PointCNN

Yangyan Li, Rui Bu, Mingchao Sun, Baoquan Chen

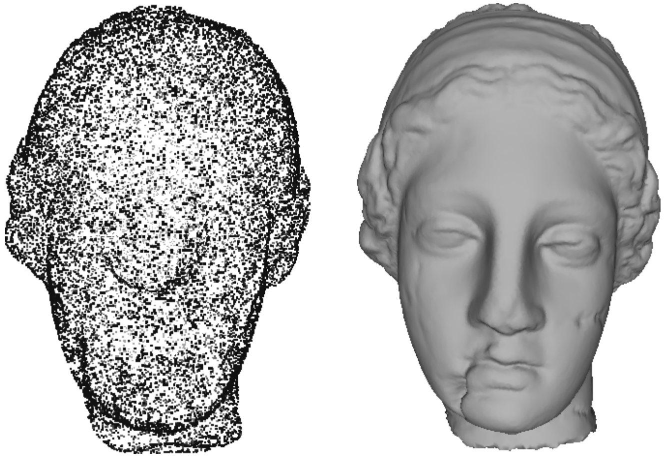
HANYANG UNIV. AI LAB

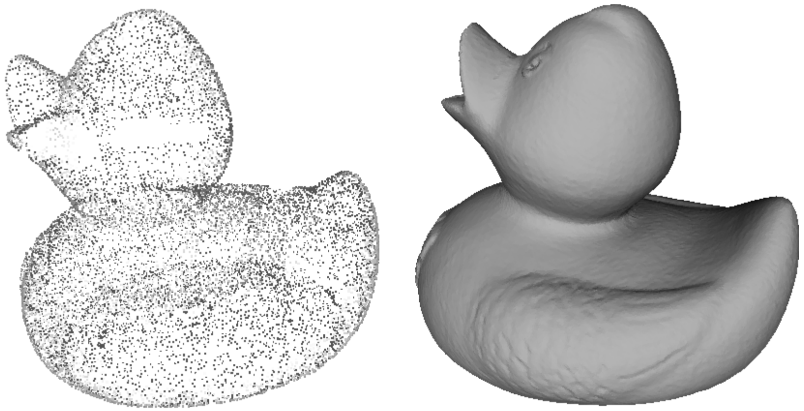
BYEONGJO KIM

Problem - Point Cloud

Point Cloud

- more effective in 3D space, or line sketches in 2D (less dimension)
- irregular, unordered

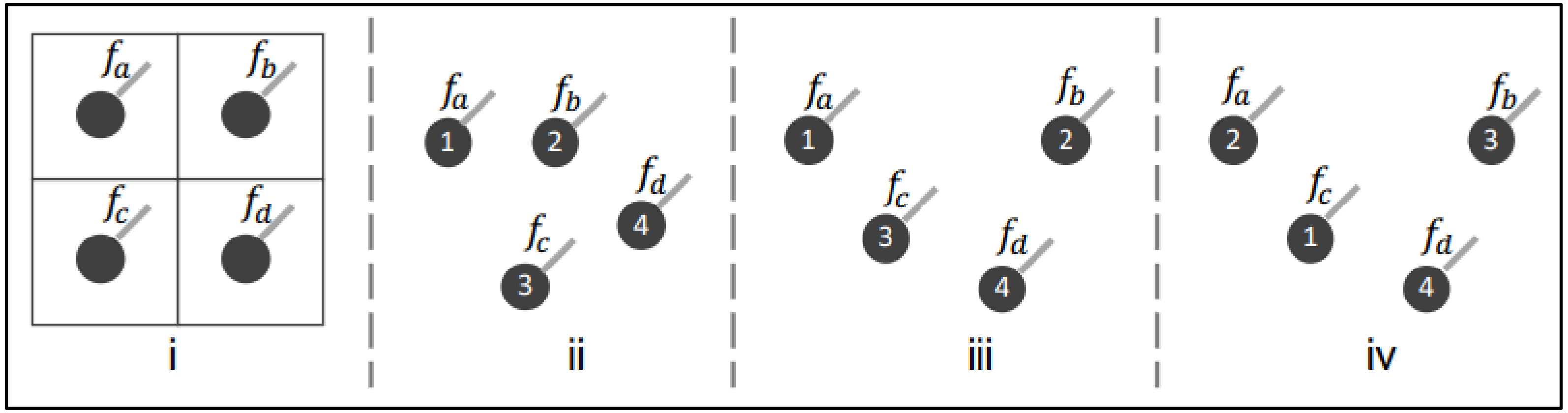




Problem - Point Cloud

Point Cloud

- more effective in 3D space, or line sketches in 2D (less dimension)
- irregular, unordered



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

$$\boxed{\begin{array}{l} f_{ii} \equiv f_{iii} \\ f_{iii} \neq f_{iv} \end{array}}$$

$$f_{ii} = Conv(\mathbf{K}, \mathcal{X}_{ii} \times [f_a, f_b, f_c, f_d]^T),$$
$$f_{iii} = Conv(\mathbf{K}, \mathcal{X}_{iii} \times [f_a, f_b, f_c, f_d]^T),$$
$$f_{iv} = Conv(\mathbf{K}, \mathcal{X}_{iv} \times [f_c, f_a, f_b, f_d]^T),$$
$$\mathcal{X} = MLP(p_1, p_2, \dots, p_K)$$

