EECS 230 Deep Learning
Lecture 14: Point Cloud Network

Some materials from Charles Qi and Hengshuang Zhao

Data modality

- Image and video
  - Regular data (2D or 3D grid)
- Text
  - Sequence to sequence model
- Neural network for Point Cloud?
- Neural network for Graph?

Outline

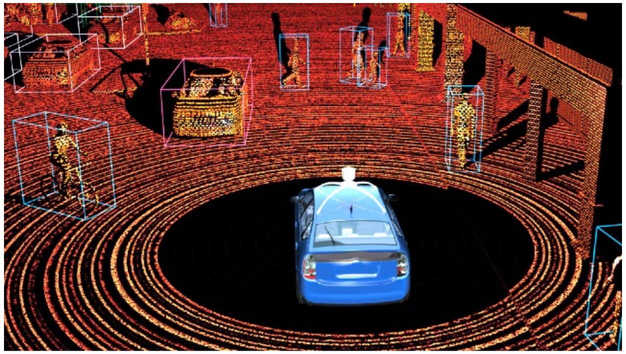
- ❑Neural network for point cloud
  - ❑PointNet
  - ❑PointNet++
  - ❑EdgeConv
  - ❑Point Transformer
- ❑Graph Neural Network
  - ❑Graph convolutional network
  - ❑Graph attention network
- ❑An application to correspondence matching
  - ❑SuperGlue for visual localization

UC MERCED

Neural Network for Point Cloud

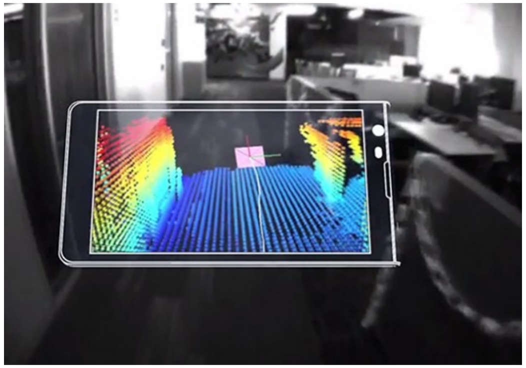
3D Applications

Robot Perception



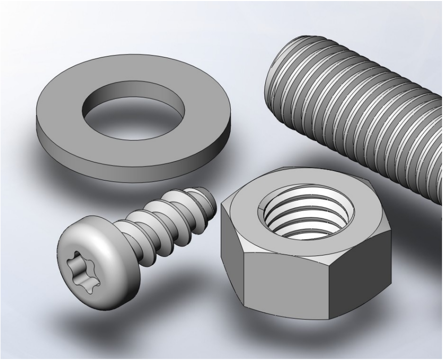
source: Scott J Grunewald

Augmented Reality



source: Google Tango

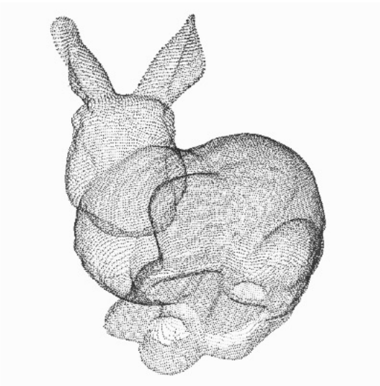
Shape Design

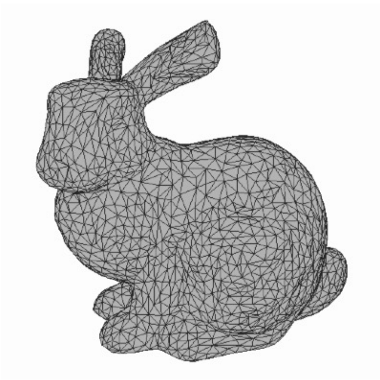


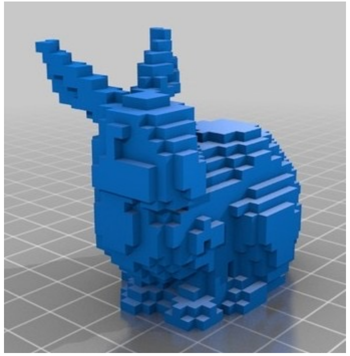
source: solidsolutions

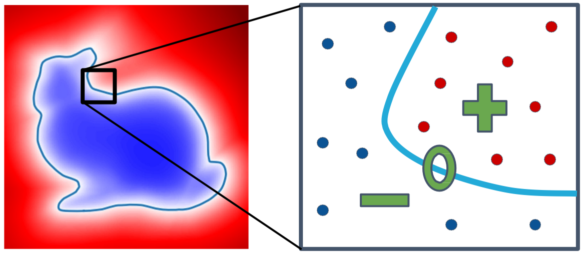
Shape representation

- How to represent a shape in computer?


Point Cloud


Mesh

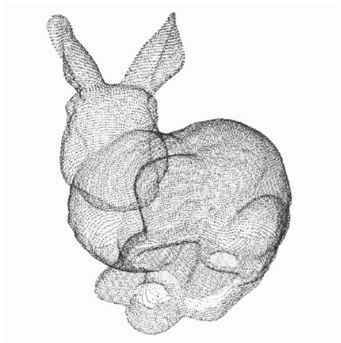

Volumetric


Implicit function

Point Cloud from raw sensor

LiDAR

Depth Sensor

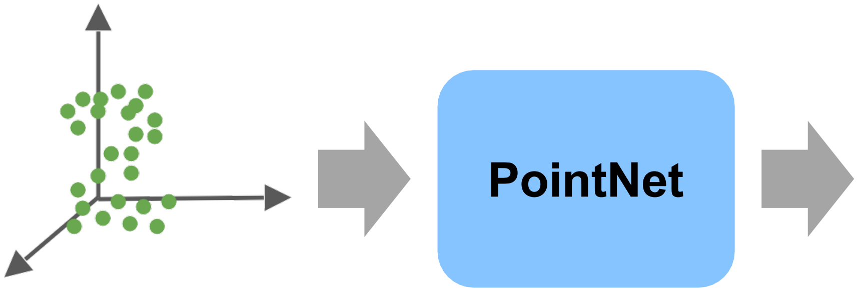


Point Cloud

PointNet

End-to-end learning for **scattered, unordered** point data

Unified framework for various tasks



Object Classification

Object Part Segmentation

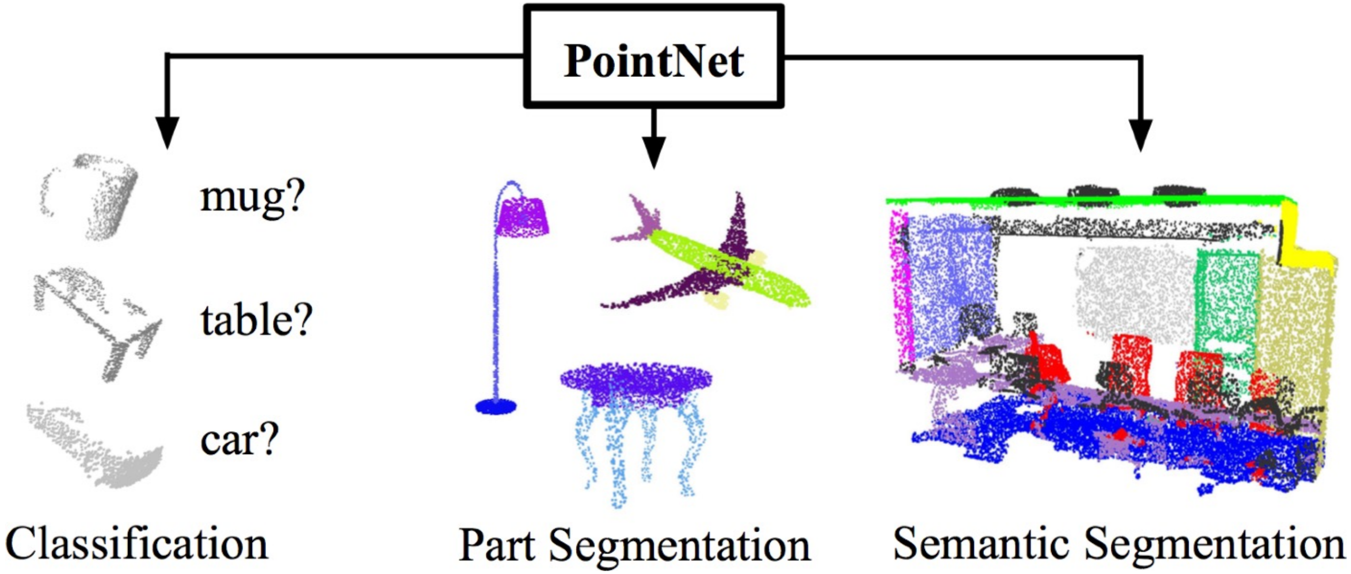
Semantic Scene Parsing

...

PointNet

End-to-end learning for **scattered, unordered** point data

Unified framework for various tasks

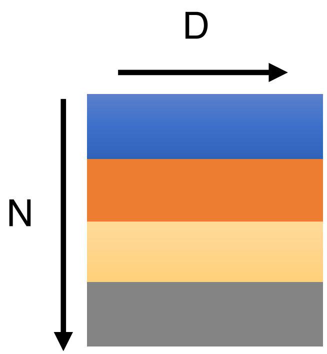


Challenges in Point Cloud Processing

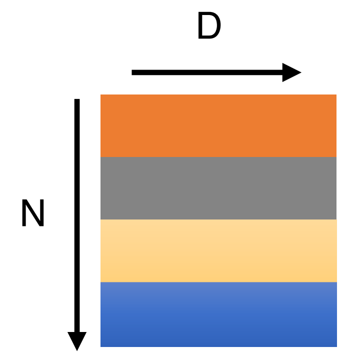
- Unordered point set as input
  - Model needs to be invariant to N! permutations.
- Invariance under geometric transformations
  - Point cloud rotations should not alter classification results.

Unordered Input

Point cloud: N **orderless** points, each represented by a D dim vector



represents the same **set** as



Model needs to be invariant to N! permutations

Symmetric functions (permutation invariant)

Examples:

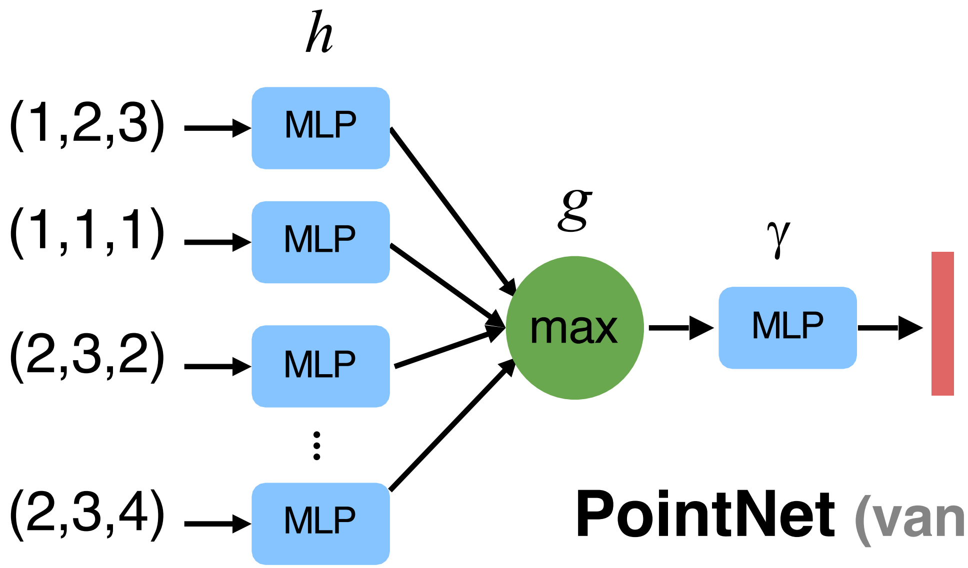
$$f(x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\}$$

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

$\ldots$

Basic PointNet architecture

Empirically, we use **multi-layer perceptron (MLP)** and **max pooling**:



PointNet (vanilla)

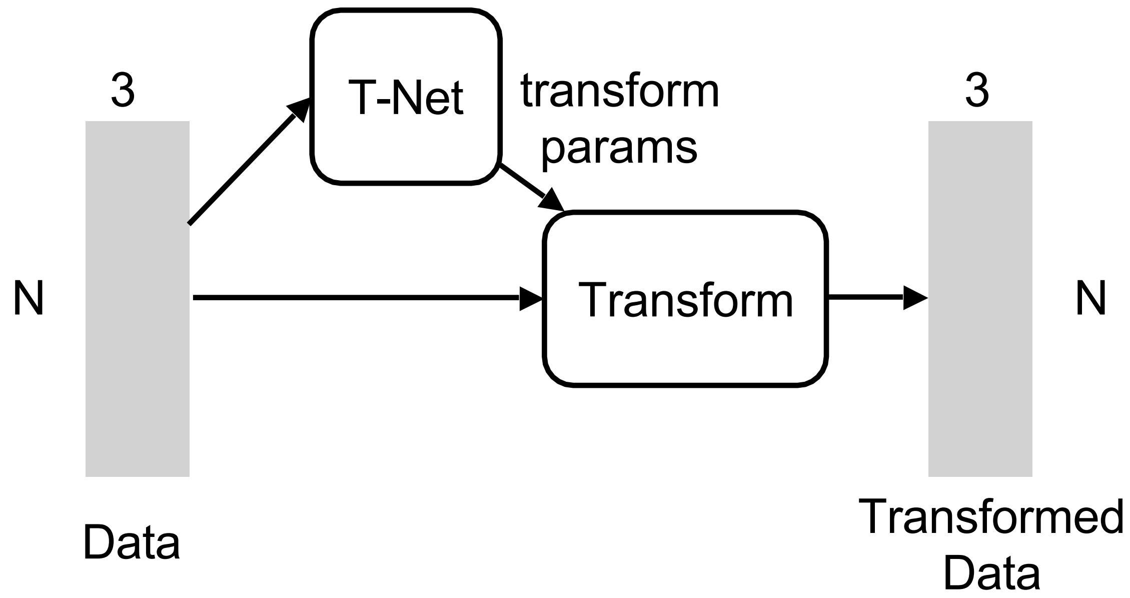
Challenges in Point Cloud Processing

- 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.

