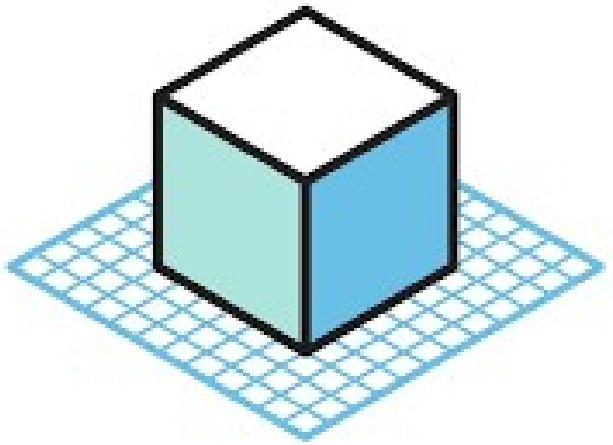
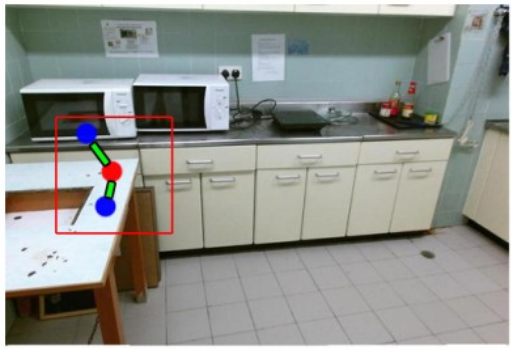
# Deep Learning Models for Processing 3D Data

Narada Warakagoda

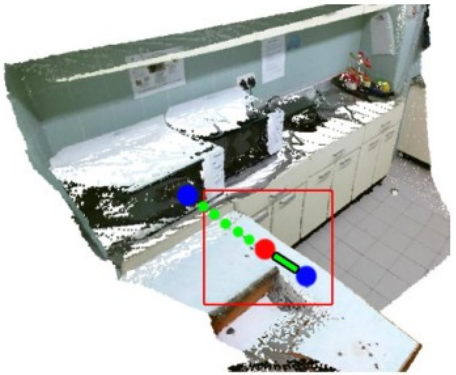


# Why 3D?

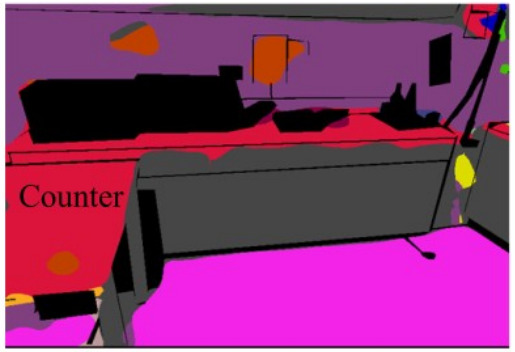
Real world applications of autonomous systems need to perceive depth information



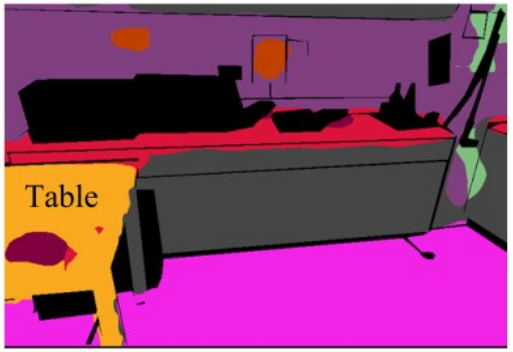
(a). 2D Image



(b) 3D Point Cloud




(c) 2D Prediction

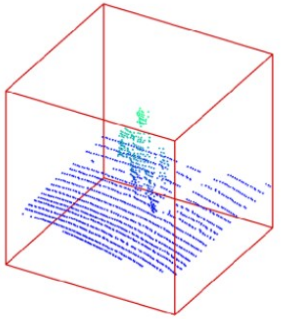


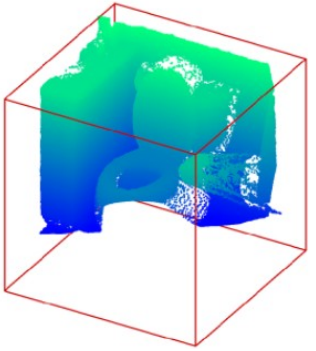

(d) Ours (3D Prediction)

3D Graph Neural Networks for RGBD Semantic Segmentation, Qi et. al

# Common 3D Sensor Outputs

- Point Cloud
  - Set of points $\{(x, y, z, a_1, a_2, \cdots, a_n)\}$
    - $x, y, z$ co-ordinates of the point
    - $a_1, a_2, \cdots, a_n$ are attributes of the point
  - Examples:
    - LiDAR
    - Camera images with depth (RGBD)





- Multi-view images
  - Stereo camera images

# 3D Processing Approaches

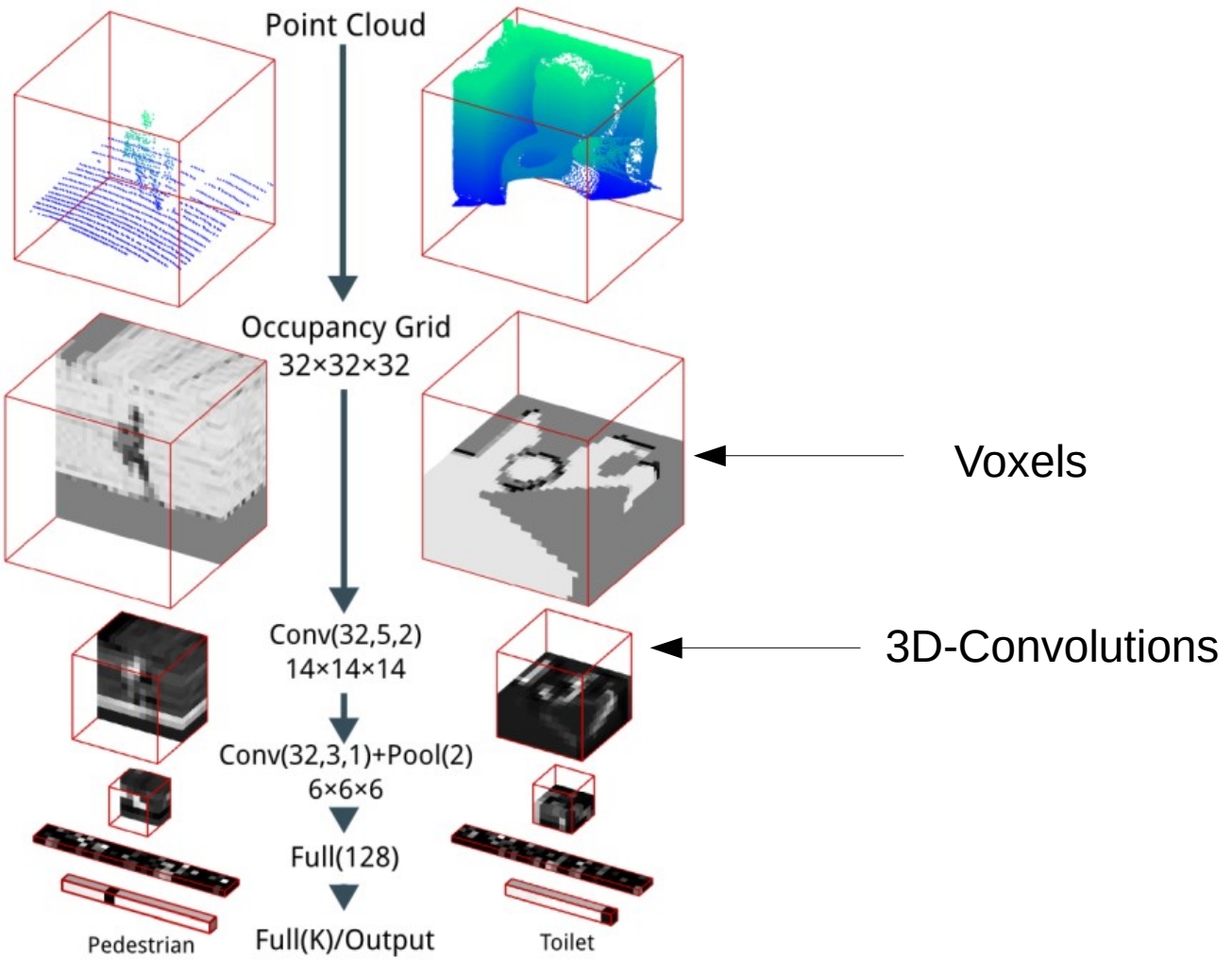
- Use point clouds directly as input to a deep network
  - Not straight-forward as deep networks are designed for regular grid inputs (eg: pixelized images)
- Convert the point cloud to a grid-like structure and input to a deep network
  - Voxel inputs
  - 3D convolutions
- Represent the point cloud as a graph and input to a suitable neural network
  - Graph neural networks

