Dynamic Graph CNN for Learning on PointClouds (DGCNN)

ACM Transaction on Graphics, 2019 (Preceding SIGGRAPH 2019)

Yue Wang[1], Yongbin Sun[1], Ziwei Liu[2], Sanjay E. Sarma[1] and Michael M. Bronstein[3] and Justin M. Solomon[1]

1:MIT 2:UCBerkeley 3:ImperialCollegeLondon

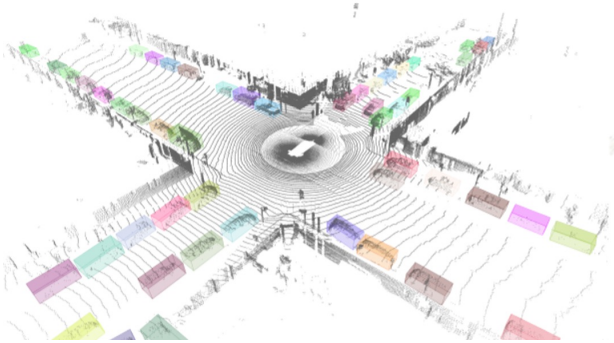
UTCS CS 395T Robot Learning Fall 2023

Presenter: Yuezhi Yang

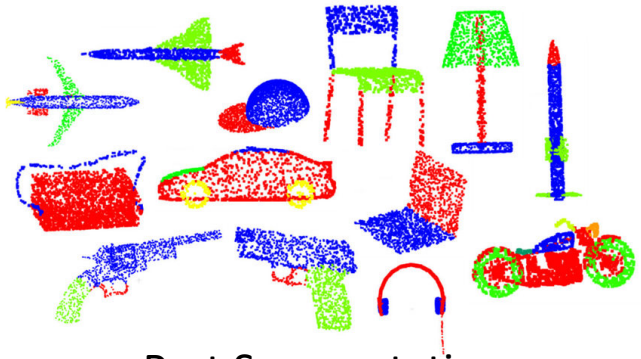
Motivation

- Point clouds are popular in 3D representation for its
  - Easy acquisition
  - flexible geometric representation
  - Vast application

Deep 3D Perception on Point Clouds

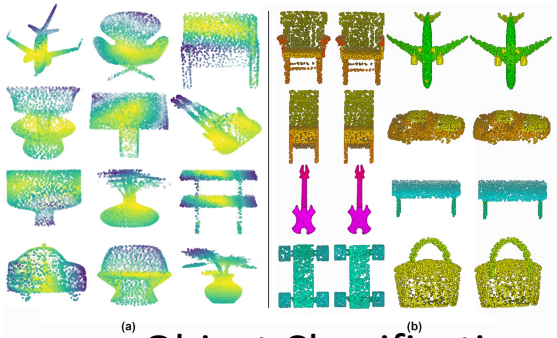


3D Object Detection



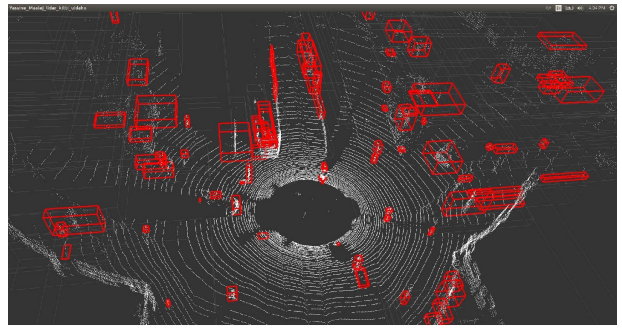
Part Segmentation

Scene Segmentation



Object Classification

Robot Manipulation



3D Object Tracking

Motivation

- How to build model to enable better 3D perception on point cloud?

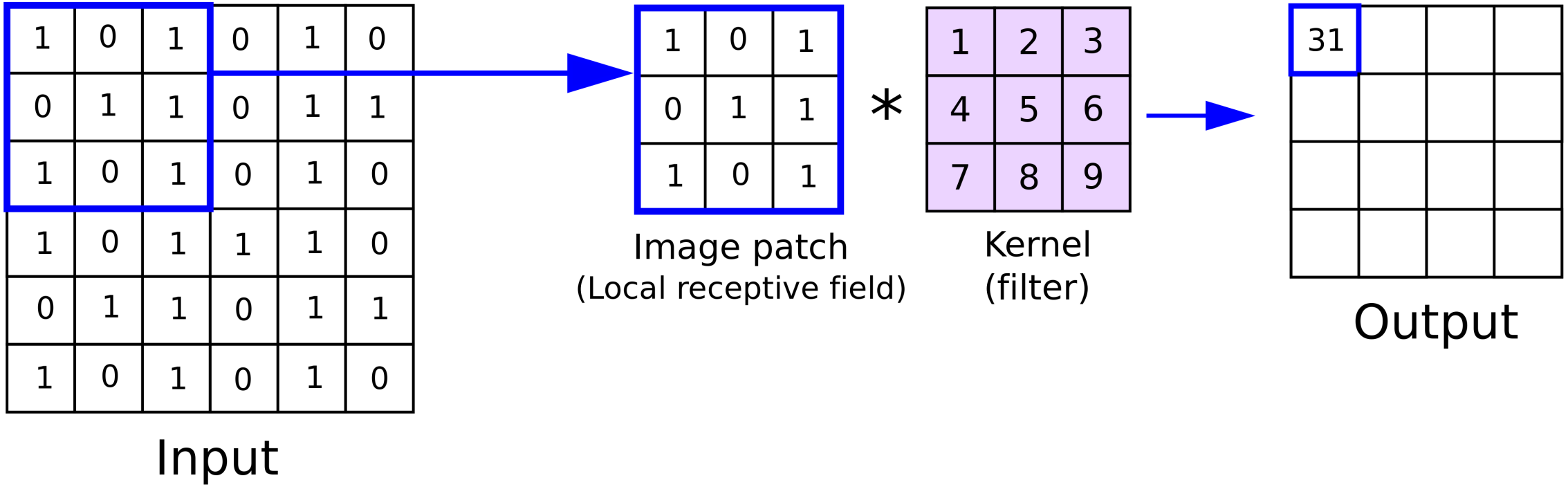
- Overwhelming success of CNNs for image perception ->
  Can we adopt CNN on point clouds perception?

