Geometry Processing and Geometric Deep Learning

MVA Course, Lecture 4

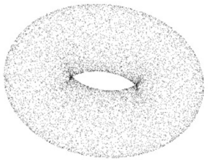
23/ 10 / 2024

Maks Ovsjanikov

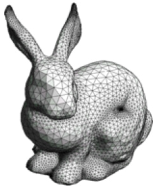
Deep Learning for 3D shapes

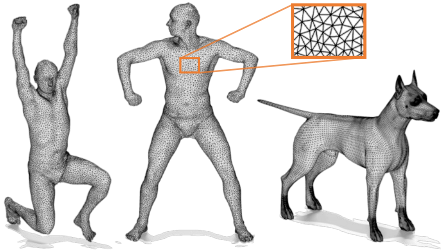
- Main Challenge

3D shapes (typically) do not have a canonical (grid-like) representation!



3D point cloud: an *unorganized collection of 3D coordinates*





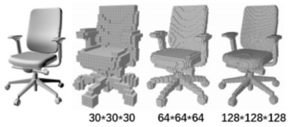
3D mesh: a *collection of points* and *triangles* connecting them.

Last time: Deep Learning on 3D shapes

- Multi-view approaches
- Volumetric approaches
- Spectral methods, pros and cons
- Intrinsic approaches
- Learning via diffusion

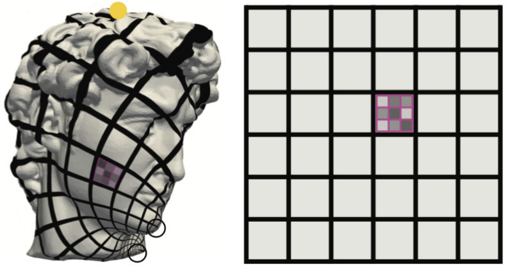
Different formulations of Non-Euclidean CNNs

View-based



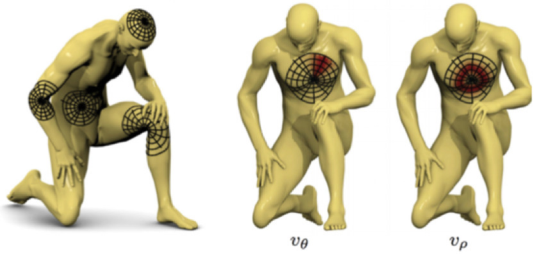
Voxel-based

Extrinsic



Embedding domain

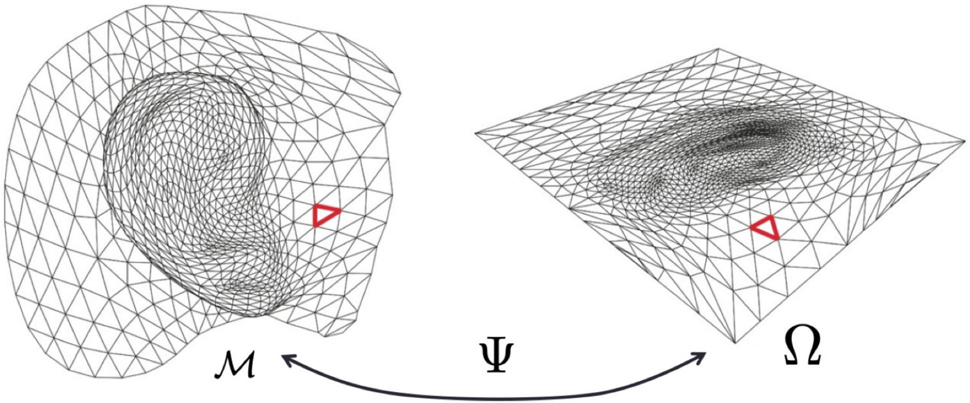
Spectral domain



Intrinsic (surface-based)

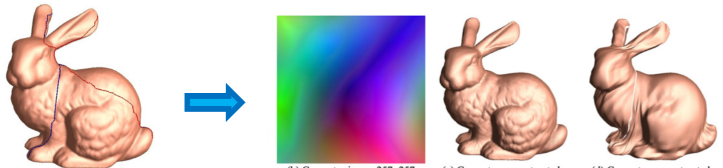
Global parametrization methods

Key idea: map the input surface to some parametric domain (e.g. 2D plane) where operations can be defined more easily.



Global parametrization methods

Key idea: map the input surface to some parametric domain (e.g. 2D plane) where operations can be defined more easily.



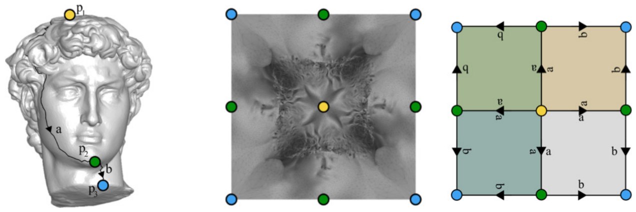
(a) Original mesh with cut 70K faces; genus 0

(b) Geometry image 257×257 (b∗) Compr. to 1.5KB (not shown)

(c) Geometry reconstructed entirely from b

(d) Geometry reconstructed entirely from b∗

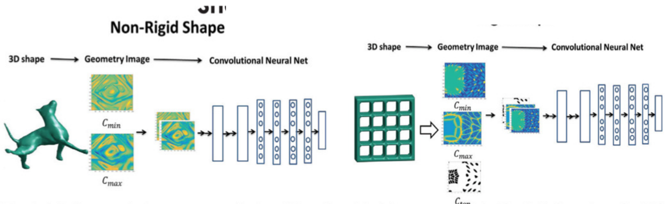
Torus 4-cover



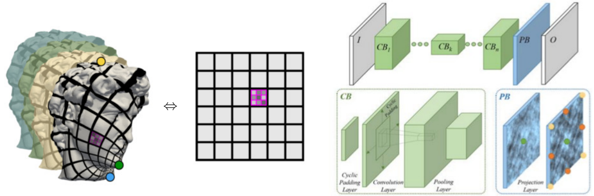
Surface $\mathcal{S}$ with sphere topology

Flat-torus $\mathcal{T}$ with 4 replicas of $\mathcal{S}$

Standard Euclidean 2D CNN architectures can now be used on $\mathcal{T}$.



The pixels in the geometry image corresponding to points on the original shape are encoded with principal curvatures for rigid shapes and HKS for non-rigid shapes. Then a standard CNN architecture can be modeled to learn the 3D shape.



Gu, Xianfeng, Steven J. Gortler, and Hugues Hoppe. "Geometry images." SIGGRAPH 2002.

Sinha, Ayan et al. "Deep learning 3D shape surfaces using geometry images." ECCV 2016

Maron, Haggai, et al. "Convolutional neural networks on surfaces via seamless toric covers." *ACM TOG* 2017

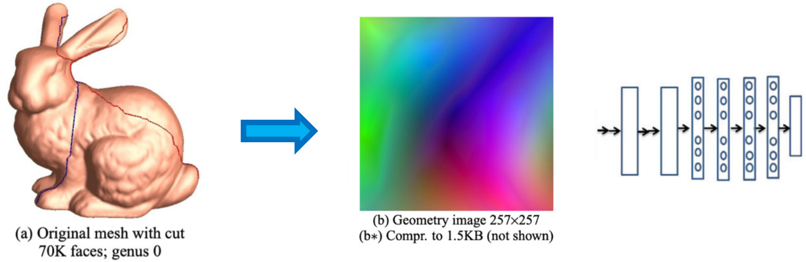
Projection-based Methods.

Advantages

- Represent the shape as a whole (rather than *partial* views)
- Can reuse shape parametrization methods
- Enables adoption of Euclidean (2D) learning techniques

Limitations

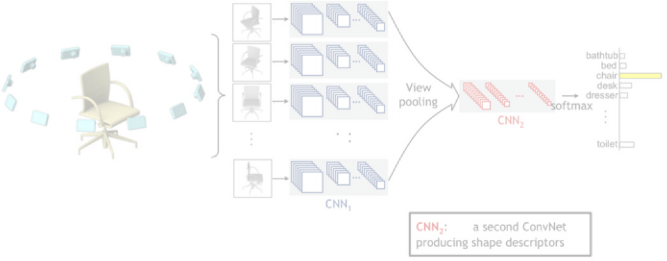
- Parametrizations are not unique
- Can induce (often heavy) *distortion*
- Rarely used in practice anymore



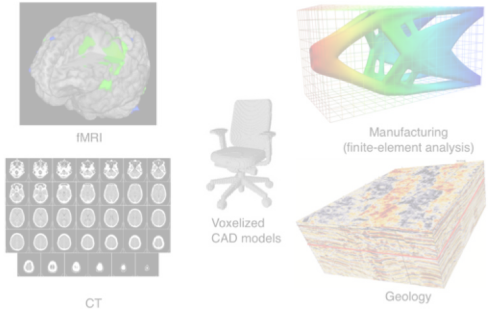
Main question:

How to enable neural networks to operate *directly on 3D data*?

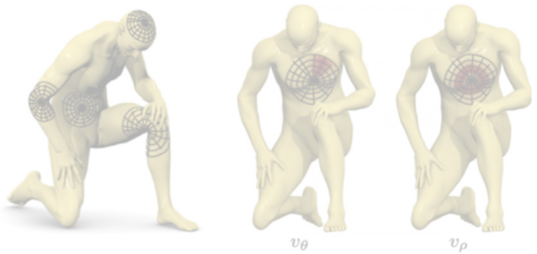
Approaches for 3D Deep-Learning



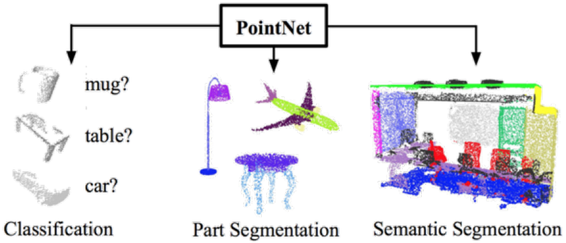
Multi-view based



Volumetric



Intrinsic (surface-based)



Point-based

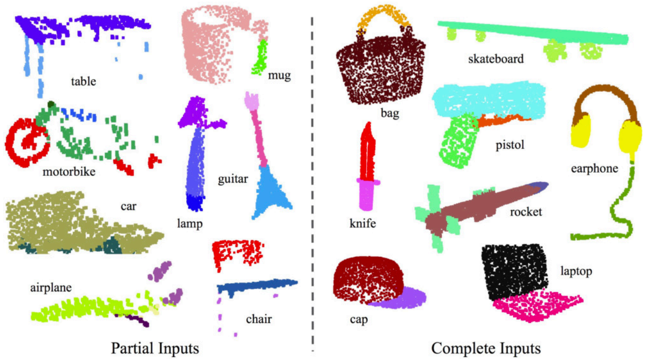
Today: Deep Learning on 3D shapes

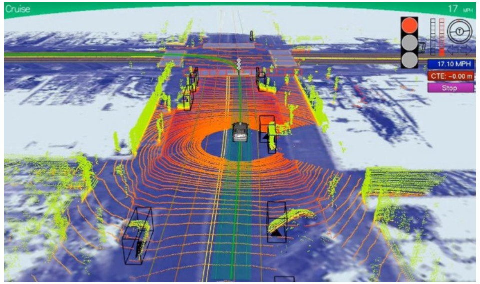
Learning on Point clouds

- Main architectures (PointNet, PointNet++, DGCNN, KPConv, Point Cloud Transformers)
- Applications (surface reconstruction, point cloud filtering)

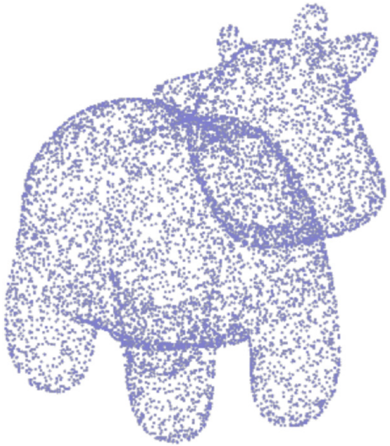
Point Clouds are everywhere!

- Simplest representation for 3D
- Very common output for 3D scanning
- Can be used jointly with images
- Sometimes have collections of points in higher dimensions!

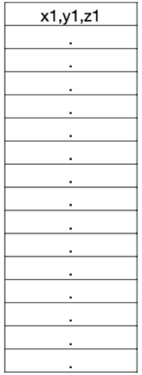




Recap: Point Clouds

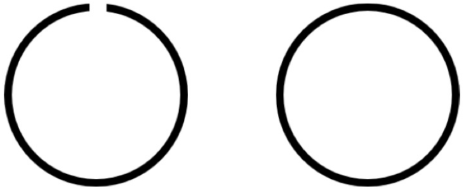


$$\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\}$$



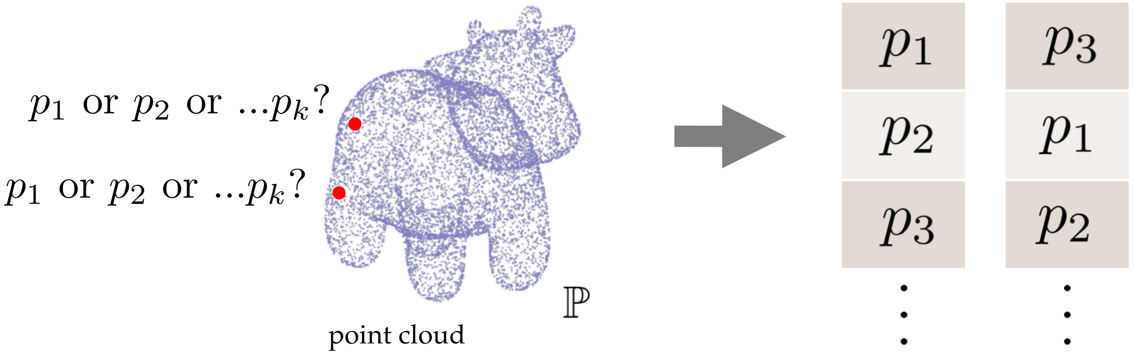
Often represented as a NX3 array

No explicit 'connectivity' information



Learning on Point Clouds Overview

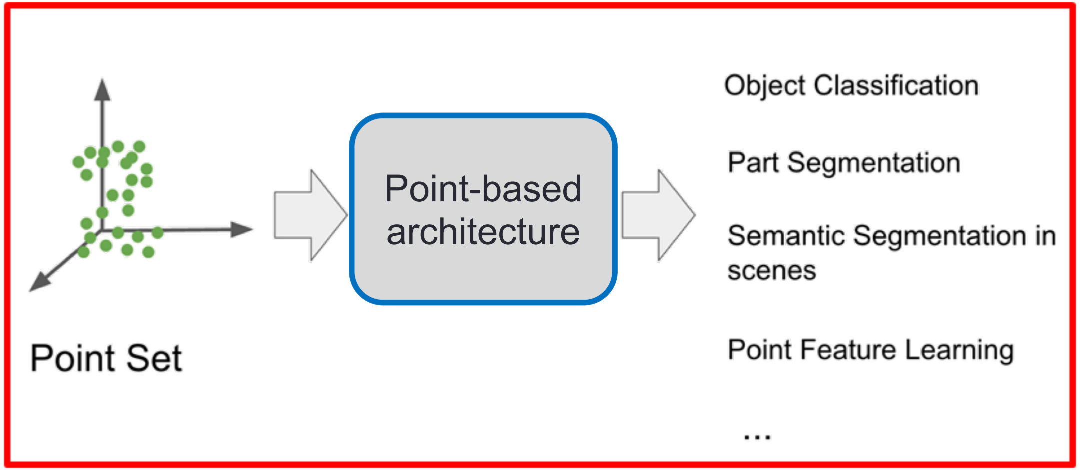
Essential challenge in point-based learning: **order invariance**!



Cannot use any method that *depends on the order of the points.*

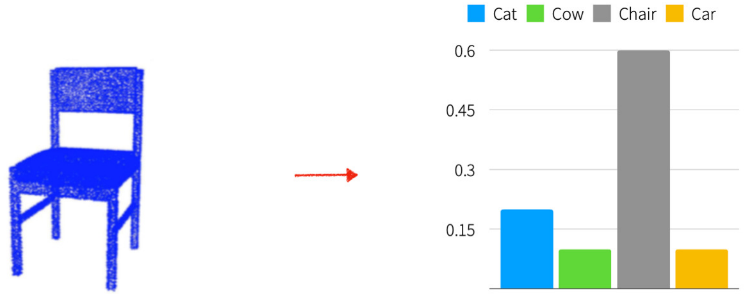
Point-Based methods

- **Goal:** design a NN architecture that can work *directly* with 3D point clouds
- Must deal with *unstructured, unordered* data



Two representative tasks:

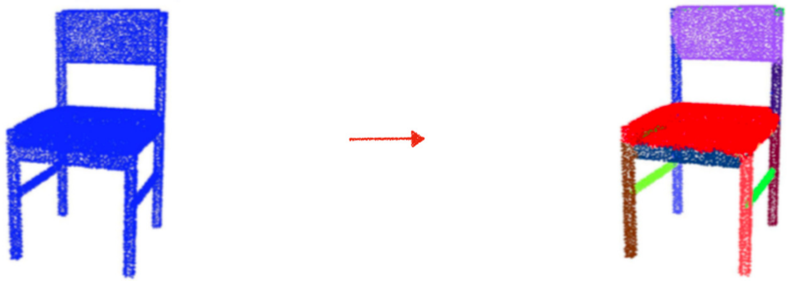
- Point Cloud *classification* and *segmentation*



Input: (B x) N x 3

Output: (B x) C

Representative example of a ‘global’ task

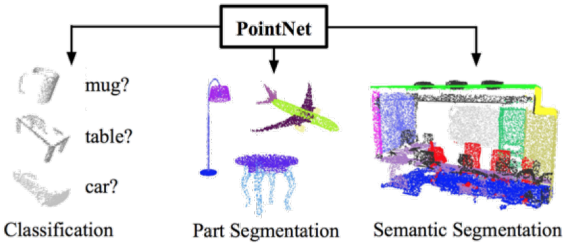


Input: N x 3

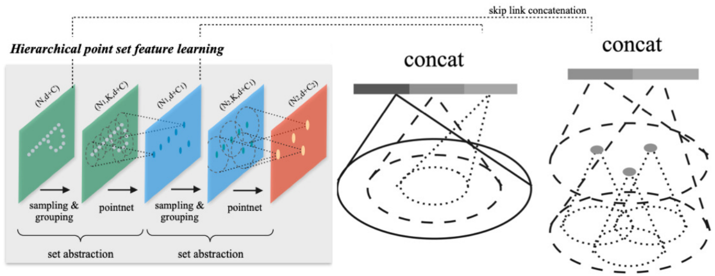
Output: N x C

Representative example of a pointwise prediction task

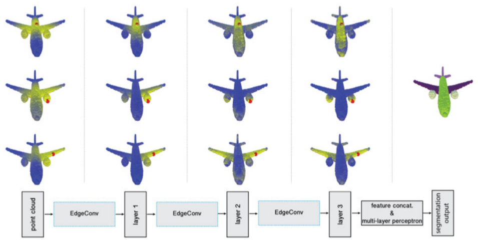
Point-based Architectures



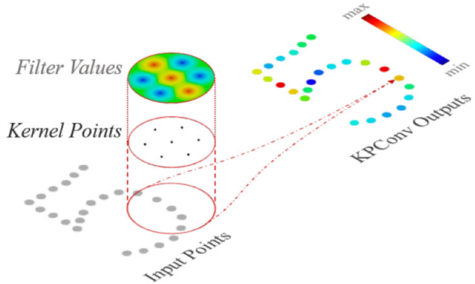
PointNet



PointNet++

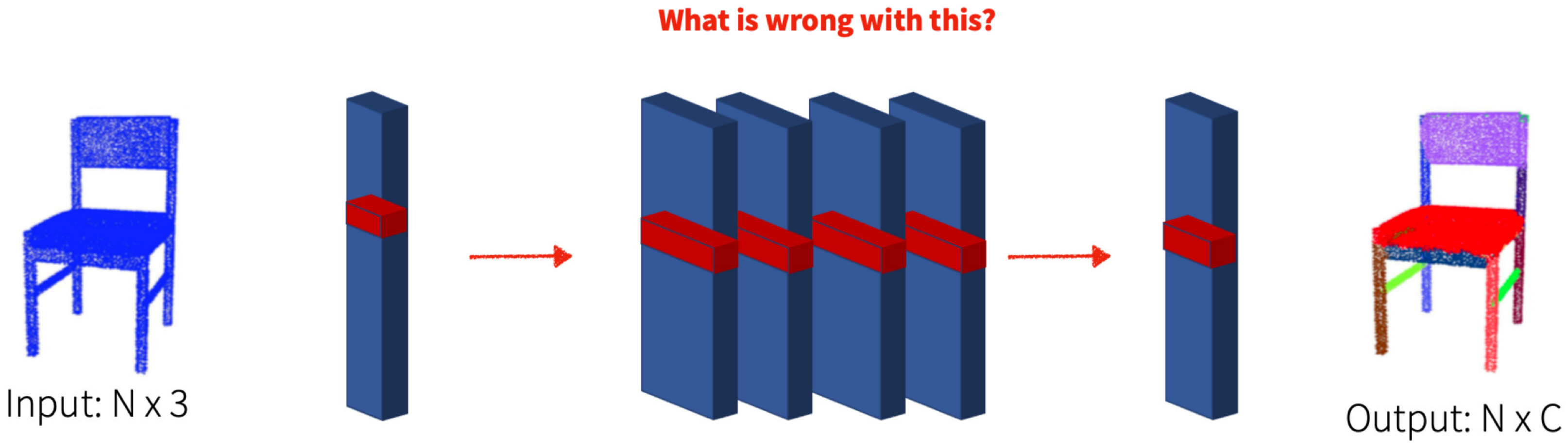


DGCNN (EdgeConv)



KPConv

Naive segmentation network 1

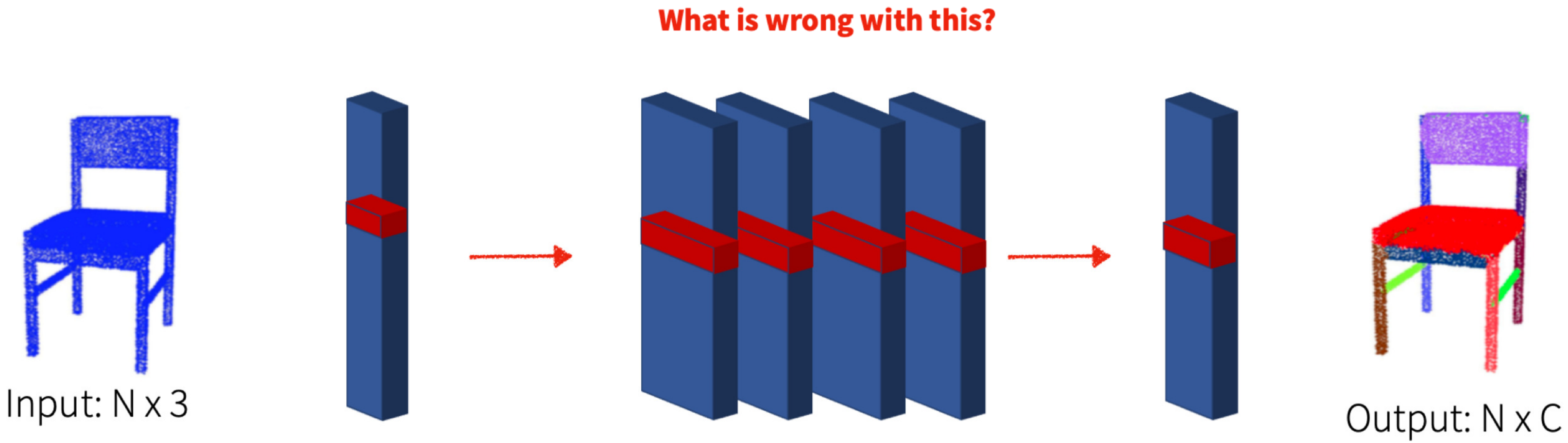


A simple network (shared MLP).