2

1

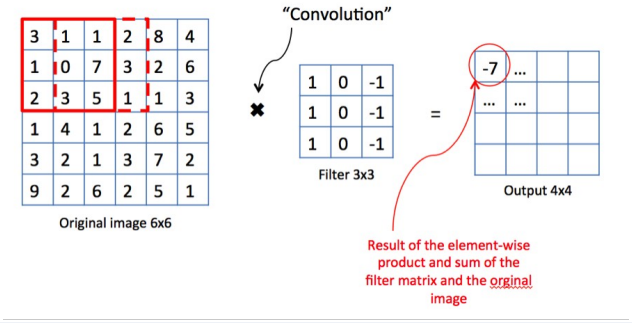
3

# Learning with Voxel Inputs

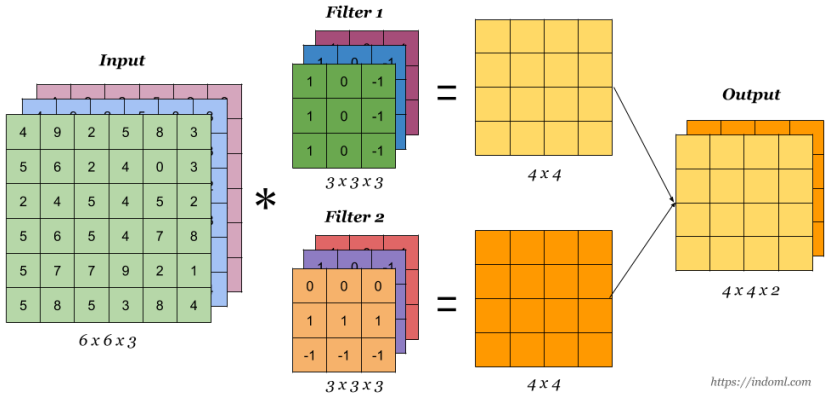



VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition, Maturana and Scherer

