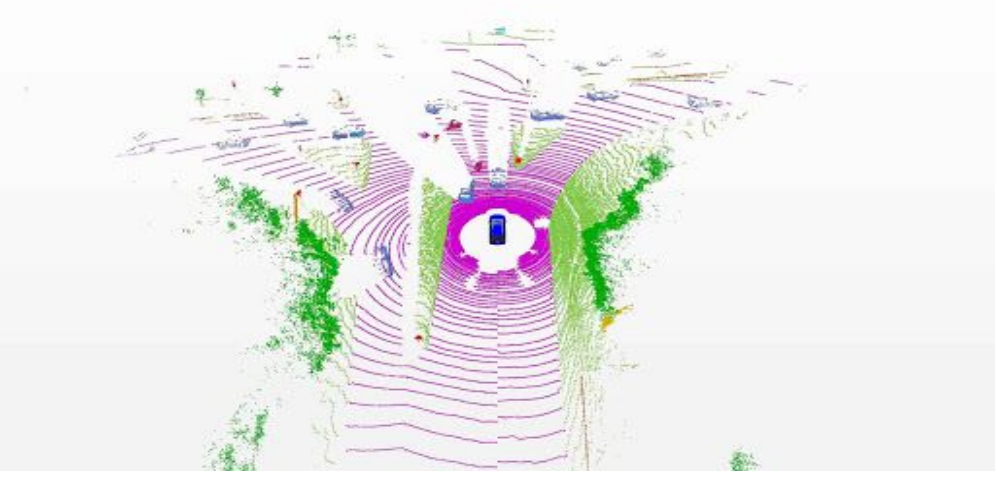
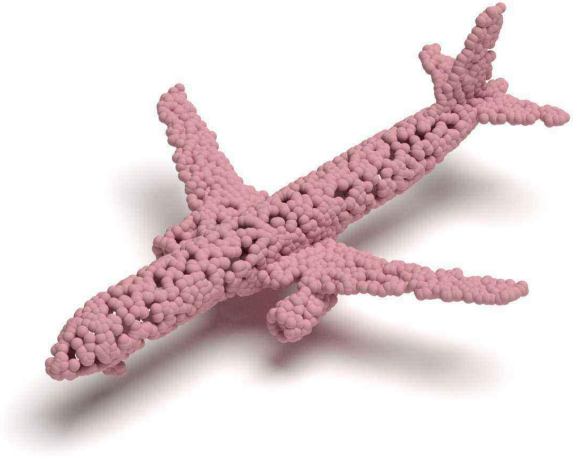
Intro to Deep Volumetric & Point Cloud Learning

Charu Sharma

Assistant Professor

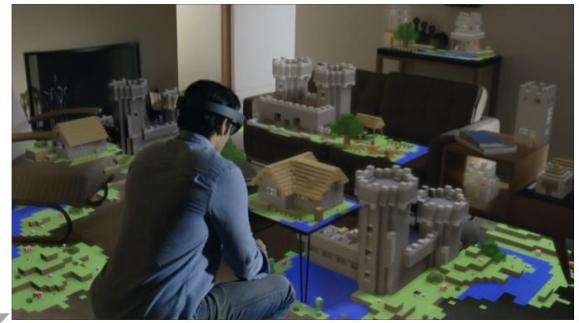
Machine Learning Lab, IIIT Hyderabad



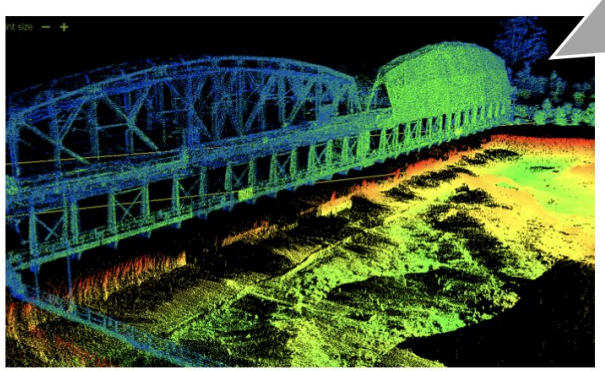


Applications of 3D Data

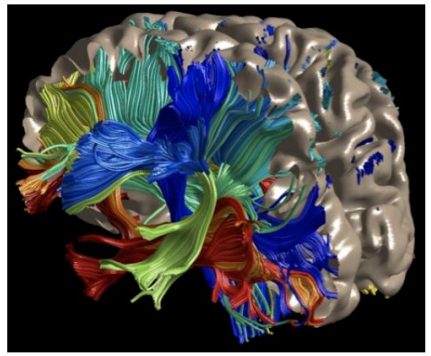
Robotics



Augmented Reality

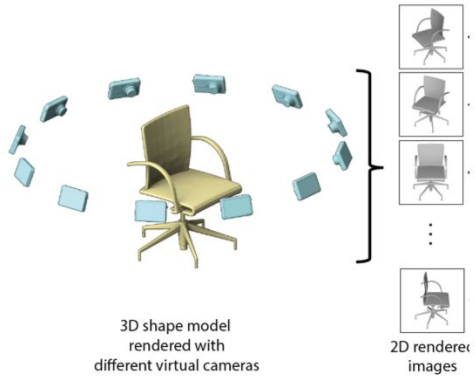


Autonomous driving

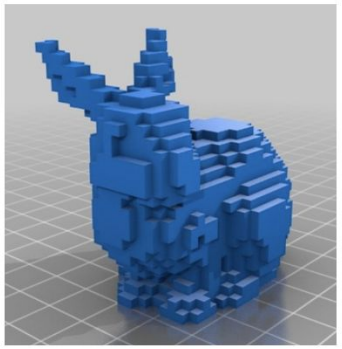


Medical Image Processing

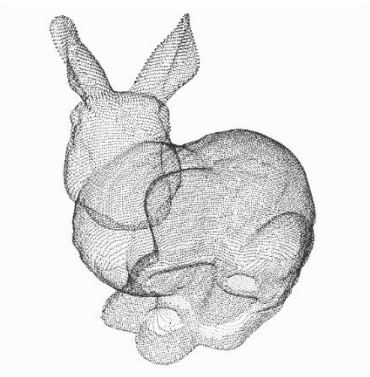
3D Data Representations



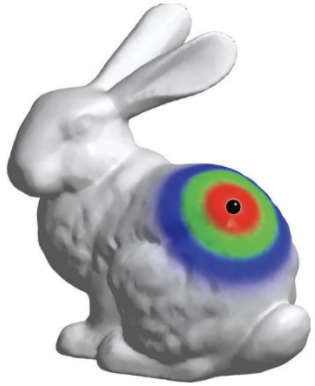
Multi-view



Volumetric

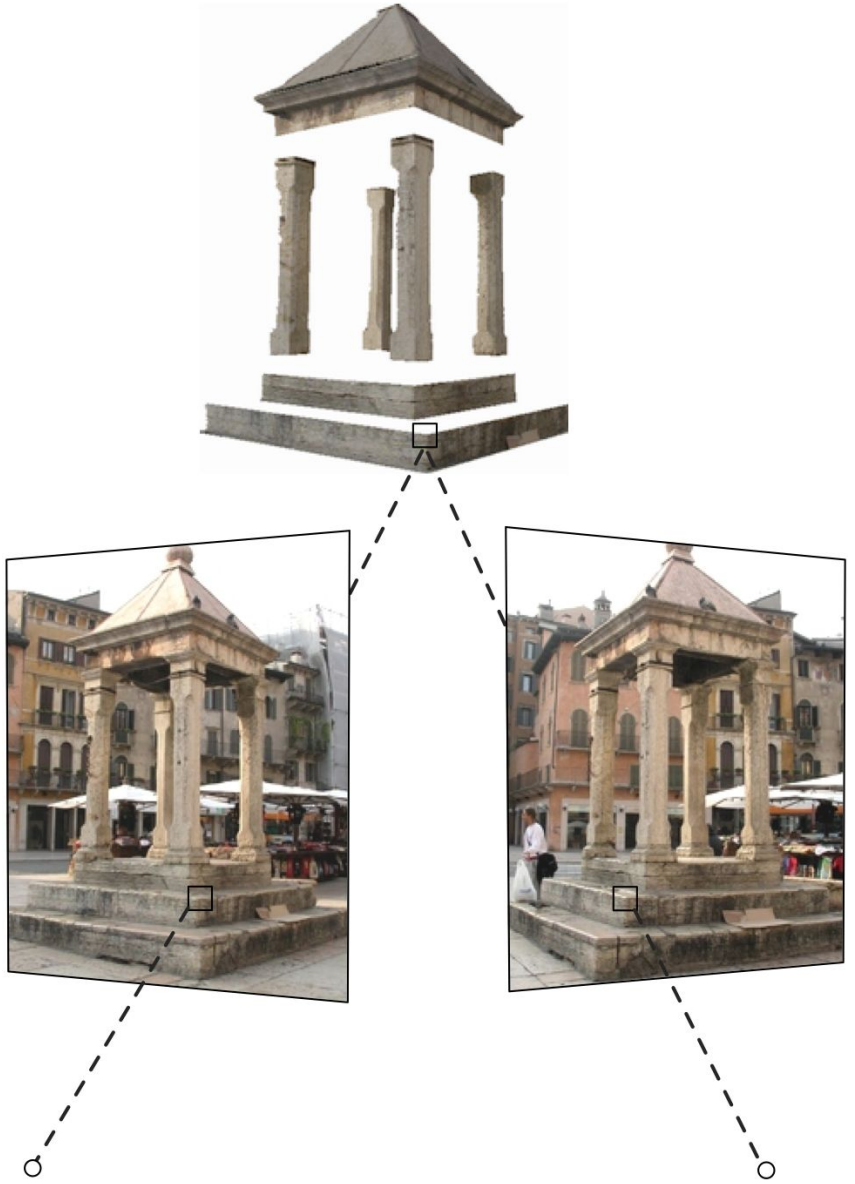


Point Cloud

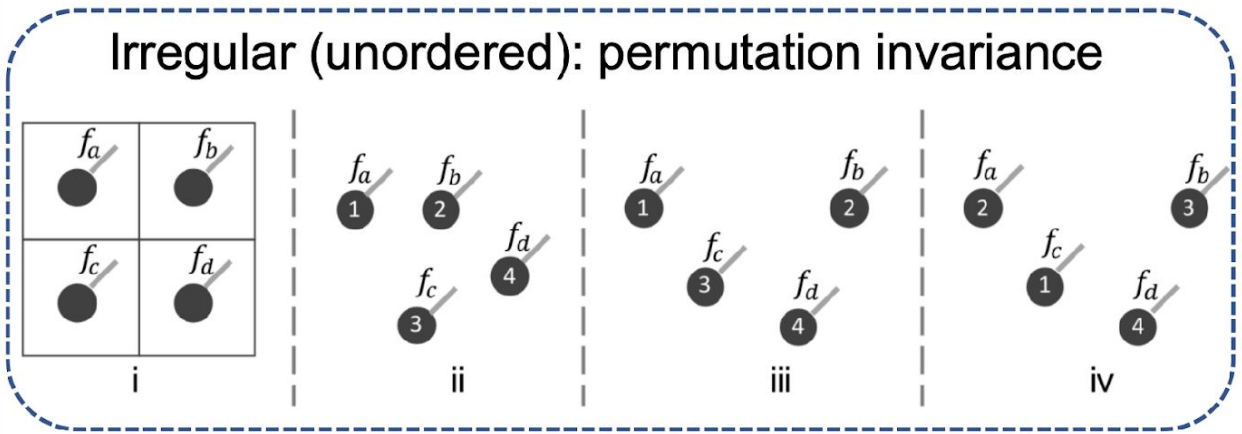


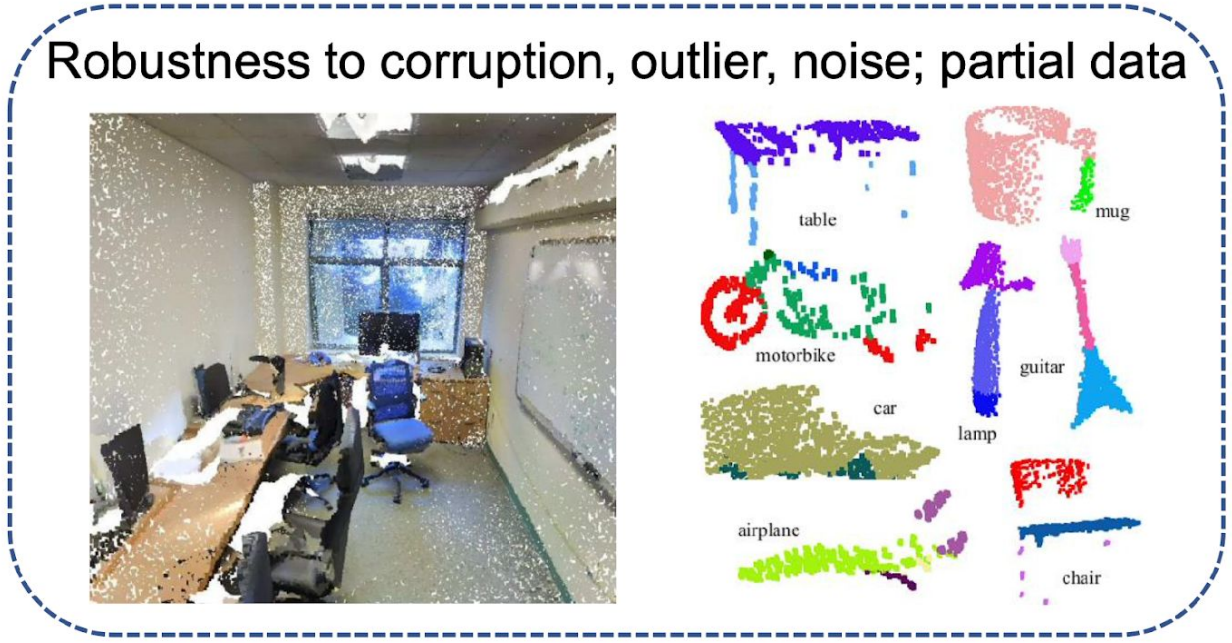
Mesh (Graph CNN)

Traditional 3D Vision: Multi View Geometry



Challenges



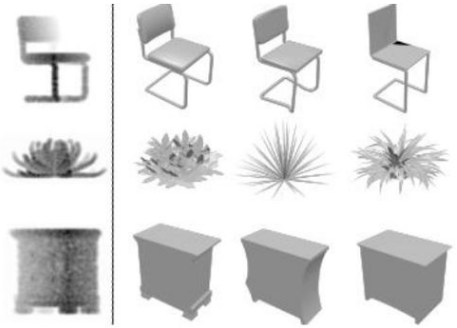


- A point cloud consists a set of points
  - Unstructured
  - Irregularly distributed in the 3D space
  - Unordered
- While ConvNets are great for images, they are not suitable for point clouds
- Deep learning requires a large amount of data, but annotating point cloud is challenging

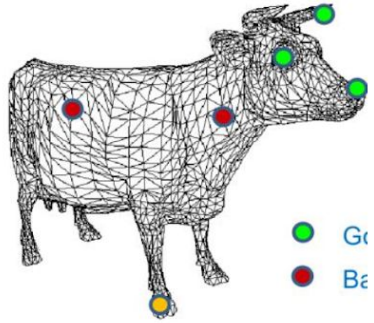
Tasks

lamp

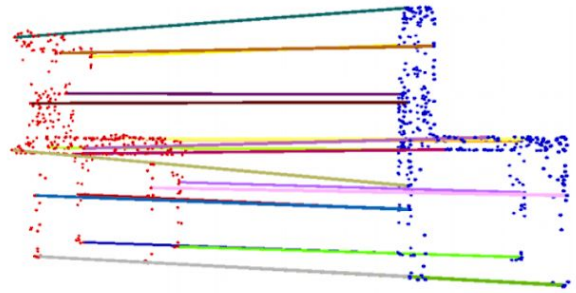
shape classification



shape retrieval

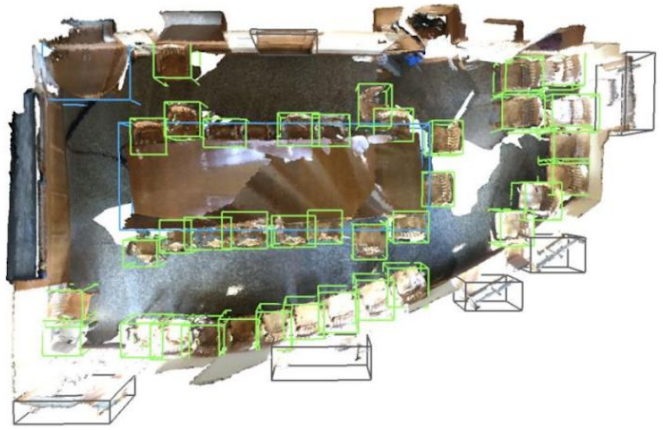


keypoint detection

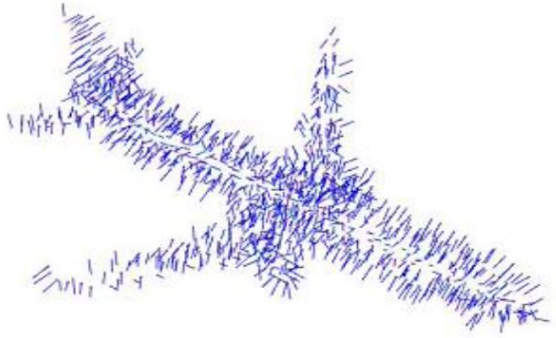


shape correspondence

semantic segmentation



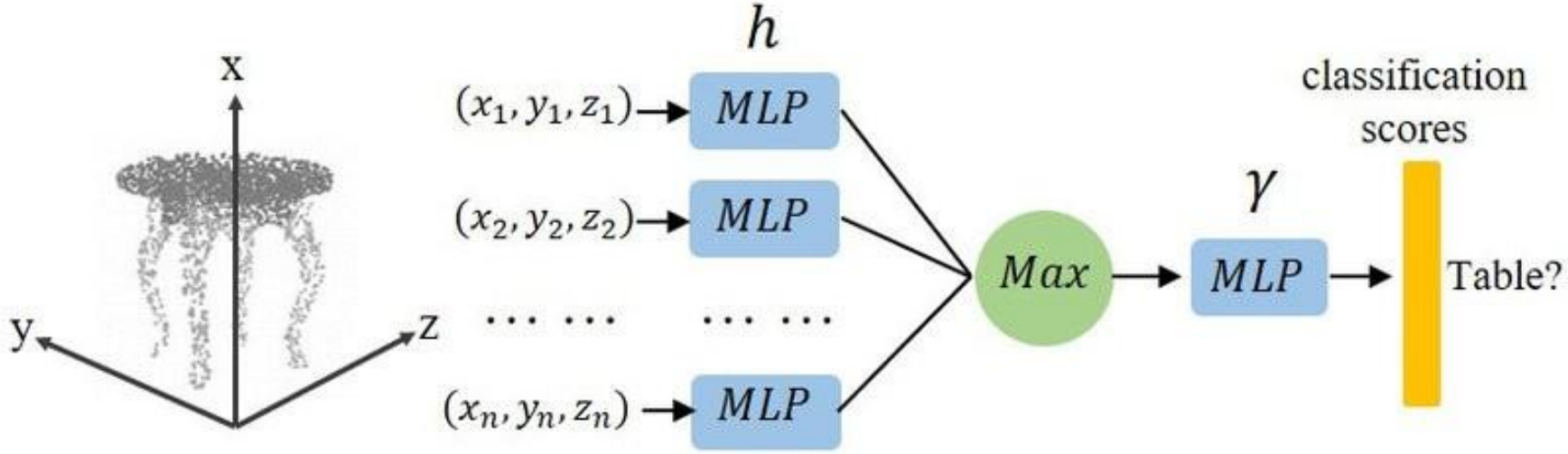
object detection



normal estimation

.....