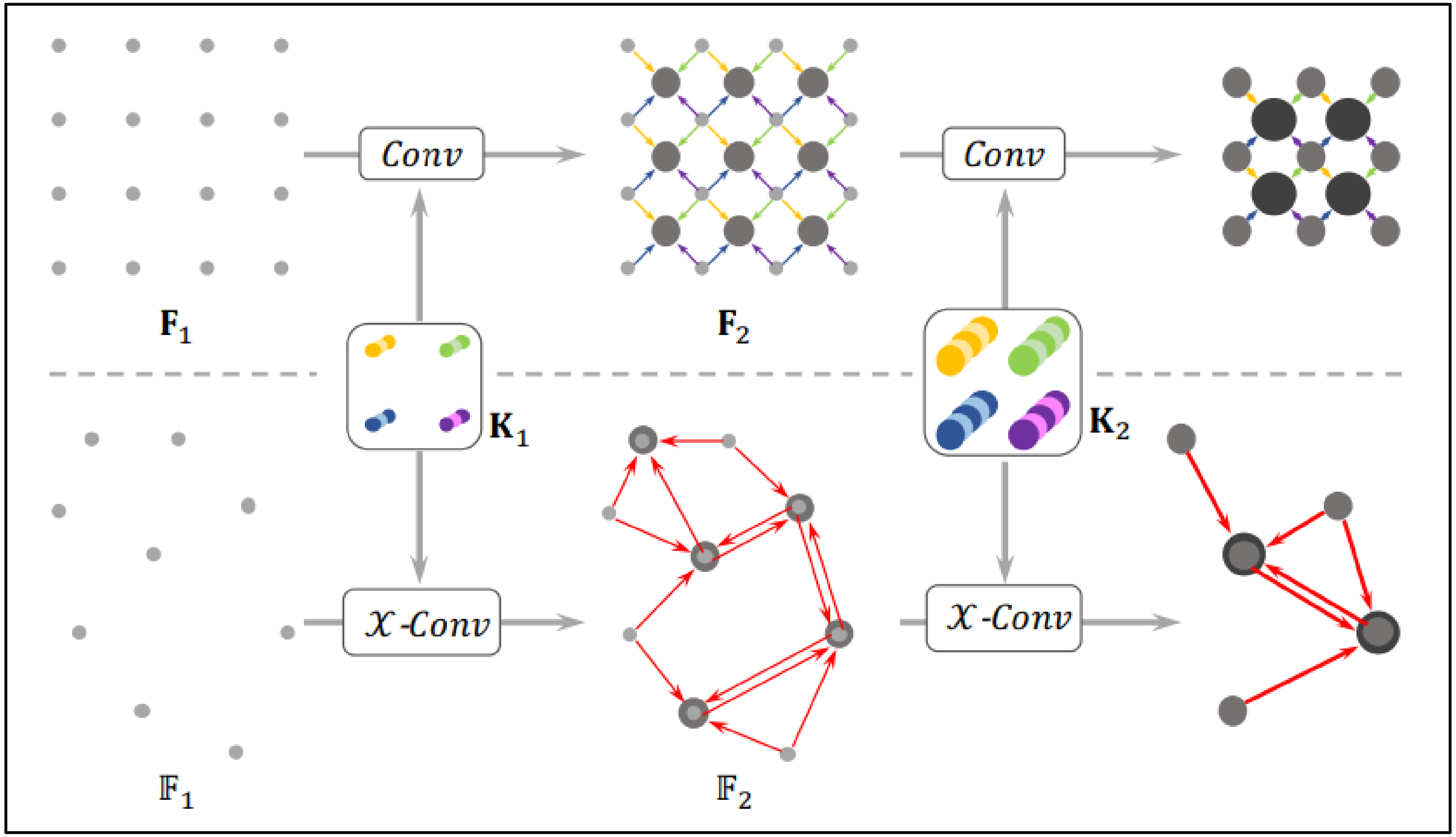
PointCNN

Hierarchical Convolution

X-Conv Operator

PointCNN Architectures

Hierarchical Convolution

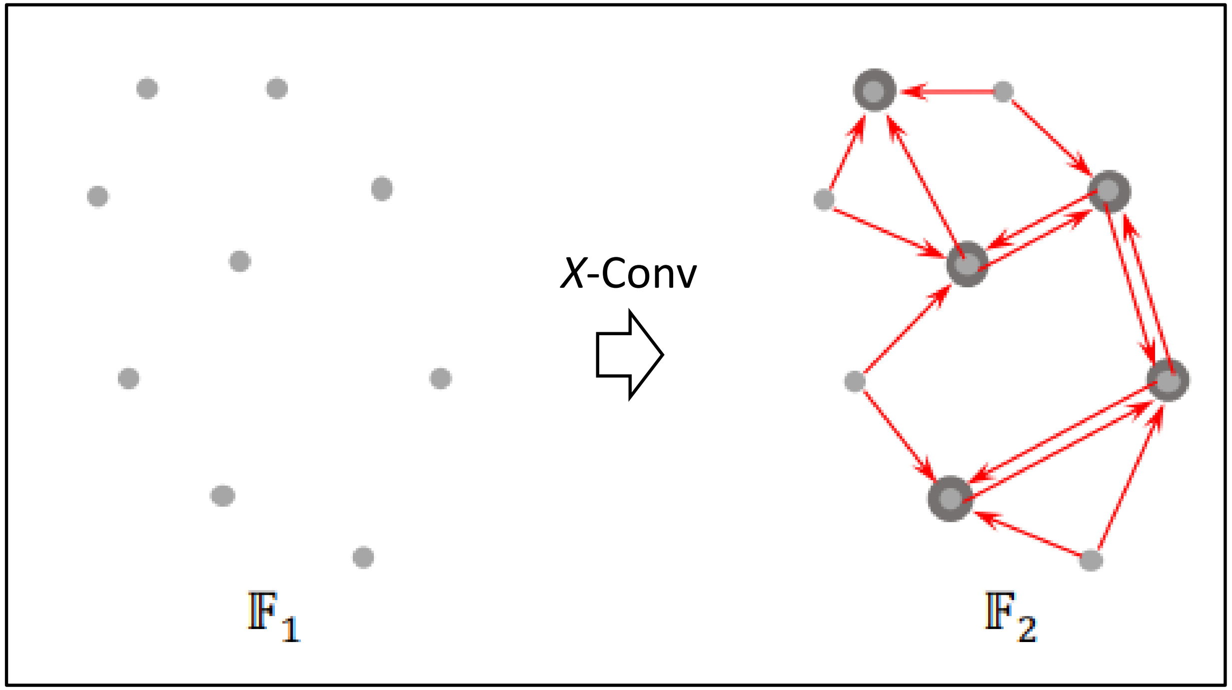


Hierarchical Convolution

$\mathbb{F}_1 = \{(p_{1,i}, f_{1,i}) : i = 1, 2, \ldots, N_1\}$ $\{p_{1,i} : p_{1,i} \in \mathbb{R}^D\}$ $\{f_{1,i} : f_{1,i} \in \mathbb{R}^{C_1}\}$

D : dimension of points
C : feature channel depth

$\mathbb{F}_2 = \{(p_{2,i}, f_{2,i}) : f_{2,i} \in \mathbb{R}^{C_2}, i = 1, 2, \ldots, N_2\}$



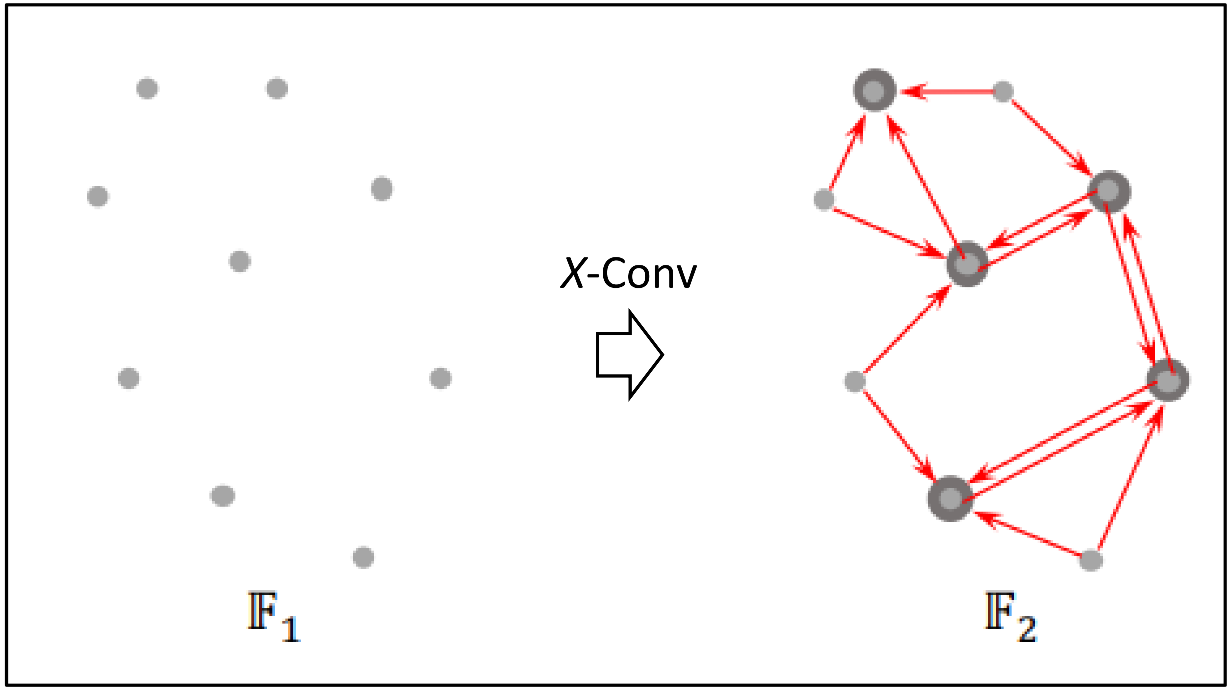
- $N_2 < N_1$, and $C_2 > C_1$
- $\{p_{2,i}\}$ is $\{p_{1,i}\}$'s representative points but not necessarily a subset of $\{p_{1,i}\}$
- $\{p_{2,i}\}$ is $\{p_{1,i}\}$'s
  random down-sampling for classification tasks
  farthest point sampling for segmentation tasks

Hierarchical Convolution

$\mathbb{F}_1 = \{(p_{1,i}, f_{1,i}) : i = 1, 2, \ldots, N_1\}$ $\{p_{1,i} : p_{1,i} \in \mathbb{R}^D\}$ $\{f_{1,i} : f_{1,i} \in \mathbb{R}^{C_1}\}$

D : dimension of points
C : feature channel depth

$\mathbb{F}_2 = \{(p_{2,i}, f_{2,i}) : f_{2,i} \in \mathbb{R}^{C_2}, i = 1, 2, \ldots, N_2\}$



- $N_2 < N_1$, and $C_2 > C_1$
- $\{p_{2,i}\}$ is $\{p_{1,i}\}$'s representative points but not necessarily a subset of $\{p_{1,i}\}$
- $\{p_{2,i}\}$ is $\{p_{1,i}\}$'s random down-sampling for classification tasks farthest p[illegible]ation tasks

Future Work!