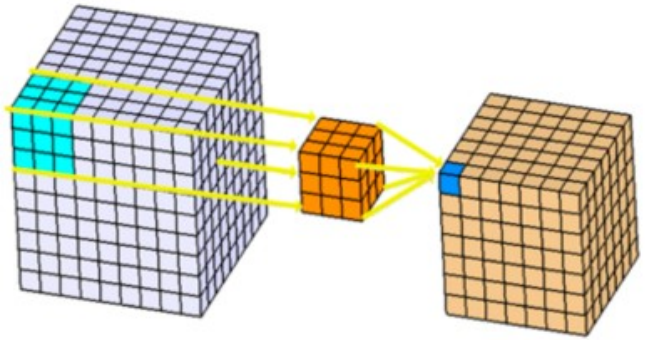
# Convolutions




2D




2.5D



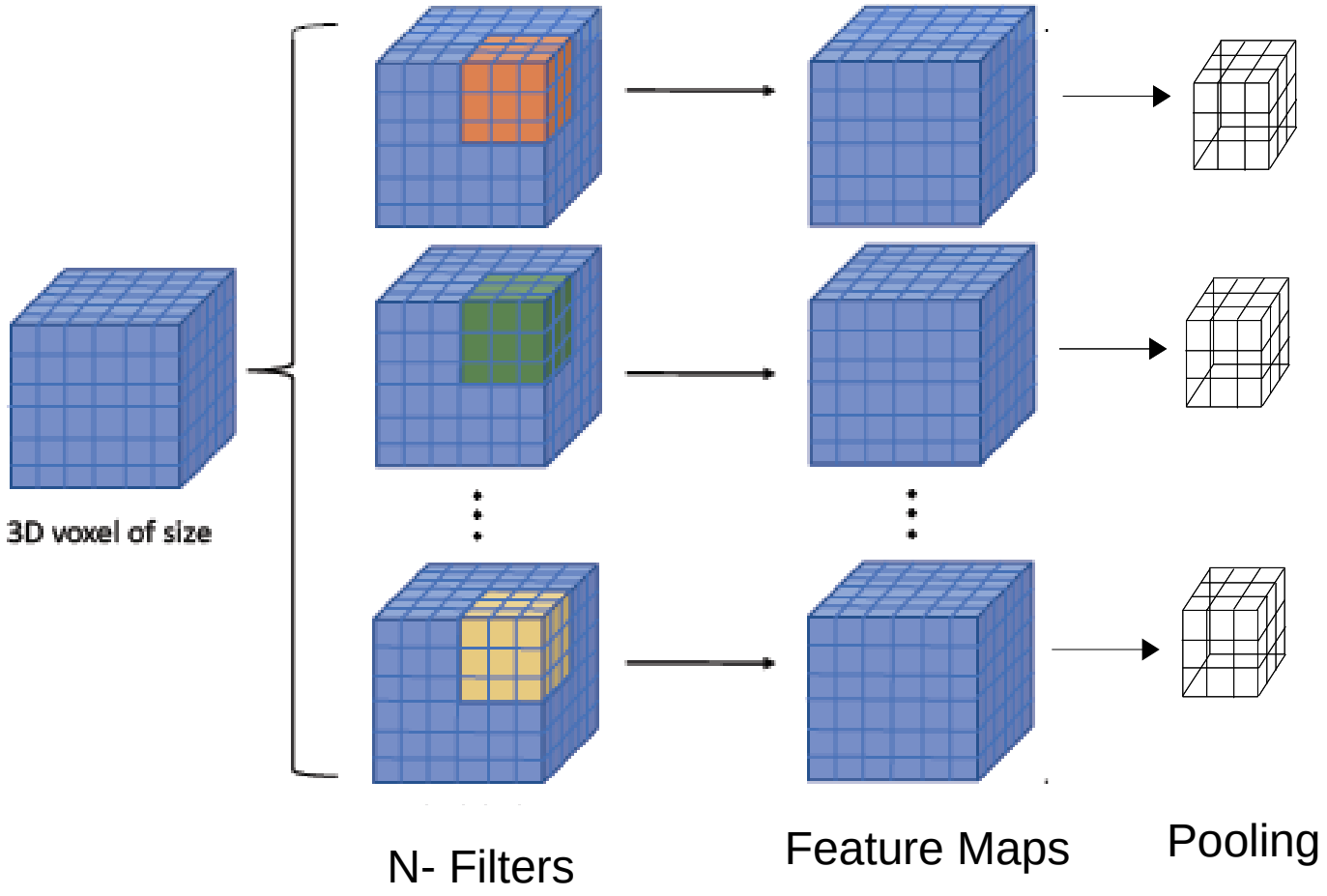
3D

Michele Cavaioni https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524

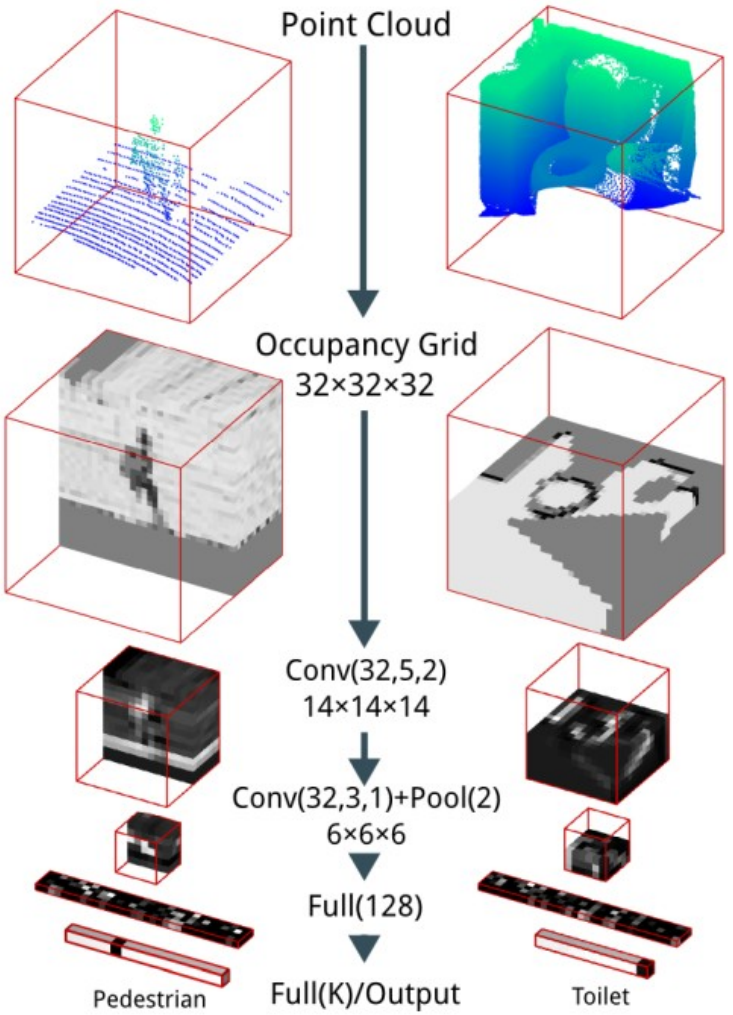
https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

https://www.kaggle.com/shivamb/3d-convolutions-understanding-use-case

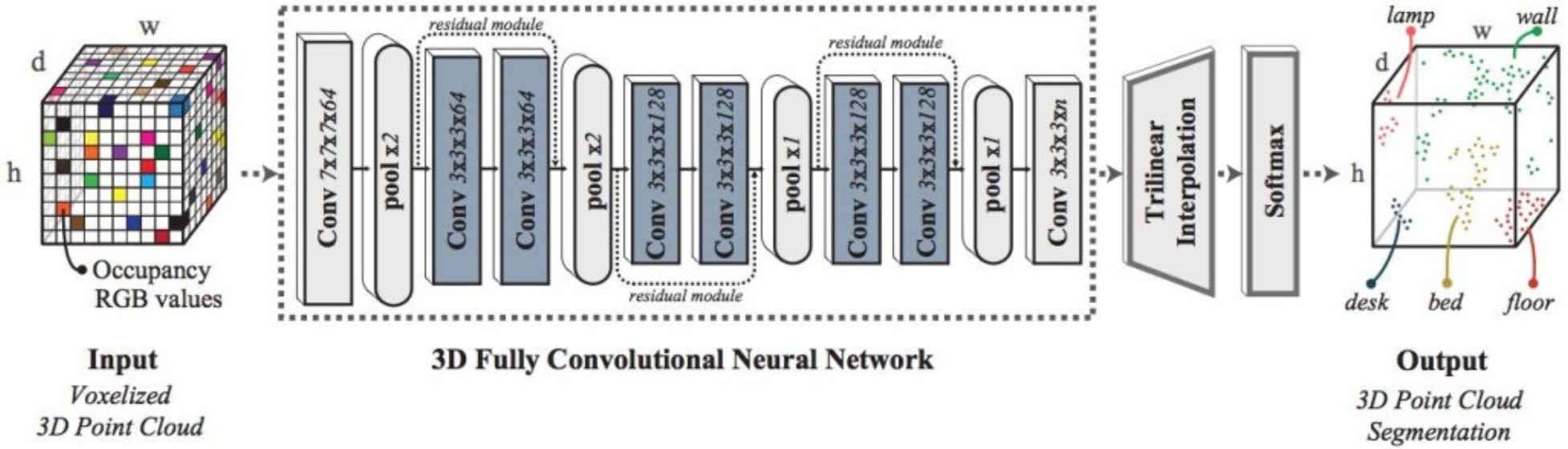
# 3D Convolutional Network



# Object Classification



# 3D Segmentation

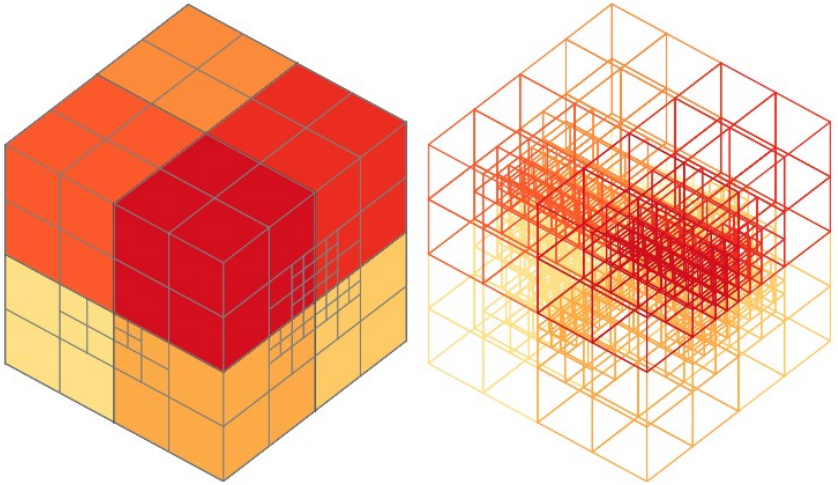


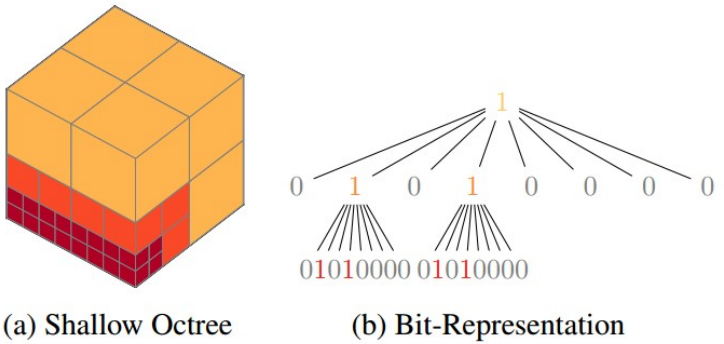

SEGCloud: Semantic Segmentation of 3D Point Clouds, Tchapmi et.al

# Problems of Voxelization

- High memory consumption
  - Affects the possible resolution
- Most of the space is empty (zero voxels)
  - Inefficient use of resources
- More memory efficient voxelization solutions exist:
  - Eg: OctNets