3D Object Classification



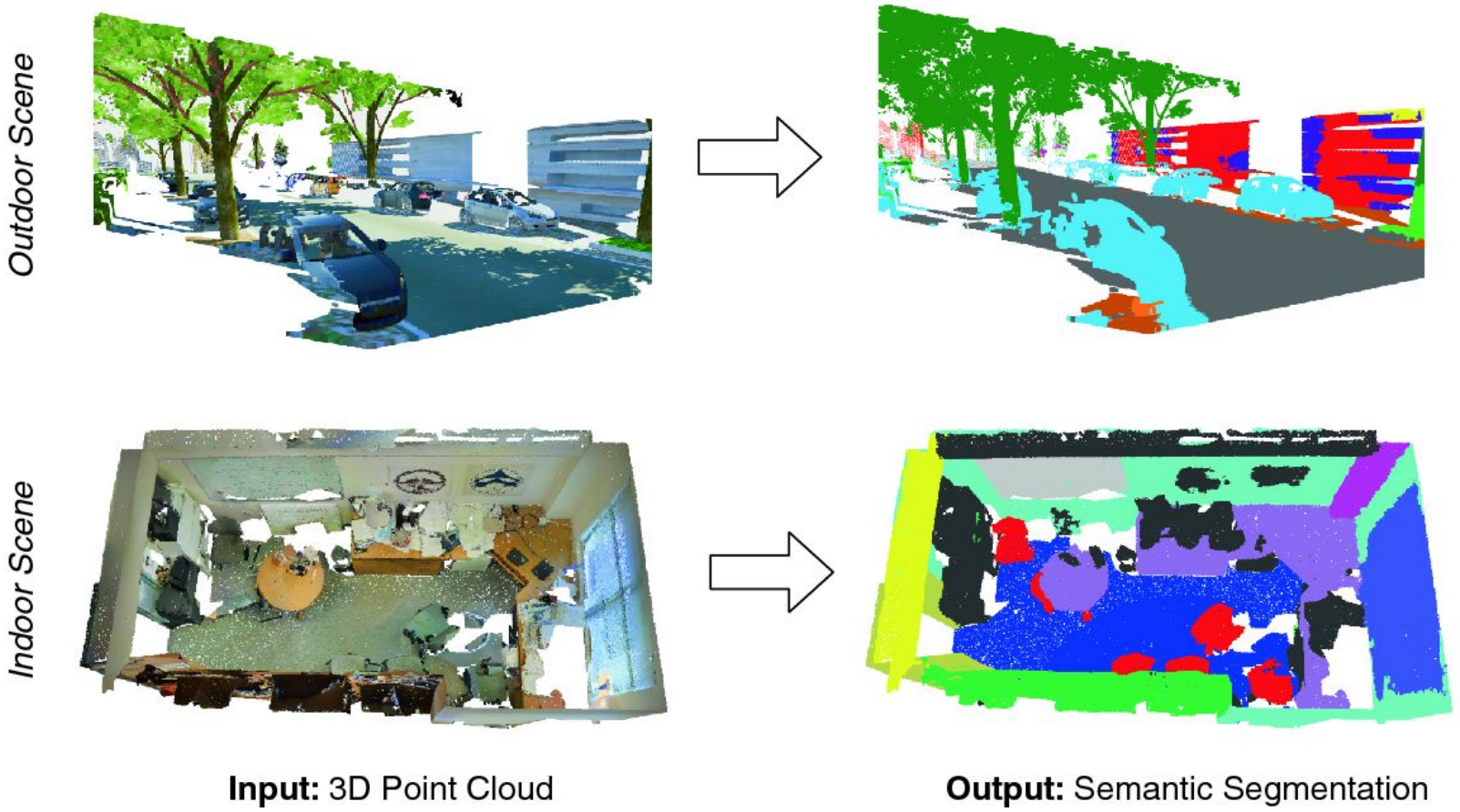
- For the object classification, we consider labeled point cloud (*P*, *y*), where *P* is a point cloud in a collection of point clouds P and *y* is an integer class label (table, chair, aeroplane) that takes values in the set $Y = \{1, \cdots, K\}$.
- Model outputs *K* scores for all the *K* classes.

Part Segmentation

For part segmentation, model will output $n \times m$ scores for each of the n points and each of the m part categories (wings of the aeroplane, earbuds of the headphone) of objects.

Semantic Segmentation

For semantic segmentation, model outputs $n \times m$ scores for each of the n points and each of the m semantic sub-categories (car, tree in outdoor scene).



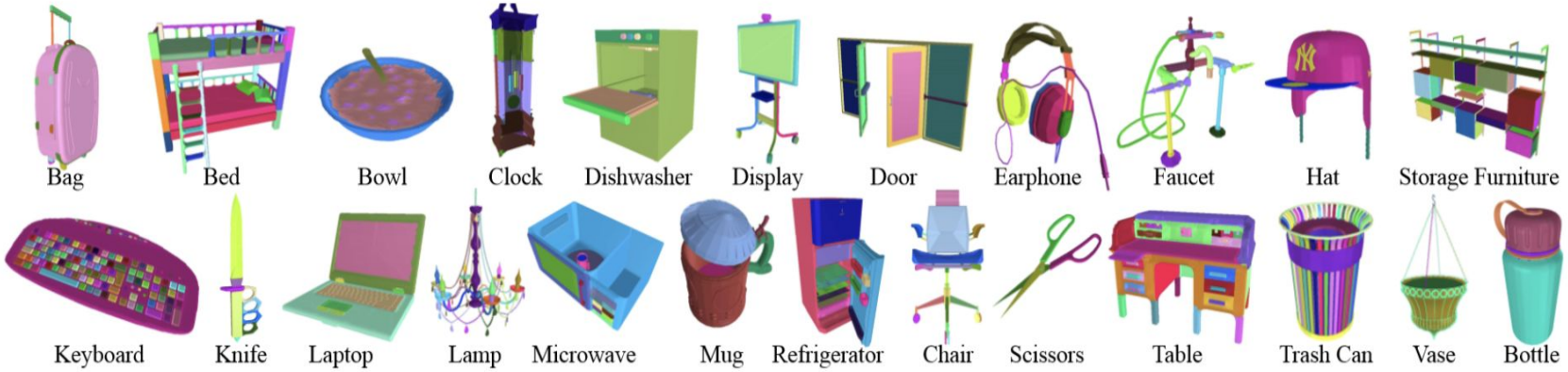
Dataset for 3D Objects: ShapeNet

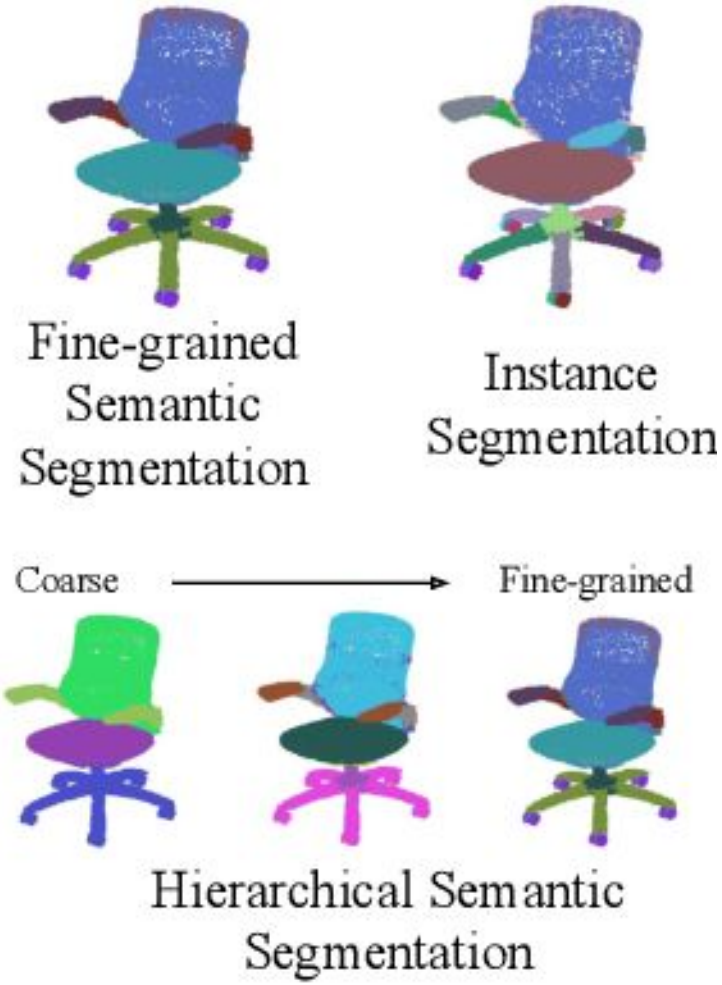
- 3D Object Scans
- Similar to ModelNet
- Used for classification

Chang et al., “**ShapeNet: An Information-Rich 3D Model Repository**” , *arXiv 2015.*

Dataset for 3D Object Parts: ShapeNetPart

- Fine-grained (towards mobility)
- Instance-level
- Hierarchical

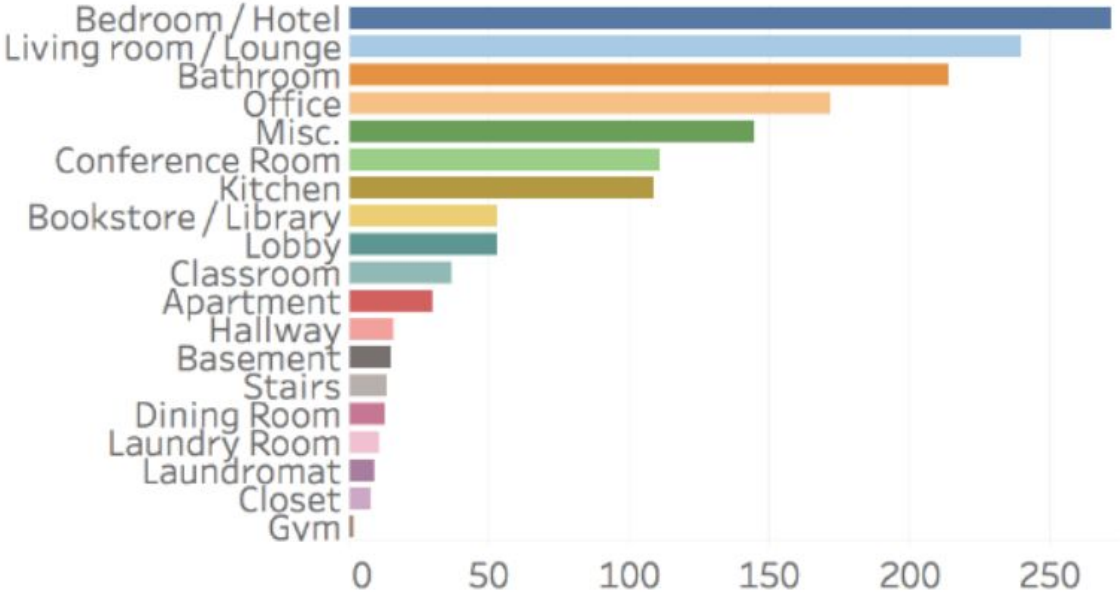




Mo et al., "**PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding** ", *CVPR 2019*

Datasets for Indoor 3D Scenes: ScanNet

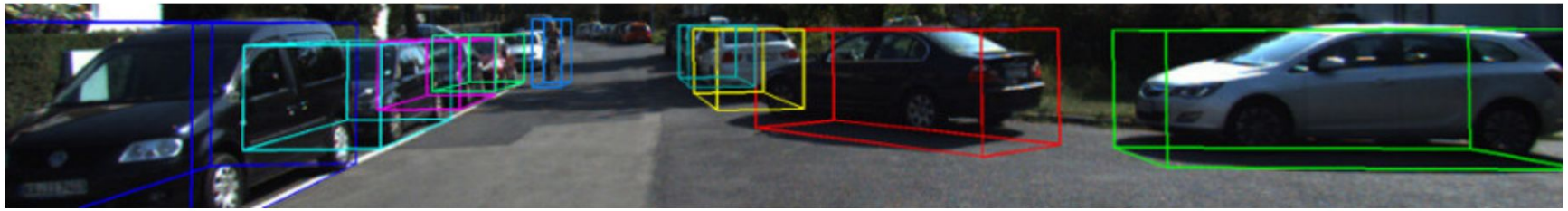
- 2.5 M Views in 1500 RGBD scans
  - 3D camera poses
  - surface reconstructions
  - Instance-level semantic segmentations



Dai et al., "**ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes**", *CVPR 2017*

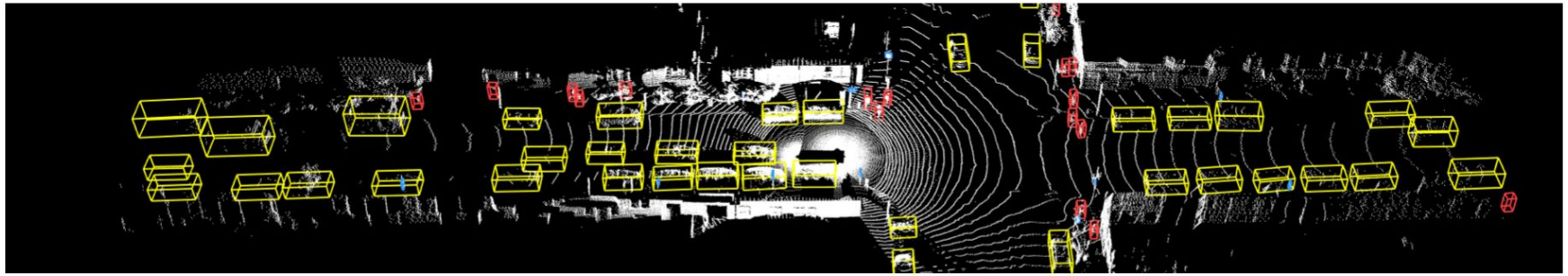
Datasets for Outdoor 3D Scenes

KITTI: LiDAR data, labeled by 3D b.boxes



Semantic KITTI: LiDAR data, labeled per point

Waymo Open Dataset: LiDAR data, labeled by 3D b.boxes

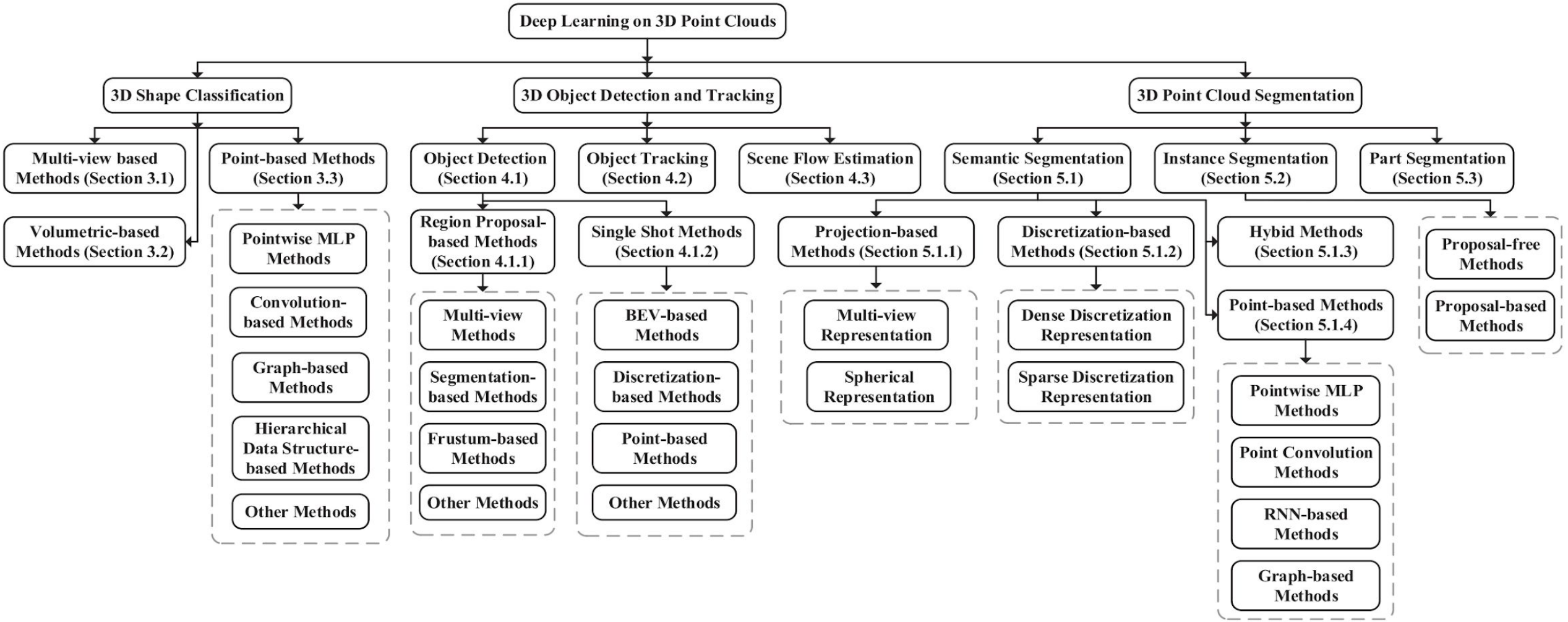


Datasets for 3D Shape Classification							
Name and Reference	Year	#Samples	#Classes	#Training	#Test	Type	Representation
McGill Benchmark [23]	2008	456	19	304	152	Synthetic	Mesh
Sydney Urban Objects [24]	2013	588	14	-	-	Real-World	Point Clouds
ModelNet10 [6]	2015	4899	10	3991	605	Synthetic	Mesh
ModelNet40 [6]	2015	12311	40	9843	2468	Synthetic	Mesh
ShapeNet [8]	2015	51190	55	-	-	Synthetic	Mesh
ScanNet [11]	2017	12283	17	9677	2606	Real-World	RGB-D
ScanObjectNN [7]	2019	2902	15	2321	581	Real-World	Point Clouds
Datasets for 3D Object Detection and Tracking							
Name and Reference	Year	#Scenes	#Classes	#Annotated Frames	#3D Boxes	Secne Type	Sensors
KITTI [14]	2012	22	8	15K	200K	Urban (Driving)	RGB & LiDAR
SUN RGB-D [25]	2015	47	37	5K	65K	Indoor	RGB-D
ScanNetV2 [11]	2018	1.5K	18	-	-	Indoor	RGB-D & Mesh
H3D [26]	2019	160	8	27K	1.1M	Urban (Driving)	RGB & LiDAR
Argoverse [27]	2019	113	15	44K	993K	Urban (Driving)	RGB & LiDAR
Lyft L5 [28]	2019	366	9	46K	1.3M	Urban (Driving)	RGB & LiDAR
A*3D [29]	2019	-	7	39K	230K	Urban (Driving)	RGB & LiDAR
Waymo Open [30]	2020	1K	4	200K	12M	Urban (Driving)	RGB & LiDAR
nuScenes [31]	2020	1K	23	40K	1.4M	Urban (Driving)	RGB & LiDAR
Datasets for 3D Point Cloud Segmentation							
Name and Reference	Year	#Points	#Classes[1]	#Scans	Spatial Size	RGB	Sensors
Oakland [32]	2009	1.6M	5(44)	17	-	N/A	MLS
ISPRS [33]	2012	1.2M	9	-	-	N/A	ALS
Paris-rue-Madame [34]	2014	20M	17	2	-	N/A	MLS
IQmulus [35]	2015	300M	8(22)	10	-	N/A	MLS
ScanNet [11]	2017	-	20(20)	1513	8×4×4	Yes	RGB-D
S3DIS [10]	2017	273M	13(13)	272	10×5×5	Yes	Matterport
Semantic3D [12]	2017	4000M	8(9)	15/15	250×260×80	Yes	TLS
Paris-Lille-3D [36]	2018	143M	9(50)	3	200×280× 30	N/A	MLS
SemanticKITTI [15]	2019	4549M	25(28)	23201/20351	150×100×10	N/A	MLS
Toronto-3D [37]	2020	78.3M	8(9)	4	260×350× 40	Yes	MLS
DALES [38]	2020	505M	8(9)	40	500×500×65	N/A	ALS

Guo, Yulan, et al. "**Deep learning for 3d point clouds: A survey**." *IEEE transactions on PAMI* 2020.

