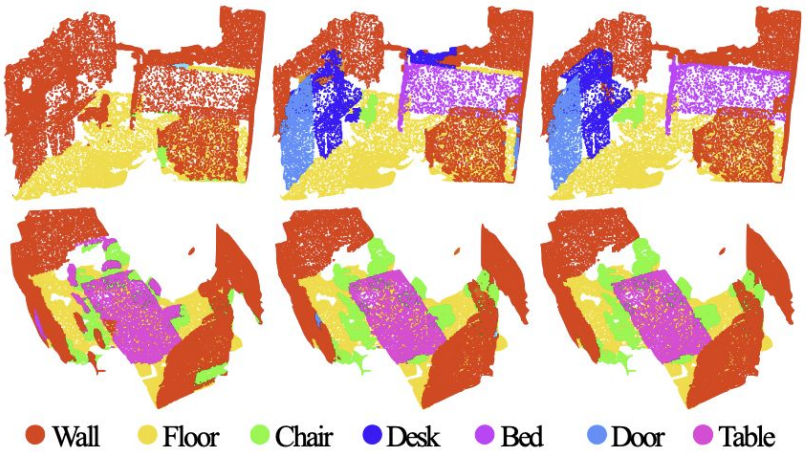
PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space

Charles R. Qi Li Yi Hao Su Leonidas J. Guibas

Conference on Neural Information Processing Systems (NIPS) 2017

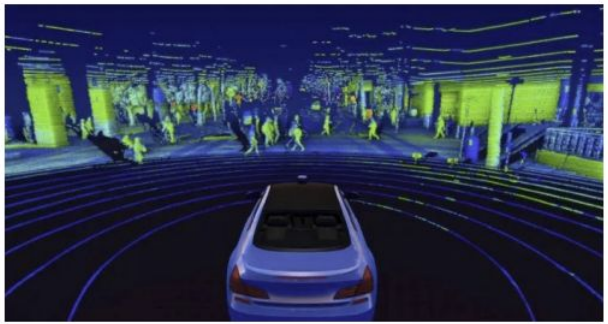


Sayem Nazmuz Zaman, Chia-Hong Hsu

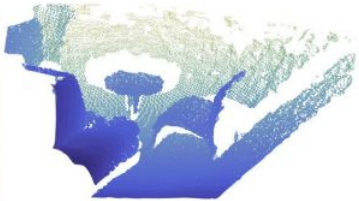
Image Credit: https://arxiv.org/pdf/1706.02413

Real World Motivation: Why Point Clouds/Point Sets?

LiDAR Sensors



Depth Cameras



It is important to effectively process **point cloud data!**

Image Credit: https://www.cs.utexas.edu/~yukez/cs391r_fall2023/slides/pre_08-31_Lance.pdf

Challenges for Deep Learning on Point Sets

Deep Learning for point sets need to address unique characteristics of the point set/cloud data:

1. Density variability.
2. Irregular data structure (no fixed grid).

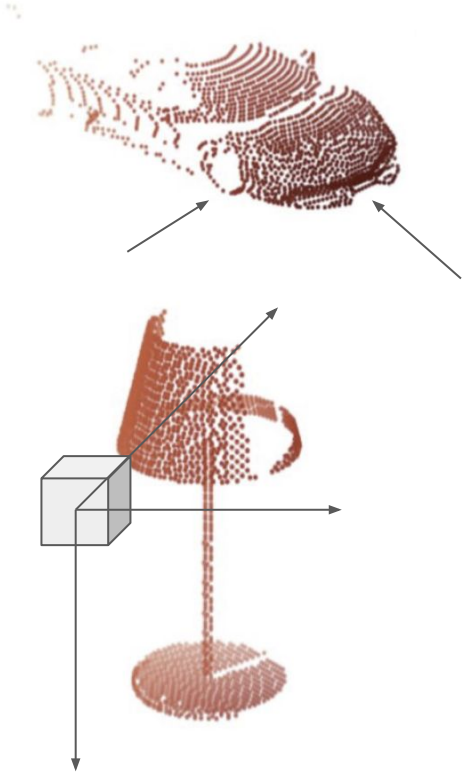


Image Credit: https://www.cs.utexas.edu/~yukez/cs391r_fall2023/slides/pre_08-31_Lance.pdf

Challenges for Deep Learning on Point Sets

Deep Learning for point sets need to address unique characteristics of the point set/cloud data:

1. Density variability.
2. Irregular data structure (no fixed grid).
3. Importance of both local and global contexts in point set data.

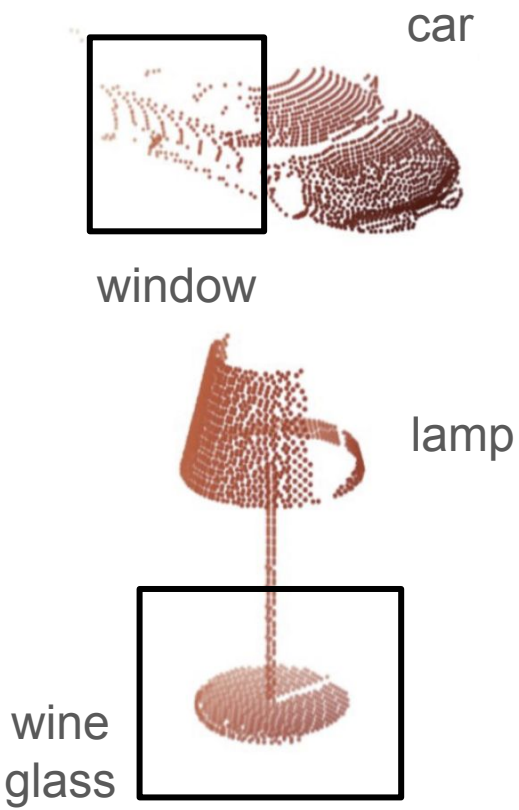


Image Credit: https://www.cs.utexas.edu/~yukez/cs391r_fall2023/slides/pre_08-31_Lance.pdf

Agenda

1. Challenges for Deep Learning on Point Sets ✓
2. Problem Statement
3. Method
   a. PointNet++ & PointNet
   b. Comparison with Unet & CNN
4. Experiments
5. Summary
6. Q&A / Feedbacks

Problem Statement

$$\mathcal{X} = (M, D)$$

$M \subseteq \mathbb{R}^d$: a set of points in d-dimensional space

D: Euclidean distance metric

Objective: Learning set functions f that takes the metric space χ as input and produce information of semantic interest (for classification & segmentation)

Problem Statement

Visually,

Point Cloud (Set):

1. (x, y, z) coordinates
2. Color channels (r,g,b)
3. …

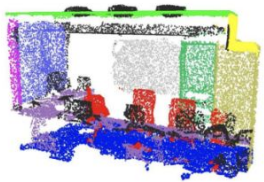
$$\underbrace{\qquad\qquad}_{\mathcal{X}}$$

PointNet++

$$f(\mathcal{X})$$

Classification

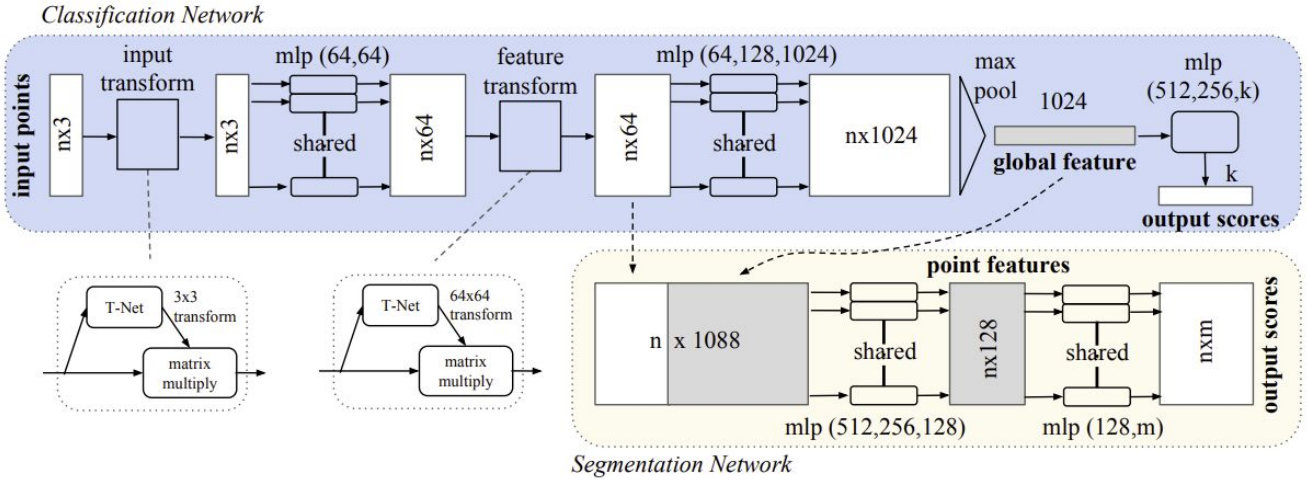
Part Segmentation



Semantic Segmentation

Image Credit: https://github.com/charlesq34/pointnet

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation



How does PointNet satisfy the properties of Point Sets?

Property 1: Unordered

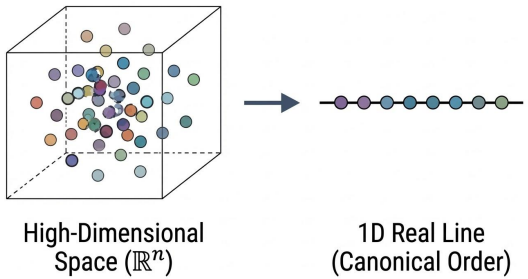
Point set representations should be invariant to input ordering.

This can be done in three ways:

1. Sort to a canonical order
2. Use an RNN and augment training with random orders
3. Use a symmetric function

We want a bijection map between high-dim space and 1d real line for sorting

$$f : \mathbb{R}^N \rightarrow \mathbb{R}$$



However, this spatial proximity is not guaranteed as dimension is reduced!

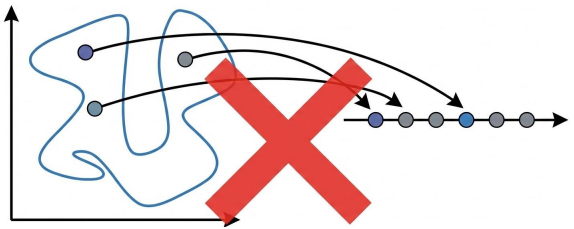


Image Credit: Generated with NanoBananaPro

How does PointNet satisfy the properties of Point Sets?

Property 1: Unordered

Point set representations should be invariant to input ordering.

This can be done in three ways:

1. Sort to a canonical order
2. Use an RNN and augment training with random orders
3. Use a symmetric function

"RNN has good robustness to input ordering for sequences with small length but does not scale"
-- OrderMatters

How does PointNet satisfy the properties of Point Sets?

Property 1: Unordered

Point set representations should be invariant to input ordering.

This can be done in three ways:

1. Sort to a canonical order
2. Use an RNN and augment training with random orders
3. Use a symmetric function

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n))$$

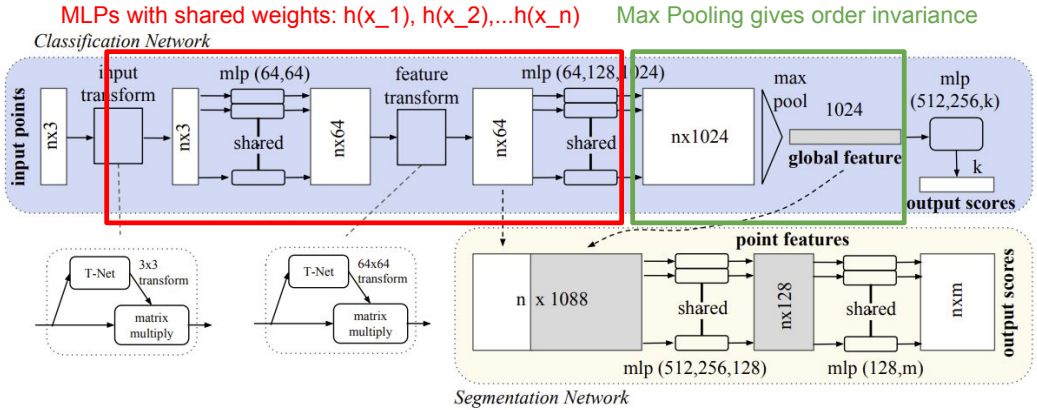
1. $f : 2^{\mathbb{R}^N} \to \mathbb{R}$
2. $h : \mathbb{R}^N \to \mathbb{R}^K$
3. $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ (Symmetric Function)

PointNet is **invariant to input ordering** due to **Max Pooling**

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n))$$

$$f(x_1, x_2, ..., x_n) = \gamma \left(\underset{i=1,\ldots,n}{\text{MAX}} \{h(x_i)\} \right)$$

1. $f : 2^{\mathbb{R}^N} \to \mathbb{R}$
2. $h : \mathbb{R}^N \to \mathbb{R}^K$ **MLPs**
3. $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ **Max Pooling + Global Signature**



Does PointNet satisfy the properties of Point Sets?

Property 2: Invariance under transformations

1. Changing pose **should not** change object identity
2. Representations **should be invariant** to rigid pose transformations (translation and rotations)

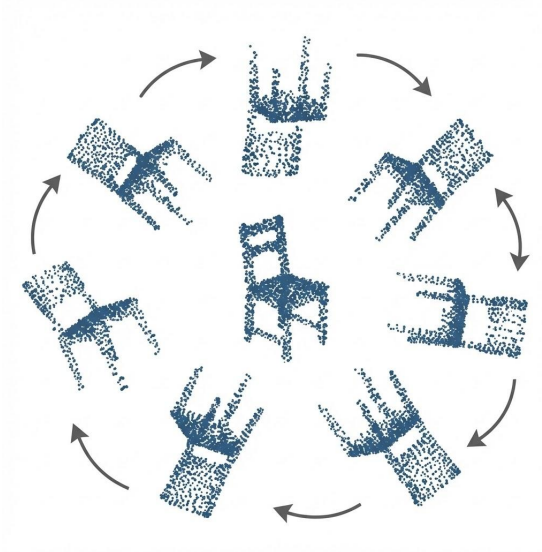


Image Credit: Generated with NanoBananaPro

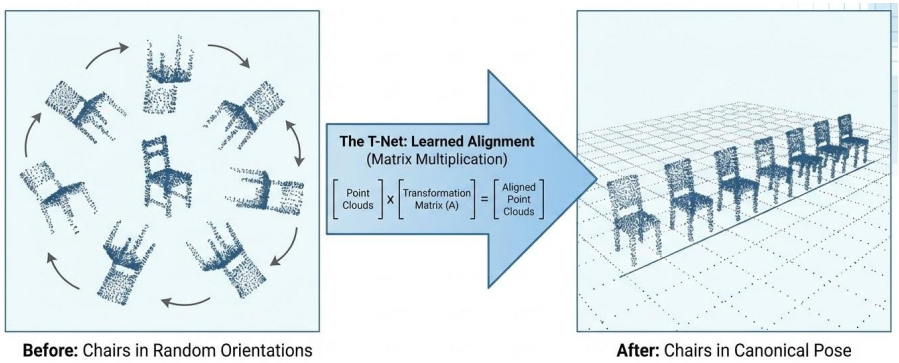
Does PointNet satisfy the properties of Point Sets?

