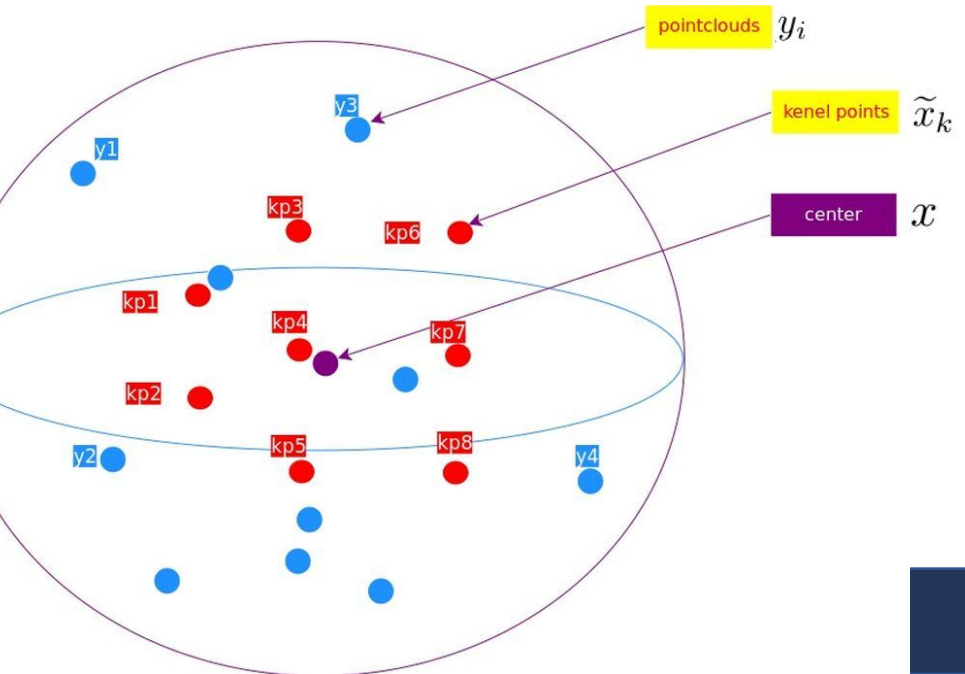
where neighborhood $\mathcal{N}_x = \{ x_i \in \mathcal{P} \mid \|x_i - x\| \leqslant r \}$

Kernel Point Convolution

Point convolution of ***F*** by a kernel ***g*** at a point ***x*** is defined as follows:

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i$$

We want ***g*** to apply different weights to different areas inside the neighborhood



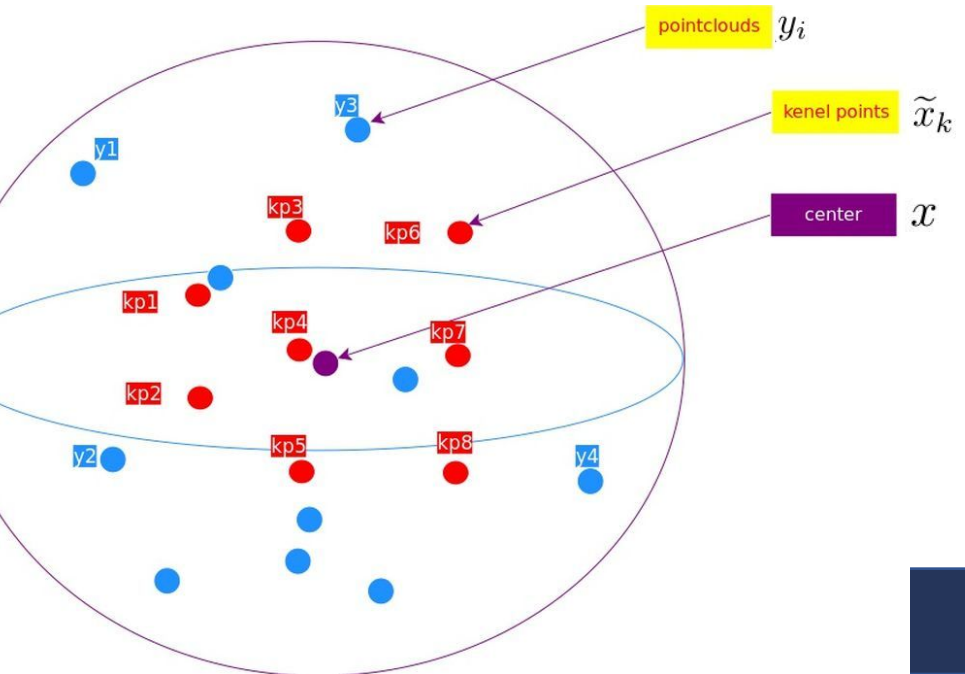
$$g(y_i) = \sum_{k<K} h\left(y_i, \widetilde{x}_k\right) W_k$$