$\mathcal{X}$-Conv Operator

$$
\begin{aligned}
\mathbf{F}_p &= \mathcal{X}\text{–}Conv(\mathbf{K}, p, \mathbf{P}, \mathbf{F}) \\
&= Conv(\mathbf{K}, MLP(\mathbf{P} - p) \times [MLP_\delta(\mathbf{P} - p), \mathbf{F}]),
\end{aligned}
$$

ALGORITHM 1: $\mathcal{X}$-Conv Operator

Input : $\mathbf{K}$, p, $\mathbf{P}$, $\mathbf{F}$

Output: $\mathbf{F}_p$ ▷ Features "projected", or "aggregated", to p

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$ ▷ Move $\mathbf{P}$ to local coordinate system of p

2: $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ ▷ **Individually** lift each point into C_δ dim. space

3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ ▷ Concatenate $\mathbf{F}_\delta$ and $\mathbf{F}$, $\mathbf{F}_*$ is a $K \times (C_\delta + C_1)$ matrix

4: $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ ▷ Learn the $K \times K$ $\mathcal{X}$-transformation matrix

5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$ ▷ Weight and permute $\mathbf{F}_*$ with the learnt $\mathcal{X}$

6: $\mathbf{F}_p \leftarrow \mathrm{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$ ▷ Finally, typical convolution between $\mathbf{K}$ and $\mathbf{F}_\mathcal{X}$

$\mathcal{X}$-Conv Operator

$$\mathbf{F}_p = \mathcal{X}\text{−}Conv(\mathbf{K}, p, \mathbf{P}, \mathbf{F})$$
$$= Conv(\mathbf{K}, MLP(\mathbf{P} - p) \times [MLP_\delta(\mathbf{P} - p), \mathbf{F}]),$$

ALGORITHM 1: $\mathcal{X}$-Conv Operator

Input : $\mathbf{K}, p, \mathbf{P}, \mathbf{F}$

Output : $\mathbf{F}_p$ ▷ Features "projected", or "aggregated", to p

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$ ▷ Move $\mathbf{P}$ to local coordinate system of p

2: $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ ▷ **Individually** lift each point into C_δ dim. space

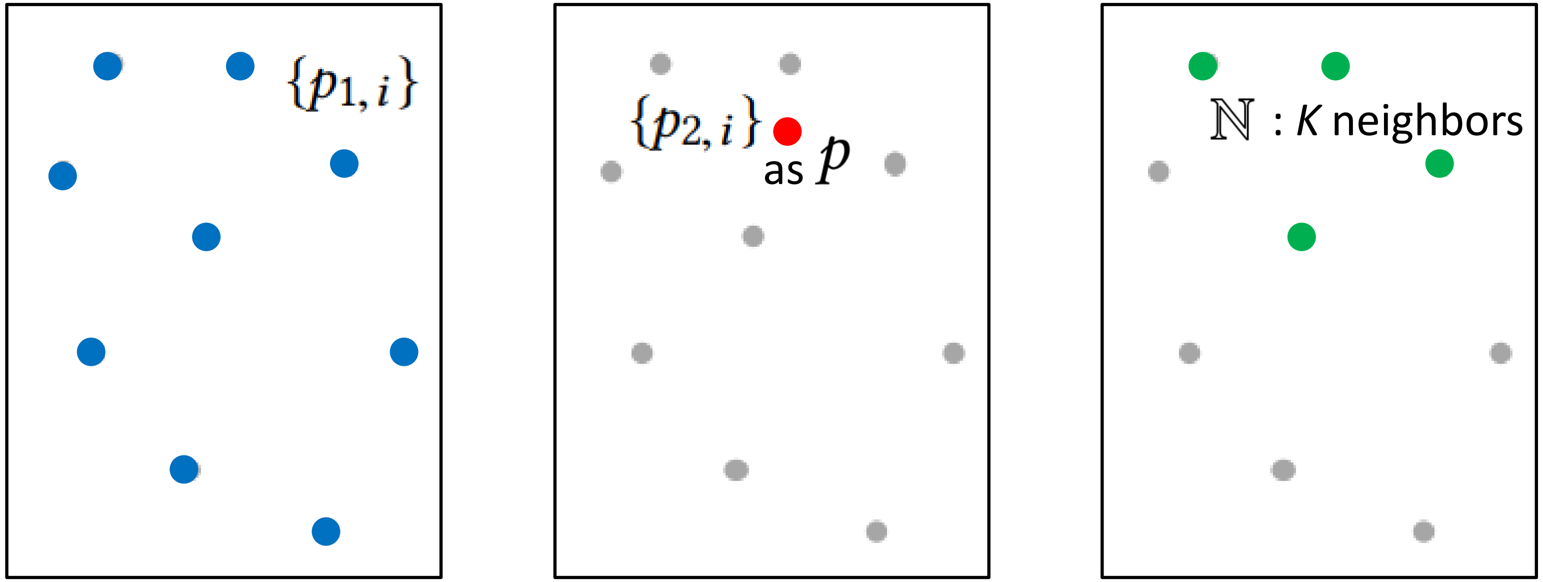
3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ ▷ Concatenate $\mathbf{F}_\delta$ and $\mathbf{F}$, $\mathbf{F}_*$ is a $K \times (C_\delta + C_1)$ matrix

4: $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ ▷ Learn the $K \times K$ $\mathcal{X}$-transformation matrix

5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$ ▷ Weight and permute $\mathbf{F}_*$ with the learnt $\mathcal{X}$

6: $\mathbf{F}_p \leftarrow \mathrm{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$ ▷ Finally, typical convolution between $\mathbf{K}$ and $\mathbf{F}_\mathcal{X}$

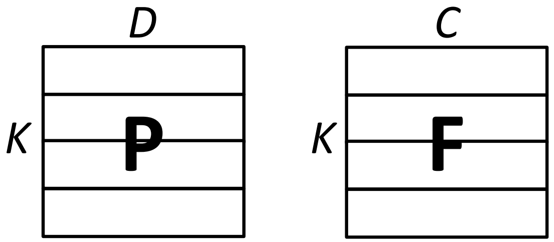
X-Conv Operator



$$\mathbb{S} = \{(p_i, f_i) : p_i \in \mathbb{N}\}$$

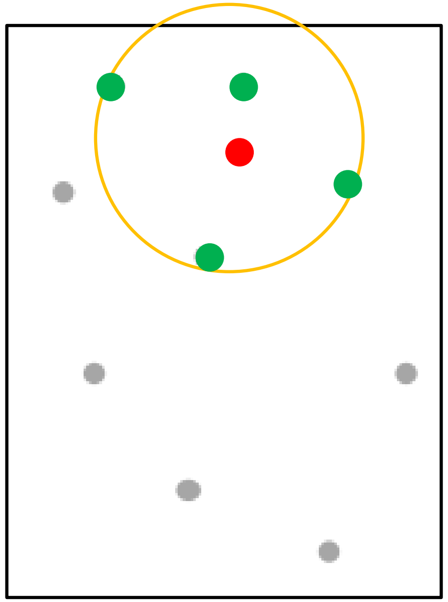
$$\mathbf{P} = (p_1, p_2, \ldots, p_K)^T$$

$$\mathbf{F} = (f_1, f_2, \ldots, f_K)^T$$



$\mathbf{F}_p$: projected or aggregated into the representative point

X-Conv Operator



K nearest neighbor search for extracting the *K* neighboring points

if point cloud with non-uniform point distribution

radius search

randomly sample *K* points out of the radius search results

$\mathcal{X}$-Conv Operator

$$\mathbf{F}_p = \mathcal{X}\text{−}Conv(\mathbf{K}, p, \mathbf{P}, \mathbf{F})$$
$$= Conv(\mathbf{K}, MLP(\mathbf{P} - p) \times [MLP_\delta(\mathbf{P} - p), \mathbf{F}]),$$

