Lecture 5 – Supplementary

主讲人 Jiaxin Li

Aptiv 自动驾驶
新加坡国立大学 博士
清华大学 本科

Outline

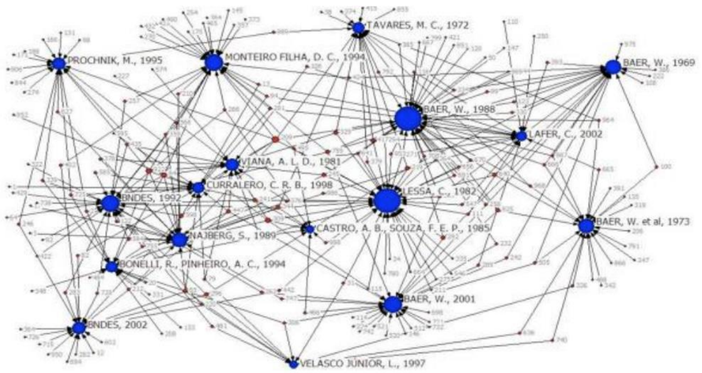
- An instance of GCN for Point Cloud
  - Dynamic Graph CNN for Learning on Point Clouds (DGCNN)

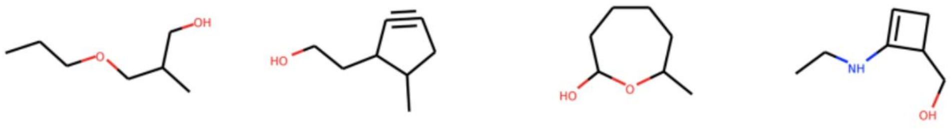
- General GCN
  - Some common GCNs
  - GCN vs. DGCNN

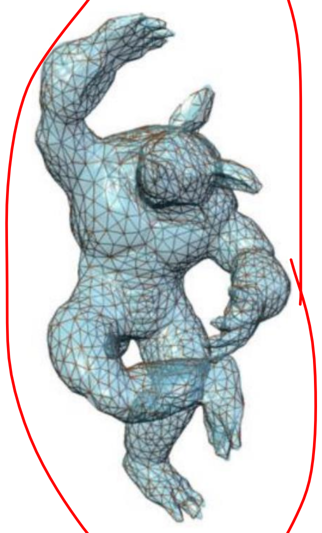
能够写成图的数据表达方式

- Classification vs. Semantic Segmentation

Graph Data







- Social Network
- Citation Network
- Molecules
- **Point Cloud**
- 3D Mesh
- … …

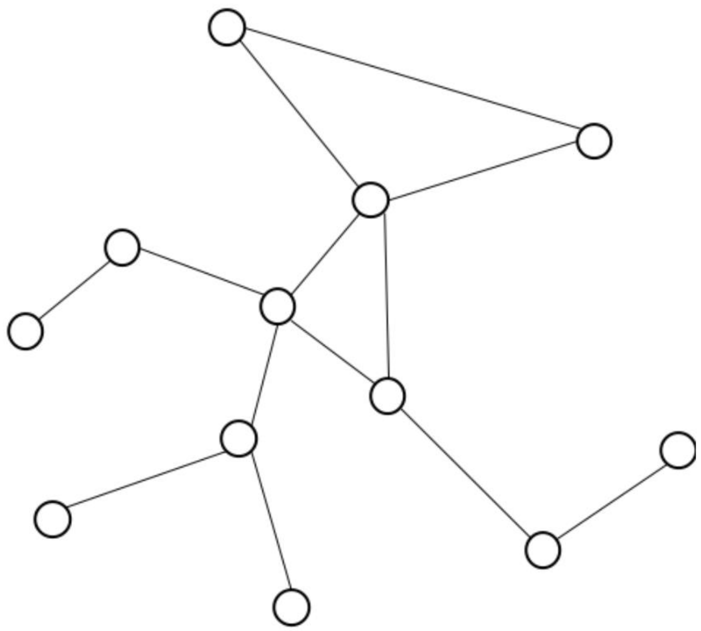
Source: GCN INTRODUCTION AND ITS APPLICATION IN 3D POINT CLOUD SEMANTIC SEGMENTATION, Yisong Li and Guohao Li.

- Graph - $G(V, E)$
  - V: a set of vertices
  - E: a set of edges

- Represent a point cloud by graph?
  - One point – one vertex
  - Edge
    - Fix – by coordinate based kNN/RadiusNN
    - Dynamic – **DGCNN**

每个节点之间怎么连是动态生成的

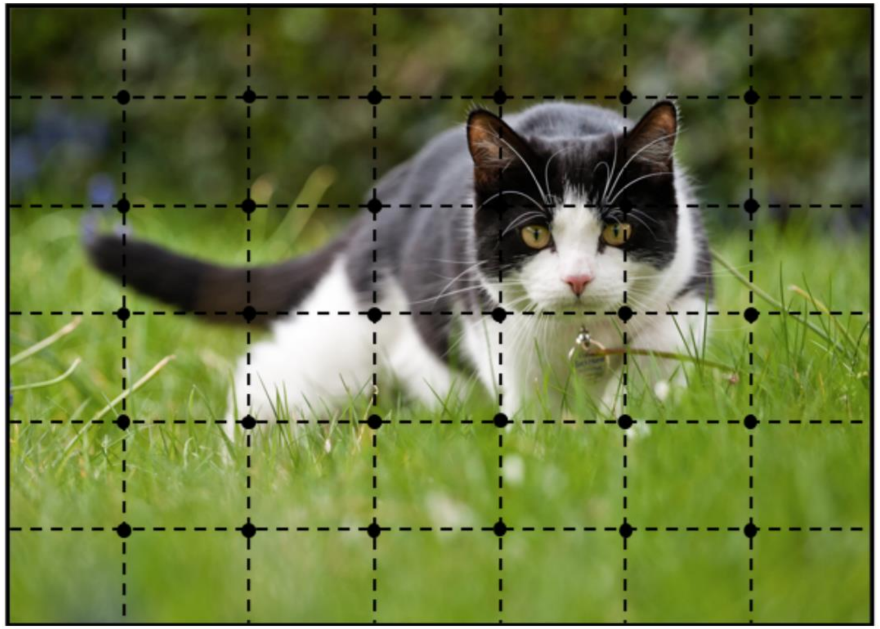
Vertex info:
Coordinate: $\mathbb{R}^3$
Feature: $\mathbb{R}^m$