Kernel Point Convolution

Point convolution of ***F*** by a kernel ***g*** at a point ***x*** is defined as follows:

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i$$

$$g(y_i) = \sum_{k<K} h\left(y_i, \widetilde{x}_k\right) W_k$$



Correlation should be higher when closer

Kernel Point Convolution

Point convolution of ***F*** by a kernel ***g*** at a point ***x*** is defined as follows:

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i$$

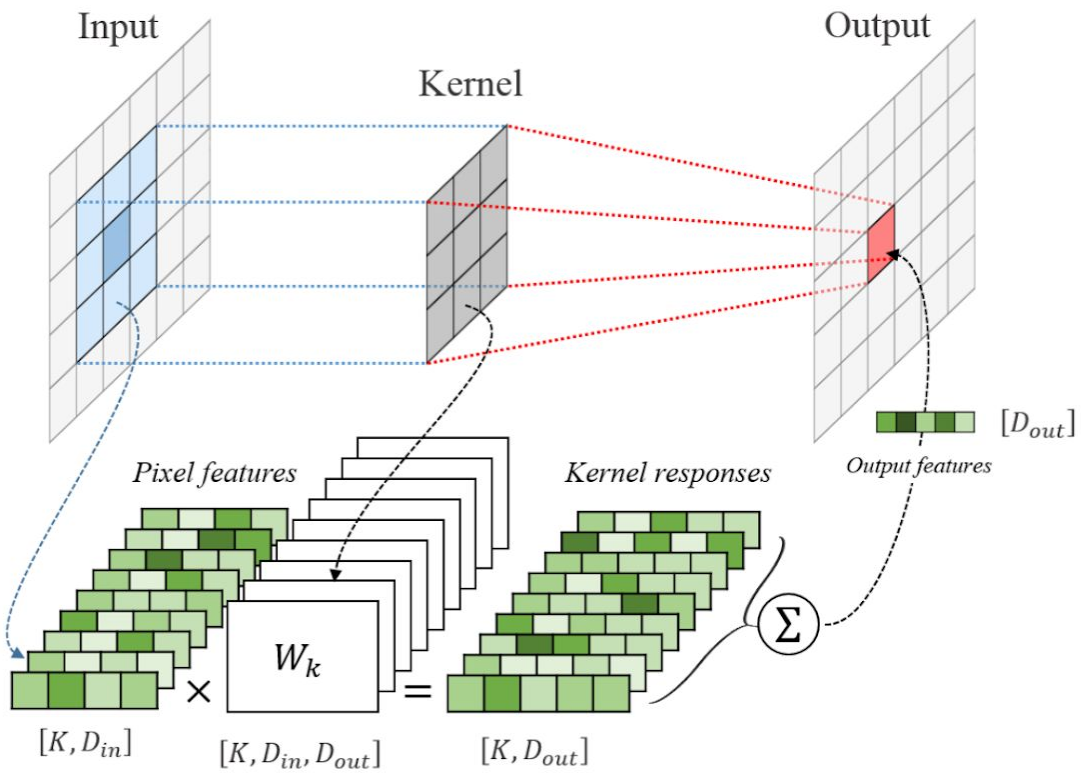
$$g(y_i) = \sum_{k<K} h\left(y_i, \widetilde{x}_k\right) W_k$$