ALGORITHM 1: $\mathcal{X}$-Conv Operator

Input : $\mathbf{K}$, p, $\mathbf{P}$, $\mathbf{F}$

Output : $\mathbf{F}_p$ ▷ Features "projected", or "aggregated", to p

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$ ▷ Move $\mathbf{P}$ to local coordinate system of p

2: $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ ▷ **Individually** lift each point into C_δ dim. space

3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ ▷ Concatenate $\mathbf{F}_\delta$ and $\mathbf{F}$, $\mathbf{F}_*$ is a $K \times (C_\delta + C_1)$ matrix

4: $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ ▷ Learn the $K \times K$ $\mathcal{X}$-transformation matrix

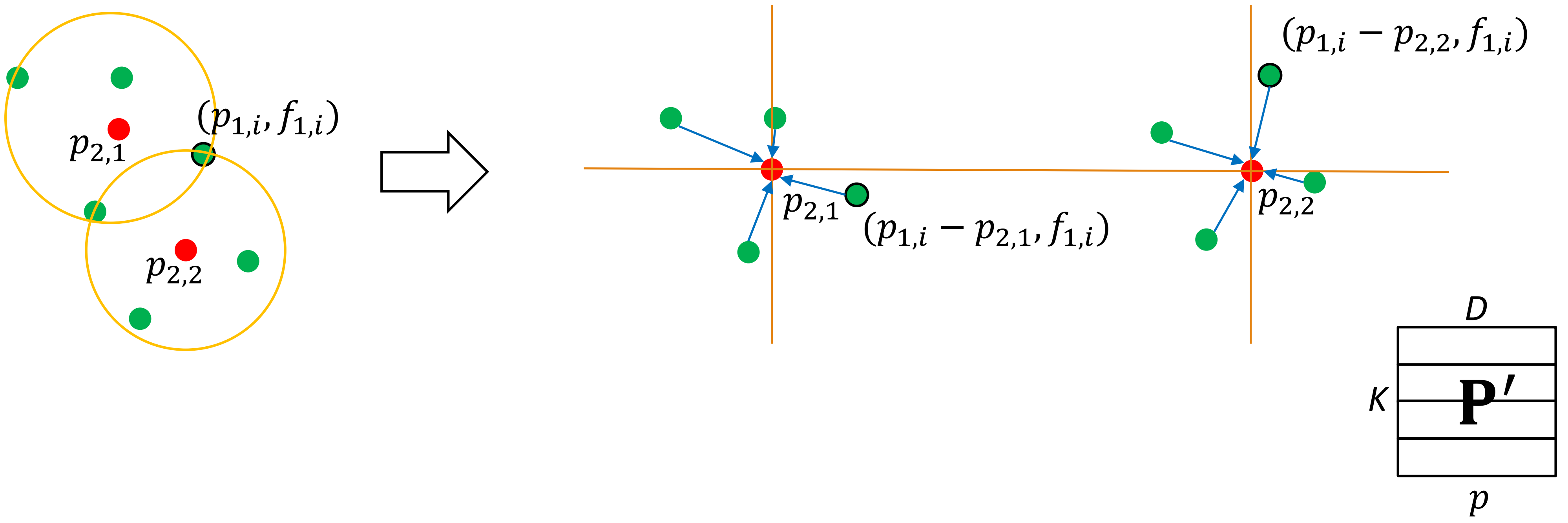
5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$ ▷ Weight and permute $\mathbf{F}_*$ with the learnt $\mathcal{X}$

6: $\mathbf{F}_p \leftarrow \mathrm{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$ ▷ Finally, typical convolution between $\mathbf{K}$ and $\mathbf{F}_\mathcal{X}$

$\mathcal{X}$-Conv Operator

Relative positions

Build local coordinate systems at the representative points



$\mathcal{X}$-Conv Operator

$$
\begin{aligned}
\mathbf{F}_p &= \mathcal{X}\text{−}Conv(\mathbf{K}, p, \mathbf{P}, \mathbf{F}) \\
&= Conv(\mathbf{K}, MLP(\mathbf{P} - p) \times [MLP_\delta(\mathbf{P} - p), \mathbf{F}]),
\end{aligned}
$$

ALGORITHM 1: $\mathcal{X}$-Conv Operator

Input : $\mathbf{K}$, p, $\mathbf{P}$, $\mathbf{F}$

Output : $\mathbf{F}_p$ ▷ Features "projected", or "aggregated", to p

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$ ▷ Move $\mathbf{P}$ to local coordinate system of p

2: $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ ▷ **Individually** lift each point into C_δ dim. space

3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ ▷ Concatenate $\mathbf{F}_\delta$ and $\mathbf{F}$, $\mathbf{F}_*$ is a $K \times (C_\delta + C_1)$ matrix

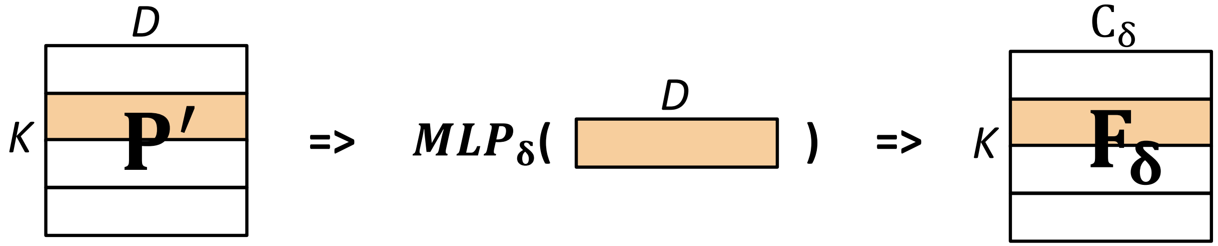
4: $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ ▷ Learn the $K \times K$ $\mathcal{X}$-transformation matrix

5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$ ▷ Weight and permute $\mathbf{F}_*$ with the learnt $\mathcal{X}$

6: $\mathbf{F}_p \leftarrow \mathrm{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$ ▷ Finally, typical convolution between $\mathbf{K}$ and $\mathbf{F}_\mathcal{X}$

$\mathcal{X}$-Conv Operator

their associated features, that defines the output features. In other word, besides the associated features, the local coordinates themselves are part of the input features as well. However, the local coordinates are of quite different dimensionality and representation than the associated features. We first lift the coordinates into an higher dimensional and more abstract representation ($\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$,



$\mathcal{X}$-Conv Operator

$$\begin{aligned}\mathbf{F}_p &= \mathcal{X}\text{–}Conv(\mathbf{K}, p, \mathbf{P}, \mathbf{F}) \\ &= Conv(\mathbf{K}, MLP(\mathbf{P}-p) \times [MLP_\delta(\mathbf{P}-p), \mathbf{F}]),\end{aligned}$$

ALGORITHM 1: $\mathcal{X}$-Conv Operator

Input : $\mathbf{K}$, p, $\mathbf{P}$, $\mathbf{F}$

Output : $\mathbf{F}_p$ ▷ Features "projected", or "aggregated", to p

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$ ▷ Move $\mathbf{P}$ to local coordinate system of p

2: $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ ▷ **Individually** lift each point into C_δ dim. space