Property 2: Invariance under transformations

How can we do this?

Align all **input sets** to a canonical space

before extracting features



Visual Idea: The T-Net learns to align data (like point clouds of chairs), canonical space to improve classification.

Image Credit: Generated with NanoBananaPro

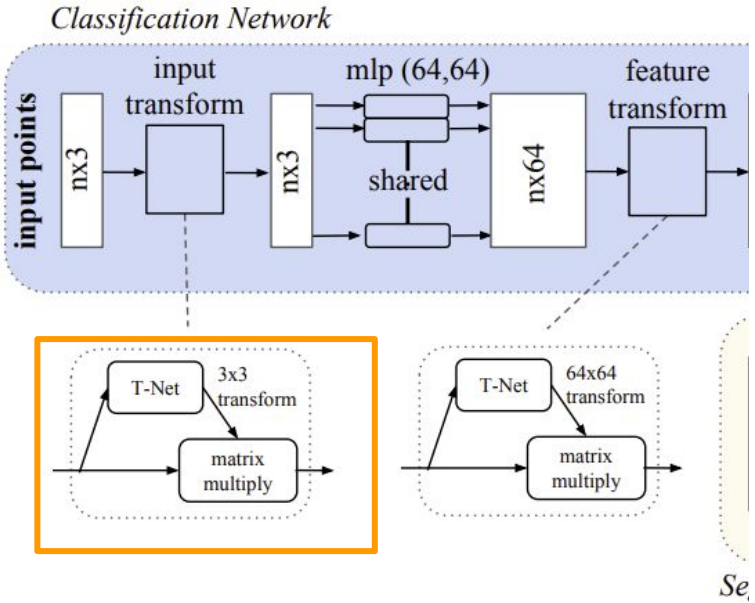
PointNet is Invariant under transformation

Align all **input sets** to a canonical space **before** extracting features

T (Transform)-Net takes **raw coordinates** and predicts an affine 3x3 transformation matrix

Just like a “Mini-PointNet”

1. Shared MLPs
2. Max Pooling
3. Fully connected Layers



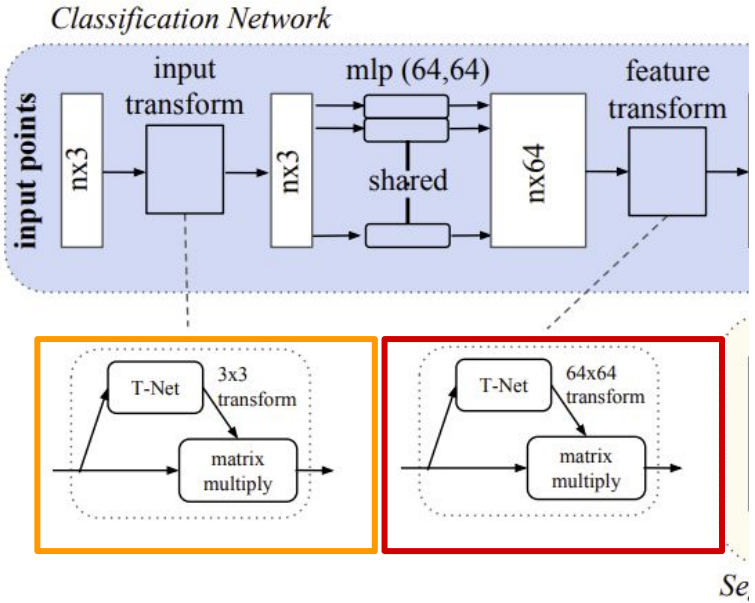
T-Net gives invariance by transforming to canonical pose

PointNet is Invariant under transformation

A second T-Net after the first few layers aligns **point features**

Second T-Net (64x64) has 4,096 parameters! Add constraints via regularization to prevent network from learning “illegal” transforms:

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



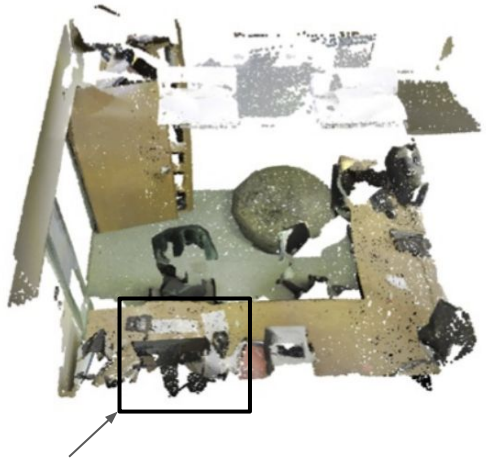
T-Net gives invariance by transforming to canonical pose

Align Point features

Does PointNet satisfy the properties of Point Sets?

Property 3: Interaction between Points

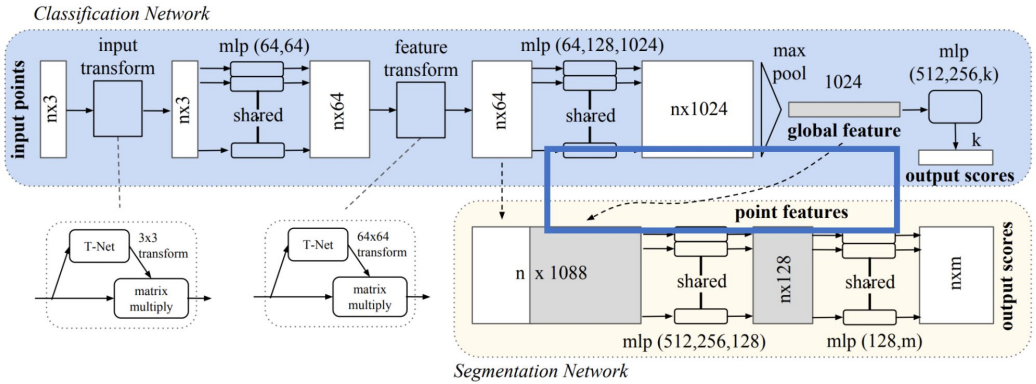
1. Points are in a metric space
2. There are meaningful local neighbourhood of points
3. Many local prediction tasks such as segmentation require summarizing information in a local neighborhood

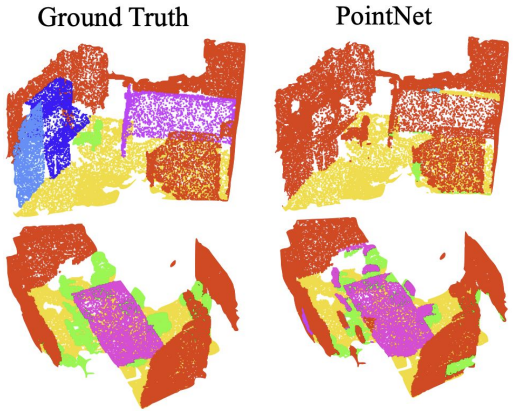


How to get local features that take global context into account?

Does **PointNet** properly **capture interaction** between **Points**?

How to get local features that take global context into account?



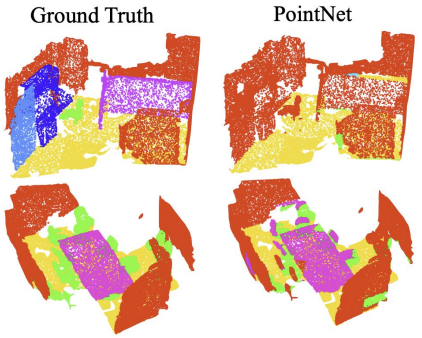


Concatenating global-to-local features combines local + global info → This prevents the network from capturing **local context** at **different scales**

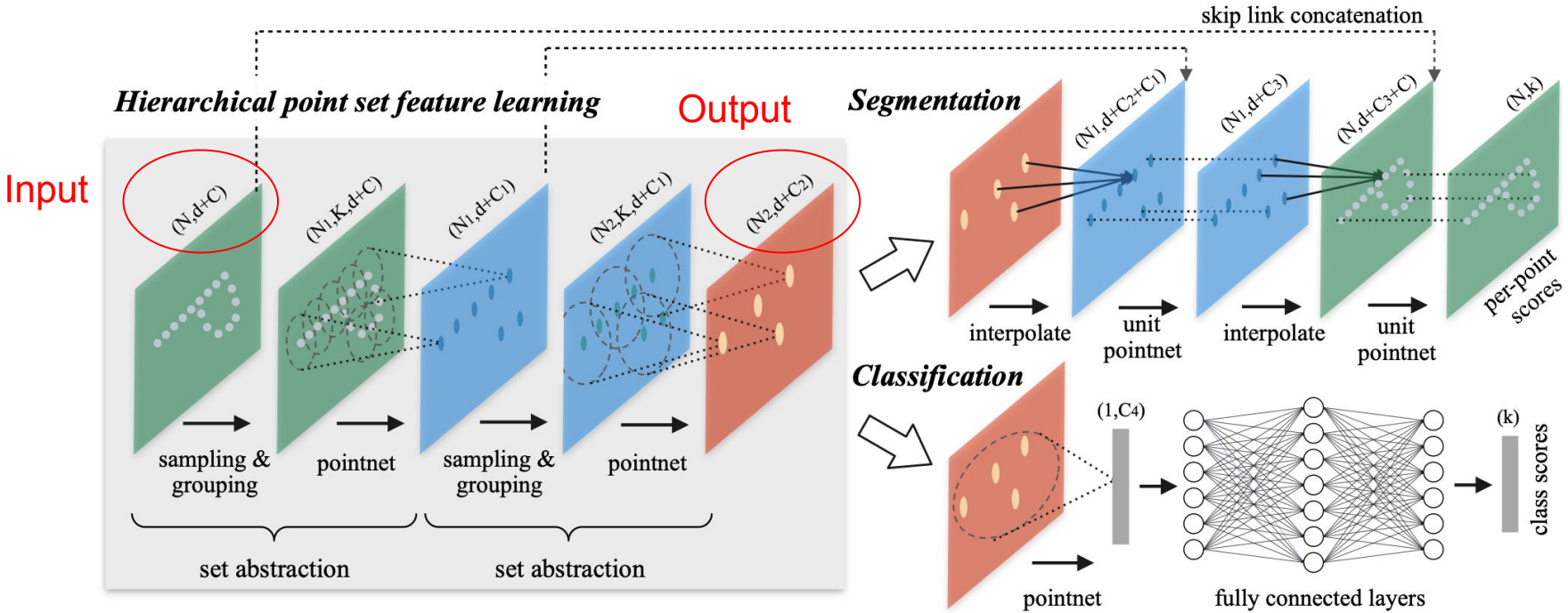
Does **PointNet** properly **capture interaction** between **Points**?

Not quite. While PointNet captures a global signature, it lacks local structure **induced** by the metric space

In addition, PointNet does not take into account **density variability,** which might come from perspective effects in 3D scanning



Method: PointNet++



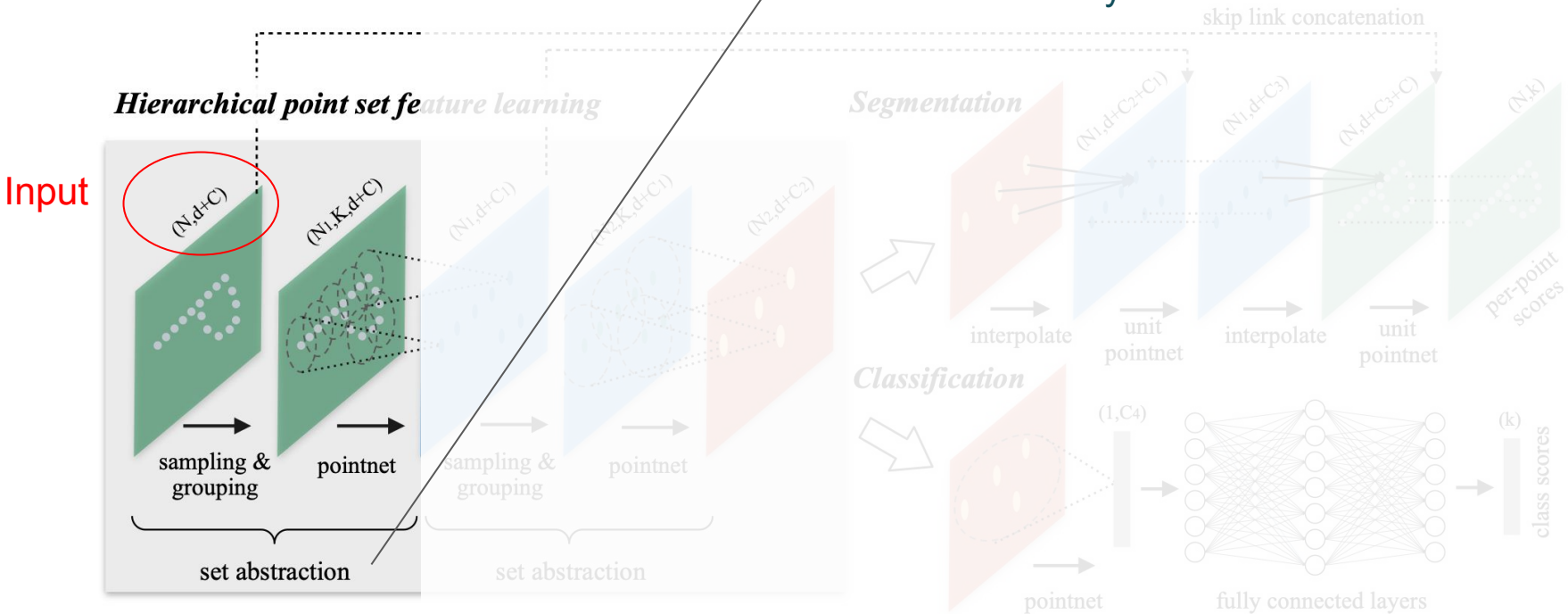
Overview of PointNet++

Image Credit: https://arxiv.org/pdf/1706.02413

Method: PointNet++

A *set abstraction* consists of three layers.