- **Input:** a 3-dimensional vector (3D coordinates)
- **Output:** C-dimensional prediction (class label). Apply *the same network to each point* of the point cloud.

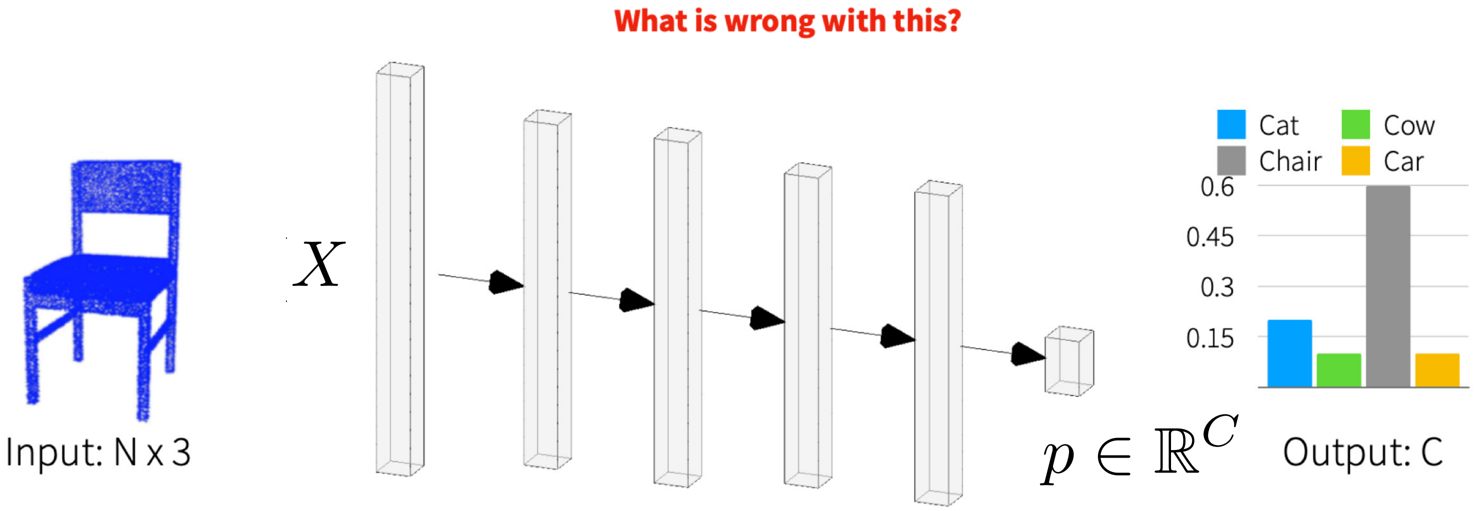
$$\psi(x_i) = p_c$$

Naive segmentation network 1



Processing each point independently! We will learn a function from 3D *coordinates* to a label. No communication between points = "shape awareness".

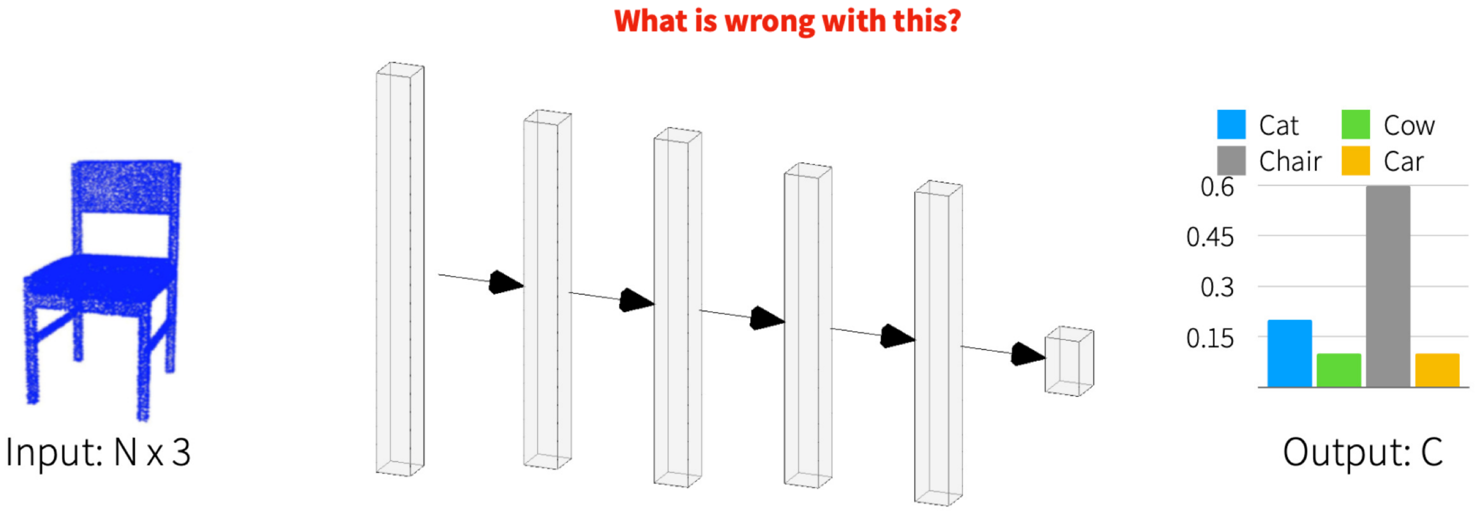
Naive segmentation network 2



Reshape input to a matrix X of size (3*N). Fully connected layers (MLP) from X to a C-dimensional vector:

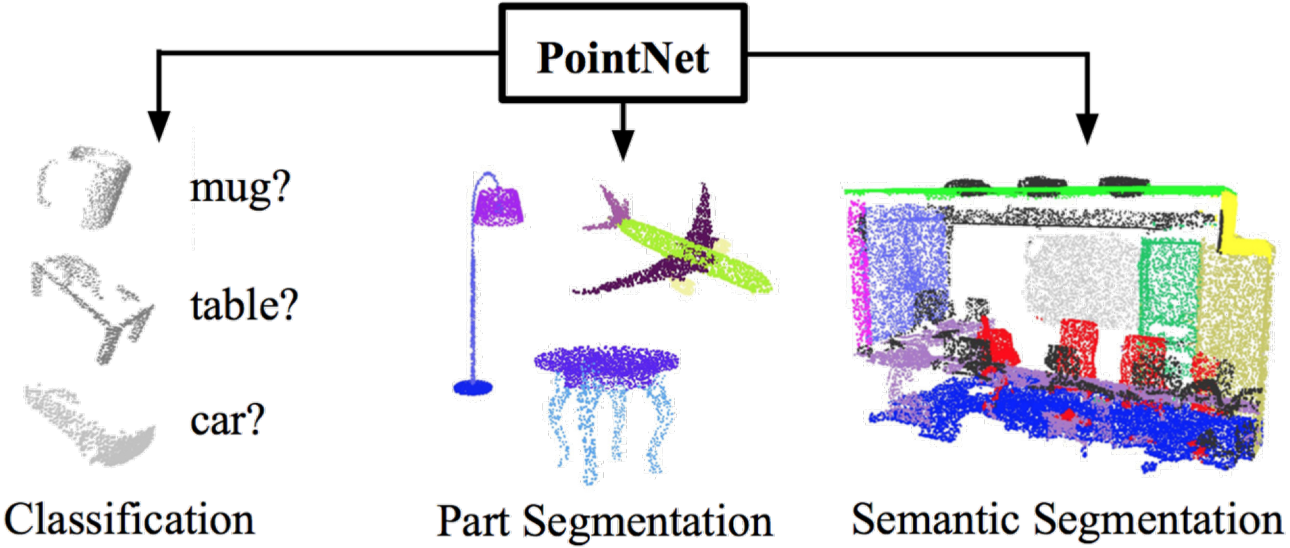
$$\mathrm{MLP}(X) = p \in \mathbb{R}^C$$

Naive segmentation network 2



1. Points tied to their 'index' = order in the point cloud (weights for 1st point not same as, e.g., 3rd point).
2. Cannot handle variable input sizes.

PointNet

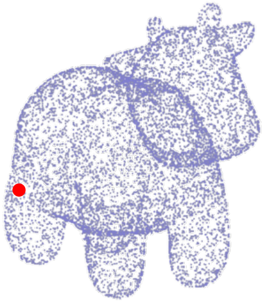


Qi et al., *Pointnet: Deep learning on point sets for 3d classification and segmentation*, CVPR 2017

PointNet Overview

First component: global feature vector through a symmetric operation!

p_i



point cloud $\mathbb{P}$

Shared MLP

p_1	p_3
p_2	p_1
p_3	p_2
$\vdots$	$\vdots$

$f(p_3)$
$f(p_1)$
$f(p_2)$

feature vector for a **point**

symmetric

order-independent

$$\sum_{p_j \in \mathbb{P}} g(f(p_j)) = F(\mathbb{P})$$

feature vector for the **point cloud**

$$f : \mathbb{R}^3 \to \mathbb{R}^k$$

learnable MLP

feature vector properties: $[f(p_i), F(\mathbb{P})]$

Only depends on the point **global**

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

PointNet: Basic Operations

MLP + Max Pooling

shared weights

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

The symmetric operation can be anything that is order-independent.

Original PointNet used max. Other variants use sum.

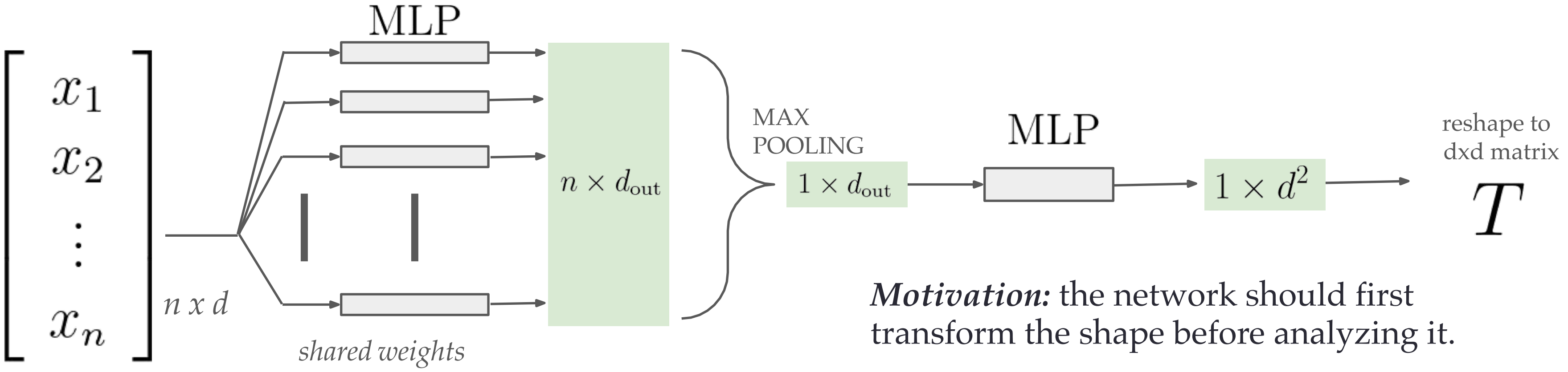
PointNet: Basic Operations

MLP + Max Pooling

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

shared weights

One more component: Learned *spatial transformation*



Motivation: the network should first transform the shape before analyzing it.

Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" CVPR 2017

PointNet Architecture

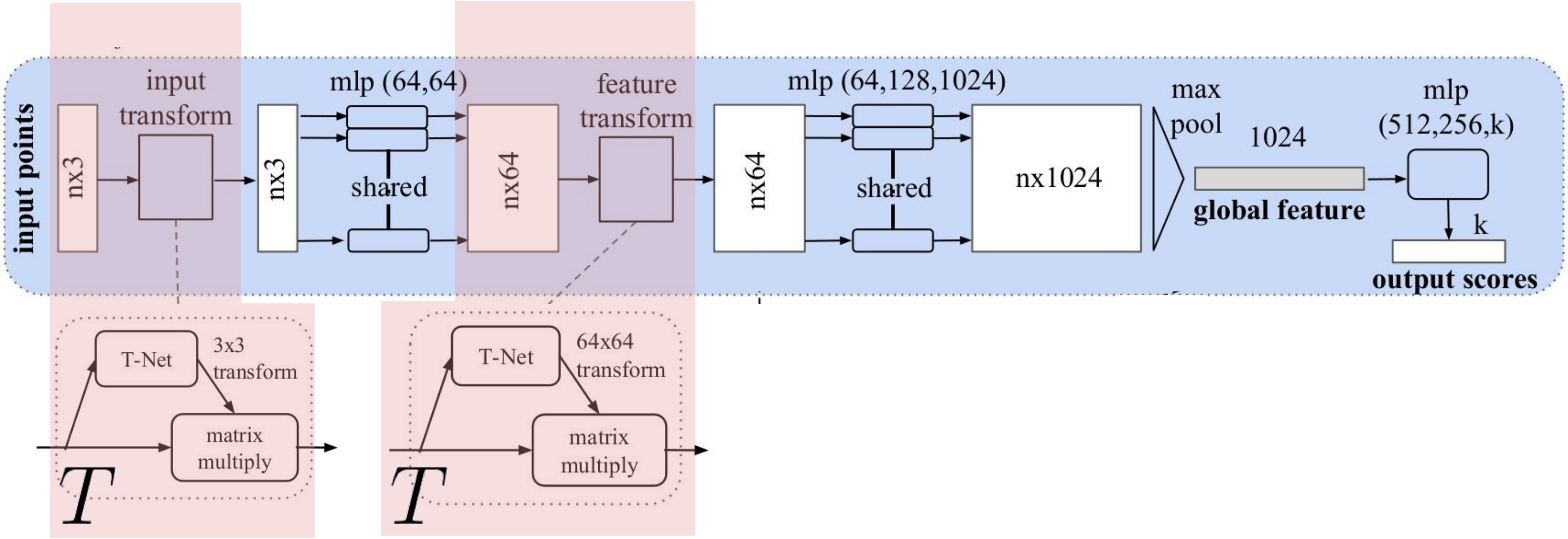
Composition of these two basic operations:

1. MLP + Max Pooling

2. Learned *transformation* matrix

Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" CVPR 2017

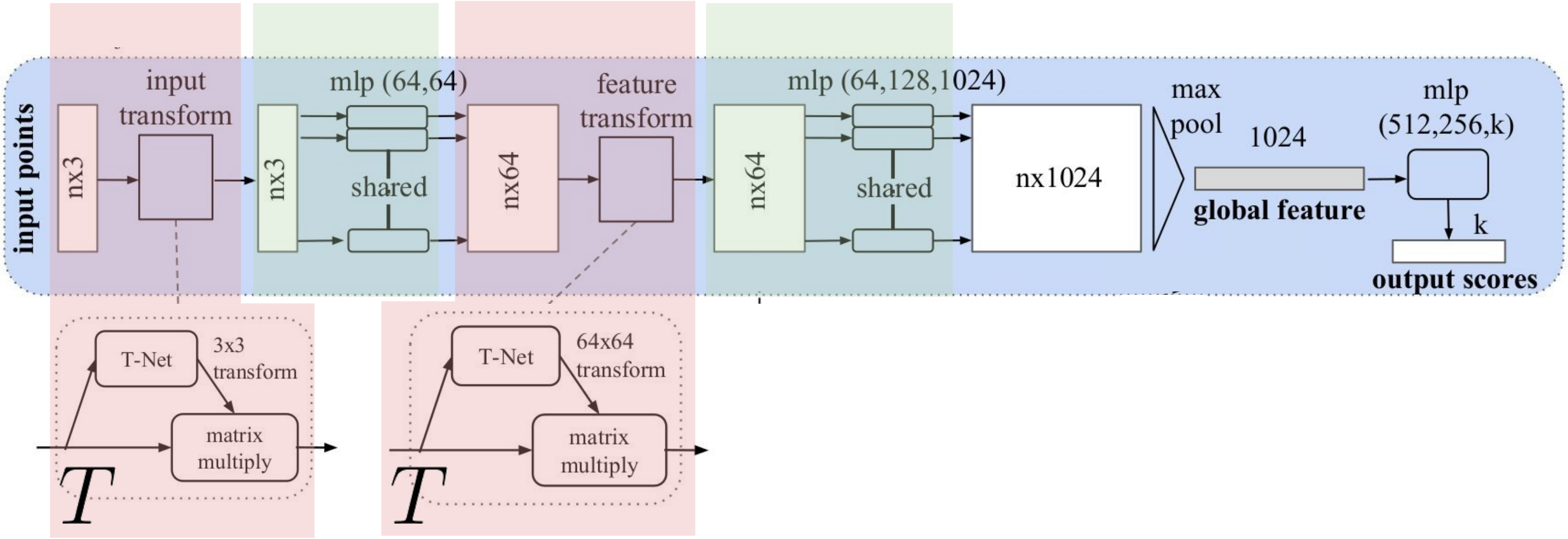
PointNet Architecture



Original design has two learned transformations: on the input 3d embedding and on the learned (64-dim) features.

Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" CVPR 2017

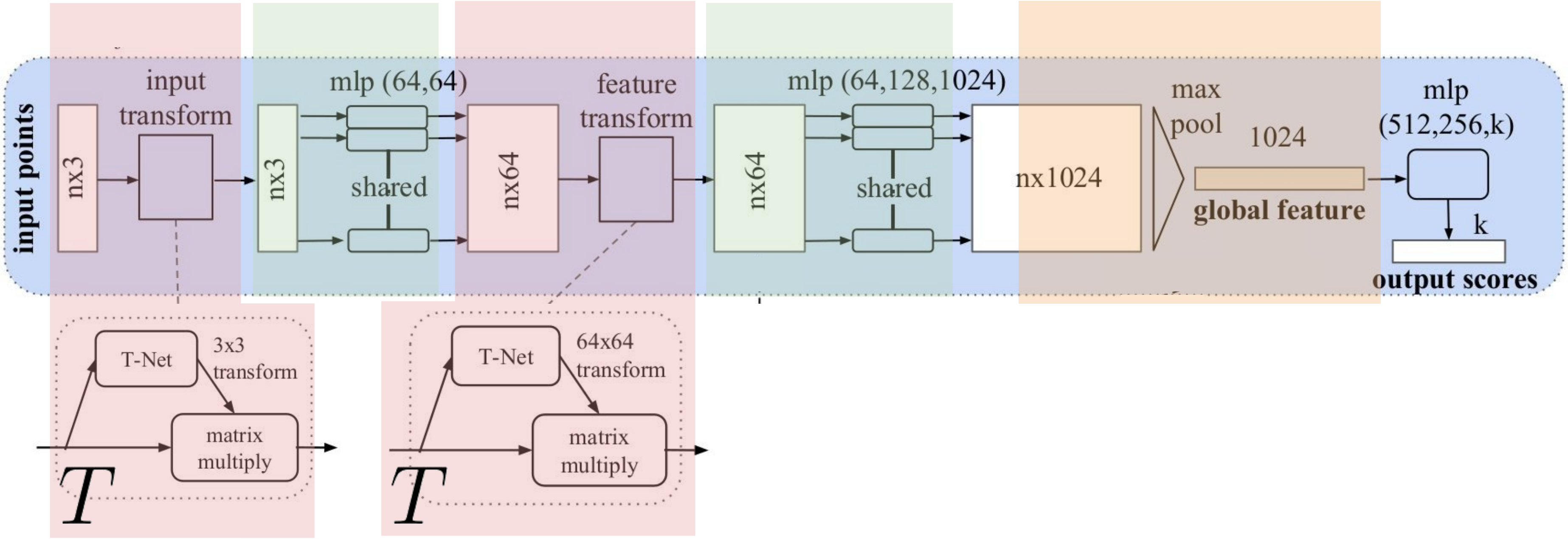
PointNet Architecture



Multi-Layer Perceptron (shared weights) **to uplift the dimensions (important).**

Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation” CVPR 2017

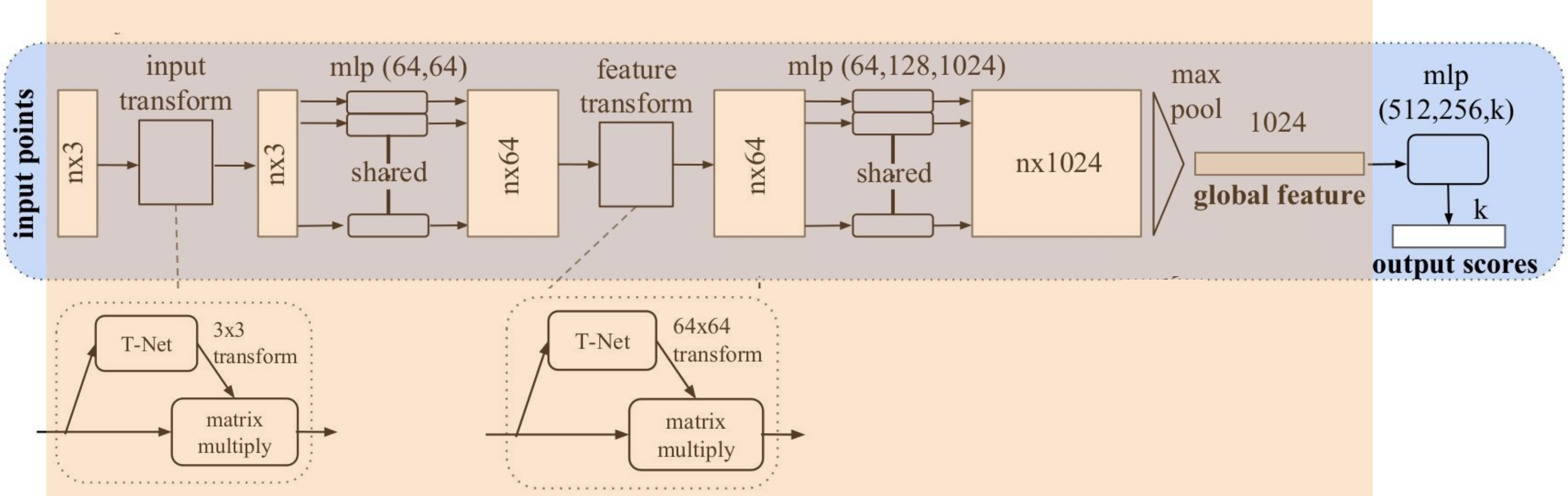
PointNet Architecture



Max Pooling **to extract a global feature** (i.e., a single vector that summarizes the entire point cloud).

Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" CVPR 2017

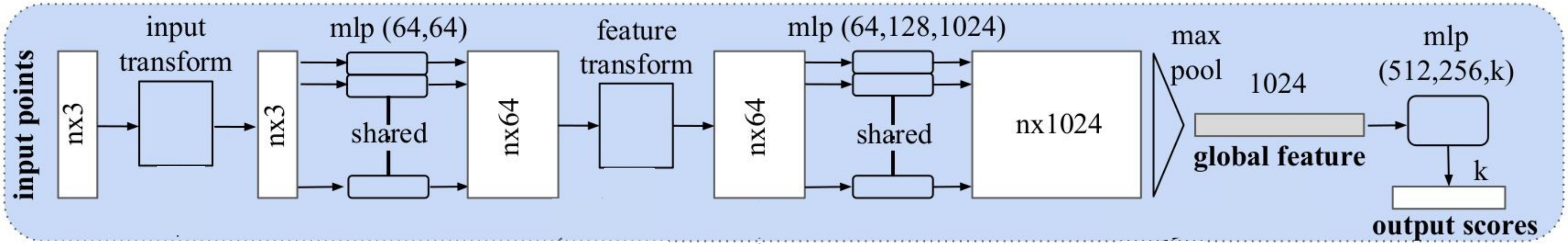
PointNet Architecture



$$f(\{x_1, x_2, \dots x_n\}) = \max\{h(x_i), h(x_2), \dots h(x_n)\}$$

Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" CVPR 2017

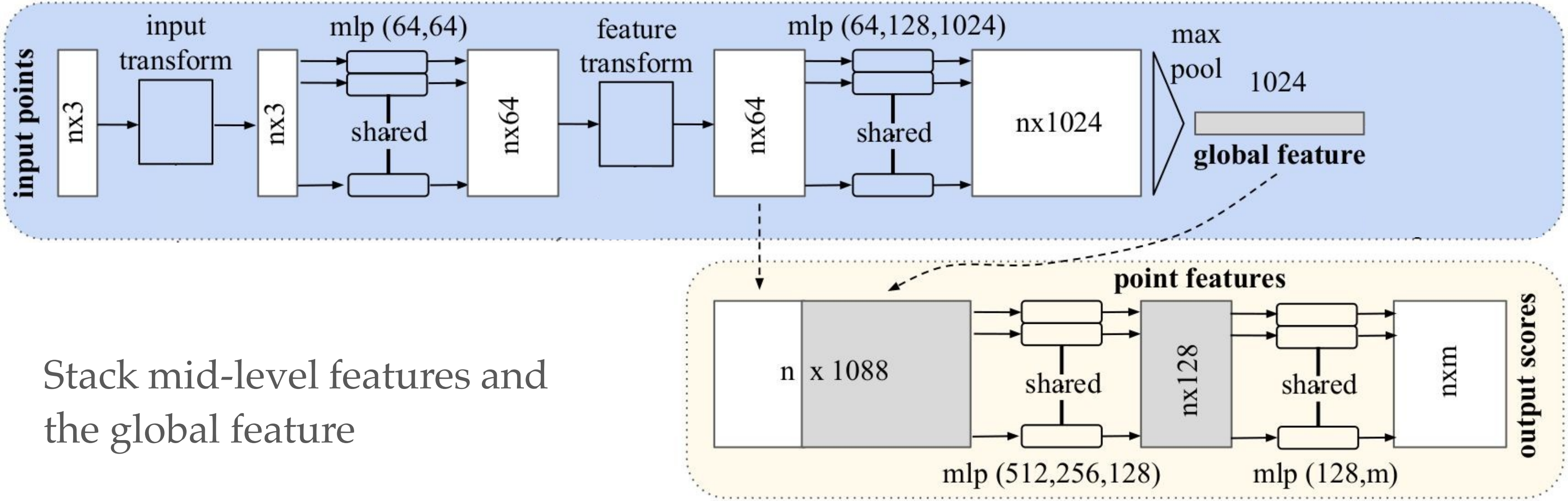
PointNet Architecture: Segmentation



- The basic version produces a global feature for the entire point cloud. Useful for shape *classification.*
- In some problems (e.g., segmentation) we need an output label *for each point*. The label has to be informed by the overall shape structure (i.e., cannot be done independently).

Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" CVPR 2017

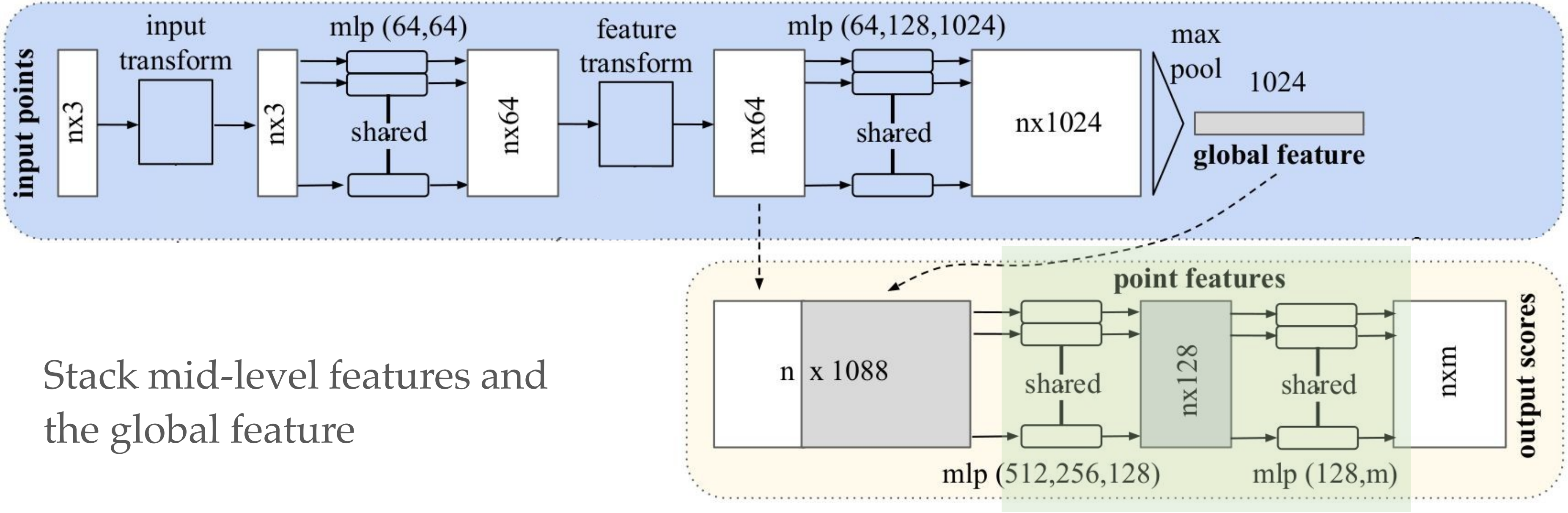
PointNet Architecture: Segmentation



Stack mid-level features and the global feature

Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" CVPR 2017

PointNet Architecture: Segmentation

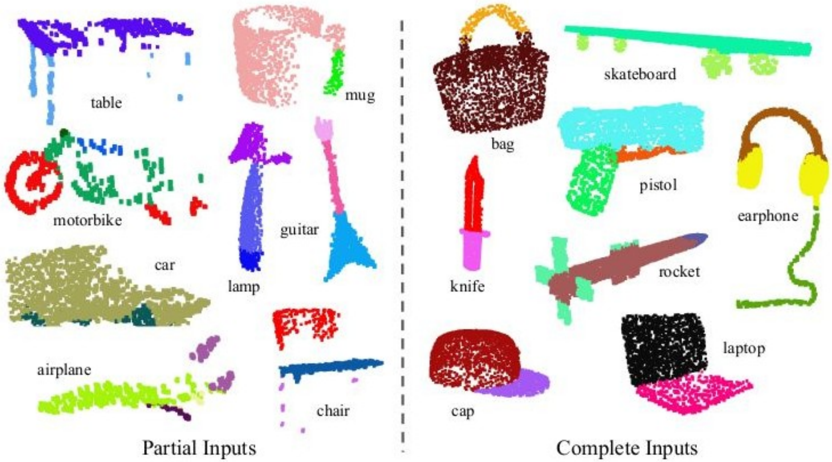


Stack mid-level features and the global feature

Another MLP to extract the final score for each point

Results

Object Part Segmentation



Object Classification

	input	#views	accuracy avg. class	accuracy overall
SPH [11]	mesh	-	68.2	-
3DShapeNets [28]	volume	1	77.3	84.7
VoxNet [17]	volume	12	83.0	85.9
Subvolume [18]	volume	20	86.0	**89.2**
LFD [28]	image	10	75.5	-
MVCNN [23]	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.

Not even state of the art in 2017.
However started a "revolution" in 3D deep learning.

Scene Segmentation

Applications in Shape Reconstruction

- Estimate a surface from its point cloud sampling



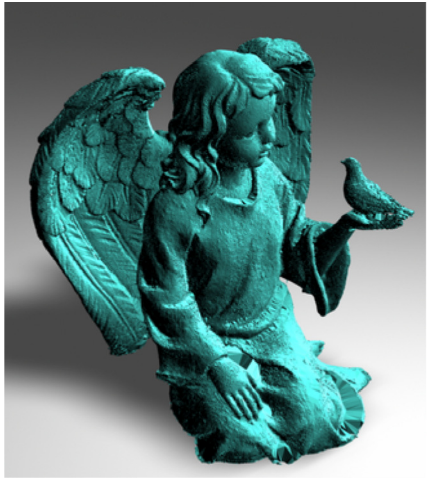
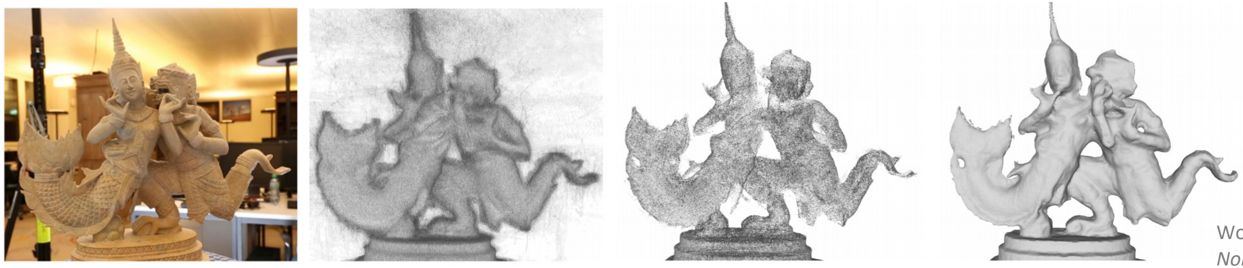


Image credit:
Kolluri et al. 2004

Common Reconstruction Pipeline

Main steps for reconstruction from point clouds:

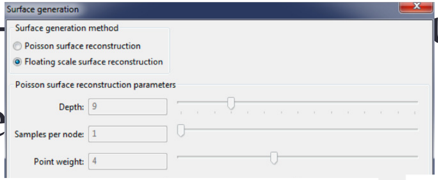
1. Outlier removal – remove points
2. If have multiple scans, align them.
3. Smoothing – remove local noise.
4. Estimate normals at the points.
5. Surface fitting (e.g. Poisson-based)
   - Triangle mesh extraction.

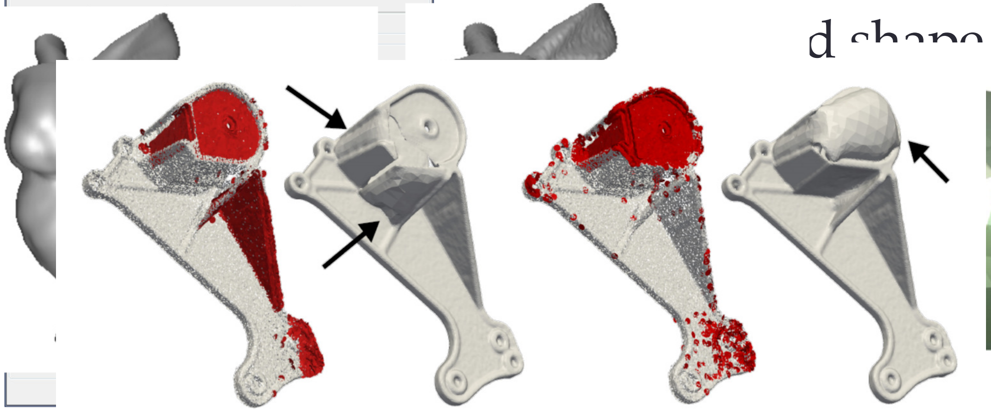


Wolf et al. / *Point Cloud Noise and Outlier Removal for Image-Based 3D Reconstruction*, 2016

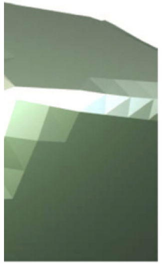
(some of the) Challenges:

1) **Tons** of parameters
2) Can over- ures or overfit to noise
3) Most ofte are required
4) Theoreti d shape models, and are often fai





[HDD* 92], [HDD* 92] + Poisson, [LW10], [LW10] + Poisson



Berger et al. / *A Survey of Surface Reconstruction from Point Clouds*, 2016

Key steps for 3D reconstruction

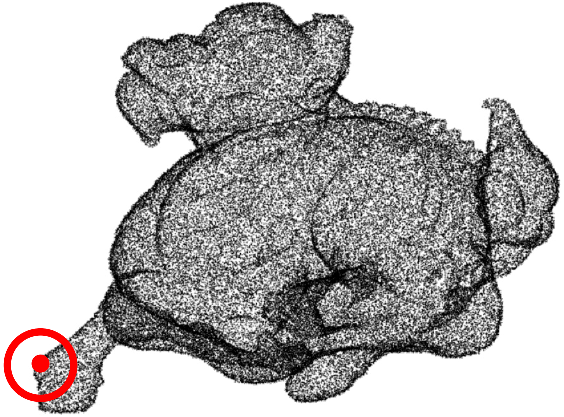
Main steps for reconstruction from point clouds:

1. **Outlier removal**
2. If have multiple scans, align them.
3. **Smoothing**
4. **Estimate normals**
5. Surface fitting (e.g. Poisson-based)
   - Triangle mesh extraction.

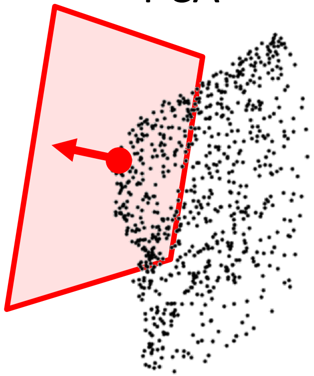
Wolf et al. / *Point Cloud Noise and Outlier Removal for Image-Based 3D Reconstruction*, 2016

Traditional Approaches – Normal Estimation

Examples:

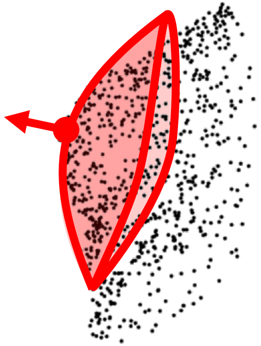


PCA



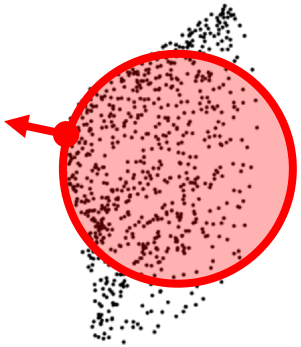
Surface reconstruction from unorganized points, Hoppe et al., 1992

Jet fitting



Estimating differential quantities using polynomial fitting of osculating jets, Cazals and Pouget, 2005

MLS Sphere Fitting

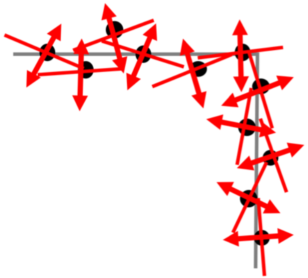


Algebraic Point Set Surfaces, Guennebaud and Gross, 2007

Limitations of Axiomatic Approaches

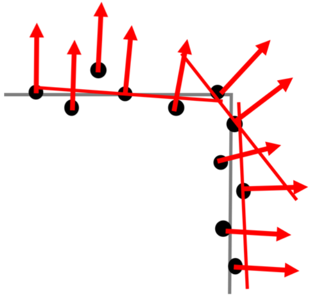
- Always rely on a user-specified neighborhood
- Can lead to under-fitting (smoothing) near sharp edges or over-fitting to noise
- Normal *orientation* is hard.

Small patch size



Sensitive to noise

Large patch size

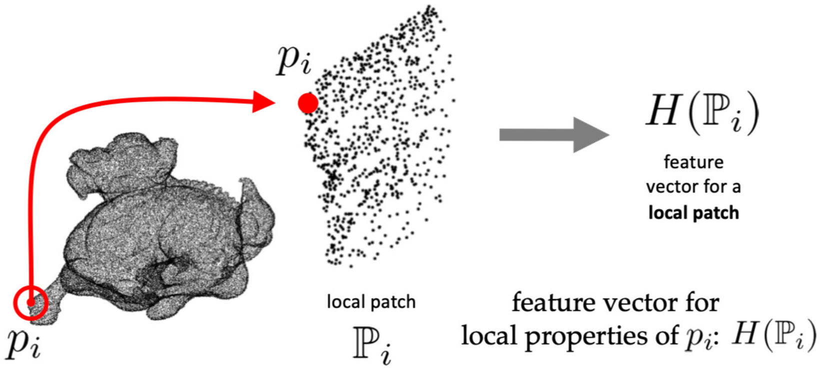


Over-smoothing

PCPNET: Learning Local Properties

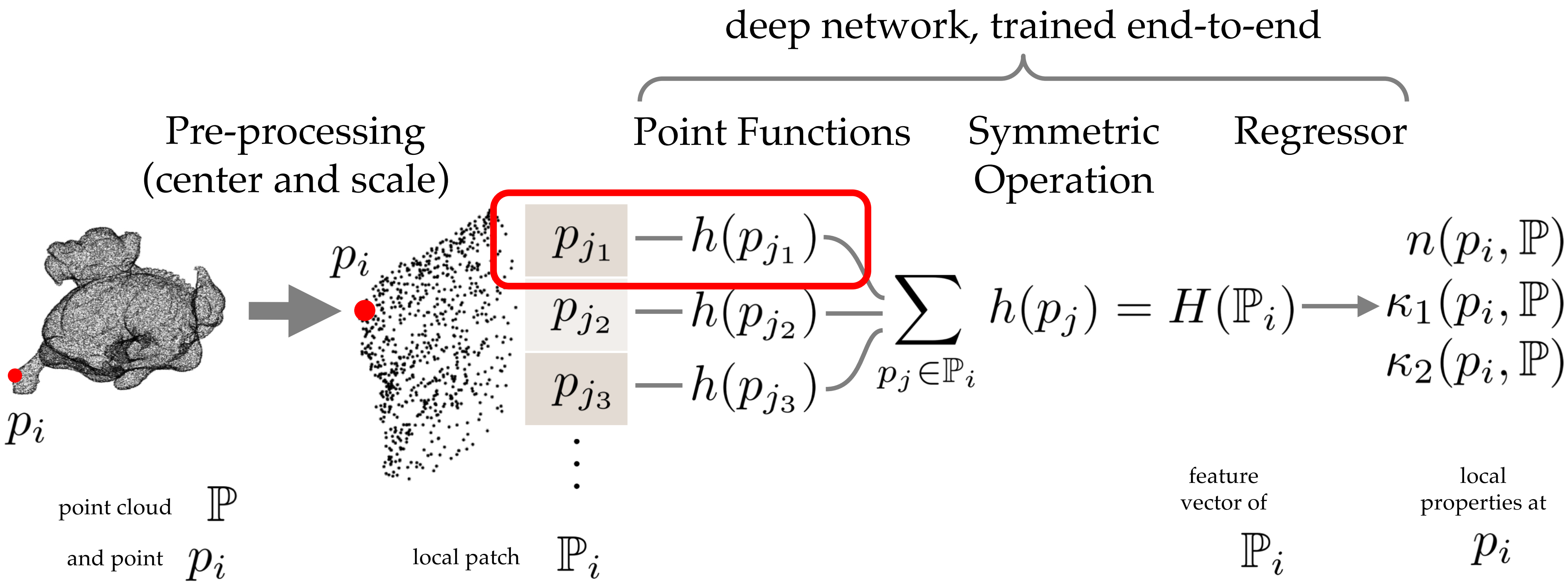
Intuition:

1) Normal and curvature estimation is a *local* operation.
2) Process shapes by *patches*.
3) Can *sample* point clouds from surfaces for almost unlimited training data.



PCPNET: Learning Local Shape Properties from Raw Point Clouds,
Guerrero, Kleiman, O., Mitra, 2018

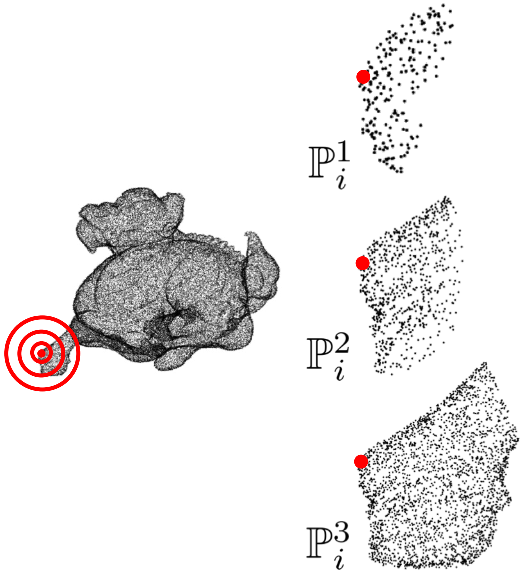
PCPNET architecture



PCPNET: Learning Local Shape Properties from Raw Point Clouds,
Guerrero, Kleiman, O., Mitra, 2018

PCPNet multi-scale

Three radii, 3072 point functions, concatenate patch features



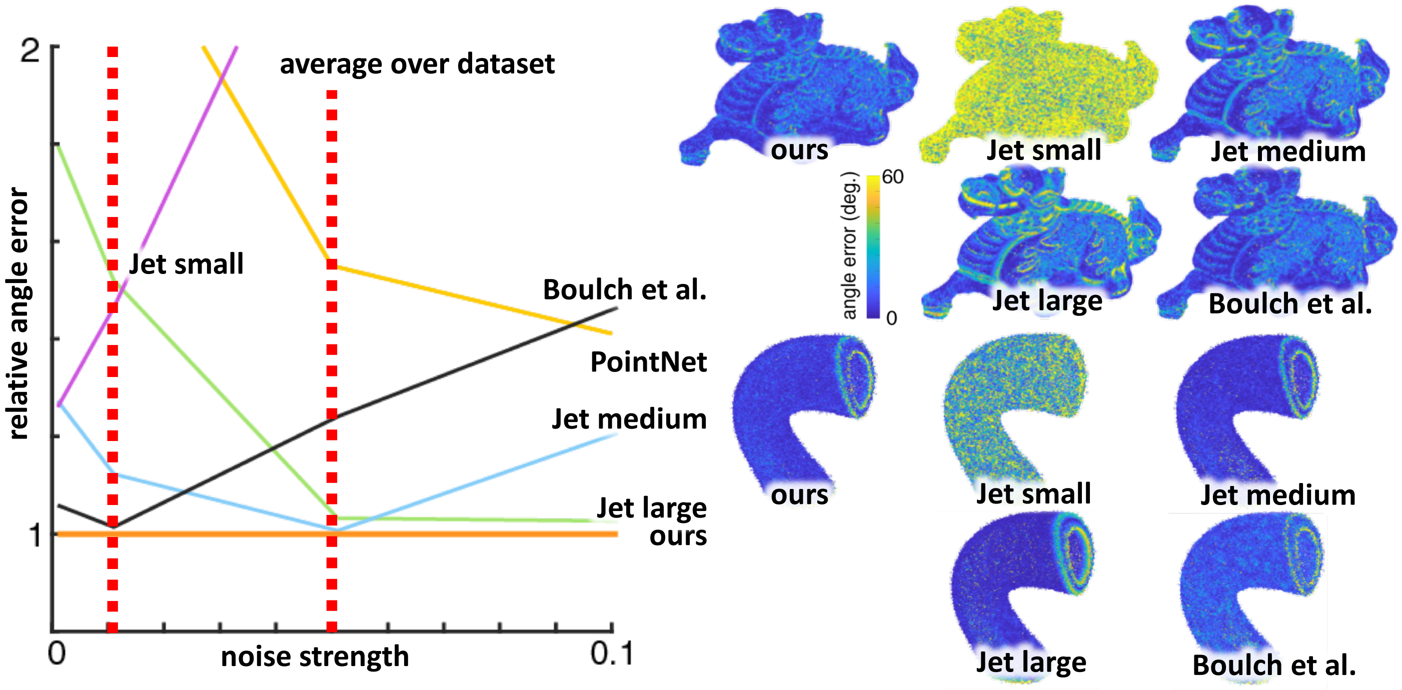
$$\sum_{p_j \in \mathbb{P}_i^1} h(p_j) = H(\mathbb{P}_i^1)$$

$$\sum_{p_j \in \mathbb{P}_i^2} h(p_j) = H(\mathbb{P}_i^2)$$

$$\sum_{p_j \in \mathbb{P}_i^3} h(p_j) = H(\mathbb{P}_i^3)$$

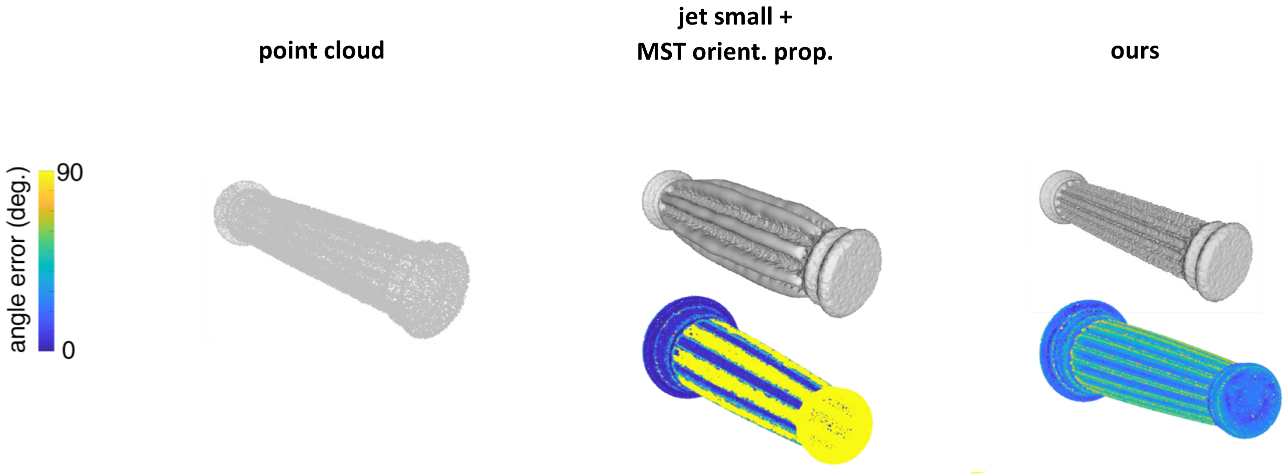
$$\begin{bmatrix} H(\mathbb{P}_i^1) \\ H(\mathbb{P}_i^2) \\ H(\mathbb{P}_i^3) \end{bmatrix} \rightarrow \begin{matrix} n(p_i, \mathbb{P}) \\ \kappa_1(p_i, \mathbb{P}) \\ \kappa_2(p_i, \mathbb{P}) \end{matrix}$$

Normal Estimation Results



PCPNET: Learning Local Shape Properties from Raw Point Clouds, Guerrero, Kleiman, O., Mitra, 2018

Oriented Normal Estimation & Surface Reconstruction



Key steps for 3D reconstruction

Main steps for reconstruction from point clouds:

1. **Outlier removal**
2. If have multiple scans, align them.
3. **Smoothing**
4. **Estimate local differential properties**
5. Surface fitting (e.g. Poisson-based)
   - Triangle mesh extraction.