Is this a good problem?
- Local point cloud patches also contains rich geometric information

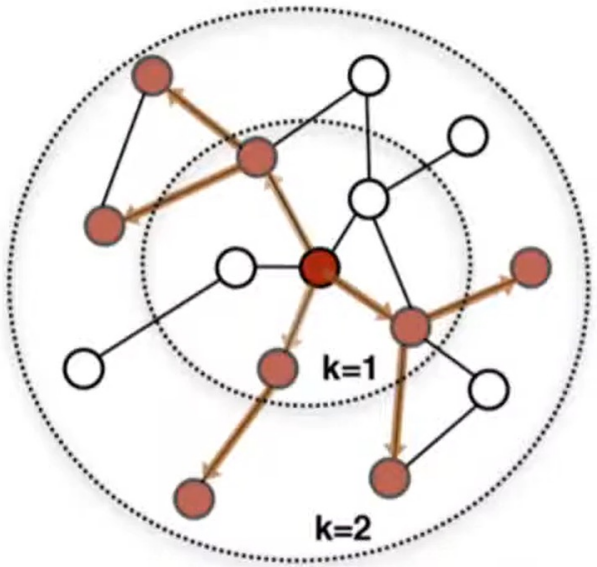
What might be potential challenges?
- Image != Point Cloud for irregularity/ lack of topological info

How far has previous methods gone?

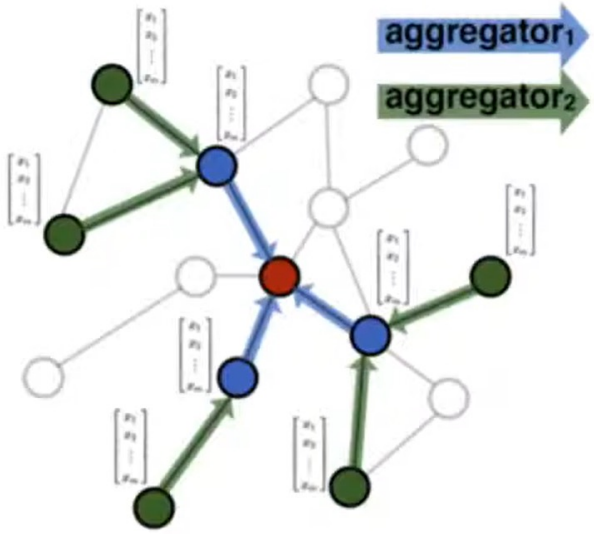
Background – Regular CNN



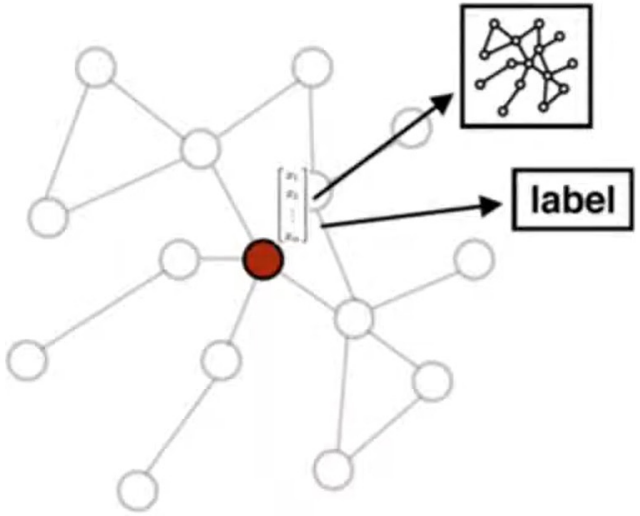
Background – Graph CNN



1. Sample neighborhood



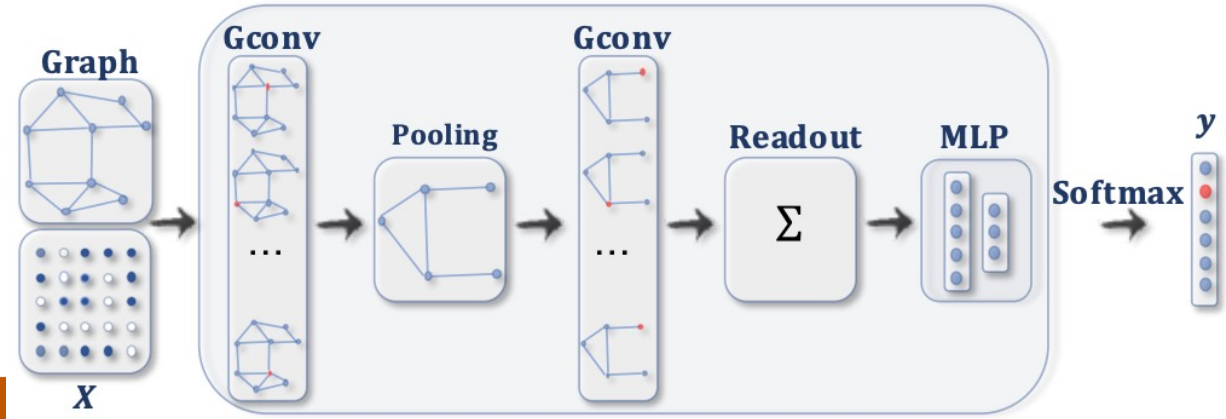
2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