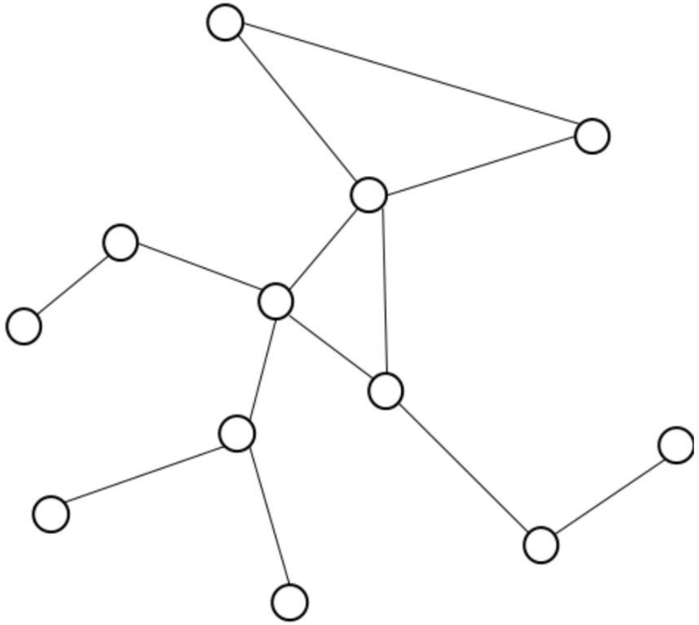


Graph – $G(V, E)$

Image source: Graph Neural Networks: A Review of Methods and Applications, Jie Zhou, et.al.

What is GCN - Convolution on Graph





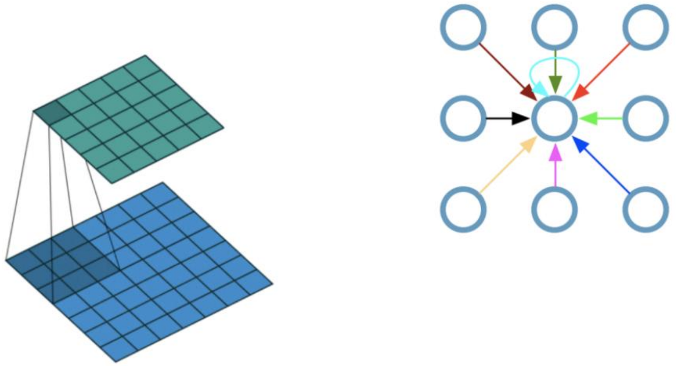
Regular Euclidean Data (Grid)
Normal convolution / pooling,
etc.

Graph in Non-Euclidean Space:
Convolution?
Pooling?

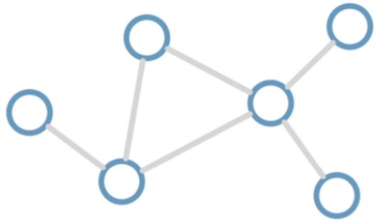
Image source: Graph Neural Networks: A Review of Methods and Applications, Jie Zhou, et.al.

How to Define Convolution on Graph?

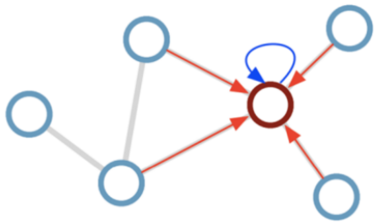
Single CNN layer with 3x3 filter:



Consider this undirected graph:



Calculate update for node in red:



A unified formulation of convolution on images and graphs:

$$x_i' = x_i W_0 + \sum_{j=1}^{k} x_j W_j$$

Trainable parameter $\mathbb{R}^{c \times c'}$

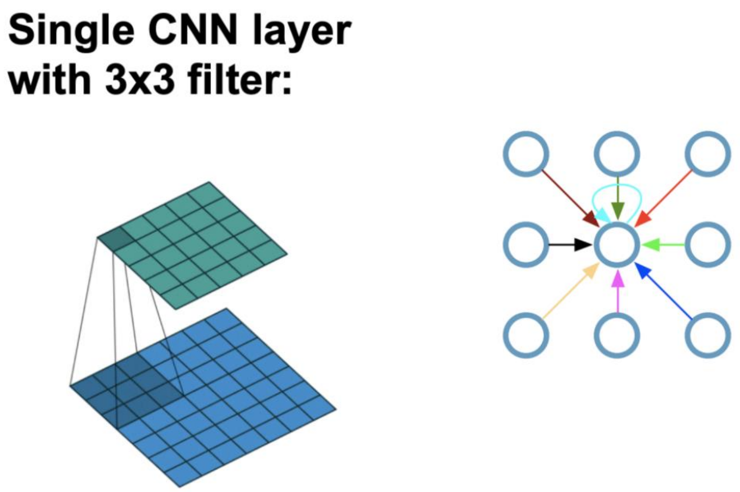
Updated vertex/pixel $\mathbb{R}^{c'}$

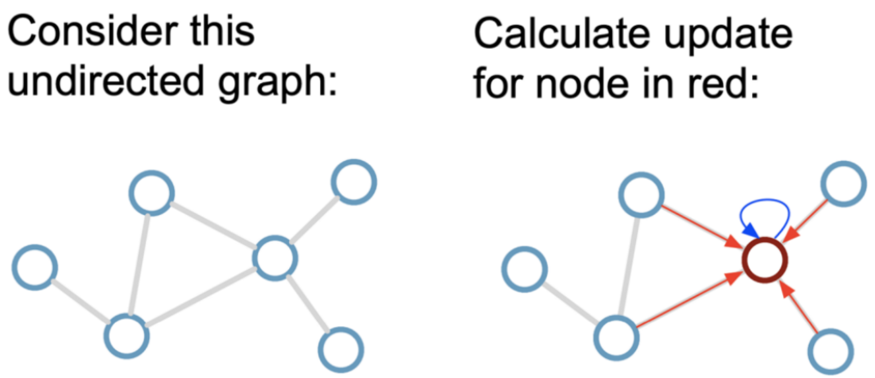
The vertex/pixel we are working on $\mathbb{R}^{c}$

Neighboring vertex/pixel $\mathbb{R}^{c}$

Source: GCN INTRODUCTION AND ITS APPLICATION IN 3D POINT CLOUD SEMANTIC SEGMENTATION, Yisong Li and Guohao Li.

How to Define Convolution on Graph?





- Problem solved? No!
  - Number of neighbor is not fixed on graph.
    - DGCNN for Point Cloud: Fixed neighbor number
  - There are edge weights not utilized.
    - DGCNN for Point Cloud: Assume identical edge weight. (Ignore edge weight)

Source: GCN INTRODUCTION AND ITS APPLICATION IN 3D POINT CLOUD SEMANTIC SEGMENTATION, Yisong Li and Guohao Li.

- EdgeConv
  - Aggregation of neighboring vertex

Aggregation function, e.g., sum, avg, min, max

Edge function (feature transform), with trainable parameter θ

$$\mathbf{x}'_i = \mathop{\square}_{j:(i,j)\in\mathcal{E}} h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j)$$

The vertex we are working on

Vertices that are connected to x_i

Source: DGCNN:Dynamic Graph CNN for Learning on Point Clouds. 2018， Wang Y , Sun Y , Liu Z , et al.

EdgeConv

$$\mathbf{x}'_i = \mathop{\square}_{j:(i,j)\in\mathcal{E}} h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j)$$

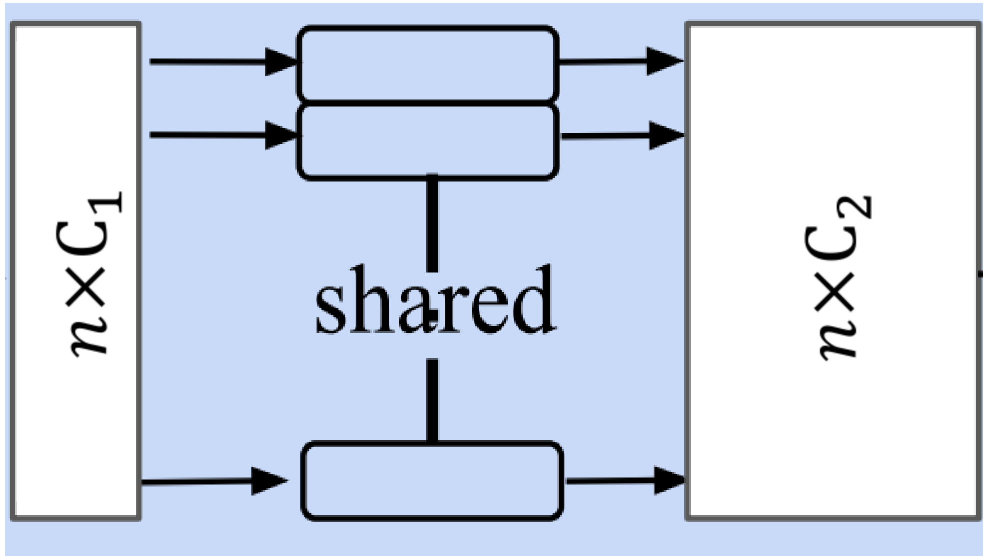
$$h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j) = h_{\Theta}(\mathbf{x}_i)$$

Identity Function

MLP

Point Feature, $\mathbb{R}^m$

PointNet



PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CR Qi et.al.

EdgeConv—PointNet++

EdgeConv

Maxpool

$$\mathbf{x}'_i = \mathop{\square}_{j:(i,j)\in\mathcal{E}} h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j)$$

$$h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j) = h_{\Theta}(\mathbf{x}_j)$$