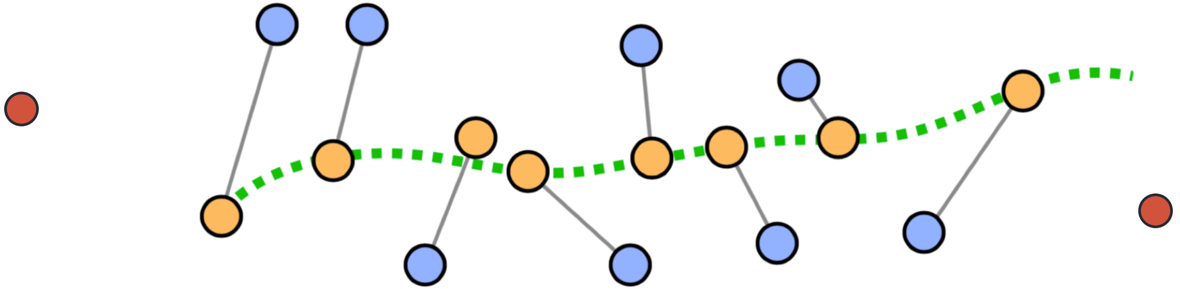
Wolf et al. / *Point Cloud Noise and Outlier Removal for Image-Based 3D Reconstruction*, 2016

PointCleanNet

Main general idea:

Learn to denoise point clouds and to remove outliers

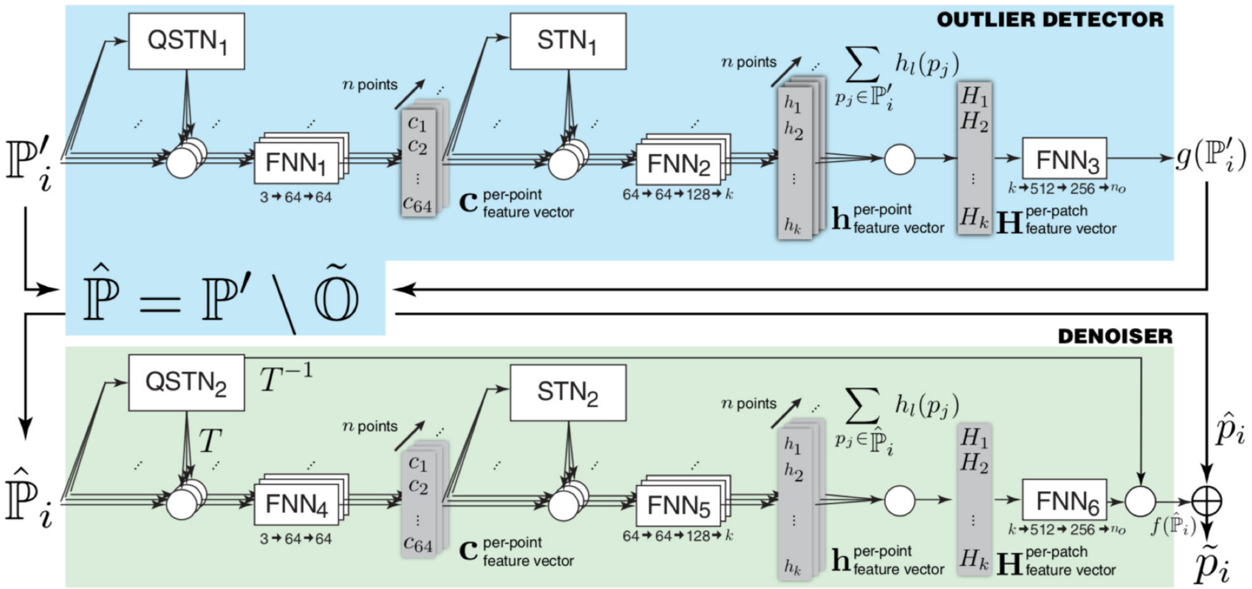
Similar approach to PCPNet, except fit a *local displacement vector*, and a *classifier score for outliers*.



PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds, M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. Mitra, M. O.

PointCleanNet – Architecture

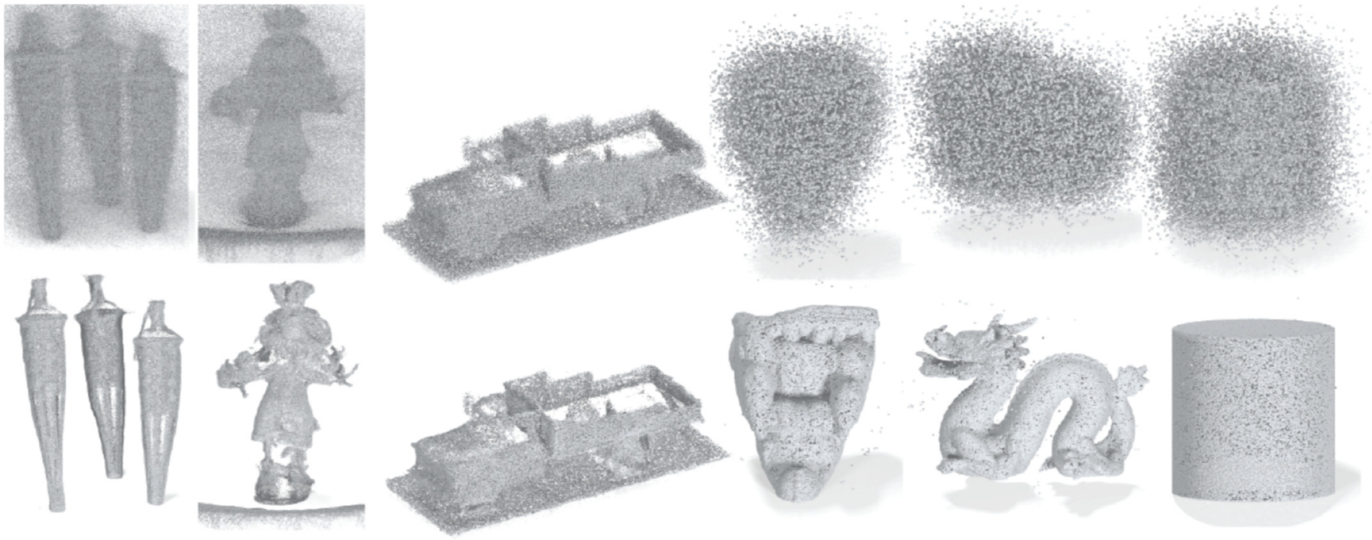
PointCleanNet Architecture and description



PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds,
M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. Mitra, M. O., 2019

PointCleanNet – Results

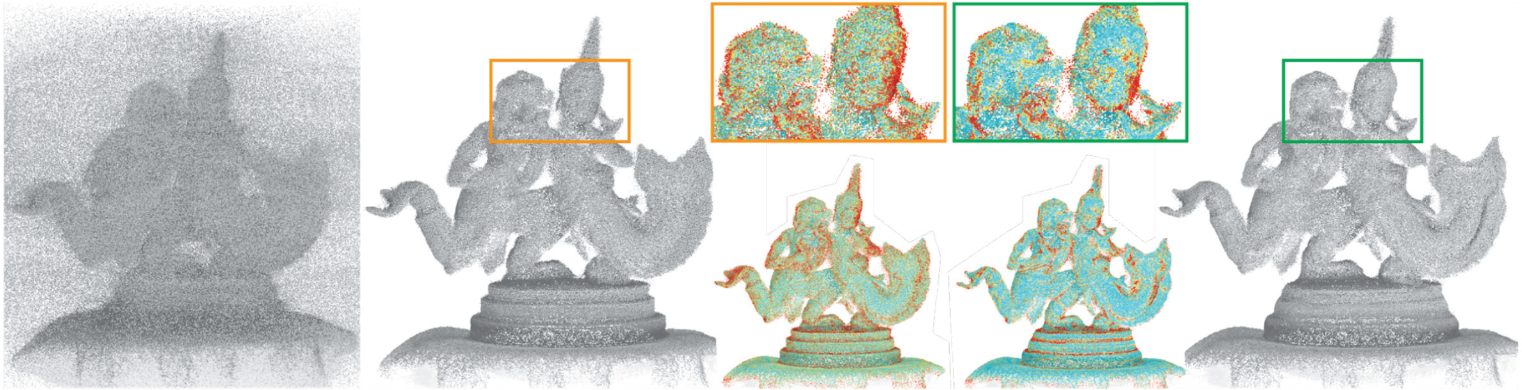
Results on real data



PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds,
M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. Mitra, M. O., 2019

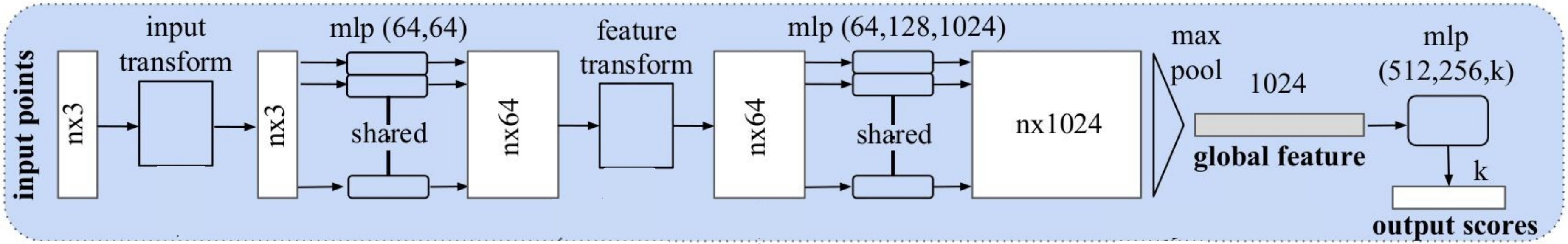
PointCleanNet – Results

Results on real data

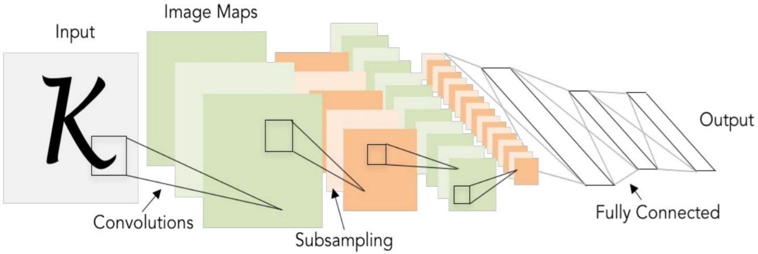


PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds,
M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. Mitra, M. O., 2019

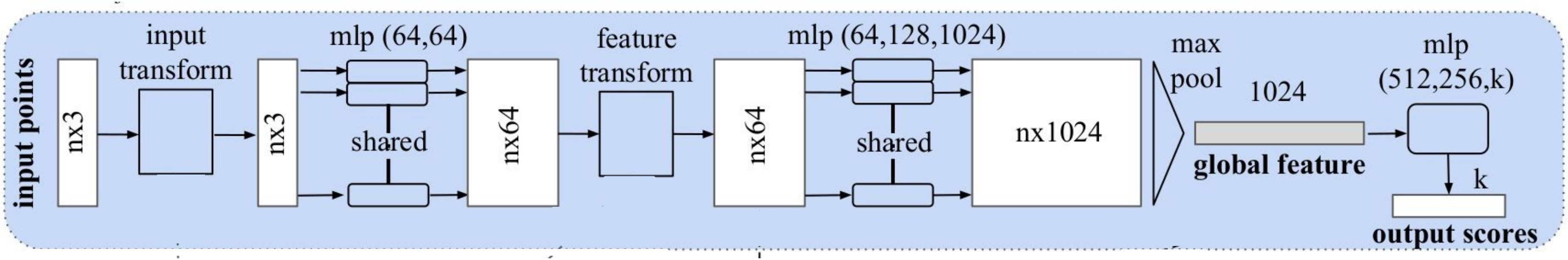
Limitations of PointNet



Does not extract a sequence of hierarchical features; except a global feature

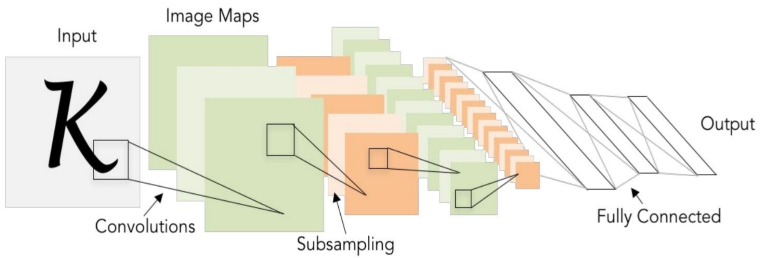


Limitations of PointNet

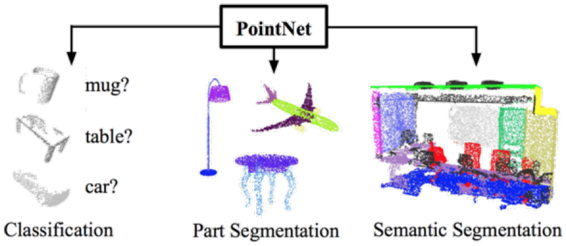


Does not extract a sequence *of hierarchical features* (like CNNs); except a global feature

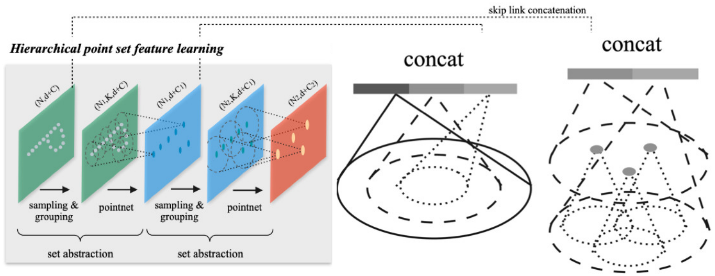
Does not take into account the *local geometry* formed by points.



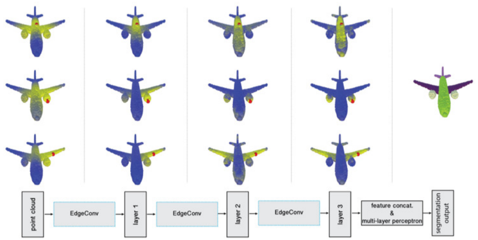
Point-based Architectures



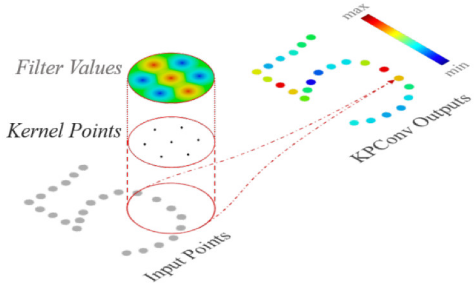
PointNet



PointNet++



DGCNN (EdgeConv)

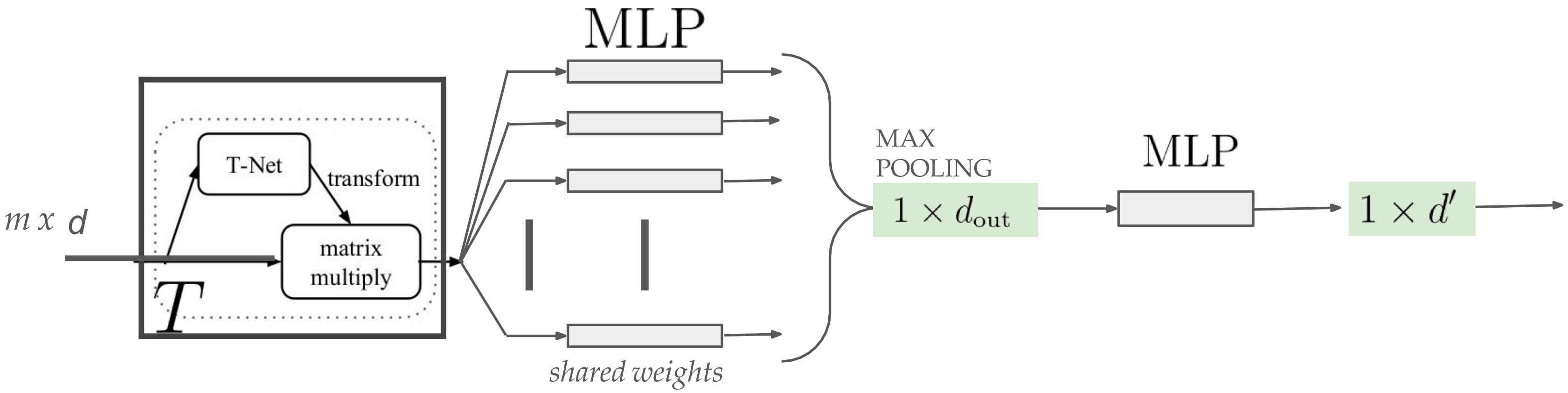


KPConv

PointNet++

Uses PointNet module as a building block

Transforms a set of m points to a single point with a feature vector



PointNet module

Extracts hierarchical features by recursively applying PointNet module

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

PointNet++

Sampling

Samples n' points using farthest point sampling

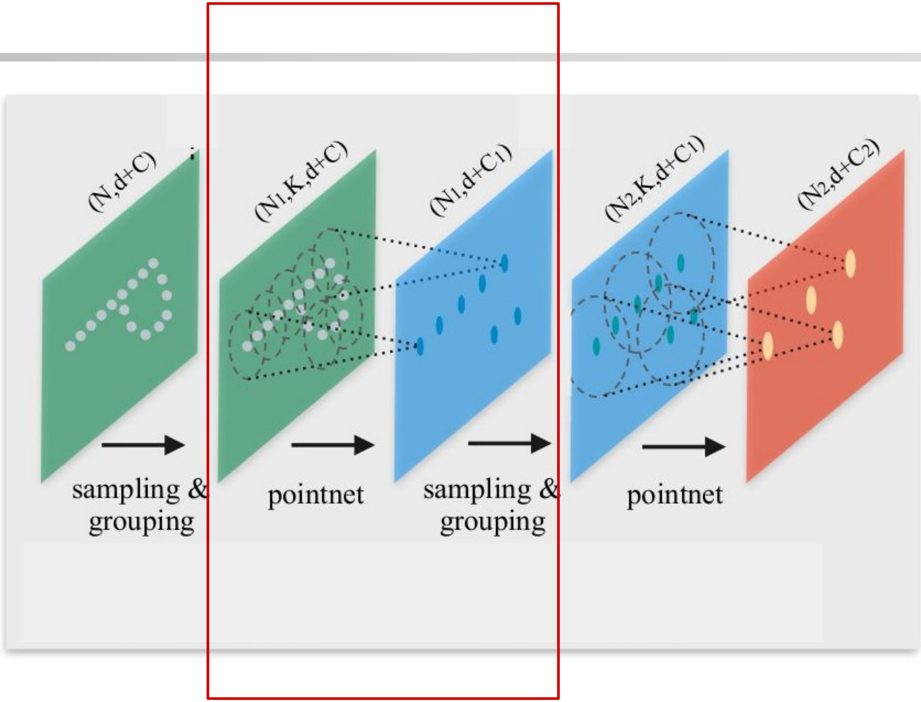
Grouping

For each of the sampled point, selects K points using either

- K-nearest neighbors or
- K points within maximum radius of R

PointNet Layer

Applies PointNet-module to each K-grouping of points and generates a feature vector



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

PointNet++

Sampling

Samples n' points using farthest point sampling

Grouping

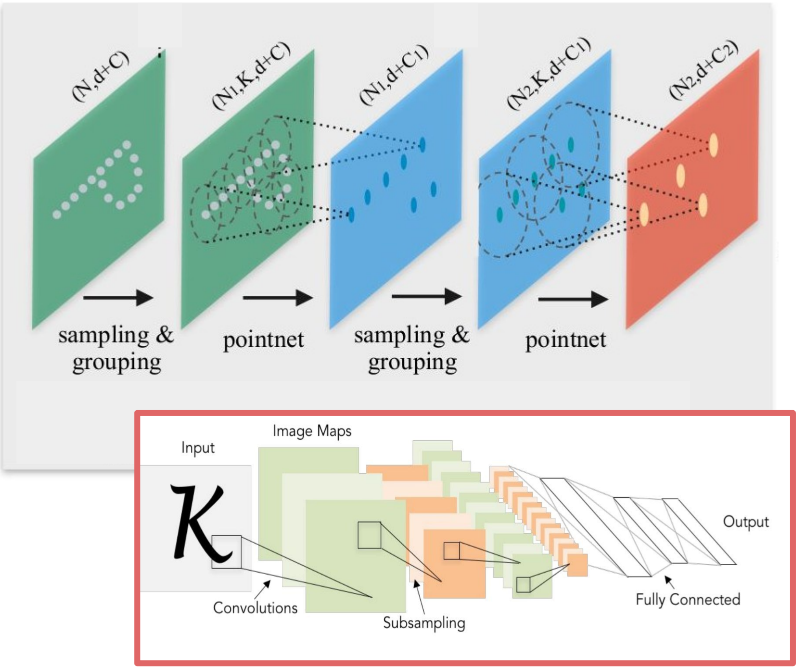
For each of the sampled point, selects K points using either

- K-nearest neighbors or
- K points within maximum radius of R

PointNet Layer

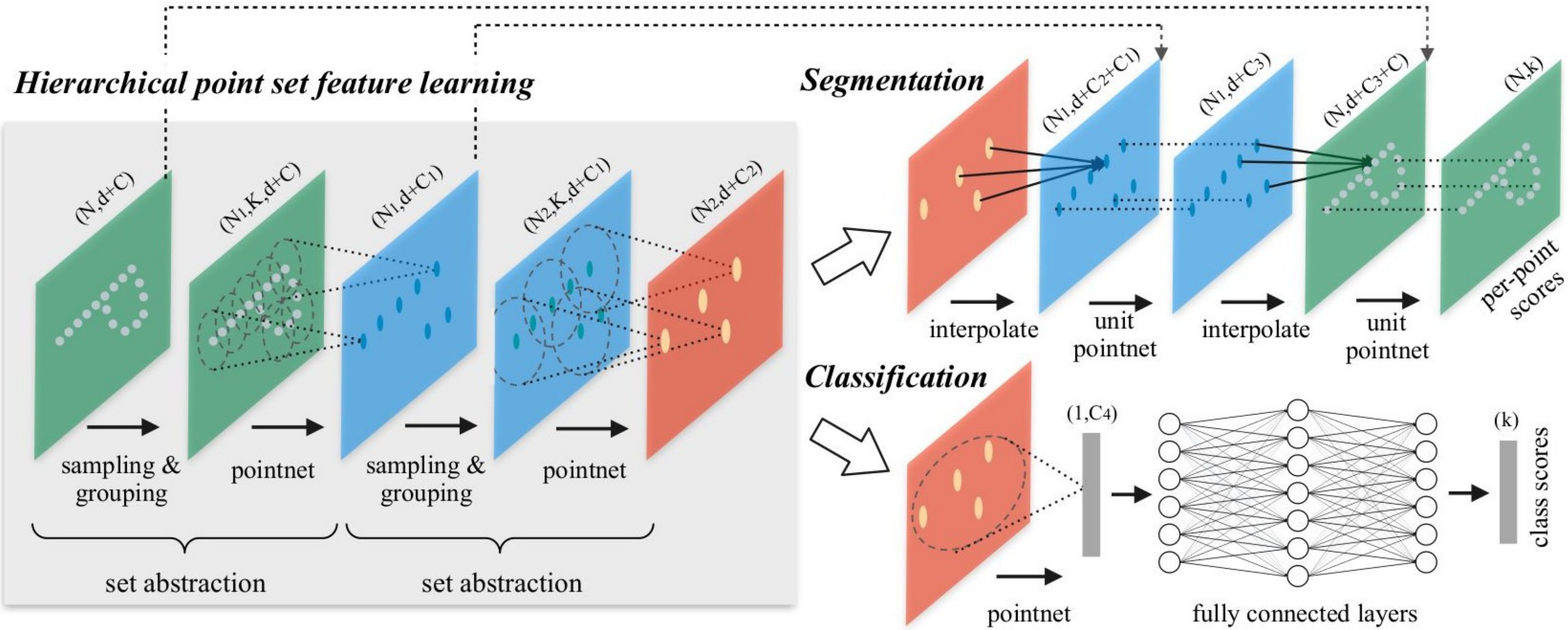
Applies PointNet-module to each K-grouping of points and generates a feature vector

Similar to convolution + pooling!



