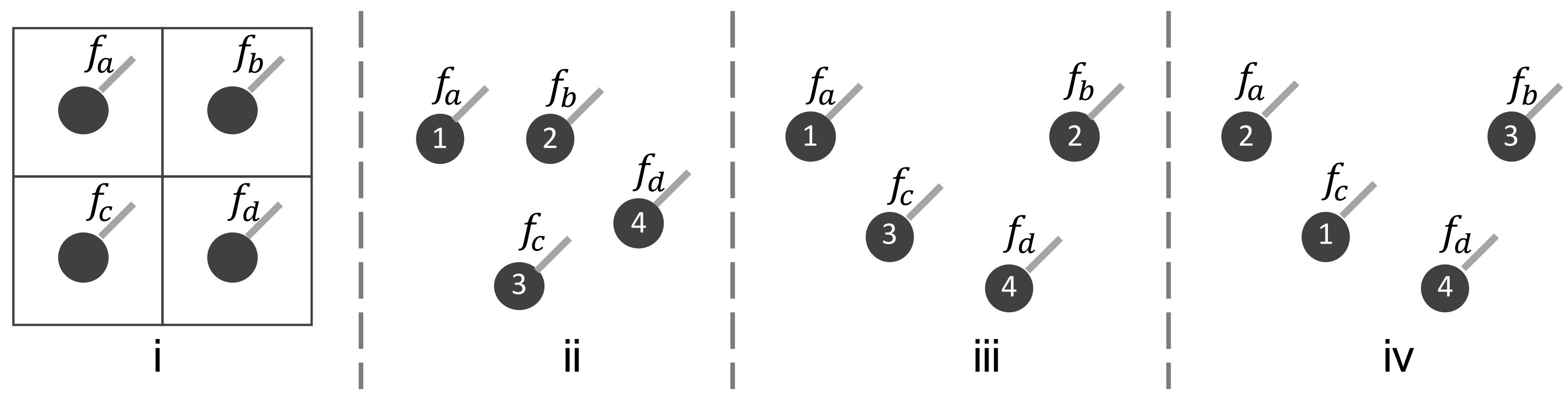
PointCNN

On Feature Learning from Point Cloud Data

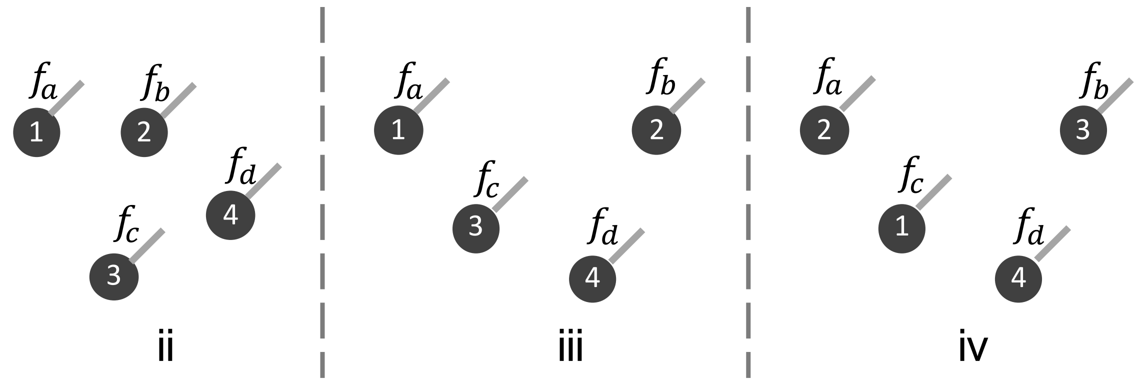
Motivation



Lack of regular grids poses the challenge of sorting the points into canonical orders

- CNNs are successful by picking-up on spatially-local correlations
- Point clouds (ii, iii, iv) are irregular and unordered
- Directly convolving against point cloud/features will lose shape information and be variant to the ordering
- **Can we learn a transformation to first permute the points to some canonical order?**

Challenges Applying CNNs



$$f_{ii} = Conv(\mathbf{K}, [f_a, f_b, f_c, f_d]^T),$$
$$f_{iii} = Conv(\mathbf{K}, [f_a, f_b, f_c, f_d]^T),$$
$$f_{iv} = Conv(\mathbf{K}, [f_c, f_a, f_b, f_d]^T).$$