MLP

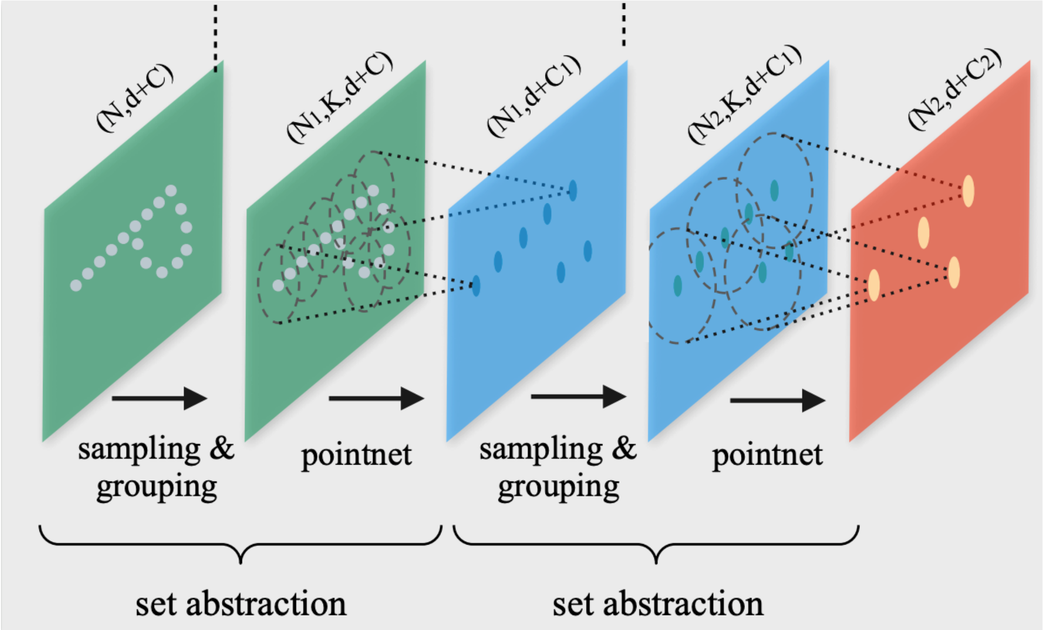
Features of dimension C, C_1, C_2

$$h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j) = h_{\Theta}(\mathbf{x}_j - \mathbf{x}_i)$$

MLP

Coordinate of dimension d = 3

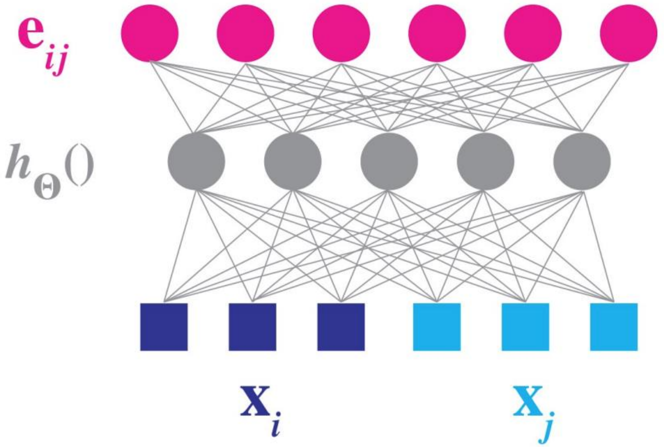
PointNet++

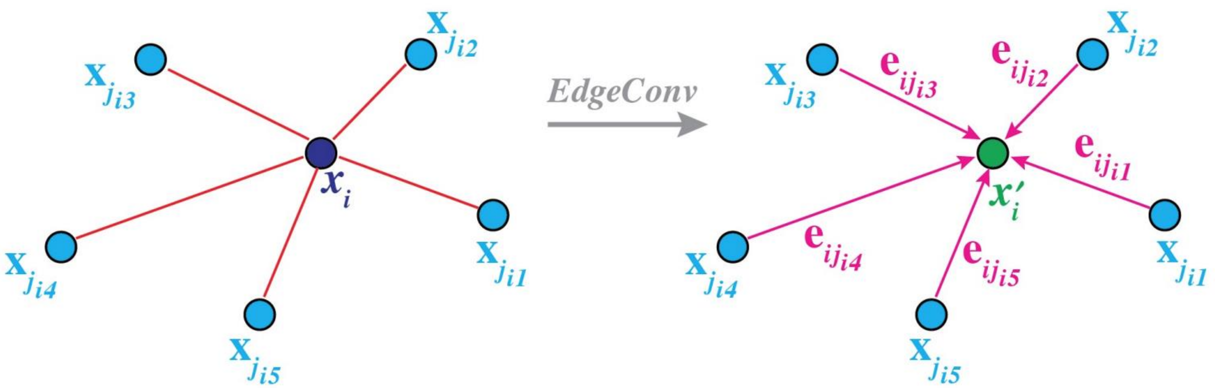


PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space

EdgeConv—DGCNN

$$\mathbf{x}'_i = \mathop{\square}_{j:(i,j)\in\mathcal{E}} h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j)$$





MLP with trainable param Θ

MLP with trainable param ϕ

$$h_{\Theta}(x_i, x_j) = h_{\Theta}(x_j - x_i) + h_{\phi}(x_i)$$

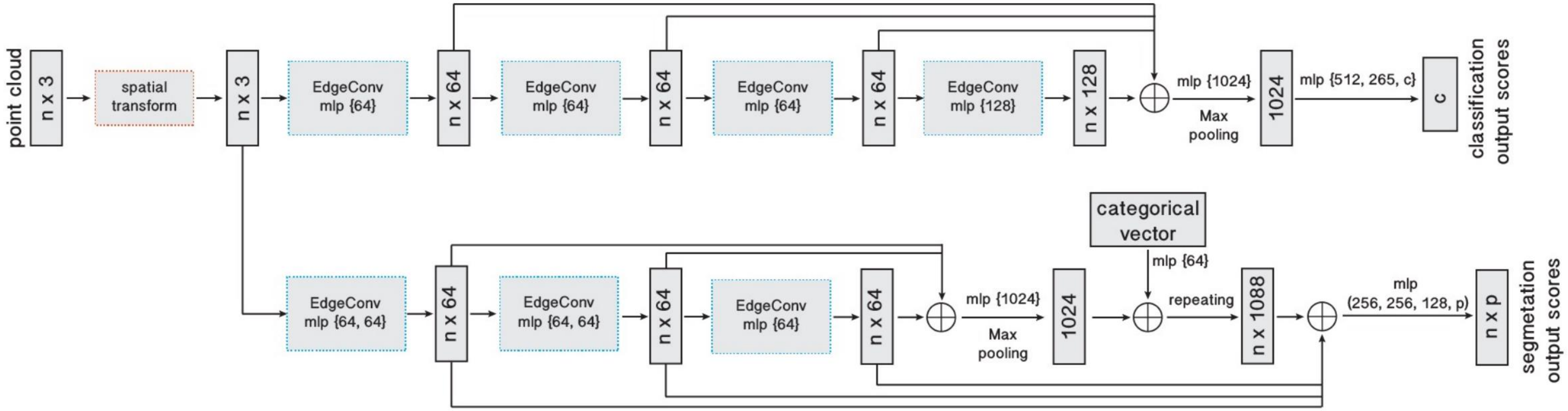
$$x'_i = maxpool_j\left(h_{\Theta}(x_i, x_j)\right)$$