Popular Datasets

- ModelNet40 (classification)
- ShapeNet (classification, segmentation)
- ModelNet10 (subset of ModelNet40, classification)
- Sydney (Classification)
- S3DIS (semantic segmentation)
- ScanNet (segmentation)
- Semantic3D, SemanticKITTI (segmentation)
- KITTI, nuScenes, Waymo (object detection)



Guo, Yulan, et al. "**Deep learning for 3d point clouds: A survey**." *IEEE transactions on pattern analysis and machine intelligence* 2020

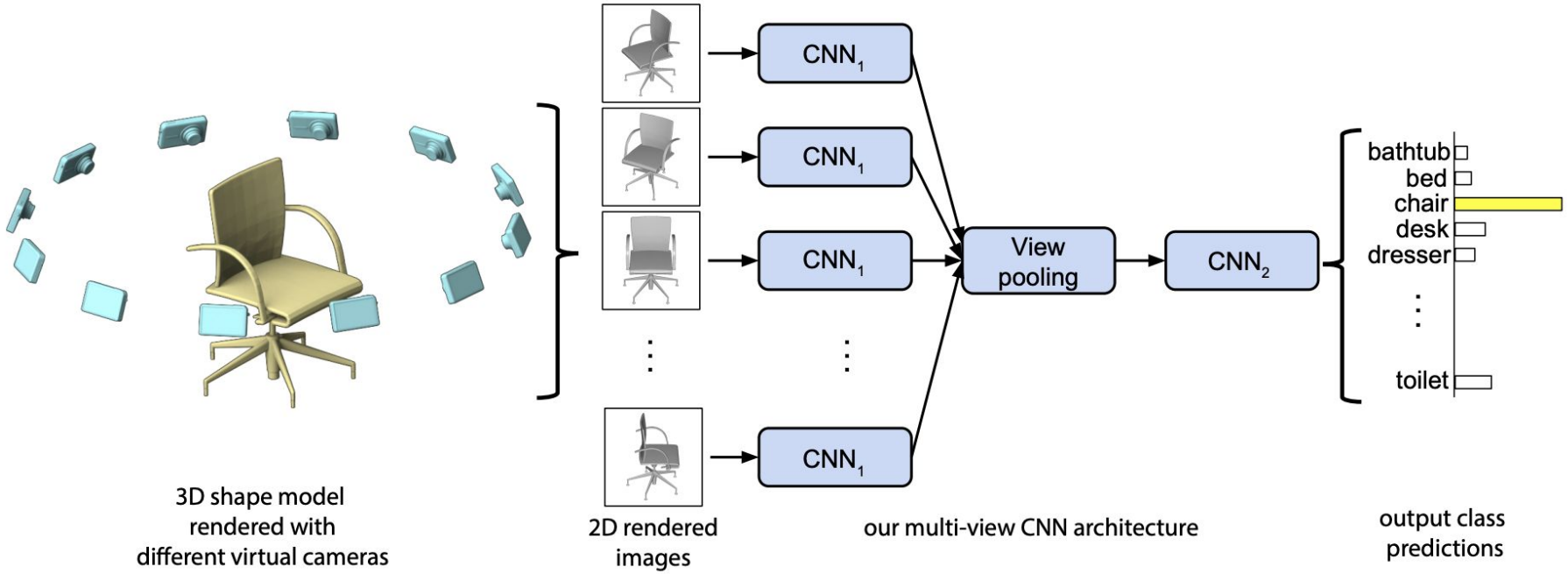
3D Data Representation

- View (image)-based methods
- Volumetric-based methods
  - VoxNet
- Point-based methods
  - PointNet
- Graph-based methods
  - DGCNN
- Mesh-based methods
  - MeshCNN

View-based

Multi-view CNN

3D -> 2D Projections -> Neural Nets (2D CNN)



Multi-view Convolutional Neural Networks for 3D Shape Recognition, ICCV 2015.

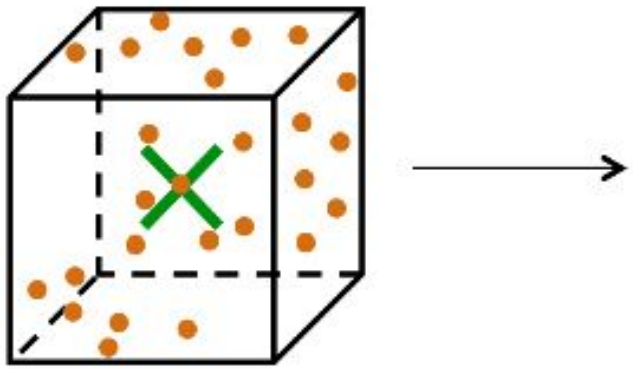
Research Question

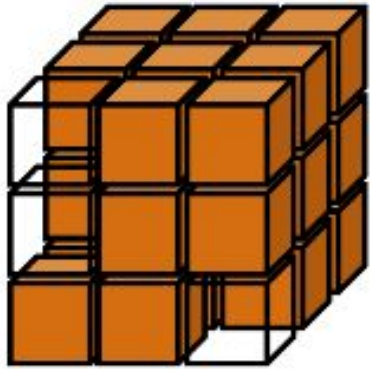
Can we use CNNs without 2D-3D projection?

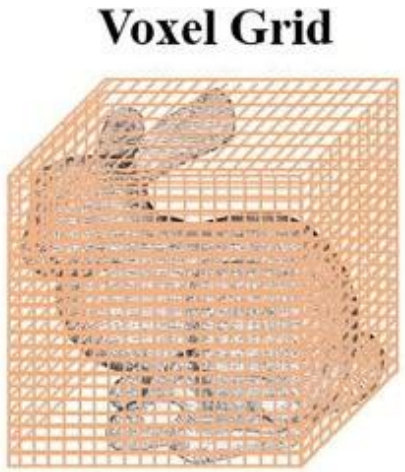
3D convolution?

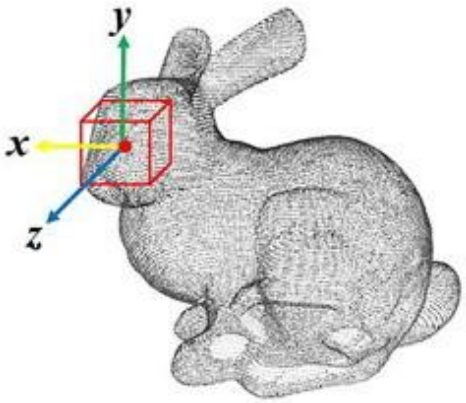
Volumetric-based Approaches

Voxelization







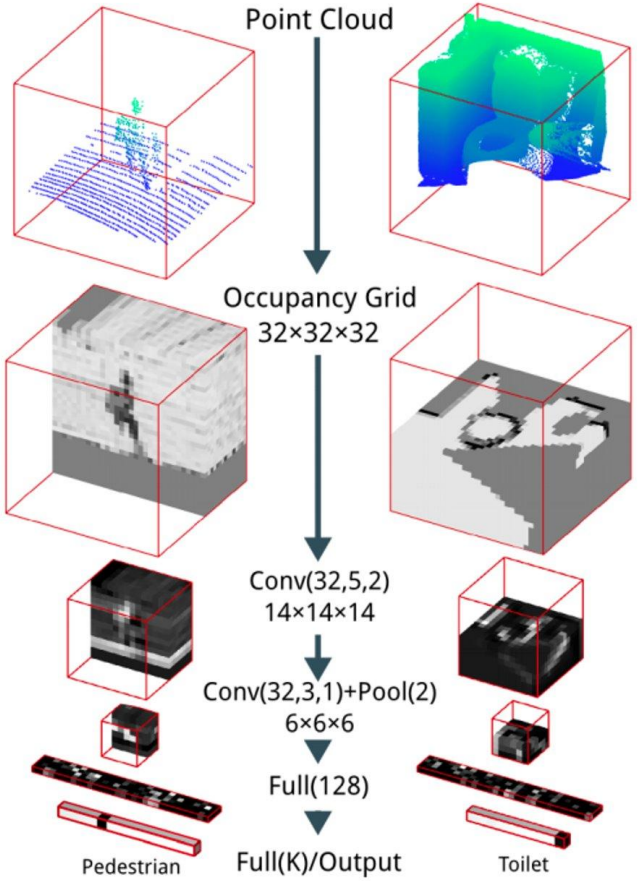


Volumetric-based Approaches

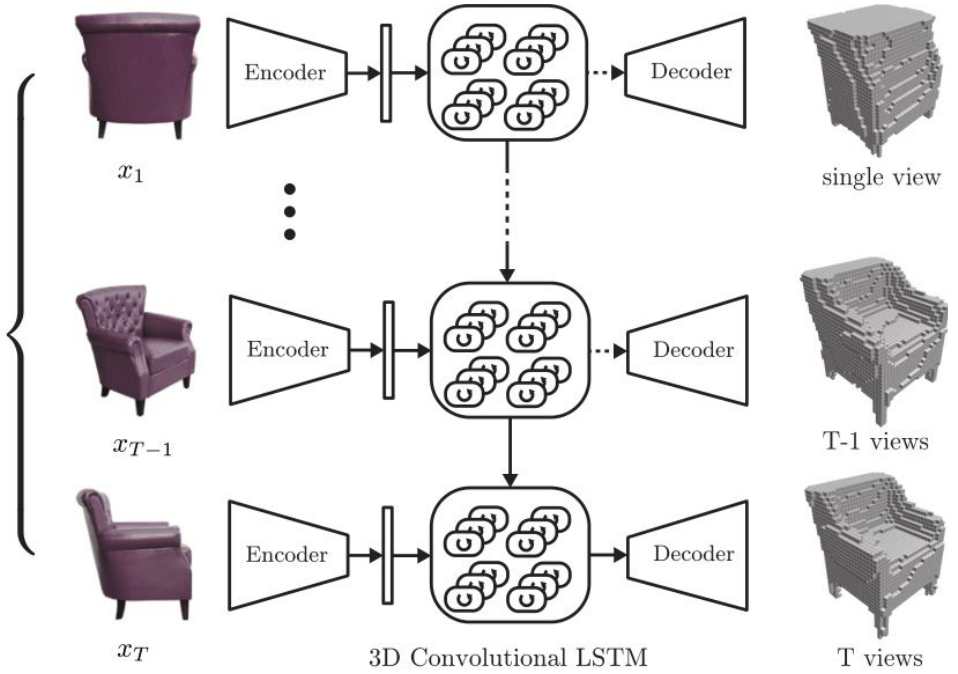
3D Points -> Voxels -> Neural Nets (3D CNN)

- VoxNet (Classification and Segmentation), IROS 2015
  - Integrating a volumetric Occupancy Grid representation with a supervised 3D Convolutional Neural Network (3D CNN).
- 3D-R2N2 (3D Recurrent Reconstruction Neural Network), ECCV 2016
  - Given one or multiple views of an object, the network generates voxelized reconstruction of the object in 3D.
- 3DGAN (Generation), NeurIPS 2016
  - Generates 3D objects from a probabilistic space by leveraging advances in volumetric CNN and GANs.
- VoxelNet (Detection), CVPR 2018
  - A generic 3D detection network that unifies feature extraction and bounding box prediction into a single stage, end-to-end trainable deep network.

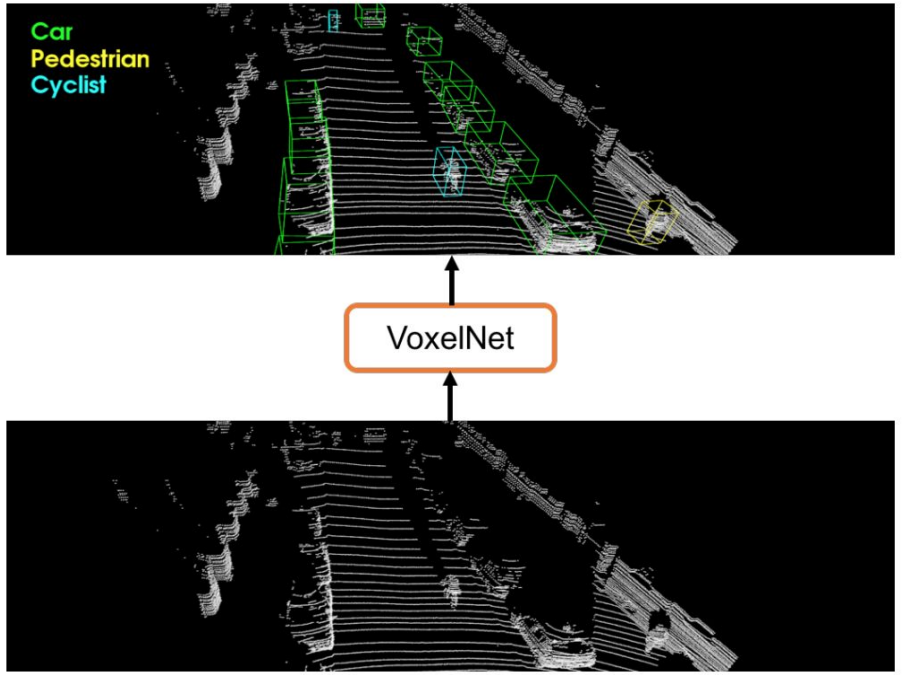
VoxNet



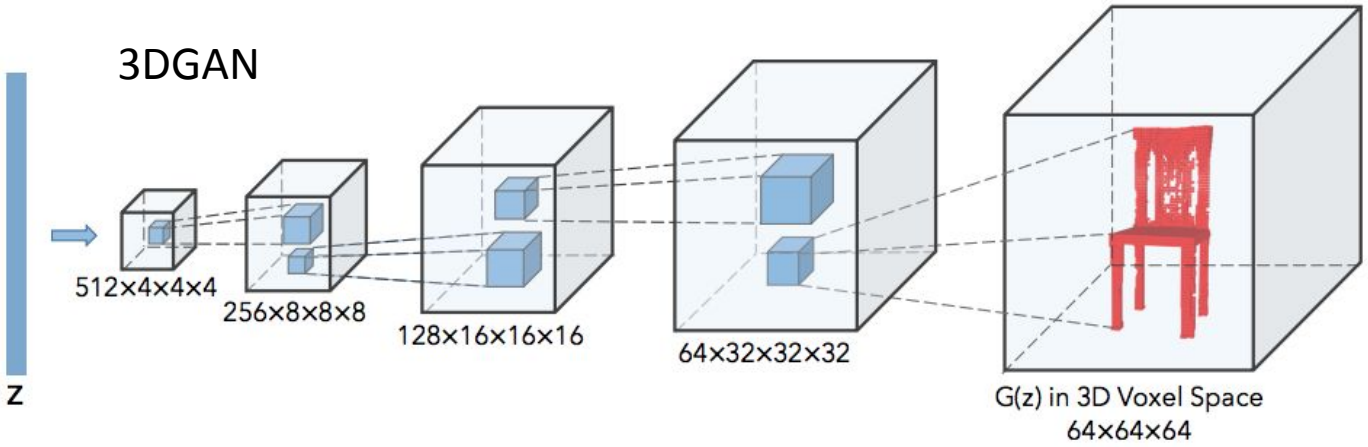
3D-R2N2



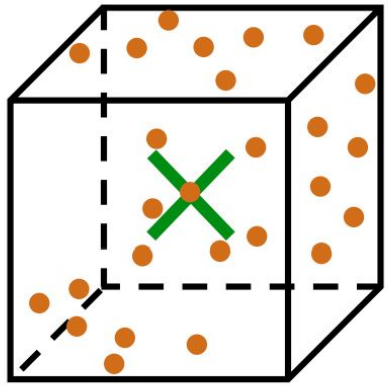
VoxelNet



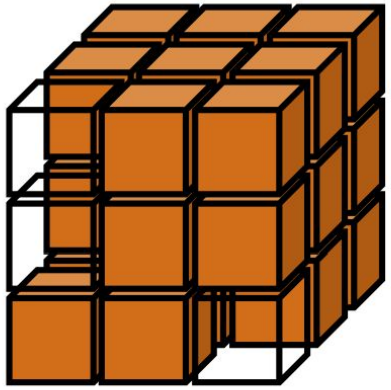
3DGAN



Volumetric-based: 3DCNN



Goal: classify center of neighbourhood (sliding 3D window)



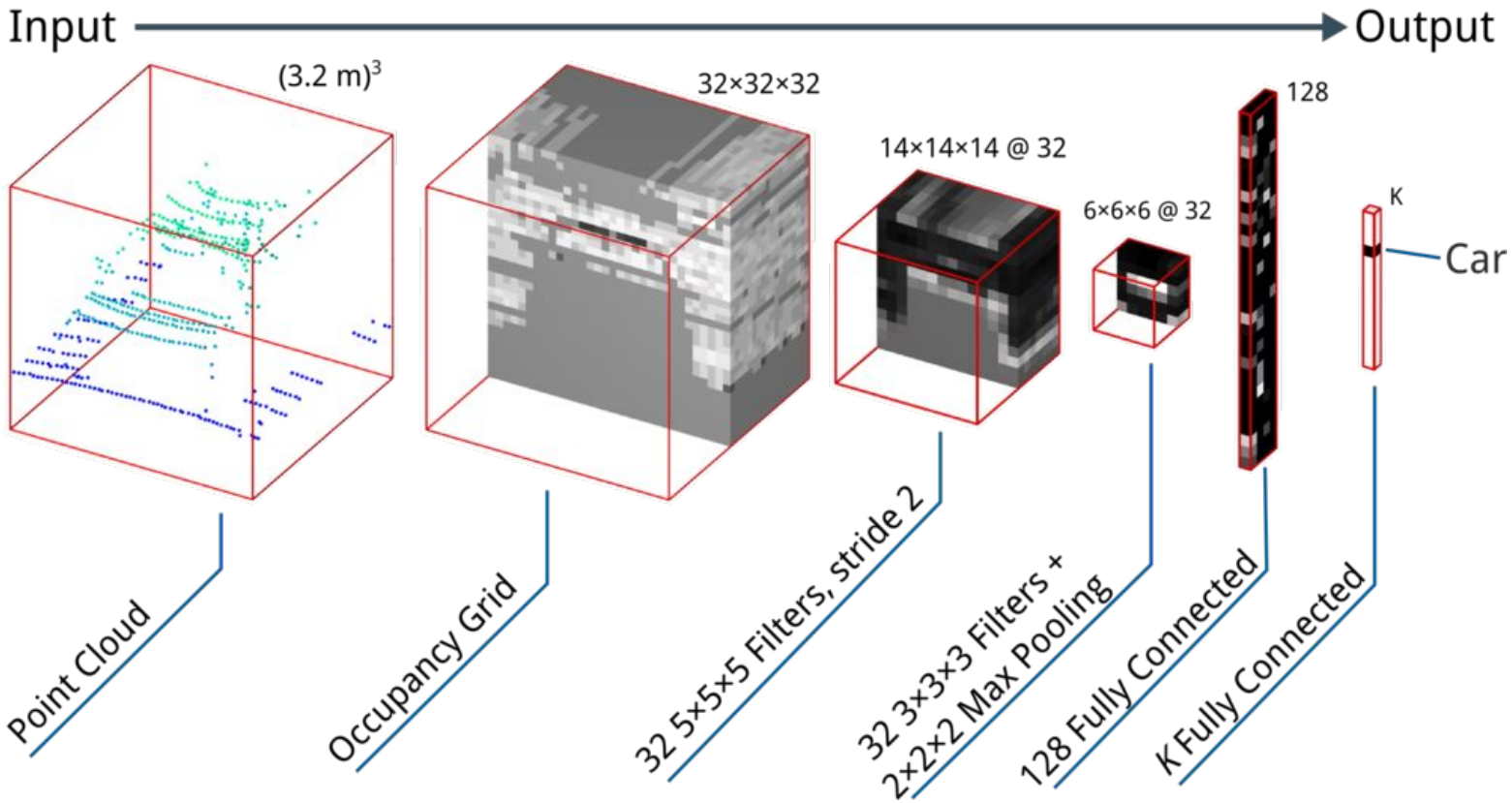
Voxel occupancy grid: any point inside voxel? (0/1)

3D Convolutional Neural Network

Semantic label

VoxNet

- Two components:
  - A volumetric grid representing estimate of spatial occupancy
  - 3D CNN that predicts a class label directly from the occupancy grid
- Each point cloud will be mapped into an occupancy grid
- Grid will then be passed through two 3D convolutional layers

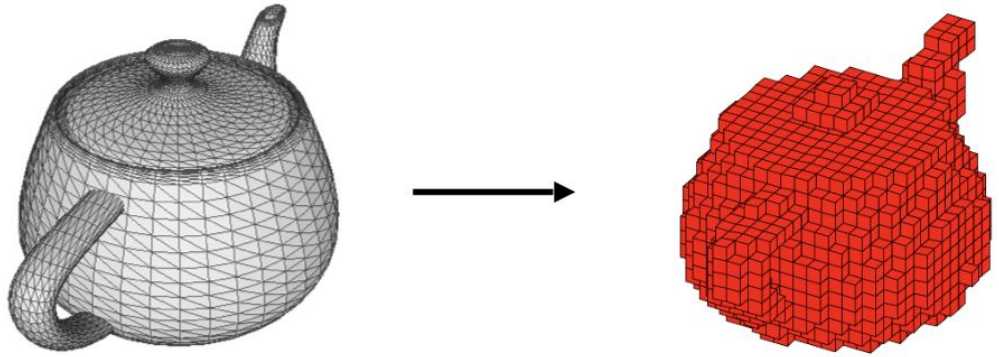


VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition , IROS 2015.