UCMERCED

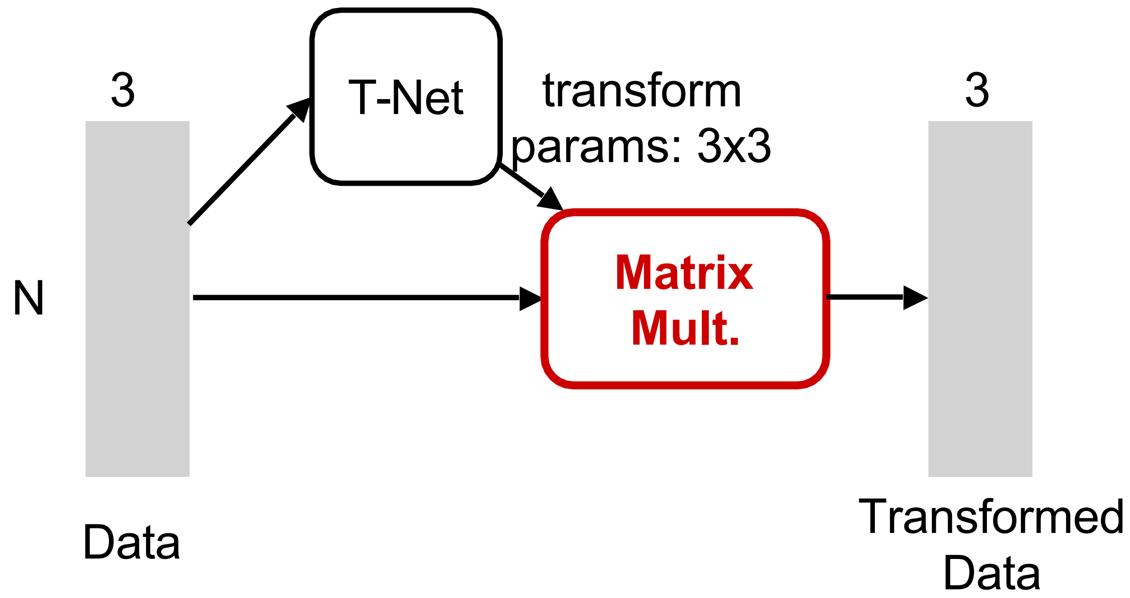
Data Transformation

Idea: Data dependent transformation for automatic alignment

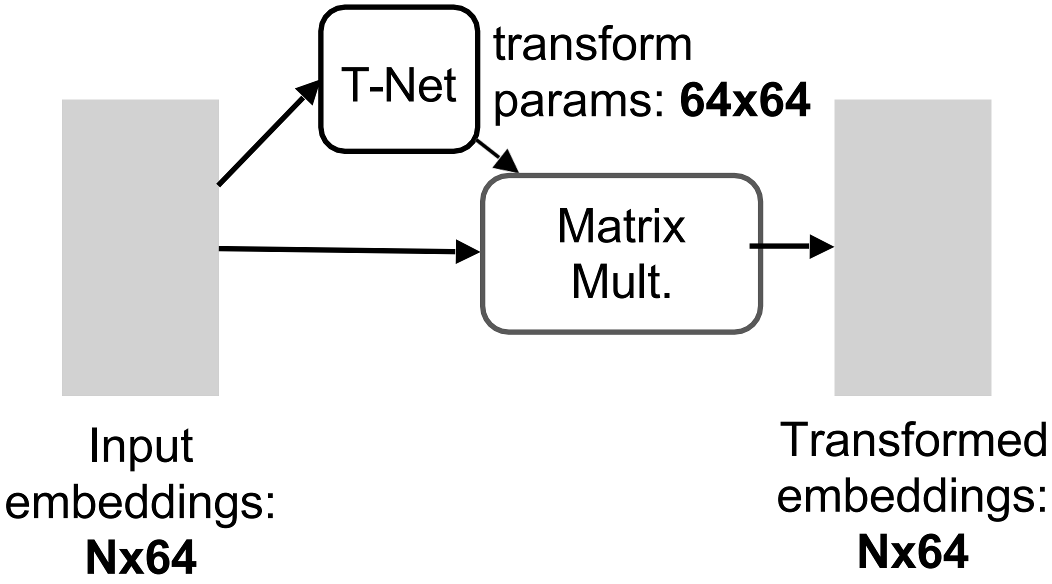


Data Transformation

The transformation is just matrix multiplication!



Embedding space alignment



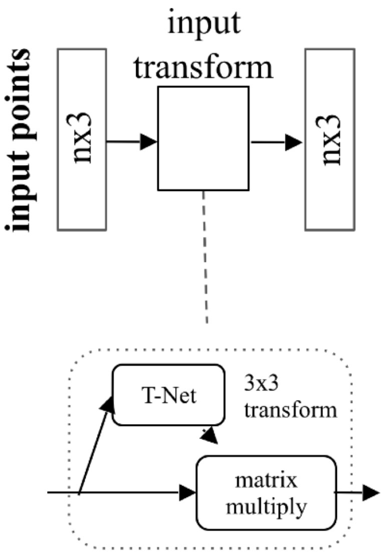
Regularization:

Transform matrix A 64x64 close to orthogonal:

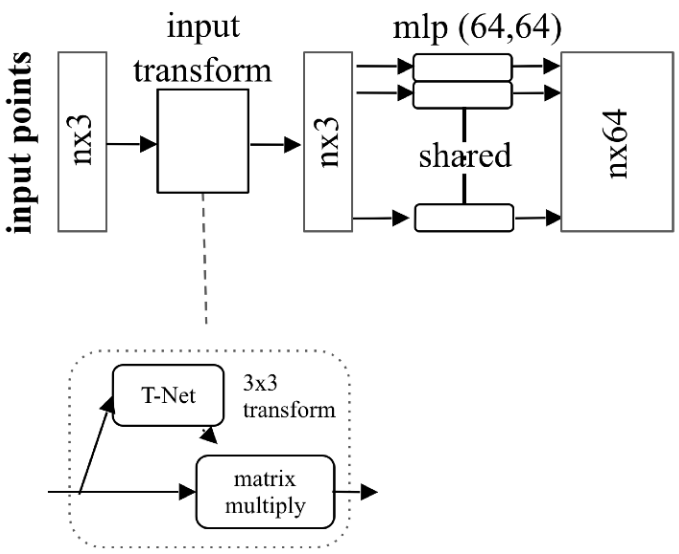
$$L_{reg} = \|I - AA^T\|_F^2$$

Point Classification Network

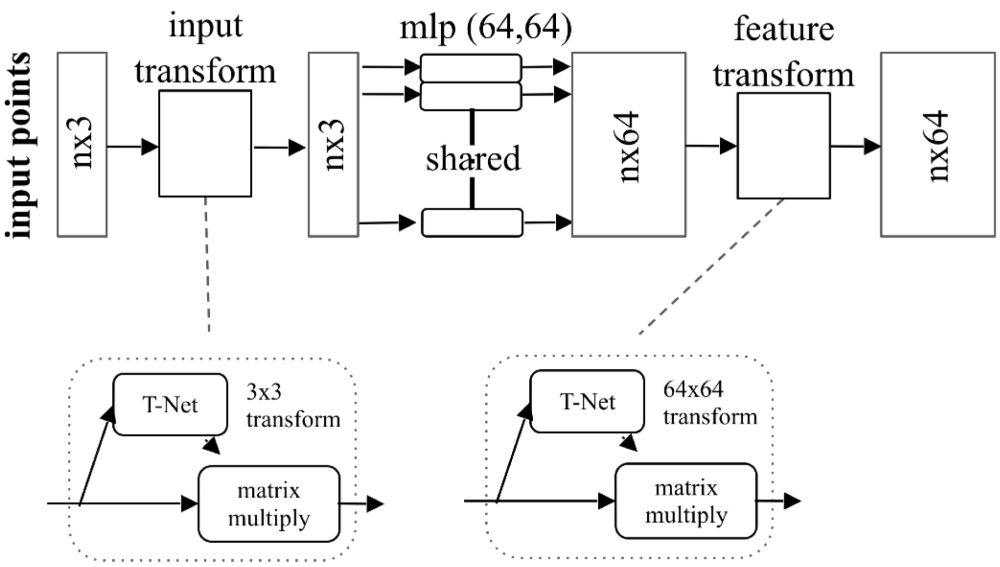
Point Classification Network



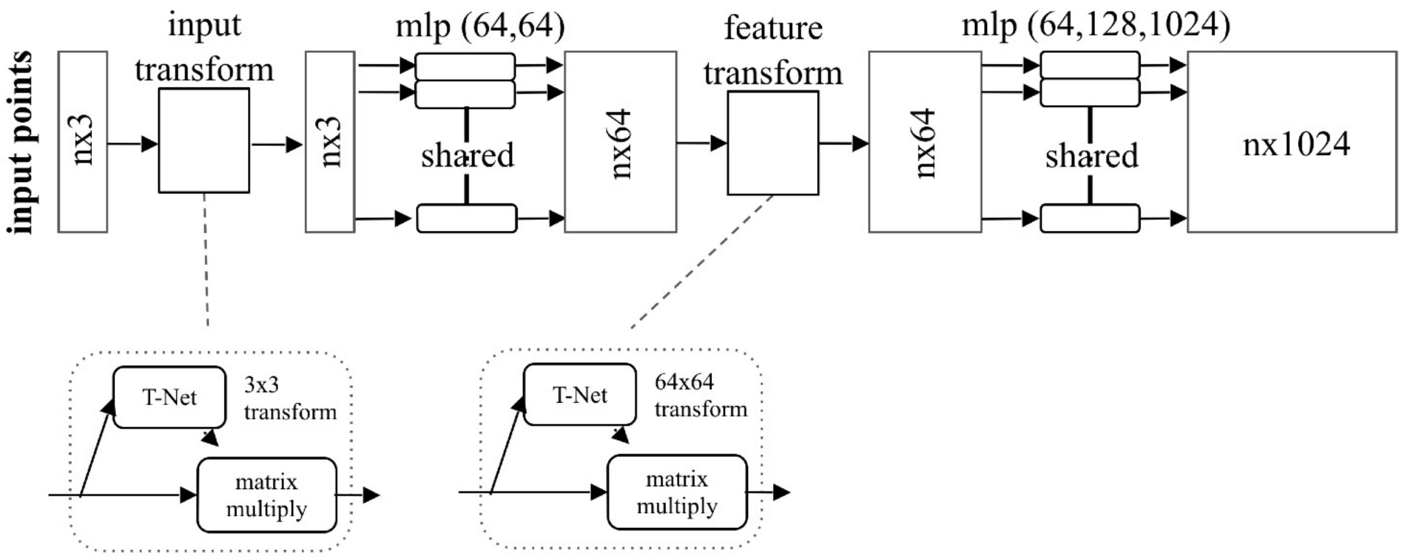
Point Classification Network



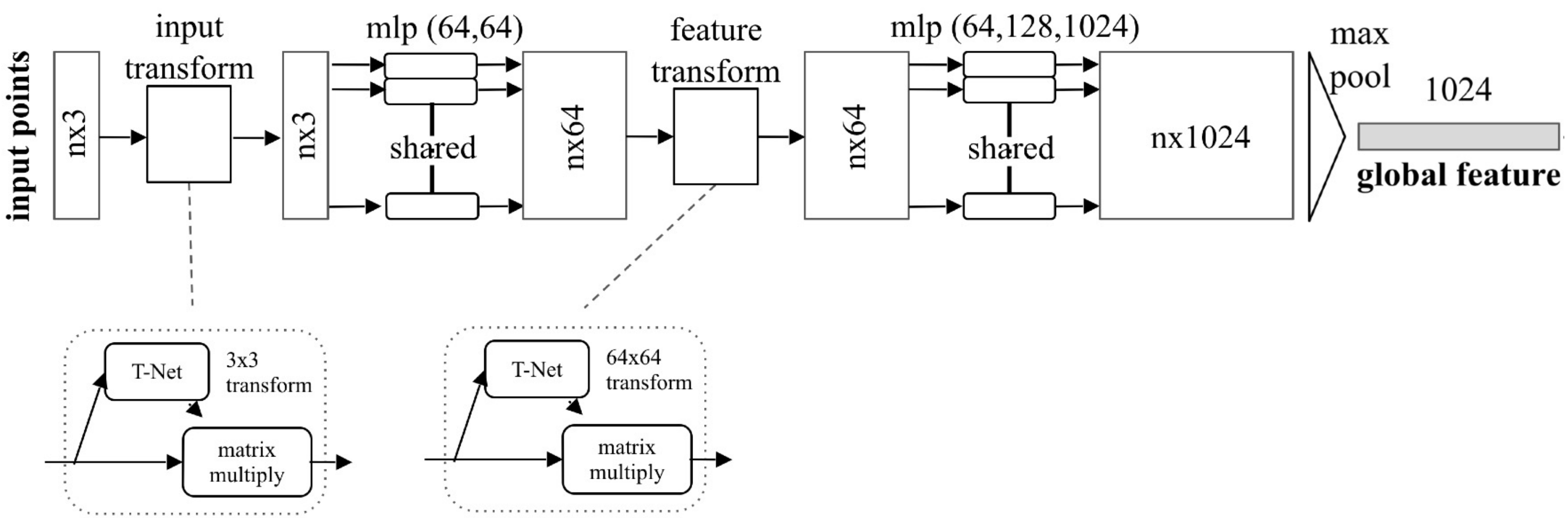
Point Classification Network



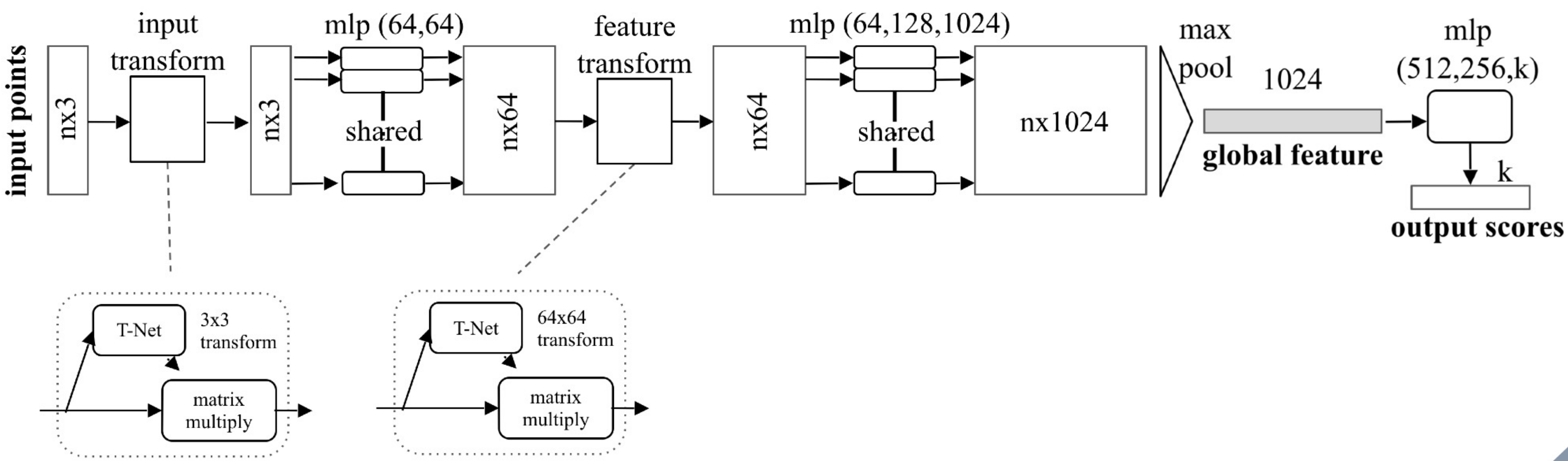
Point Classification Network



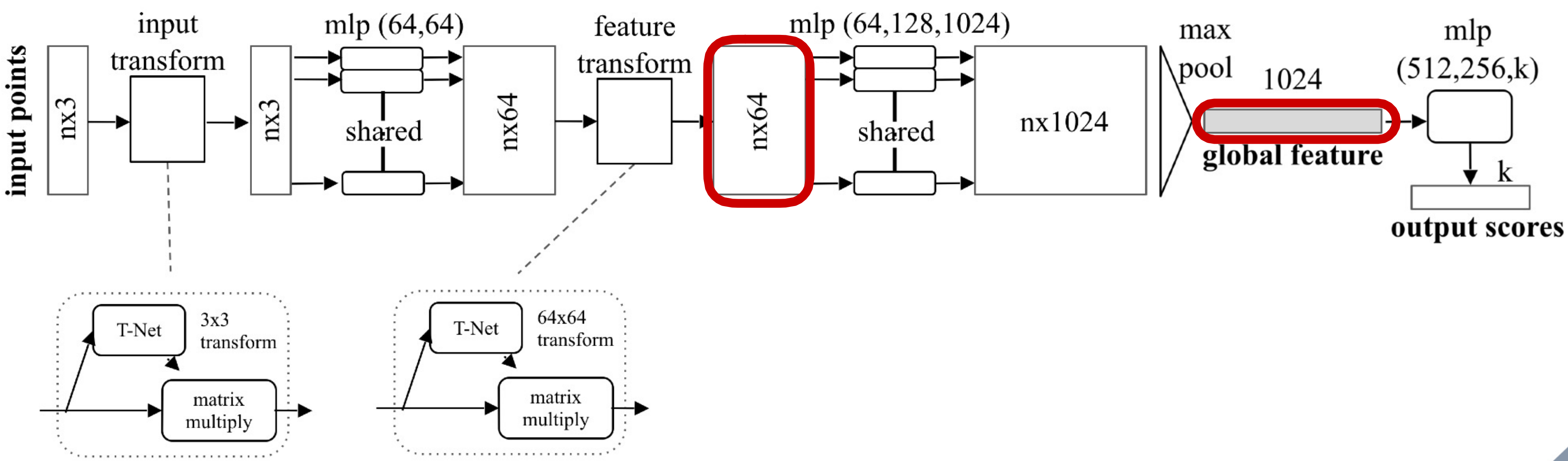
Point Classification Network



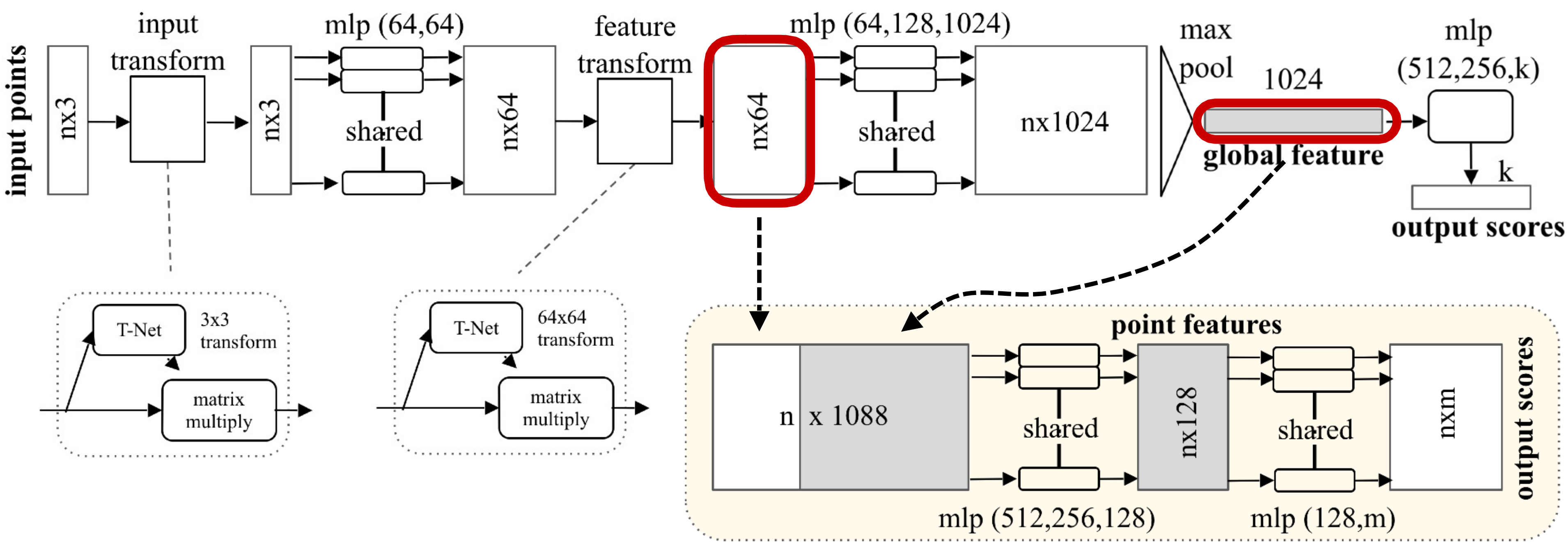
Point Classification Network



Point Classification Network



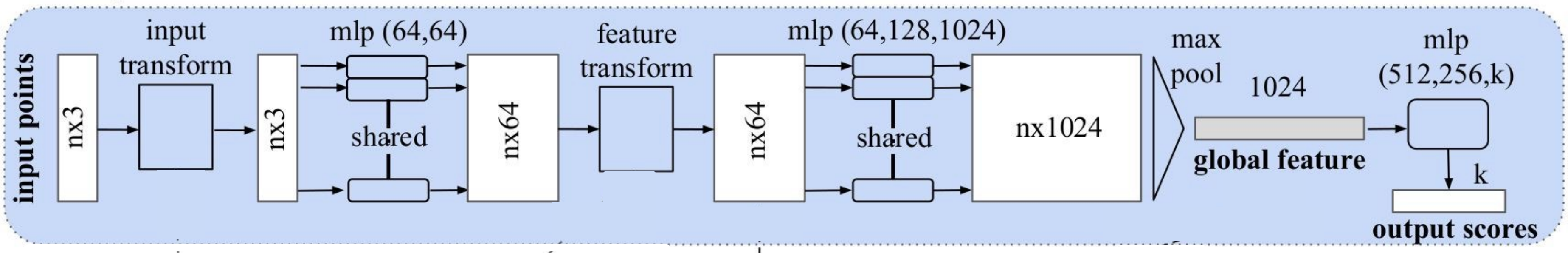
Extend to Point Segmentation Network



Result on point cloud segmentation

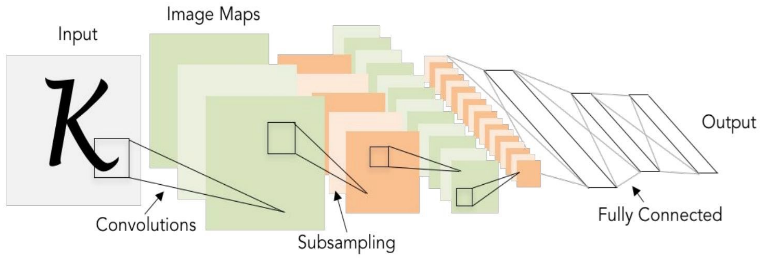
dataset: Stanford 2D-3D-S (Matterport scans)

