ELTE | FACULTY OF INFORMATICS

3D Point Cloud processing and analysis
Deep Learning Based Point Cloud Analysis
Classification and Segmentation

Massinissa Aouragh:
Faculty of Informatics, Department of Artificial Intelligence
Robert Bosch Kft
m2j7au@inf.elte.hu

Dynamic Graphic CNN

- Novel Method introduced in Dynamic Graph CNN for Learning on Point Clouds in 2018
- Uses EdgeConv operators to help capture the local neighborhood information of points
- Computes graphs from point clouds in every layer
- Stacked graph operation to learn semantic relationships between groups of points
- Paper

Dynamic Graphic CNN

- Local neighborhood graph is constructed from point cloud
- Convolution like operator is applied on the edges for a given layer
- X = $\{x_1, x_2, \ldots, x_n\} \in R^F$ set of n points with dimension F, Initial layer can be 3D coordinates (can also have RGB and surface normal)
- Directed graph is created G= (V, E) is computed wherein the 3D points represent the vertices and E $\subset$ V * V are the edges
- Features at the edges are calculated with $e_{ij} = h_\theta(x_i, x_j)$ where $h_\theta: R^F \times R^F \rightarrow R^{F'}$ is nonlinear function with trainable parameters θ

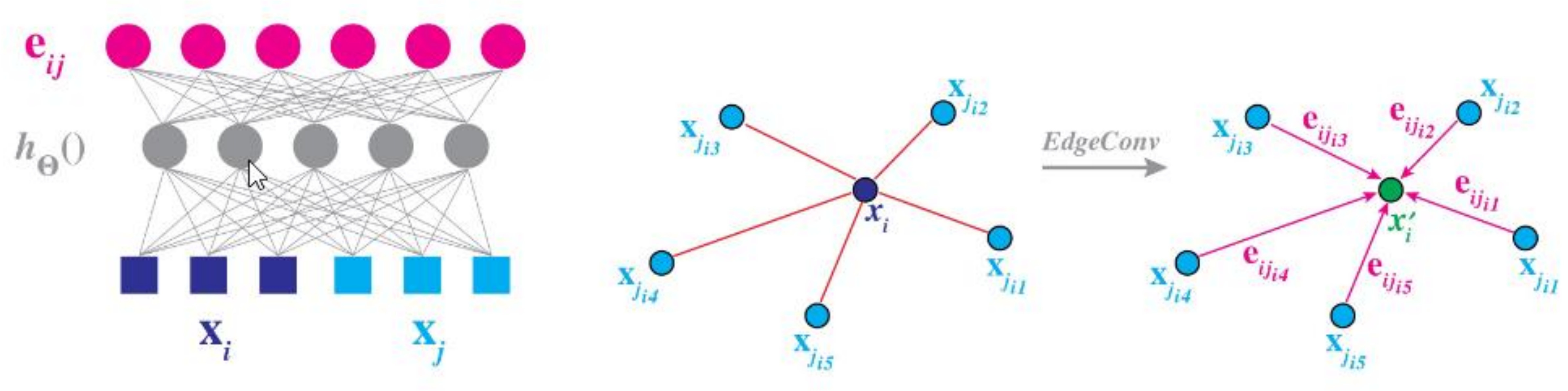
Dynamic Graphic CNN

- Channel wise symmetric function max or summation is applied on the edge features that outputs n points with dimension F’