# OctNet






(a) Shallow Octree (b) Bit-Representation

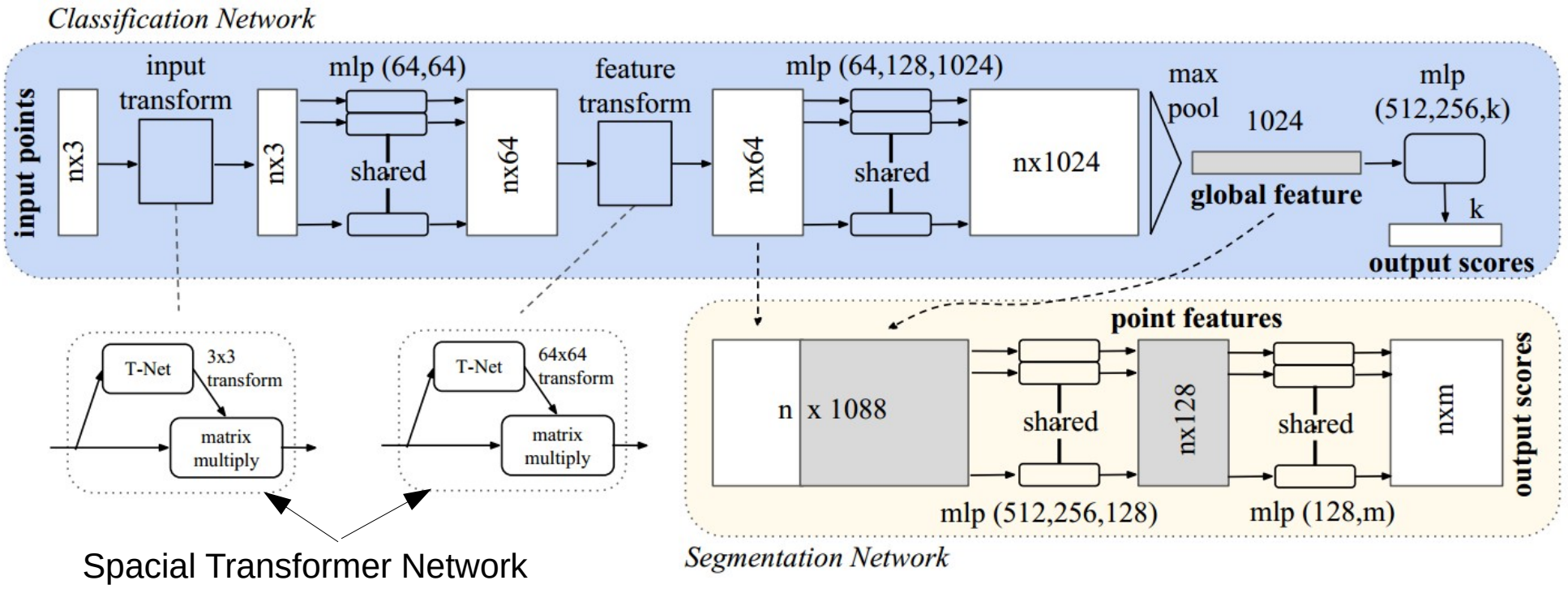
- Create an irregular grid
  - Iterative split voxels into 8 child voxels
  - Limits to depth 3
- Split only important areas
  - Areas where objects lie (high resolution)
  - Do  not split empty areas (low resolution)
- Memory and speed savings,

OctNet: Learning Deep 3D Representations at High Resolutions, Riegler et. al

# Direct Point Cloud Input

- Challenges of direct point cloud processing
  - Point clouds can be unordered  (it is a set), therefore we need permutation invariance
  - Need to model interaction among points
  - Need to model transformation invariance (eg: rotation, translation)
- PointNet addresses those challenges

# PointNet

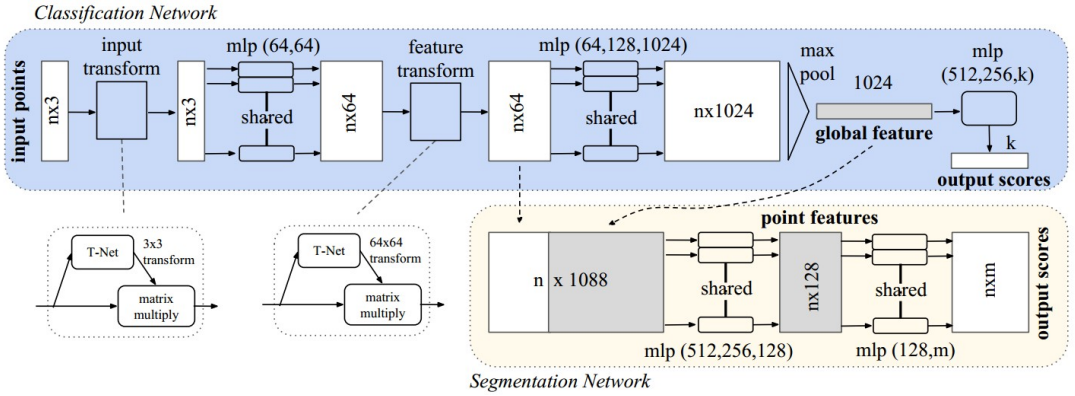



- Can be used for classification and segmentation

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

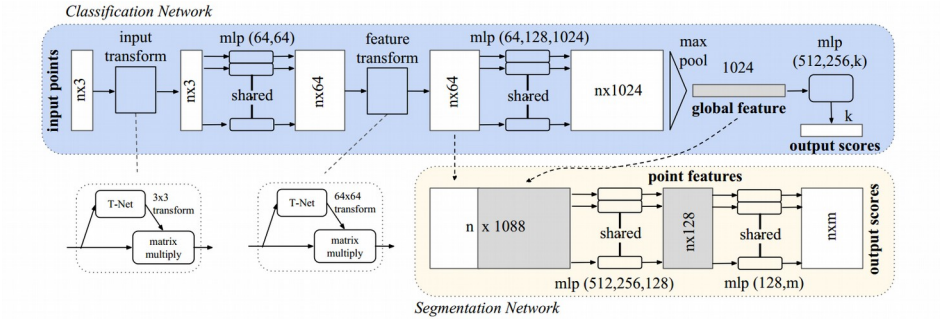
# PointNet Permutation Invariance

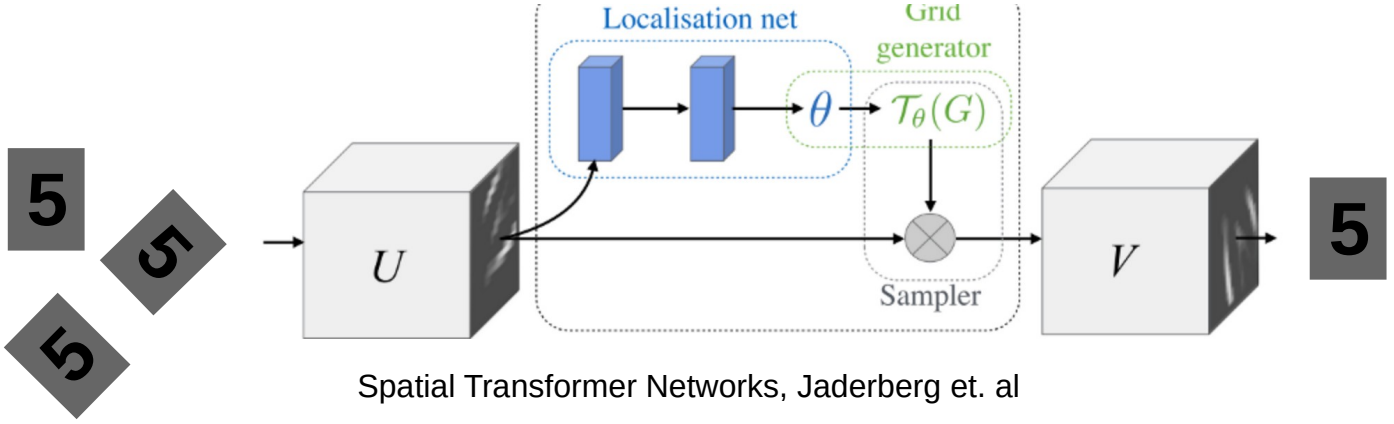
- Network should respond equally to different input orders of the point cloud of the same scene
  - $\{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)\}$
  - $\{(x_3, y_3, z_3), (x_1, y_1, z_1), (x_2, y_2, z_2)\}$
- Solution:
  - Process each point individually with the same transform (MLP)
  - Finally apply a symmetric function (eg: max-pooling)