Steps:

1. Construct graph
2. Apply filter $h_\Theta(\mathbf{x}_i, \mathbf{x}_j)$
3. Aggregate local features $\mathbf{x}'_i = \square_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j)$
4. Additional pooling/Repeat operations
5. Predict



Previous work

PointNet (Qi et al. 2017)

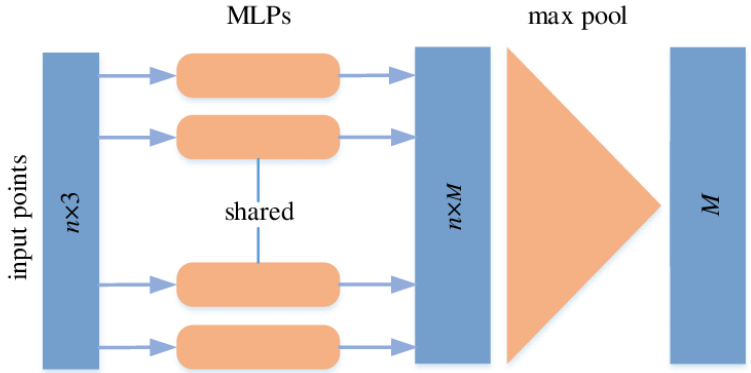
Step:

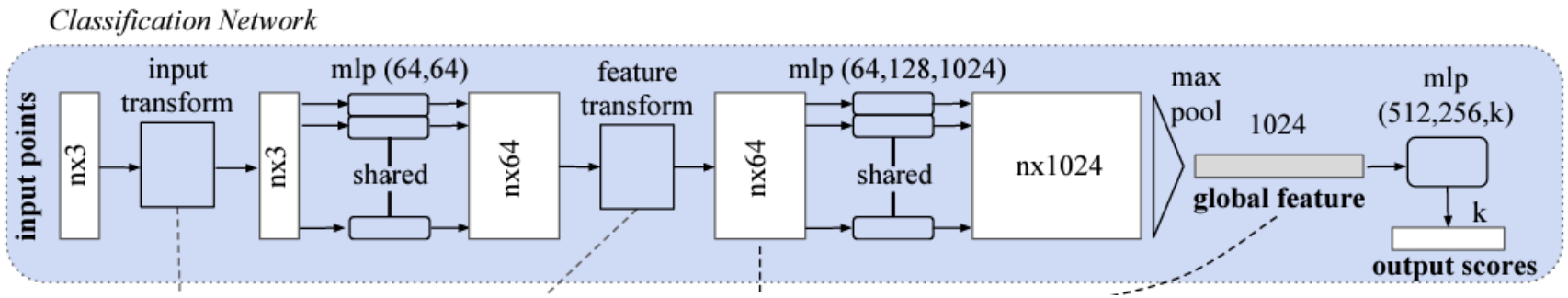
1. Construct a edge-empty graph with only vertices
2. Apply filter $h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j) = h_{\Theta}(\mathbf{x}_i)$
3. No local aggregation
4. Repeat stack of filters and spatial transform
5. Global aggregation $\max_i h_{\Theta}(\mathbf{x}_i)$ Predict

- Pioneer work in direct point cloud processing
- Permutation invariant: global feature will not change if we change the order of the input points

Problem:

- Consider each point independently, No local context for each point -> Only extract global features





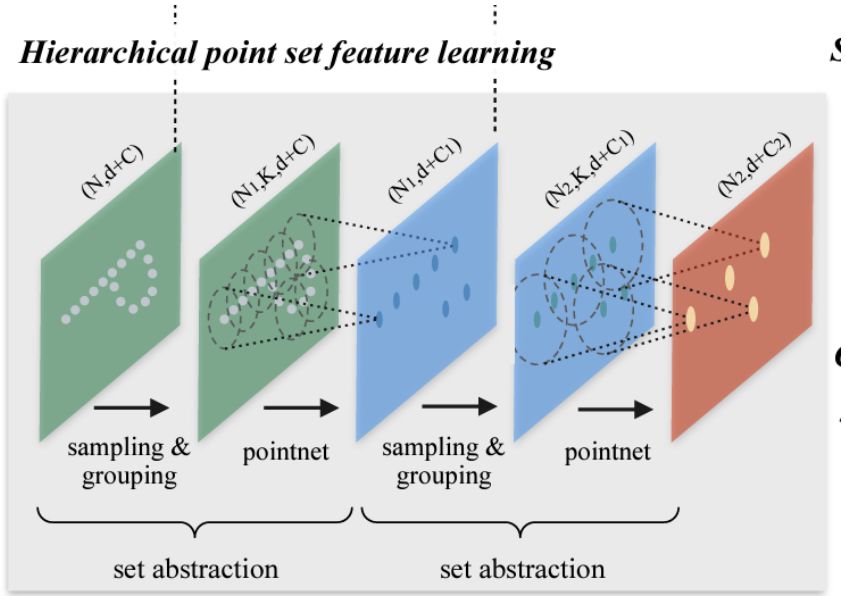
Previous work

PointNet++ (Qi et al. 2017)

Step:

1. Sample anchor points (Farthest Point Sampling) and find neighbour points
2. Apply filter $h_\Theta(\mathbf{x}_i, \mathbf{x}_j) = h_\Theta(\mathbf{x}_j)$
3. Aggregate local features to anchor points $\mathbf{x}'_i = \max_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_j)$
4. Pooling layers to only keep anchor points and discard the rest
5. Repeat 1-4
6. Predict

Basically apply PointNet++ locally and Hierarchically



Problem:

- Computes pairwise distances using point input Euclidean coordinates
  -> Fix graph structure, inadequate point-wise metric
- explore local features through operator aggregation
  Treat points independently at **local scale** as use PointNet as local operator

Previous work – More

More variant of filter and Aggregation:

PCNN (Atzmon et al. 2018)

- Filter: $h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j) = h_{\Theta}(\mathbf{x}_j) = (\boldsymbol{\theta}_m \cdot \mathbf{x}_j)$
- Aggregation: $x'_{im} = \sum_{j \in \mathcal{V}} (h_{\theta(\mathbf{x}_j)}) g(u(\mathbf{x}_i, \mathbf{x}_j))$, g: gaussian kernal compute pairwise distance in Euclidean

MoNet (Monti et al. 2017):

- filter: $h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j) = h_{\Theta}(\mathbf{x}_j) = (\boldsymbol{\theta}_m \cdot \mathbf{x}_j)$
- Aggregation: $x'_{im} = \sum_{j \in \mathcal{V}} \boldsymbol{\theta}_m \cdot (\mathbf{x}_j \odot g_{\boldsymbol{w}_n}(u(\mathbf{x}_i, \mathbf{x}_j)))$

Can we design a network that 1. capture local features ?

2. dynamically update graph structure?
3. keep permutation invariance?

- Yes, this paper propose “EdgeConv”

Contributions

- Present a novel operation, EdgeConv, for transforming local points into a graph and applying convolution on edges to better capture local geometric features

- Show that model can learn to semantically group points by dynamically updating a graph of relationships from layer to layer

- Demonstrate that EdgeConv can be integrated into multiple existing pipelines for point cloud processing

- Achieves state-of-the-art performance in point cloud classification, part segmentation and indoor scene segmentation

Method

Step: （**Edge Convolution**）

1. Construct graph using K nearest neighbour (KNN), point-wise distance measured in **feature space**
2. Apply filter $h_\Theta(\mathbf{x}_i, \mathbf{x}_j) = \bar{h}_\Theta(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i)$

Global info　　Local info

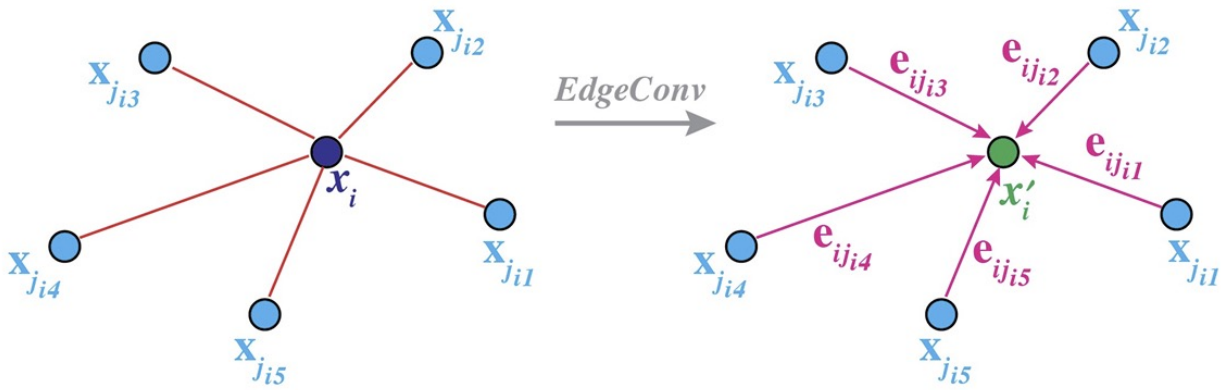
Specifically: $h_\Theta(\mathbf{x}_i, \mathbf{x}_j) = e'_{ijm} = \mathrm{ReLU}(\theta_m \cdot (\mathbf{x}_j - \mathbf{x}_i) + \phi_m \cdot \mathbf{x}_i)$

Learning Parameters

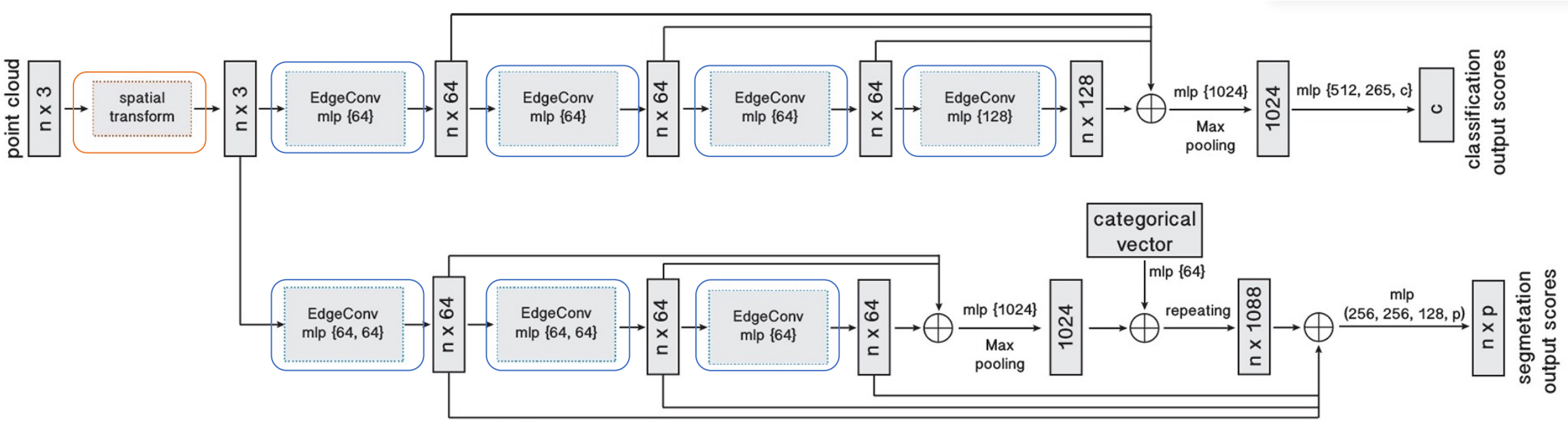
3. Aggregate via max: $x'_{im} = \max_{j:(i,j)\in\mathcal{E}} e'_{ijm}$
4. Repeat Step 1-3, Recompute the graph each time repeat (**Dynamically** update in each layer)
5. Predict