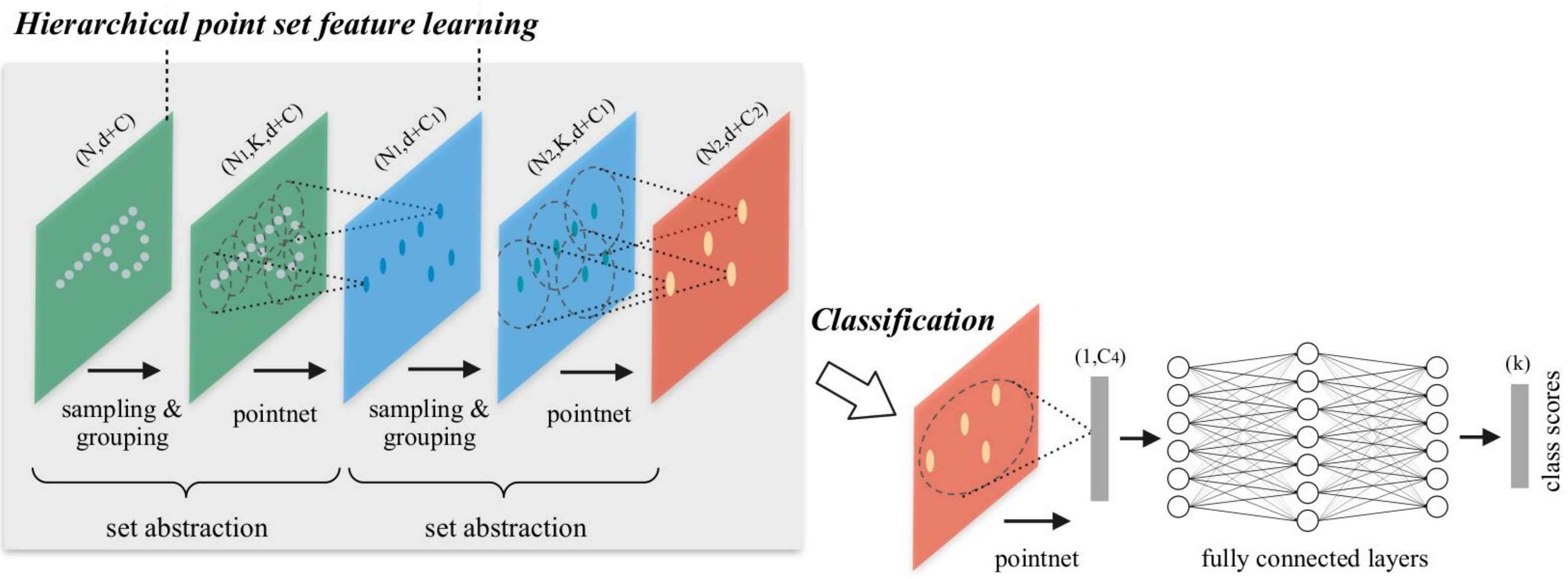
Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space" 2017

PointNet++ for Classification and Segmentation



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

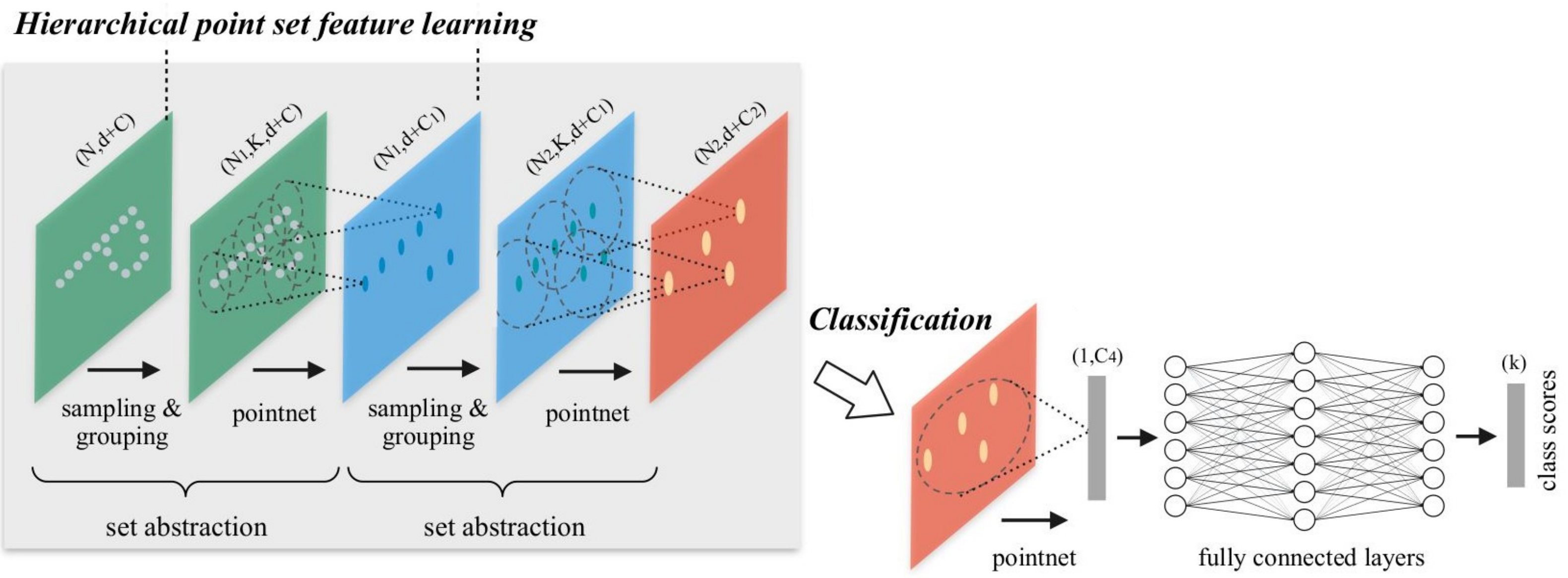
PointNet++ for Classification



Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space” 2017

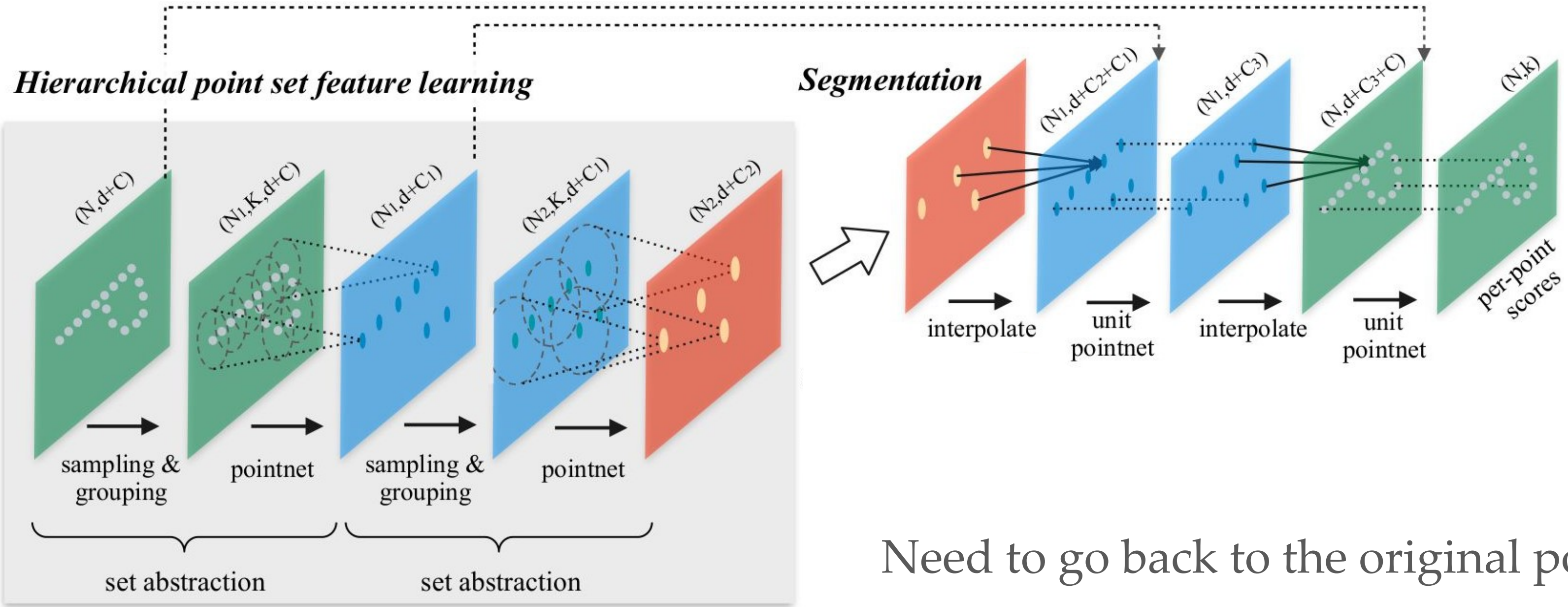
PointNet++ for Classification

Max Pool + MLP on features of the final layer



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

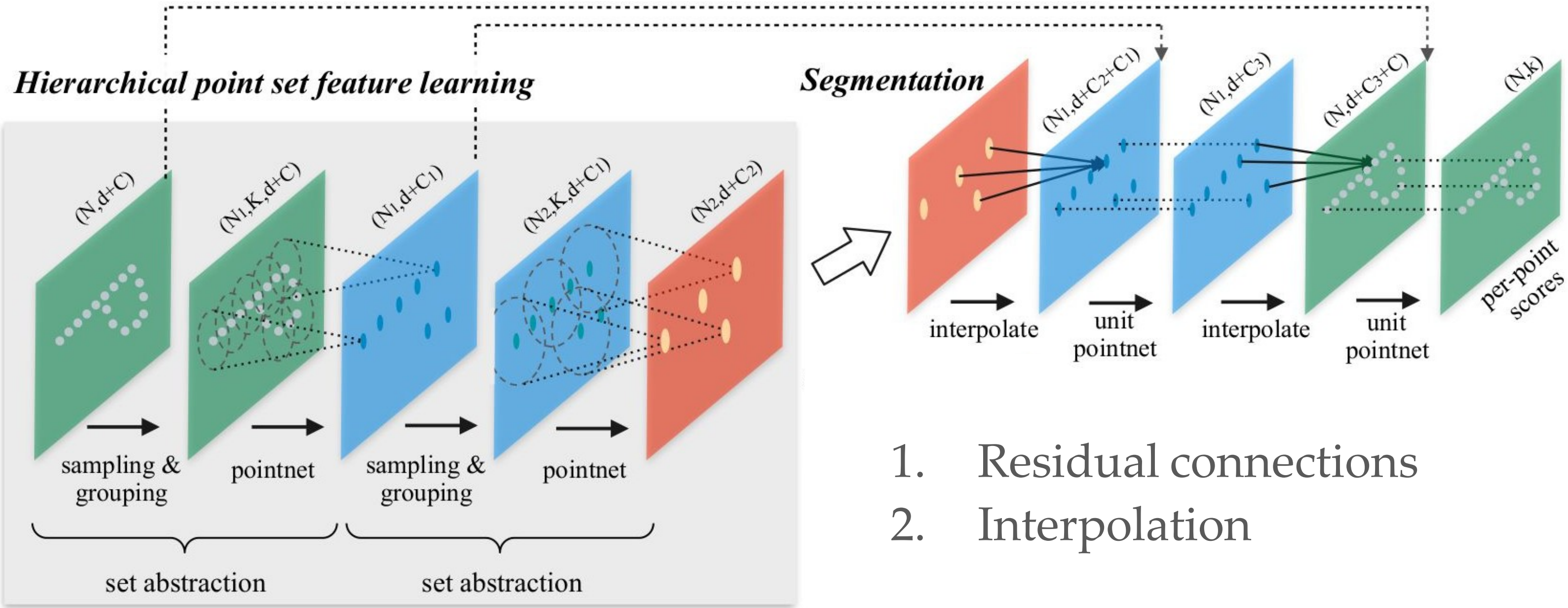
PointNet++ for Segmentation



Need to go back to the original points

Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space” 2017

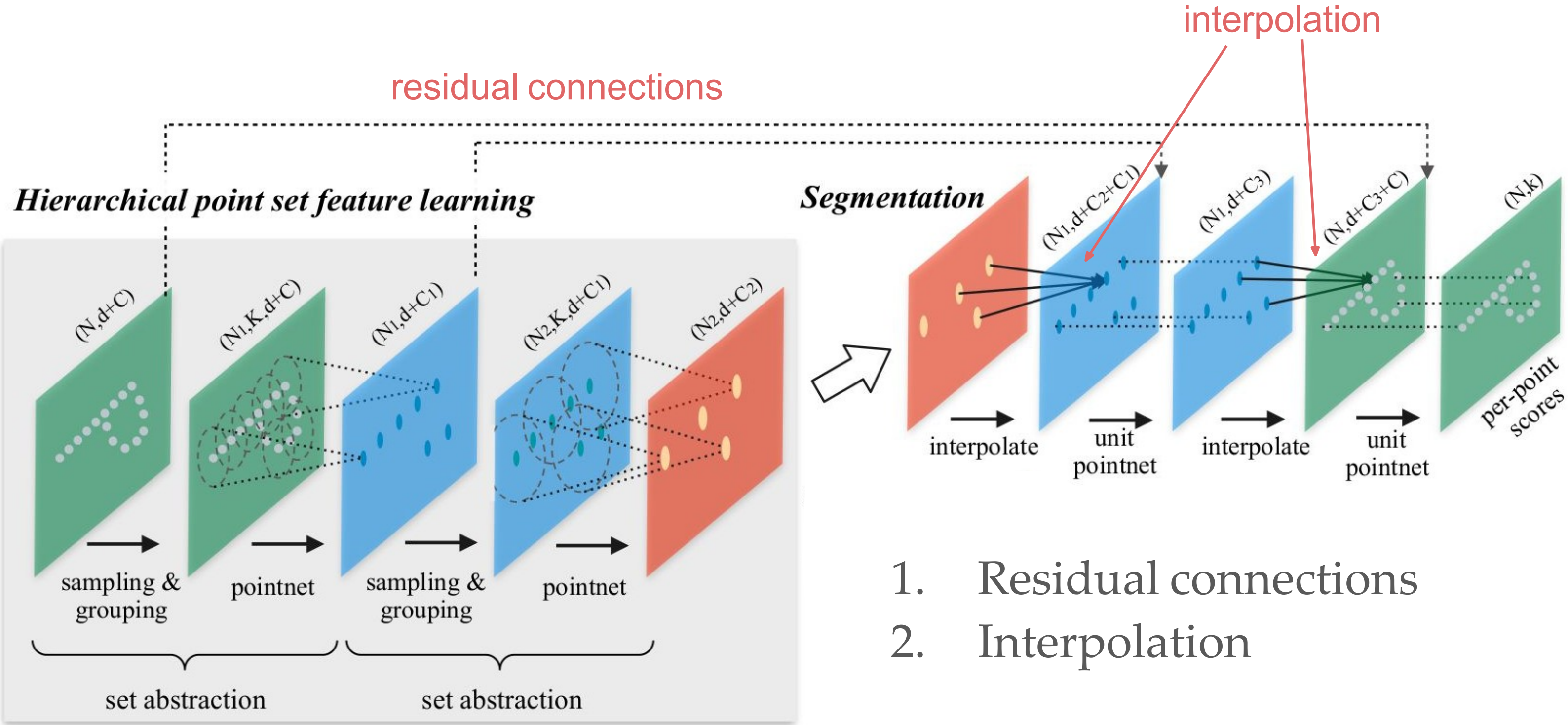
PointNet++ for Segmentation



1. Residual connections
2. Interpolation

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

PointNet++ for Segmentation



1. Residual connections
2. Interpolation

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

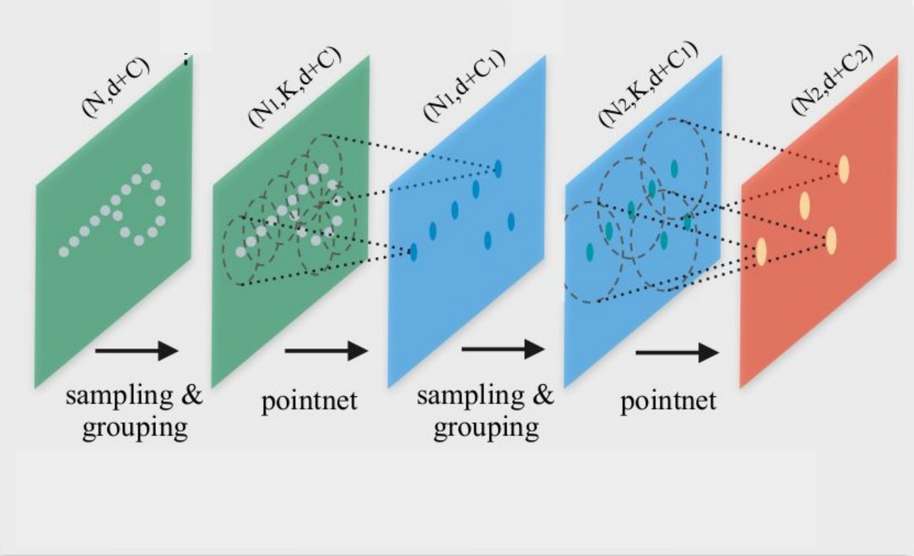
Non-uniform Point Density

PointNet and PointNet ++

implicitly assumes uniform point density

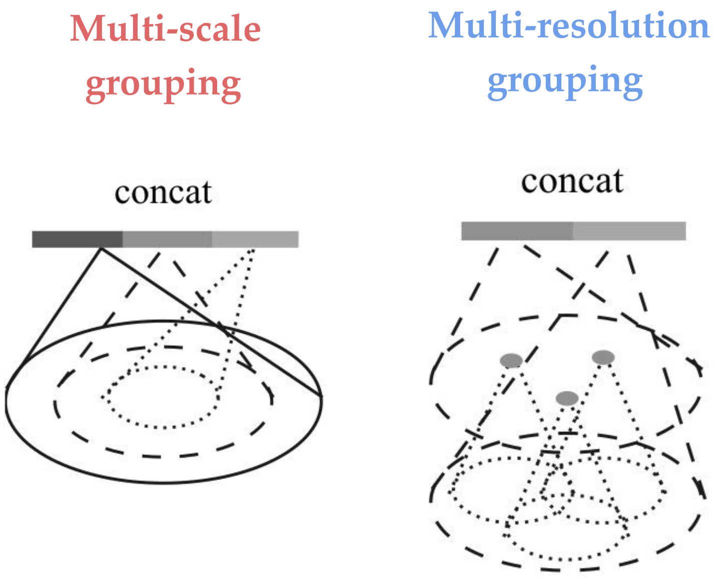
- e.g. k-nearest neighbors in the grouping