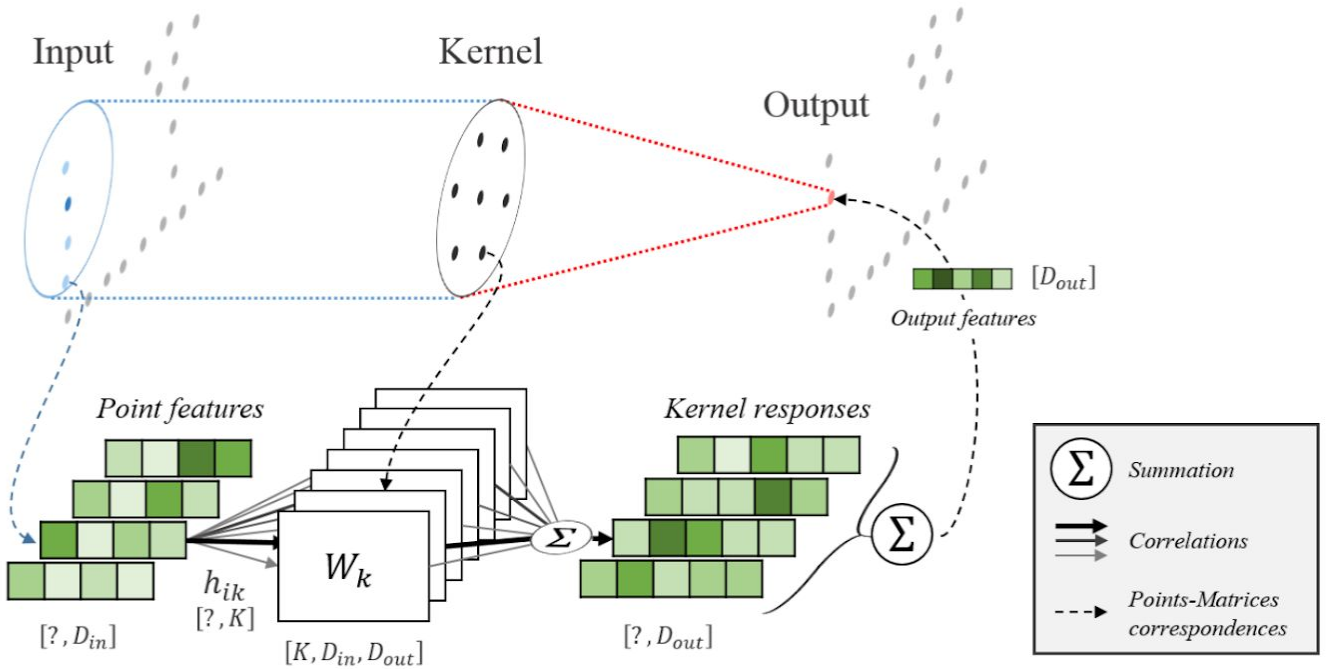
Correlation should be higher when closer

$$h\left(y_i, \widetilde{x}_k\right) = \max\left(0, 1 - \frac{\|y_i - \widetilde{x}_k\|}{\sigma}\right)$$

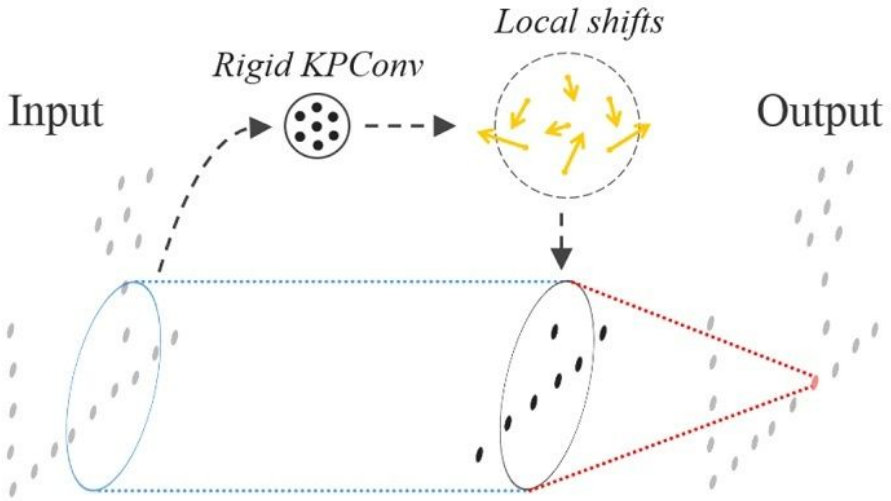
where σ is the influence distance chosen according to the input density

2D Convolution vs. Kernel Point Convolution





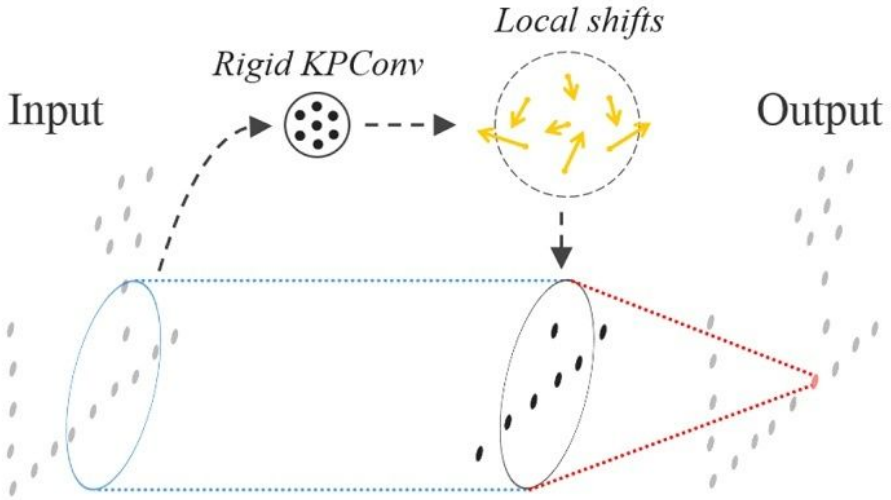
Deformable Kernel Point Convolution



$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g_{deform}(x - x_i, \Delta(x)) f_i$$