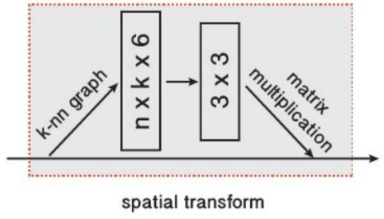
Neighboring vertex, $\mathbb{R}^m$

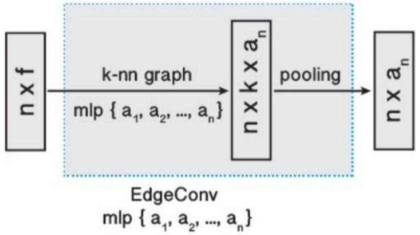
The vertex we are working on, $\mathbb{R}^m$

Source: DGCNN:Dynamic Graph CNN for Learning on Point Clouds. 2018， Wang Y , Sun Y , Liu Z , et al.

DGCNN





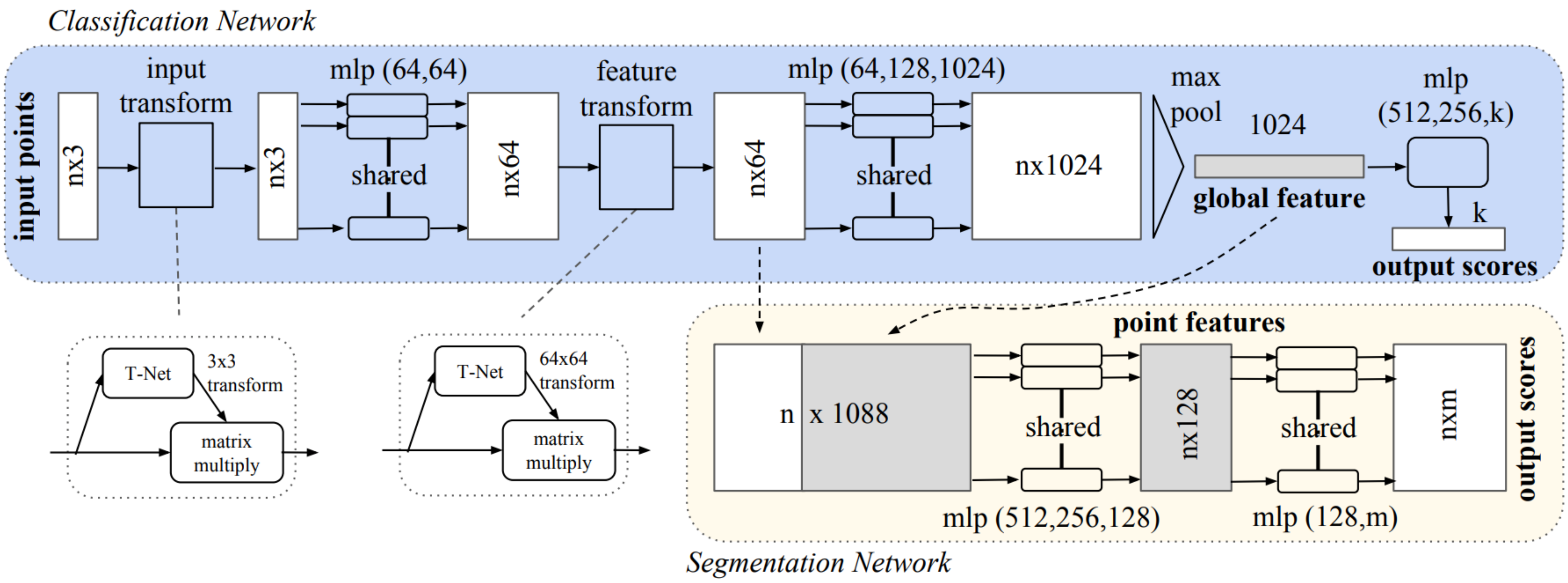


- Similar to T-Net of PointNet,
- NOT presented in the author's open-source code

n points, each is $x_i' = maxpool\left(h_\Theta\left(x_j - x_i\right) + h_\phi(x_i)\right)$

Source: DGCNN:Dynamic Graph CNN for Learning on Point Clouds. 2018， Wang Y , Sun Y , Liu Z , et al.

PointNet



PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CR Qi et.al.

DGCNN—Dynamic Graph

Aggregation function, e.g., sum, avg, min, max

Edge function (feature transform), with trainable parameter θ

$$\mathbf{x}'_i = \mathop{\square}_{j:(i,j)\in\mathcal{E}} h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j)$$

The vertex we are working on

Vertices that are connected to x_i

- How to define "neighbor x_j"?
  - For first EdgeConv – kNN on coordinates ($x_i \in \mathbb{R}^3$)
  - For subsequent EdgeConv – kNN on features ($x'_i \in \mathbb{R}^m$)
    - Graph is dynamic, not static.