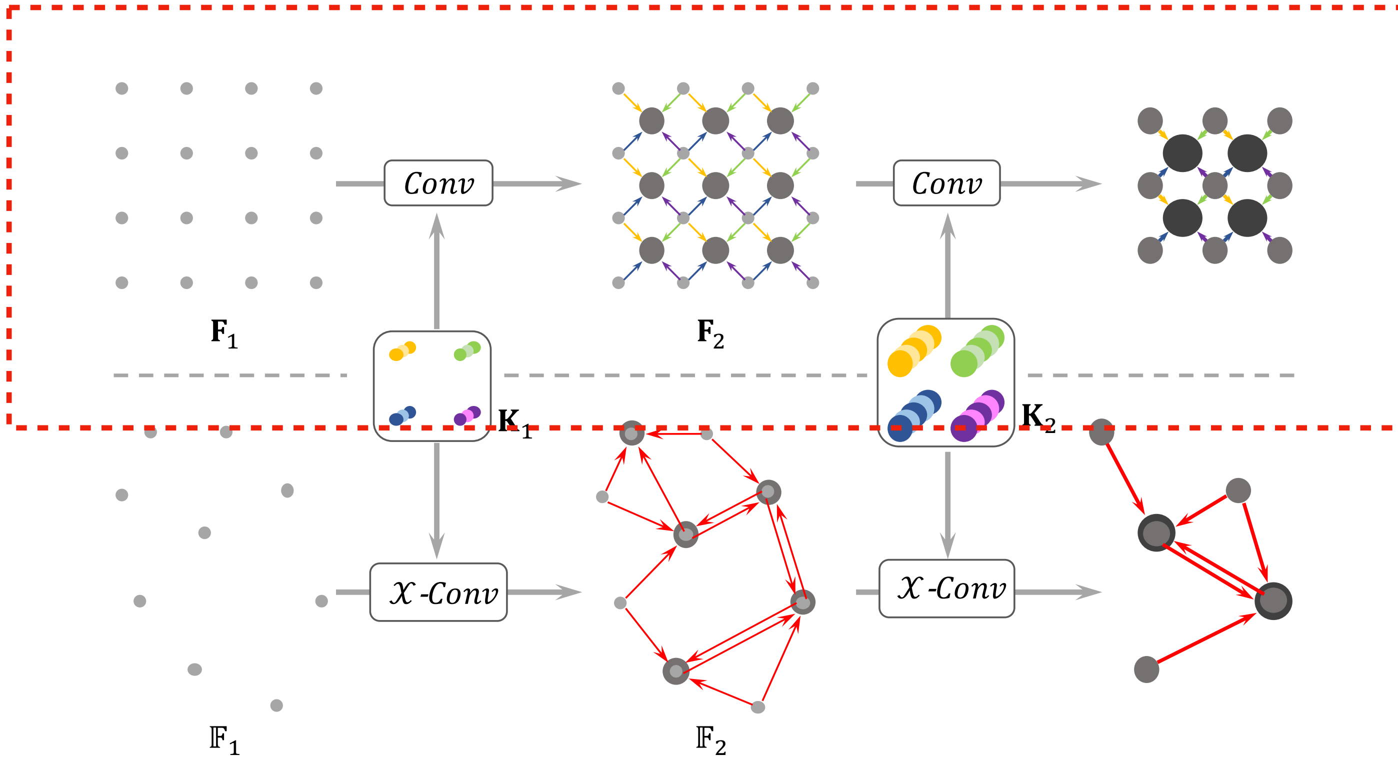
- Conv(**K**, […]) is simply an element-wise product followed by a sum
- By directly applying Conv(**K**, […]) we,
  1. Lose shape information ($f_{ii} = f_{iii}$)
  2. Become variant to the ordering ($f_{iii} \neq f_{iv}$)

X-Transformation

- Learns a K x K transformation with an MLP to weight and permute the input features
- Then apply the typical convolution on the transformed features
- X = MLP(p_1, p_2, …, p_K)

$$f_{ii} = Conv(\mathbf{K}, \mathcal{X}_{ii} \times [f_a, f_b, f_c, f_d]^T),$$
$$f_{iii} = Conv(\mathbf{K}, \mathcal{X}_{iii} \times [f_a, f_b, f_c, f_d]^T),$$
$$f_{iv} = Conv(\mathbf{K}, \mathcal{X}_{iv} \times [f_c, f_a, f_b, f_d]^T),$$

Hierarchal Convolution (*for Images*)