$$g_{deform}(y_i, \Delta(x)) = \sum_{k<K} h\left(y_i, \widetilde{x}_k + \Delta_k(x)\right) W_k$$

Deformable Kernel Point Convolution



$$\mathcal{L}_{reg} = \sum_{x} \mathcal{L}_{fit}(x) + \mathcal{L}_{rep}(x)$$

$$\mathcal{L}_{fit}(x) = \sum_{k<K} \min_{y_i} \left(\frac{\|y_i - (\widetilde{x}_k + \Delta_k(x))\|}{\sigma} \right)^2$$

$$\mathcal{L}_{rep}(x) = \sum_{k<K} \sum_{l \neq k} h\left(\widetilde{x}_k + \Delta_k(x), \widetilde{x}_l + \Delta_l(x)\right)^2$$

Experimental Results

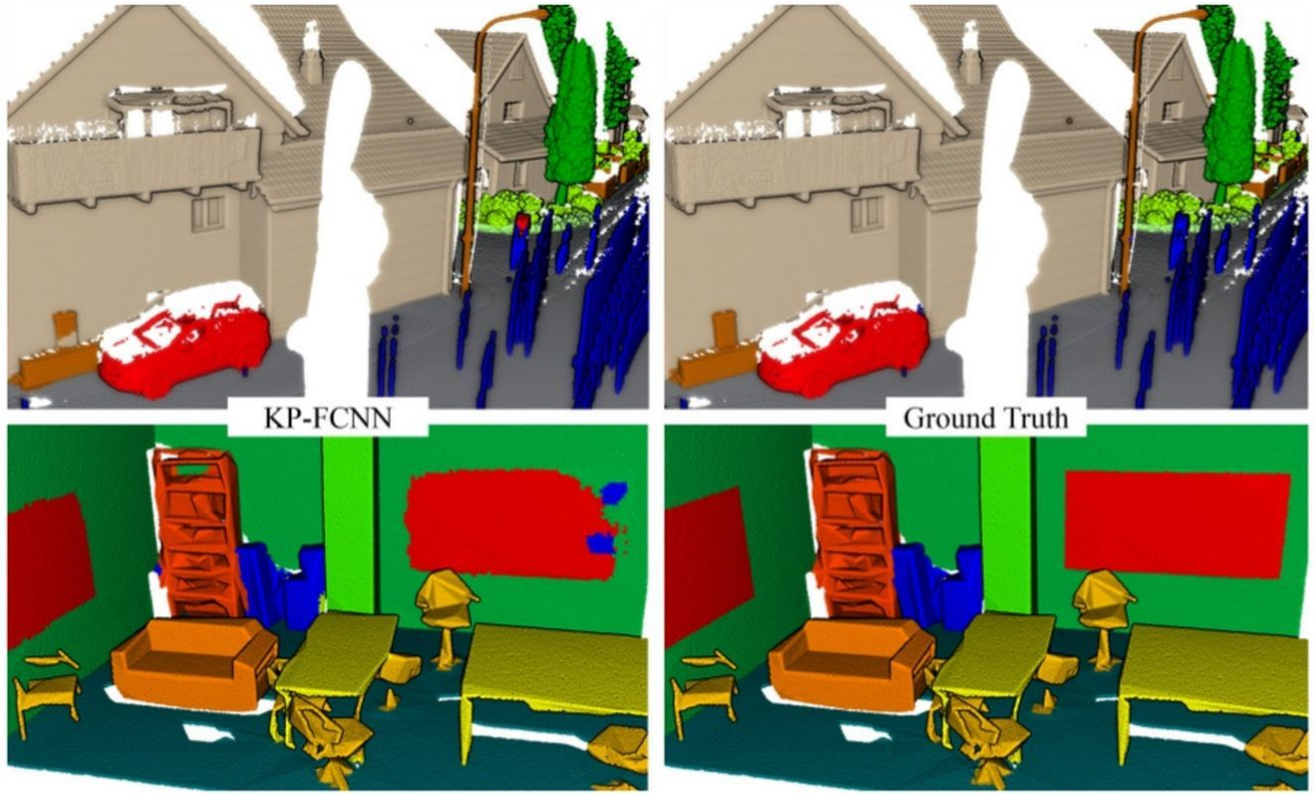
	ModelNet40	ShapeNetPart	
Methods	OA	mcIoU	mIoU
SPLATNet [34]	-	83.7	85.4
SGPN [42]	-	82.8	85.8
3DmFV-Net [9]	91.6	81.0	84.3
SynSpecCNN [48]	-	82.0	84.7
RSNet [15]	-	81.4	84.9
SpecGCN [40]	91.5	-	85.4
PointNet++ [27]	90.7	81.9	85.1
SO-Net [19]	90.9	81.0	84.9
PCNN by Ext [2]	92.3	81.8	85.1
SpiderCNN [45]	90.5	82.4	85.3
MCConv [13]	90.9	-	85.9
FlexConv [10]	90.2	84.7	85.0
PointCNN [20]	92.2	84.6	86.1
DGCNN [43]	92.2	85.0	84.7
SubSparseCNN [9]	-	83.3	86.0
KPConv *rigid*	**92.9**	85.0	86.2
KPConv *deform*	92.7	**85.1**	**86.4**