Problems with Voxelization

- Memory (1024x1024x1024x1024)
- Lots of zeros
- Resolution
- Quantization artifacts due to continuous to discrete conversion

Quantization Artifacts and Memory Issue

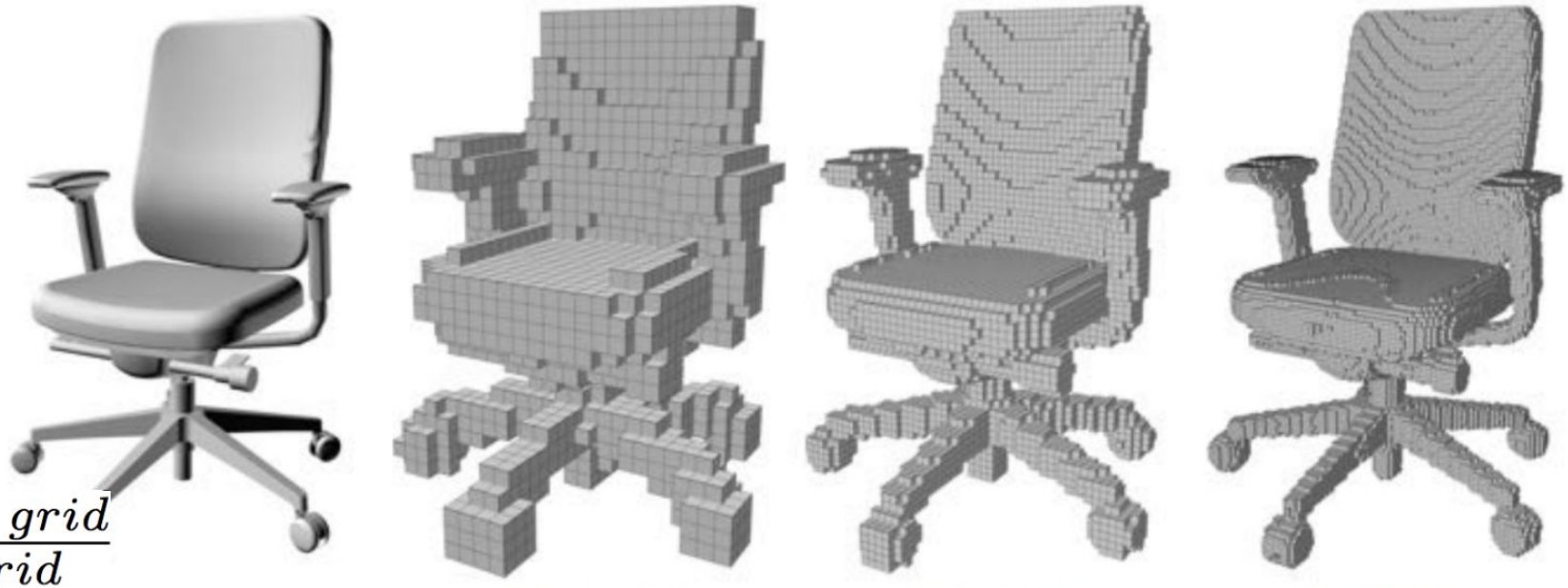


Polygon Mesh

Occupancy Grid
30x30x30

Information loss in voxelization

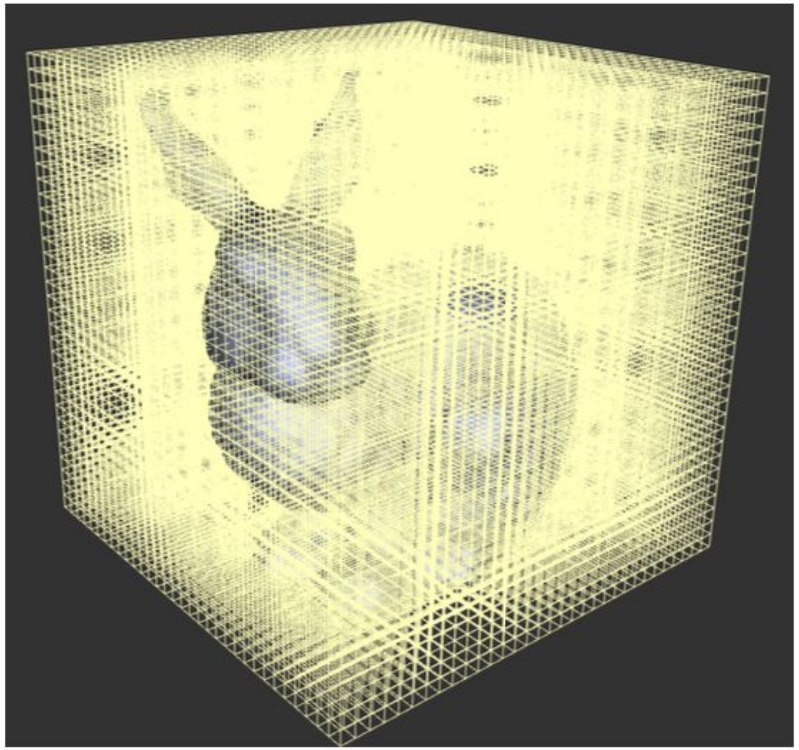
Increase Resolution, Leverage Sparsity

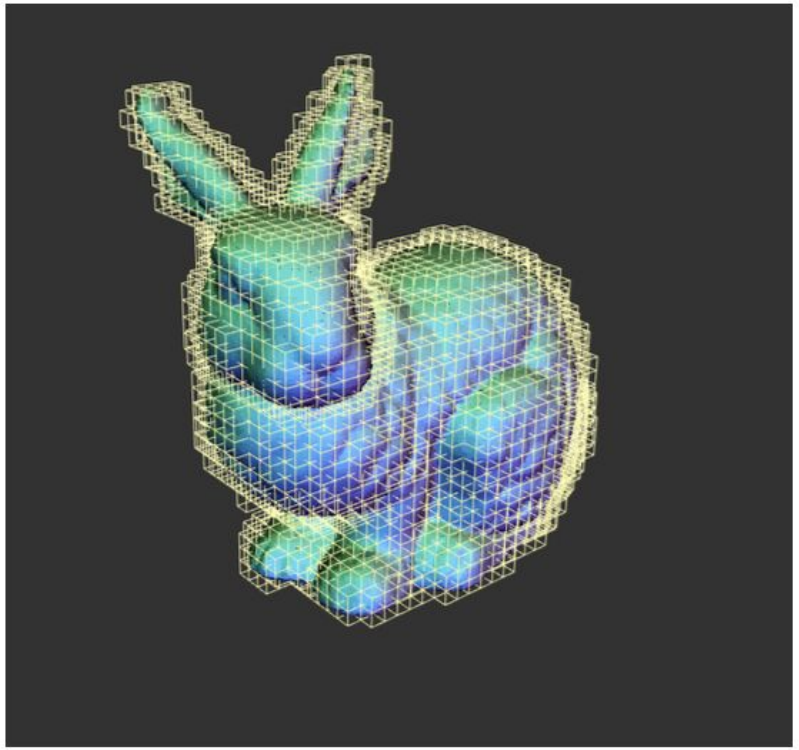


$$\frac{\#occupied\ grid}{\#total\ grid}$$

Occupancy:	10.41%	5.09%	2.41%
Resolution:	32	64	128

Store only the Occupied Grids





Research Question

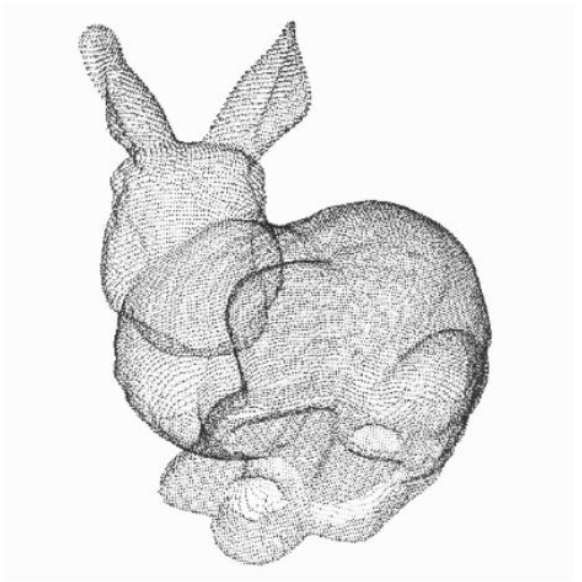
Can we achieve effective **feature learning directly** on point clouds?

Point Cloud

- The most common 3D sensor data
- Canonical model

LiDAR

Depth Sensor



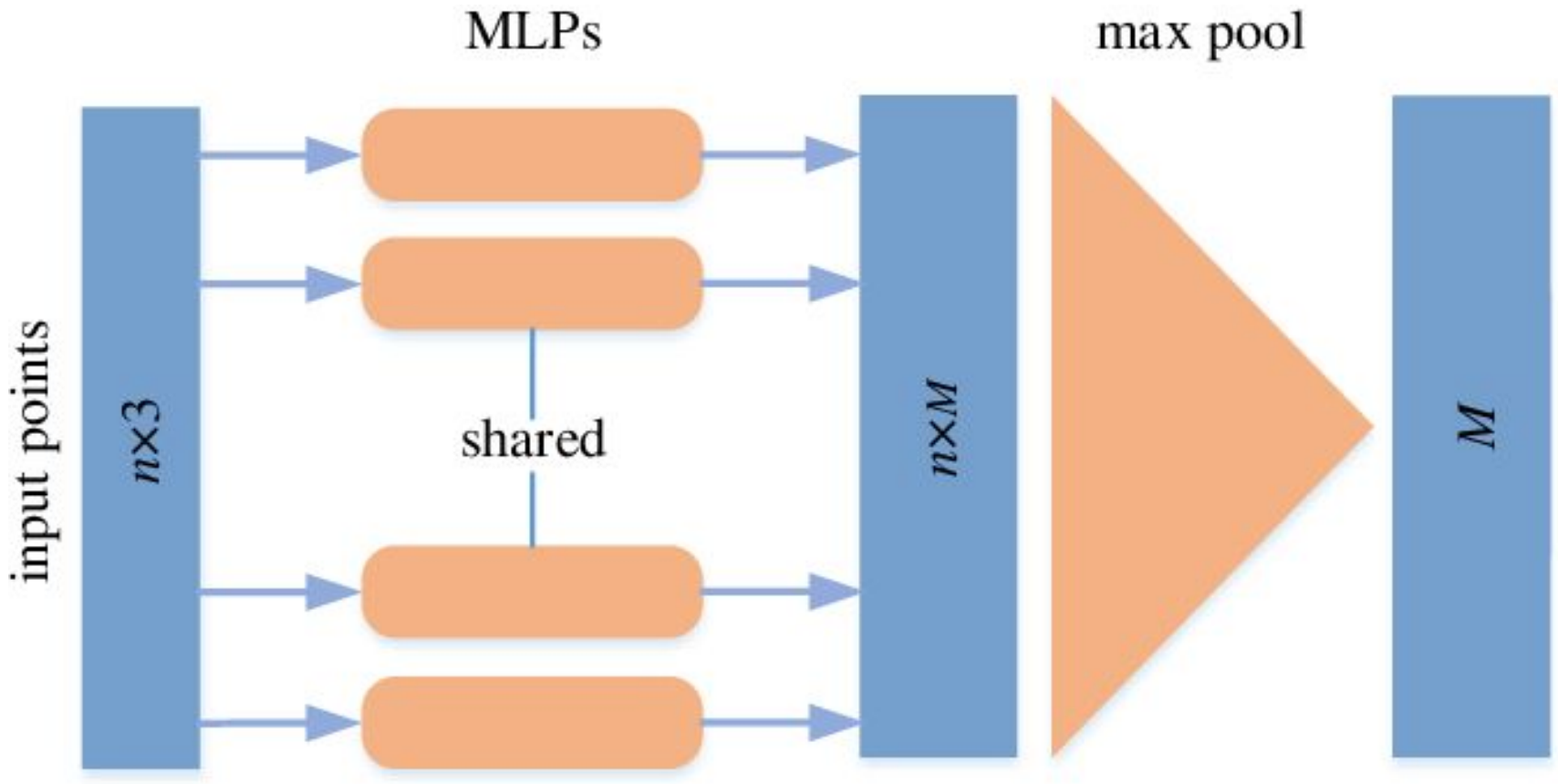
Point Cloud

Point-based Methods

3D Points -> Neural Nets

- PointNet, CVPR 2017
  - Directly manipulating raw point cloud data and considers global geometry. Achieves permutation invariance of points by operating on each point independently
- PointNet++, NeurIPS 2017
  - Hierarchical neural network that applies PointNet recursively on a nested partitioning of the input point set. Able to learn local features.

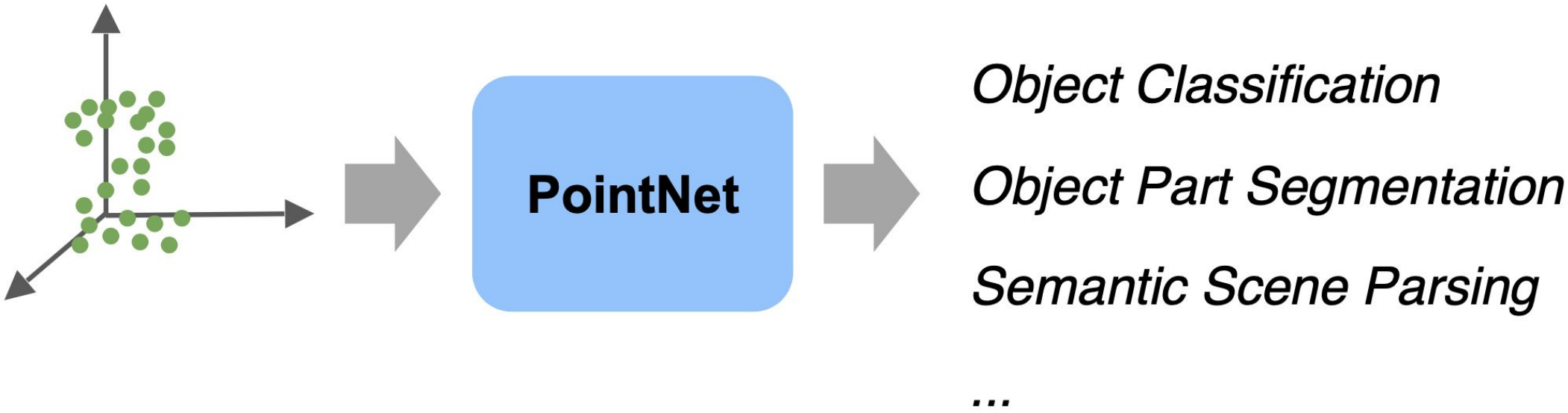
Point Cloud Learning Architecture



Guo, Yulan, et al. "**Deep learning for 3d point clouds: A survey**." *IEEE transactions on pattern analysis and machine intelligence* 2020

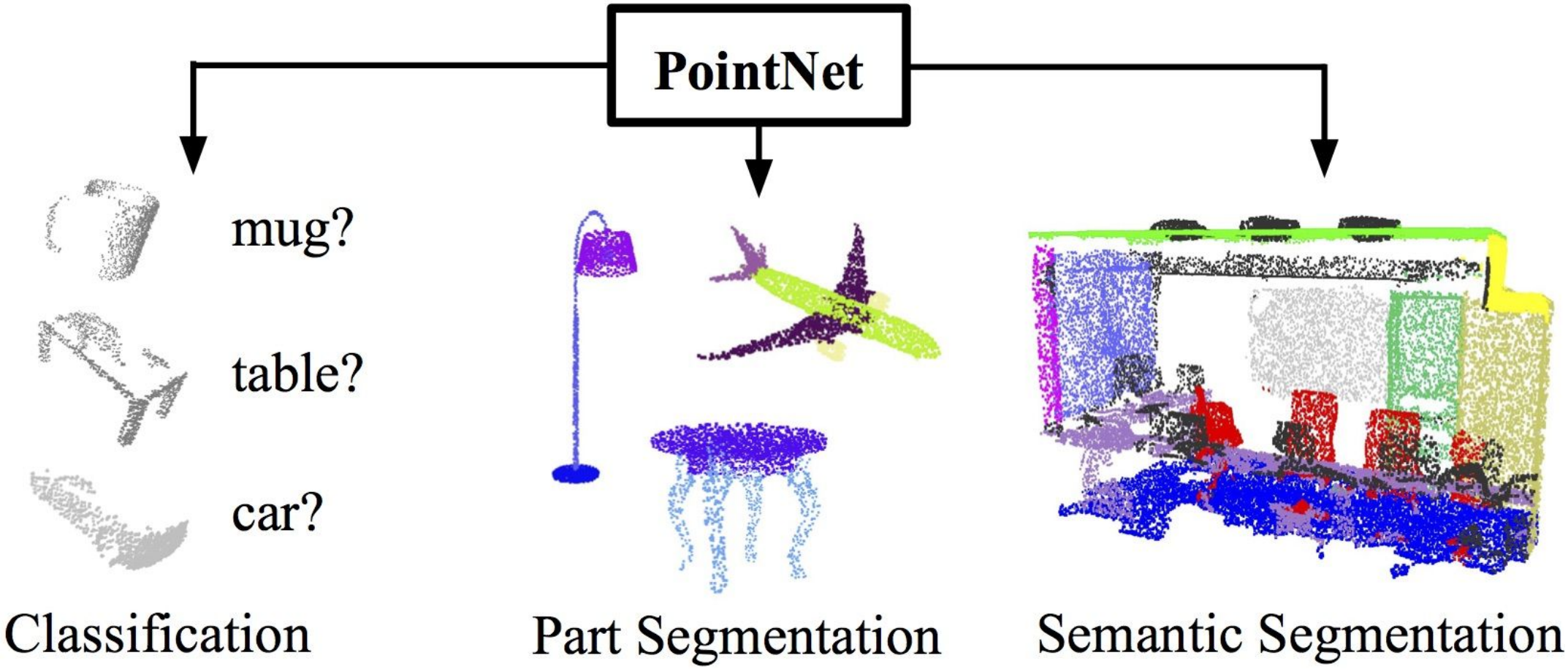
PointNet

- Learning directly on **scattered, unordered** point cloud



Qi, Charles R., et al. "**Pointnet: Deep learning on point sets for 3d classification and segmentation**", CVPR 2017

PointNet: Various Tasks



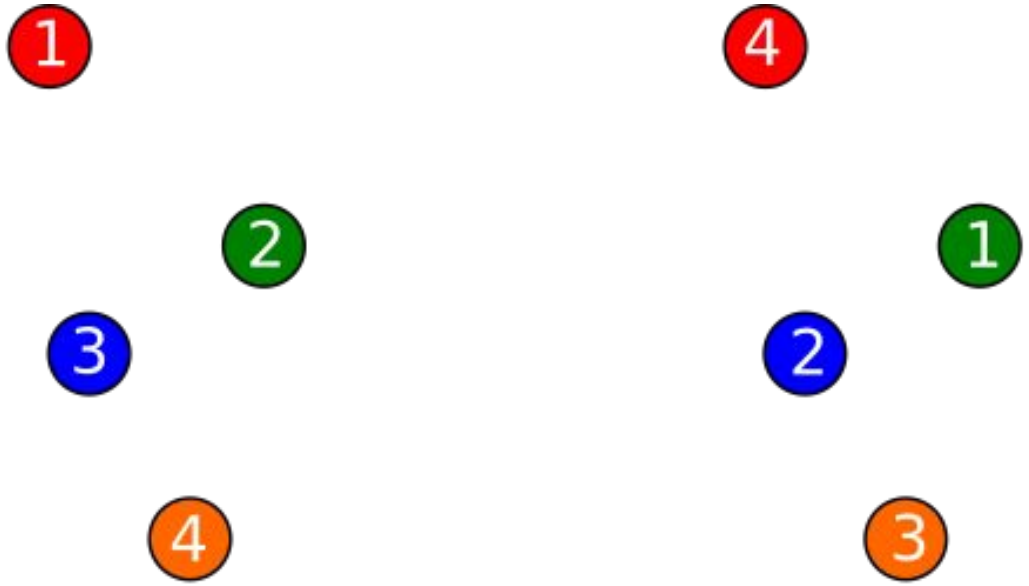
Challenges

- **Unordered point set as input**
  - Model needs to be invariant to N! permutations.