A Limitation of PointNet



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

Does not take into account the local geometry formed by points



Point Clouds

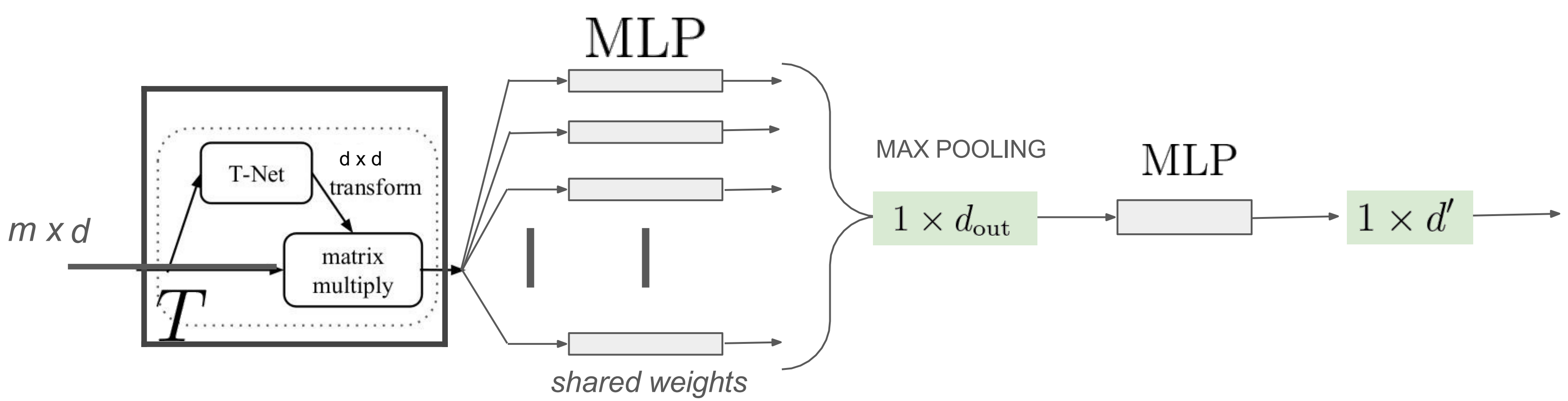
PointNet

PointNet++

PointNet++

Uses PointNet module as a building block

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

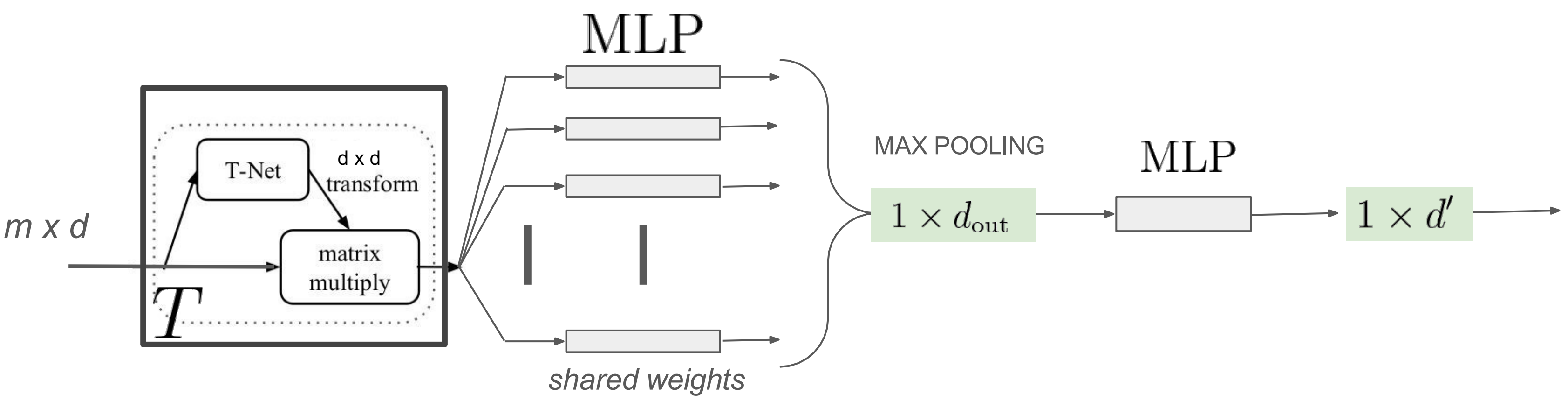


PointNet module

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

PointNet++

Extracts hierarchical features by recursively applying PointNet module



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

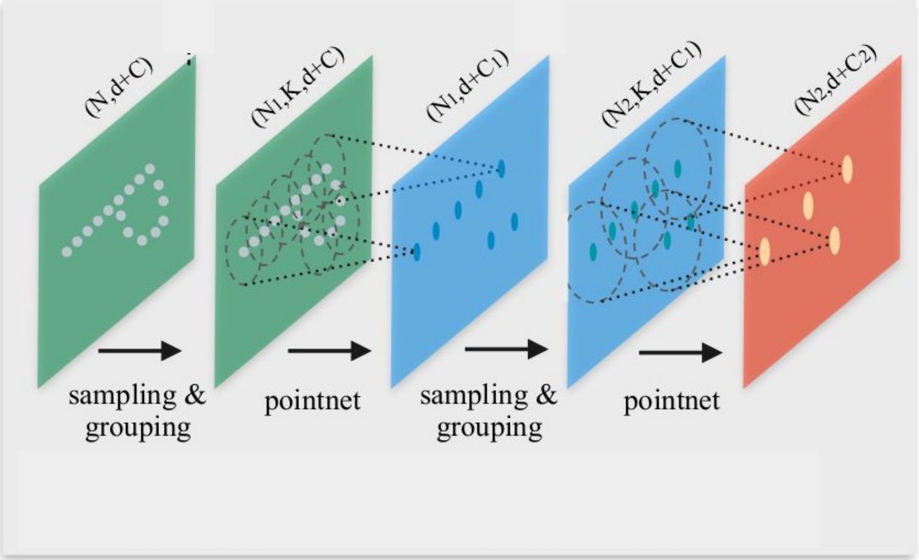
Grouping

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

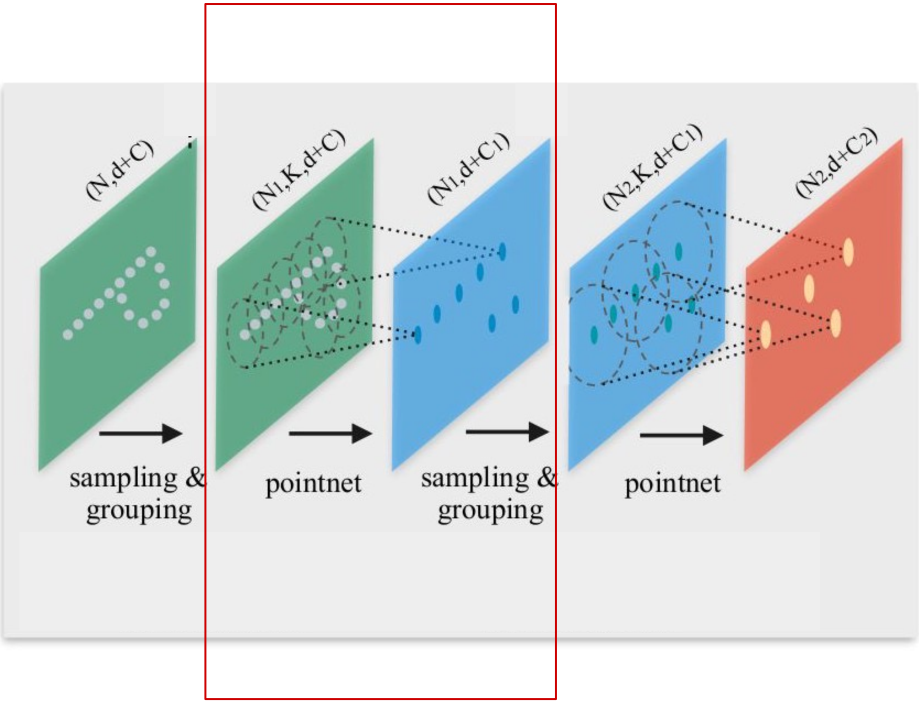
Grouping

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



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

PointNet++

Looks similar to convolution + pooling?