Table 1. 3D Shape Classification and Segmentation results. For

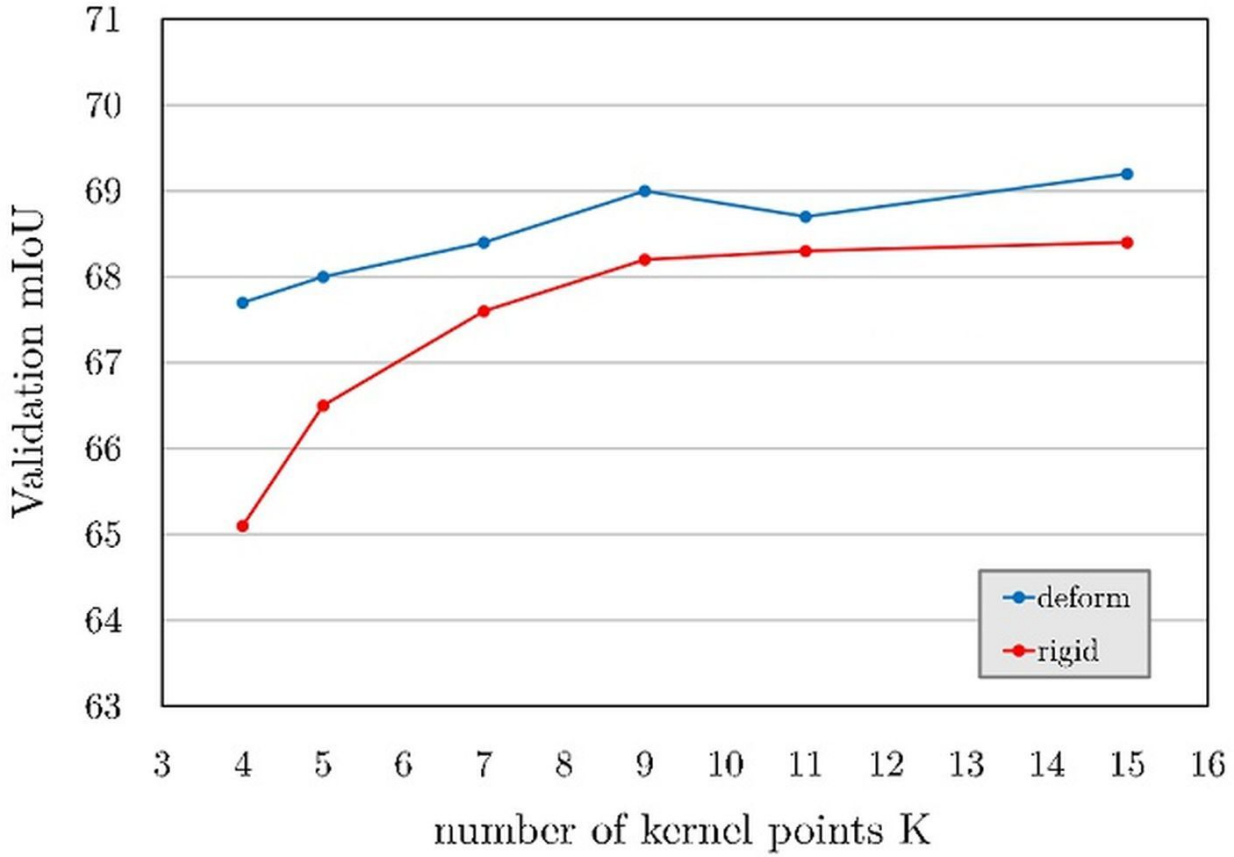
Methods	Scannet	Sem3D	S3DIS	PL3D
Pointnet [26]	-	-	41.1	-
Pointnet++ [27]	33.9	-	-	-
SnapNet [4]	-	59.1	-	-
SPLATNet [34]	39.3	-	-	-
SegCloud [37]	-	61.3	48.9	-
RF_MSSF [38]	-	62.7	49.8	56.3
Eff3DConv [50]	-	-	51.8	-
TangentConv [36]	43.8	-	52.6	-
MSDVN [30]	-	65.3	54.7	66.9
RSNet [15]	-	-	56.5	-
FCPN [28]	44.7	-	-	-
PointCNN [20]	45.8	-	57.3	-
PCNN [2]	49.8	-	-	-
SPGraph [17]	-	73.2	58.0	-
ParamConv [41]	-	-	58.3	-
SubSparseCNN [9]	**72.5**	-	-	-
KPConv *rigid*	68.6	**74.6**	65.4	72.3
KPConv *deform*	68.4	73.1	**67.1**	**75.9**

