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

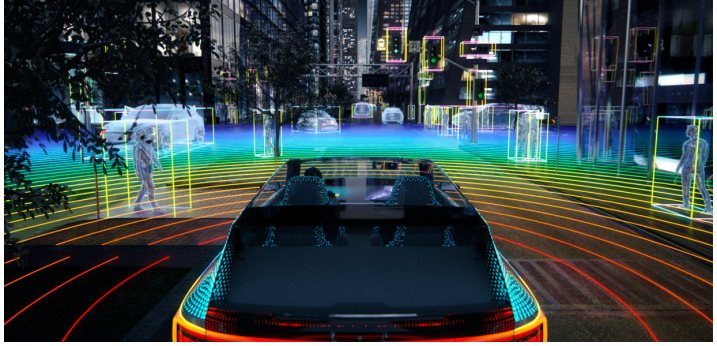
Presenter: Liangchen Liu

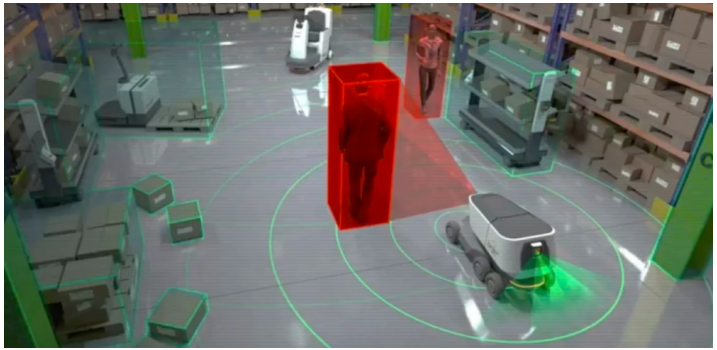
2021/09/07

Why 3D deep learning?

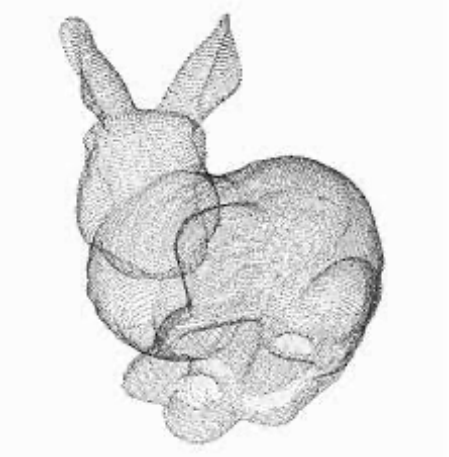
- **Perception of the environment** (Robot Vision)
  - Self-driving cars
  - Autonomous robots

- **Operations on 3D objects** (Less Robotics) (Modelling, registration, classification, etc.)
  - AR
  - Interior design





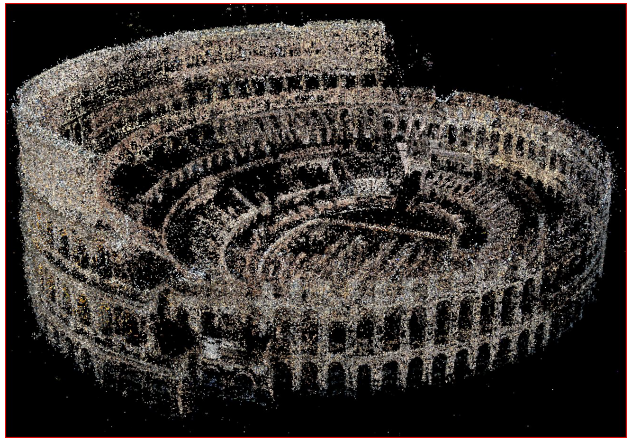
Why Point Sets (Point Clouds)?



- Raw outputs from 3D scanners
  - No quantization loss (mesh, voxel grid, etc.)
  - No artifacts (rendering, projection, etc.)