Sampling

Samples *n'* points using farthest point sampling

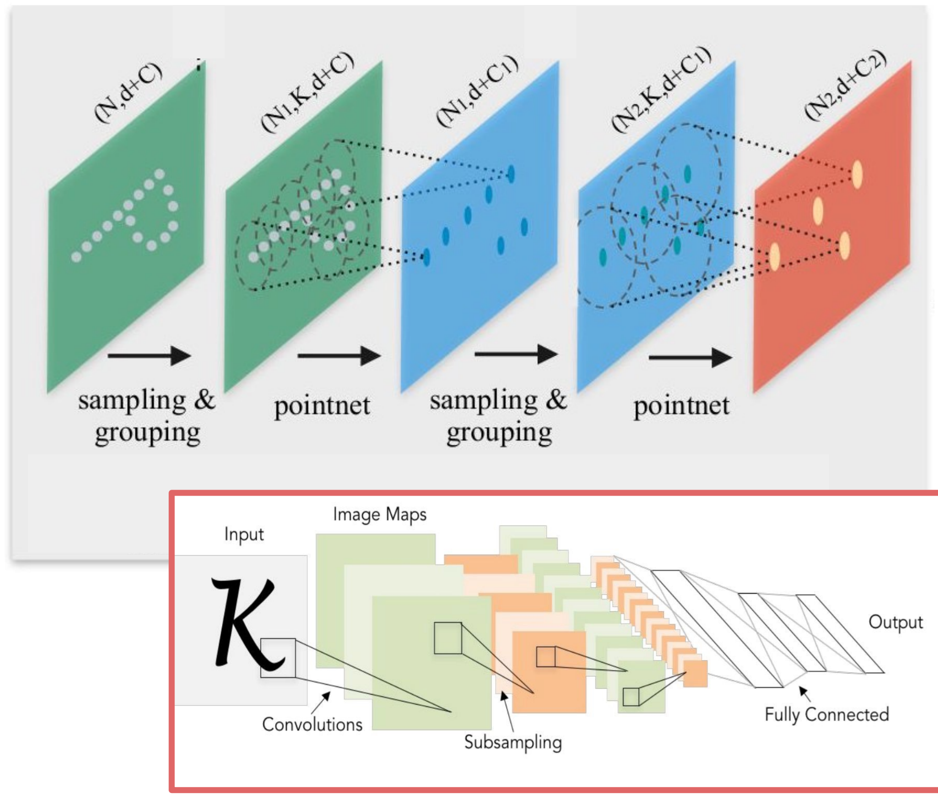
Grouping

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

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

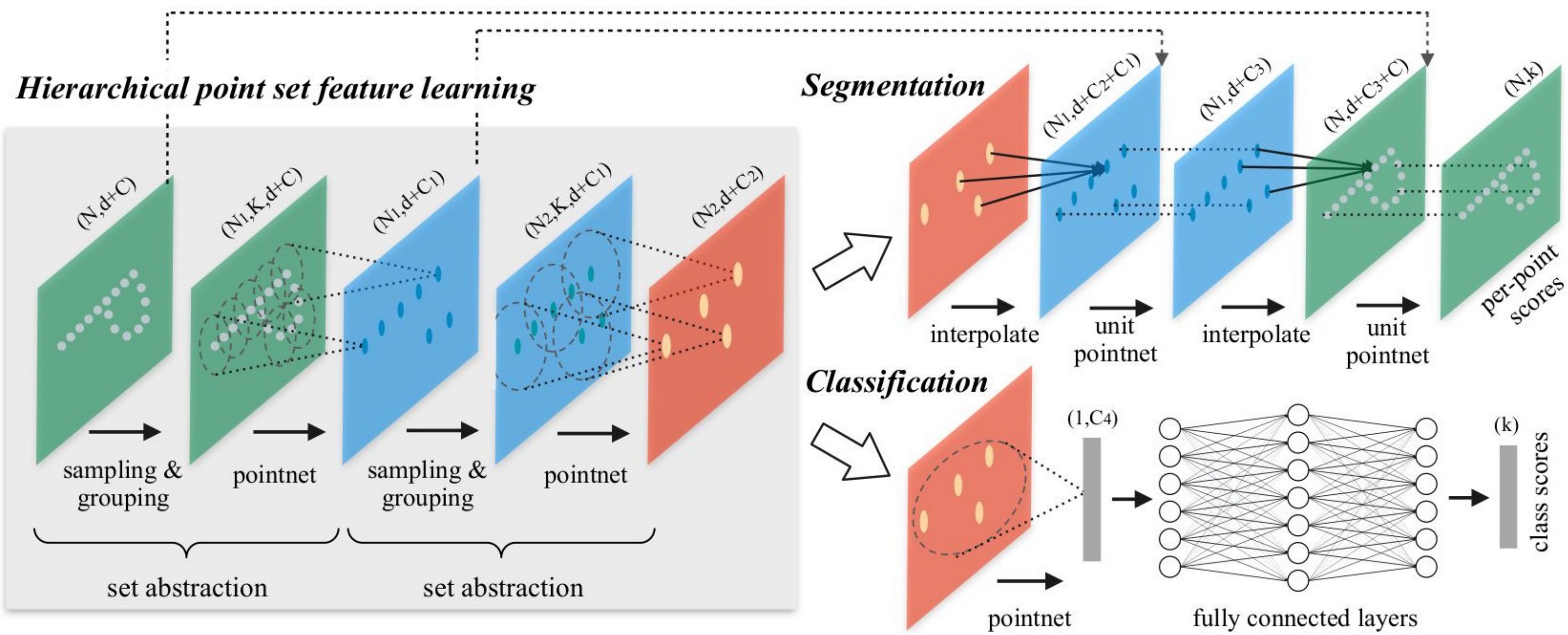
PointNet Layer

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



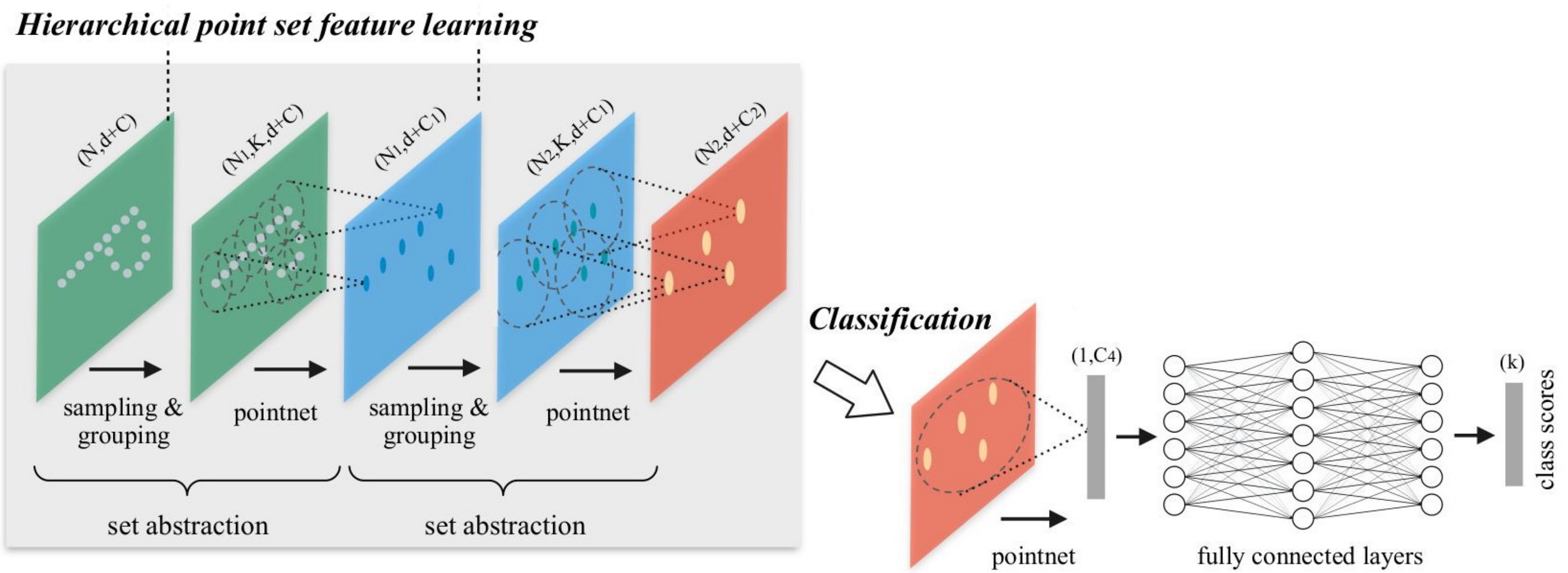
Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space” 2017