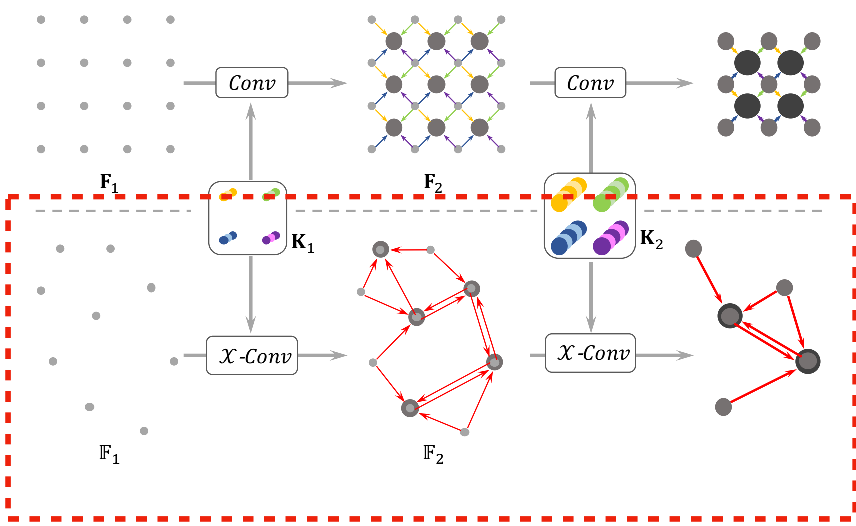
- $\mathbf{F}_2$ = Conv($\mathbf{K}$, $\mathbf{F}_1$)
    - $\mathbf{F}_1$: R_1 x R_1 x C_1
    - $\mathbf{F}_2$: R_2 x R_2 x C_2
        - $C_2 > C_1$ (deeper)
        - $R_2 < R_1$ (lower-rez)

- Applied recursively
- Producing feature maps in fewer and fewer spatial resolution
- But deeper and deeper channels

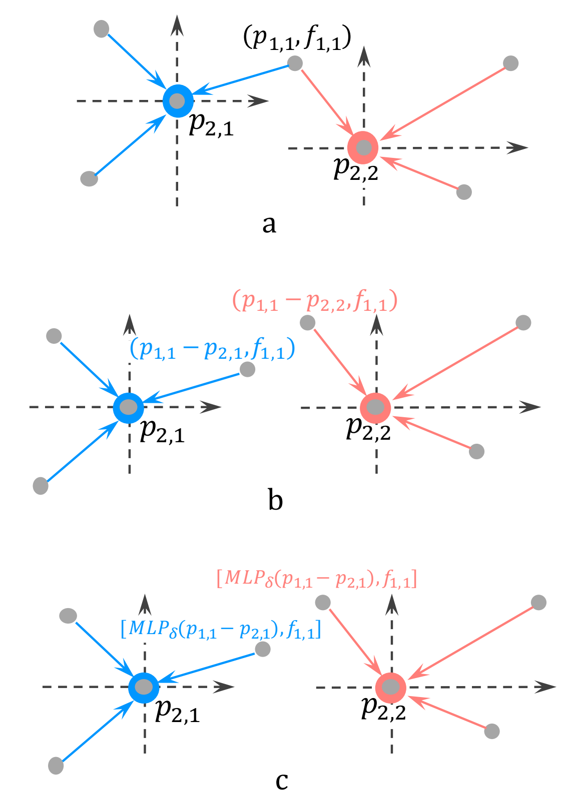
Hierarchal Convolution (*X-Conv*)

- $\mathbf{F}_p = \text{X-Conv}(\mathbf{K}, p, \mathbf{P}, \mathbf{F})$

  $= \text{Conv}(\mathbf{K}, \mathit{MLP}(\mathbf{P} - p) \times [\mathit{MLP}_\delta(\mathbf{P} - p), \mathbf{F}])$