Becomes fragile with non-uniform point density

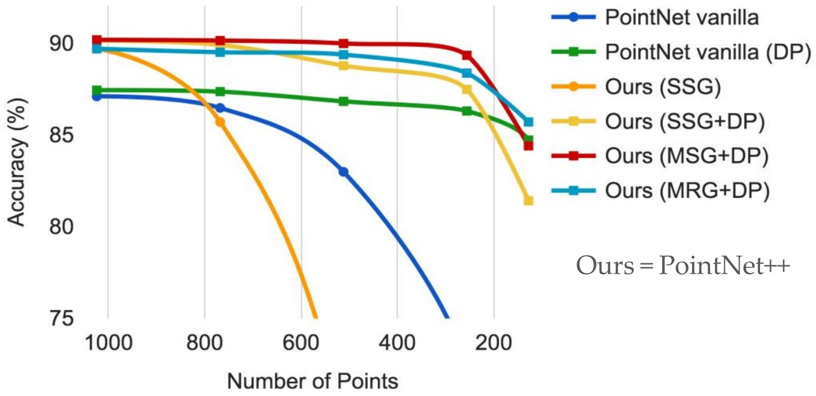


Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space" 2017

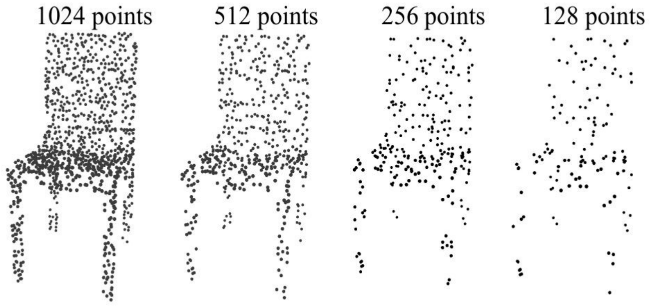
Fix for Non-uniform Point Density



+ Random Point Dropout at Training



Ours = PointNet++



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

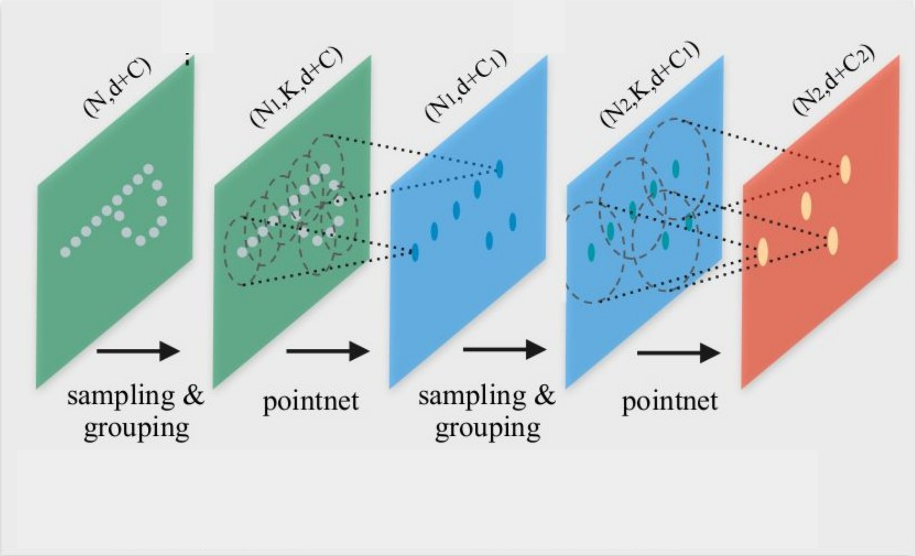
PointNet++

Better Performance than PointNet

Increased Compute Time

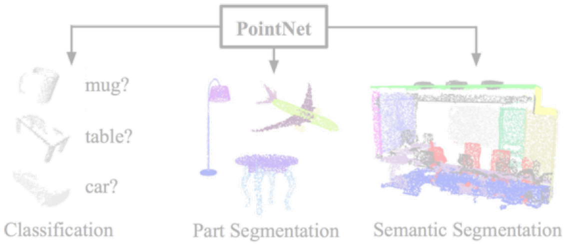
Might not take into account the *local relations* between points

Geometry of hierarchical features is pre-determined

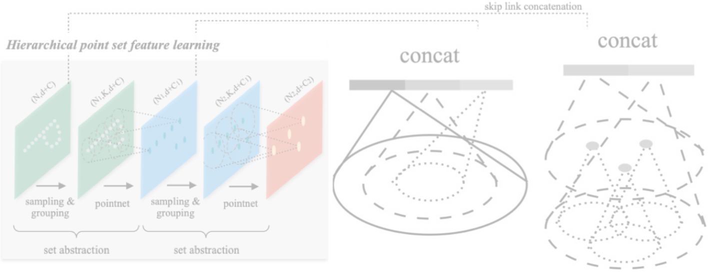


Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space" 2017

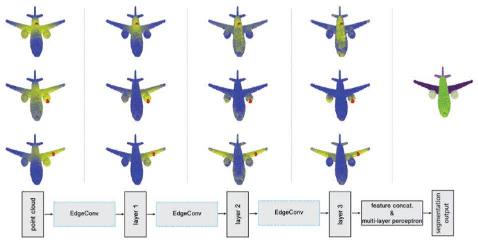
Point-based Architectures



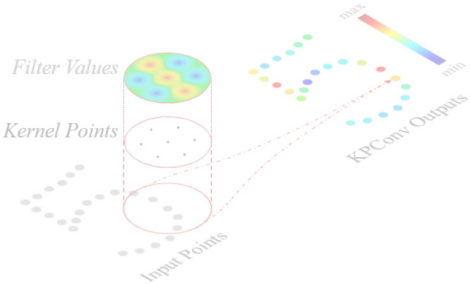
PointNet



PointNet++



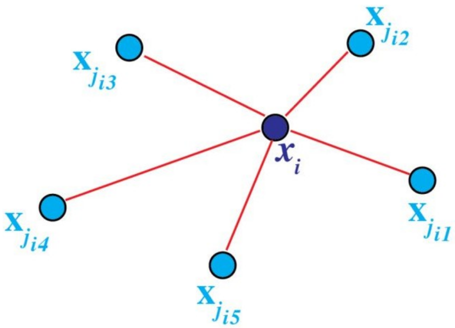
DGCNN (EdgeConv)



KPConv

DGCNN (EdgeConv): Basic Idea

Form a local graph by connecting nearby points

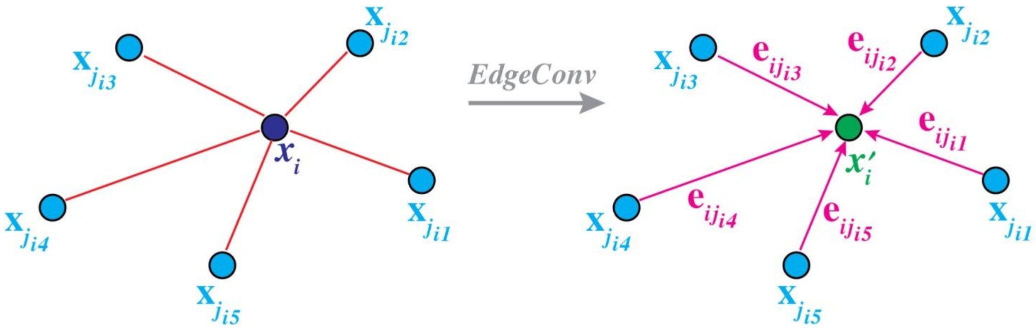


Wang et al. “Dynamic Graph CNN for Learning on Point Clouds” ACM Trans. Graph 2019

DGCNN (EdgeConv): Basic Idea

Form a local graph by connecting nearby points

Apply convolution-like operation on this graph



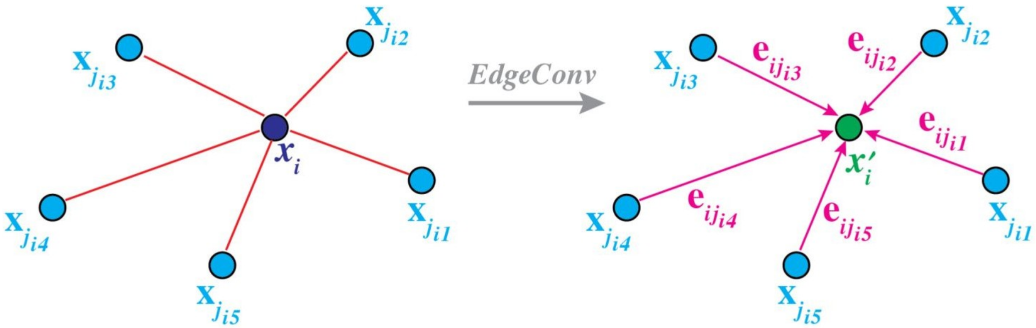
$$x_i' = \square_{j:(i,j)\in E} \; h_\Theta(x_i, x_j)$$

Wang et al. “Dynamic Graph CNN for Learning on Point Clouds” ACM Trans. Graph 2019

DGCNN (EdgeConv): Basic Idea

Form a local graph by connecting nearby points

Apply convolution-like operation on this graph



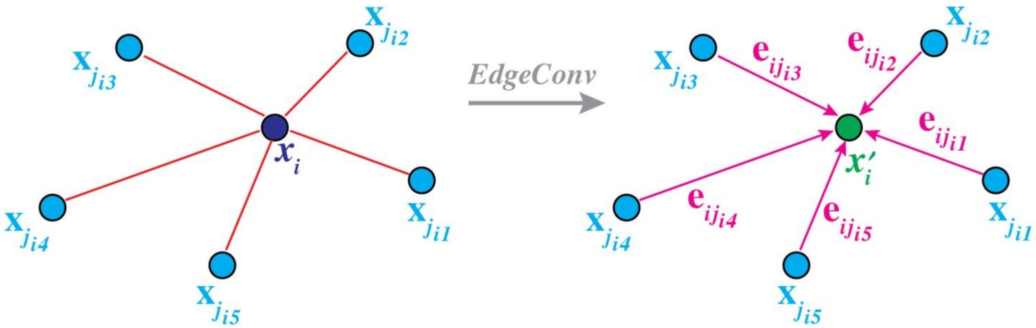
$$x_i' = \square_{j:(i,j)\in E} \; h_{\Theta}(x_i, x_j)$$

invariant function like max or sum

DGCNN (EdgeConv): Basic Idea

Form a local graph by connecting nearby points

Apply convolution-like operation on this graph



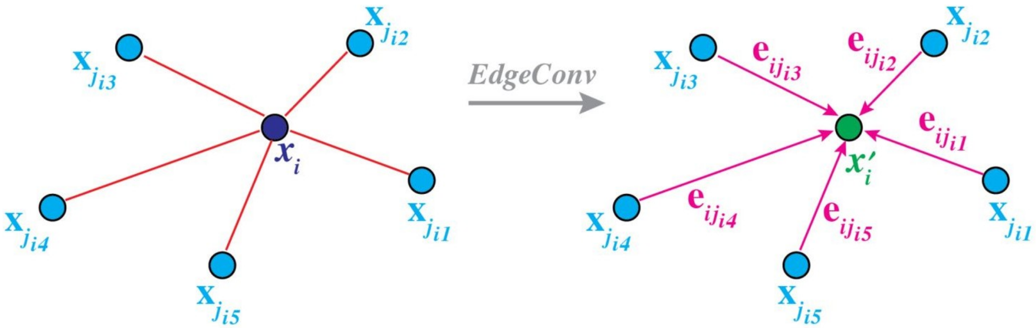
$$x'_i = \square_{j:(i,j)\in E}\ h_\Theta(x_i, x_j)$$

invariant function like max or sum

Nearby: with respect to node feature vectors x_i

EdgeConv: Basic Idea

Form a local graph by connecting nearby points



PointNet++

Connects k-NN from position of points

EdgeConv

Connects k-NN from feature vectors of points

Does this at each layer

EdgeConv Architecture

Step 1: Form a local graph by connecting nearby points with respect to x_i

Step 2: Update feature vectors

$$x_i \leftarrow x_i' = \square_{j:(i,j)\in E} \; h_\Theta(x_i, x_j)$$

EdgeConv Architecture

Step 1: Form a local graph by connecting nearby points with respect to x_i

Step 2: Update feature vectors

$$x_i \leftarrow x_i' = \square_{j:(i,j)\in E} \; h_\Theta(x_i, x_j)$$

iterate

Need to compute a new graph at each stage

EdgeConv Architecture

Step 1: Form a local graph by connecting nearby points with respect to x_i

Step 2: Update feature vectors

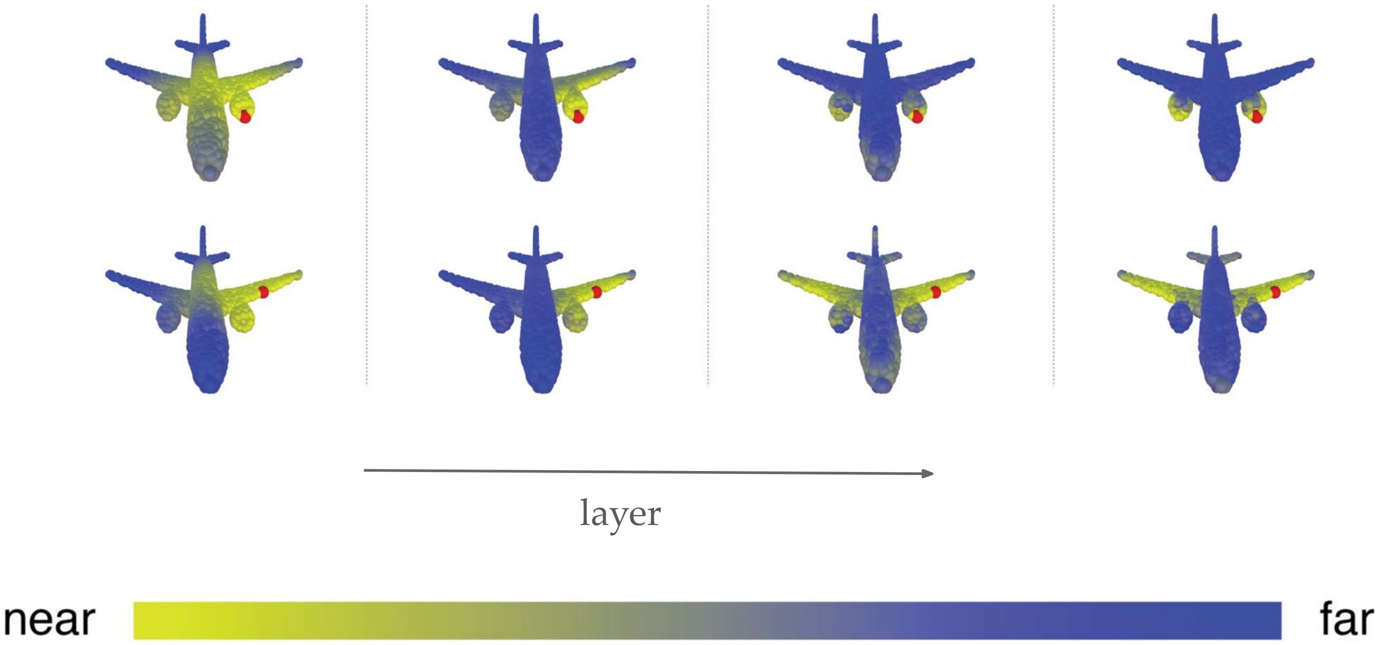
$$x_i \leftarrow x_i' = \square_{j:(i,j)\in E} \; h_\Theta(x_i, x_j)$$

Example

iterate

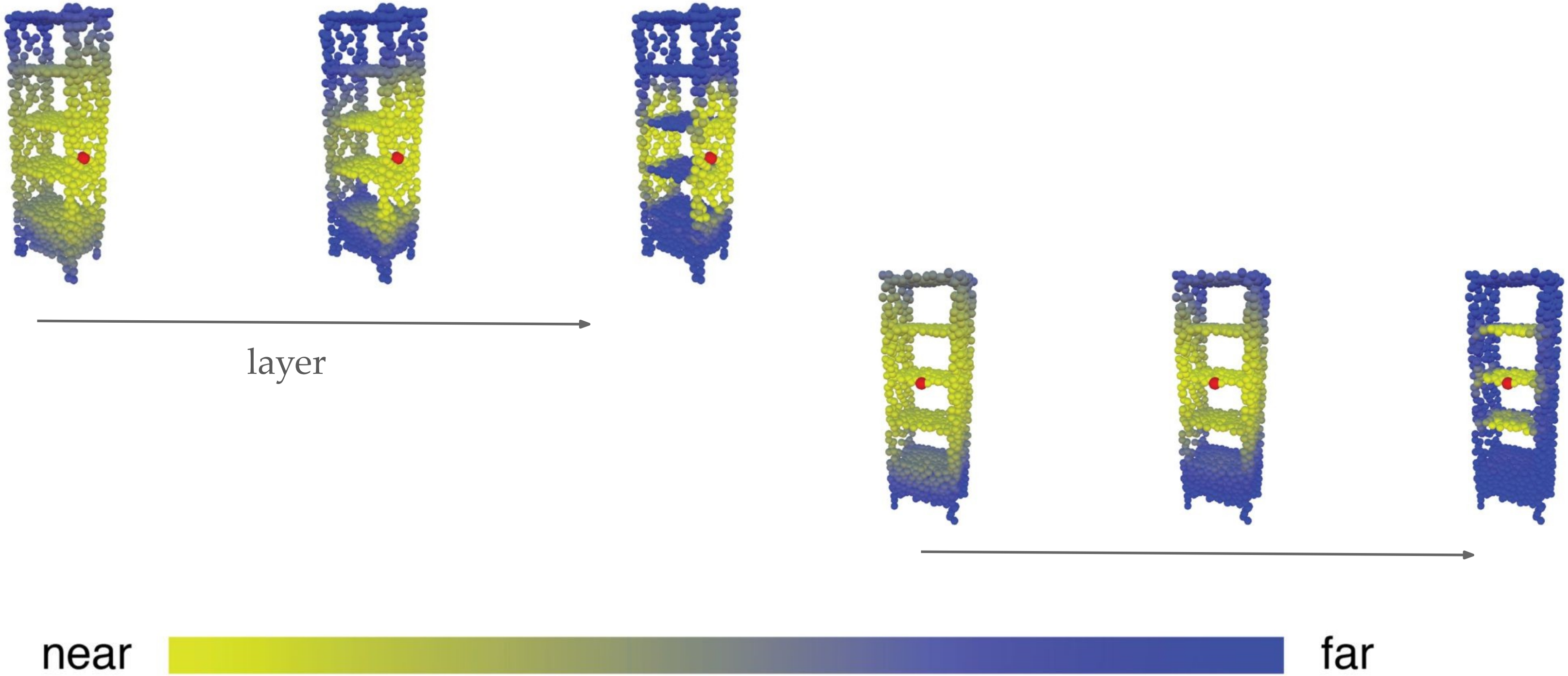
$$h_\Theta(x_i, x_j) = \sigma(\Theta_a \cdot (x_j - x_i) + \Theta_b x_i)$$

Feature Space and Semantically Similar Structures



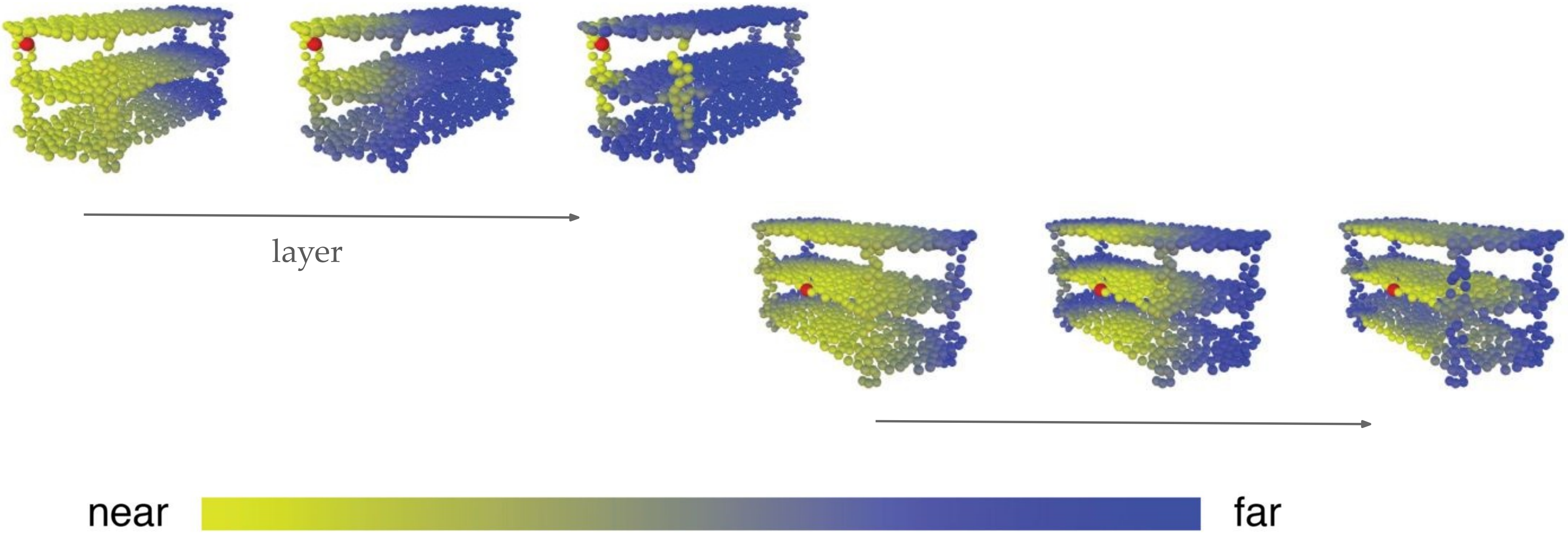
Wang et al. "Dynamic Graph CNN for Learning on Point Clouds" ACM Trans. Graph 2019

Feature Space and Semantically Similar Structures



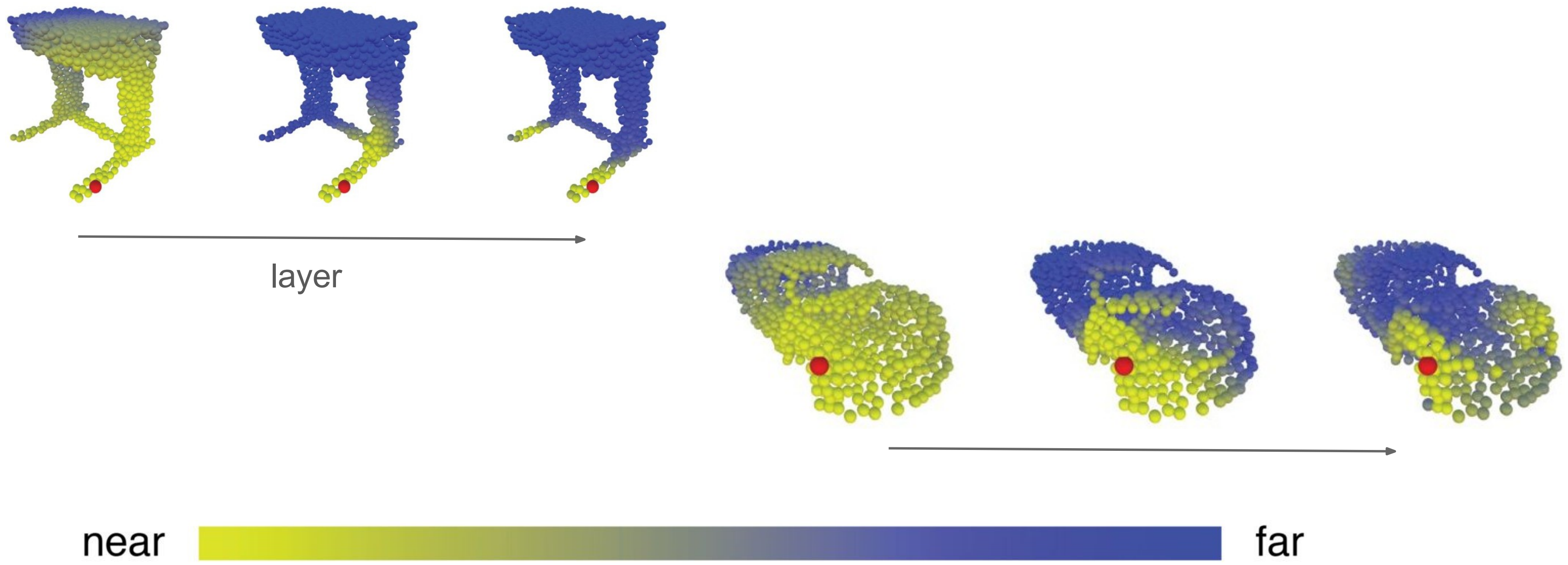
Wang et al. “Dynamic Graph CNN for Learning on Point Clouds” ACM Trans. Graph 2019

Feature Space and Semantically Similar Structures



Wang et al. "Dynamic Graph CNN for Learning on Point Clouds" ACM Trans. Graph 2019

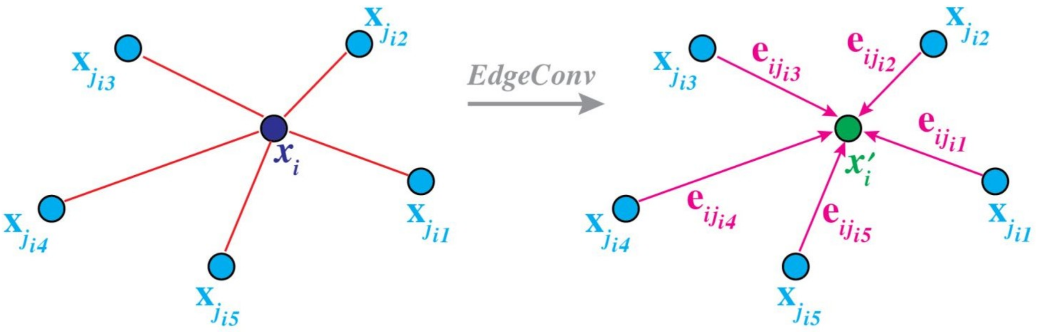
Feature Space and Semantically Similar Structures