1. Sampling layer
2. Grouping layer
3. PointNet layer

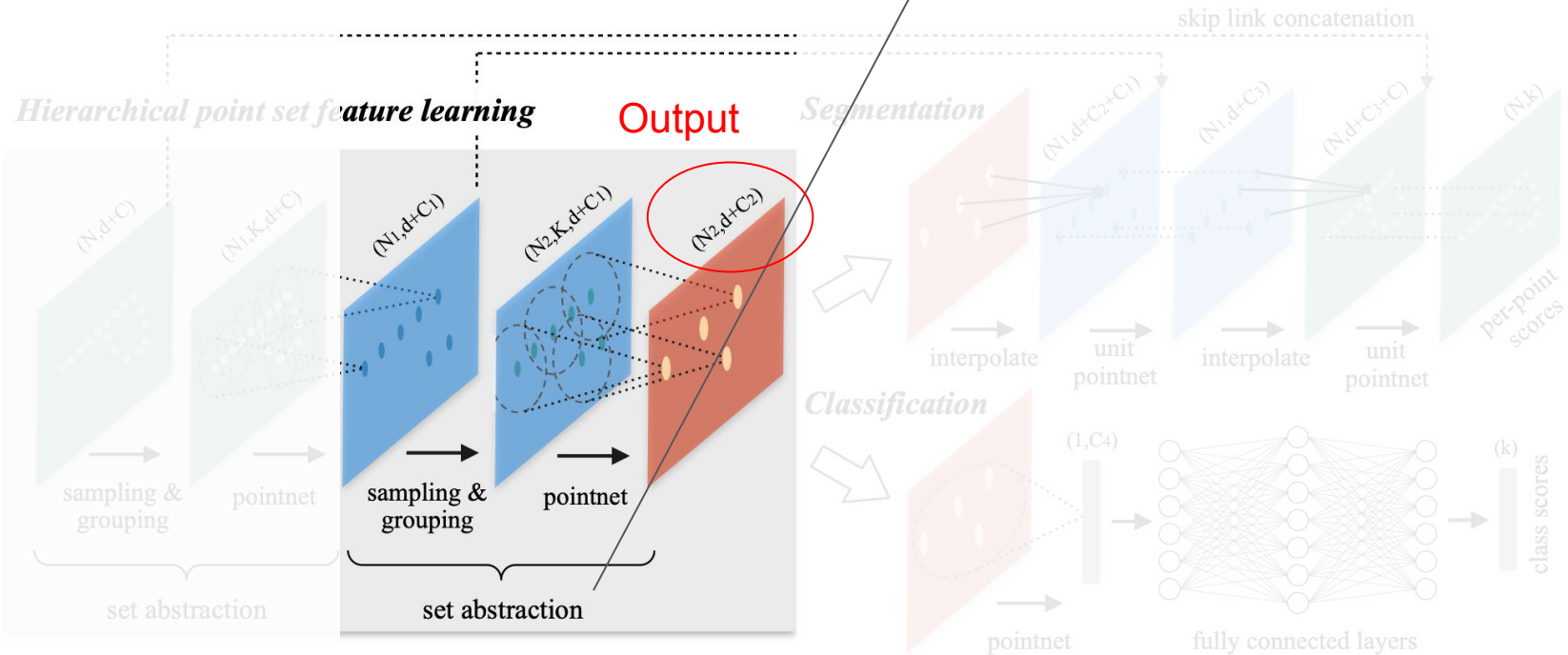


Overview of PointNet++

Image Credit: https://arxiv.org/pdf/1706.02413

Method: PointNet++

Then, *set abstraction* is *applied recursively* to obtain the output.

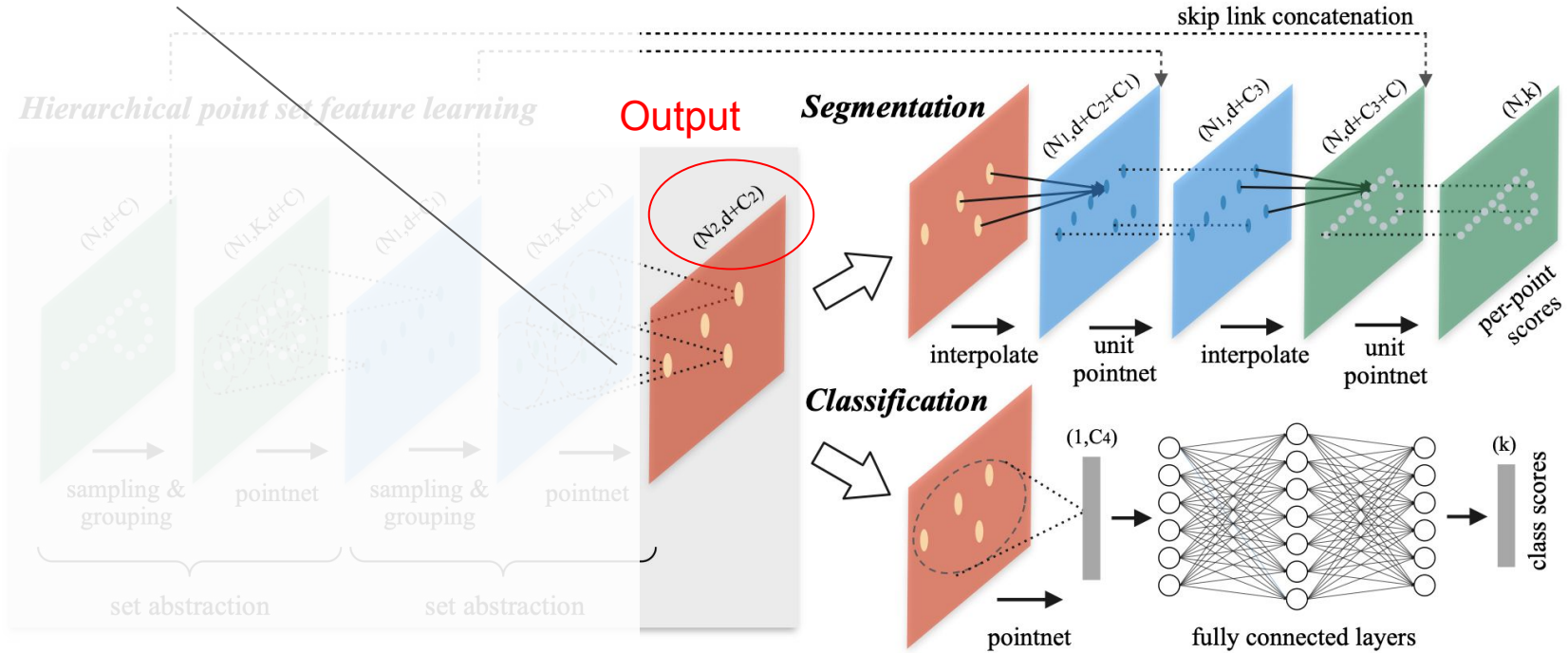


Overview of PointNet++

Image Credit: https://arxiv.org/pdf/1706.02413

Method: PointNet++

Apply different aggregation methods given the output for segmentation and classification.



Overview of PointNet++

Image Credit: https://arxiv.org/pdf/1706.02413

Set abstraction: **Sampling Layer**

Given point set $P = \{x_1, \ldots, x_N\}$ run *farthest point sampling (FPS)* to obtain a **subset** of the point set, $S = \{x_{i_1}, \ldots, x_{i_{N'}}\}$

1. **Initialization:** Select a random initial point from the point set P and add it to the sampled set S. Determine set size N'.
2. **Distance Calculation:** For all remaining points in P, calculate their distance to the *nearest* point in the currently sampled set S.
3. **Selection:** Add the point with the *maximum* calculated minimum distance to S.
4. **Repeat Step 2 and 3** until obtain N' points in S.

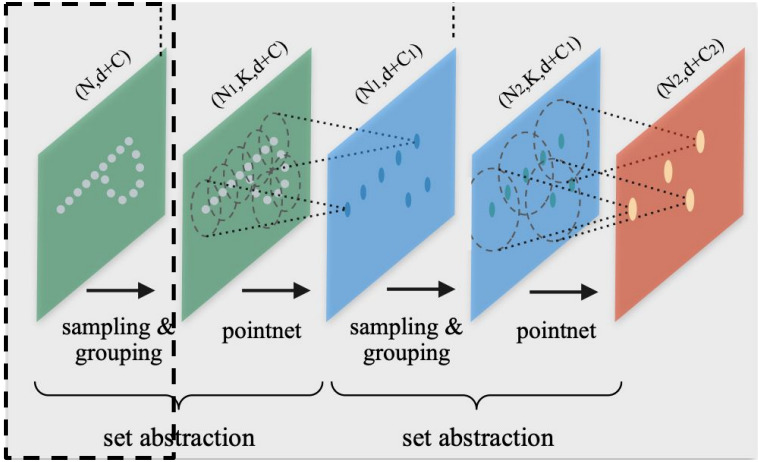


Image Credit: https://arxiv.org/pdf/1706.02413

Set abstraction: **Grouping layer**

Treat the sampled S from the last step as a set of N' *centroids*.

Construct a group set G that is the N' local regions from the point set P. Each region consists of K neighboring points, where each point has a dimension of $d + C$ representing spatial coordinates and feature attributes.

The K neighboring points are selected via *kNN* or *ball query* of the centroids in S, resulting in G's shape being $(N', K, d+C)$.

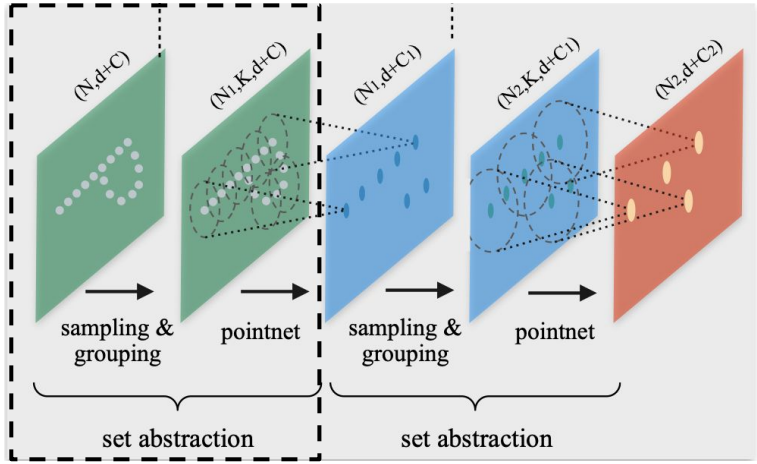


Image Credit: https://arxiv.org/pdf/1706.02413

Set abstraction: **PointNet Layer**

The abstraction layer processes the grouped point set G with shape $(N', K, d+C)$ to produce *an aggregated representation* for each local region with shape $(N', d+C')$. This is the set abstraction F for P.

- A PointNet architecture is applied to each of the N' local regions, aggregating the K points into a single, compact feature vector.
- Feature Transformation. The layer maps the original feature dimension C to a new, abstracted feature dimension C'. The spatial dimension d is also passed to the next layer.