Table 2. 3D scene segmentation scores (mIoU). Scannet, Se-

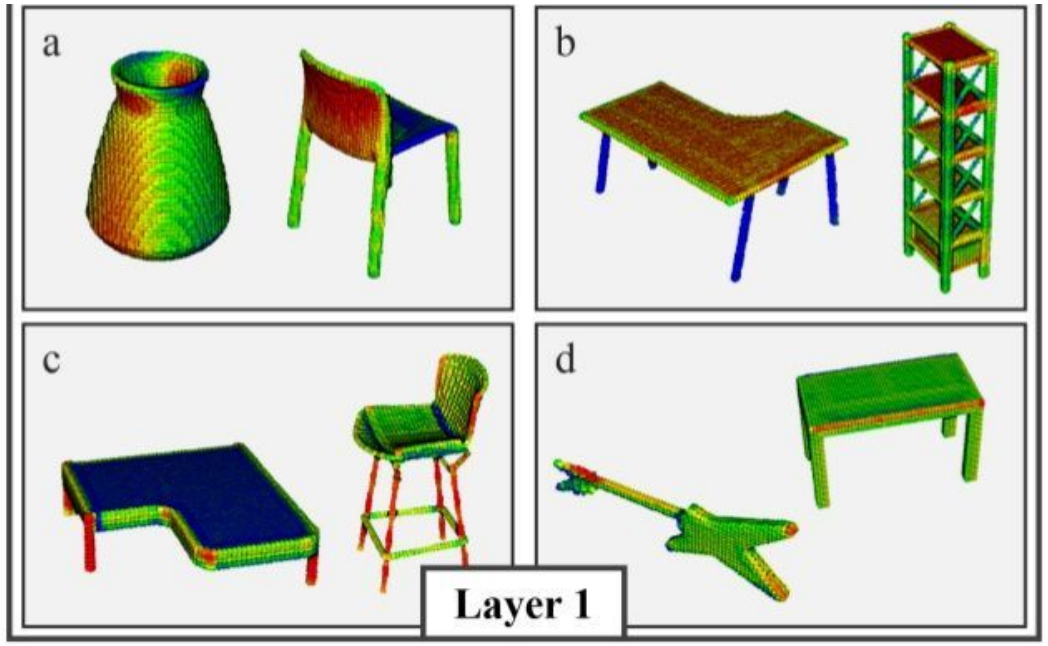
Qualitative Results

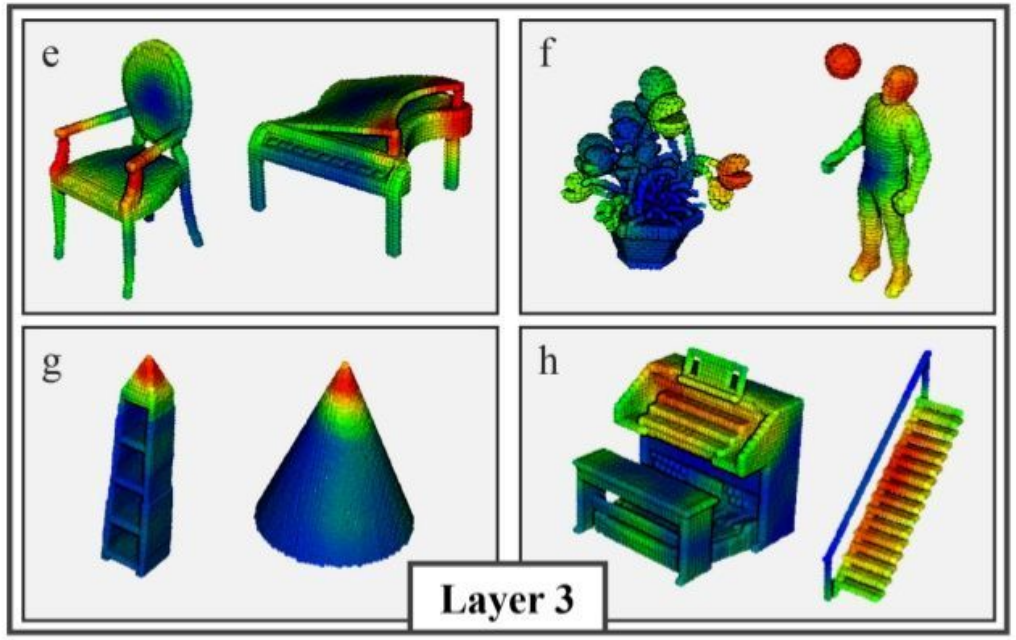


Ablation Study

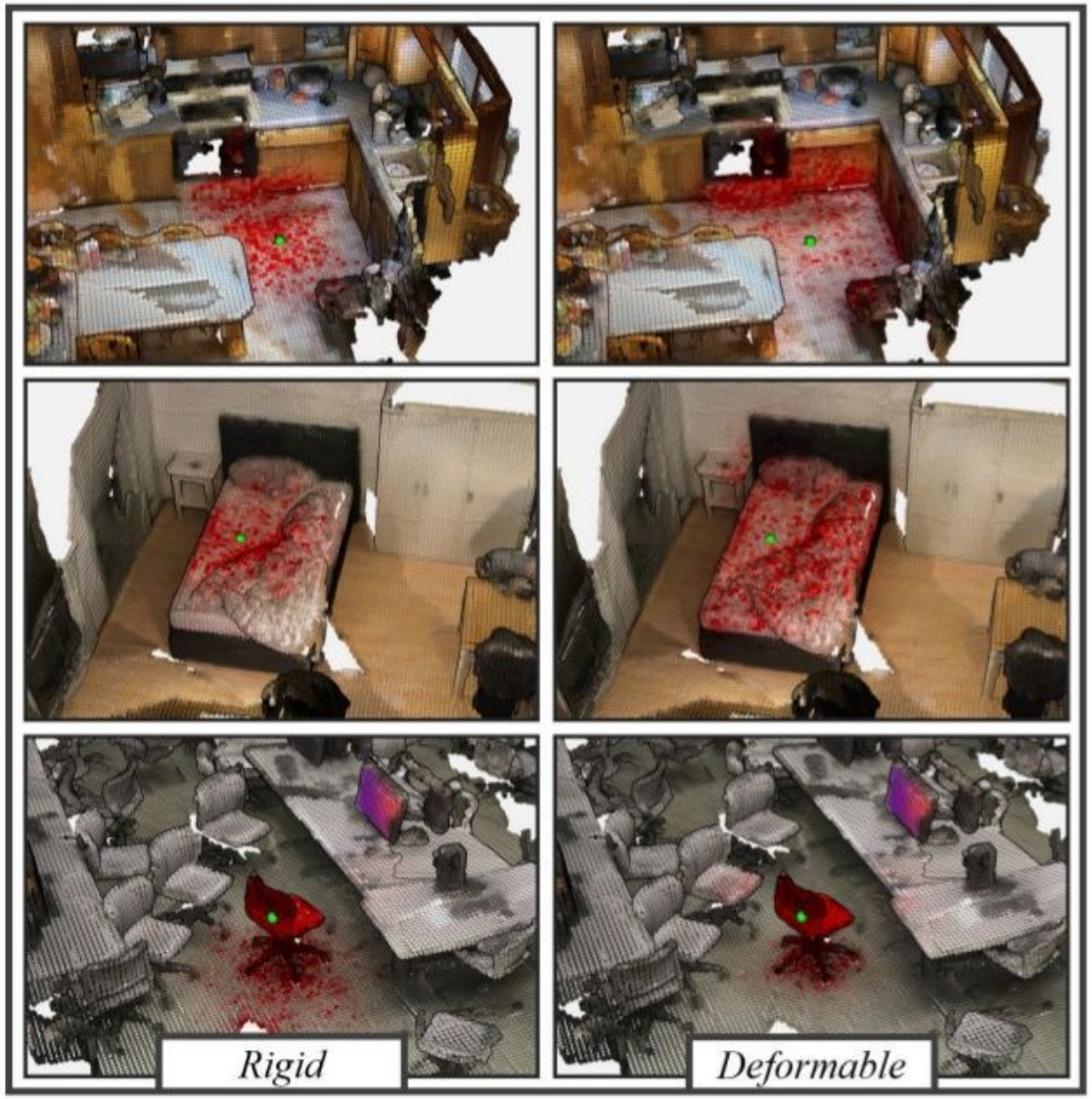


Analysis of Learned Filters

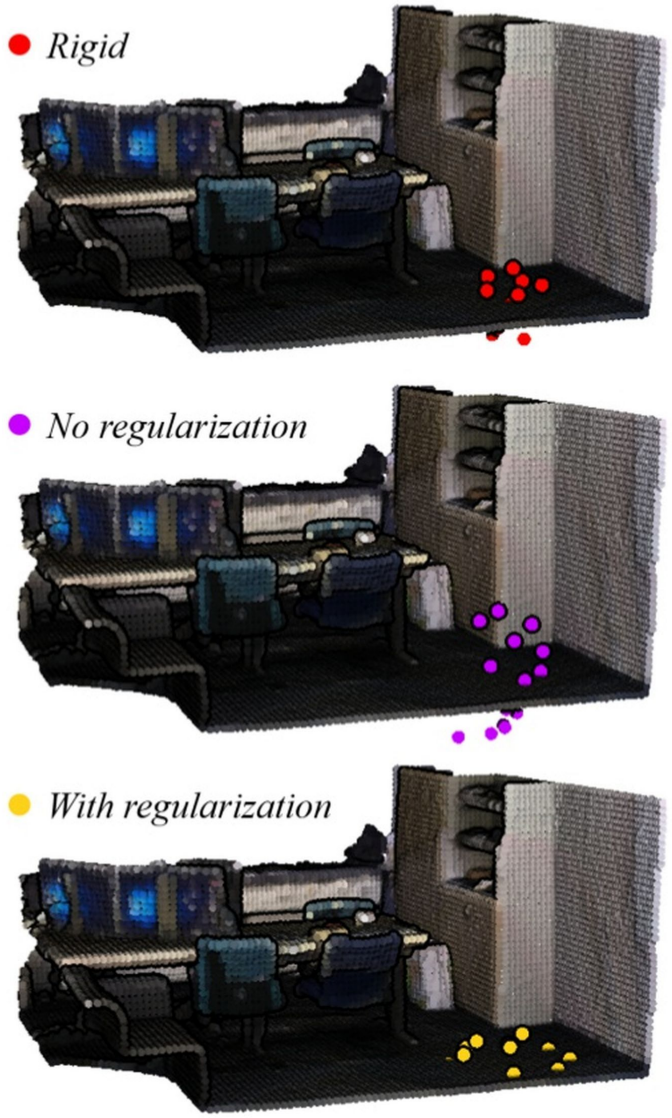




Analysis of Receptive Field



Analysis of Deformation Regularization



Open Issues

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i$$

$$g(y_i) = \sum_{k<K} h\left(y_i, \widetilde{x}_k\right) W_k$$

$$h\left(y_i, \widetilde{x}_k\right) = \max\left(0, 1 - \frac{\|y_i - \widetilde{x}_k\|}{\sigma}\right)$$

- Determine the neighborhood is hand-crafted and less flexible
- May cause problem in outdoor LiDAR point clouds

- The following paper learns the distance metric between points in high-dimensional feature space, and achieves better performance in outdoor LiDAR point clouds.

3DSSD: Point-based 3D Single Stage Object Detector

Zetong Yang† Yanan Sun‡ Shu Liu† Jiaya Jia†

†The Chinese University of Hong Kong ‡The Hong Kong University of Science and Technology

{tomztyang, now.syn, liushuhust}@gmail.com leojia@cse.cuhk.edu.hk

Contributions (Recap)

- Prior works
  - **Projection networks:** intermediate structure with limited receptive field
  - **Graph convolution:** feature aggregation over edges ignores local shape
  - **Pointwise MLP:** hard to capture local shape in an efficient manner
  - **Point convolution:** limited flexibility or representative power
- This paper proposes a new type of point convolution that
  - Learns *kernel* function to compute *point-wise* filters
  - Increases the discriminative power with *deformable* kernels
- This paper shows
  - State-of-the-art performance in a large variety of tasks (object classification, segmentation of small or large scenes)
  - Fast training and inference time for deep architectures