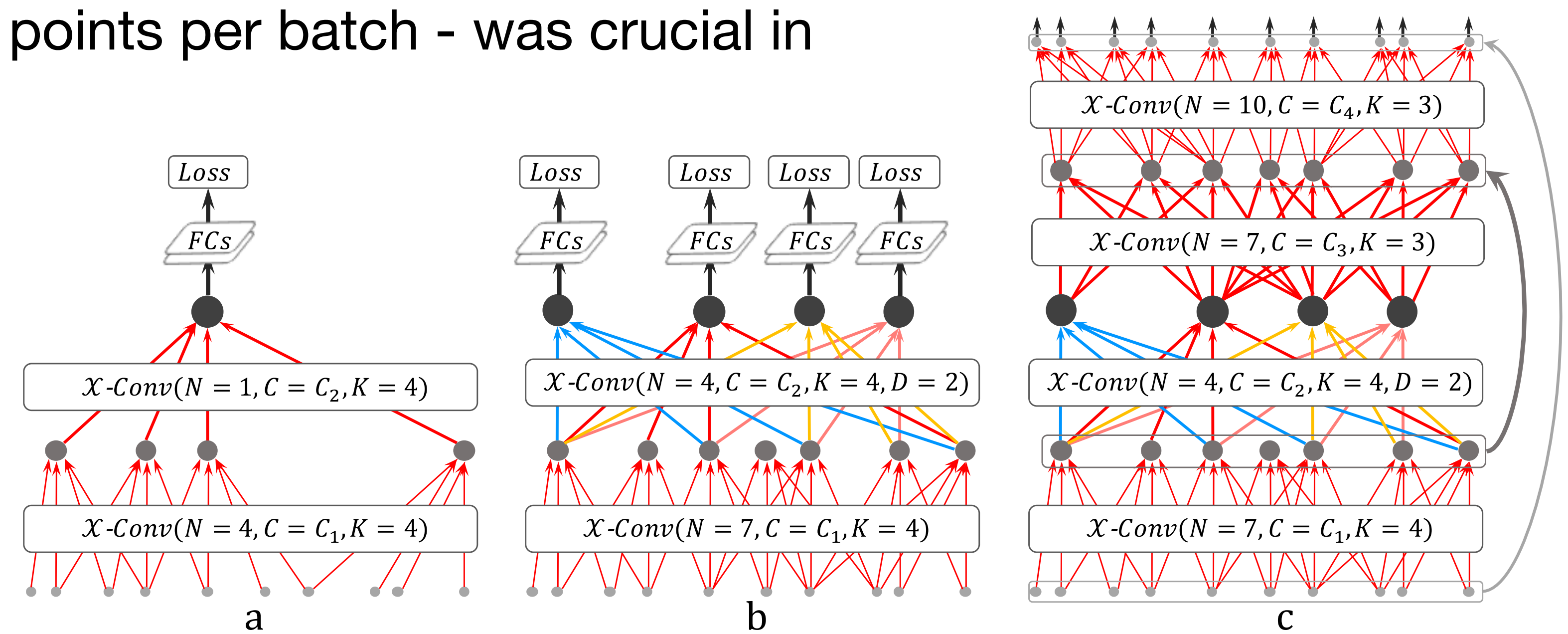


- ($\mathbf{P}' \leftarrow \mathbf{P} - p$) Since X-Conv is designed to work on local regions take point p as origin and look at K neighbouring points
- ($\mathbf{F}_\delta \leftarrow \mathit{MLP}_\delta(\mathbf{P}')$) Lift the coordinates into a higher dimensional/abstract representation
- ($\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$) Then combine w/ associated features



Architecture

- Receptive field defined by K/N
  - K: # of selected neighbouring points (in prev. layer)
  - N: # of points (in prev. layer)
  - K/N = 1 = “global view of entire shape”
- Used dilation rate D = 2, sampling K points from (K x D) / N
- For segmentation used Conv-DeConv (reverse: more points, fewer channels)
- ELU activations, batch normalization, dropout, ADAM optimizer
- Randomly sample + shuffle input points per batch - was crucial in training



Experiments & Results

Method	Input	Core Operator	ModelNet40	ScanNet
MVCNN [Su et al. 2015]	Images	2D Conv	90.1	-
FPNN [Li et al. 2016]	3D Dist. Field	1D Conv	87.5	-
Vol. CNN [Qi et al. 2016]	Voxels	3D Conv	89.9	74.9
O-CNN [Wang et al. 2017]	Octree Voxels	Sparse 3D Conv	90.6	-
PointNet [Qi et al. 2017a]	Point Cloud	Pointwise MLP	89.2	-
PointNet++ [Qi et al. 2017b]	Point Cloud	Multiscale Pointwise MLP	90.7	76.1
PointCNN	Point Cloud	$\mathcal{X}$-Conv	**91.7**	**77.9**

Table 1: Comparisons of classification accuracy (%) on ModelNet40 [Wu et al. 2015b] and ScanNet [Dai et al. 2017].

Method	ShapeNet Parts	S3DIS	ScanNet
PointNet [Qi et al. 2017a]	83.7	47.6	73.9
PointNet++ [Qi et al. 2017b]	85.1	-	84.5
SyncSpecCNN [Yi et al. 2017a]	84.74	-	-
Pd-Network [Klokov and Lempitsky 2017]	85.49	-	-
SSCN [Graham et al. 2017]	85.98	-	-
SegCloud [Tchapmi et al. 2017]	-	48.92	-
SPGraph [Landrieu and Simonovsky 2017]	-	54.06	-
PointCNN	**86.13**[2]	**62.74 (54.1, w/o RGB)**	**85.1**

Table 2: Segmentation comparisons on ShapeNet Parts [Yi et al. 2016] in part averaged IoU (%), S3DIS [Armeni et al. 2016] in mean IoU (%), and ScanNet [Dai et al. 2017] in per voxel accuracy (%).