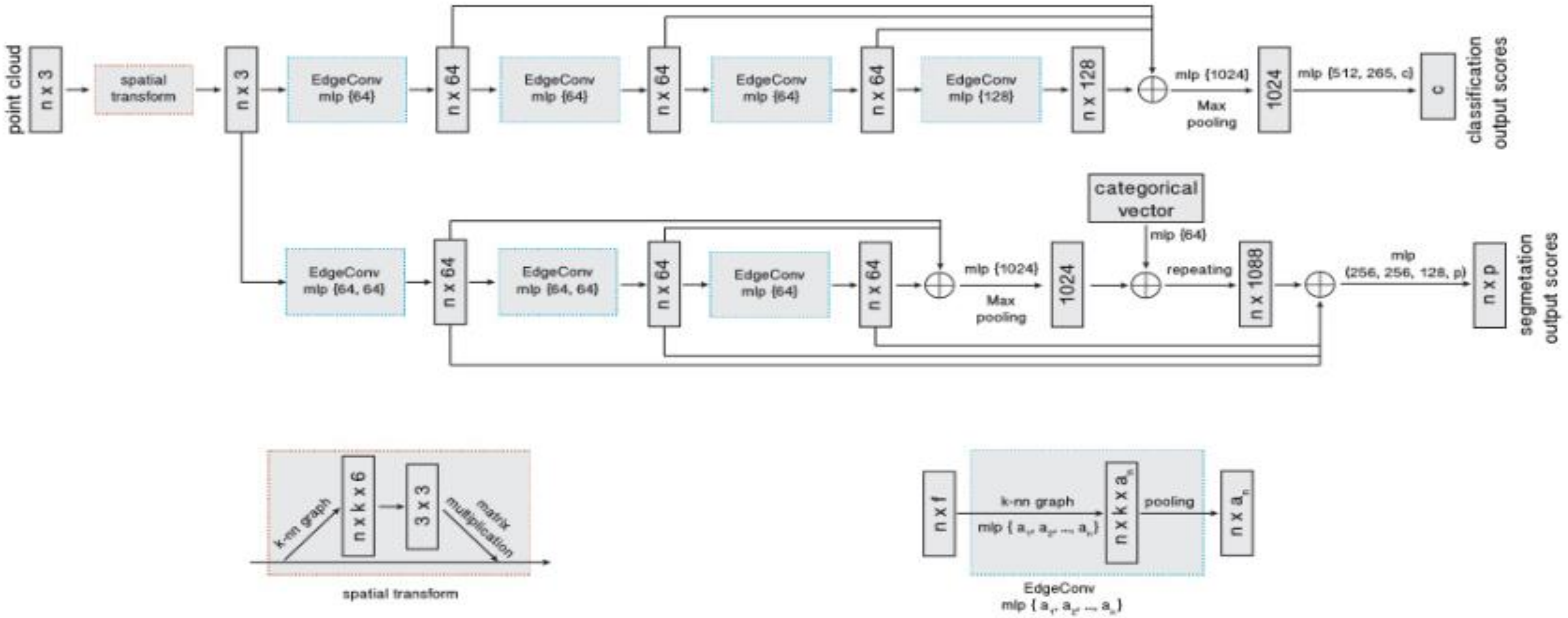
$$x' = \Omega_{j:(i,j)\in E}\ h_\theta(x_i, x_j)$$

- The choice of Ω is crucial and affects directly the performance of DGCNN
- Considering the local neighborhood structure $x_j - x_i$ the edge features are calculated as $e'_{ijm} = ReLU(\theta_m . (x_j - x_i) + \phi_m . x_i)$
- With max as symmetric function we have $x'_{im} = \max_{\{j:(i,j)\in \mathrm{E}\}} e'_{ijm}$

Dynamic Graphic CNN



Dynamic Graphic CNN



Dynamic Graphic CNN

- Unlike PointNet++ the point cloud downsampling is absent in DGCNN
- In every layer the same number of points is used
- DGCNN achieves an accuracy of 93.5% for point cloud classification on the ModelNet40
- IoU for the ShapeNet part segmentation dataset is 85.2%
- Overall accuracy for semantic segmentation on the S3DIS is 84.1%