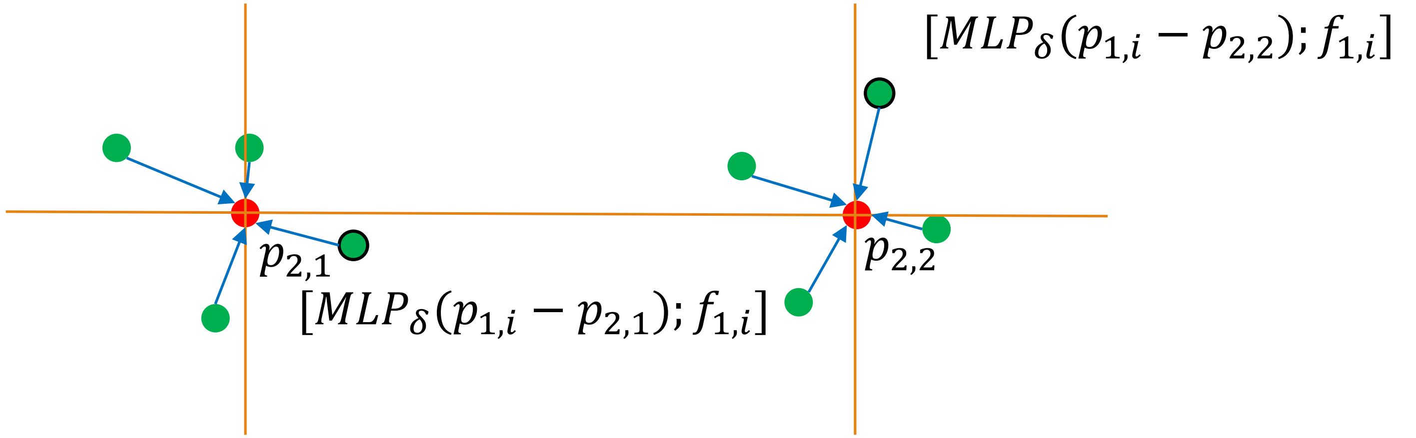
3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ ▷ Concatenate $\mathbf{F}_\delta$ and $\mathbf{F}$, $\mathbf{F}_*$ is a $K \times (C_\delta + C_1)$ matrix

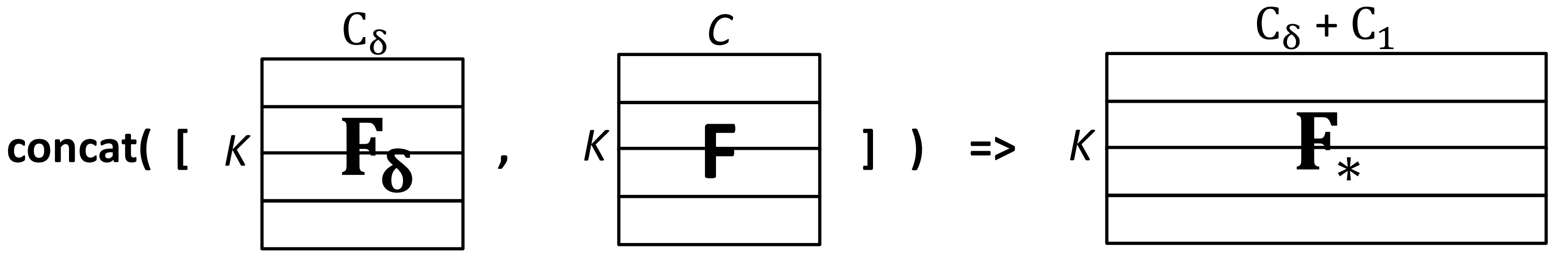
4: $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ ▷ Learn the $K \times K$ $\mathcal{X}$-transformation matrix

5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$ ▷ Weight and permute $\mathbf{F}_*$ with the learnt $\mathcal{X}$

6: $\mathbf{F}_p \leftarrow \mathrm{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$ ▷ Finally, typical convolution between $\mathbf{K}$ and $\mathbf{F}_\mathcal{X}$

$\mathcal{X}$-Conv Operator





$\mathcal{X}$-Conv Operator

$$\mathbf{F}_p = \mathcal{X}\text{−}Conv(\mathbf{K}, p, \mathbf{P}, \mathbf{F})$$
$$= Conv(\mathbf{K}, MLP(\mathbf{P} - p) \times [MLP_\delta(\mathbf{P} - p), \mathbf{F}]),$$

ALGORITHM 1: $\mathcal{X}$-Conv Operator

Input : $\mathbf{K}$, p, $\mathbf{P}$, $\mathbf{F}$

Output : $\mathbf{F}_p$ ▷ Features "projected", or "aggregated", to p

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$ ▷ Move $\mathbf{P}$ to local coordinate system of p

2: $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ ▷ **Individually** lift each point into C_δ dim. space

3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ ▷ Concatenate $\mathbf{F}_\delta$ and $\mathbf{F}$, $\mathbf{F}_*$ is a $K \times (C_\delta + C_1)$ matrix

4: $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ ▷ Learn the $K \times K$ $\mathcal{X}$-transformation matrix

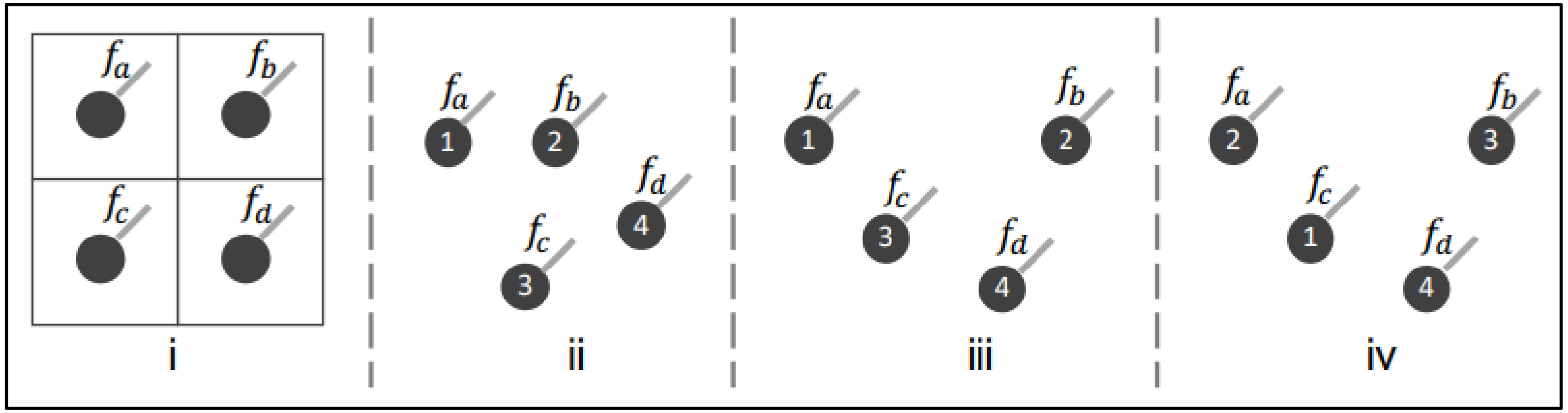
5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$ ▷ Weight and permute $\mathbf{F}_*$ with the learnt $\mathcal{X}$

6: $\mathbf{F}_p \leftarrow \mathrm{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$ ▷ Finally, typical convolution between $\mathbf{K}$ and $\mathbf{F}_\mathcal{X}$

$\mathcal{X}$-Conv Operator

Point Cloud

- more effective in 3D space, or line sketches in 2D (less dimension)
- irregular, unordered



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

$$\boxed{\begin{array}{c} f_{ii} \equiv f_{iii} \\ f_{iii} \neq f_{iv} \end{array}}$$