PointNet++ for Classification and Segmentation



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

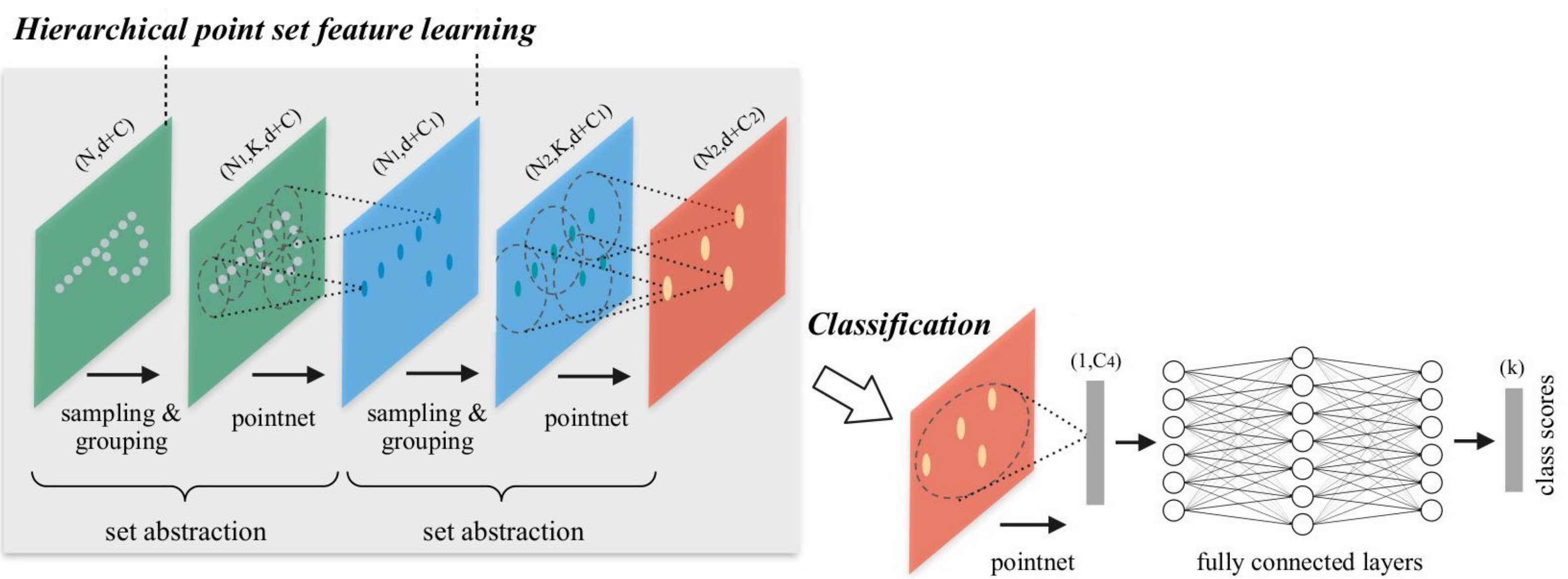
PointNet++ for Classification



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

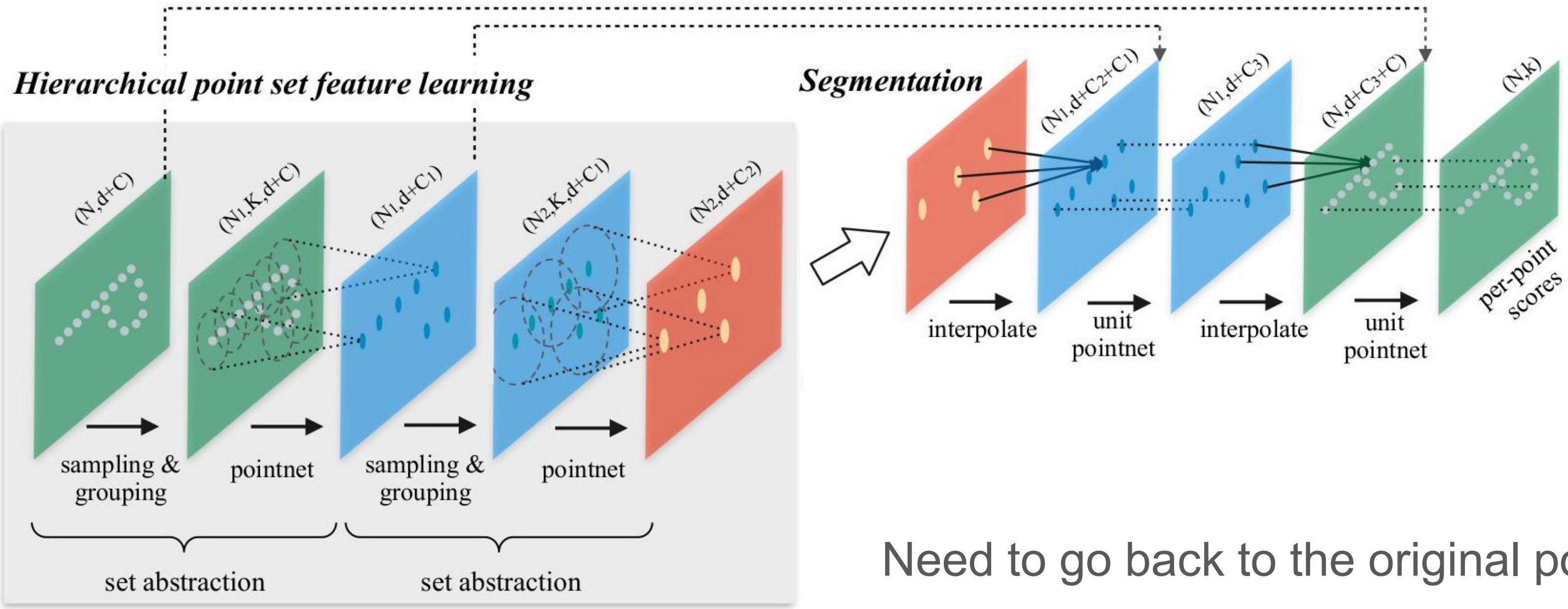
PointNet++ for Classification

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

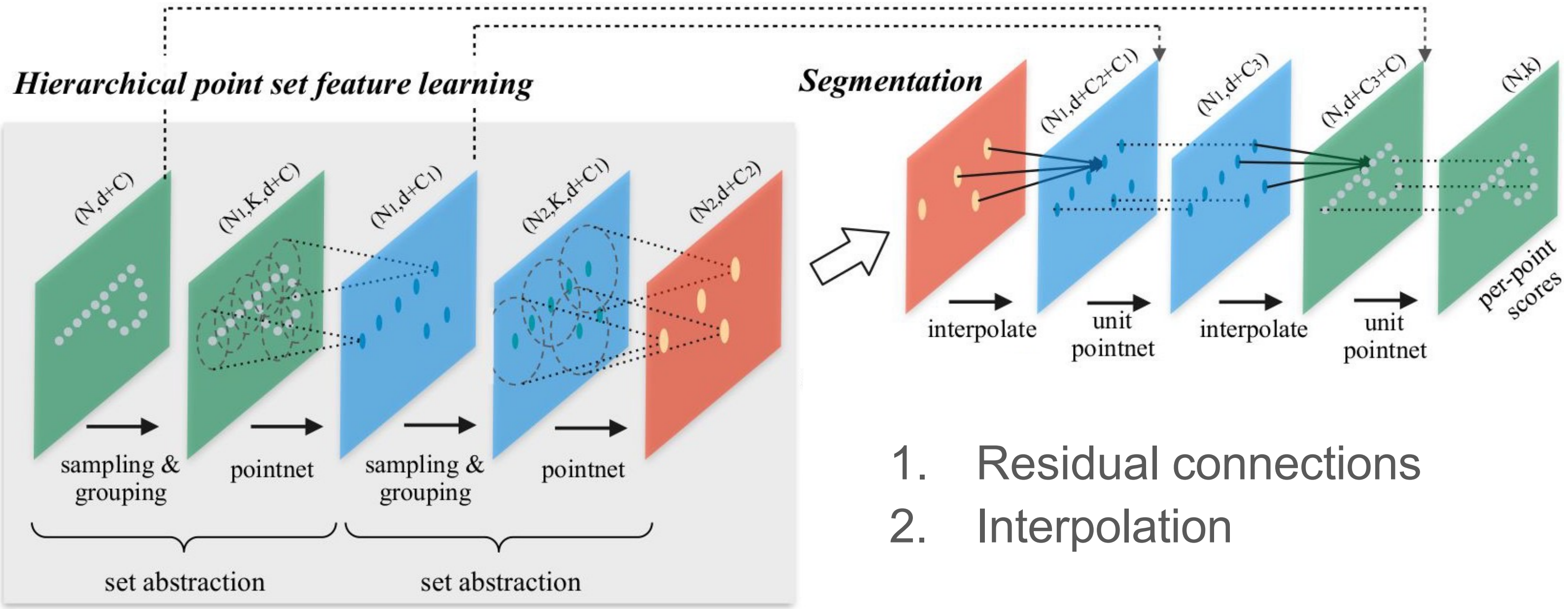
PointNet++ for Segmentation



Need to go back to the original points

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

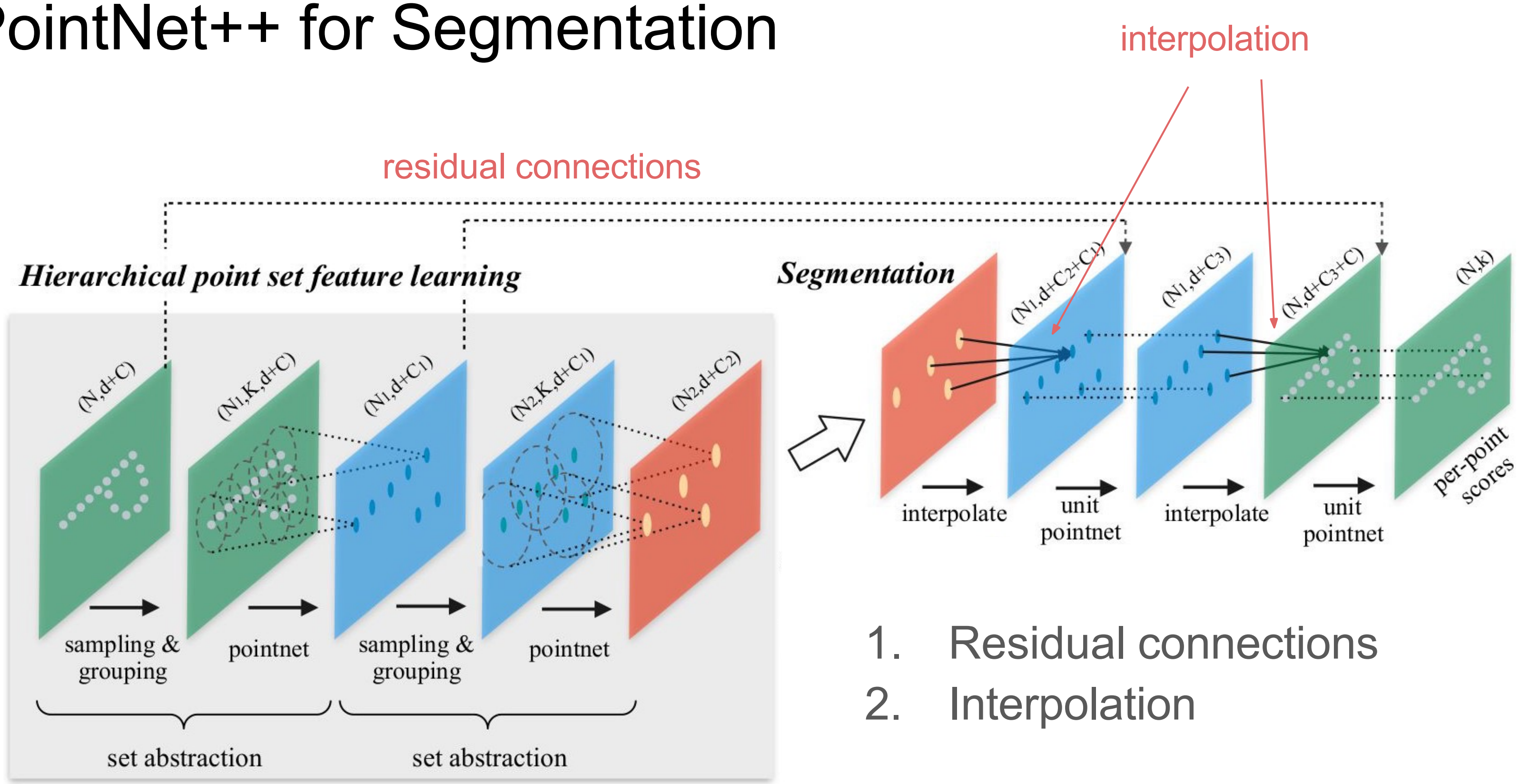
PointNet++ for Segmentation



1. Residual connections
2. Interpolation

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

PointNet++ for Segmentation



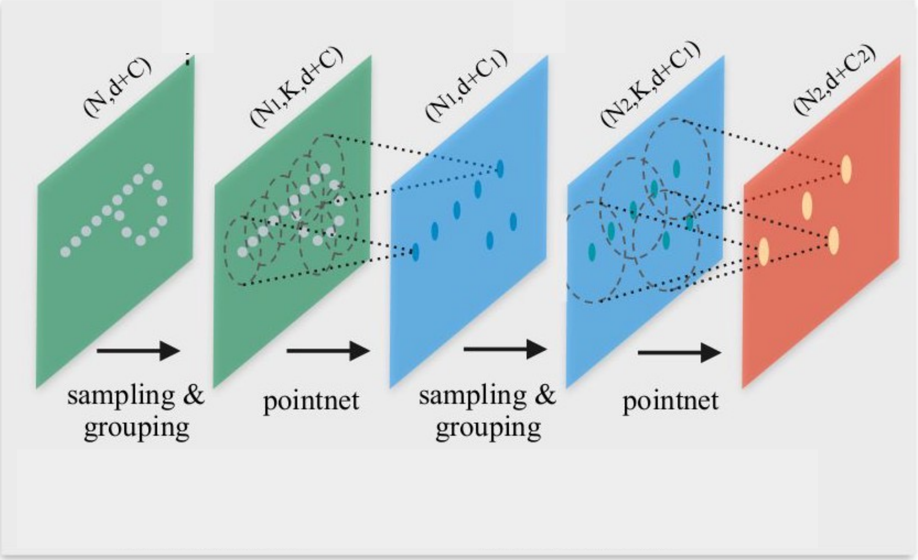
1. Residual connections
2. Interpolation

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

PointNet++

Better Performance than PointNet

Increased Compute Time

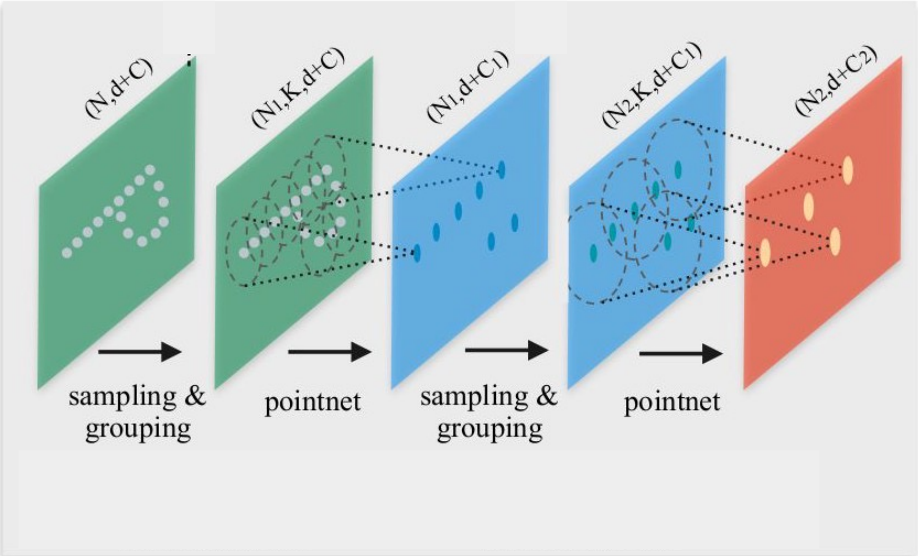


Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space” 2017

Limitations of PointNet++

Does not take into account the local geometry formed by points

Geometry of hierarchical features are pre-determined



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

Point Clouds

PointNet

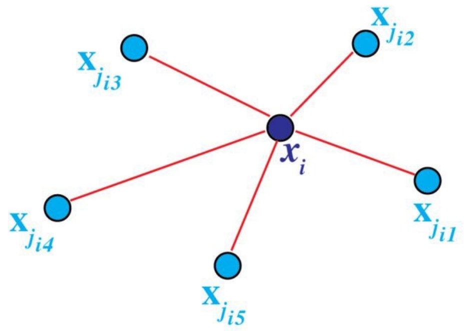
PointNet++

EdgeConv

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

EdgeConv: Basic Idea

Form a local graph by connecting nearby points

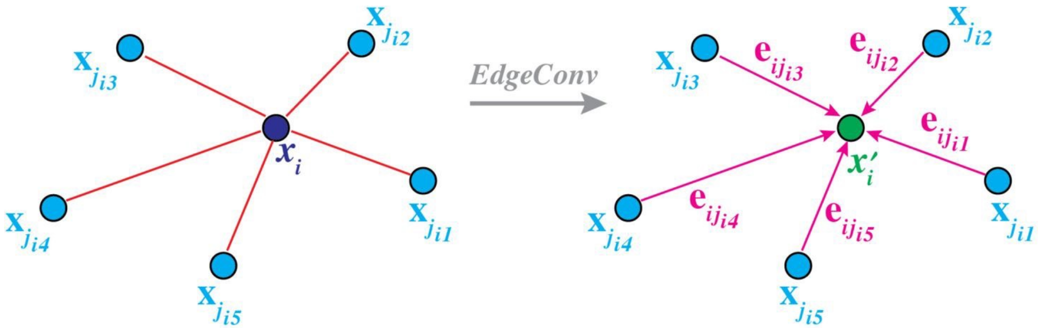


Wang et al. "Dynamic Graph CNN for Learning on Point Clouds" ACM Trans. Graph 2019

EdgeConv: Basic Idea

Form a local graph by connecting nearby points

Apply convolution-like operation on this graph



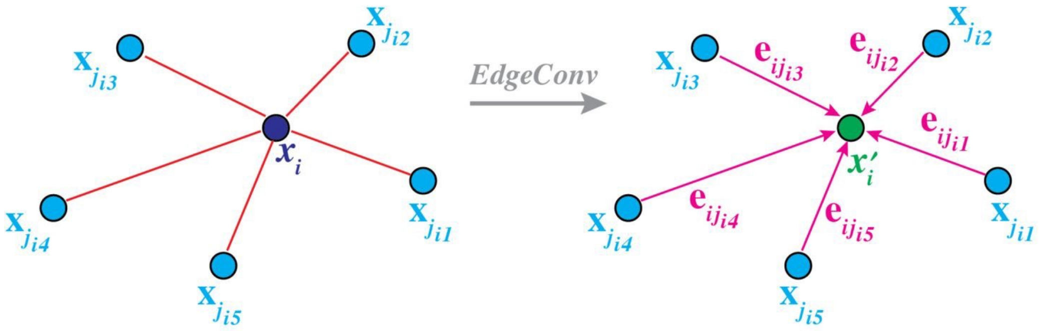
$$x_i' = \square_{j:(i,j)\in E} \; h_\Theta(x_i, x_j)$$

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

EdgeConv: Basic Idea

Form a local graph by connecting nearby points

Apply convolution-like operation on this graph



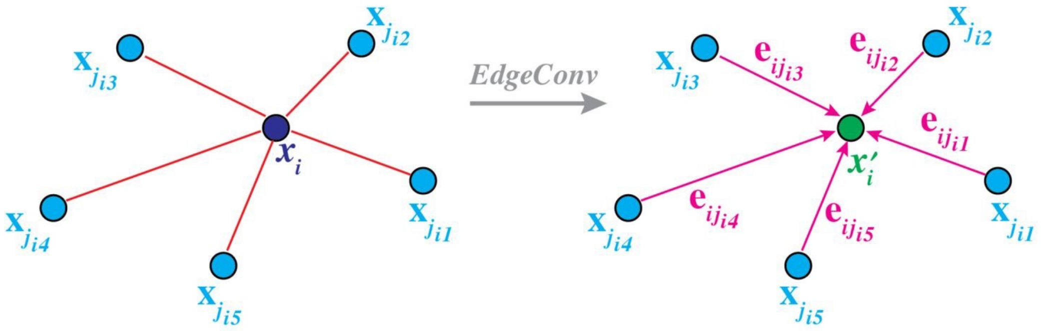
$$x_i' = \square_{j:(i,j)\in E} \; h_\Theta(x_i, x_j)$$

invariant function like max or sum

EdgeConv: Basic Idea

Form a local graph by connecting nearby points

Apply convolution-like operation on this graph



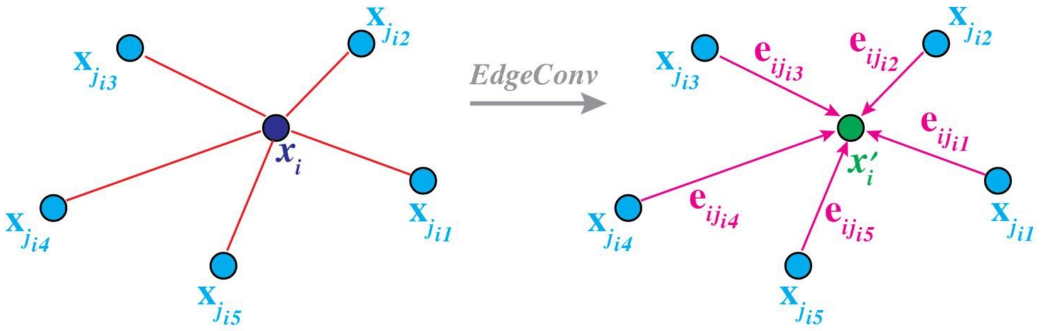
$$x'_i = \square_{j:(i,j)\in E} \; h_\Theta(x_i, x_j)$$

invariant function like max or sum

EdgeConv: Basic Idea

Form a local graph by connecting nearby points

Apply convolution-like operation on this graph



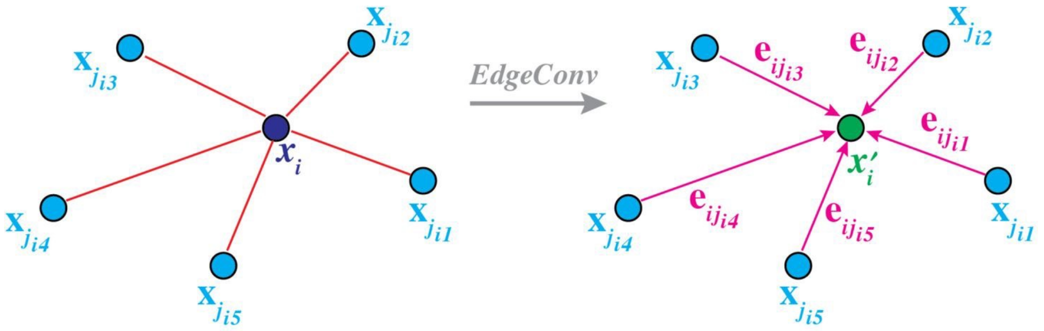
$$x_i' = \square_{j:(i,j)\in E} \; h_\Theta(x_i, x_j)$$