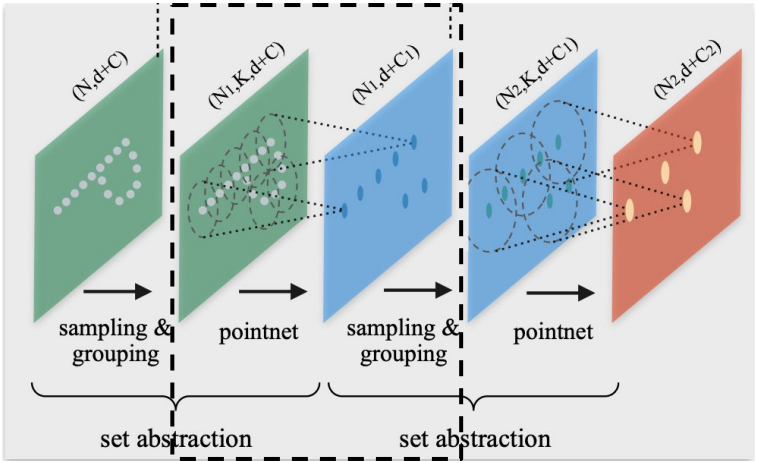


Image Credit: https://arxiv.org/pdf/1706.02413

Feature propagation for segmentation & classification

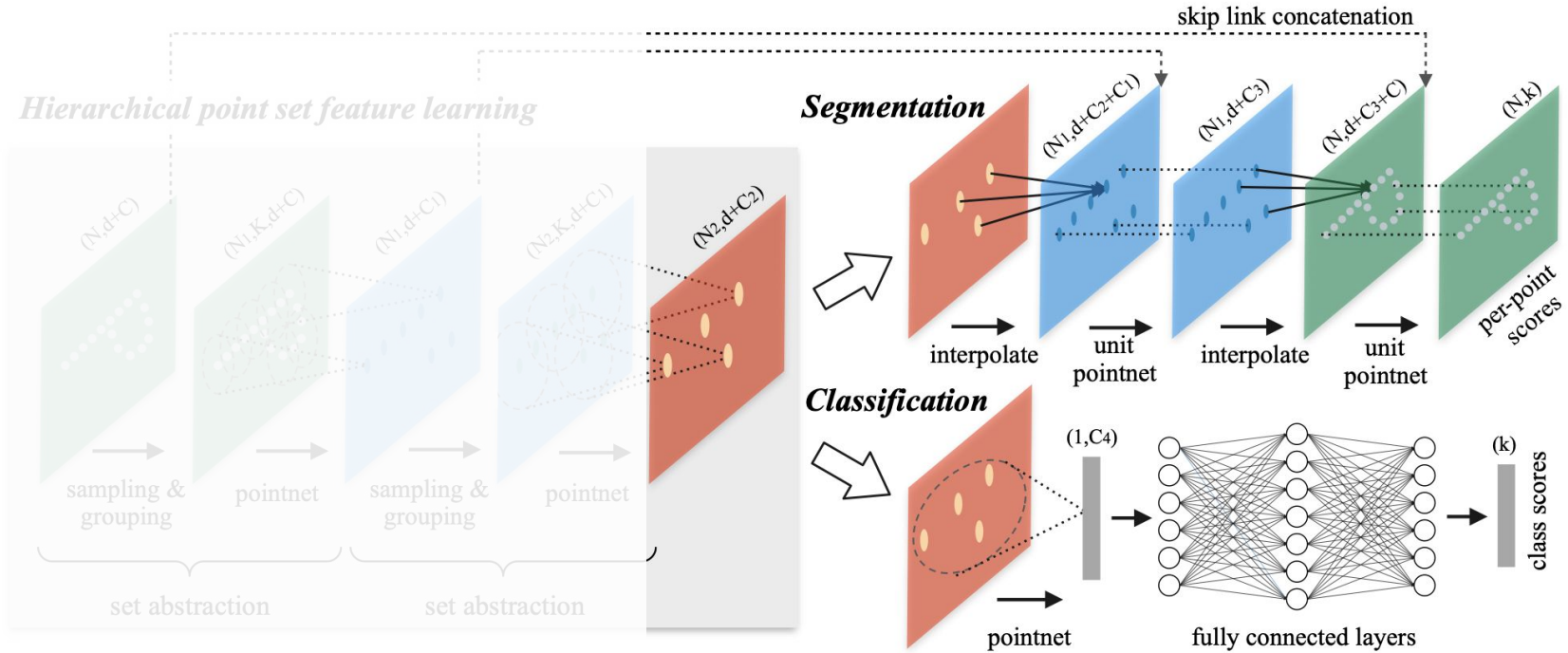


Image Credit: https://arxiv.org/pdf/1706.02413

Feature propagation for segmentation & classification

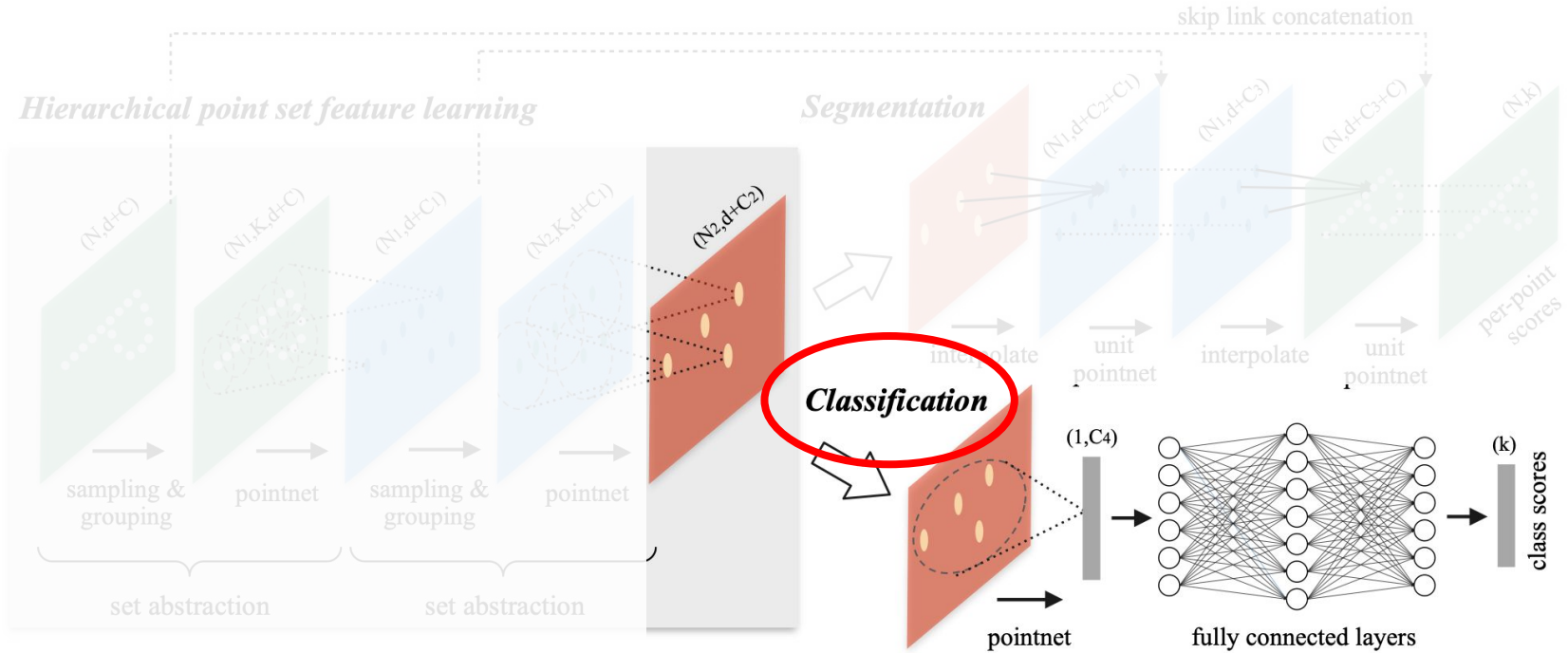


Image Credit: https://arxiv.org/pdf/1706.02413

Feature propagation for segmentation & classification

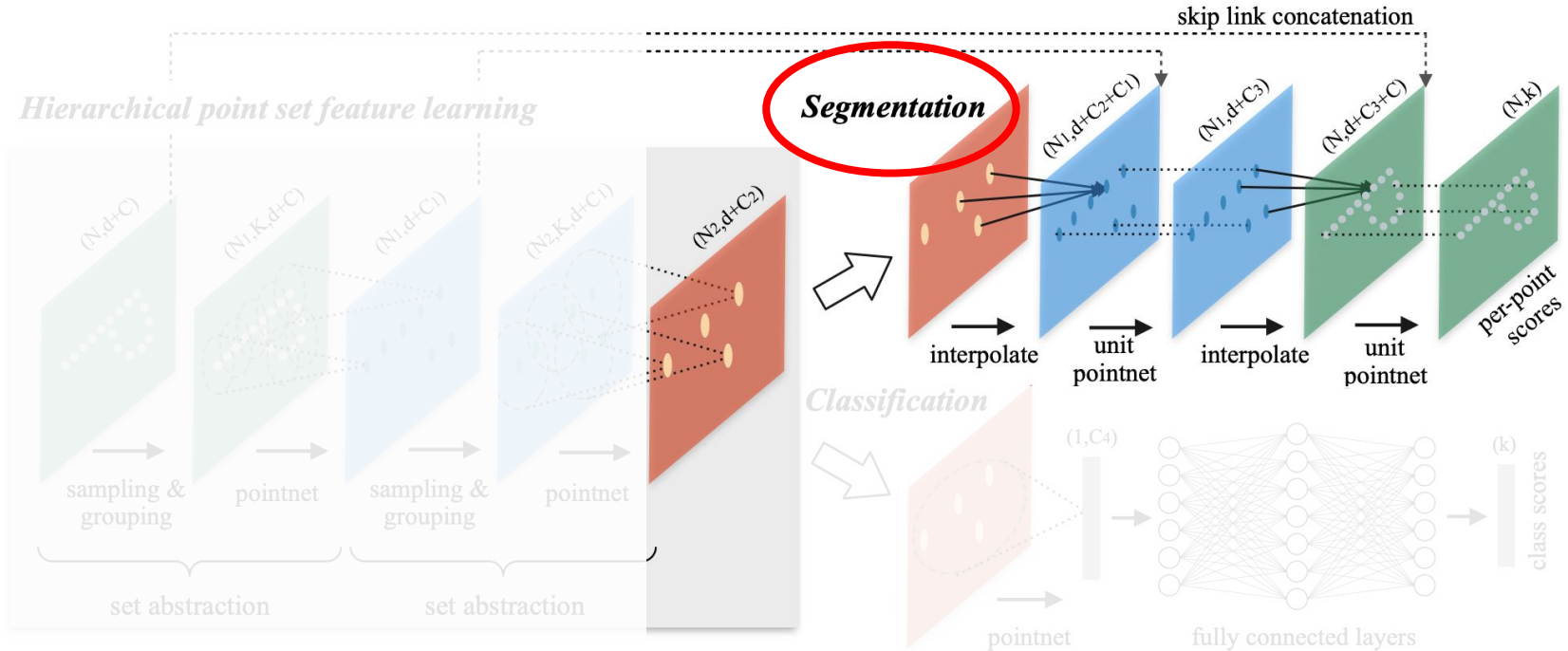
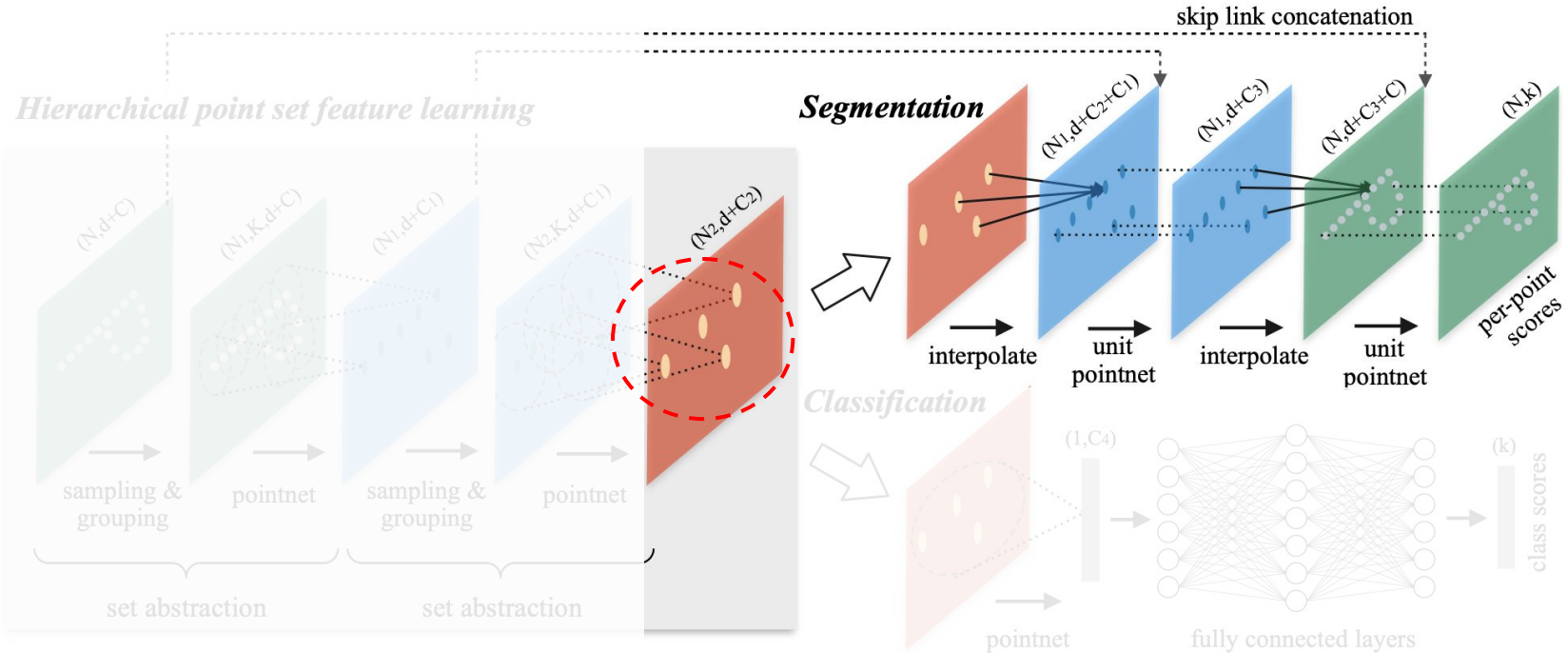


Image Credit: https://arxiv.org/pdf/1706.02413

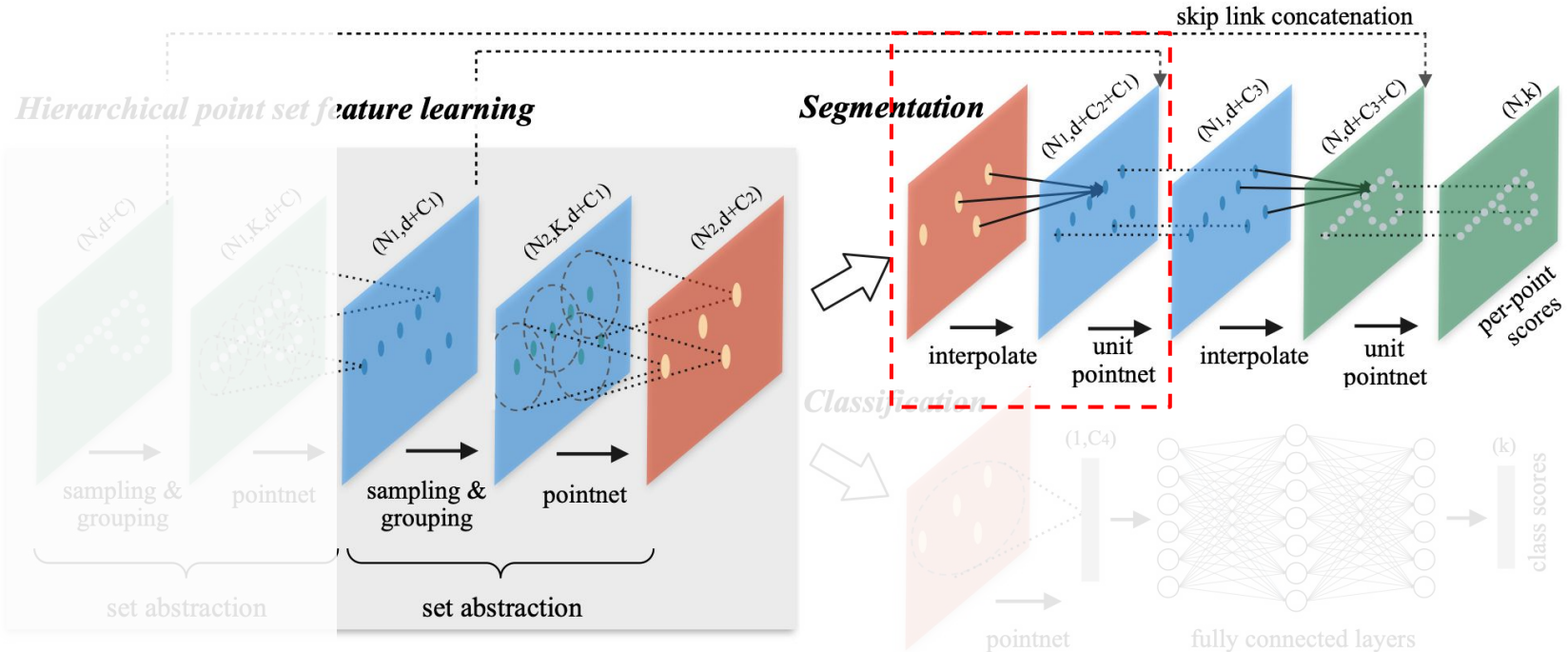
Feature propagation for segmentation & classification



In the end of each set abstraction level, the original set was *subsampled*, i.e., *smaller*.