Property:

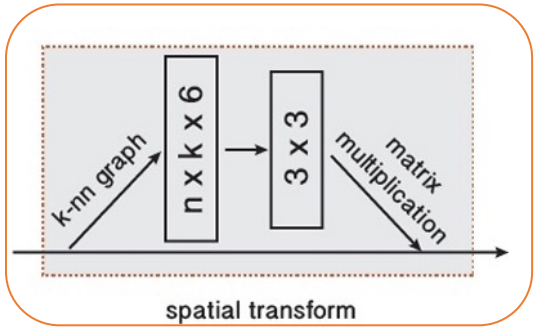
- Extract local info for **every points**
- Permutation invariant
- Graph constructed by KNN in **feature space**
- **Dynamically update** graph each layer
- Analogy to **transformer**

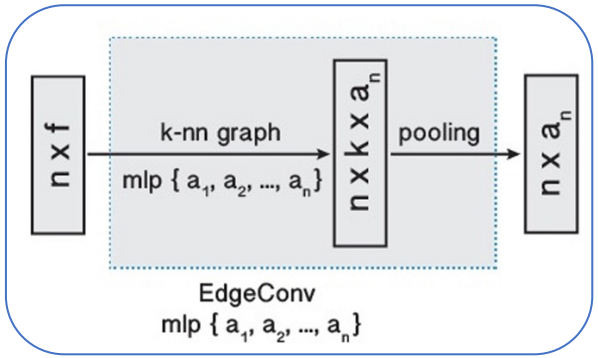


Network Architecture



Converts point cloud into **canonical space** using coordinate differences of point in K-neighbour





Point Cloud Classification

State-of-the-art performance in classification

- Dataset: ModelNet40, with 12,311 models over 40 categories
- 1024 points are sampled **uniformly** in the mesh

	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.

Ours (Baseline): Static graph computed from input
Ours: Dynamic Graph with K=20
Ours(2048 pts) : Dynamic Graph with 2048 points K=40

Model Complexity

	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

Part Segmentation

State-of-the-art comparable performance in part segmentation

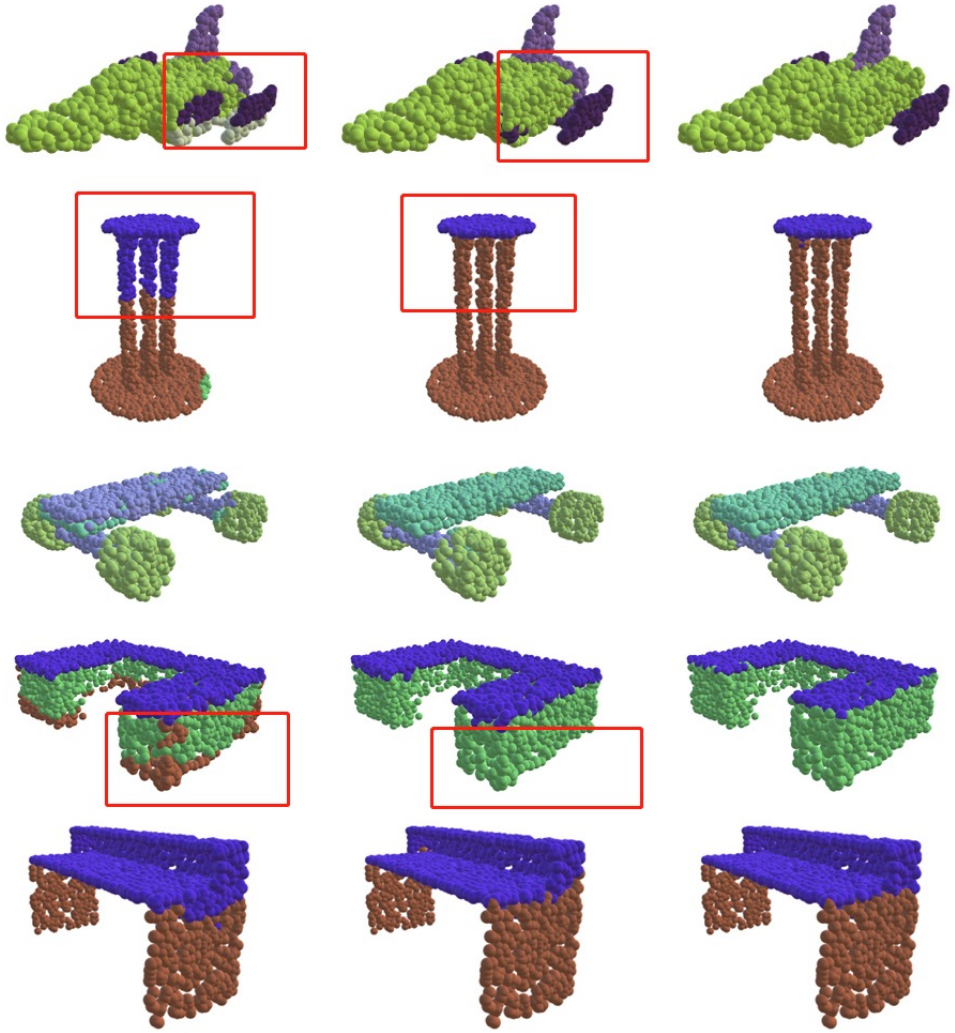
- Dataset: ShapeNet Part, to predict 50 predefined part labels
- 2048 points are sampled **uniformly** in the mesh
- 16,881 shapes from 16 object categories