# PointNet Transformation Invariance

- Network should respond equally to transformed (rotated, shifted) version of the same point cloud subset
- Solution:
  - Employ the Spacial Transformer Network




Spatial Transformer Networks, Jaderberg et. al

# Interaction among Local Points

- PointNet does not address this issue
- Despite this, PointNet works fairly well

| | input | #views | accuracy avg. class | accuracy overall |
|---|---|---|---|---|
| SPH [11] | mesh | - | 68.2 | - |
| 3DShapeNets [25] | volume | 1 | 77.3 | 84.7 |
| VoxNet [15] | volume | 12 | 83.0 | 85.9 |
| Subvolume [16] | volume | 20 | 86.0 | **89.2** |
| LFD [25] | image | 10 | 75.5 | - |
| MVCNN [20] | image | 80 | **90.1** | - |
| Ours baseline | point | - | 72.6 | 77.4 |
| Ours PointNet | point | 1 | 86.2 | **89.2** |

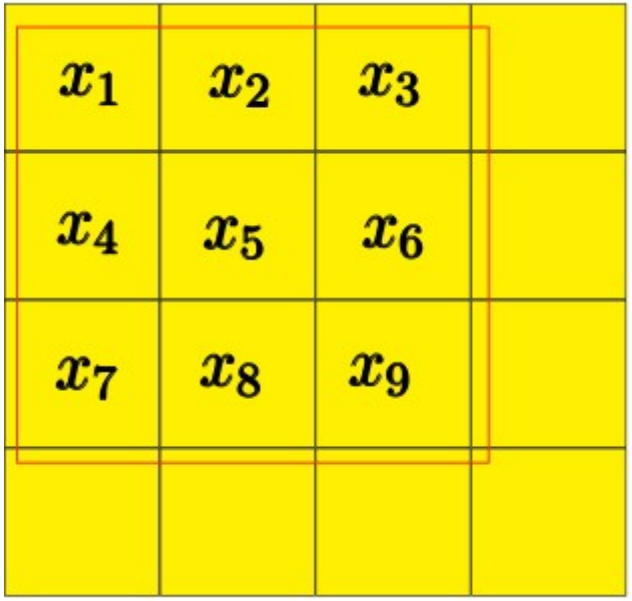
Table 1. **Classification results on ModelNet40.** Our net achieves state-of-the-art among deep nets on 3D input.

# Graph Neural Networks

- Aim to exploit locality
- Relationships among local points in a neighborhood carries important information
- Locality in point clouds is represented as a graph
- Take the inspiration from regular convolution in CNNs
- Extend regular convolution to graph convolution and CNN to GNN (Graph Neural Network)

# Convolution as a Operation on a Graph

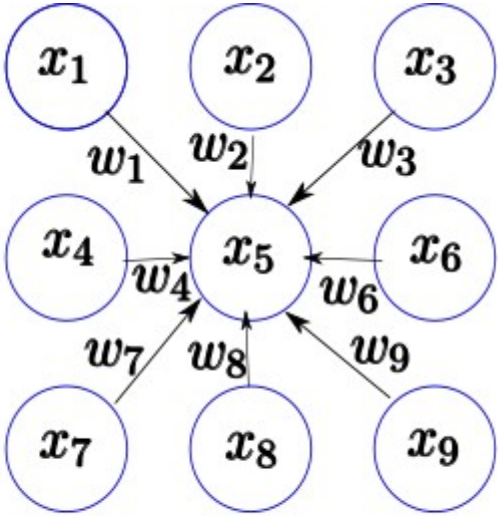
Image




Filter

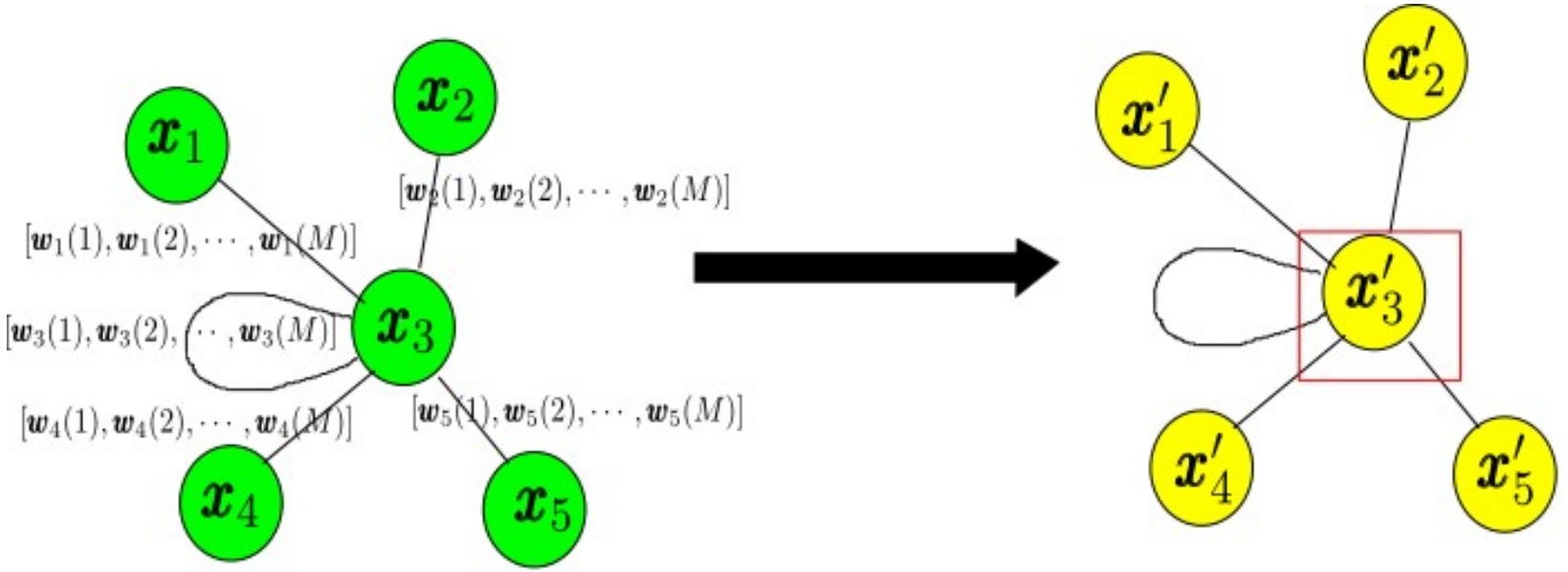
| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