Image Credit: https://arxiv.org/pdf/1706.02413

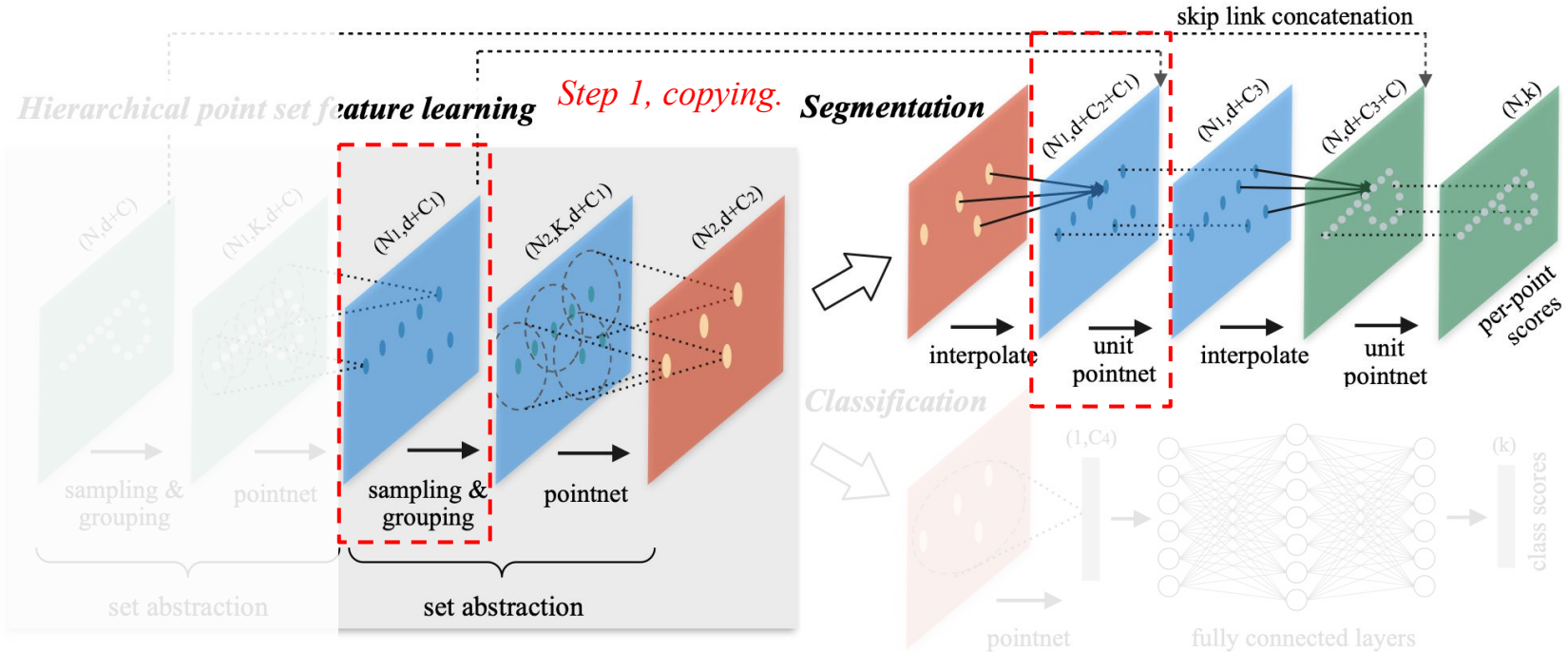
Feature propagation for segmentation & classification



Feature maps from the subsampled set back to the original set is propagated via interpolation.
How to propagate?

Image Credit: https://arxiv.org/pdf/1706.02413

Feature propagation for segmentation & classification

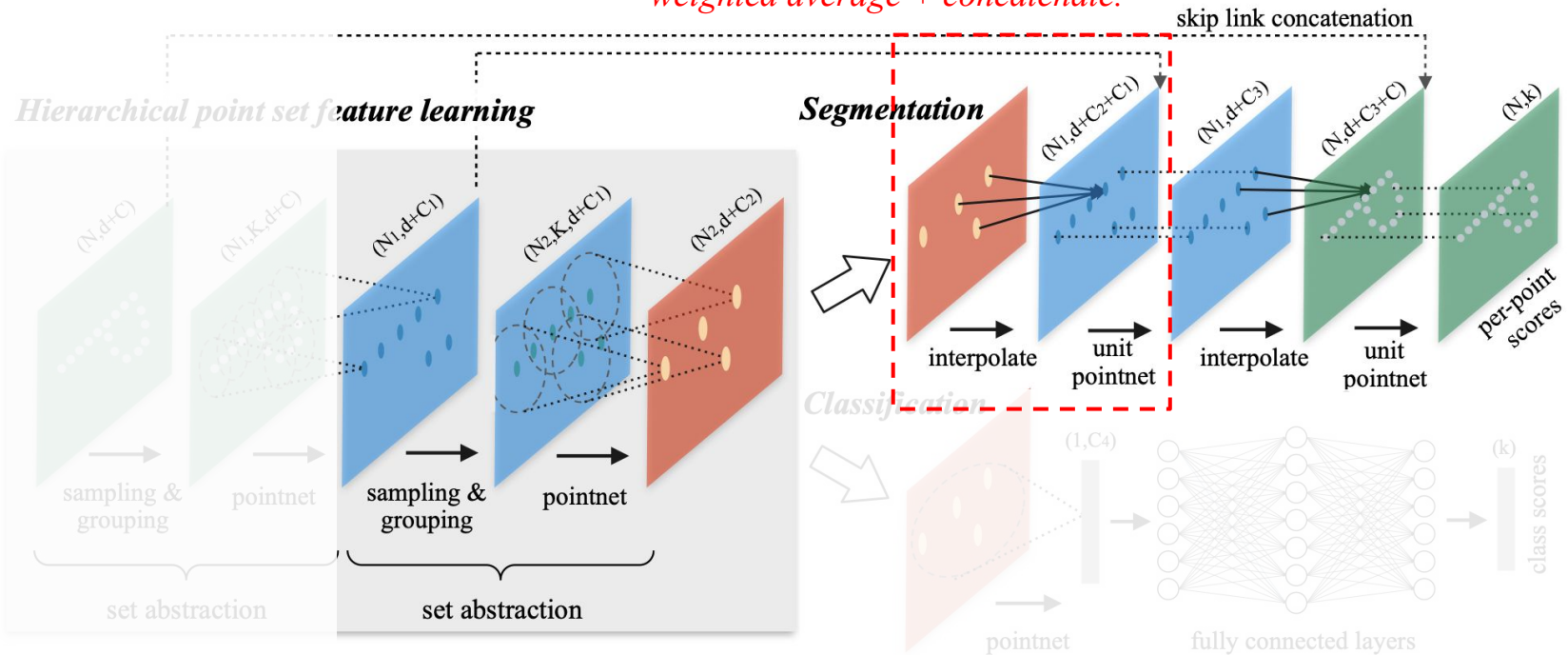


Step 1, get the original set's feature map via *skip connection*, i.e., *copying*.

Image Credit: https://arxiv.org/pdf/1706.02413

Feature propagation for segmentation & classification

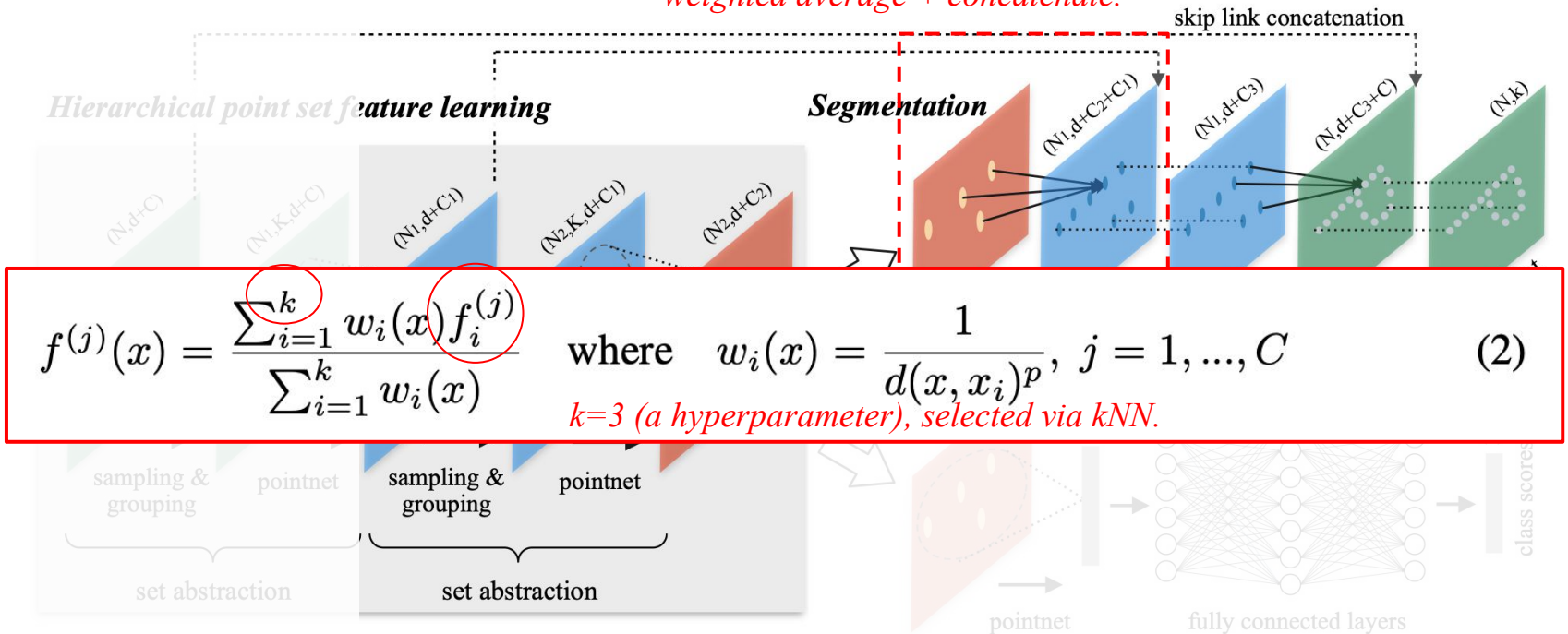
Step 2, compute inverse distance weighted average + concatenate.



Step 2, for each point in the original set, compute the *inverse distance weighted average* of features from the subsample, concatenate the interpolated feature with the current feature map.

Feature propagation for segmentation & classification

Step 2, compute inverse distance weighted average + concatenate.



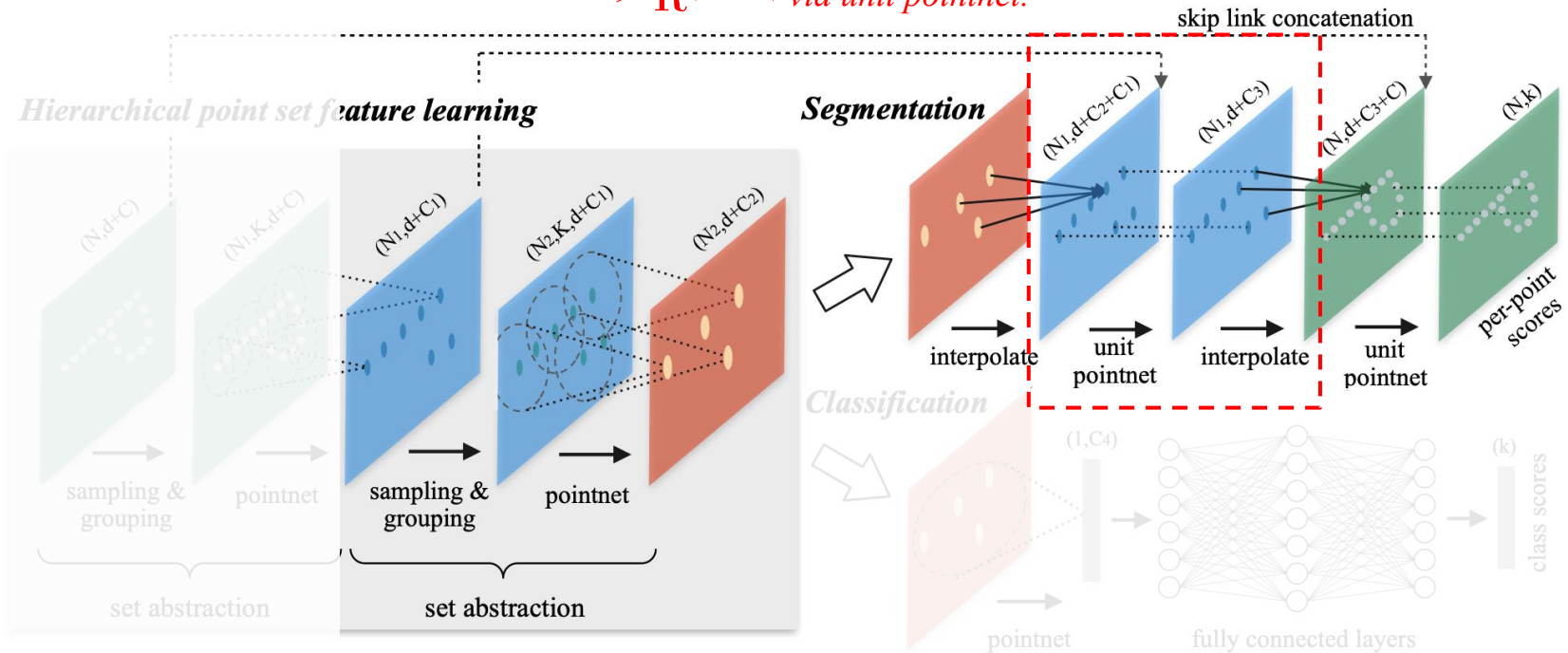
$$f^{(j)}(x) = \frac{\sum_{i=1}^{k} w_i(x) f_i^{(j)}}{\sum_{i=1}^{k} w_i(x)} \quad \text{where} \quad w_i(x) = \frac{1}{d(x, x_i)^p}, \; j = 1, ..., C \qquad (2)$$

k=3 (a hyperparameter), selected via kNN.

Step 2, for each point in the original set, compute the *inverse distance weighted average* of features from the subsample, concatenate the interpolated feature with the current feature map.