	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

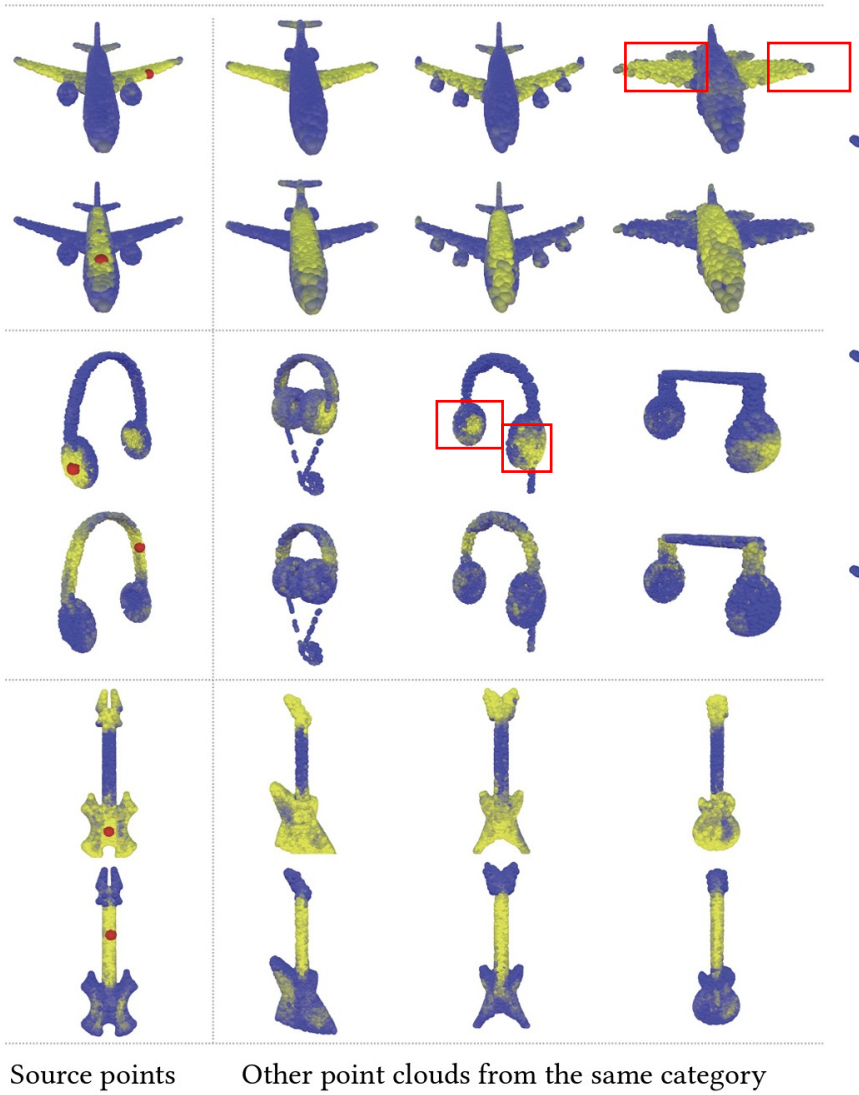
Table 6. Part segmentation results on ShapeNet part dataset. Metric is mIoU(%) on points.