Graph Representation of the grid



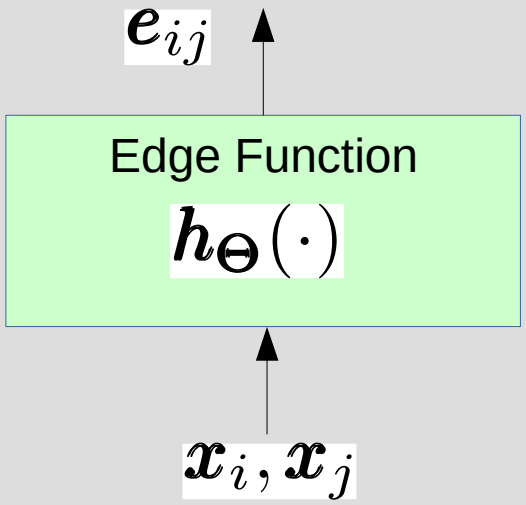

$$x_5' = \sum_{i=1}^{9} x_i w_i$$

# Generalized Convolution on a Graph

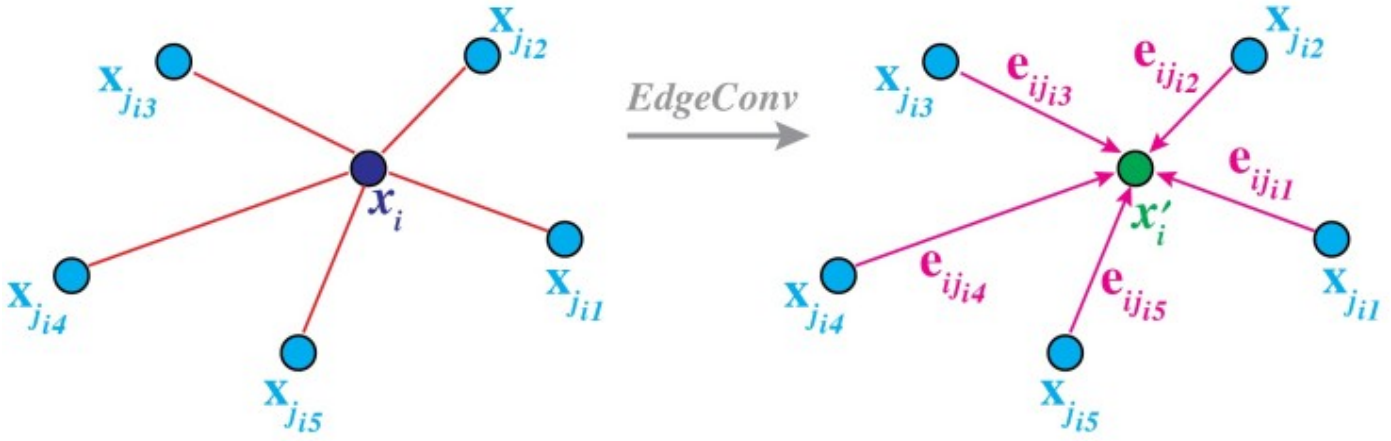



$$x'_3(1) = \boldsymbol{w}_1^T(1)\boldsymbol{x}_1 + \boldsymbol{w}_2^T(1)\boldsymbol{x}_2 + \boldsymbol{w}_3^T(1)\boldsymbol{x}_3 + \boldsymbol{w}_4^T(1)\boldsymbol{x}_4 + \boldsymbol{w}_5^T(1)\boldsymbol{x}_5$$
$$x'_3(2) = \boldsymbol{w}_1^T(2)\boldsymbol{x}_1 + \boldsymbol{w}_2^T(2)\boldsymbol{x}_2 + \boldsymbol{w}_3^T(2)\boldsymbol{x}_3 + \boldsymbol{w}_4^T(2)\boldsymbol{x}_4 + \boldsymbol{w}_5^T(2)\boldsymbol{x}_5$$
$$\vdots$$
$$x'_3(M) = \boldsymbol{w}_1^T(M)\boldsymbol{x}_1 + \boldsymbol{w}_2^T(M)\boldsymbol{x}_2 + \boldsymbol{w}_3^T(M)\boldsymbol{x}_3 + \boldsymbol{w}_4^T(M)\boldsymbol{x}_4 + \boldsymbol{w}_5^T(M)\boldsymbol{x}_5$$

# Even More Generalized Convolution Operation on a Graph




$$e_{ij} = h_\Theta(\mathbf{x}_i, \mathbf{x}_j), \text{ where } h_\Theta : \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}^{F'}$$




Aggregation operation, eg: SUM or MAX

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

- Different architectures for different $h_\Theta(\cdot)$ and aggregation operations

Dynamic Graph CNN for Learning on Point Clouds, Wang et. al

# Different Architectures

| | Aggregation | Edge Function | Learnable parameters |
|---|---|---|---|
| PointNet [Qi et al. 2017b] | – | $h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j) = h_{\Theta}(\mathbf{x}_i)$ | $\Theta$ |
| PointNet++ [Qi et al. 2017c] | max | $h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j) = h_{\Theta}(\mathbf{x}_j)$ | $\Theta$ |
| MoNet [Monti et al. 2017a] | $\sum$ | $h_{\theta_m, \mathbf{w}_n}(\mathbf{x}_i, \mathbf{x}_j) = \theta_m \cdot (\mathbf{x}_j \odot g_{\mathbf{w}_n}(u(\mathbf{x}_i, \mathbf{x}_j)))$ | $\mathbf{w}_n, \theta_m$ |
| PCNN [Atzmon et al. 2018] | $\sum$ | $h_{\theta_m}(\mathbf{x}_i, \mathbf{x}_j) = (\theta_m \cdot \mathbf{x}_j) g(u(\mathbf{x}_i, \mathbf{x}_j))$ | $\theta_m$ |

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

Dynamic Graph CNN for Learning on Point Clouds, Wang et. al

# Standard Convolution

- Edge function is the dot product
- Aggregation is the SUM

$$x'_{im} = \sum_{j:(i,j)\in\mathcal{E}} \theta_m \cdot \mathbf{x}_j,$$

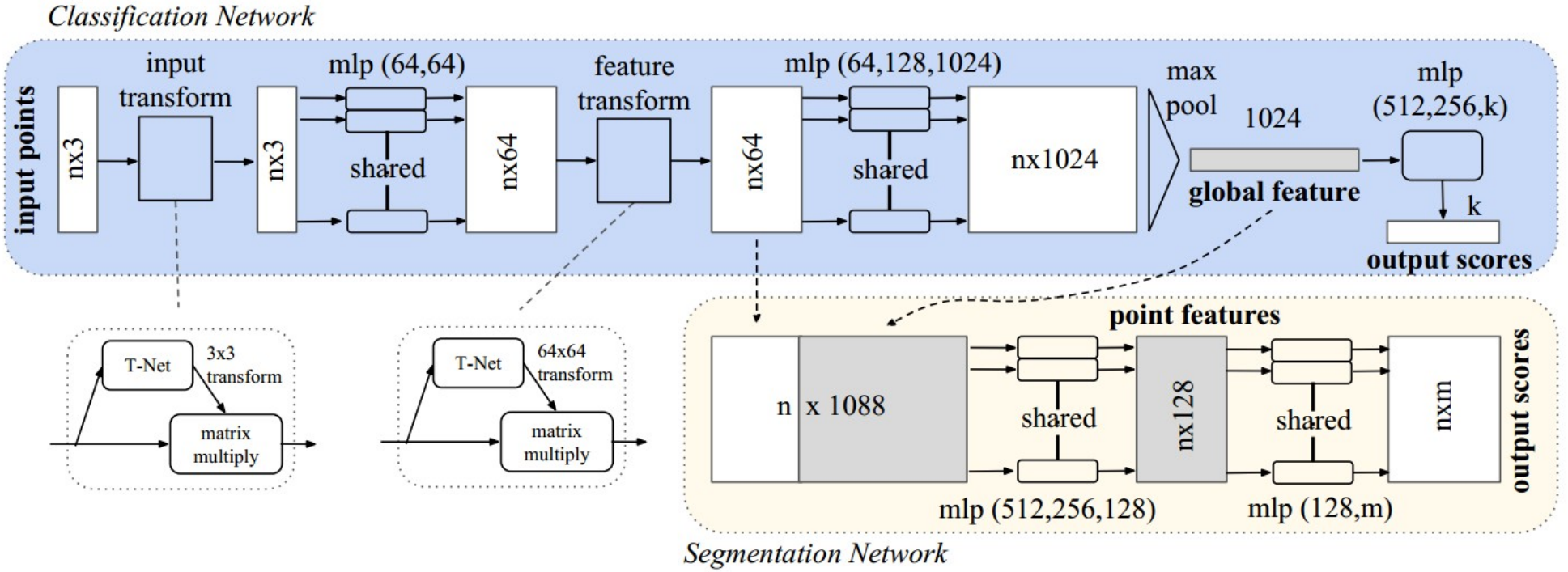
$$\mathbf{x}'_i = \square_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j).$$