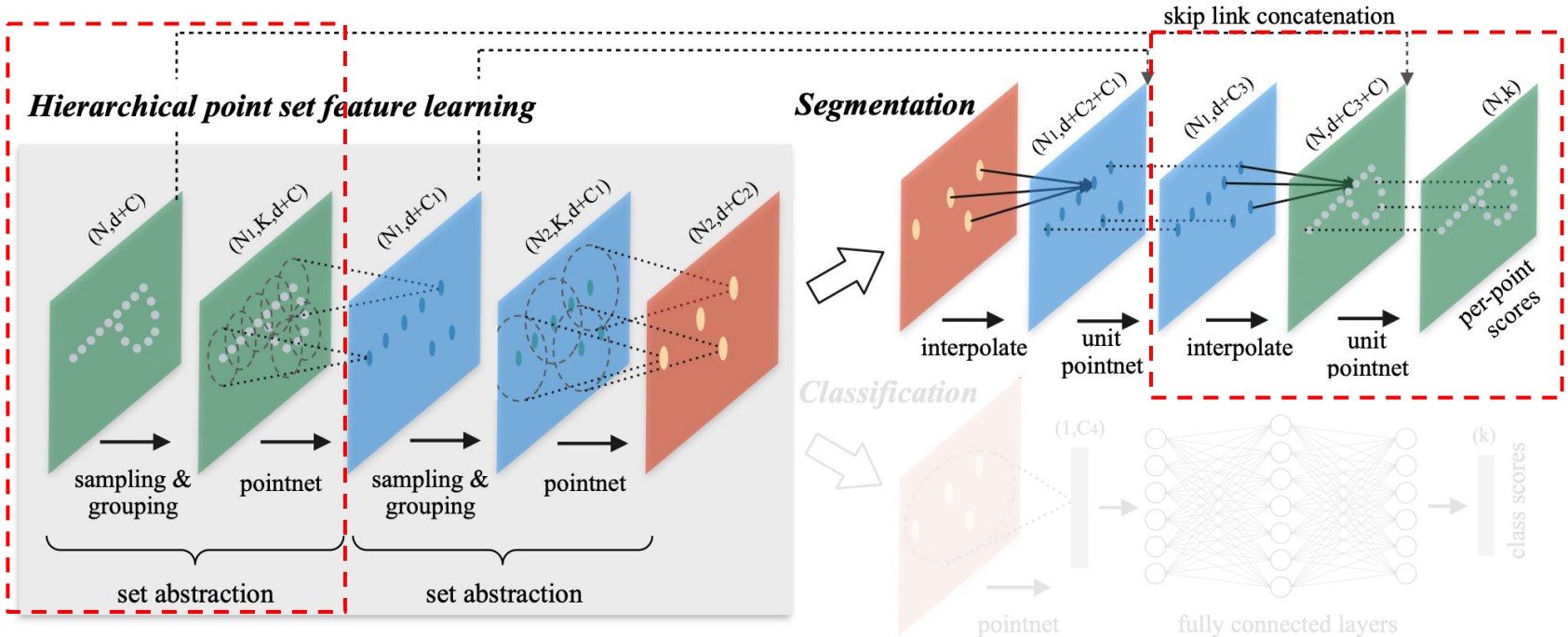
Feature propagation for segmentation & classification

Step 3, concatenated features $\mathbf{R}^{(d+C2+C1)}$
-> $\mathbf{R}^{(d+C3)}$ *via unit pointnet.*



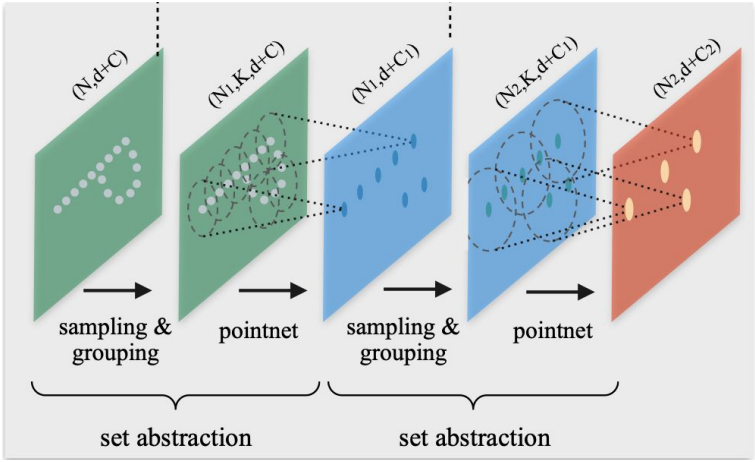
Step 3, pass the concatenated feature map to a *unit pointnet* that only modifies the channel dimensions (similar to 1x1 convolutional network).

Feature propagation for segmentation & classification



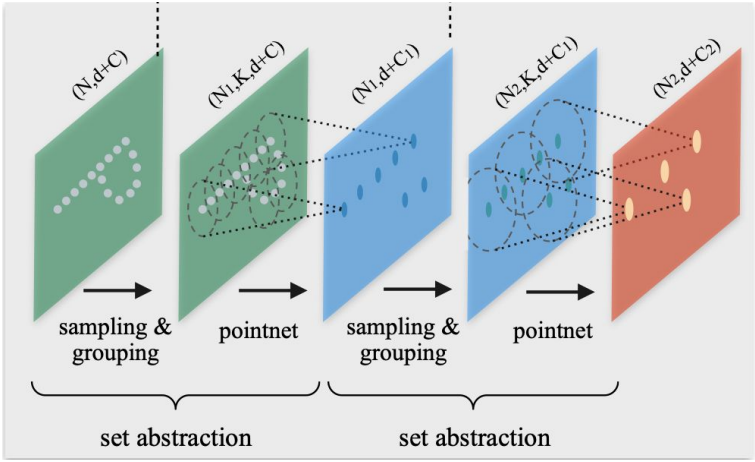
Repeat, until we recover all N points from the initial set P represented by N feature vectors.

Image Credit: https://arxiv.org/pdf/1706.02413



Models trained for *sparse* point sets may not recognize *fine-grained* local structures.

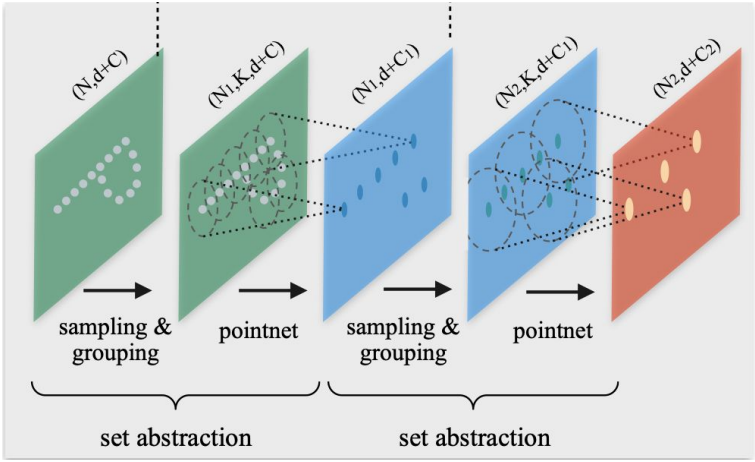
Image Credit: https://arxiv.org/pdf/1706.02413



Models trained for *sparse* point sets may not recognize *fine-grained* local structures.

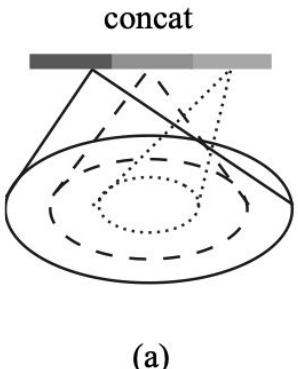
This challenge comes from the *non-uniform density* of point sets.

Image Credit: https://arxiv.org/pdf/1706.02413



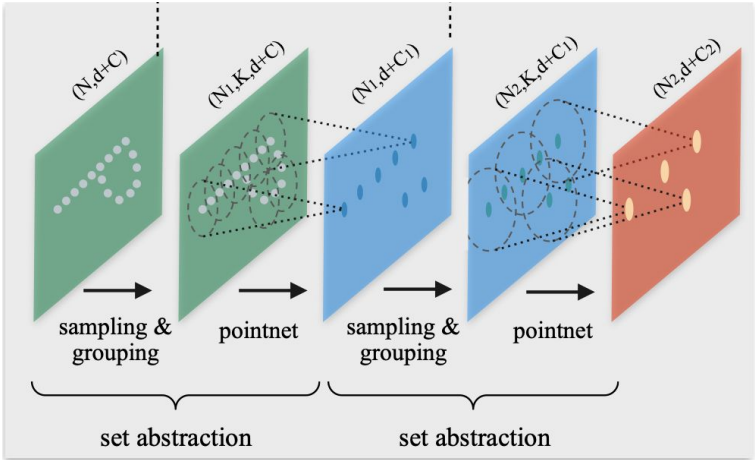
Models trained for *sparse* point sets may not recognize *fine-grained* local structures.

This challenge comes from the *non-uniform density* of point sets.



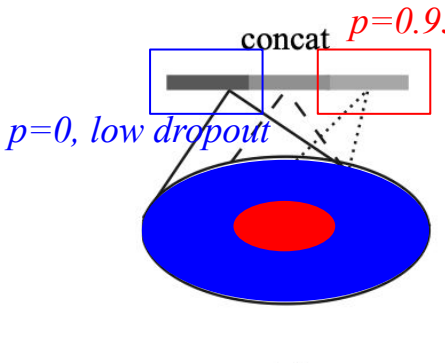
Multi-scale Grouping (MSG): The idea is to apply different PointNet for different group scale. Then, features of different scales are concatenated to form a multi-scale feature.

Image Credit: https://arxiv.org/pdf/1706.02413



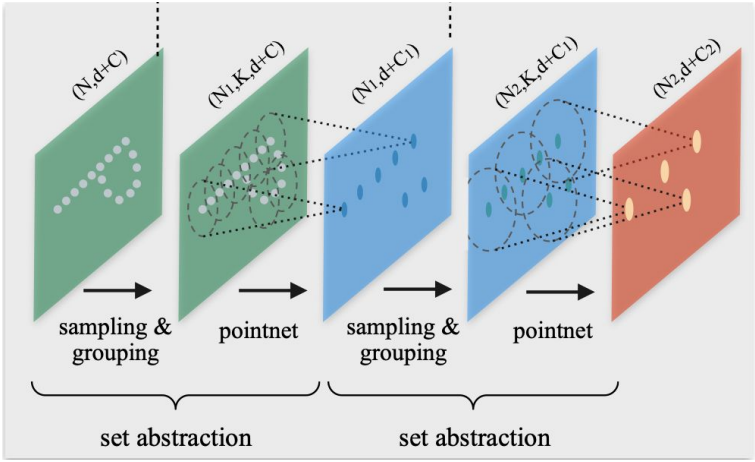
Models trained for *sparse* point sets may not recognize *fine-grained* local structures.

This challenge comes from the *non-uniform density* of point sets.



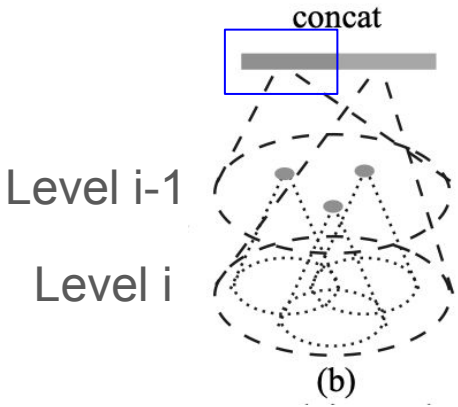
Multi-scale Grouping (MSG): In practice, a method the authors called *random input dropout* throws away random amounts of training dataset after grouping at training. At testing, all available points are kept.

Image Credit: https://arxiv.org/pdf/1706.02413



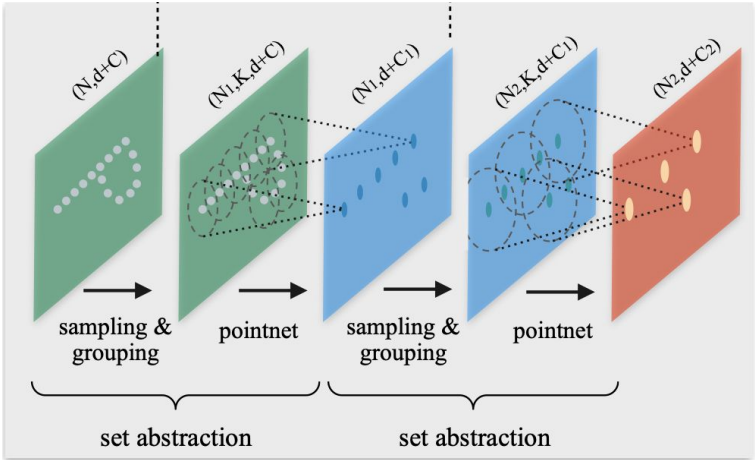
Models trained for *sparse* point sets may not recognize *fine-grained* local structures.

This challenge comes from the *non-uniform density* of point sets.



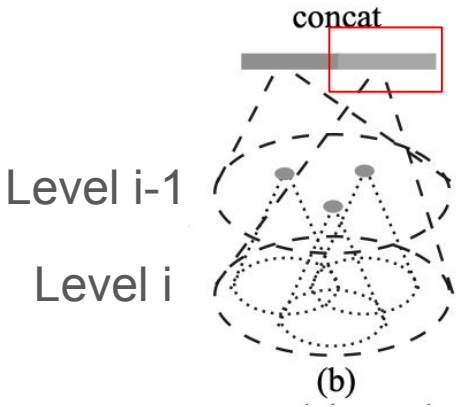
Multi-resolution Grouping (MRG): Two parts of features, one from the PointNet subsampling of a lower level feature map.

Image Credit: https://arxiv.org/pdf/1706.02413



Models trained for *sparse* point sets may not recognize *fine-grained* local structures.

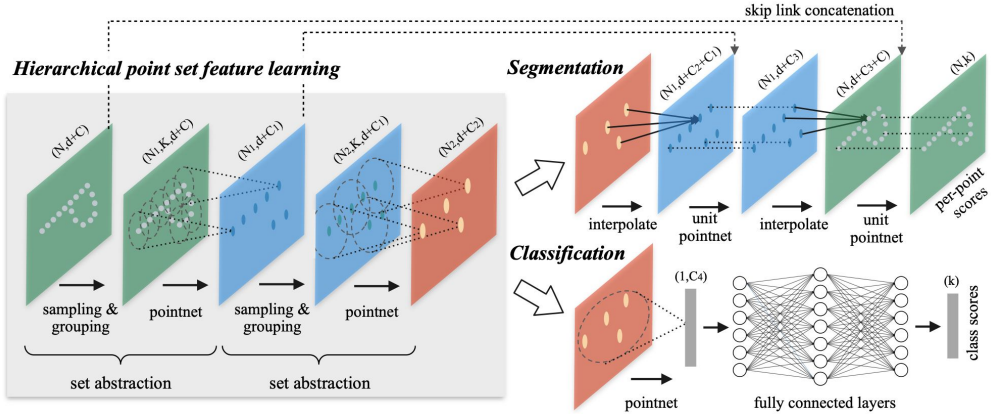
This challenge comes from the *non-uniform density* of point sets.