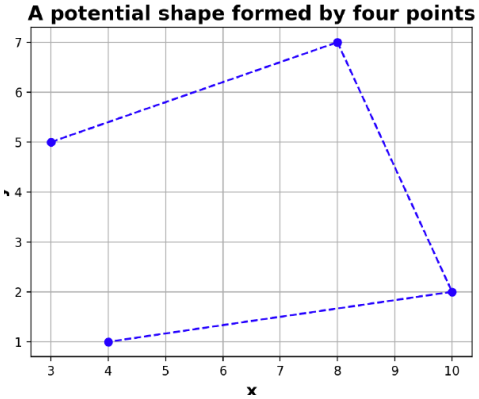
- 3D scanners become widely accessible
  (e.g. LiDAR; Kinect, Tango, any RGB-D type…)

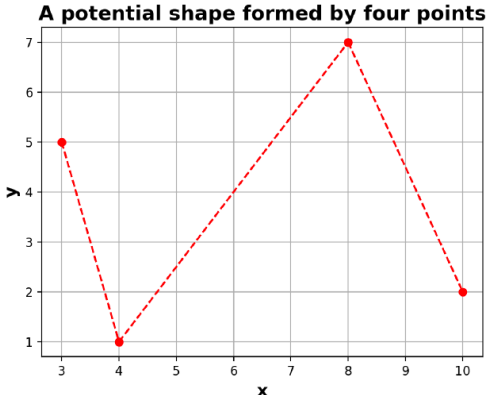
- In data science, the point cloud is a representation of data in high dimension (less relevant)

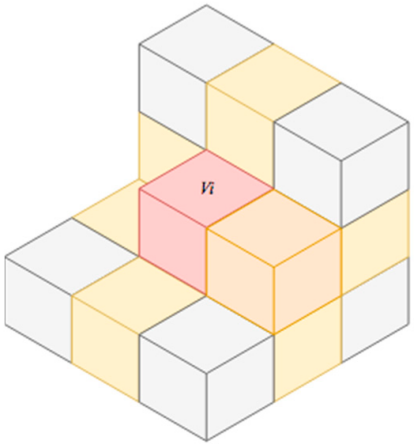


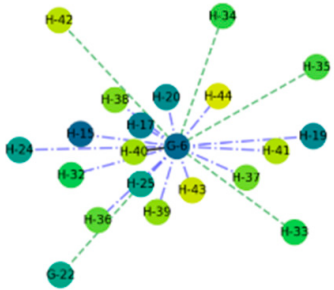
Notorious features of Point Clouds

- **Ambiguous Connectivity**
- Lack of Computation Structure
  - c.f. Mesh, Voxel, Graph
- Does not have an order
  - Where to start?
  - Where to go next?









Key insights of PointNet ++

1. Develop a framework that is invariant to the input order (from PointNet) (Symmetric operations on points, such as mean(), max(), …)

2. Define the concept of neighborhood to reproduce convolution-like techniques (Local feature extraction, hierarchical abstraction)

3. Use novel techniques to deal with non-uniform density.

Problem: deep learning on 3D point clouds

- The point cloud: $M \subseteq \mathbb{R}^d$ $(d = 3)$, a set of discrete points. (Assume $|M| = n$)
- Equipping with the normal Euclidean metric d: $\mathcal{X} = (M, d)$, a discrete metric space.
- Goal: learn the set function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{Y}$ depends on tasks:
  - Classification: $\mathcal{Y} = \{1, \ldots, m\}$ (per object labeling)
  - Segmentation: $\mathcal{Y} = \{1, \ldots, m\}^n$ (per point labeling)

Related Work

Traditional Machine Learning: Handcrafted feature descriptors (really hard)

HKS, WKS, PFH, FPFH (only for your reference…) Also “3D deep shape descriptors”.