Nearby: with respect to node feature vectors x_i

invariant function like max or sum

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

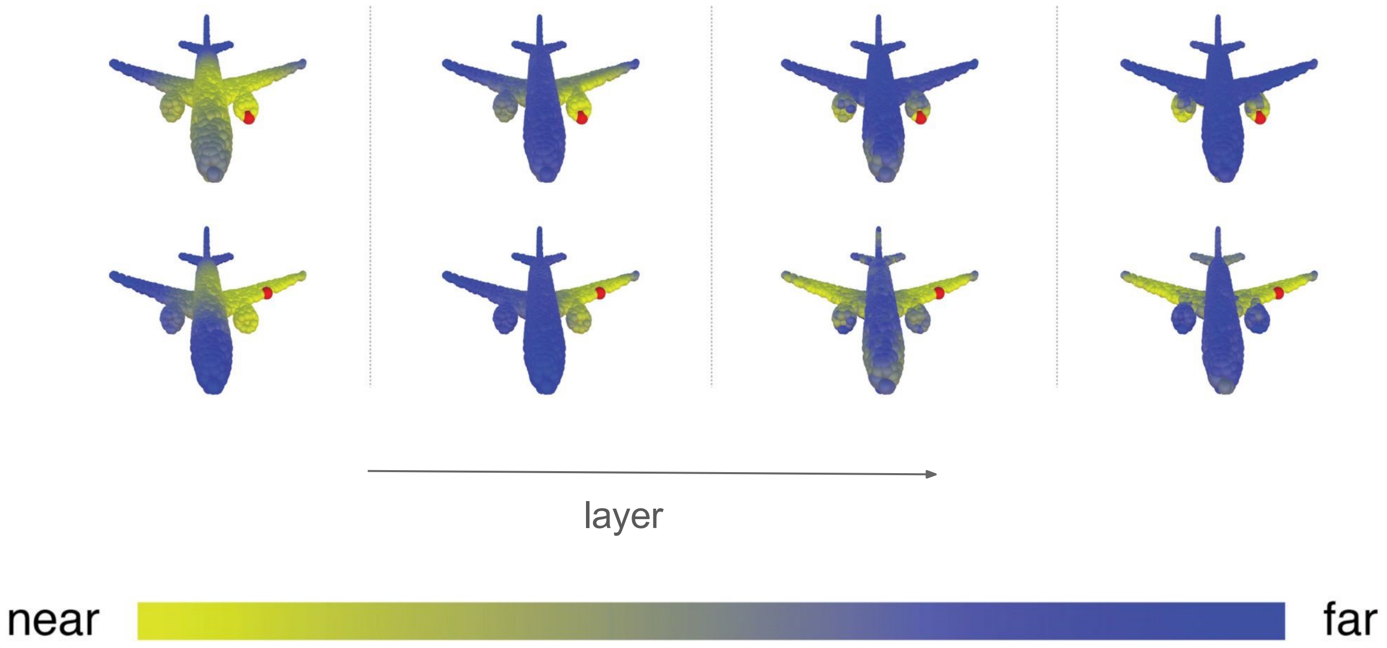
$$x_i \leftarrow x_i' = \square_{j:(i,j)\in E} \; h_\Theta(x_i, x_j)$$

Example

iterate

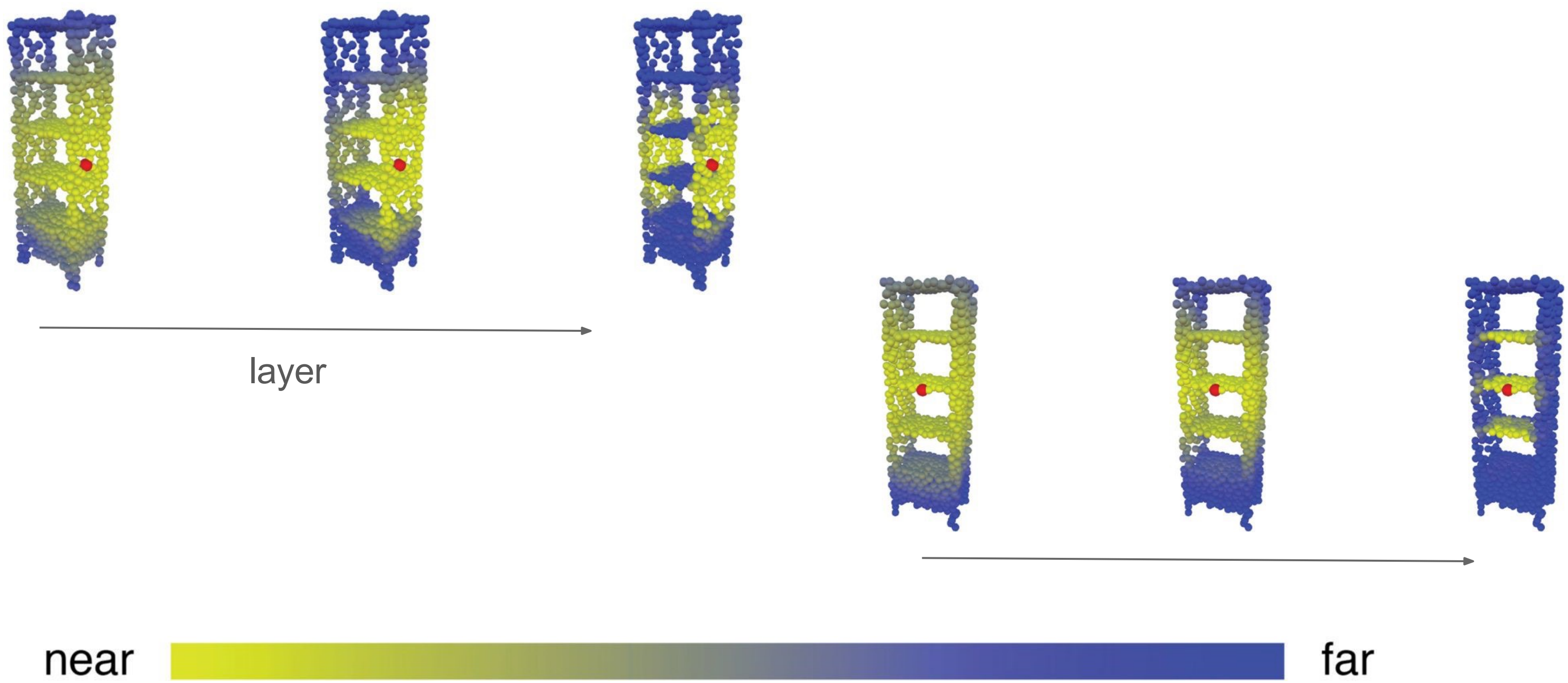
$$h_\Theta(x_i, x_j) = \sigma(\Theta_a \cdot (x_j - x_i) + \Theta_b x_i)$$

Feature Space and Semantically Similar Structures



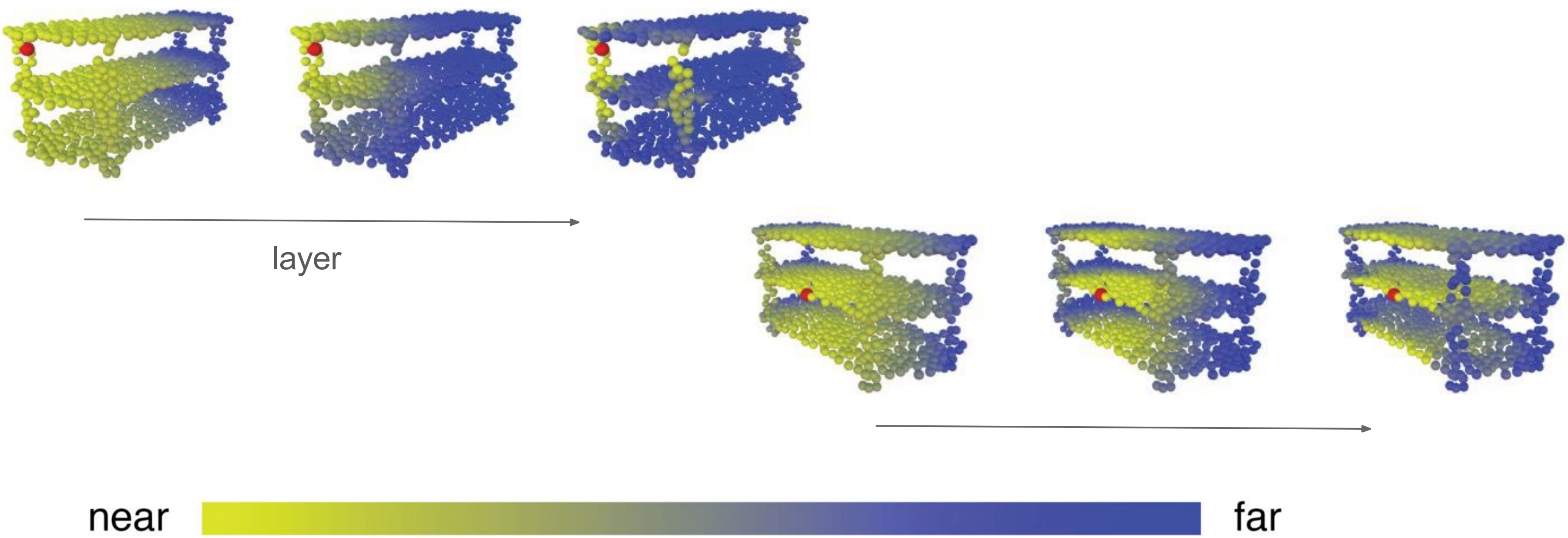
Wang et al. “Dynamic Graph CNN for Learning on Point Clouds” ACM Trans. Graph 2019

Feature Space and Semantically Similar Structures



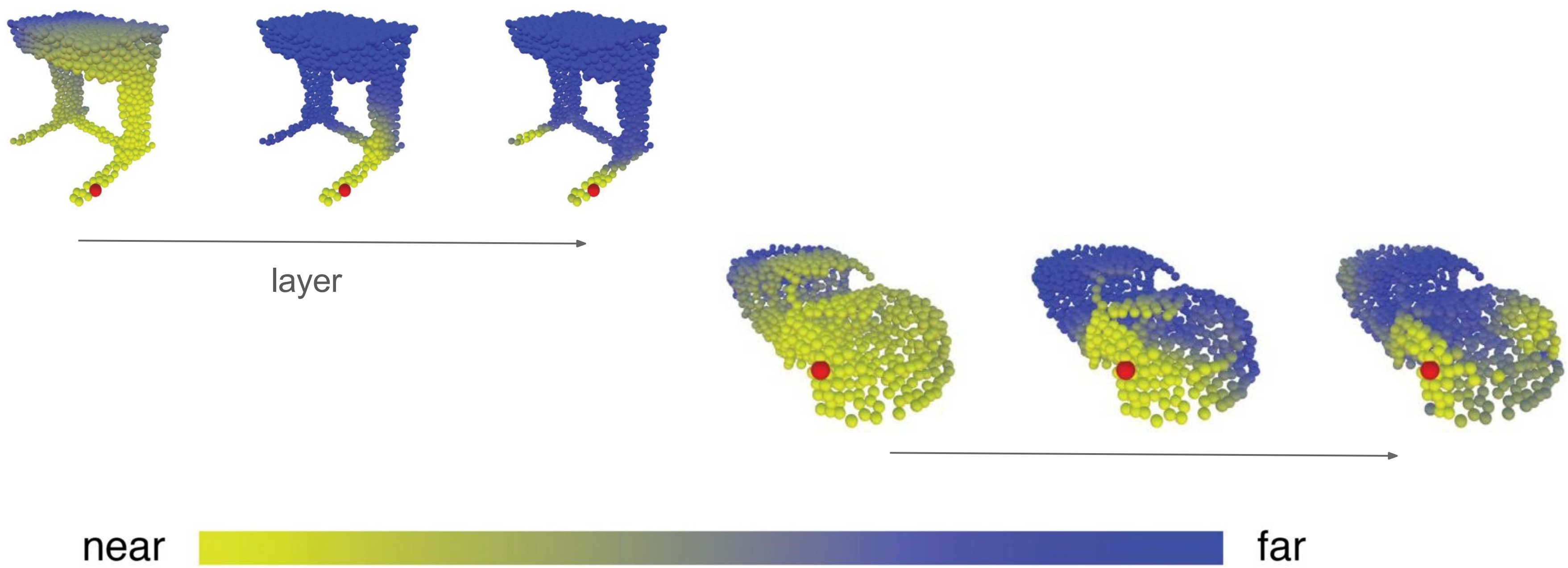
Wang et al. “Dynamic Graph CNN for Learning on Point Clouds” ACM Trans. Graph 2019