Multi-resolution Grouping (MRG): Two parts of features, one from the PointNet subsampling of a lower level feature map. The other from individual points in the local regions of the current map passed through a PointNet.

Image Credit: https://arxiv.org/pdf/1706.02413

PointNet++ ⇔ PointNet



UNet ⇔ CNN

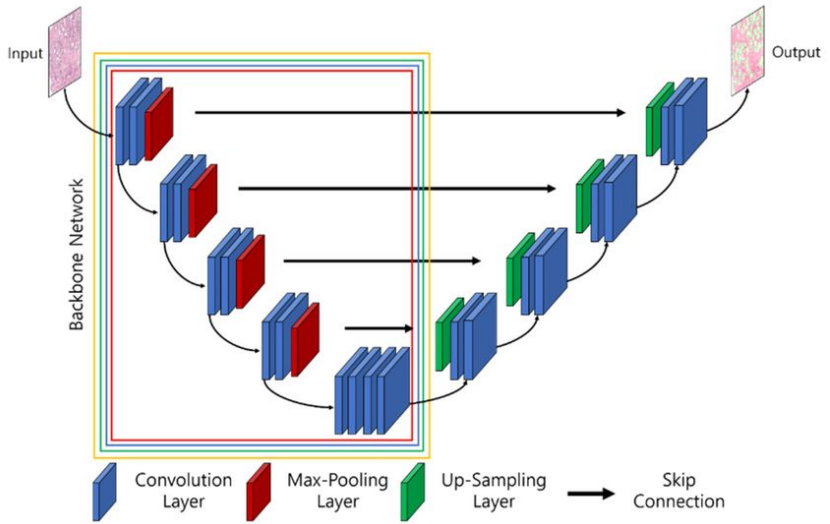


Image Credit: https://www.researchgate.net/figure/The-architecture-of-UNet-and-the-pretrained-convolution-neural-network-ie-backbones_fig5_367472621

Agenda

1. Challenges for Deep Learning on Point Sets ✓
2. Problem Statement ✓
3. Method
    a. PointNet++ & PointNet ✓
    b. Comparison with Unet & CNN ✓
4. Experiments
5. Summary
6. Q&A / Feedbacks

Experiments

Experimental Setup: Algorithms

PointNet++ Variants

1. SSG (Single-Scaled Grouping)
2. SSG + DP (with input dropout)
3. MSG + DP (Multi-Scaled Grouping)
4. MRG + DP (Multi-Resolution Grouping)

Baselines (3D)

1. Subvolume (volumetric CNN)
2. MVCNN (multi-view CNN)
3. PointNet
4. PointNet (vanilla)

Baselines (2D, MNIST)

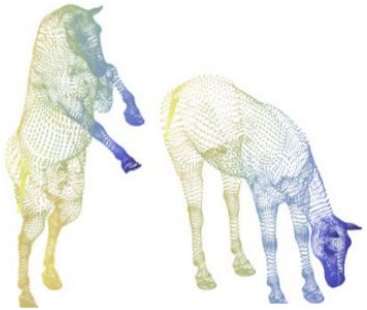
1. MLP
2. LeNet5
3. Network in Network (CNN)

Experimental Setup: Datasets

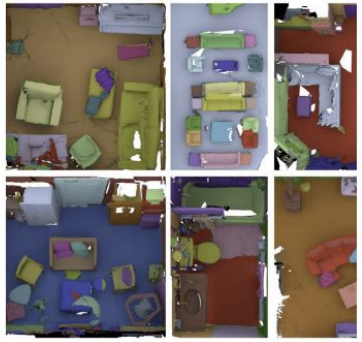
PointNet++ was evaluated on **four datasets** in **various domains**

MNIST
(2D)

ModelNet40
(3D models)



SHREC15
(non-rigid 3D models)



ScanNet
(3D Indoor Scenes)

Complexity

Experiment: **MNIST (2D)**

MNIST images were converted to 2D point clouds of digital pixel locations.
PointNet++ is achieving an accuracy close to Network In Network CNN

MNIST

(2D)

Method	Error rate (%)
Multi-layer perceptron [24]	1.60
LeNet5 [11]	0.80
Network in Network [13]	**0.47**
PointNet (vanilla) [20]	1.30
PointNet [20]	0.78
PointNet++	0.51

Experiment: **ModelNet40**

CAD model for 40 categories for shape classification.

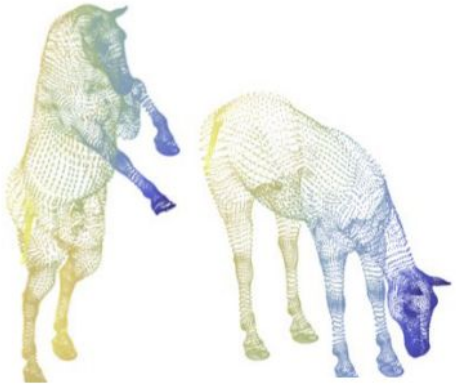
ModelNet40
(3D models)

Method	Input	Accuracy (%)
Subvolume [21]	vox	89.2
MVCNN [26]	img	90.1
PointNet (vanilla) [20]	pc	87.2
PointNet [20]	pc	89.2
PointNet++	pc	90.7
PointNet++ (with normal)	pc	**91.9**

PointNet is significantly more computationally efficient and achieves higher accuracy than voxel and multiview based architectures using only 1024 points!

Experiment: **SHREC15**

Shapes in **SHREC15** are 2D surfaces embedded in 3D space



SHREC15
(non-rigid 3D models)

	Metric space	Input feature	Accuracy (%)
DeepGM [14]	-	Intrinsic features	93.03
PointNet++	Euclidean	XYZ	60.18
	Euclidean	Intrinsic features	94.49
	Non-Euclidean	Intrinsic features	**96.09**

Intrinsic features refer to geometric properties that describe a shape's underlying structure independently of its specific pose. (e.g WKS, HKS)

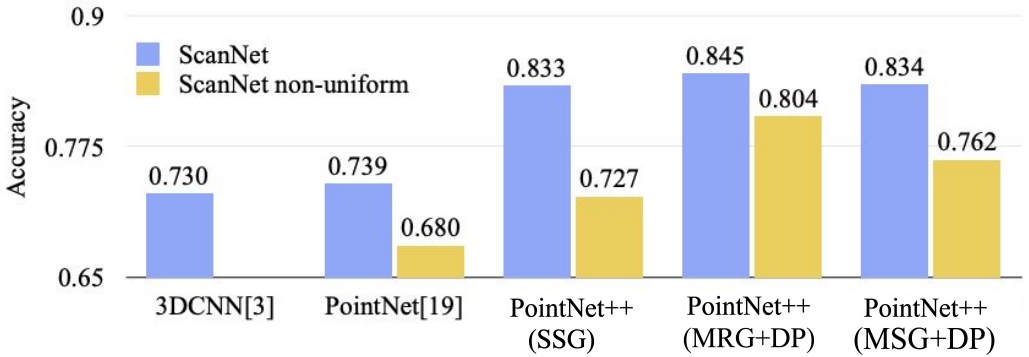
Non Euclidean metric refers to a **Geodesic Metric**, where distance is measured geodesically (along surface of the shape) instead through empty space

Experiment: **ScanNet**

PointNet++ is suitable for large scale point cloud analysis. The goal is to predict semantic object label for points in indoor scans.

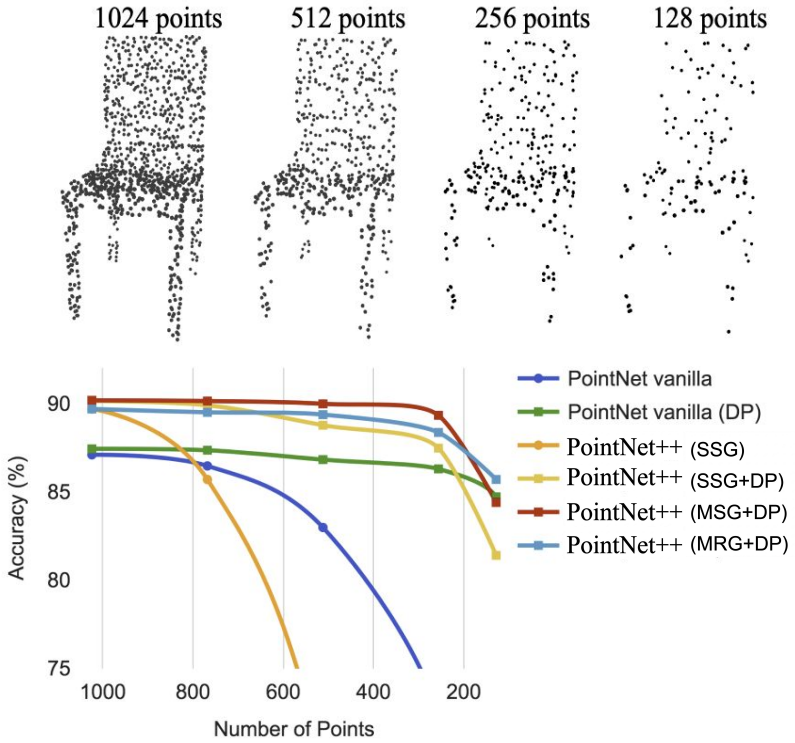
ScanNet

(3D Indoor Scenes)



A key challenge in ScanNet is that points are much denser near the sensor and sparse further away (ScanNet non-uniform)

PointNet++ is robust to **Sampling Density Variation**



Real world sensor data suffers from severe irregular sampling issues.

Randomly drop points during test time to validate the network's robustness to non-uniform and sparse data.

Strengths

Strength 1: Theoretically Soundness

Paper builds on the proof that **PointNet** is a **universal continuous set function approximator:**

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n))$$

1. $f : 2^{\mathbb{R}^N} \to \mathbb{R}$
2. $h : \mathbb{R}^N \to \mathbb{R}^K$
3. $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$

PointNet accurately satisfies ⅔ of the properties of Point Sets:
1. Unordered
2. Invariant to Affine Transformations

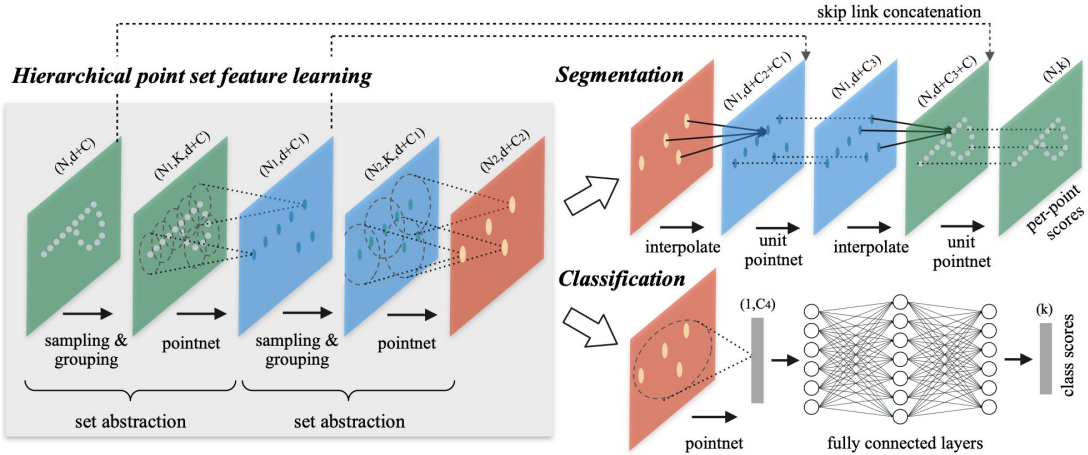
PointNet++ is designed to capture **interaction between points** in a Point Set while preserving **unordered** and **invariance** properties

Strength 2: Architectural Novelty

1. First work to introduce hierarchical structure to point sets, enabling the capture of fine-grained patterns and local context.

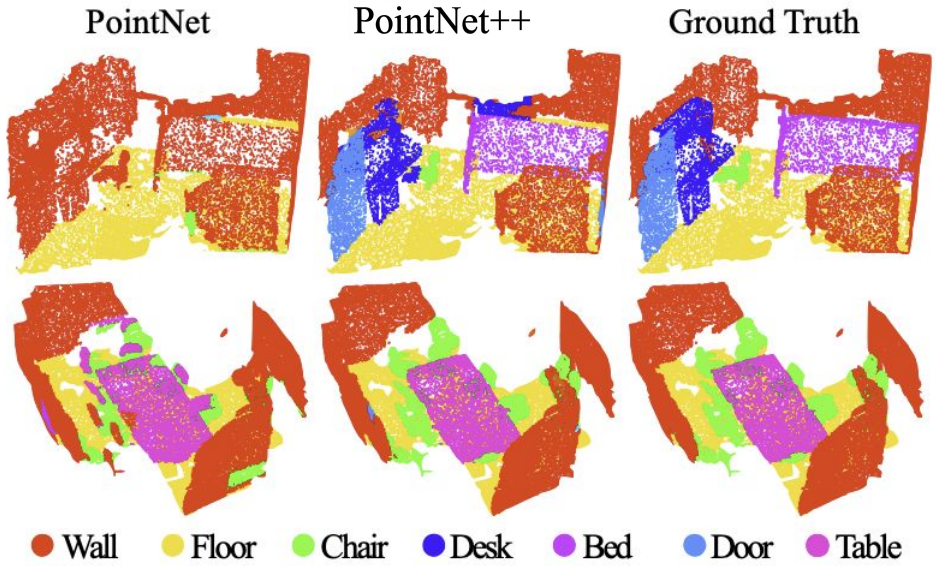
2. The architecture is also **Metric Space Agnostic**: Generalizable beyond 3D Euclidean space

3. Maintains $O(N)$ linear time and space complexity!



Strength 3: Handing Non-Uniformity

The introduction of **density-adaptive layers (MSG/MRG)** provides a robust solution to the **"sampling deficiency"** problem common in real-world LiDAR data.



Weaknesses

Weaknesses and Limitations of PointNet++

1. **Loss of very fine geometry in sparse regions:** Sampling layer uses Farthest Point Sampling (FPS) which might lose very fine geometry in very sparse regions

2. **Manual Hyperparameter tuning:** Choosing the correct radii for Ball Queries and the number of points for each level requires significant empirical tuning.

3. **Latency in Inference:** Multi-Scale Grouping (MSG) and Multi-Resolution Grouping (MRG) are computationally expensive and MSG is 2x slower than Single-Scale Grouping (SSG). Maybe this can be improved?