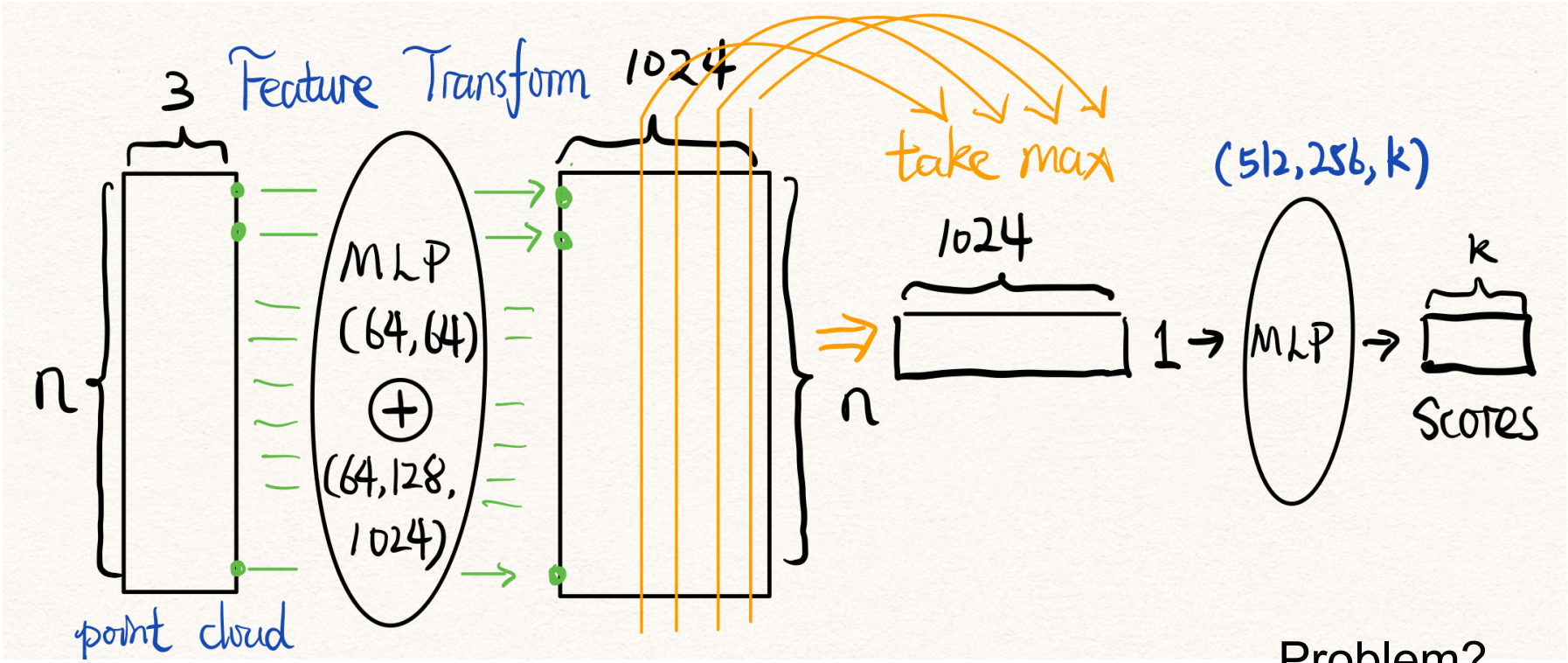
3D deep learning on other representations:

- Voxel grids: VoxNets, ShapeNets, Volumetric CNNs…
  Problem: resolution, computational cost, non-uniform density

- Mesh: Spectral CNNs, 3D mesh Labeling, Geodesic CNNs…
  Problem: constrained by mesh modelling and rendering

- Projected Images / Stereotypical Projections: MultiView CNNs, Rotation Nets…
  Problem: dominance in classification, but non-trivial to segmentation and complicated 3D scenes

Related Work: PointNet (the ancestry)



A global feature extractor

Proposed Method: PointNet++

Key Idea: Learn local features based on PointNet

Key Approach: Use PointNet recursively on small neighborhood to extract local feature