Part Segmentation Result Comparison

State-of-the-art performance with high efficiency

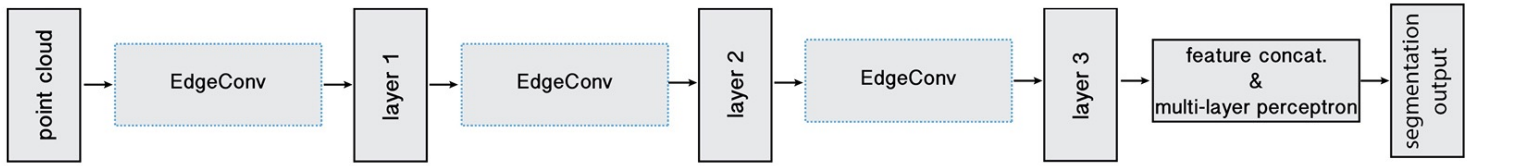


Semantic exploration



Source points Other point clouds from the same category

Supervised on segmentation loss



Indoor Scene Segmentation

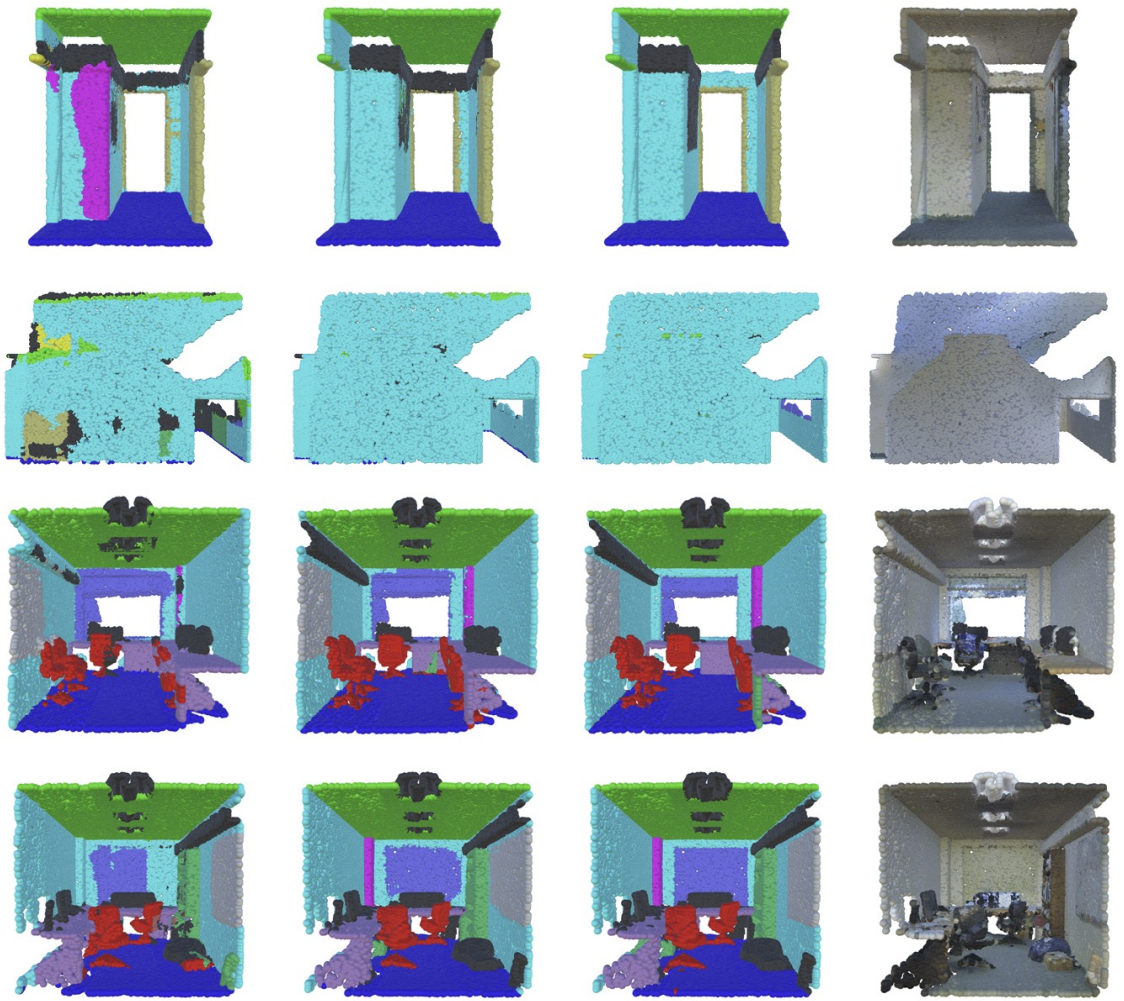


Fig. 10. Semantic segmentation results. From left to right: PointNet, ours, ground truth and point cloud with original color. Notice our model outputs smoother segmentation results, for example, wall (cyan) in top two rows, chairs (red) and columns (magenta) in bottom two rows.

Dataset:
Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS)

- 3D scan point clouds for 6 indoor areas including 272 rooms in total
- Each point belongs to one of 13 semantic categories
- 4,096 points are sampled

	Mean IoU	overall accuracy
PointNet (baseline) [Qi et al. 2017b]	20.1	53.2
PointNet [Qi et al. 2017b]	47.6	78.5
MS + CU(2) [Engelmann et al. 2017]	47.8	79.2
G + RCU [Engelmann et al. 2017]	49.7	81.1
PointCNN [Li et al. 2018a]	**65.39**	-
Ours	56.1	**84.1**

Table 7. 3D semantic segmentation results on S3DIS. MS+CU for multi-scale block features with consolidation units; G+RCU for the grid-blocks with recurrent consolidation Units.

Summary

- Problem: Point cloud perception with deep networks
- Apply CNN technique to irregular point clouds to extract local information
- By transforming point clouds into graph, convolution can be applied with EdgeConv
- Dynamically update graph structure and choose appropriate filter to achieve SOTA result on point cloud classification and segmentation

Limitation & Future Direction

- Require pair-wise distance computation between **all** points-> High GPU memory consumption
  Memory bottleneck when scaling towards large point cloud, DGCNN does not have pooling layer to compress points in forward pass
  Need fast data structure (such as Oct-tree) to accelerate point-wise distance computation

- Fix K value KNN may not be enough to tackle point cloud of uneven density

- More experiment in point clouds processing downstream tasks to be explore
  How about 3D object detection? -> Object DGCNN (Wang et al., NeurIPS 2021)

- Determining the parameter K is still a handcraft job, need an algorithm to determine k given point cloud density and distribution

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

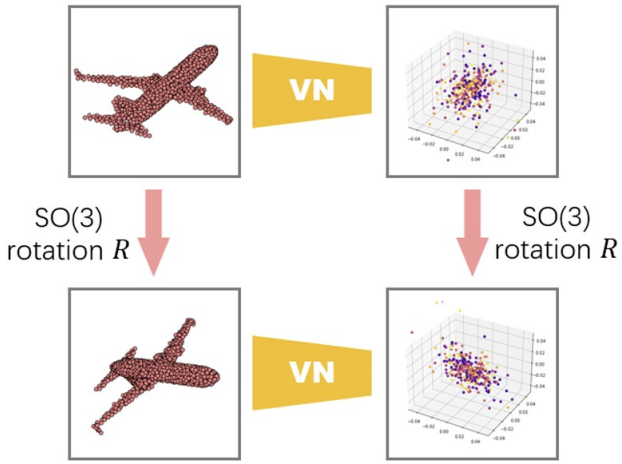
Table 5. Results of our model with different numbers of nearest neighbors.