Wang et al. "Dynamic Graph CNN for Learning on Point Clouds" ACM Trans. Graph 2019

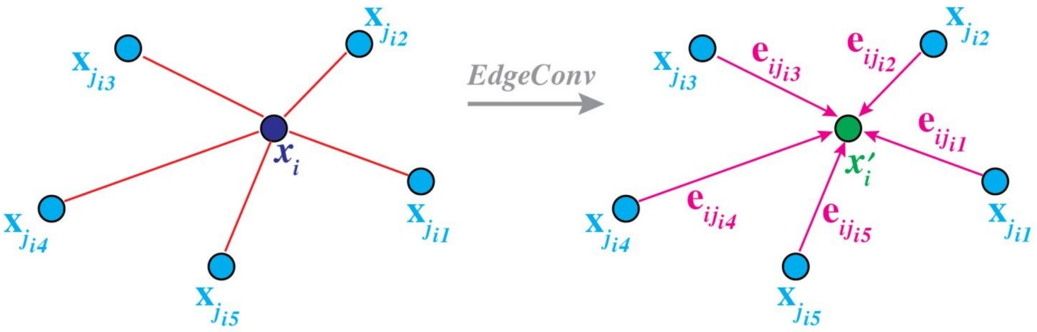
Limitations of EdgeConv

Computationally more expensive than PointNet and PointNet++



Limitations of EdgeConv

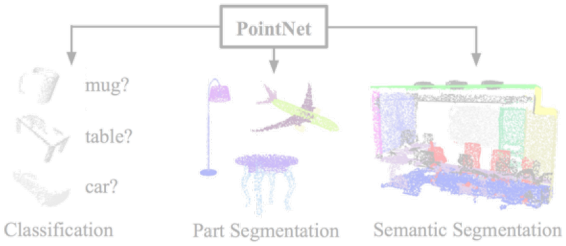
Computationally more expensive than PointNet and PointNet++



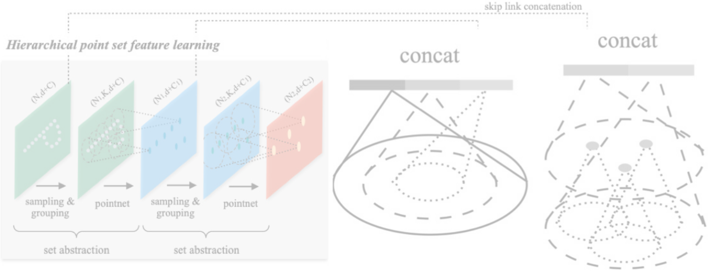
Is this really a convolution operation?

$$x_i \leftarrow x'_i = \square_{j:(i,j)\in E} \; h_\Theta(x_i, x_j)$$

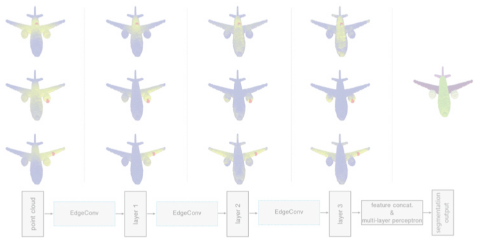
Point-based Architectures



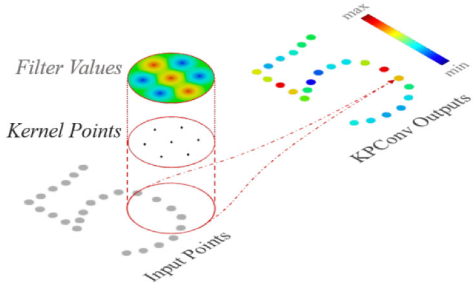
PointNet



PointNet++



DGCNN (EdgeConv)

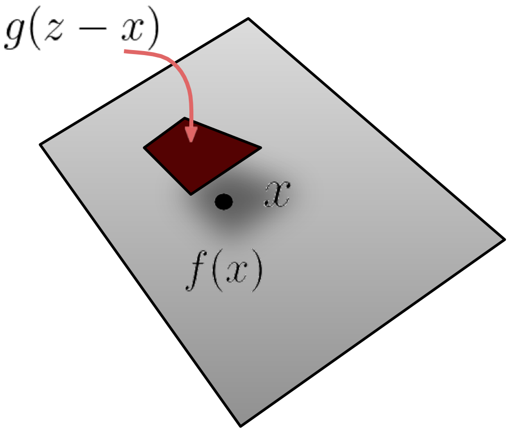


KPConv

Convolution based architectures for point clouds

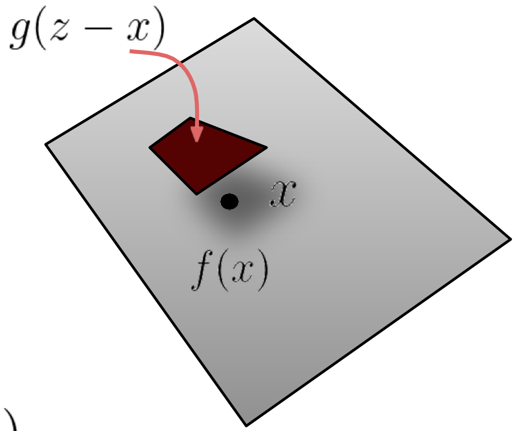
Convolution

$$(f * g)(x) = \int_{\mathcal{X}} f(z)g(z - x)dz$$

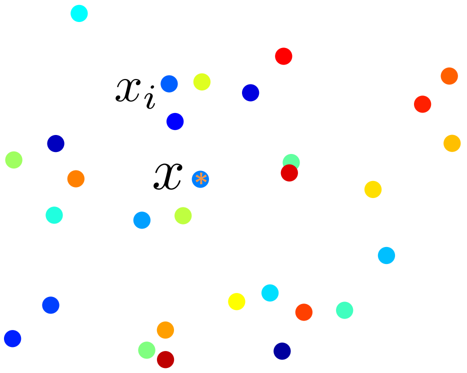


Convolution on Point Clouds?

$$(f * g)(x) = \int_{\mathcal{X}} f(z)g(z-x)dz$$



We only have points on $\mathcal{X}$

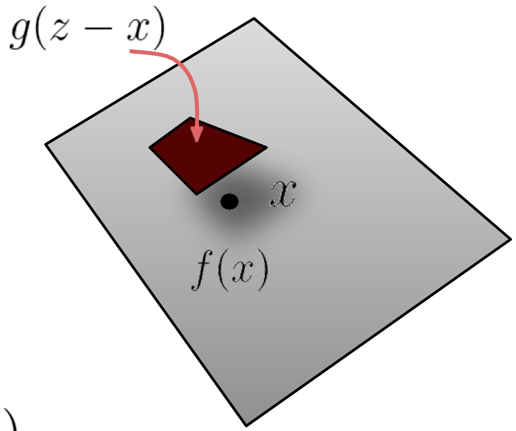


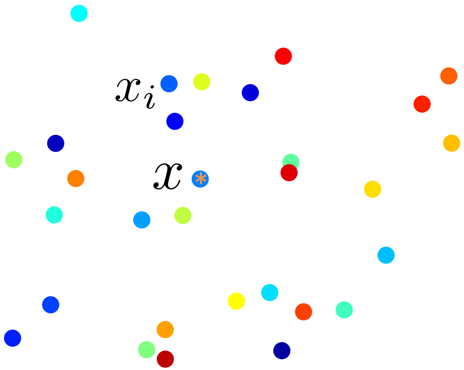
$f(x_i)$

$$\mathcal{F} = \{(x_i, f_i)\}_i$$

Convolution on Point Clouds?

$$(\mathcal{F} * g)(x) = \sum_i f(x_i) g(x_i - x)$$





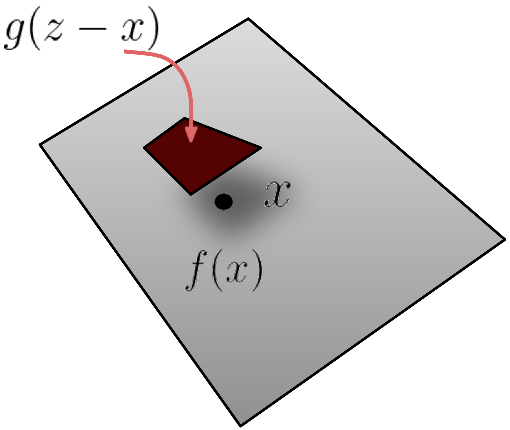
$f(x_i)$

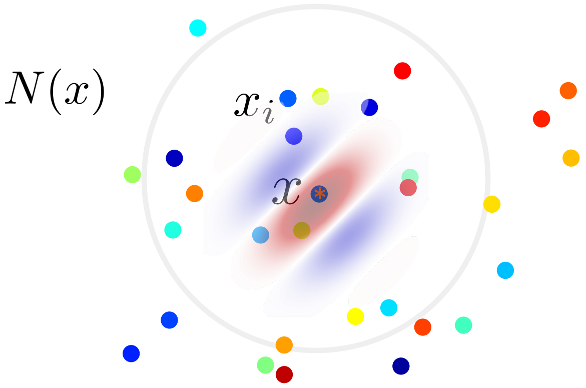
$$\mathcal{F} = \{(x_i, f_i)\}_i$$

Convolution on Point Clouds

$$(\mathcal{F} * g)(x) = \sum_{i \in N(x)} f_i \cdot g(x_i - x)$$

neighborhood of x





$$\mathcal{F} = \{(x_i, f_i)\}_i$$

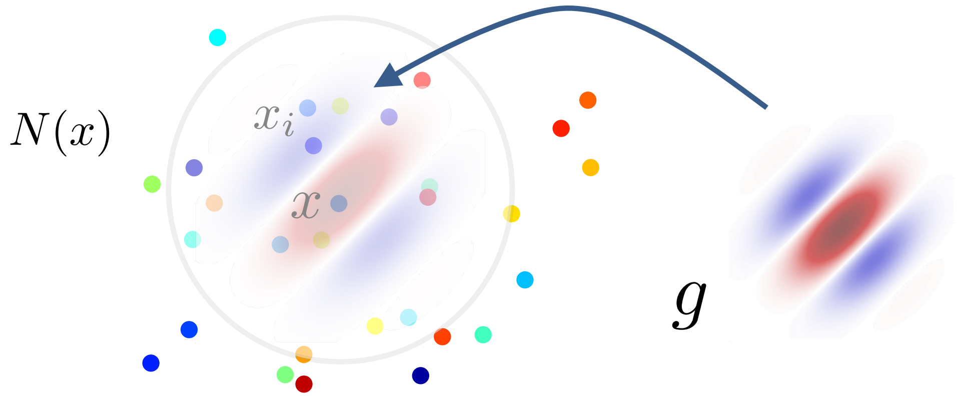
Convolution on Point Clouds

$$(\mathcal{F} * g)(x) = \underbrace{\sum_{i \in N(x)}}_{\text{Neighborhood}} f_i \cdot \underbrace{g}_{\text{Kernel}}(x_i - x)$$

Point Cloud

$$\mathcal{F} = \{(x_i, f_i)\}_i$$

Key question: how to represent the kernel function g ?



Convolution on Point Clouds

$$(\mathcal{F} * g)(x) = \sum_{i \in N(x)} f_i \cdot g(x_i - x)$$

Neighborhood

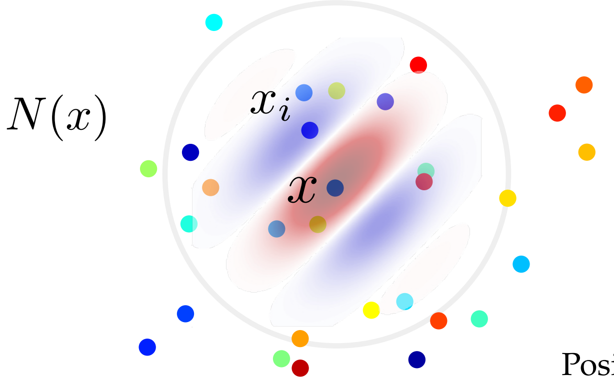
Kernel

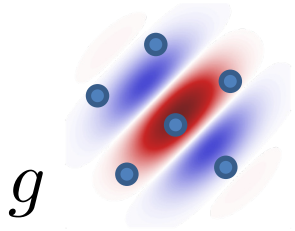
Point Cloud

$$\mathcal{F} = \{(x_i, f_i)\}_i$$

Key question: how to represent the kernel function g ?

Option: have discrete *kernel points*, and *interpolate elsewhere*.



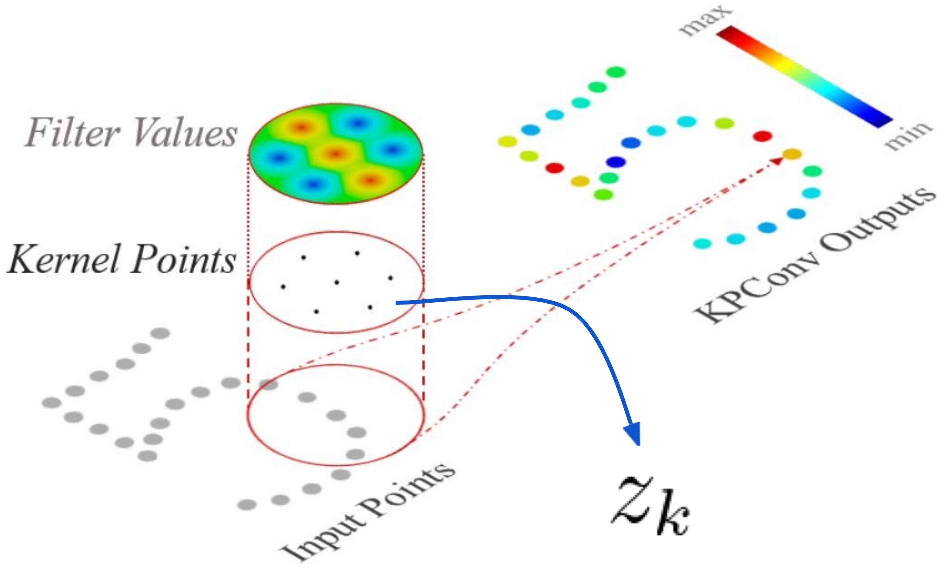


Positions of the kernel points are independent of the point cloud.

Kernel Point Convolution (KPConv)

$$g(z) = \sum_{1 \leq k \leq K} h(z, z_k) W_k$$

A specific choice of kernel function



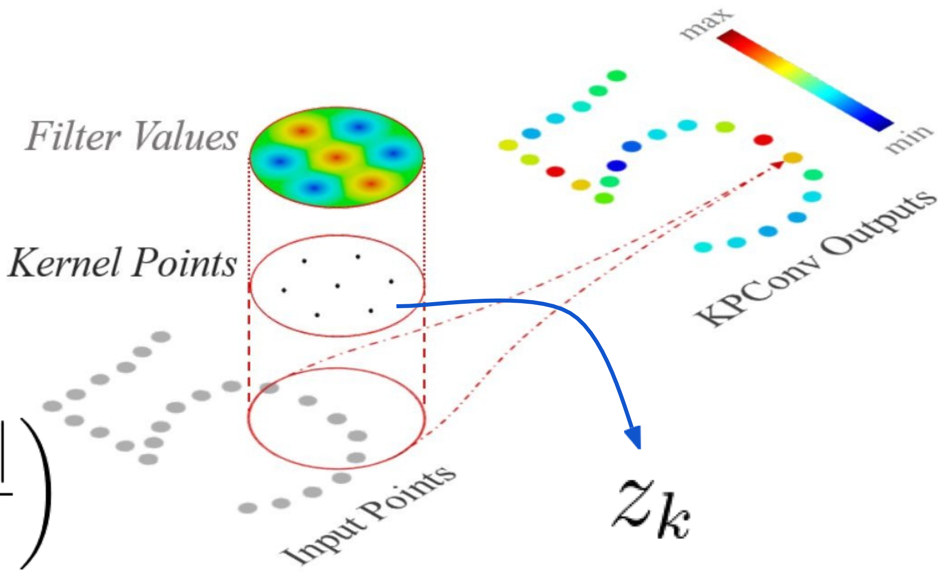
Thomas et al. "KPConv: Flexible and Deformable Convolution for Point Clouds" 2019

Kernel Point Convolution (KPConv)

$$g(z) = \sum_{1 \leq k \leq K} h(z, z_k) W_k$$

where

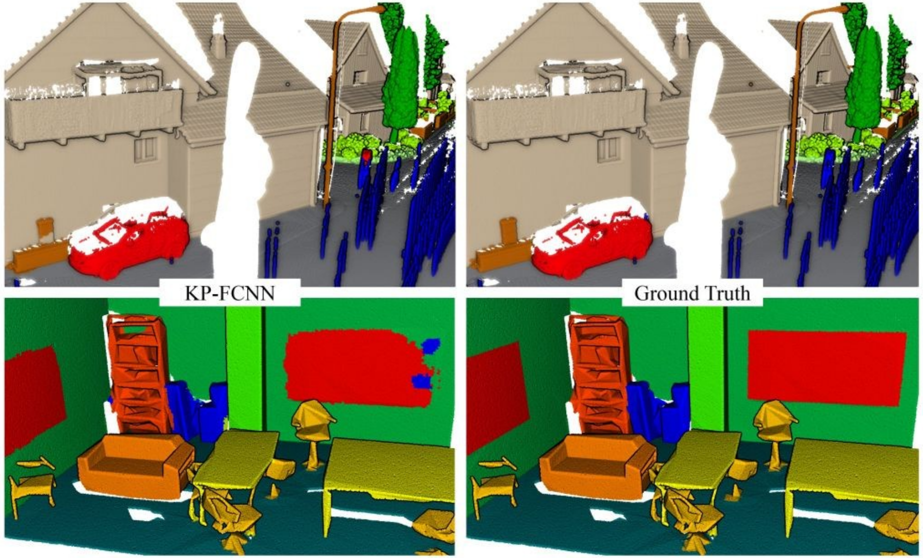
$$h(z, z_k) = \max\left(0, 1 - \frac{||z - z_k||}{\sigma}\right)$$



Thomas et al. "KPConv: Flexible and Deformable Convolution for Point Clouds" 2019

KPConv Performance

	ModelNet40	ShapeNetPart	
Methods	OA	mcIoU	mIoU
SPLATNet [34]	-	83.7	85.4
SGPN [42]	-	82.8	85.8
3DmFV-Net [9]	91.6	81.0	84.3
SynSpecCNN [48]	-	82.0	84.7
RSNet [15]	-	81.4	84.9
SpecGCN [40]	91.5	-	85.4
PointNet++ [27]	90.7	81.9	85.1
SO-Net [19]	90.9	81.0	84.9
PCNN by Ext [2]	92.3	81.8	85.1
SpiderCNN [45]	90.5	82.4	85.3
MCConv [13]	90.9	-	85.9
FlexConv [10]	90.2	84.7	85.0
PointCNN [20]	92.2	84.6	86.1
DGCNN [43]	92.2	85.0	84.7
SubSparseCNN [9]	-	83.3	86.0
KPConv *rigid*	**92.9**	85.0	86.2
KPConv *deform*	92.7	**85.1**	**86.4**

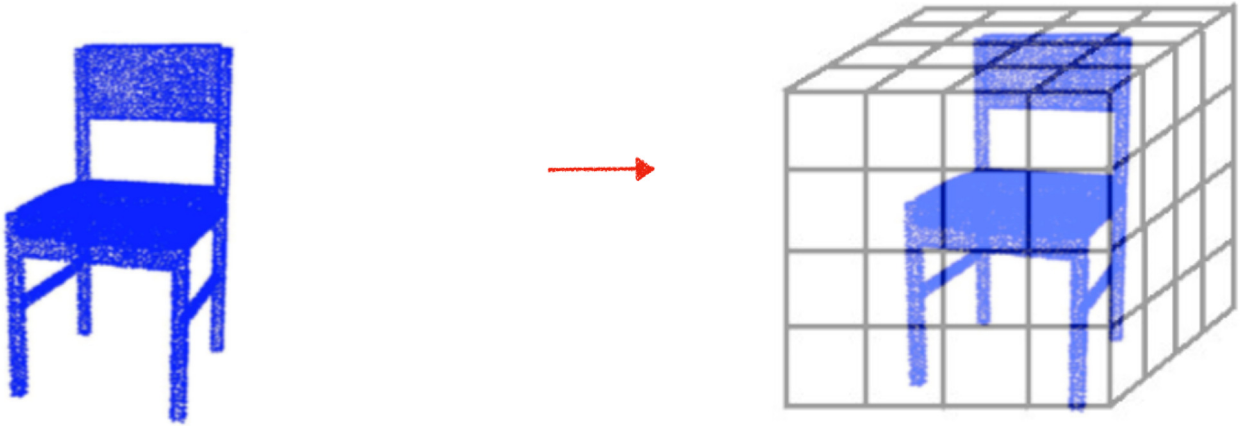


Convolution-based approaches often perform better than PointNet, etc. especially on *local* understanding tasks, such as semantic segmentation.

Thomas et al. “KPConv: Flexible and Deformable Convolution for Point Clouds” 2019

Sparse Volumes: An Alternate Approach

Approaches so far:

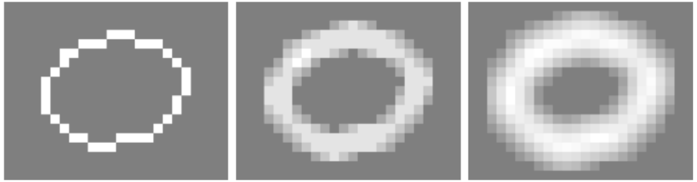


Point cloud: N X 3 array
(or N X (3 + k) if additional pointwise features)

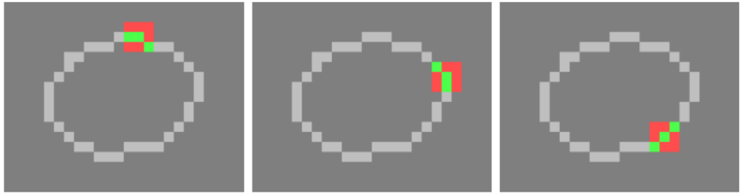
(Sparsely Occupied) 3D Grid
with per-voxel occupancy + optional features

4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks, Choy et al. 2019

Sparse Volumes: An Alternate Approach



A 'normal' convolution spreads information to initially empty regions



Sparse convolution: Unoccupied cells always have zero features (i.e. only apply operator on occupied cells)

(analogous to a graph)

4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks, Choy et al. 2019

Sparse Volumes: An Alternate Approach

Minkowski Engine enables convolution with sparse tensors

3D: XYZ

4D: XYZ + time

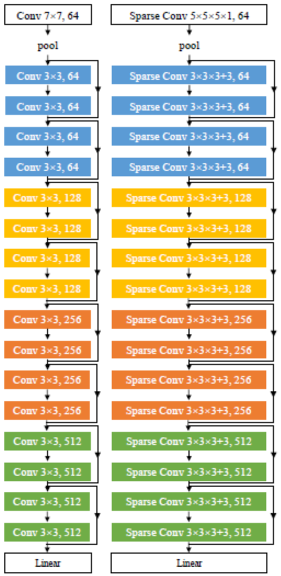


Figure 4: Architecture of ResNet18 (left) and MinkowskiNet18 (right). Note the structural similarity. × indicates a hypercubic kernel, + indicates a hypercross kernel. (best viewed on display)

4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks, Choy et al. 2019

Sparse Convolution for semantic segmentation

Figure 7: Visualization of Scannet predictions. From the top, a 3D input pointcloud, a network prediction, and the ground-truth.