PointCNN

- Proposes a method for learning a x-Transformation that can transform input points to a feature representation
- The feature representation is arranged in a way that a convolution operation can be applied
- The goal of the x-Transformation is to learn an order invariant K x K matrix for K input points using an MLP x = MLP($p_1, p_2, \dots p_k$)
- Under the invariant case, the output of the convolution operation f = Conv(K, x, $[p_1, p_2, \dots p_k]^T$)
- Paper

PointCNN

- Similar to CNN feature maps in images after each convolution operation the spatial resolution of the image is reduced with an increase in the number of the channels
- Input to a layer in PointCNN is $F_1 = \{(p_{1,i}, f_{1,i}): i = 1,2, \ldots N_1\}$, where $\{p_{1,i} : p_{1,i} \in R^{Dim}\}$ is the set of points and $\{f_{1,i} : f_{1,i} \in R^{C_1}\}$ is the set of features
- PointCNN seeks to apply x-Conv to F_1 to get set of $F_2 = \{(p_{2,i}, f_{2,i}): i = 1,2, \ldots N_2\}$
- Like the image case $N_2 < N_1$, meaning a smaller spatial resolution and $C_2 > C_1$ indicates deeper feature channels

PointCNN

- Using x-Conv we can transform $F_1\ to\ F_2$ let p be one representative point from the set $\{p_{2,i}\}$ of points in F_2 and f its features to be learned
- First a set of K nearest neighbors of p in the set of input points $\{p_{1,i}\}$ is retrieved
- For point p we will have K x Dim matrix P of points which has a corresponding K x C_1 matrix F of features and finally **K** the convolution kernels

PointCNN

- Having **K,** p, **P**, **F**
- The set P is centered at p to obtain P' as $P' \leftarrow P - p$
- The points are individually lifted to a higher dimensional space (C_δ dimensions) using MLP as $F_\delta \leftarrow MLP_\delta(P')$ like PointNet
- F_δ and F are concatenated to obtain K x (C_δ + C_1) dimensional matrix F_* as $F_* \leftarrow [F_\delta F]$