Discussion

What is next? on point cloud processing network?

- Explore rotation equivariance/invariance of point cloud
  - Vector Neurons : A General Framework for SO(3)- Equivariant Networks (Deng et al.) ICCV 2021
  - Build upon DGCNN and PointNet

Vector Neurons



$$R \in \mathrm{SO}(3)$$

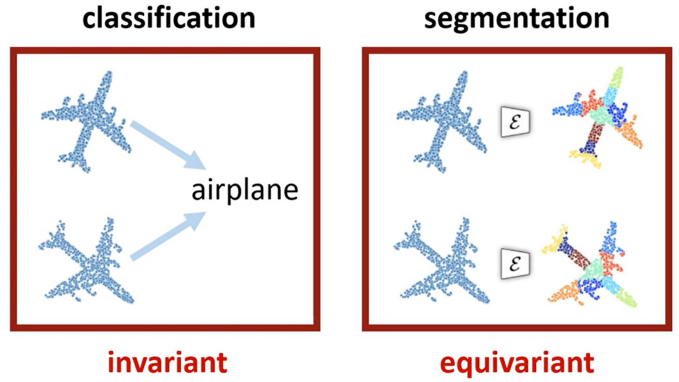
$$f(\mathcal{V}R;\theta) = f(\mathcal{V};\theta)$$

$$f(\mathcal{V}R;\theta) = f(\mathcal{V};\theta)R$$

- What is rotation equivariance and invariance?
  - invariance: output is independent of input pose of object
  - equivariance: output change accordingly to input pose
- Why do we need rotation equivariance and invariance?
  - Shape in wild may not come with canonical poses
  - Eliminate the prior where all the shape in dataset(e.g. ShapeNet) are aligned

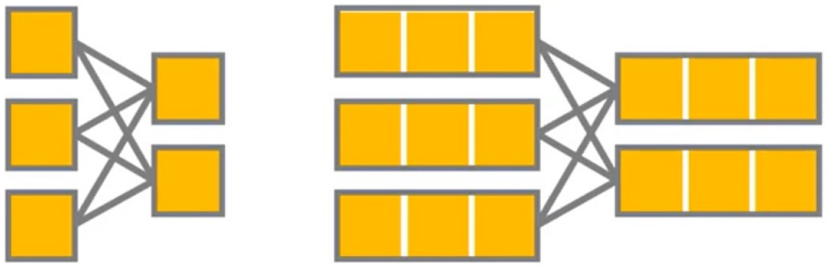
- Why do we care if we have data augmentation and spatial transformer?
  - Theoretical vs Empirical perspective
  - data vs network perspective
  - 3D SO(3) group are large and augmentation maybe resource consuming

- Can we extend equivariance/ invariant logic towards different property of 3D PCs?
  - Global pose (As Rigid As Possible), scale (ACAP), density



Equivariance Network

Vector Neurons



(classical) scalar neurons

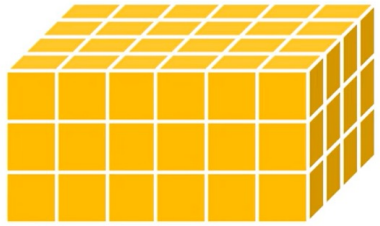
vector neurons

Features

Classical:

$N \times C \times 1$ feature

VN:



$N \times C \times 3$ feature

Feature

Linear Layer

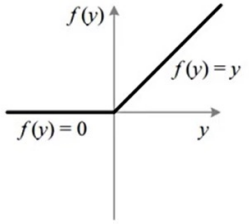
Linear operator $f_{\text{lin}}(\,\cdot\,;\mathbf{W})$ with learnable weights $\mathbf{W} \in \mathbb{R}^{C' \times C}$:

$$\boldsymbol{V}' = f_{\text{lin}}(\boldsymbol{V};\mathbf{W}) = \mathbf{W}\boldsymbol{V} \in \mathbb{R}^{C' \times 3}$$

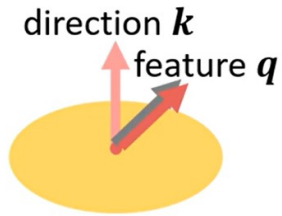
Equivariance to rotation $R \in \mathrm{SO}(3)$:

$$f_{\text{lin}}(\boldsymbol{V}R;\mathbf{W}) = \mathbf{W}\boldsymbol{V}R = f_{\text{lin}}(\boldsymbol{V};\mathbf{W})R = \boldsymbol{V}'R$$

ReLU in 2D



ReLU in 3D



- Can we extend equivariance/ invariant logic towards different property?
  - Global pose (As Rigid As Possible), scale (ACAP), density

Resource

- Project homepage: https://liuziwei7.github.io/projects/DGCNN

- Code
  - Official Tensorflow & Pytorch Implementation: https://github.com/WangYueFt/dgcnn

  - Pytorch Geometric Package: https://pytorchgeometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/conv/edge_conv.html

  - MXNet DGL (Deep Graph Library) https://docs.dgl.ai/generated/dgl.nn.mxnet.conv.EdgeConv.html