Discussions

Questions (and **possible extensions**) regarding PointNet++

1. PointNet++ uses **max-pooling** to achieve **permutation invariance** (via PointNet layer), but does this architectural choice **limit the network's ability to learn spatial relationships** between specific points? How might we incorporate 'order' without losing invariance?

2. PointNet++ uses **Farthest Point Sampling (FPS)** to reduce the point count. In extreme cases of noise or outliers, how might **FPS** negatively impact the selection of centroids compared to a uniform volumetric grid?

3. **Integrating Vision-Language Models (VLMs):** Given the irregular and sparse nature of point clouds, could combining geometric features with semantic priors from VLMs improve recognition in cluttered scenes or with heavily occluded objects?

Take Home Message

Why should we care about PointNet++?

1. **Local Context is Essential**

   The leap from PointNet to PointNet++ proves that global signatures are insufficient for complex scenes; local hierarchical abstraction is the "secret sauce" for 3D geometric deep learning.

2. **Point Clouds vs. Voxels**

   Directly processing point sets avoids the cubic complexity ($O(N^3)$) and quantization artifacts of voxels, making it superior for high-resolution scene understanding.

3. **Density Adaptivity is a Requirement**

   In real-world applications (like autonomous driving), a network must be able to switch between fine-grained and coarse-grained features depending on sensor distance.

Thank you!

Q&A / Feedbacks

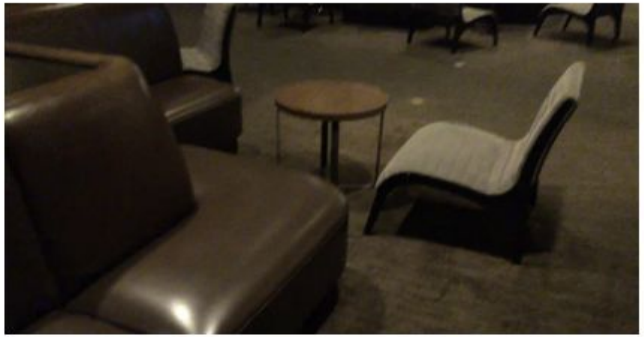
Q&A / Feedbacks

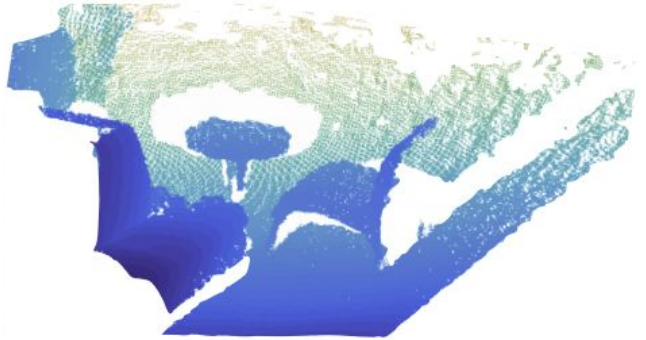
Additional Strengths

What are Point Sets?

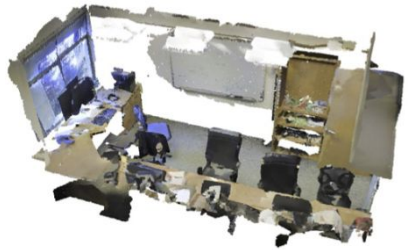
A collection of points $\{P_{i} | i=1,...,n\}$ in a d-dimensional Euclidean space, typically R^3 for 3D applications

Properties:
1. Unordered
3. Transformation Invariance
3. Interaction among points





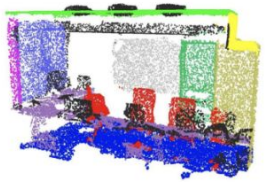
PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation



PointNet

Classification

Part Segmentation

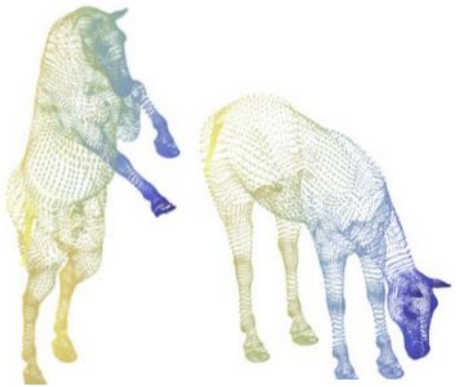


Semantic Segmentation

Point Cloud (Set):

1. (x, y, z) coordinates
2. Color channels (r,g,b)
3. …

Experiment: **SHREC15**



SHREC15
(non-rigid 3D models)

	Metric space	Input feature	Accuracy (%)
DeepGM [14]	-	Intrinsic features	93.03
PointNet++	Euclidean	XYZ	60.18
	Euclidean	Intrinsic features	94.49
	Non-Euclidean	Intrinsic features	**96.09**

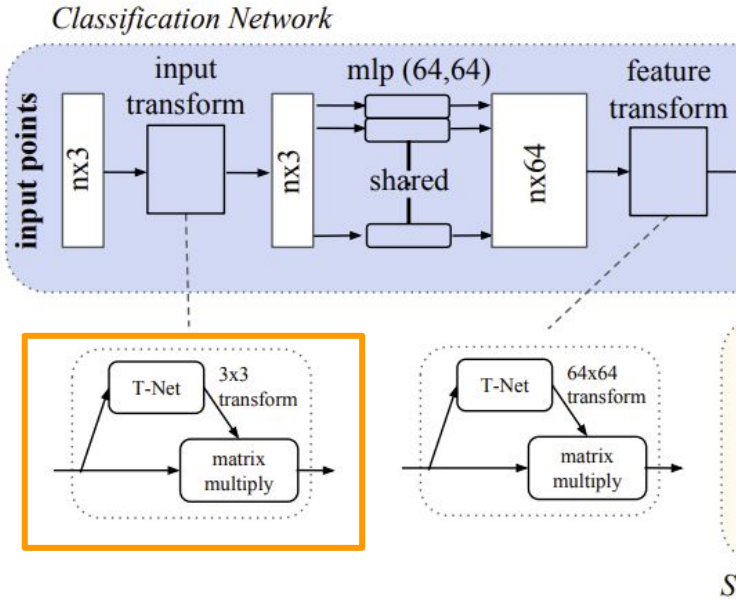
PointNet is Invariant under transformation

Align all **input sets** to a canonical space **before** extracting features

T (Transform)-Net takes **raw coordinates** and predicts a 3x3 transformation matrix

“Mini-PointNet”

1. Shared MLPs
2. Max Pooling
3. Fully connected Layers

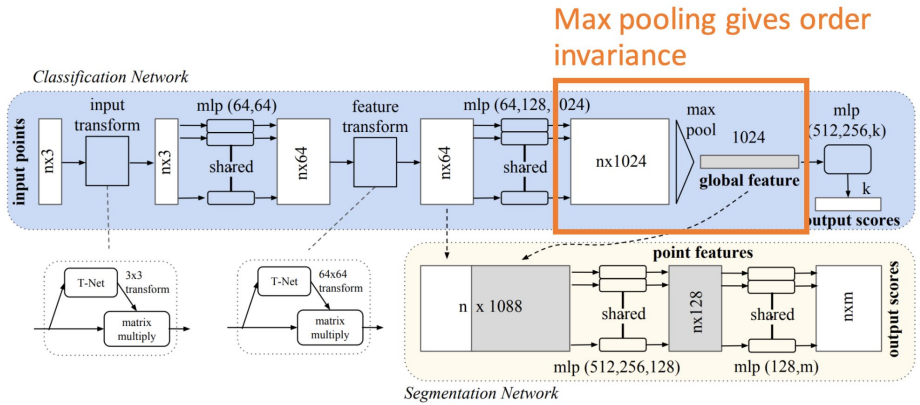


How does PointNet satisfy the properties of Point Sets?

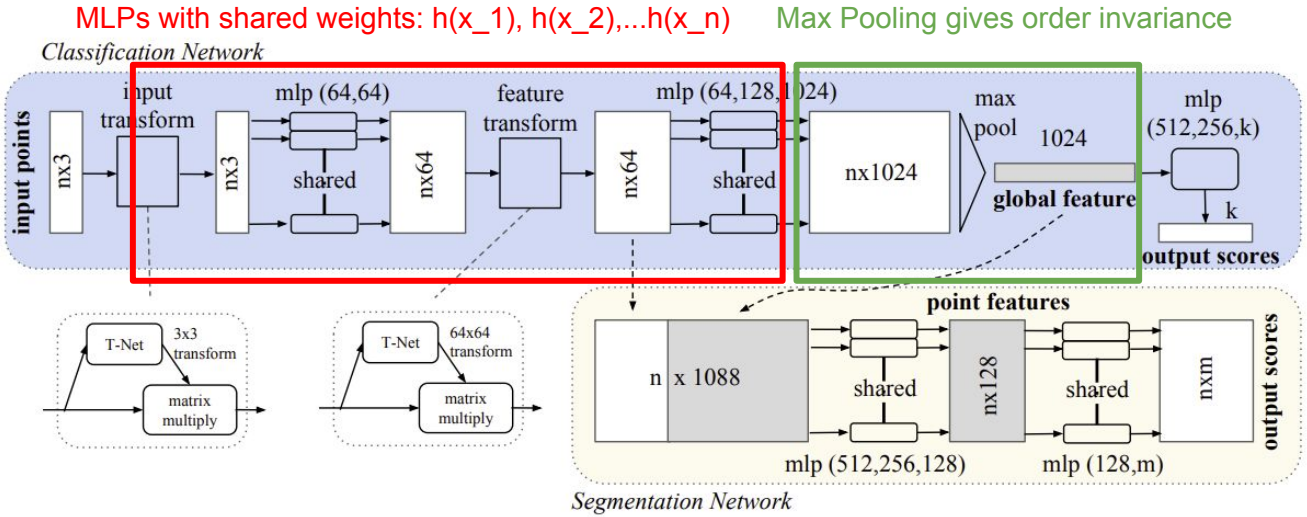
PointNet is invariant to input ordering due to Max Pooling

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)) \qquad f(x_1, x_2, ..., x_n) = \gamma \left(\underset{i=1,\ldots,n}{\text{MAX}} \{h(x_i)\} \right)$$

1. $f : 2^{\mathbb{R}^N} \to \mathbb{R}$
2. $h : \mathbb{R}^N \to \mathbb{R}^K$ **MLPs**
3. $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ **Max Pooling + Global Signature**



PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation



Agenda

1. Challenges for Deep Learning on Point Sets ✓
2. Problem Statement ✓
3. Method
   a. PointNet++ ✓
   b. PointNet, UNet ✓
4. Experiments ✓
5. Summary ✓
6. Q&A / Feedbacks

Problem Statement

Suppose that $\mathcal{X} = (M, d)$ is a discrete metric space whose metric is inherited from a Euclidean space R^d, where $M \subseteq$ R^d is the set of points and d is the distance metric. In addition, the density of M in the ambient Euclidean space may not be uniform everywhere. We are interested in learning set functions f that take such $\mathcal{X}$ as the input (along with additional features for each point) and produce information of semantic interest regrading $\mathcal{X}$. In practice, such f can be classification function that assigns a label to $\mathcal{X}$ or a segmentation function that assigns a per point label to each member of M.

Problem Statement

Suppose that $\mathcal{X} = (M, d)$ is a discrete metric space whose metric is inherited from a Euclidean space R^d, where $M \subseteq$ R^d is the set of points and d is the distance metric. In addition, the density of M in the ambient Euclidean space may not be uniform everywhere. We are interested in learning set functions f that take such $\mathcal{X}$ as the input (along with additional features for each point) and produce information of semantic interest regrading $\mathcal{X}$. In practice, such f can be classification function that assigns a label to $\mathcal{X}$ or a segmentation function that assigns a per point label to each member of M.

Input coord. set **I_x** = {x1,...,xN} in **R^d** + their feature vectors {h1,...,hN} in **R^C**.

A set abstraction level takes an $N \times (d + C)$ matrix as input that is from N points with d-dim coordinates and C-dim point feature. It outputs an $N' \times (d + C')$ matrix of N' subsampled points with d-dim coordinates and new C'-dim feature vectors summarizing local context. We introduce the layers of a set abstraction level in the following paragraphs.

Multi-scale grouping (MSG). As shown in Fig. 3 (a), a simple but effective way to capture multi-scale patterns is to apply grouping layers with different scales followed by according PointNets to extract features of each scale. Features at different scales are concatenated to form a multi-scale feature.

We train the network to learn an optimized strategy to combine the multi-scale features. This is done by randomly dropping out input points with a randomized probability for each instance, which we call *random input dropout*. Specifically, for each training point set, we choose a dropout ratio θ uniformly sampled from $[0, p]$ where $p \leq 1$. For each point, we randomly drop a point with probability θ. In practice we set $p = 0.95$ to avoid generating empty point sets. In doing so we present the network with training sets of various sparsity (induced by θ) and varying uniformity (induced by randomness in dropout). During test, we keep all available points.

Multi-resolution grouping (MRG). The MSG approach above is computationally expensive since it runs local PointNet at large scale neighborhoods for every centroid point. In particular, since the number of centroid points is usually quite large at the lowest level, the time cost is significant.

Here we propose an alternative approach that avoids such expensive computation but still preserves the ability to adaptively aggregate information according to the distributional properties of points. In Fig. 3 (b), features of a region at some level L_i is a concatenation of two vectors. One vector (left in figure) is obtained by summarizing the features at each subregion from the lower level L_{i-1} using the set abstraction level. The other vector (right) is the feature that is obtained by directly processing all raw points in the local region using a single PointNet.

Problem Statement

Suppose that $\mathcal{X} = (M, d)$ is a discrete metric space whose metric is inherited from a Euclidean space R^d, where $M \subseteq$ R^d is the set of points and d is the distance metric. In addition, the density of M in the ambient Euclidean space may not be uniform everywhere. We are interested in learning set functions f that take such $\mathcal{X}$ as the input (along with additional features for each point) and produce information of semantic interest regrading $\mathcal{X}$. In practice, such f can be classification function that assigns a label to $\mathcal{X}$ or a segmentation function that assigns a per point label to each member of M.

Input coord. set **I_x** = {x1,...,xN} in **R^d** + their feature vectors {h1,...,hN} in **R^C**.

Output coord. set **O_x** = {xi1,...,xiN'} is a subset of **I_x** + their feature vectors {h'i1,...,h'iN'} in **R^C'**.

A set abstraction level takes an $N \times (d + C)$ matrix as input that is from N points with d-dim coordinates and C-dim point feature. It outputs an $N' \times (d + C')$ matrix of N' subsampled points with d-dim coordinates and new C'-dim feature vectors summarizing local context. We introduce the layers of a set abstraction level in the following paragraphs.