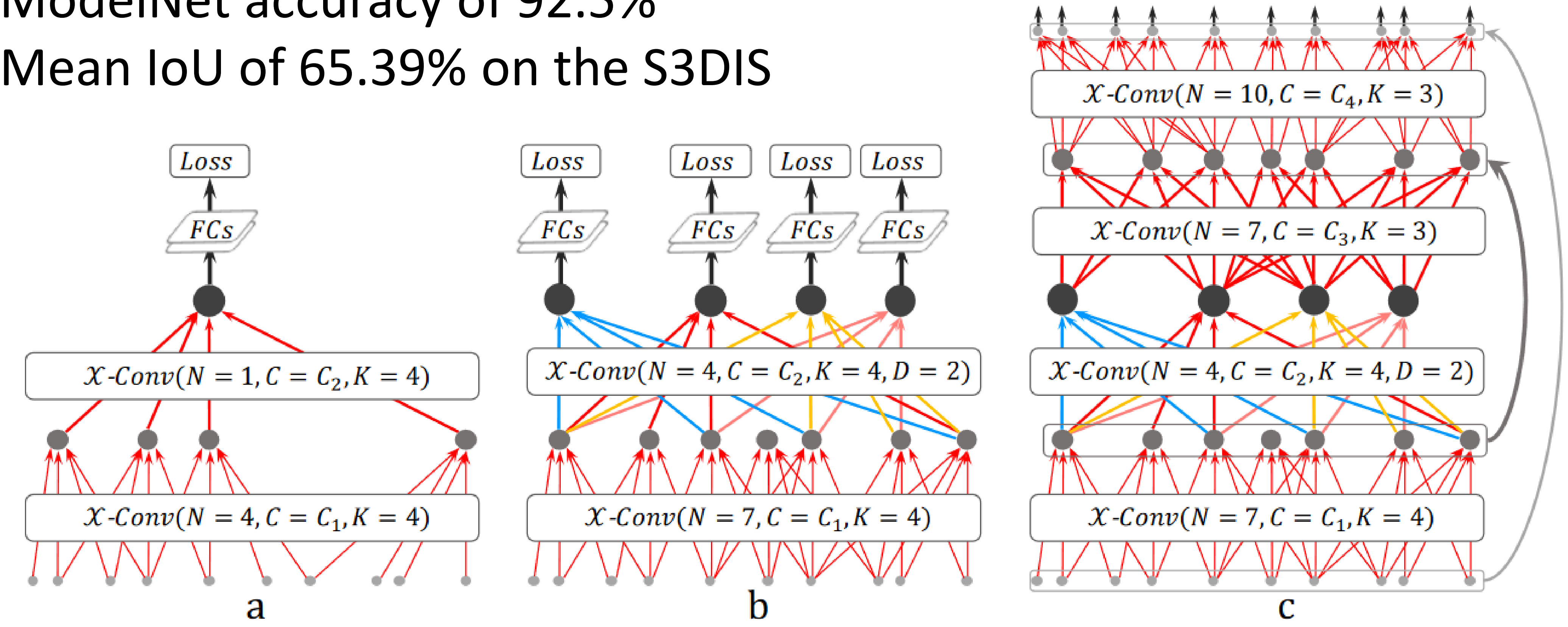
PointCNN

- The K x K x-transformation matrix is learned from P’ as $x \leftarrow MLP(P')$, x is aware of the order P’ and helps to achieve the permutation invariance
- F_* is weighed and permuted x to obtain the order invariant neighboring features F_x
- Convolution of F_x with kernel K yields the output F_p as $F_p \leftarrow Conv(K, F_x)$

PointCNN

ModelNet accuracy of 92.5%

Mean IoU of 65.39% on the S3DIS



PointSIFT

- Scale Invariant Feature Transform (SIFT) widely used 2D key-point detector and descriptor
- PointSIFT designs a scale and orientation aware module for 3D point cloud semantic segmentation tasks
- PointSIFT benefits from feature learning unlike the handcrafted features in SIFT
- PointSIFT can be inserted in PoinNet-like architectures to enrich the point features
- PointSIFT has orientation encoding unit that convolves features of nearest neighbors in eight orientations
- Paper