- **Invariance under geometric transformations**
  - Point cloud rotations should not alter classification results.

Unordered Point Set

- Point cloud: N orderless points, each represented by a D dim vector
- Model needs to be invariant to N! permutations



Permutation Invariance: Symmetric Function

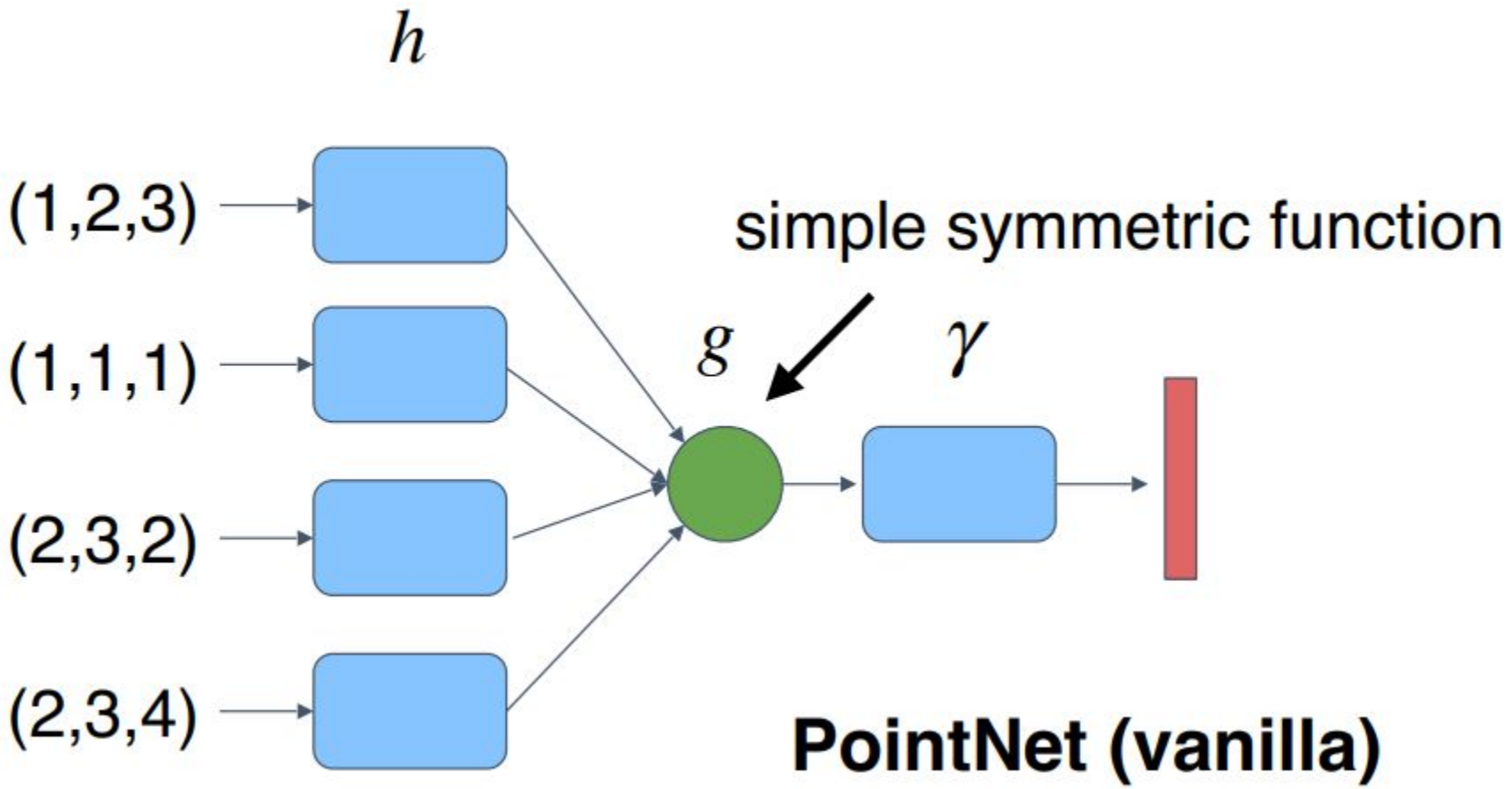
- How to construct a family of symmetric functions by neural networks?
- Examples:

$$f(x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\}$$

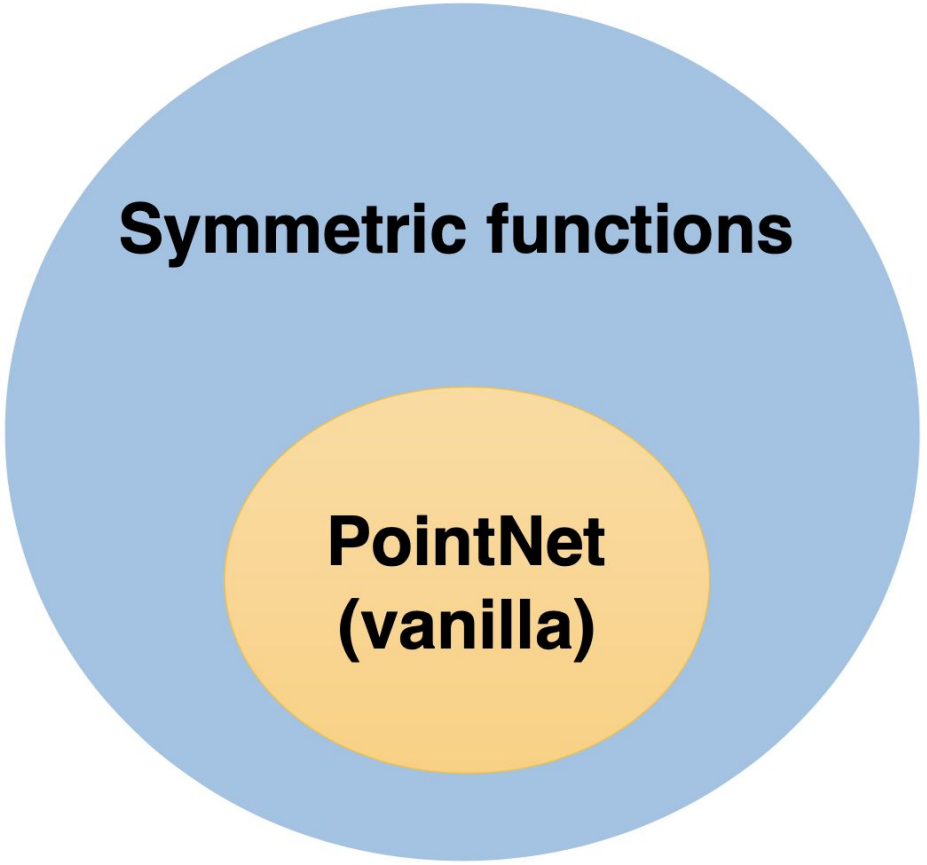
$$f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$$

Permutation Invariance: Symmetric Function

- f is symmetric if g is symmetric



Permutation Invariance: Symmetric Function



Universal Set Function Approximator

Theorem:

A Hausdorff continuous symmetric function $f : X \to \mathbb{R}$ can be arbitrarily approximated by PointNet.

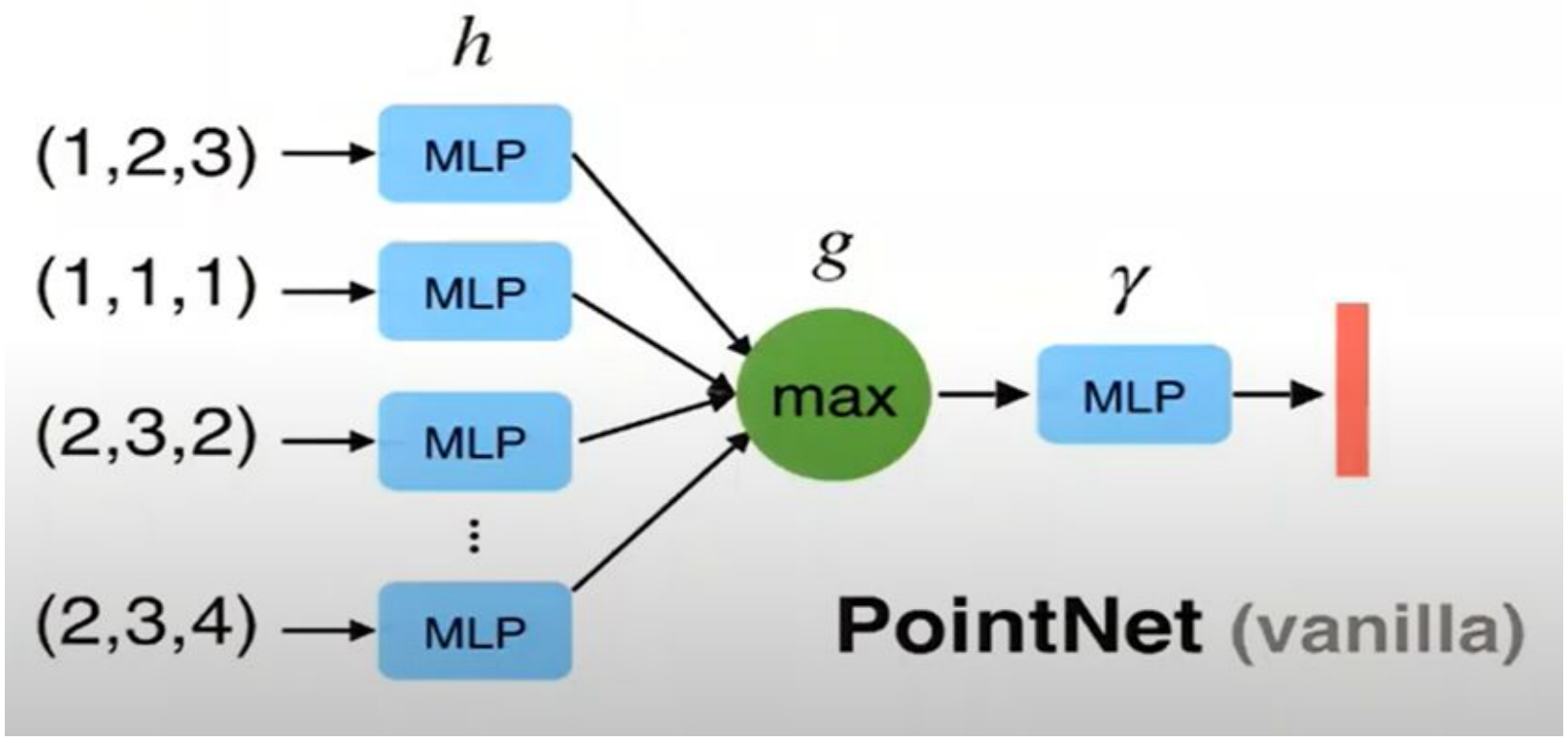
$$\left| f(S) - \gamma \left(\underset{x_i \in S}{\mathrm{MAX}} \{h(x_i)\} \right) \right| < \epsilon$$

$S \subseteq \mathbb{R}^d$

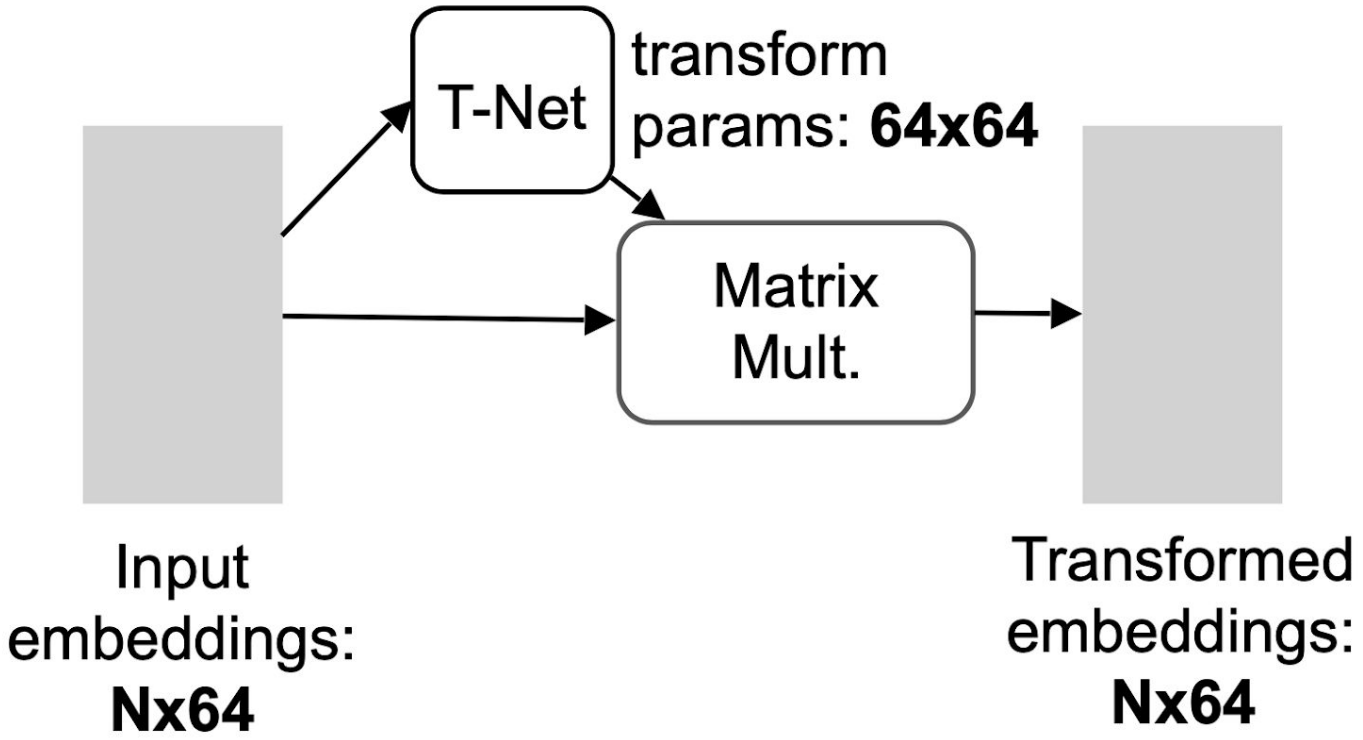
PointNet (vanilla)

Basic PointNet Architecture

- PointNet uses **multi-layer perceptron (MLP)** and **max pooling**:



Embedding Space Alignment

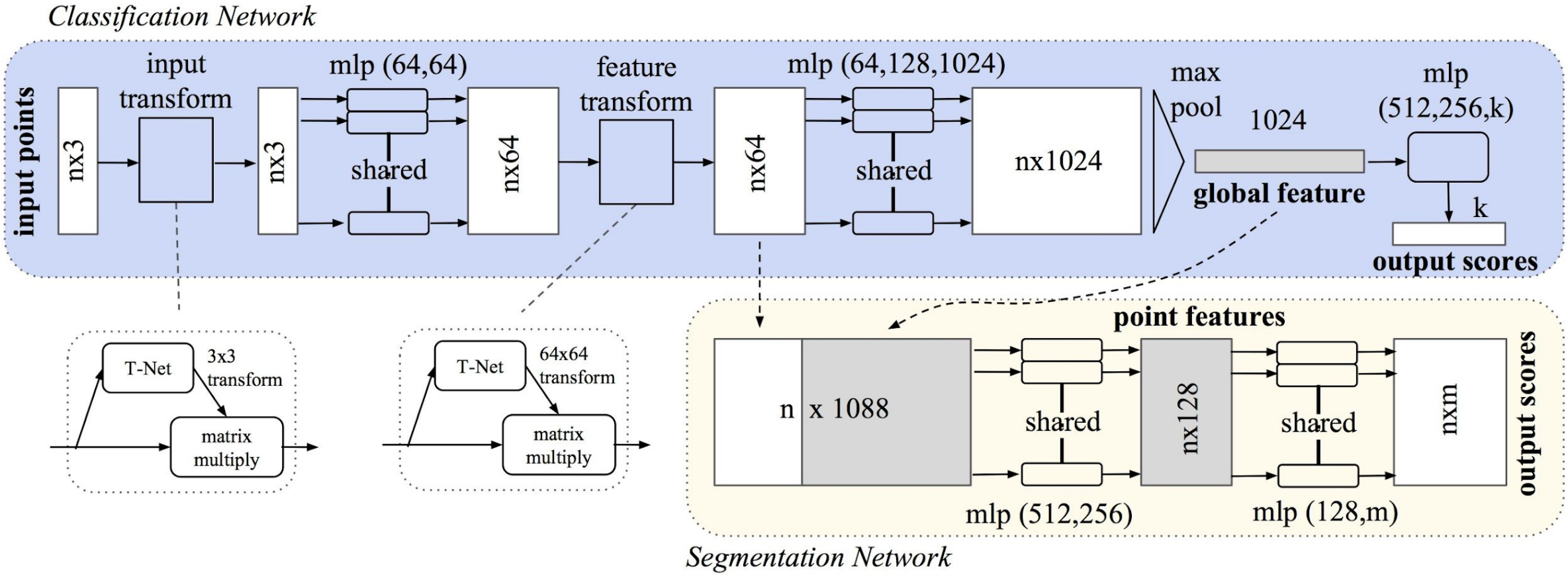


Regularization:

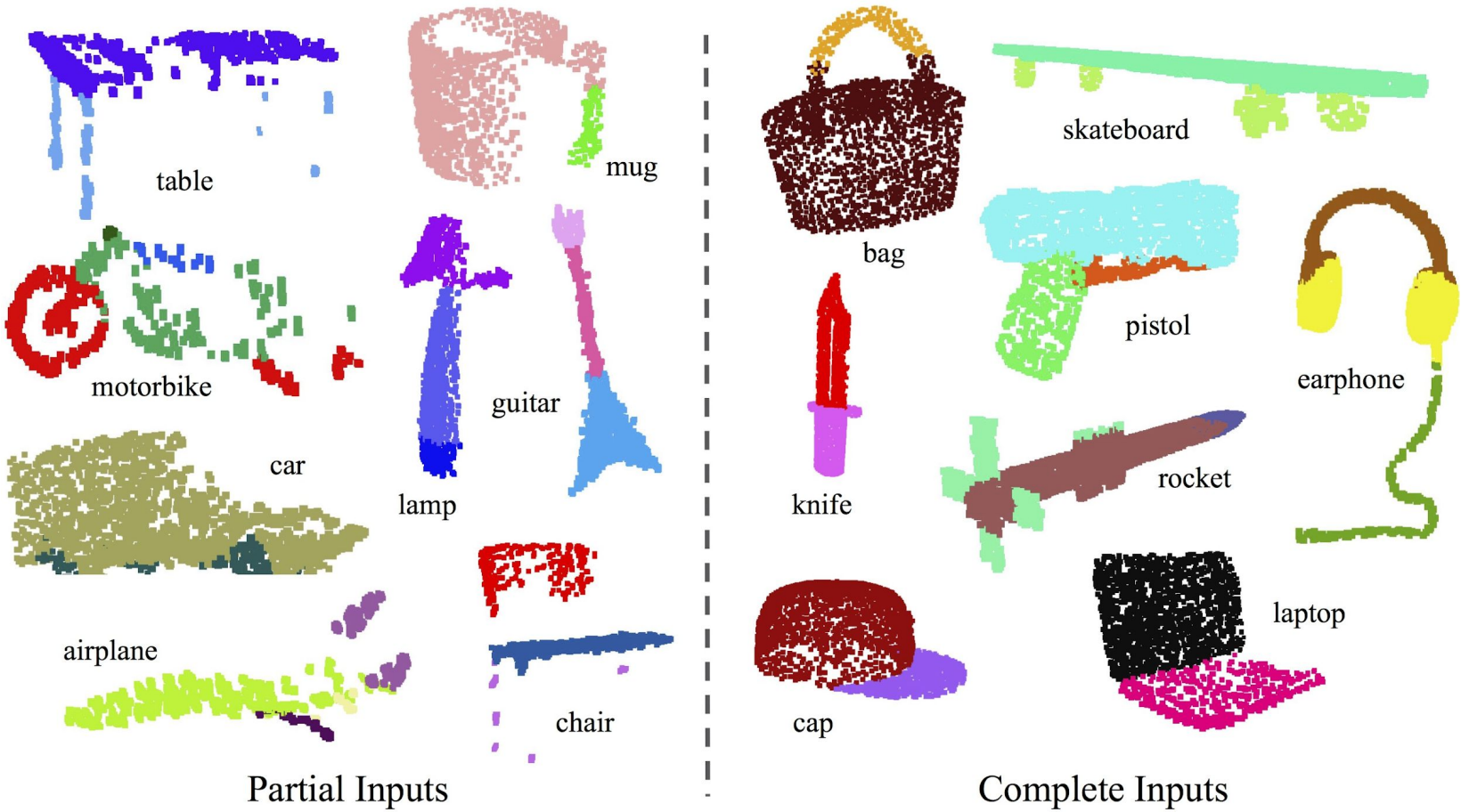
Transform matrix A 64x64 close to orthogonal:

$$L_{reg} = \|I - AA^T\|_F^2$$

PointNet: Classification & Segmentation Network



Part Segmentation

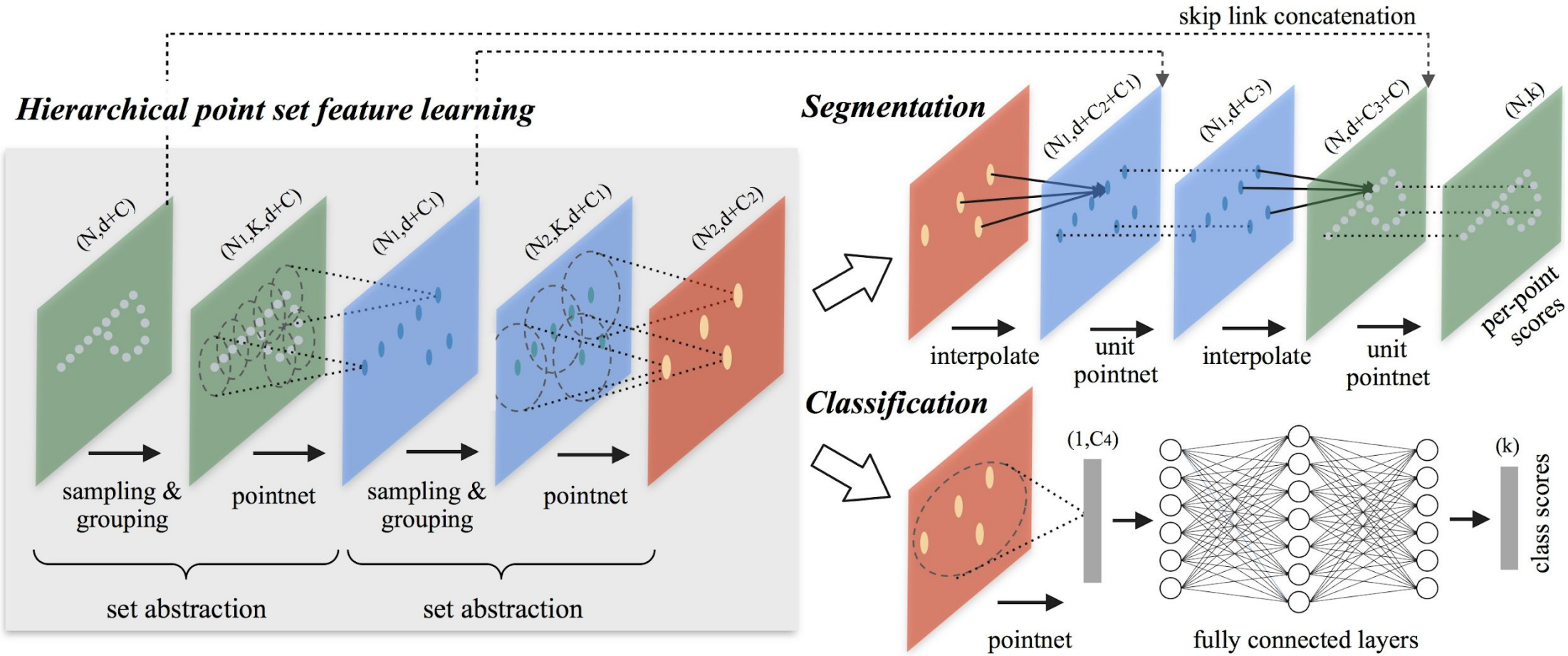


Semantic Scene Segmentation

Limitations of PointNet

- No local context for each point!
  - Cannot capture local information
- Works on individual points
- Ignores the geometric cues
- Global feature depends on absolute coordinate. Hard to generalize to unseen scene configurations!

PointNet++



Qi, Charles R., et al. **"PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space."**, NeurIPS 2017

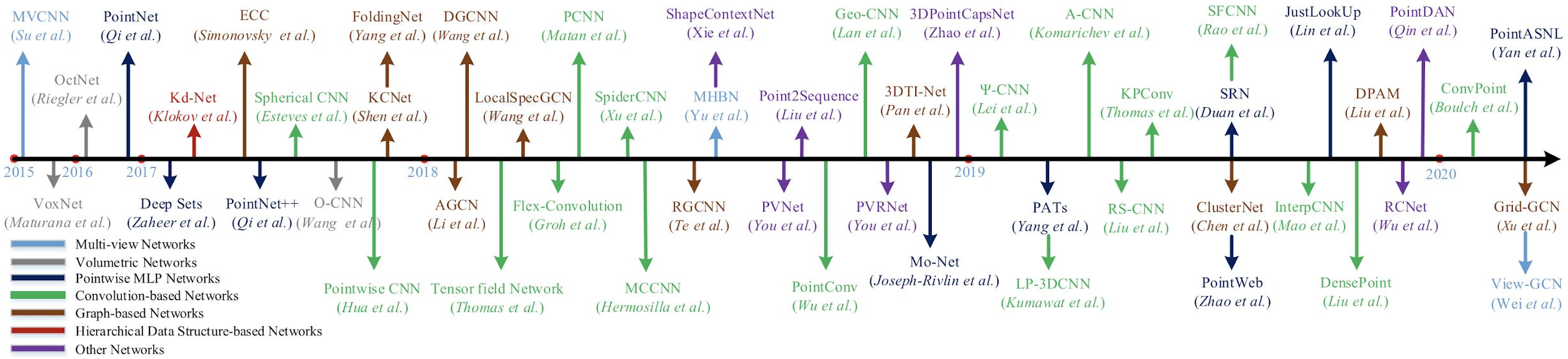
Point Cloud Processing

- Normal Estimation
  - Important properties of a geometric surface
  - How to compute normals directly on points?
    - Obtain surface and compute!
    - Approximate by fitting the plane!
- Sampling
  - Upsampling
  - Downsampling
    - Needed for large point clouds
    - Uniform sampling
    - FPS (Farthest Point Sampling)
- Filtering
  - Outlier removal
  - Denoising
  - Get important points: boundary points?

Learning types

- Supervised
- Unsupervised
- Self-supervised
- Few-shot

Existing Methods



Guo, Yulan, et al. "**Deep learning for 3d point clouds: A survey**." *IEEE transactions on pattern analysis and machine intelligence* 2020

Resources: Github: awesome-point-cloud-analysis

awesome-point-cloud-analysis 

for anyone who wants to do research about 3D point cloud.

If you find the awesome paper/code/dataset or have some suggestions, please contact hualin.vvv@gmail.com.
Thanks for your valuable contribution to the research community 😃

For more recent papers, please visit awesome-point-cloud-analysis-2020

```
- Recent papers (from 2017)
```

Keywords

`dat.` : dataset | `cls.` : classification | `rel.` : retrieval | `seg.` : segmentation
`det.` : detection | `tra.` : tracking | `pos.` : pose | `dep.` : depth
`reg.` : registration | `rec.` : reconstruction | `aut.` : autonomous driving
`oth.` : other, including normal-related, correspondence, mapping, matching, alignment, compression, generative model...

Statistics: 🔥 code is available & stars >= 100 | ⭐ citation >= 50

2017

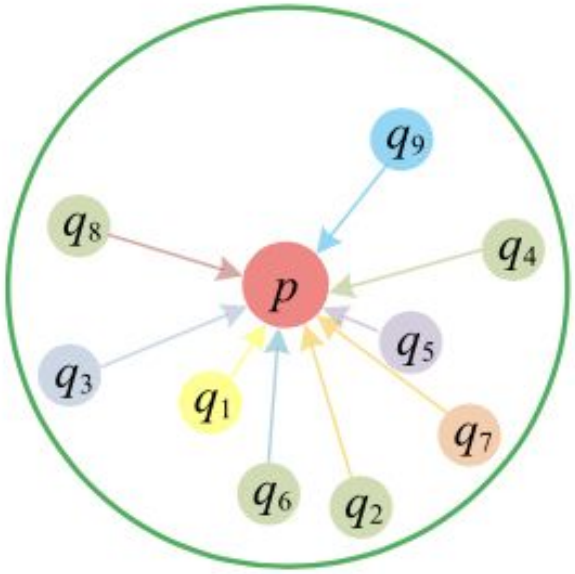
- [CVPR] PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. [tensorflow][pytorch] [`cls.` `seg.` `det.`] 🔥⭐
- [CVPR] Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. [`cls.`] ⭐

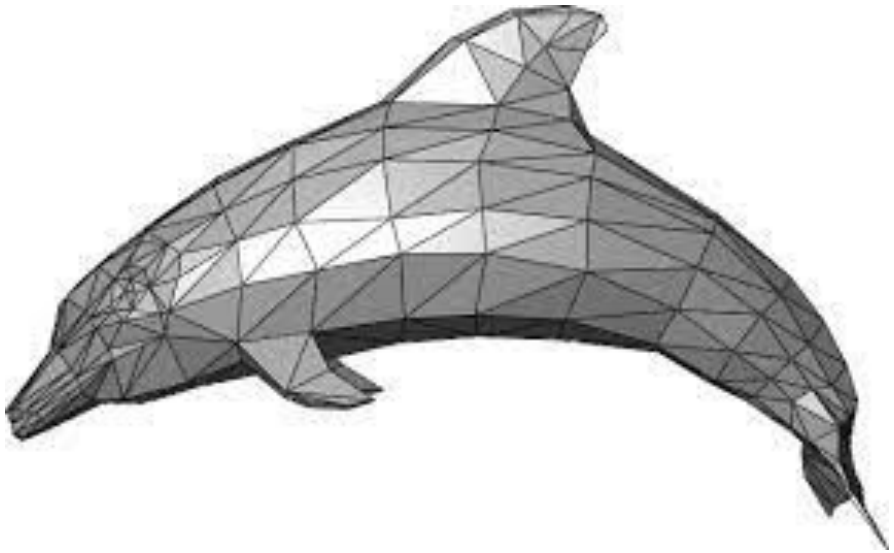
Deep Learning for Graphs & Meshes for 3D Data

Charu Sharma

Assistant Professor

Machine Learning Lab, IIIT Hyderabad

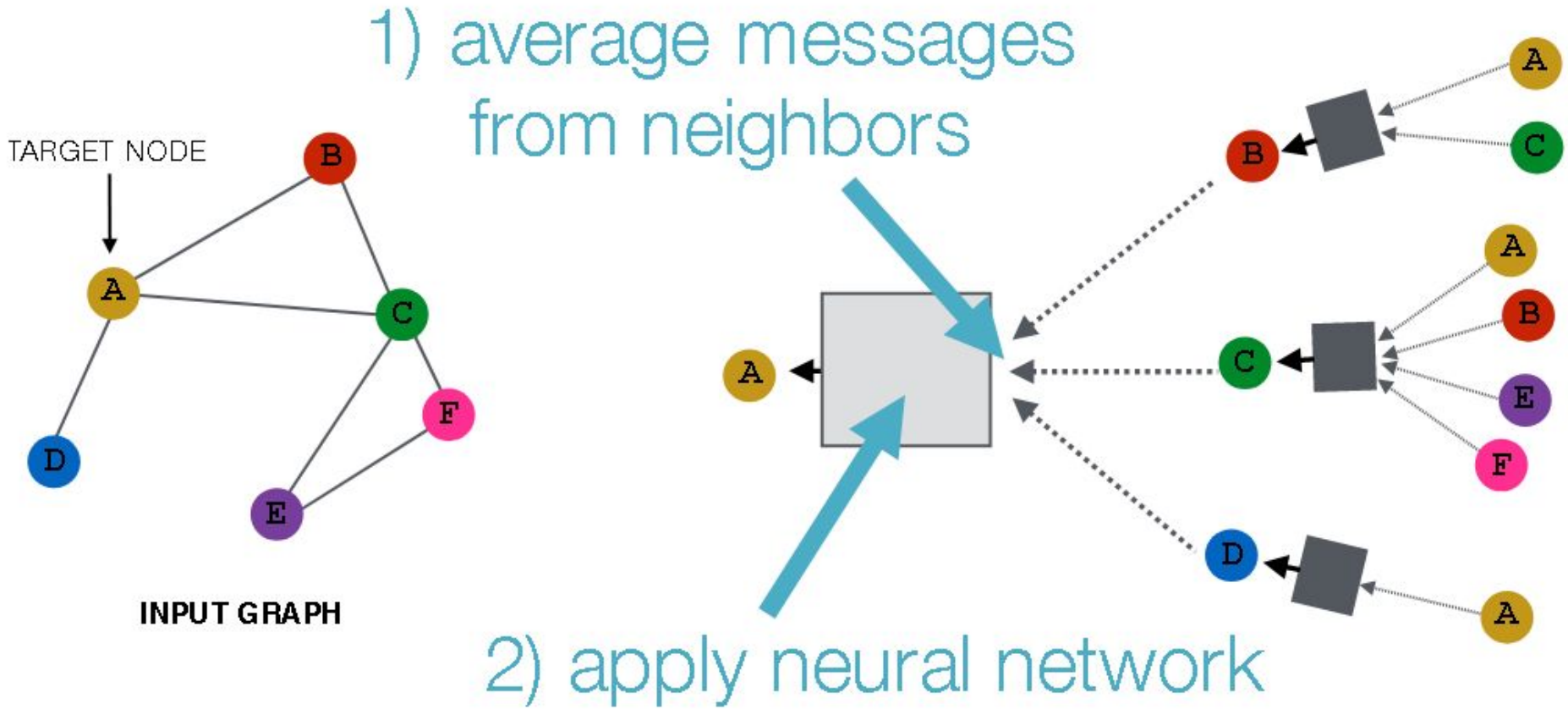




Generalizing convolutions to irregular graphs?

- General Perspective on GNNs
  - Message Passing Mechanism
  - GCN (Graph Convolutional Networks)

General Perspective on GNNs



Graph-based Approaches for Point Clouds

- Points -> Nodes
- Neighborhood -> Edges
- Point Convolution as Graph Convolution
- Graph CNN for point cloud processing

- DGCNN
  - Graph-based Point Cloud Learning Architecture
  - Computes graph on point clouds
  - Constructs graph dynamically
  - The local structure itself is informative.

Wang et al., "**Dynamic Graph CNN for Learning on Point Clouds**", *Transactions on Graphics, 2019*

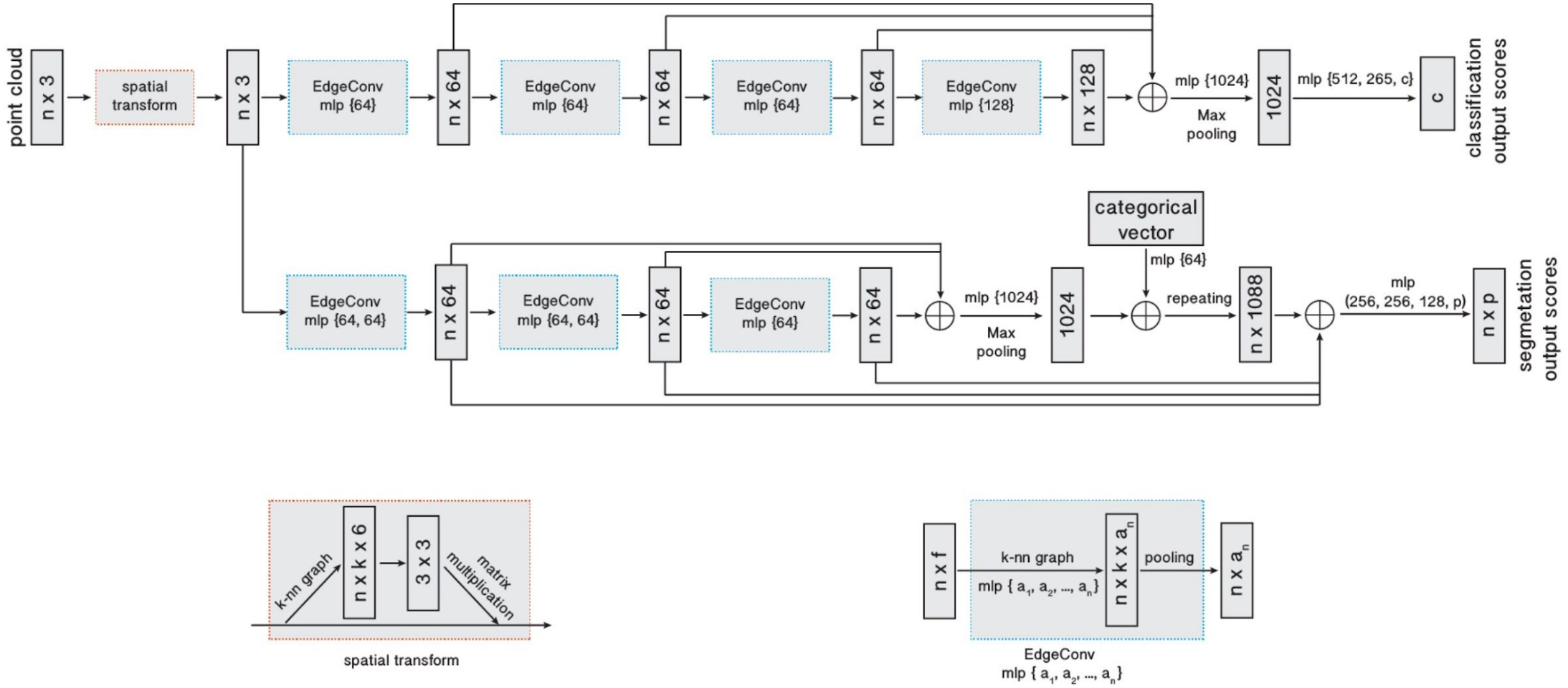


Fig. 3. **Model architectures:** The model architectures used for classification (top branch) and segmentation (bottom branch). The classification model takes as input n points, calculates an edge feature set of size k for each point at an EdgeConv layer, and aggregates features within each set to compute EdgeConv responses for corresponding points. The output features of the last EdgeConv layer are aggregated globally to form an $1D$ global descriptor, which is used to generate classification scores for c classes. The segmentation model extends the classification model by concatenating the $1D$ global descriptor and all the EdgeConv outputs (serving as local descriptors) for each point. It outputs per-point classification scores for p semantic labels. $\oplus$: concatenation. **Point cloud transform block:** The point cloud transform block is designed to align an input point set to a canonical space by applying an estimated 3×3 matrix. To estimate the 3×3 matrix, a tensor concatenating the coordinates of each point and the coordinate differences between its k neighboring points is used. **EdgeConv block:** The EdgeConv block takes as input a tensor of shape $n \times f$, computes edge features for each point by applying a multi-layer perceptron (mlp) with the number of layer neurons defined as $\{a_1, a_2, \ldots, a_n\}$, and generates a tensor of shape $n \times a_n$ after pooling among neighboring edge features.

EdgeConv Operation

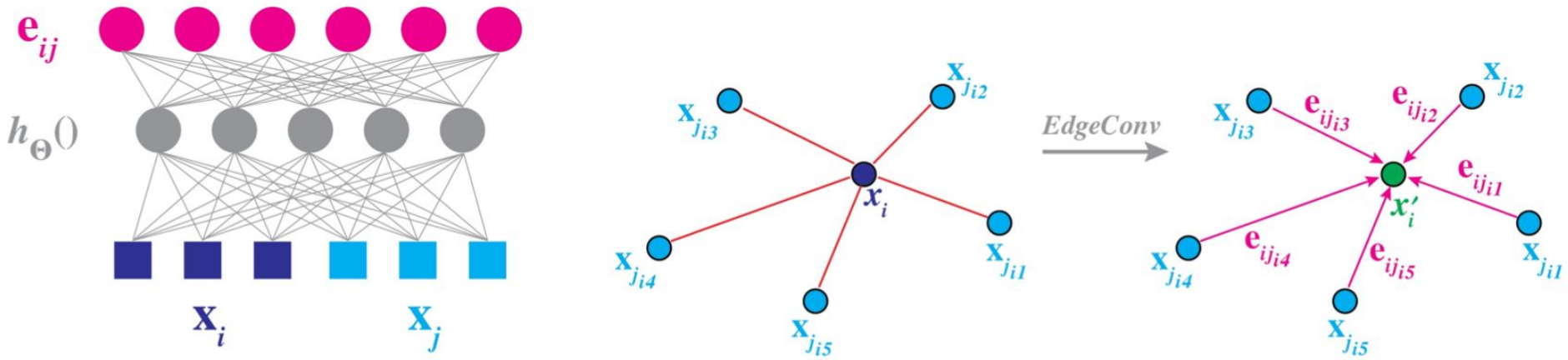


Fig. 2. **Left**: Computing an edge feature, $\mathbf{e}_{ij}$ (top), from a point pair, $\mathbf{x}_i$ and $\mathbf{x}_j$ (bottom). In this example, $h_\Theta()$ is instantiated using a fully connected layer, and the learnable parameters are its associated weights. **Right**: The EdgeConv operation. The output of EdgeConv is calculated by aggregating the edge features associated with all the edges emanating from each connected vertex.