在特征空间里，学习连接

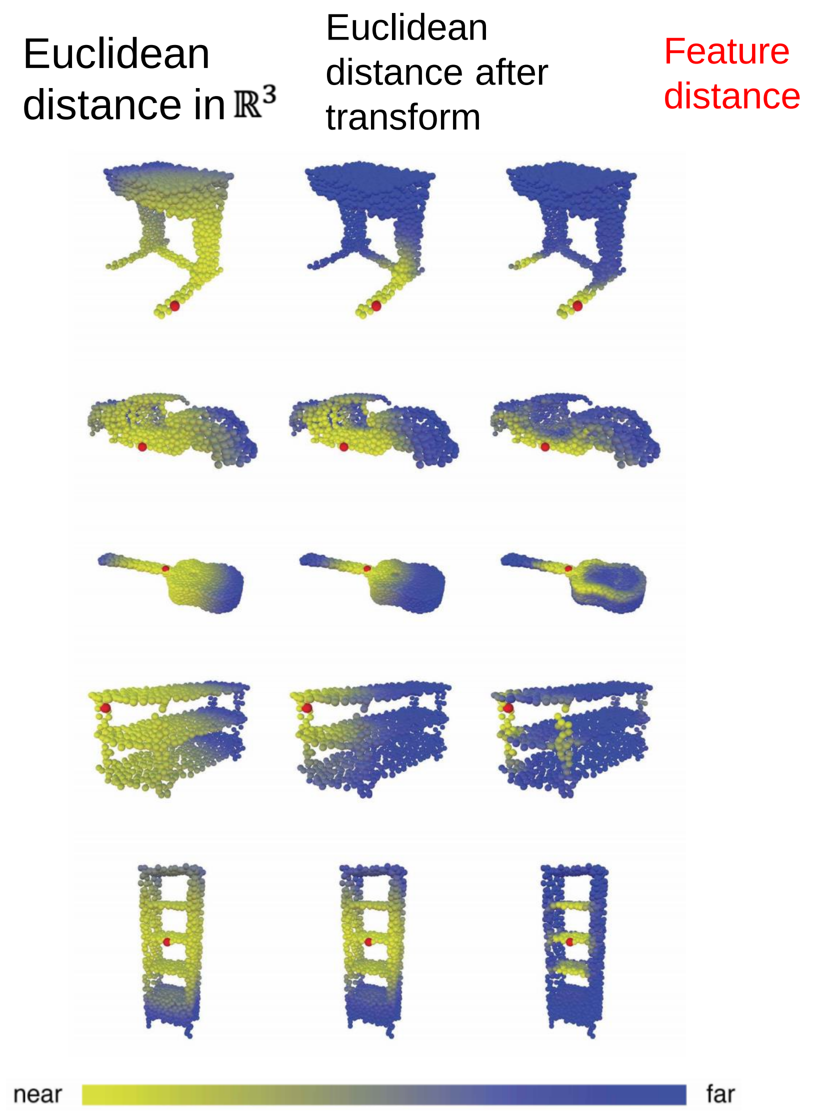
动态建图，每个点携带坐标信息和特征信息

Source: DGCNN:Dynamic Graph CNN for Learning on Point Clouds. 2018， Wang Y , Sun Y , Liu Z , et al.

DGCNN—Dynamic Graph

- Left
  - Euclidean distance in $\mathbb{R}^3$
- Middle
  - Euclidean distance in $\mathbb{R}^3$
  - After 3×3 matrix transform
- Right
  - Feature distance in $\mathbb{R}^3$
  - From the last layer
  - Semantic & dynamic

挑选谁跟谁更近



Source: DGCNN:Dynamic Graph CNN for Learning on Point Clouds. 2018， Wang Y , Sun Y , Liu Z , et al.

Classification on ModelNet40 – Outperforms PointNet++

	Model size(MB)	Time(ms)	Accuracy(%)
PointNet (Baseline) [Qi et al. 2017b]	9.4	6.8	87.1
PointNet [Qi et al. 2017b]	40	16.6	89.2
PointNet++ [Qi et al. 2017c]	12	163.2	90.7
PCNN [Atzmon et al. 2018]	94	117.0	92.3
Ours (Baseline)	11	19.7	91.7
Ours	21	27.2	92.9

Table 3. Complexity, forward time, and accuracy of different models

Number of nearest neighbors (k)	Mean Class Accuracy(%)	Overall Accuracy(%)
5	88.0	90.5
10	88.9	91.4
20	90.2	92.9
40	89.4	92.4

Table 5. Results of our model with different numbers of nearest neighbors.

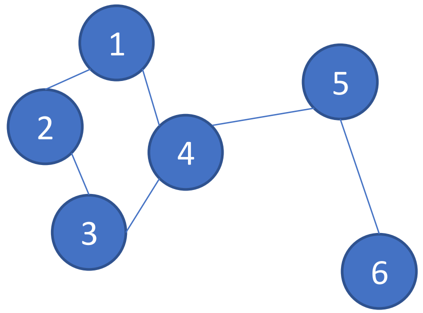
Source: DGCNN:Dynamic Graph CNN for Learning on Point Clouds. 2018， Wang Y , Sun Y , Liu Z , et al.

- Graph structure is represented by **Similarity Matrix $A \in \mathbb{R}^{n \times n}$**
  - Un-connected: 0
  - Connected: similarity score, e.g., 1

- Concepts derived from A (Please refer to Lecture 4)
  - Degree matrix $D \in \mathbb{R}^{n \times n}$
  - Laplacian matrix $L = D - A \in \mathbb{R}^{n \times n}$
  - Normalized Laplacian matrix
    $L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

- Why do we need these A, D, L, L_{sym}?