Experiments & Results

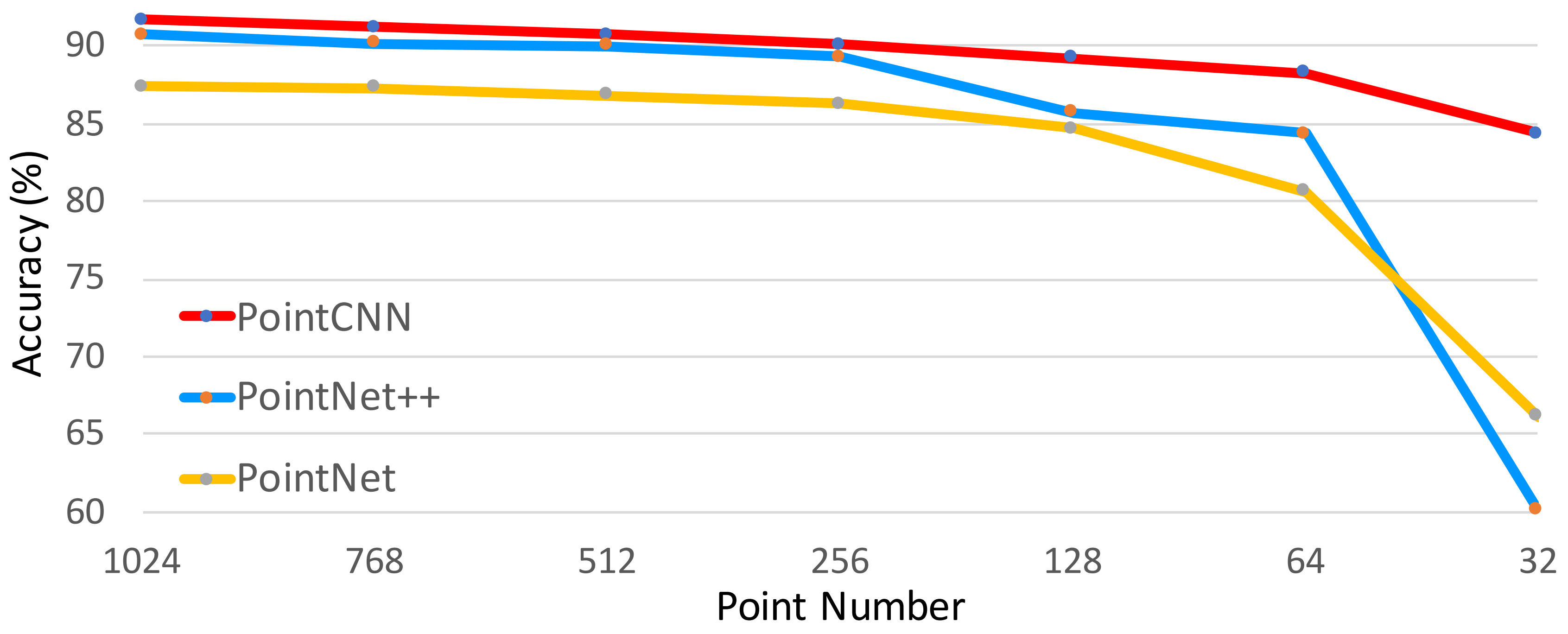
Method	TU-Berlin	Quick Draw
Sketch-a-Net [Yu et al. 2017]	**77.95**	-
AlexNet [Krizhevsky et al. 2012]	68.60	-
PointNet++ [Qi et al. 2017b]	66.53	51.58
PointCNN	67.72	**56.75**

Table 3: Accuracy (%) comparisons on Tu-Berlin [Eitz et al. 2012] and Quick Draw [Ha and Eck 2017] classification.

	PointCNN	w/o $\mathcal{X}$	w/o $\mathcal{X}$ (wider)	w/o $\mathcal{X}$ (deeper)
Core Layers	$\mathcal{X}$-Conv ×4	*Conv* × 4	*Conv* × 4	*Conv* × 6
# Parameter	0.45M	0.23M	0.49M	0.4M
Accuracy (%)	**91.7**	89.1	86.1	88.3

Table 4: Ablation test of PointCNN variants on ModelNet40 classification. $\mathcal{X}$-Conv is the key to PointCNN performance.

Experiments & Results



Stress test on few number of points