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

Point Clouds

PointNet

PointNet++

EdgeConv

Point Transformer

Point Transformers

Based on the idea of attention

Attention based architectures gained popularity in NLP and Computer Vision

Attention Is All You Need 2017

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

urrent or
The best
attention
sformer,

Image Transformer 2017

Niki Parmar *[1] **Ashish Vaswani** *[1] **Jakob Uszkoreit**[1]
Łukasz Kaiser[1] **Noam Shazeer**[1] **Alexander Ku**[2 3] **Dustin Tran**[4]

Abstract

Image generation has been successfully cast as an autoregressive sequence generation or transformation problem. Recent work has shown that self-attention is an effective way of modeling tex-

Attention

Collection of points

Attention

v_1

v_2

v_i

v_j

v_n

Each point has a value

Attention

$v_1 \bullet k_1$

$v_2 \bullet k_2$

$v_i \bullet k_i$

$v_j \bullet k_j$

$v_n \bullet k_n$

Each point has a value and a key

Attention

$v_1 \bullet k_1$

$v_2 \bullet k_2$

$v_i \bullet k_i$

$v_j \bullet k_j$

$v_n \bullet k_n$

Query q

In comes a query q

Attention

Query q

$v_1 \bullet k_1$

$v_2 \bullet k_2$

$v_i \bullet k_i$

$v_j \bullet k_j$

$v_n \bullet k_n$

Output $= v_{i^*}$

$$i^* = \arg\max_i q^T k_i$$

Output value, who's key matches the query

Attention

$v_1 \bullet k_1$

$v_2 \bullet k_2$

$v_i \bullet k_i$

$v_j \bullet k_j$

$v_n \bullet k_n$

Query q

$$\text{Output} = \sum_i (q^T k_i) \cdot v_i$$

Or more like a weighted average

Attention to Point Cloud

Query q

$v_1 \bullet k_1$

$v_2 \bullet k_2$

$v_i \bullet k_i$

$v_j \bullet k_j$

$v_n \bullet k_n$

$$\text{Output} = \sum_i (q^T k_i) \cdot v_i$$

How to develop this idea for an architecture over point clouds?

Attention to Point Cloud

Query q

$\cancel{v_1} \bullet \cancel{k_1}$ x_1

$\cancel{v_2} \bullet \cancel{k_2}$ x_2

$\cancel{v_i} \bullet \cancel{k_i}$ x_i

$\cancel{v_j} \bullet \cancel{k_j}$ x_j

$\cancel{v_n} \bullet \cancel{k_n}$ x_n

$$\text{Output} = \sum_i (q^T k_i) \cdot v_i$$

We don't have values and keys.

We have position, input features.

Attention to Point Cloud

Query $\cancel{q}$ x_j

$\cancel{v_1} \bullet \cancel{k_1}$ x_1

$\cancel{v_2} \bullet \cancel{k_2}$ x_2

$\cancel{v_i} \bullet \cancel{k_i}$ x_i

$\cancel{v_j} \bullet \cancel{k_j}$ x_j

$\cancel{v_n} \bullet \cancel{k_n}$ x_n

Output $= \sum_i (q^T k_i) \cdot v_i$

Query is a point on the point cloud

Attention to Point Cloud

$v_1 \bullet k_1$

$v_2 \bullet k_2$

$v_i \bullet k_i$

$v_j \bullet k_j$

$v_n \bullet k_n$

Query q $q = \phi(x_j)$

$$\text{Output} = \sum_i (q^T k_i) \cdot v_i$$

$$v_i = \alpha(x_i)$$
$$k_i = \psi(x_i)$$

Use trainable functions (MLP) to obtain key, value, and query from features vectors x_i

Attention to Point Cloud

Query q $\qquad q = \phi(x_j)$

$v_1 \bullet k_1$

$v_2 \bullet k_2$

$v_i \bullet k_i$

$v_j \bullet k_j$

$v_n \bullet k_n$

$$x'_j = \sum_i \rho(\phi(x_j)^T \psi(x_i)) \cdot \alpha(x_i)$$

$$v_i = \alpha(x_i)$$

$$k_i = \psi(x_i)$$

Generates update for point j

Zhao et al. “Point Transformer” 2020

Point Transformer

Basic version

$$x'_j = \sum_{i \in N(x_j)} \rho(\phi(x_j)^T \psi(x_i)) \cdot \alpha(x_i)$$

Zhao et al. “Point Transformer” 2020

Point Transformer

Basic version

$$x'_j = \sum_{i \in N(x_j)} \rho(\phi(x_j)^T \psi(x_i)) \cdot \alpha(x_i)$$

Incorporating point feature + location; and using vector for attention

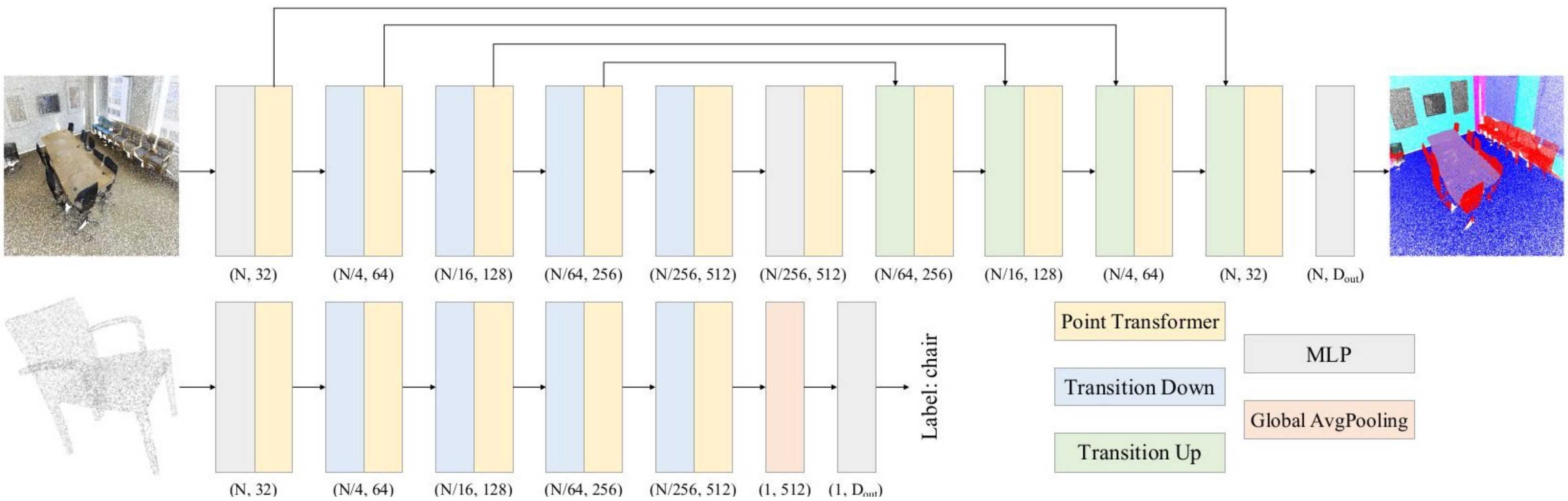
$$x'_j = \sum_{i \in N(x_j)} \rho[\beta(\phi(x_j), \psi(x_i)) + \delta(p_j - p_i)] \odot \alpha(x_i)$$

function other than
dot product

position of points

Zhao et al. “Point Transformer” 2020

Point Transformer



Pooing, un-pooling, and residual connections similar to PointNet++

Zhao et al. “Point Transformer” 2020

Point Transformer

Object Classification (ModelNet40)

Method	input	mAcc	OA
3DShapeNets [43]	voxel	77.3	84.7
VoxNet [20]	voxel	83.0	85.9
Subvolume [23]	voxel	86.0	89.2
MVCNN [30]	image	–	90.1
PointNet [22]	point	86.2	89.2
PointNet++ [24]	point	–	91.9
SpecGCN [36]	point	–	92.1
PointCNN [18]	point	88.1	92.2
DGCNN [40]	point	90.2	92.2
PointWeb [50]	point	89.4	92.3
SpiderCNN [44]	point	–	92.4
PointConv [42]	point	–	92.5
KPConv [33]	point	–	92.9
InterpCNN [19]	point	–	93.0
PointTransformer	point	**90.6**	**93.7**

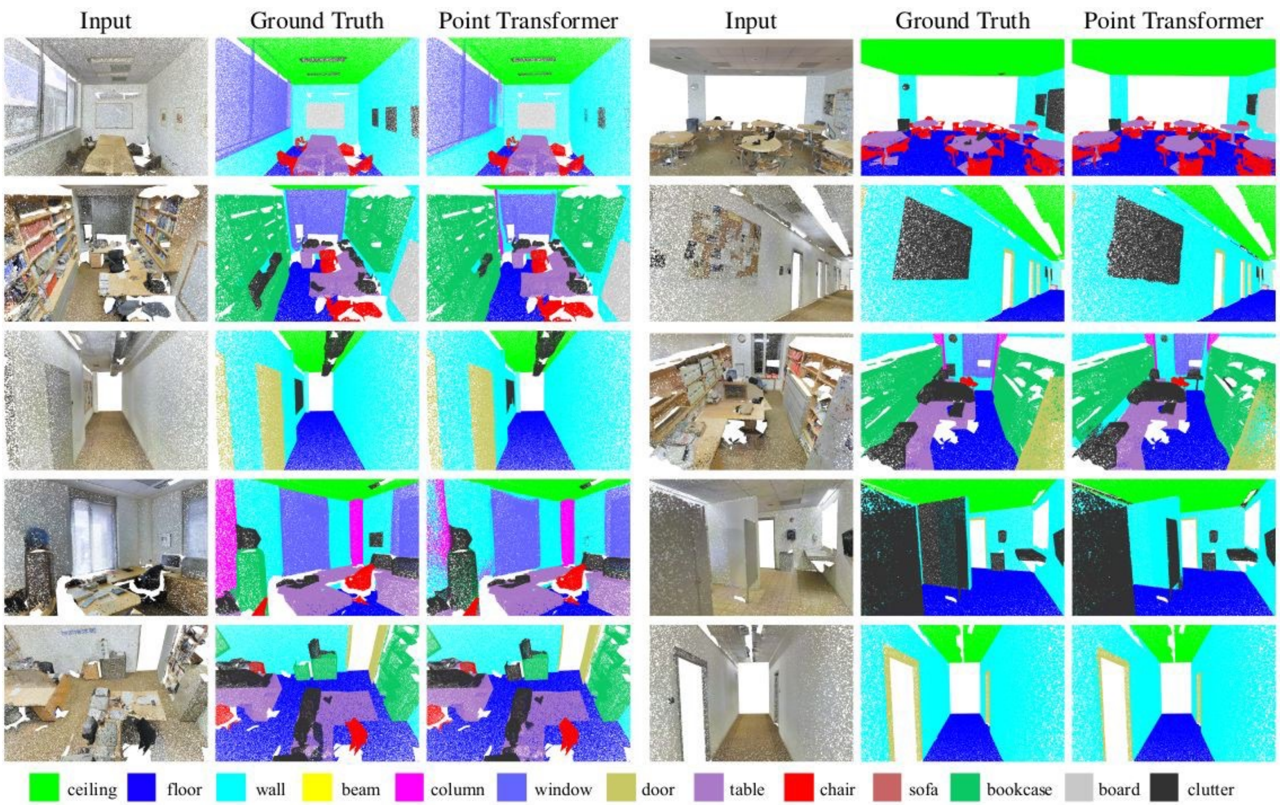
Object Part Segmentation (ShapeNetPart Dataset)

Method	cat. mIoU	ins. mIoU
PointNet [22]	80.4	83.7
PointNet++ [24]	81.9	85.1
SPLATNet	83.7	85.4
SpiderCNN [44]	81.7	85.3
PCNN [38]	81.8	85.1
PointCNN [18]	84.6	86.1
DGCNN [40]	82.3	85.1
SGPN [39]	82.8	85.8
PointConv [42]	82.8	85.7
InterpCNN [19]	84.0	86.3
KPConv [33]	**85.1**	86.4
PointTransformer	83.7	**86.6**

State-of-the-art @2020

Zhao et al. “Point Transformer” 2020

Point Transformer



Semantic Segmentation on S3DIS Dataset

https://paperswithcode.com/sota/semantic-segmentation-on-s3dis

State-of-the-art @2020

Zhao et al. "Point Transformer" 2020