- Concatenate the point features, pass the concatenated feature through a MLP to get edge features
- A symmetric aggregation function is applied upon the edge features to get new point feature

EdgeConv Operation

- Input point cloud / features in the intermediate layers.

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathbb{R}^F$$

- A k-nearest neighbor graph (only nodes that are kNNs are connected)

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

- Edge features, where h is a nonlinear function with learnable parameters

$$\boldsymbol{e}_{ij} = h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j)$$

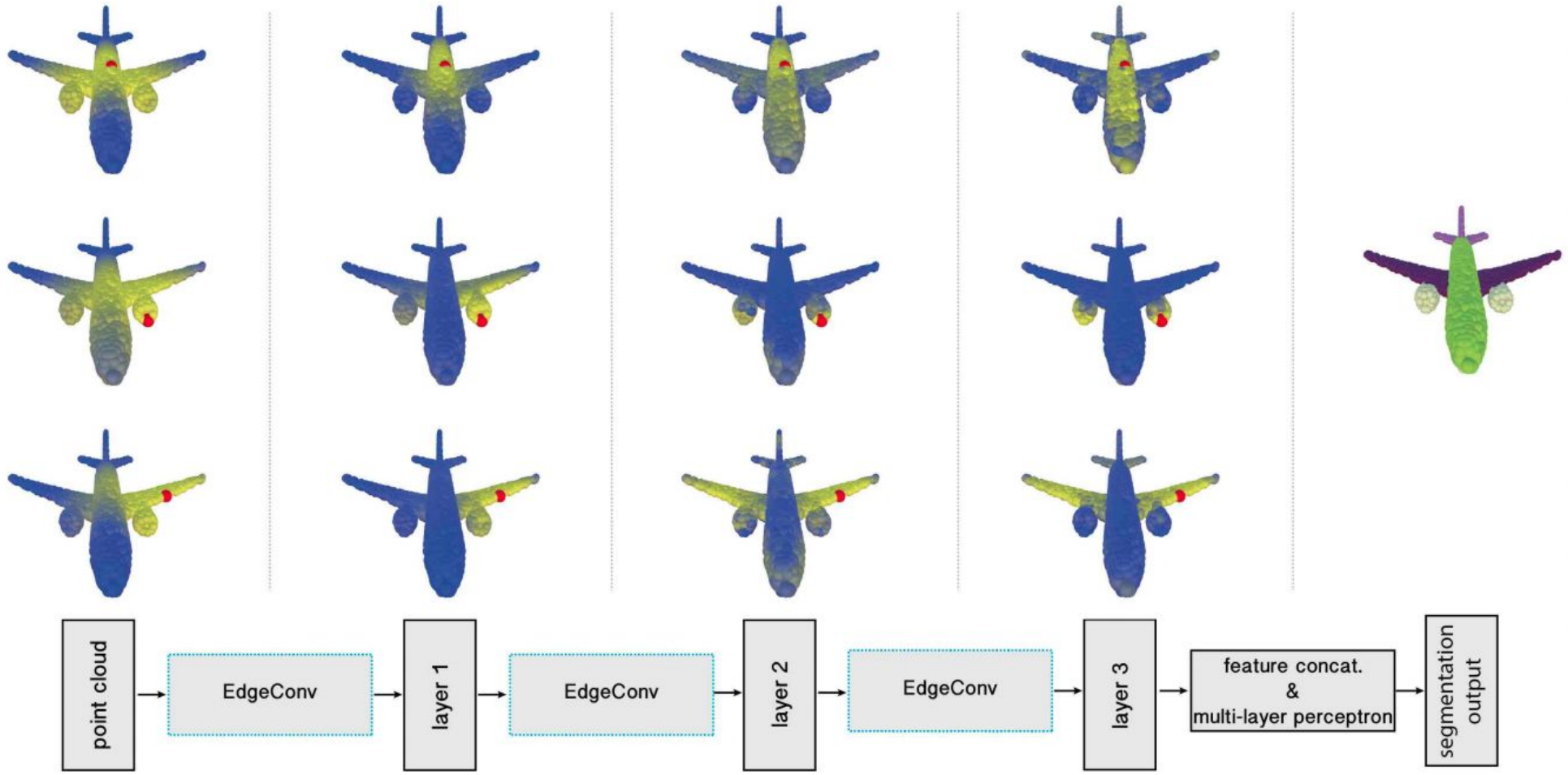
- An aggregation function, can be sum or max

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

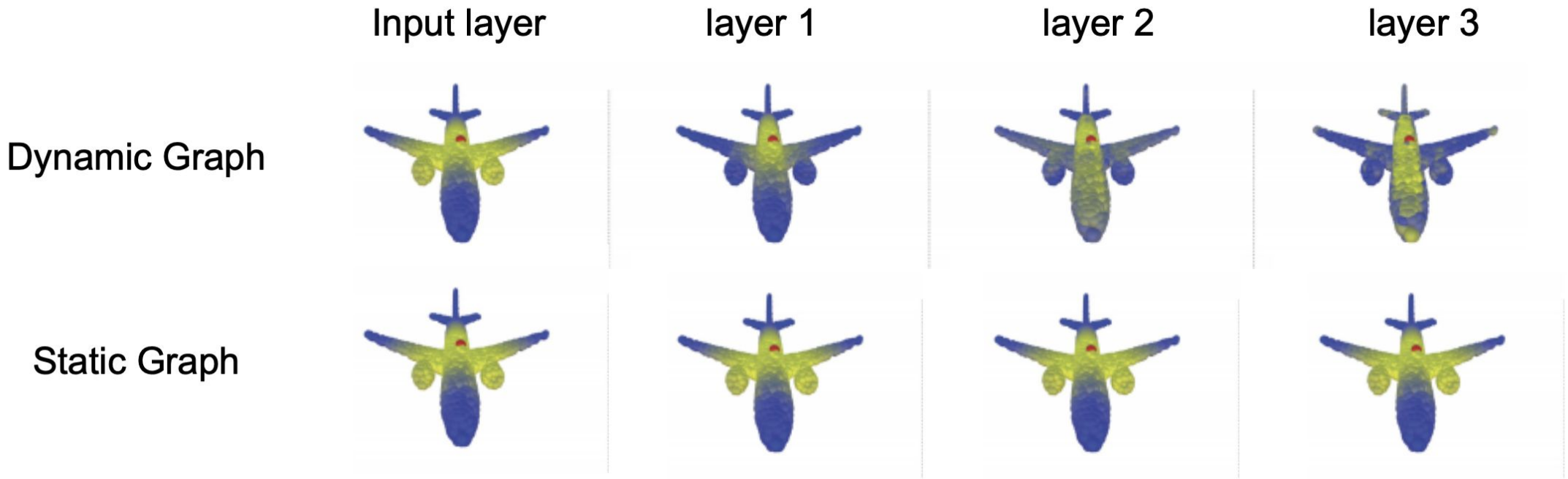
EdgeConv

- EdgeConv extracts local information for **every point.**
- EdgeConv is beyond the coordinate space, instead, **it works in the feature space** so that it can draw connection between semantically close points.
- EdgeConv is **permutation invariant and easy to implement** because it only involves kNN and symmetric operation.
- By simply plug in EdgeConvs into a network, we get Dynamic Graph CNN (DGCNN).
- **Dynamic Graph!**
  - The authors find it beneficial to recompute the graph using the nearest neighbors in the feature space produced by each layer.
  - A **new graph, instead of a static graph** for each layer!
  - A major difference from previous graph CNNs.

DGCNN: Learning in Layers



Why Dynamic Graph?



In later stages, DGCNN learns to gather semantically similar points, despite the large distance in the original coordinate space.

Main Results

	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**

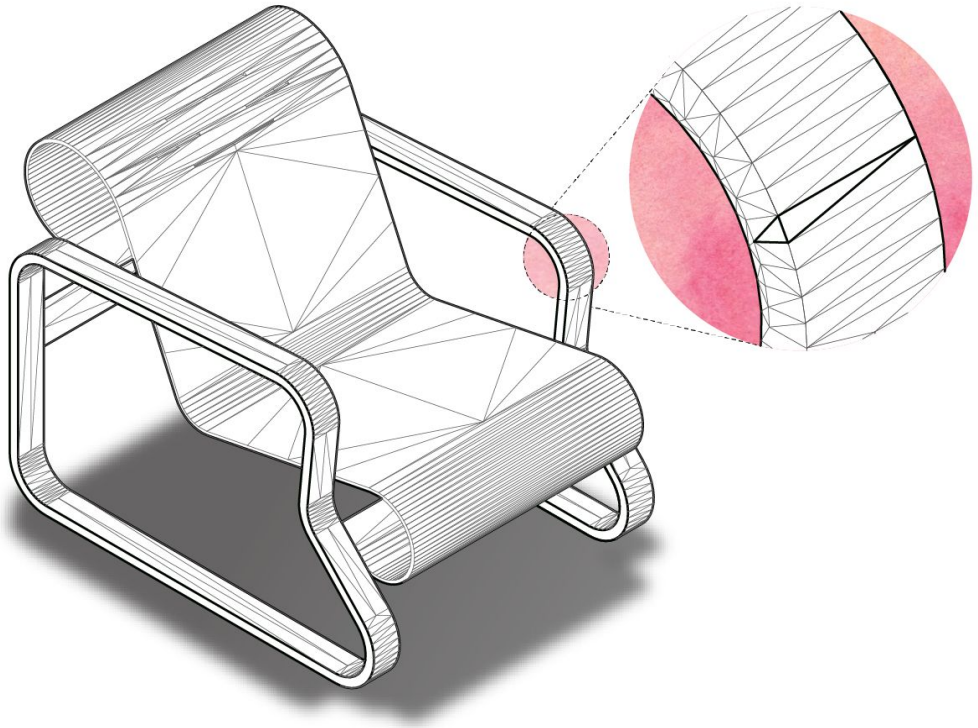
Table 2. Classification results on ModelNet40.

Limitations of DGCNN

- GPU Memory Consumption:
  - Construct the graph requires to compute all pairwise distances between all the points
  - Very high GPU memory consumption
- Varying Density:
  - Point clouds can have varying density at different positions; a fixed *k* may be not flexible enough
- Large-scale Point Cloud:
  - DGCNN does not compress the number of points during forward pass

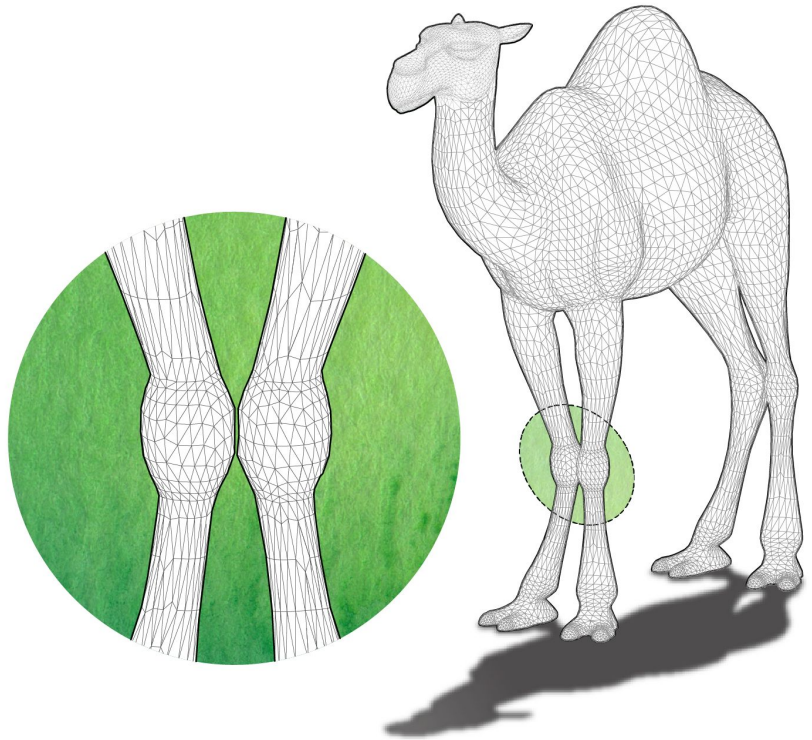
Meshes are Irregular

- Adapt to the surface
  - Large polygons in flat regions
  - Small polygons in detailed regions
- Non-Uniform & efficient representation for 3D shapes
- Capture both shape surface and topology.



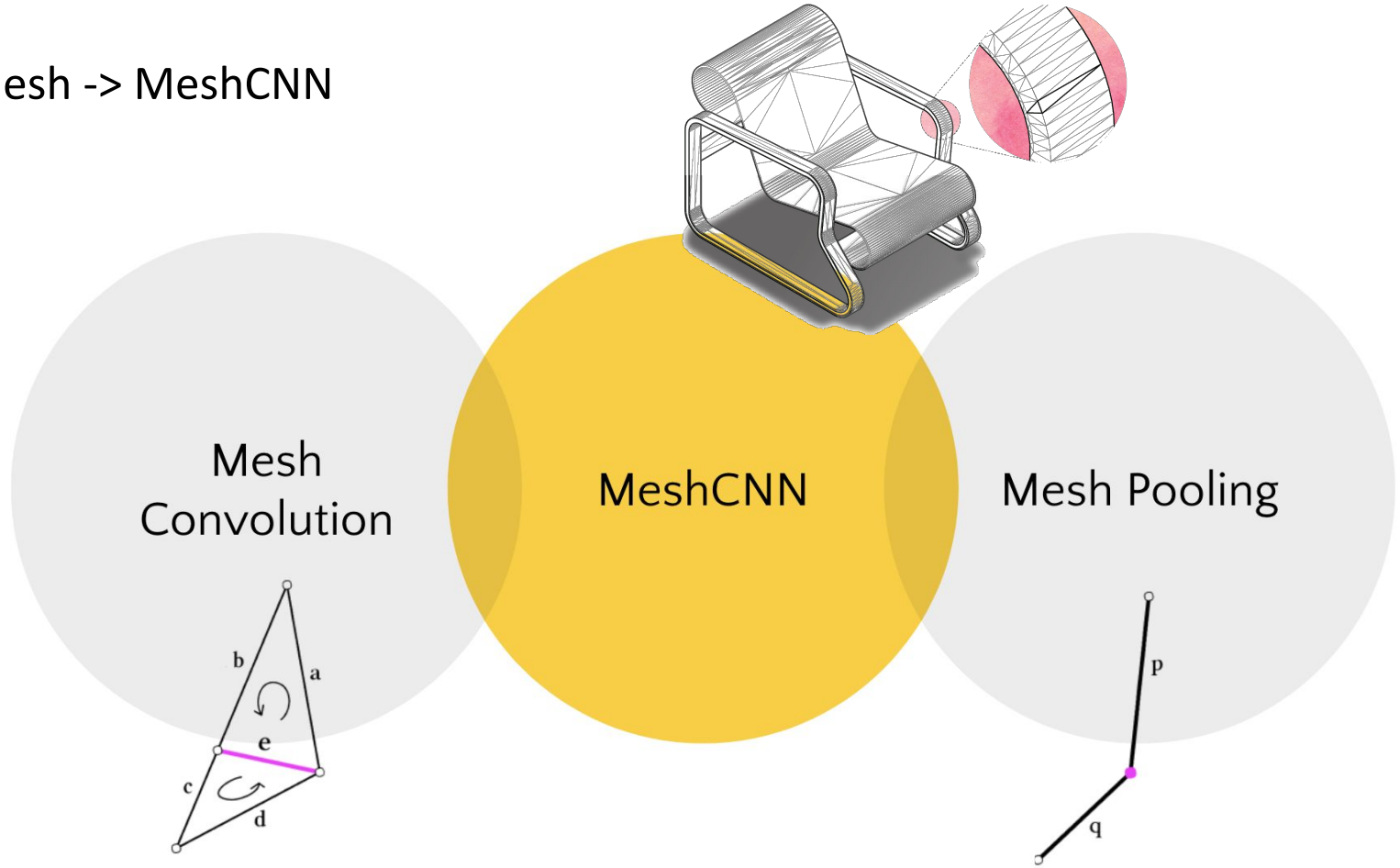
Meshes contain Topology

- Accurate portrayal of shape structure
  - Geodesic proximity
  - Disambiguate nearby surfaces



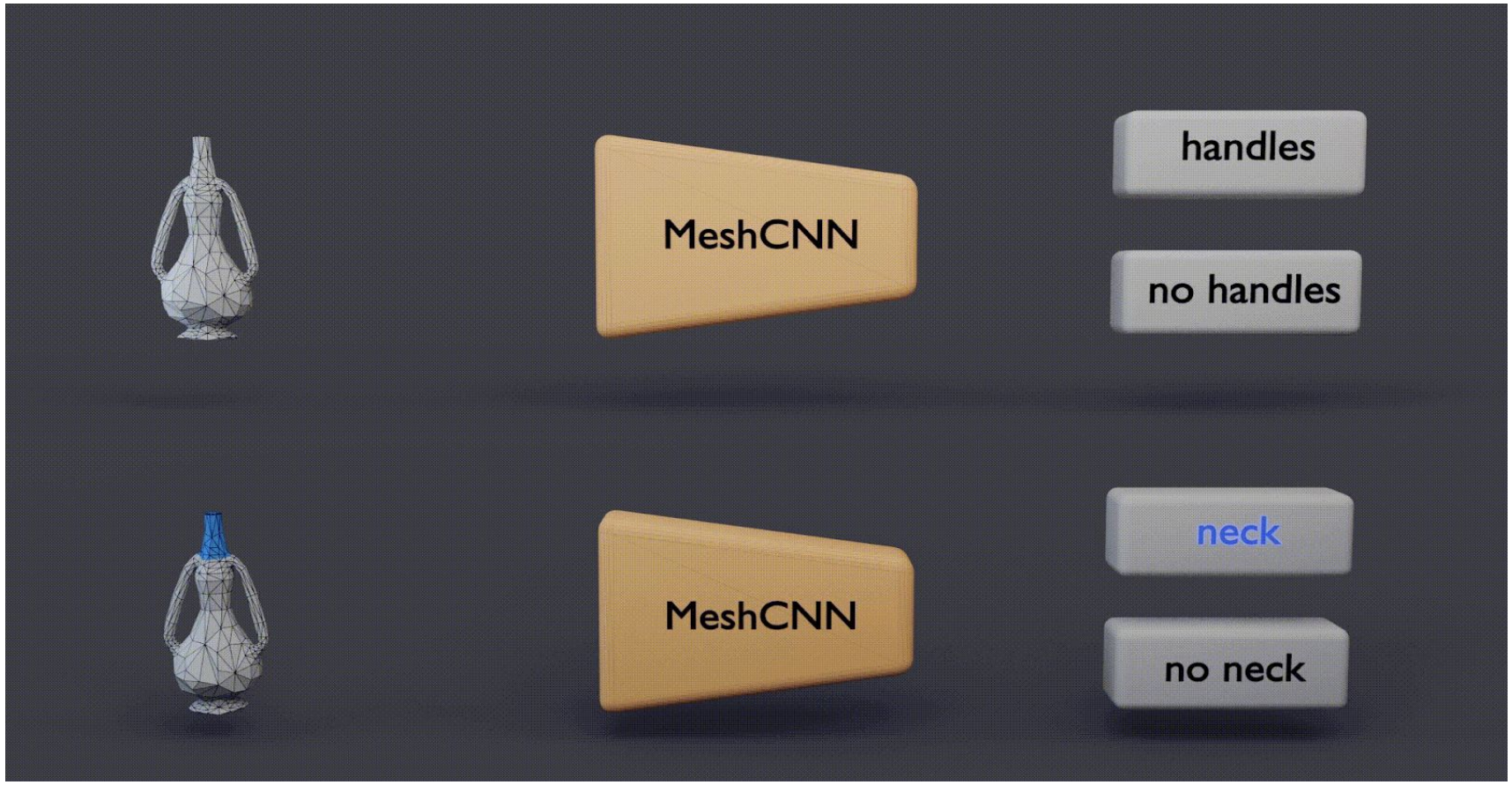
Goal: CNN directly on the irregular mesh elements