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

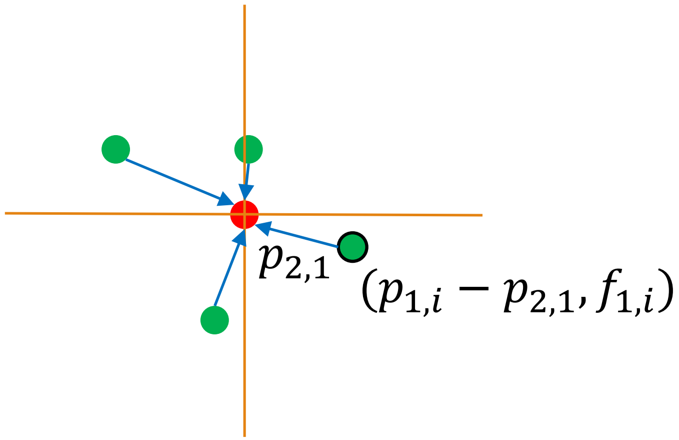
$$\mathcal{X} = MLP(p_1, p_2, ..., p_K)$$

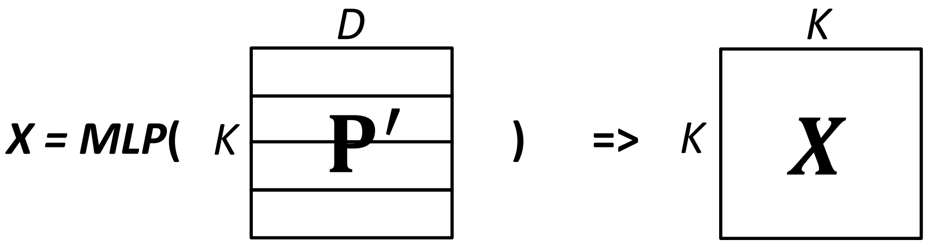
X-Conv Operator

The lifted features are weighted and permuted by X-transformation with the associated features

- Not individually, but together(be applied on the entire neighboring points)

X is dependent on the order of the points.





$\mathcal{X}$-Conv Operator

$$
\begin{aligned}
\mathbf{F}_p &= \mathcal{X}\!-\!Conv(\mathbf{K}, p, \mathbf{P}, \mathbf{F}) \\
&= Conv(\mathbf{K}, MLP(\mathbf{P} - p) \times [MLP_\delta(\mathbf{P} - p), \mathbf{F}]),
\end{aligned}
$$

ALGORITHM 1: $\mathcal{X}$-Conv Operator

Input : $\mathbf{K}$, p, $\mathbf{P}$, $\mathbf{F}$

Output : $\mathbf{F}_p$ ▷ Features "projected", or "aggregated", to p

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$ ▷ Move $\mathbf{P}$ to local coordinate system of p

2: $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ ▷ **Individually** lift each point into C_δ dim. space

3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ ▷ Concatenate $\mathbf{F}_\delta$ and $\mathbf{F}$, $\mathbf{F}_*$ is a $K \times (C_\delta + C_1)$ matrix

4: $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ ▷ Learn the $K \times K$ $\mathcal{X}$-transformation matrix

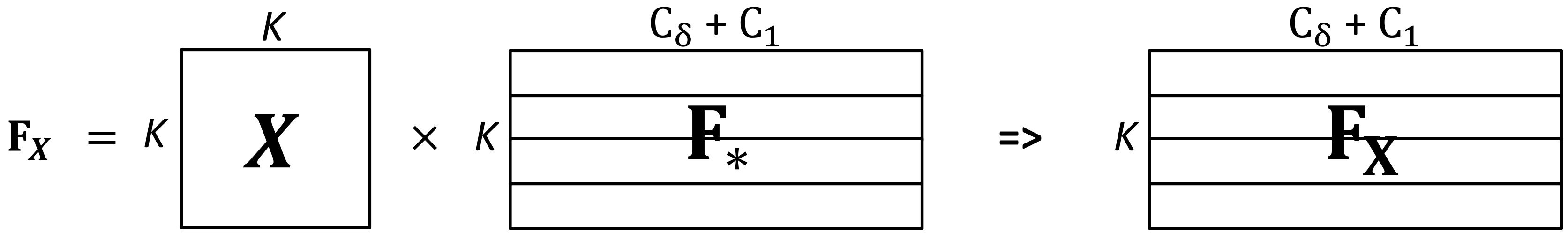
5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$ ▷ Weight and permute $\mathbf{F}_*$ with the learnt $\mathcal{X}$

6: $\mathbf{F}_p \leftarrow \mathrm{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$ ▷ Finally, typical convolution between $\mathbf{K}$ and $\mathbf{F}_\mathcal{X}$

$\mathcal{X}$-Conv Operator

$\mathcal{X}$ is supposed to permute F_* according to the input points.

It has to be aware of the specific input order.



$\mathcal{X}$-Conv Operator

$$
\begin{aligned}
\mathbf{F}_p &= \mathcal{X}\text{–}Conv(\mathbf{K}, p, \mathbf{P}, \mathbf{F}) \\
&= Conv(\mathbf{K}, MLP(\mathbf{P} - p) \times [MLP_\delta(\mathbf{P} - p), \mathbf{F}]),
\end{aligned}
$$

ALGORITHM 1: $\mathcal{X}$-Conv Operator

Input : $\mathbf{K}$, p, $\mathbf{P}$, $\mathbf{F}$

Output : $\mathbf{F}_p$ ▷ Features "projected", or "aggregated", to p

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$ ▷ Move $\mathbf{P}$ to local coordinate system of p

2: $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ ▷ **Individually** lift each point into C_δ dim. space

3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ ▷ Concatenate $\mathbf{F}_\delta$ and $\mathbf{F}$, $\mathbf{F}_*$ is a $K \times (C_\delta + C_1)$ matrix

4: $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ ▷ Learn the $K \times K$ $\mathcal{X}$-transformation matrix

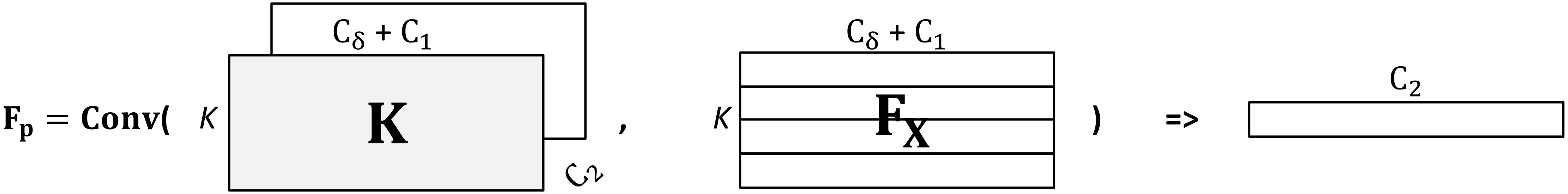
5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$ ▷ Weight and permute $\mathbf{F}_*$ with the learnt $\mathcal{X}$

6: $\mathbf{F}_p \leftarrow \mathrm{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$ ▷ Finally, typical convolution between $\mathbf{K}$ and $\mathbf{F}_\mathcal{X}$

$\mathcal{X}$-Conv Operator

$\mathrm{F}_p = \mathrm{Conv}(\mathbf{K}, \mathrm{F}_X)$

K => trainable kernel



$\mathcal{X}$-Conv Operator

$$
\begin{aligned}
\mathbf{F}_p &= \mathcal{X}\text{−}Conv(\mathbf{K}, p, \mathbf{P}, \mathbf{F}) \\
&= Conv(\mathbf{K}, MLP(\mathbf{P} - p) \times [MLP_\delta(\mathbf{P} - p), \mathbf{F}]),
\end{aligned}
$$

ALGORITHM 1: $\mathcal{X}$-Conv Operator

Input : $\mathbf{K}$, p, $\mathbf{P}$, $\mathbf{F}$

Output : $\mathbf{F}_p$ ▷ Features "projected", or "aggregated", to p

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$ ▷ Move $\mathbf{P}$ to local coordinate system of p

2: $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ ▷ **Individually** lift each point into C_δ dim. space