Graph



Connectivity / Similarity matrix A

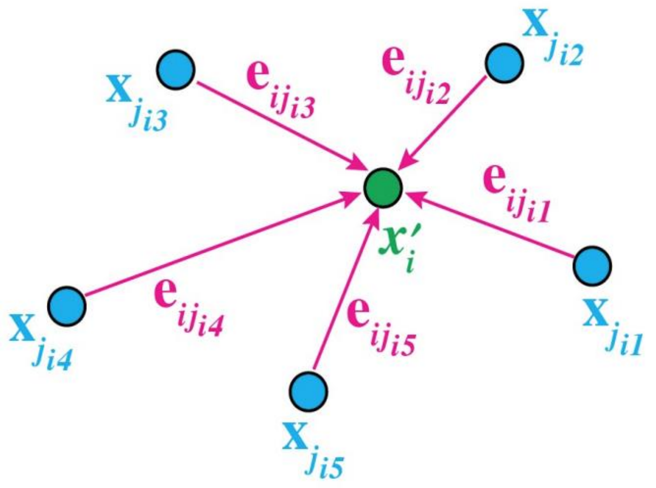
$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Degree Matrix D

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

General GCN

- Input: $X \in \mathbb{R}^{n \times C_{in}}$
- Output: $H \in \mathbb{R}^{n \times C_{out}}$



- A GCN layer consists of two steps
  - Aggregation
    - Gather features from neighbors
  - Update MLP
    - Apply learnable layer to transform the aggregated features

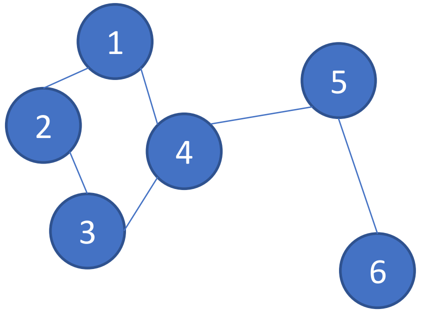
Image Source: DGCNN:Dynamic Graph CNN for Learning on Point Clouds. 2018， Wang Y , Sun Y , Liu Z , et al.

General GCN—Similarity Matrix

- Input: $X \in \mathbb{R}^{n \times C_{in}}$
- Output: $H \in \mathbb{R}^{n \times C_{out}}$

- **Aggregation**
  - Weighted sum of neighbors
  - $N = \underline{A} \cdot X$
- **Update**
  - Linear (Fully Connected) Layer with learnable param $W \in \mathbb{R}^{C_{in} \times C_{out}}$
  - Activation function σ, e.g, ReLU
  - $H = \sigma(N \cdot W) = \sigma(AXW)$

Graph



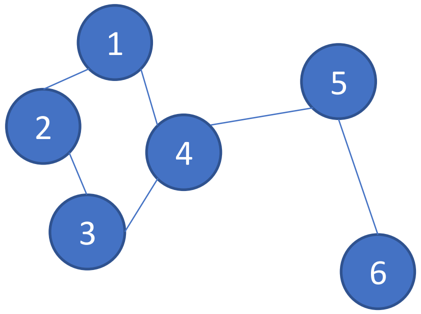
Connectivity / Similarity matrix A

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- $H = \sigma(N \cdot W) = \sigma(AXW)$

- Problem:
  - Each node itself is not considered
    - Diagonal elements of A is 0
    - Solution: Laplacian matrix
  - Naïve weighted sum puts more weights on nodes with more connections
    当节点的邻接点特别多时
    - A is not normalized
    - Solution: Normalized Laplacian matrix

Graph



Connectivity / Similarity matrix A

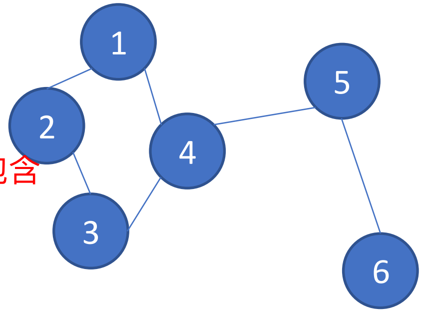
$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

General GCN—Lapacian Matrix

- Input: $X \in \mathbb{R}^{n \times C_{in}}$
- Output: $H \in \mathbb{R}^{n \times C_{out}}$

- Aggregation
  - Weighted sum of **neighbors and itself**
  - $N = (D - A) \cdot X = LX$ laplacican matrix 包含了自己也包含了邻居的信息
  - In vector form: $N_i = \sum_j \left(A_{ij}(X_i - X_j)\right)$
- Update
  - **Linear (Fully Connected) Layer** with learnable param $W \in \mathbb{R}^{C_{in} \times C_{out}}$
  - Activation function σ, e.g, ReLU
  - $H = \sigma(N \cdot W) = \sigma(LXW)$

Graph



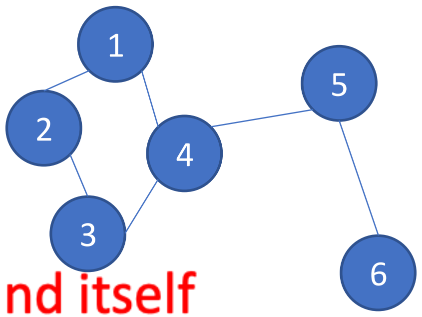
Laplacian Matrix L

$$\begin{bmatrix} 2 & -1 & 0 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ -1 & 0 & -1 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

General GCN—Normalized Lapacian Matrix

- Input: $X \in \mathbb{R}^{n \times C_{in}}$
- Output: $H \in \mathbb{R}^{n \times C_{out}}$

Graph



Normalized Laplacian Matrix L_{sym}

$$\begin{bmatrix} 1 & -\frac{1}{\sqrt{2}\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}\sqrt{3}} & 0 & 0 \\ -\frac{1}{\sqrt{2}\sqrt{2}} & 1 & -\frac{1}{\sqrt{2}\sqrt{2}} & 0 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}\sqrt{2}} & 1 & -\frac{1}{\sqrt{2}\sqrt{3}} & 0 & 0 \\ -\frac{1}{\sqrt{2}\sqrt{3}} & 0 & -\frac{1}{\sqrt{2}\sqrt{3}} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\frac{1}{\sqrt{2}\sqrt{1}} \\ 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}\sqrt{1}} & 1 \end{bmatrix}$$