Dynamic Graph CNN for Learning on Point Clouds, Wang et. al

# PointNet

- Edge function acts only upon the current point
- No aggregation

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



# MoNet

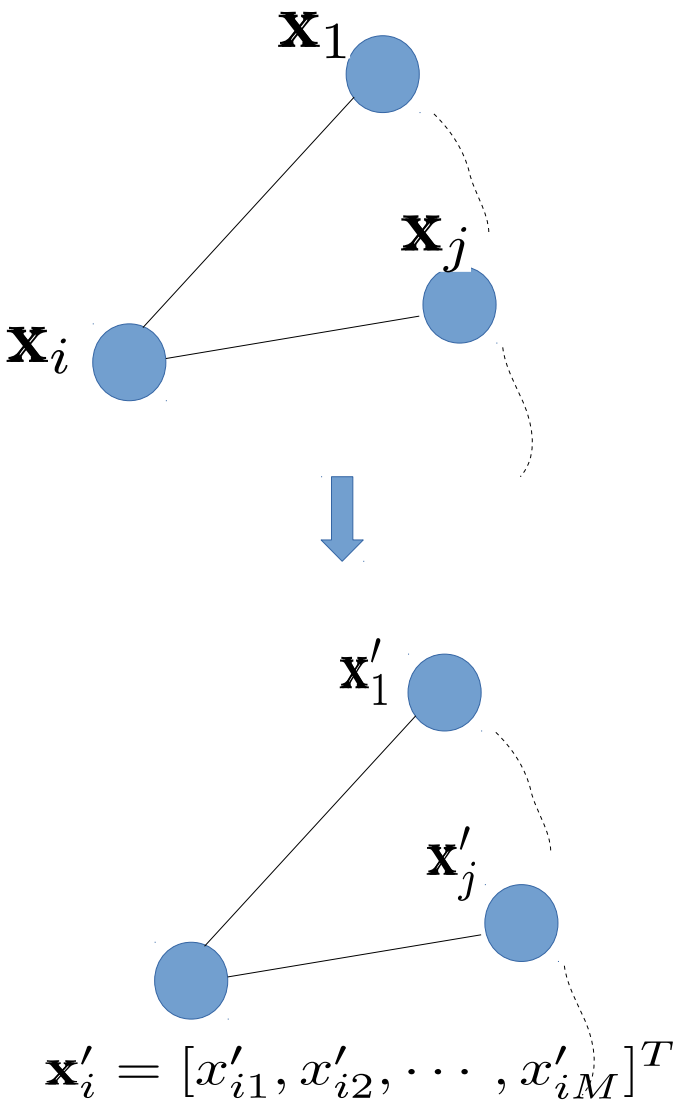
- Edge Function is Gaussian Mixture Like
- Aggregation is SUM

$$h_{\theta_m, \boldsymbol{w_n}}(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\theta}_m \cdot (\mathbf{x}_j \odot g_{\boldsymbol{w}_n}(u(\mathbf{x}_i, \mathbf{x}_j)))$$

$$x'_{im} = \sum_{j:(i,j)\in\mathcal{E}} \boldsymbol{\theta}_m \cdot (\mathbf{x}_j \odot g_{\boldsymbol{w_n}}(u(\mathbf{x}_i, \mathbf{x}_j)))$$

Dynamic Graph CNN for Learning on Point Clouds, Wang et. al

# MoNet




$$\boldsymbol{x}_i, \boldsymbol{x}_j$$

$$\boldsymbol{u}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$ ← pseudo coordinate vector

$$w_m = \exp\left(-\frac{1}{2}(\boldsymbol{u} - \boldsymbol{\mu}_m)^T \Sigma_m^{-1}(\boldsymbol{u} - \boldsymbol{\mu}_m)\right)$$

$$\boldsymbol{x}_j w_m$$

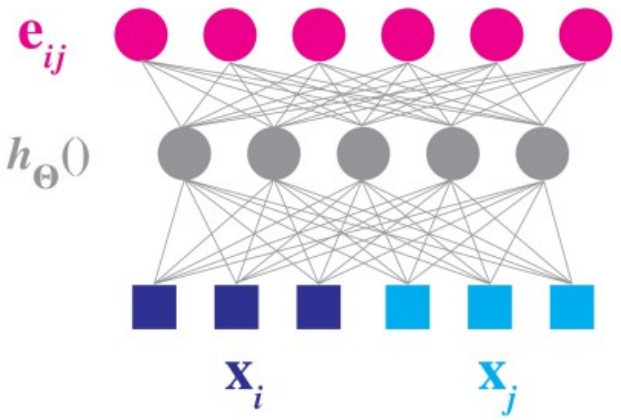
$$\boldsymbol{\theta}_m^T \cdot \boldsymbol{x}_j w_m$$

$$x'_{im} = \sum_j \boldsymbol{\theta}_m^T \cdot \boldsymbol{x}_j w_m$$

Geometric deep learning on graphs and manifolds using mixture model CNNs, Monti et. al

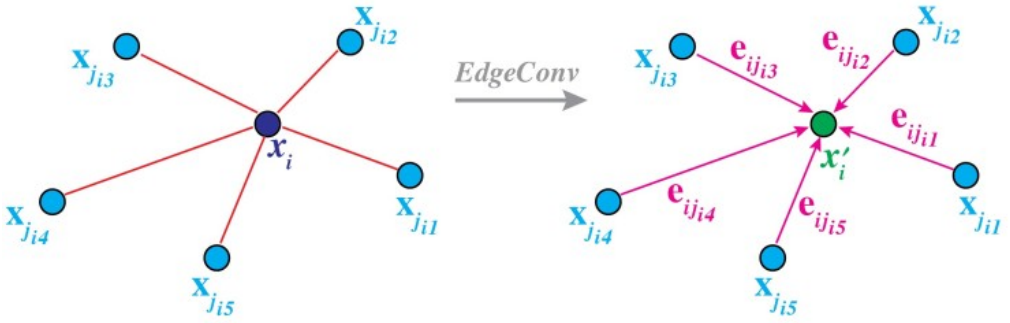
# Dynamic Graph CNN (DGCNN)