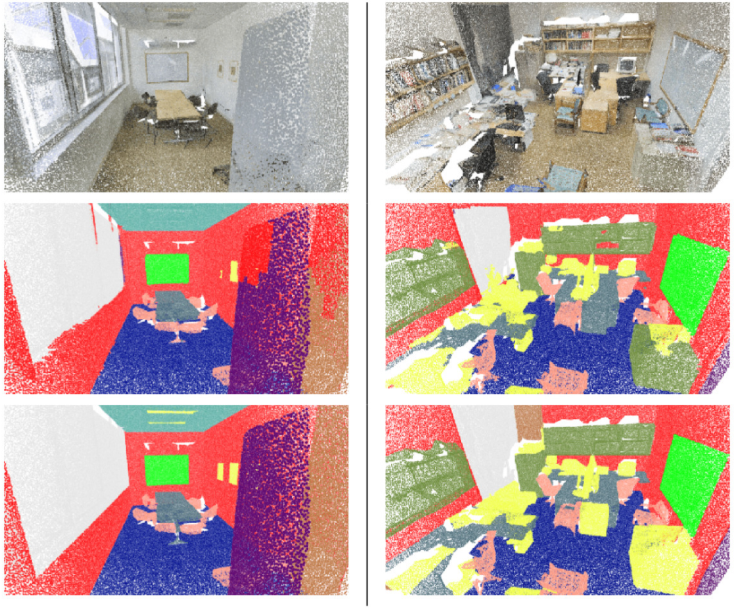


Figure 8: Visualization of Stanford dataset Area 5 test results. From the top, RGB input, prediction, ground truth.

4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks, Choy et al. 2019

Sparse Convolution for semantic segmentation

Table 1: 3D Semantic Label Benchmark on ScanNet† [5]

Method	mIOU
ScanNet [5]	30.6
SSC-UNet [10]	30.8
PointNet++ [23]	33.9
ScanNet-FTSDF	38.3
SPLATNet [28]	39.3
TangetConv [29]	43.8
SurfaceConv [20]	44.2
3DMV‡ [6]	48.4
3DMV-FTSDF‡	50.1
PointNet++SW	52.3
MinkowskiNet42 (5cm)	**67.9**
SparseConvNet [10]†	72.5
MinkowskiNet42 (2cm)†	**73.4**

Easily scalable to scenes compared to PointNet/DGCNN based methods

Although comparable performance for object-level reasoning

Can be more robust to changes in point sampling (better for transfer learning).

4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks, Choy et al. 2019

PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding, Xie et al., 2020

Sparse Convolution for semantic segmentation

Table 1: 3D Semantic Label Benchmark on ScanNet† [5]

Method	mIOU
ScanNet [5]	30.6
SSC-UNet [10]	30.8
PointNet++ [23]	33.9
ScanNet-FTSDF	38.3
SPLATNet [28]	39.3
TangetConv [29]	43.8
SurfaceConv [20]	44.2
3DMV‡ [6]	48.4
3DMV-FTSDF‡	50.1
PointNet++SW	52.3
MinkowskiNet42 (5cm)	**67.9**
SparseConvNet [10]†	72.5
MinkowskiNet42 (2cm)†	**73.4**

Easily scalable to scenes compared to PointNet/DGCNN based methods

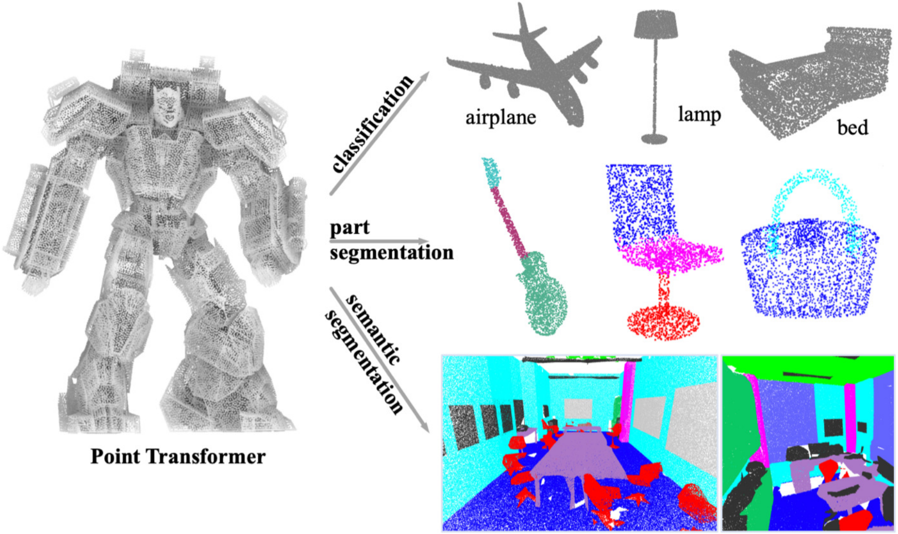
Although comparable performance for object-level reasoning

Can be more robust to changes in point sampling (better for transfer learning).

4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks, Choy et al. 2019

PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding, Xie et al., 2020

PointTransformer[s]



Many transformer architectures for point clouds.

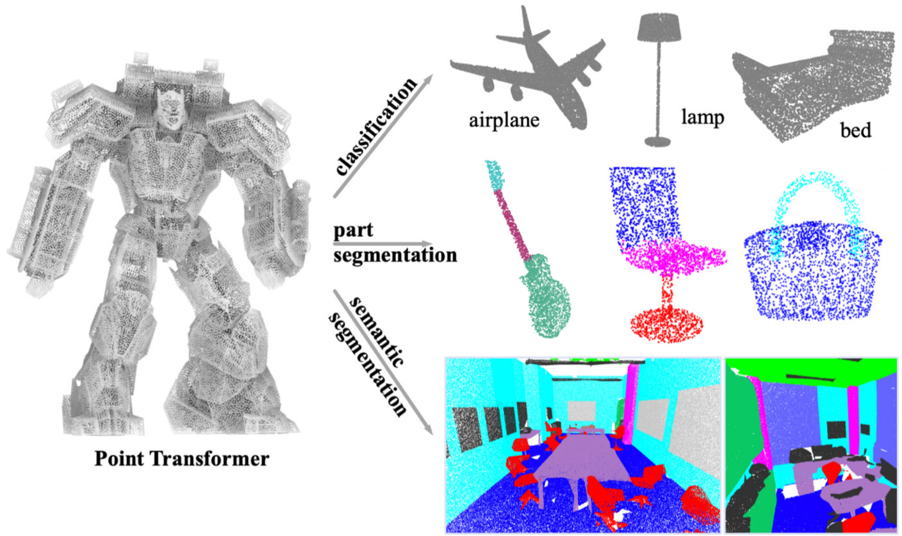
Natural fit since point couds are unordered sets anyway.

Often leads to more parameters/data to train on, but also better results.

"Apparently, the straightforward adoption of Transformers does not achieve satisfactory performance on point cloud tasks" [1]

[1] Yu, Xumin, et al. "Point-bert: Pre-training 3d point cloud transformers with masked point modeling." CVPR 2022.

PointTransformer[s]



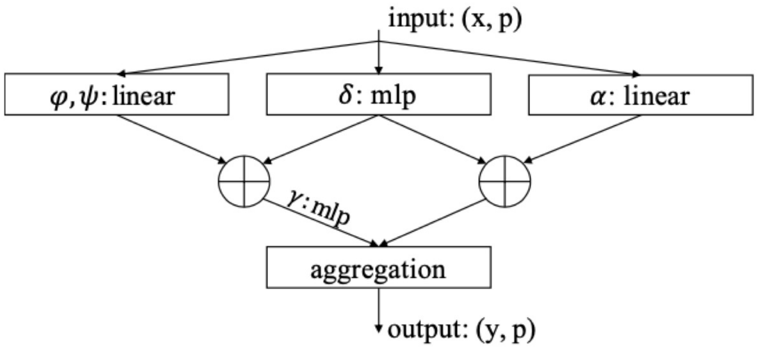
Many transformer architectures for point clouds.

Natural fit since point couds are unordered sets anyway.

Often leads to more parameters/data to train on, but also better results.

Zhao, et al. "Point transformer." CVPR 2021.
Guo et al. "PCT: Point cloud transformer," CVM, 2021.

PointTransformer[s]



Basic dot product self-attention:

$$\mathbf{y}_i = \sum_{\mathbf{x}_j \in \mathcal{X}} \rho\left(\varphi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j)\right) \alpha(\mathbf{x}_j),$$

φ : Queries

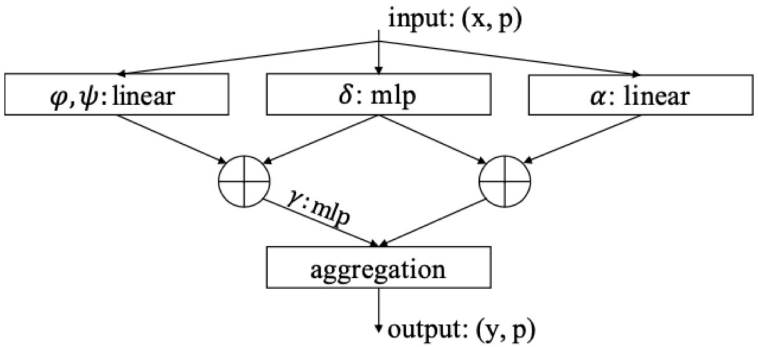
ψ : Keys

α : Values

ρ : softmax

Zhao, et al. "Point transformer." CVPR 2021.
Guo et al. "PCT: Point cloud transformer," CVM, 2021.

PointTransformer[s]



Overall very similar aggregation to DGCNN but “closer” to the Transformer attention mechanism.

Zhao, et al. "Point transformer." CVPR 2021.
Guo et al. "PCT: Point cloud transformer," CVM, 2021.

Uses a slight variant (vector attention):

$$\mathbf{y}_i = \sum_{\mathbf{x}_j \in \mathcal{X}(i)} \rho\big(\gamma(\varphi(\mathbf{x}_i) - \psi(\mathbf{x}_j) + \delta)\big) \odot \big(\alpha(\mathbf{x}_j) + \delta\big)$$

φ : Queries

ψ : Keys

α : Values

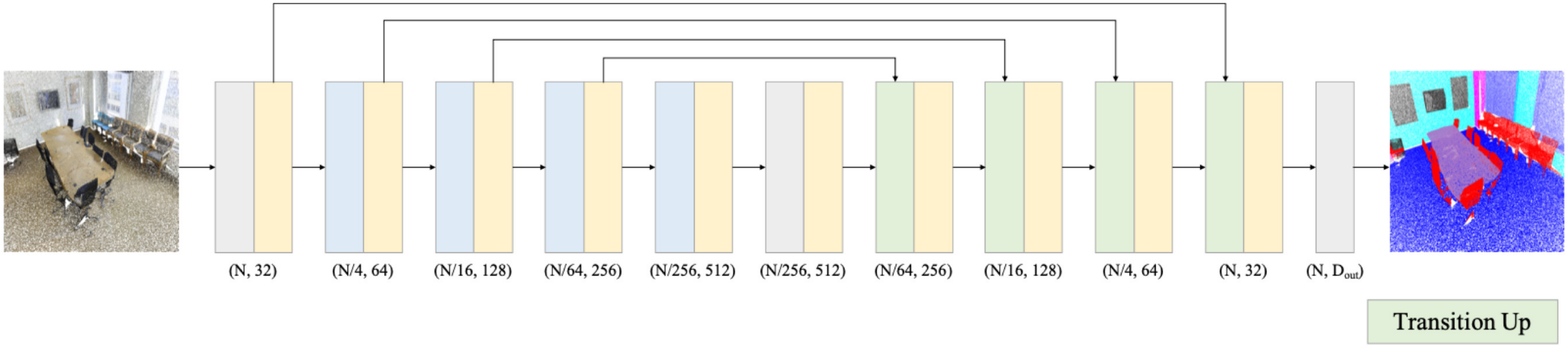
ρ : Softmax

δ : Positional encoding

γ : MLP for aggregation

$\odot$: Hadamard (pointwise) product

PointTransformer[s]



Downsampling and Upsampling for local prediction tasks

Zhao, et al. "Point transformer." CVPR 2021.

PointTransformer[s]

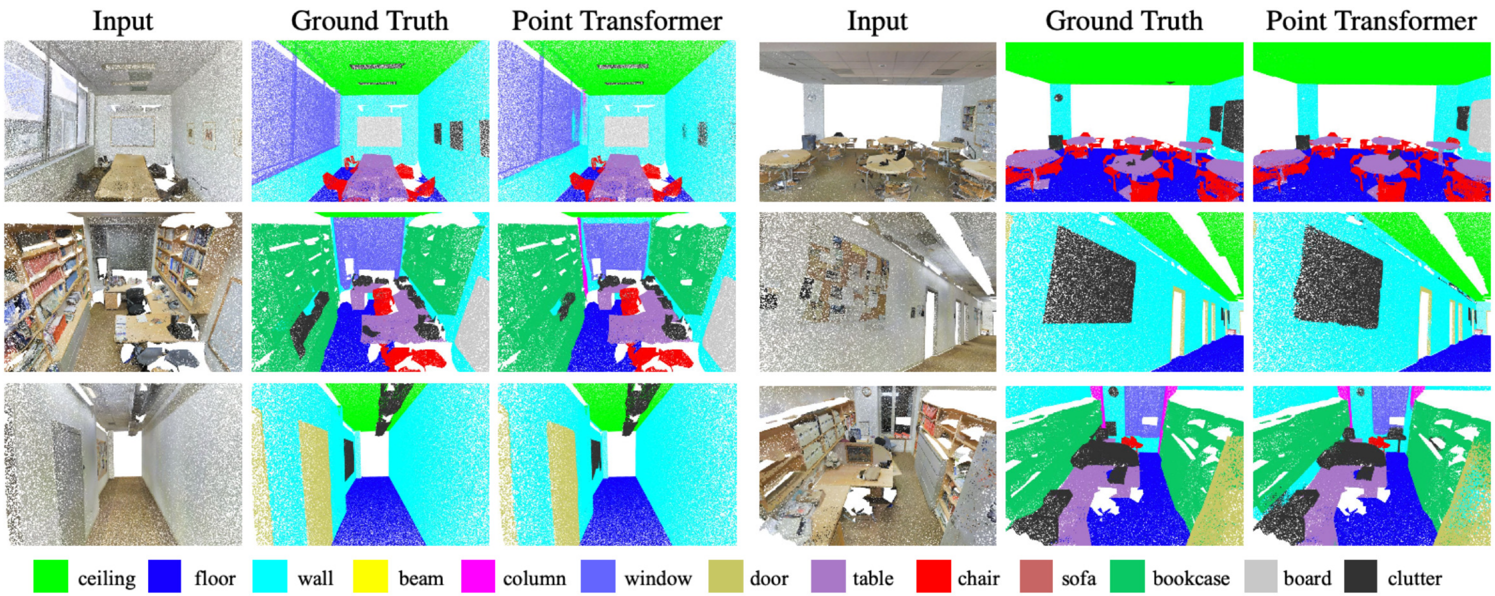


Figure 5. Visualization of semantic segmentation results on the S3DIS dataset.

Zhao, et al. "Point transformer." CVPR 2021.

Many More Point Transformers!

Architectures or Major Training Strategies

1. "Point Transformer" - Zhao et al., *CVPR*, 2021.
2. "PCT: Point cloud transformer" - Guo et al., *CVM*, 2021.
3. "Point-bert: Pre-training 3D point cloud transformers with masked point modeling" - Yu et al., *CVPR*, 2022.
4. "Masked autoencoders for point cloud self-supervised learning" - Pang et al., *ECCV*, 2022.
5. "Point-m2ae: Multi-scale masked autoencoders for hierarchical point cloud pre-training" - Zhang et al., *NeurIPS*, 2022.
6. "Point Transformer v2: Grouped vector attention and partition-based pooling" - Wu et al., *NeurIPS*, 2022.
7. "Point Transformer V3: Simpler Faster Stronger" - Wu et al., *CVPR*, 2024.
8. "Pointnext: Revisiting PointNet++ with improved training and scaling strategies" - Qian et al., *NeurIPS*, 2022.

Multi-modal Models (Often Transformer-based)

1. "Ulip: Learning a unified representation of language, images, and point clouds for 3D understanding" - Xue et al., *CVPR*, 2023.
2. "Ulip-2: Towards scalable multimodal pre-training for 3D understanding" - Xue et al., *CVPR*, 2024.
3. "3D-LLM: Injecting the 3D world into large language models" - Hong et al., *NeurIPS*, 2023.
4. "Openshape: Scaling up 3D shape representation towards open-world understanding" - Liu et al., *NeurIPS*, 2024.
5. "Learning 3D representations from 2D pre-trained models via image-to-point masked autoencoders" - Zhang et al., *CVPR*, 2023.
6. "Contrast with reconstruct: Contrastive 3D representation learning guided by generative pretraining" - Qi et al., *ICML*, 2023.
7. "Pointllm: Empowering large language models to understand point clouds" - Xu et al., *ECCV*, 2024.

Surveys

1. "A survey of visual transformers" - Liu et al., *IEEE TNNLS*, 2023.
2. "Unsupervised point cloud representation learning with deep neural networks: A survey" - Xiao et al., *TPAMI*, 2023.
3. "Mm-llms: Recent advances in multimodal large language models" - Zhang et al., *arXiv*, 2024.
4. "3D vision with transformers: A survey" - Lahoud et al., *arXiv*, 2022.
5. "Transformers in 3D point clouds: A survey" - Lu et al., *arXiv*, 2022.

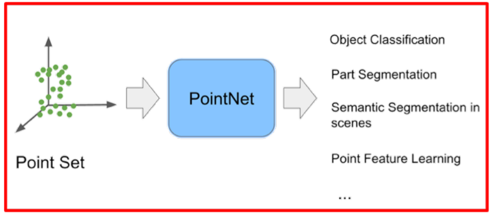
Point Cloud 3D Deep Learning

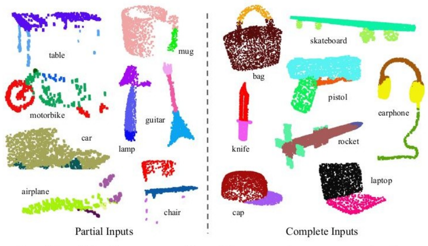
Advantages

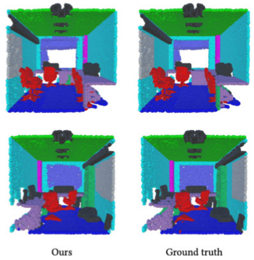
- Extreme versatility (everything is a point cloud).
- Efficiency and robustness

Limitations

- Not very adapted to *non-rigid shape* analysis
- Basic versions are not rotation-invariant
- Not great for *generative* modeling.







Point Cloud 3D Deep Learning

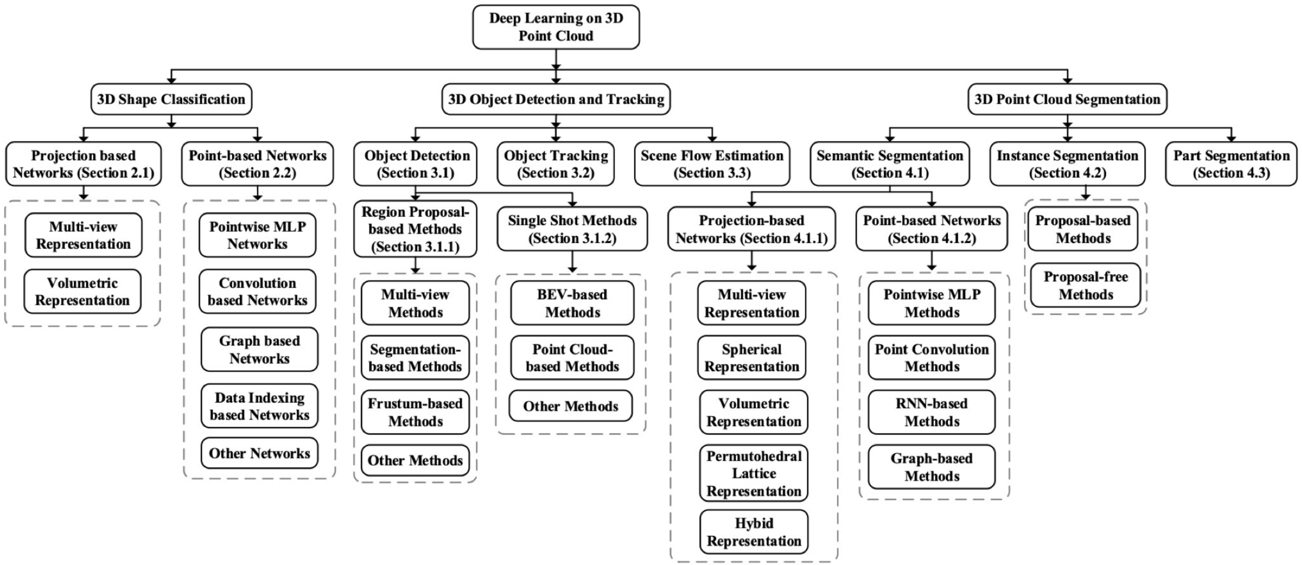


Fig. 1: A taxonomy of deep learning methods for 3D point clouds.

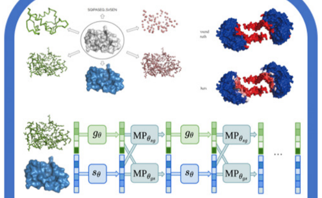
https://github.com/QingyongHu/SoTA-Point-Cloud

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

3D Deep Learning – there is much more work to do!

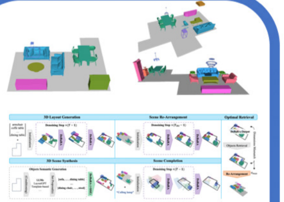
Our group at Ecole Polytechnique

- We focus on 3D shape analysis tasks & everything related!



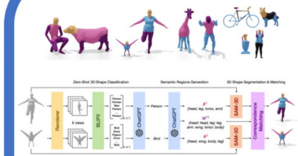
Learning on protein surfaces

Mallet et al. "AtomSurf: Surface Representation for Learning on Protein Structures." arXiv 2309.16519 (2023).



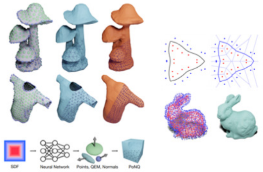
3D scene *generative modeling*

Maillard et al. "DeBaRA: Denoising-Based 3D Room Arrangement Generation." NeurIPS 2024



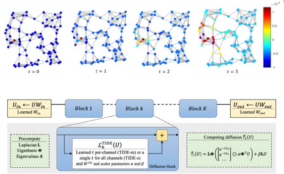
LLMs for 3D shape understanding

Abdelreheem et al. "Zero-shot 3d shape correspondence." SIGGRAPH Asia 2023.



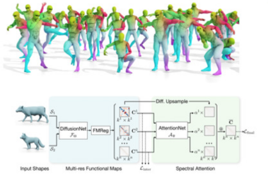
Learnable *representations* for 3D

Maruani et al. "PoNQ: a Neural QEM-based Mesh Representation." CVPR 2024



Architectures learning on graphs

Maruani et al. "PoNQ: a Neural QEM-based Mesh Representation." CVPR 2024



Learning for *non-rigid* 3D data

Attaiki et al. "Shape non-rigid kinematics (SNK)" NeurIPS 2023

3D Deep Learning – there is much more work to do!

Our group at Ecole Polytechnique

- We focus on 3D shape analysis tasks & everything related!

Internships:

- Internships available with PhD funding (priority to M2 students interested in pursuing a PhD)
- Focus on paper publications (great if you already have experience, not a deal breaker if you don't).
- Reach out to me (or Emery!) if you are interested.

Thank You

Questions?

Thank You

Questions?