Points -> Mesh -> MeshCNN



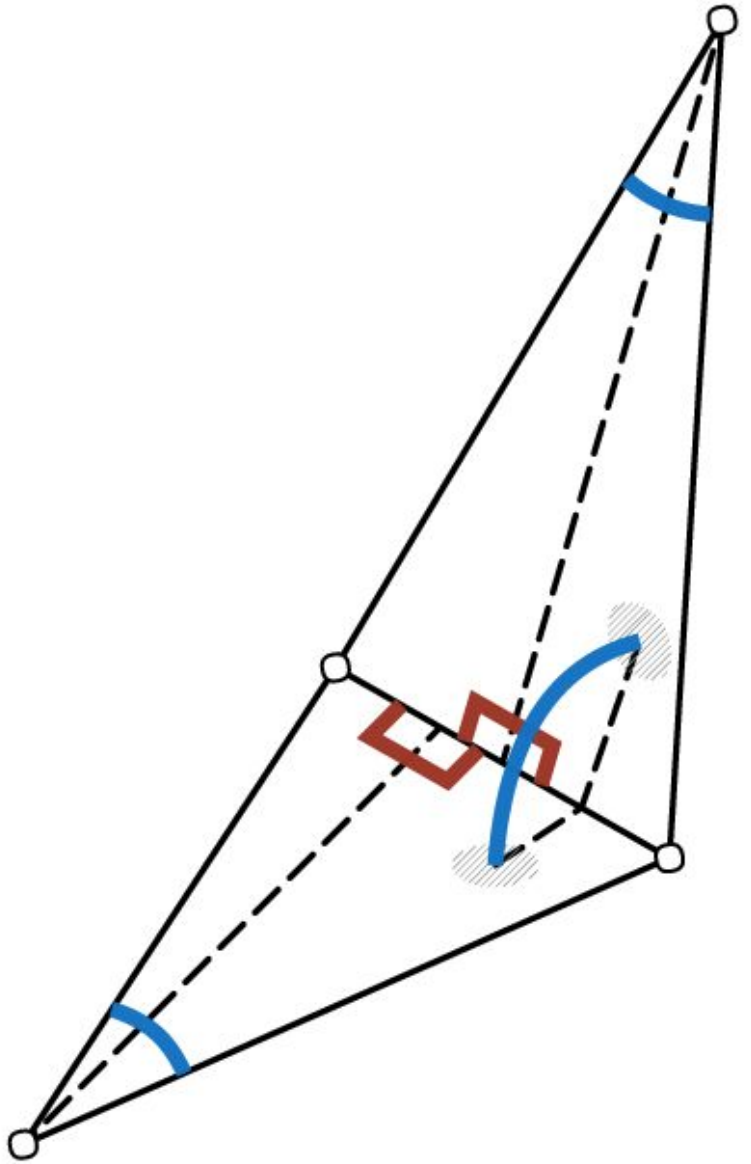
Hanocka, Rana, et al. MeshCNN: a network with an edge. *ACM Transactions on Graphics (TOG)* 2019

MeshCNN Framework


MeshCNN
handles
no handles
MeshCNN
neck
no neck

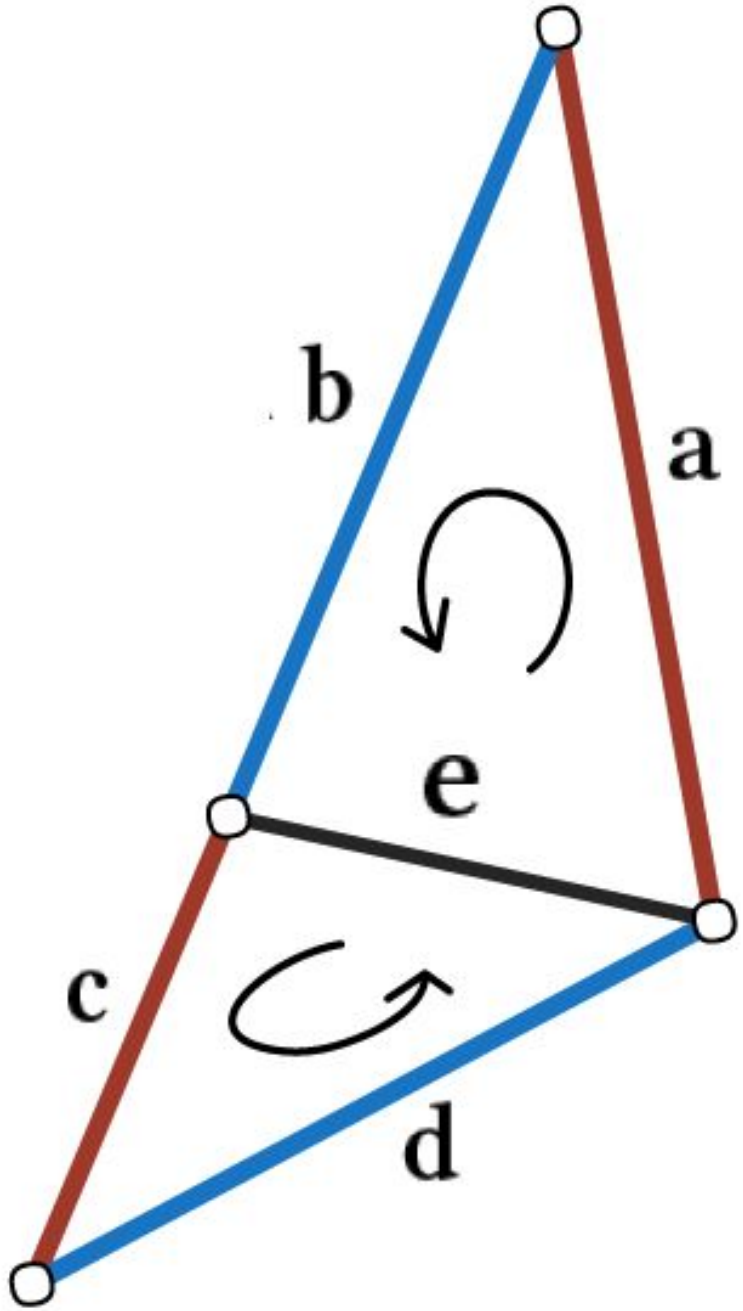
Edges are analogous to pixels

- Input feature for image
  - RGB value per pixel
- Input edge feature
  - 5-dimensional vector for every edge: the dihedral angle, two inner angles and two edge-length ratios for each face.

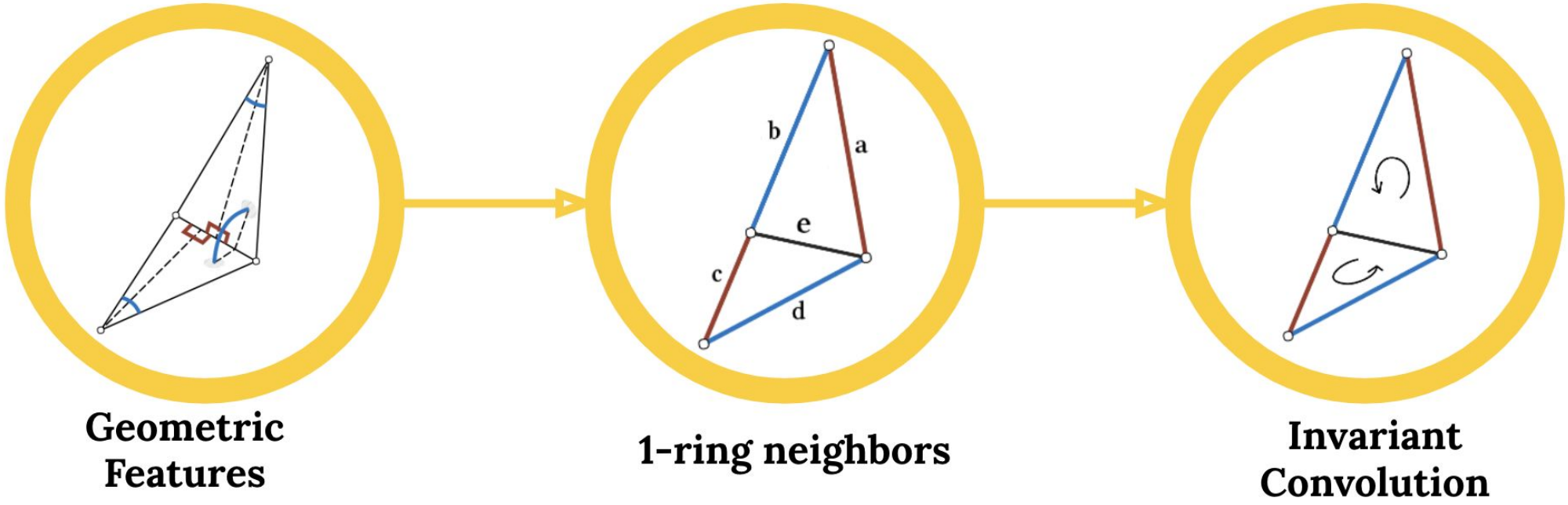


Mesh Convolution

- Learns features on the edges of the mesh
- Every edge is incident to exactly two faces (triangles)
- Defines a natural fixed-sized convolutional neighborhood of four edges.
- Two *valid* orderings: (a, b, c, d) or (c, d, a, b)
- Build symmetric features (invariant)
  - e-> (a+c, |a-c|, b+d, |b-d|)
- Conv Filter: (k_0, k_1, k_2, k_3, k_4)



Mesh Convolution

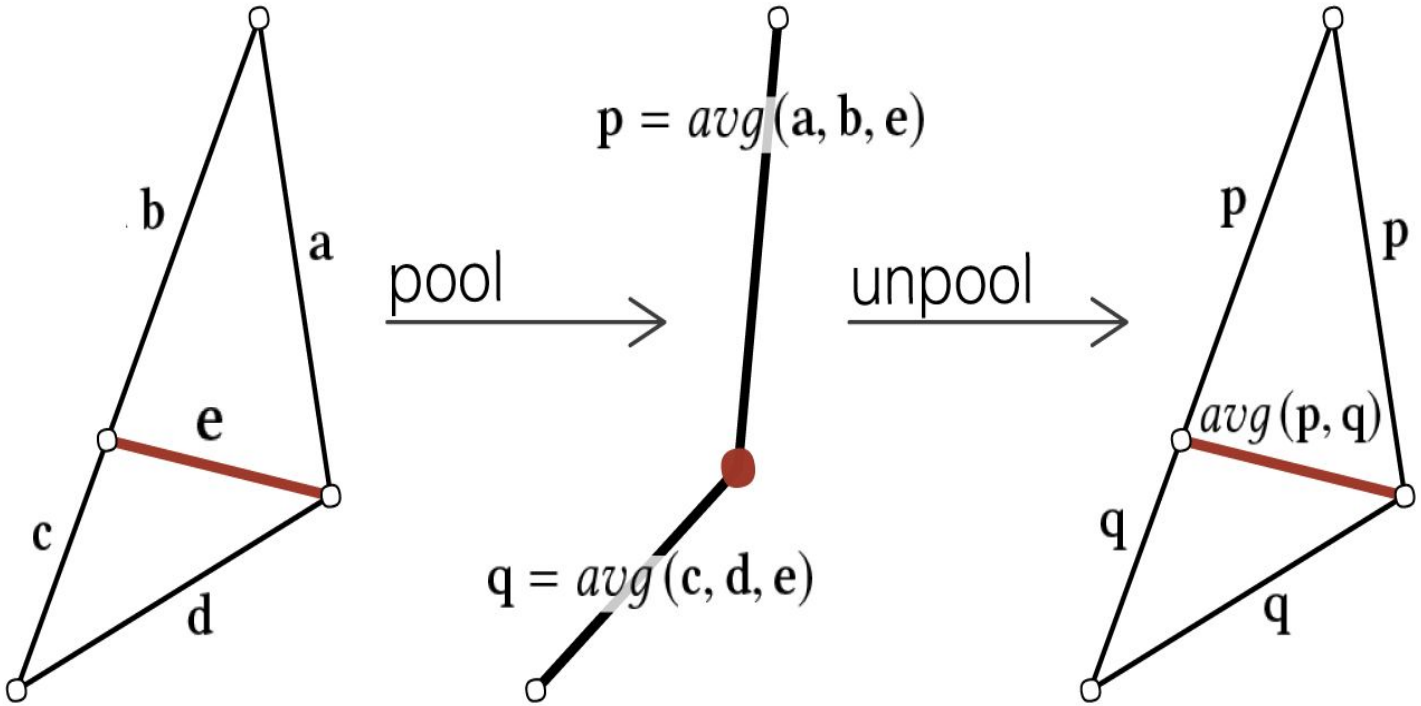


Mesh Simplification and Collapse

- **Traditional Mesh Simplification *(Geometric)***
  - Reduce information *but keep shape*
- **Edge Collapse**
  - Iteratively remove edges
- **Learned Edge Collapse**
  - Network decides collapse
  - Strengthens the learned representation
  - Visual insights from network

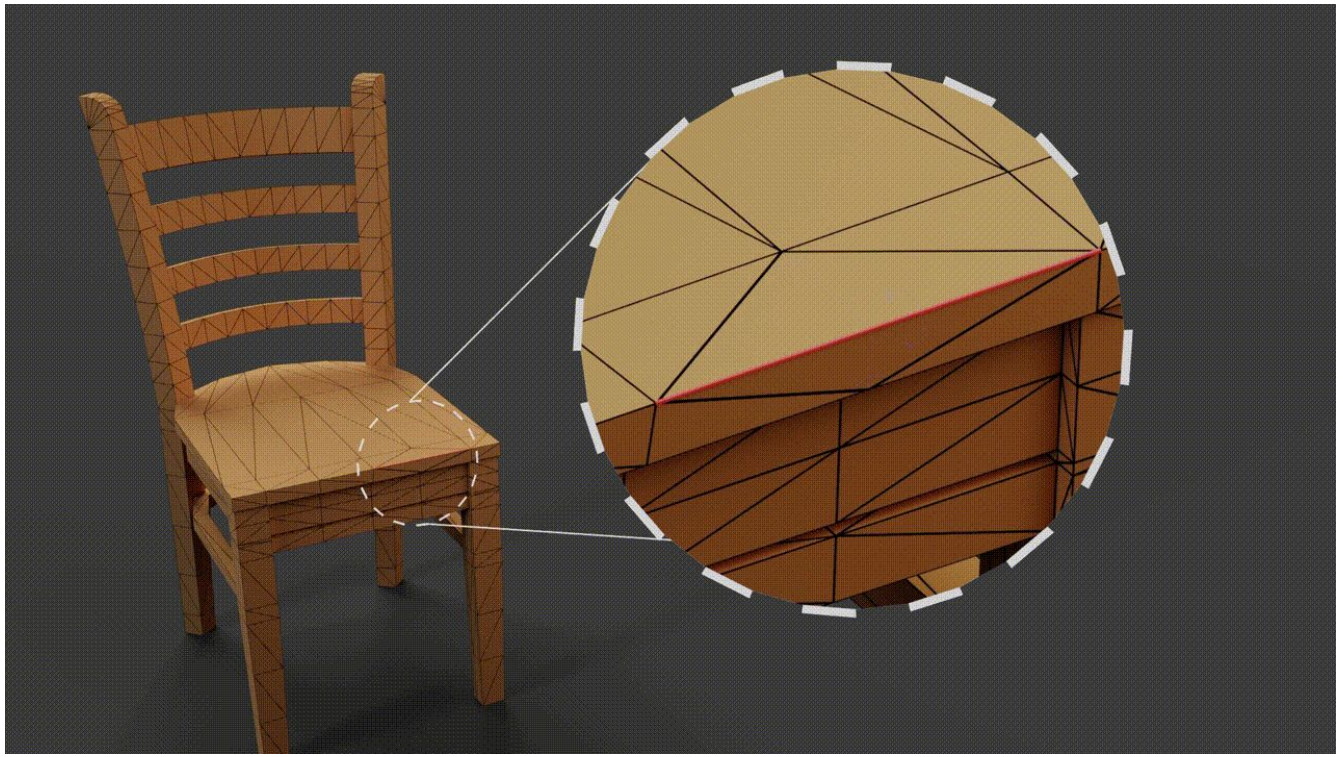
Mesh Pooling

- Delete edge with smallest feature activations
  - Aggregate features (average pooling)
  - Update topology
- Merge Edge Features
- Average pooling aggregation
  - 5 edges to 2 edges
  - Average per-channel



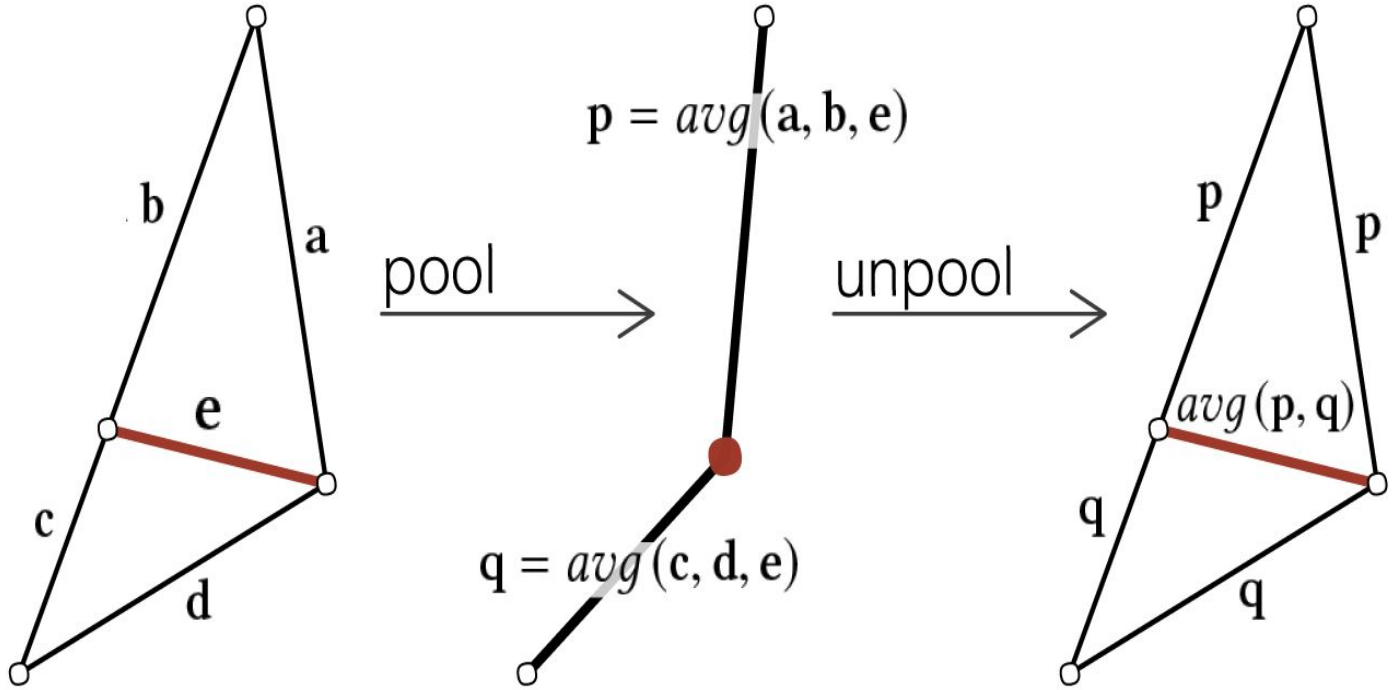
Update Mesh Topology

- New mesh is generated
  - Defines new neighbors in next convolution
  - Update relevant data structures



Mesh Unpooling

- For fully-convolutional tasks (such as segmentation), a mesh unpooling operation is used to restore the original mesh resolution.
- Partial Inverse of Pooling
  - Restores upsampled topology (reversible)
  - Unpooled features weighted combination of pooled features



Key Attributes of Mesh Pooling

- Spatially adapts to the task
- Flexible input mesh resolution
- Strengthens feature representation
- Visual insights from the network

Thank You!