- Edge function is a MLP
- Aggregation is MAX



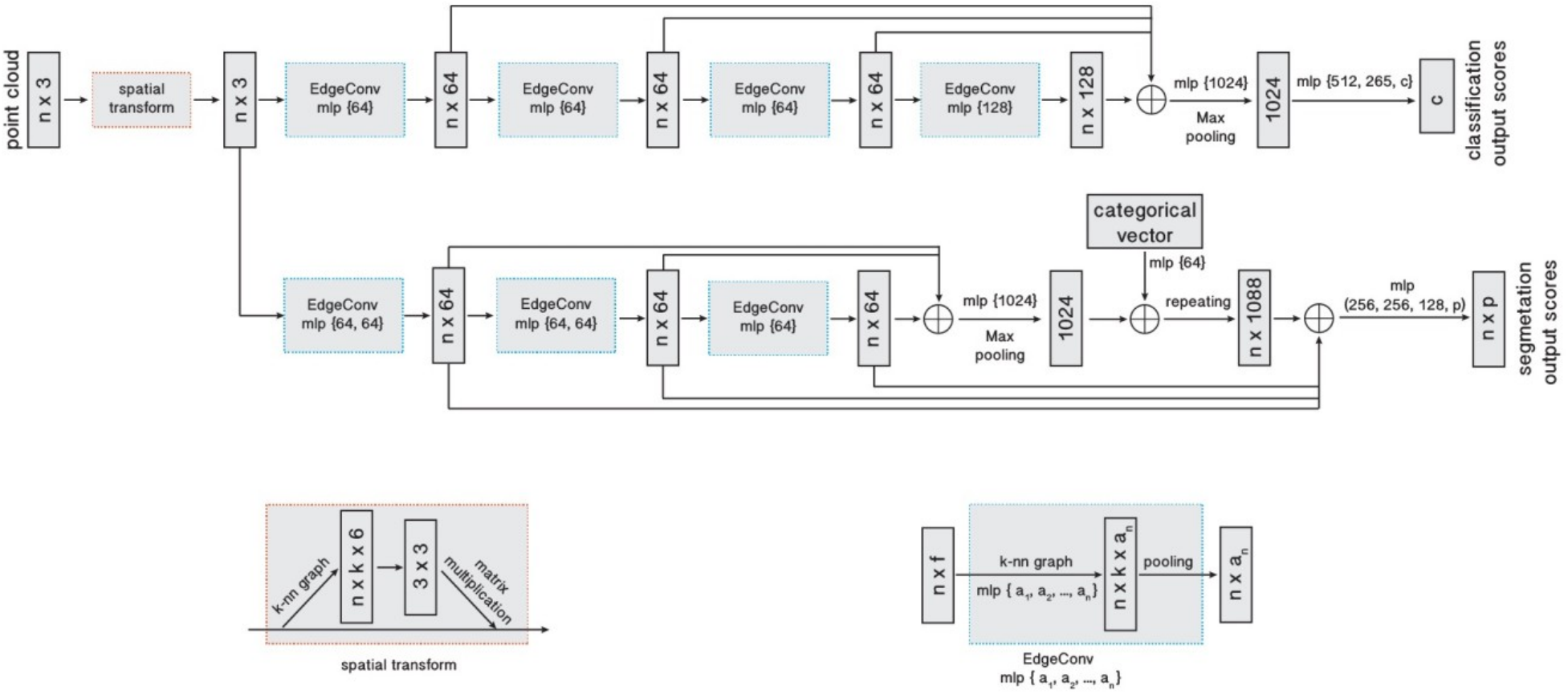

$$e'_{ijm} = \mathrm{ReLU}(\theta_m \cdot (x_j - x_i) + \phi_m \cdot x_i)$$

$$x'_{im} = \max_{j:(i,j)\in\mathcal{E}} e'_{ijm}$$




Dynamic Graph CNN for Learning on Point Clouds, Wang et. al

# DGCNN Architecture




Dynamic Graph CNN for Learning on Point Clouds, Wang et. al

# DGCNN Properties

$$e'_{ijm} = \mathrm{ReLU}(\boldsymbol{\theta}_m \cdot (\mathbf{x}_j - \mathbf{x}_i) + \boldsymbol{\phi}_m \cdot \mathbf{x}_i)$$
$$x'_{im} = \max_{j:(i,j)\in\mathcal{E}} e'_{ijm}$$

- Exploitation of locality information
  - Uses a graph of nearest neighbors and hence locality information is exploited
- Permutation Invariance
  - MAX operation does not consider order, hence the DGCNN is permutation invariant
- Translation Invariance
  - Only partially translation invariant

$$\begin{aligned} e'_{ijm} &= \boldsymbol{\theta}_m \cdot (\mathbf{x}_j + T - (\mathbf{x}_i + T)) + \boldsymbol{\phi}_m \cdot (\mathbf{x}_i + T) \\ &= \boldsymbol{\theta}_m \cdot (\mathbf{x}_j - \mathbf{x}_i) + \boldsymbol{\phi}_m \cdot (\mathbf{x}_i + T). \end{aligned}$$

Dynamic Graph CNN for Learning on Point Clouds, Wang et. al

# DGCNN Performance

| | Mean Class Accuracy | Overall Accuracy |
|---|---|---|
| 3DShapeNets [Wu et al. 2015] | 77.3 | 84.7 |
| VoxNet [Maturana and Scherer 2015] | 83.0 | 85.9 |
| Subvolume [Qi et al. 2016] | 86.0 | 89.2 |
| VRN (single view) [Brock et al. 2016] | 88.98 | - |
| VRN (multiple views) [Brock et al. 2016] | 91.33 | - |
| ECC [Simonovsky and Komodakis 2017] | 83.2 | 87.4 |
| PointNet [Qi et al. 2017b] | 86.0 | 89.2 |
| PointNet++ [Qi et al. 2017c] | - | 90.7 |
| Kd-net [Klokov and Lempitsky 2017] | - | 90.6 |
| PointCNN [Li et al. 2018a] | 88.1 | 92.2 |
| PCNN [Atzmon et al. 2018] | - | 92.3 |
| Ours (baseline) | 88.9 | 91.7 |
| Ours | **90.2** | **92.9** |
| Ours (2048 points) | **90.7** | **93.5** |

Table 2. Classification results on ModelNet40.

| | mean | areo | bag | cap | car | chair | ear phone | guitar | knife | lamp | laptop | motor | mug | pistol | rocket | skate board | table |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # shapes | | 2690 | 76 | 55 | 898 | 3758 | 69 | 787 | 392 | 1547 | 451 | 202 | 184 | 283 | 66 | 152 | 5271 |
| PointNet | 83.7 | 83.4 | 78.7 | 82.5 | 74.9 | 89.6 | 73.0 | 91.5 | 85.9 | 80.8 | 95.3 | 65.2 | 93.0 | 81.2 | 57.9 | 72.8 | 80.6 |
| PointNet++ | 85.1 | 82.4 | 79.0 | **87.7** | 77.3 | **90.8** | 71.8 | 91.0 | 85.9 | 83.7 | 95.3 | 71.6 | 94.1 | 81.3 | 58.7 | 76.4 | 82.6 |
| Kd-Net | 82.3 | 80.1 | 74.6 | 74.3 | 70.3 | 88.6 | 73.5 | 90.2 | 87.2 | 81.0 | 94.9 | 57.4 | 86.7 | 78.1 | 51.8 | 69.9 | 80.3 |
| LocalFeatureNet | 84.3 | **86.1** | 73.0 | 54.9 | 77.4 | 88.8 | 55.0 | 90.6 | 86.5 | 75.2 | **96.1** | 57.3 | 91.7 | 83.1 | 53.9 | 72.5 | **83.8** |
| PCNN | 85.1 | 82.4 | 80.1 | 85.5 | 79.5 | **90.8** | 73.2 | 91.3 | 86.0 | 85.0 | 95.7 | 73.2 | 94.8 | 83.3 | 51.0 | 75.0 | 81.8 |
| PointCNN | **86.1** | 84.1 | **86.45** | 86.0 | **80.8** | 90.6 | **79.7** | **92.3** | **88.4** | **85.3** | **96.1** | **77.2** | **95.3** | **84.2** | **64.2** | **80.0** | 83.0 |
| Ours | 85.2 | 84.0 | 83.4 | 86.7 | 77.8 | 90.6 | 74.7 | 91.2 | 87.5 | 82.8 | 95.7 | 66.3 | 94.9 | 81.1 | 63.5 | 74.5 | 82.6 |

Table 6. Part segmentation results on ShapeNet part dataset. Metric is mIoU(%) on points.

Dynamic Graph CNN for Learning on Point Clouds, Wang et. al

# DGCNN Results