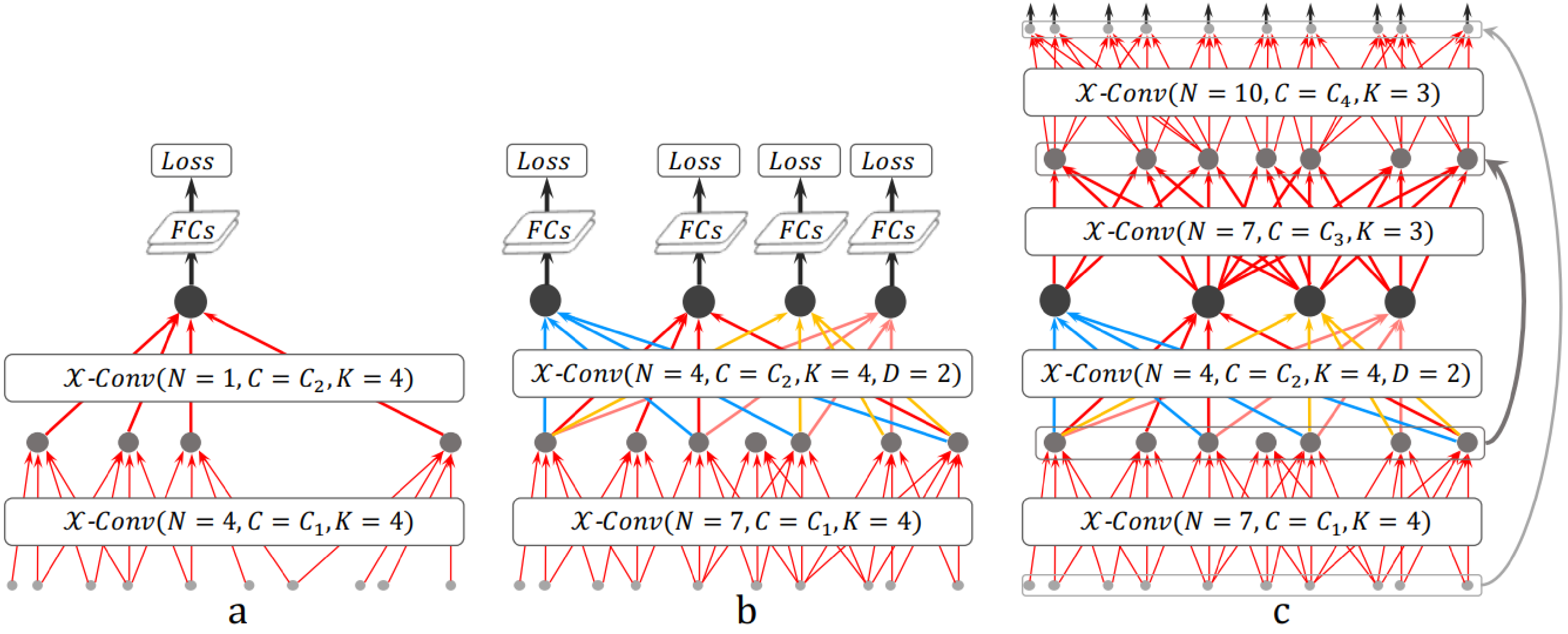
3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ ▷ Concatenate $\mathbf{F}_\delta$ and $\mathbf{F}$, $\mathbf{F}_*$ is a $K \times (C_\delta + C_1)$ matrix

4: $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ ▷ Learn the $K \times K$ $\mathcal{X}$-transformation matrix

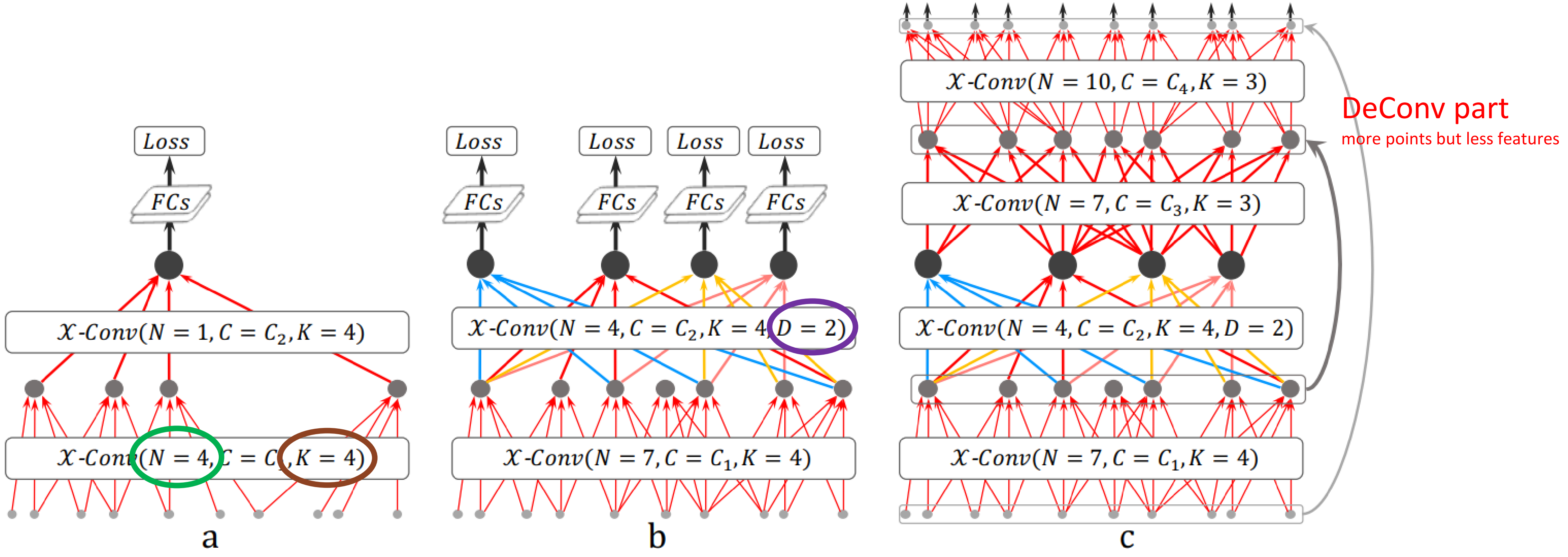
5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$ ▷ Weight and permute $\mathbf{F}_*$ with the learnt $\mathcal{X}$

6: $\mathbf{F}_p \leftarrow \mathrm{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$ ▷ Finally, typical convolution between $\mathbf{K}$ and $\mathbf{F}_\mathcal{X}$