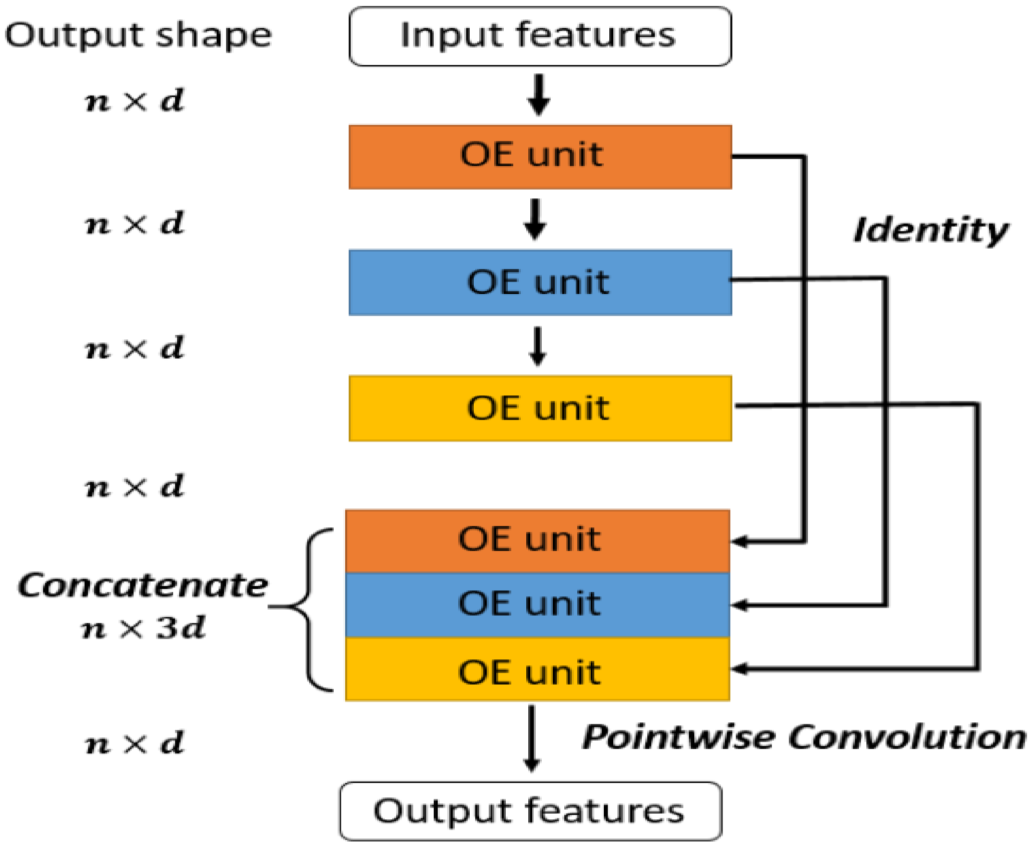
PointSIFT

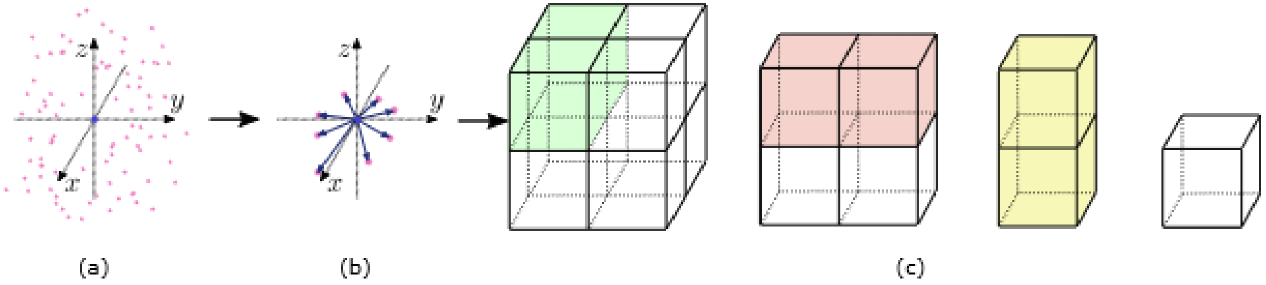
- The input is a n x d matrix of n points having d dimensional features the output is again a n x d
- The first step is the orientation encoding for every point information from the eight orientations is collected and integrated
- Collection of eight orientation is done by S8N search operation of neighboring using octree
- The features residing in voxel 2x2x2 are processed using orientation encoding convolution which convolves the voxel along the X, Y and Z axis