- Aggregation
  - Normalized weighted sum of neighbors and itself
  - $N = D^{-1/2} L D^{-1/2} \cdot X = L_{sym} X$
  - In vector form: $N_i = \sum_j \left(A_{ij} (X_i - X_j) \cdot \frac{1}{\sqrt{D_{ii} D_{jj}}} \right)$

归一化，考虑了每个节点的degree

- Update
  - Linear (Fully Connected) Layer with learnable param $W \in \mathbb{R}^{C_{in} \times C_{out}}$
  - Activation function σ, e.g, ReLU
  - $H = \sigma(N \cdot W) = \sigma(LXW)$

DGCNN vs. General GCN

- DGCNN
  - For each i
  - MLP on neighbors → get multiple feature vectors
  - Maxpool → get final feature vector for i

$n \times k \times C_{in} \rightarrow n \times k \times C_{out}$

$\downarrow$

$n \times C_{out}$

MLP with param Θ

MLP with param ϕ

$$h_\Theta(x_i, x_j) = h_\Theta(x_j - x_i) + h_\phi(x_i)$$

$$x_i' = maxpool_j\left(h_\Theta(x_i, x_j)\right)$$

Neighboring vertex, $\mathbb{R}^m$

The vertex we are working on, $\mathbb{R}^m$

- General GCN
  - For each i
  - Aggregation: Weighted sum of neighbors → get one feature vector
  - Update: MLP on the feature vector → get final feature vector for i

$n \times C_{in}$

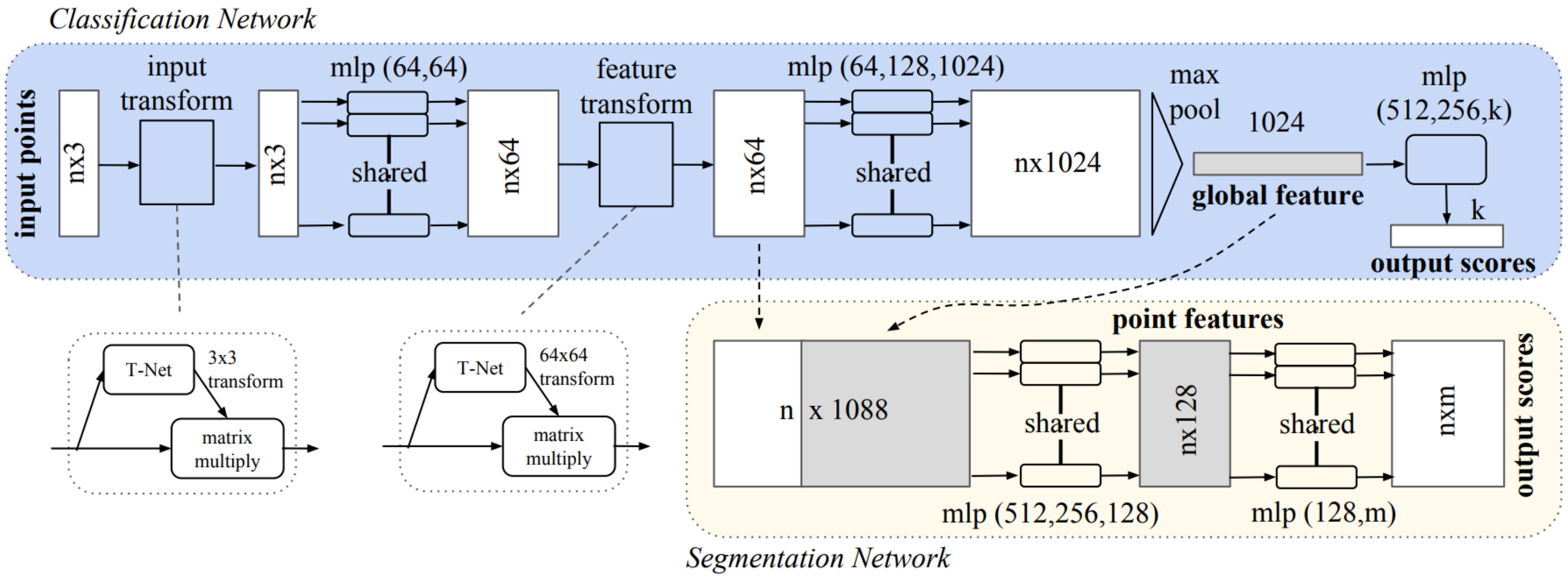
$n \times C_{out}$

Some Other Comments On Deep Learning

Classification vs. Semantic Segmentation

- Similarity
  - Both of them are classification by Softmax

- Difference
  - Classification – Per Object. Segmentation – Per Point
  - Network design
    - Classification: Global feature
    - Segmentation: Global + Local feature



PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CR Qi et.al.

- **Want practical tricks/experience?**
  - Do it yourself
    - Dataset preparation
    - Design, write, train & tune your own network
    - Don't be scared. DL is simple for implementation.
    - You have to experience it to get "experience"
  - Develop your ability to invent "tricks"
    - You won't get a good job by saying you know lots of tricks
- **Common methods?**
  - Core ideas are valid for a long time
    - PointNet series, voxel grid, data augmentation…
  - Methods/Networks are different every company, every year.

- **Some critical topics in industry**
  - Data/repo management
    - Coding ability
  - Runtime
    - TensorRT
    - Hardware-aware network design
    - Network Pruning, Quantization, etc.
  - Performance optimization
    - Active learning, e.g., data mining
    - Network Architecture Search (NAS)