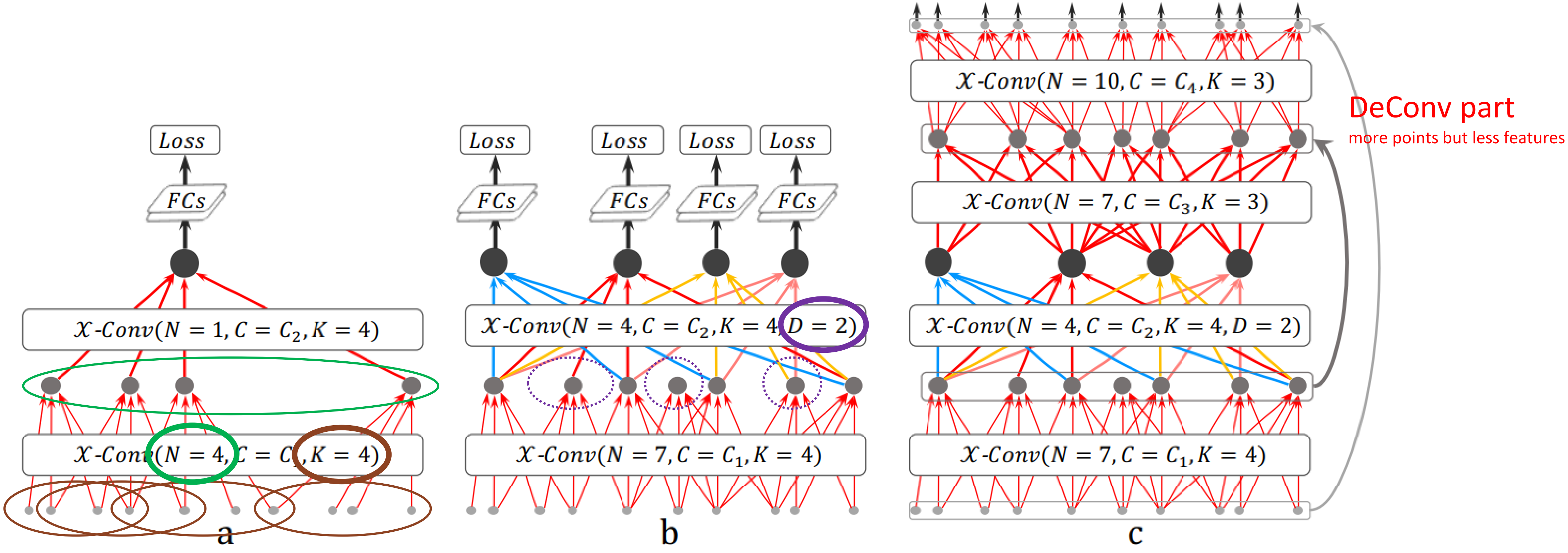
PointCNN Architectures



PointCNN Architectures



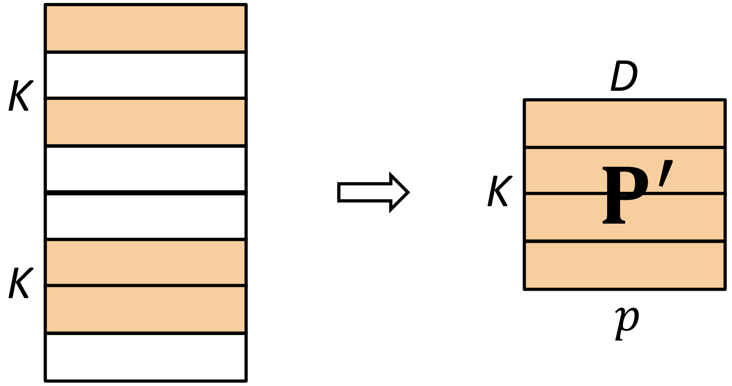
PointCNN Architectures

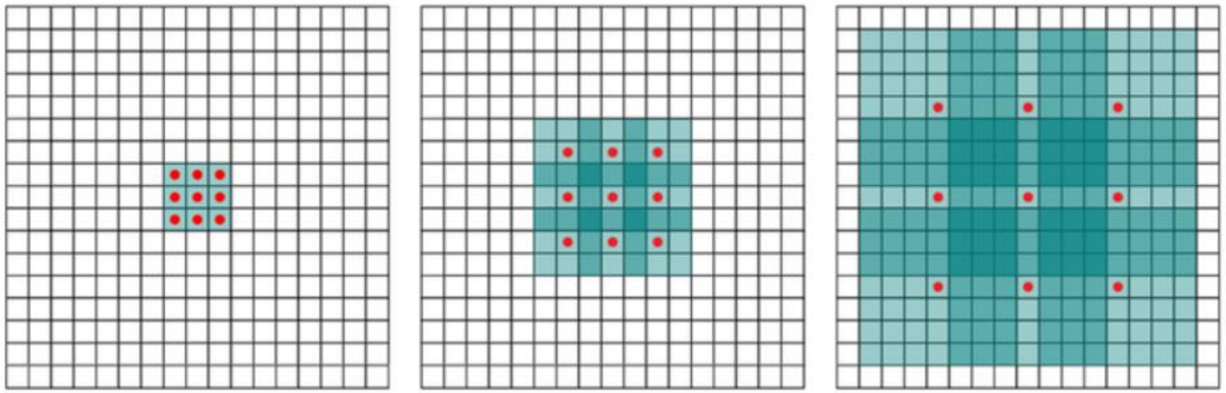


PointCNN Architectures

maintain the depth of the network && keeping the receptive field growth rate

=> Dilation rate (D)





(a) 1-dilated conv.

(b) 2-dilated conv.

(c) 4-dilated conv.

Other important things

using ELU nonlinear activation function

batch normalization is applied on P', $\mathrm{F_p}$ not F_*, X

using ADAM optimizer

0.01 learning rate

dropout before the last fully connected layer for reducing over-fitting

not beneficial if the neighboring points are always the same set in the same order

- randomly sample and shuffle the input points
- neighboring point sets and order can be different from batch to batch => Data augmentation
- N points as input => $Gaussian\ distribution\ N(\mathrm{N}, \left(\frac{N}{8}\right)^2)$ points are used for training

PointCNN model zoo

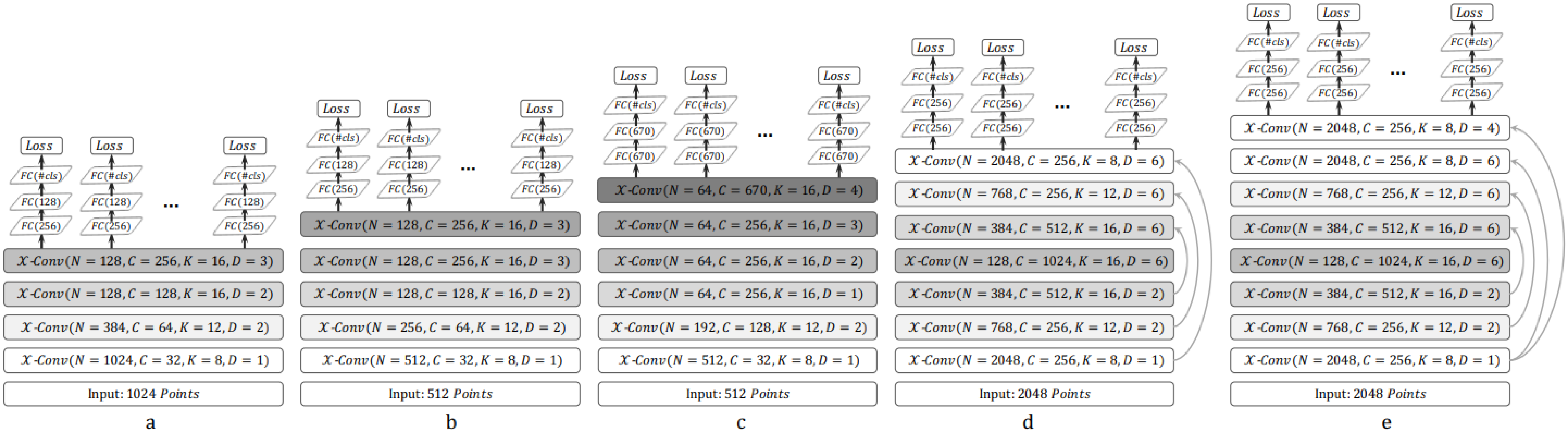


Figure 8: PointCNN model zoo, where (a) is used for ModelNet40 and ScanNet classification, (b) is used for TU-Berlin sketch classification, (c) is used for Quick Draw sketch classification, (d) is used for ScanNet and S3DIS segmentation, and (e) is used for ShapeNet Parts segmentation.

Better Than State-of-the-art

generalization of typical CNNs into learning features from point cloud, thus we call it *PointCNN*. Experiments show that PointCNN achieves on par or better performance than state-of-the-art methods on multiple challenging benchmark datasets and tasks.

Discussions and Future Work

> Strong over-fitting on small datasets.

> Is there some structural constraints in *X*?

> In PointCNN the learnt features are “projected” or “aggregated” at the specific representative points

- outperform at tasks where locations matter more.

> PointCNN performs quite well in learning the digits’ shape information.

- the sparser the data is, the more prominent the advantage of PointCNN can be observed.

> Study the principle criteria for making the choice (CNN+dense or PointCNN+point cloud)

> PointCNN + CNN for 3D model.