PointSIFT

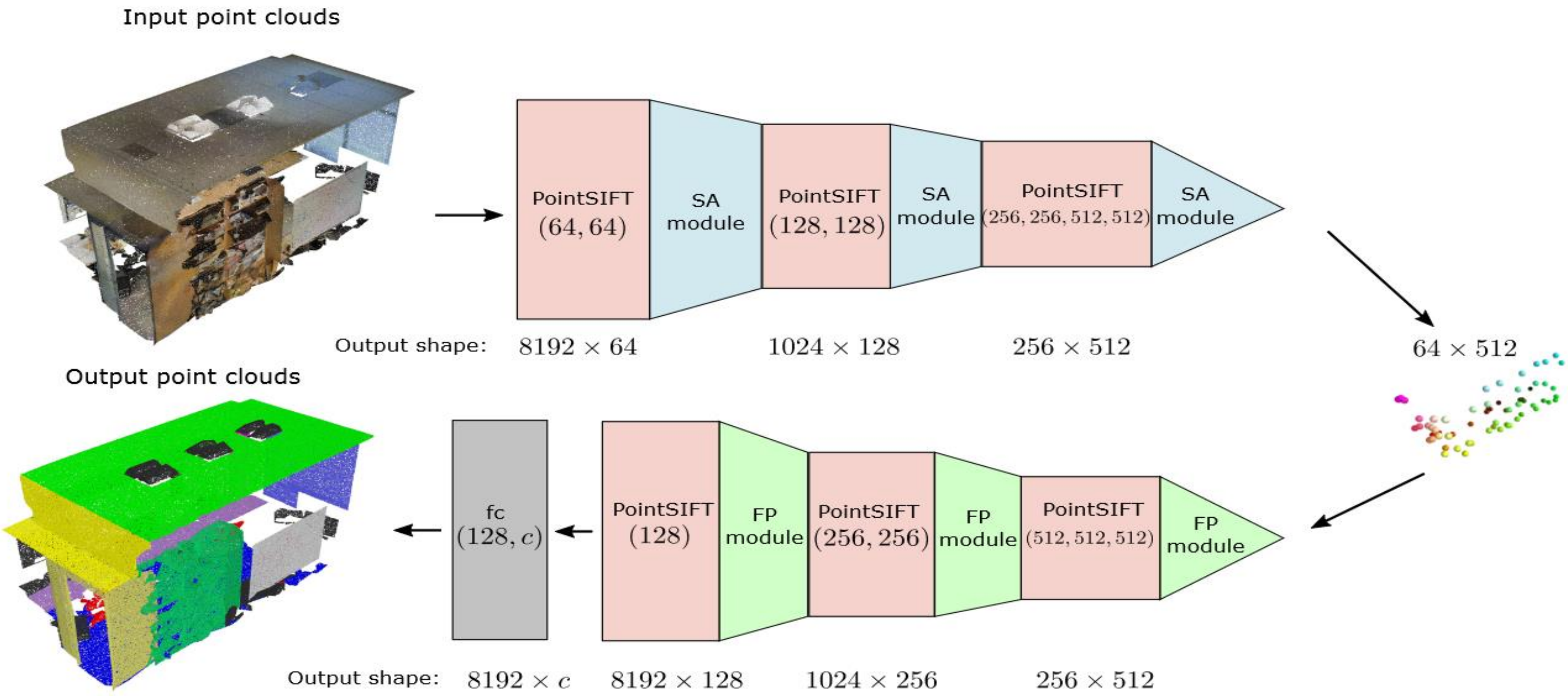
- Scale awareness is achieved by stacking multiple orientation encoding (OE) units
- Finally, the outputs of all the OE units are concatenated and another point wise convolution is performed to obtain the output feature with d-dimensions
- The overall architecture consists of an encoder decoder style structure for sematic segmentation
- The set abstraction step takes an input of N x D the output is N’ x D’ with N > N’ and D < D’
- In the down sampling step N’ centroids are found using farthest point sampling
- Feature Propagation is performed using linear interpolation of nearest neighbor features for upsampling

PointSIFT


Output shape
Input features
$n \times d$
OE unit
$n \times d$
OE unit
$n \times d$
OE unit
$n \times d$
Identity
OE unit
Concatenate
$n \times 3d$
OE unit
OE unit
$n \times d$
Pointwise Convolution
Output features


(a)
(b)
(c)

PointSIFT


Input point clouds
PointSIFT (64, 64)
SA module
PointSIFT (128, 128)
SA module
PointSIFT (256, 256, 512, 512)
SA module
Output shape:
8192 × 64
1024 × 128
256 × 512
64 × 512
Output point clouds
fc (128, c)
PointSIFT (128)
FP module
PointSIFT (256, 256)
FP module
PointSIFT (512, 512, 512)
FP module
Output shape:
8192 × c
8192 × 128
1024 × 256
256 × 512