Three repeated steps: (Set Abstractions). Input shape: $N \times (d + C)$

1. **Sampling Layer**

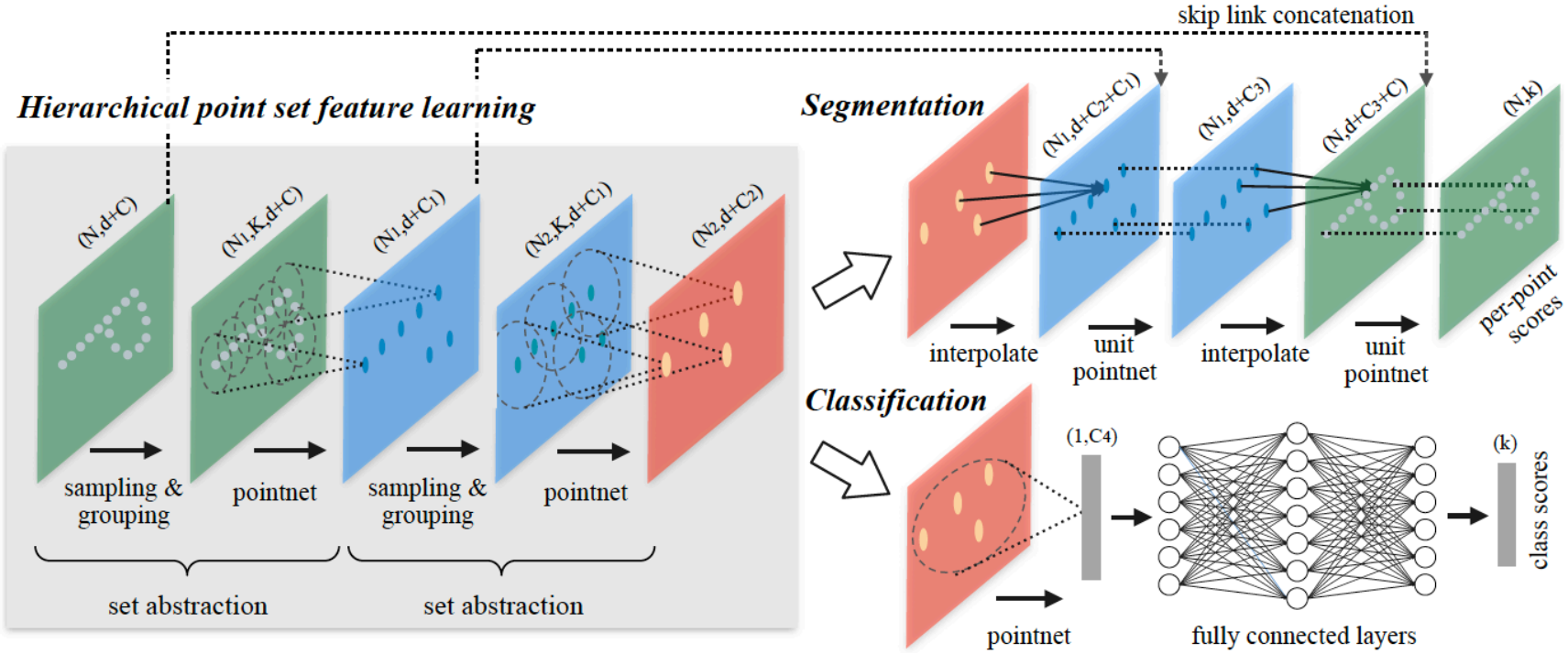
   Farthest Point Sampling (FPS): pick N' points that are most distant from the rest of the point sets recursively as clustering center (better coverage than random)

2. **Grouping Layer**

   Ball query (preferred) or kNN. Output shape: $N' \times K_{x_i} \times (d + C)$

3. **PointNet:** Apply to each local neighborhood; Output shape: $N' \times (d + C')$

Proposed Method: PointNet++



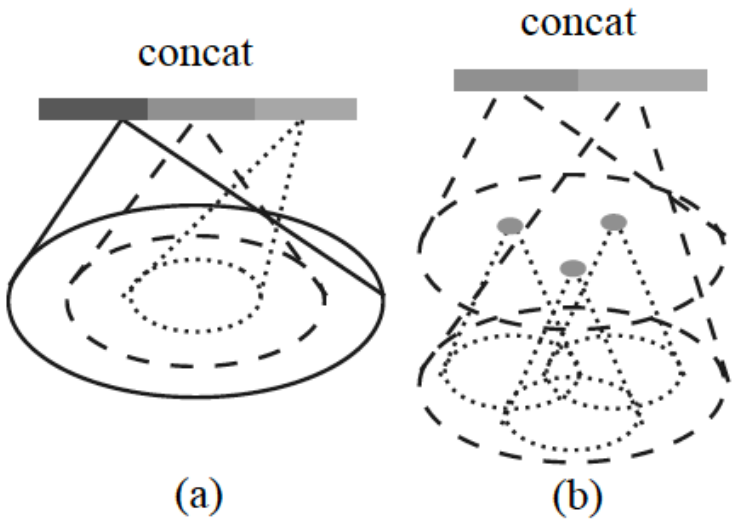
Foundation: PointNet abstraction ability

Non-Uniform Density

- Assumption: the density of the point cloud is not uniform (quite reasonable)

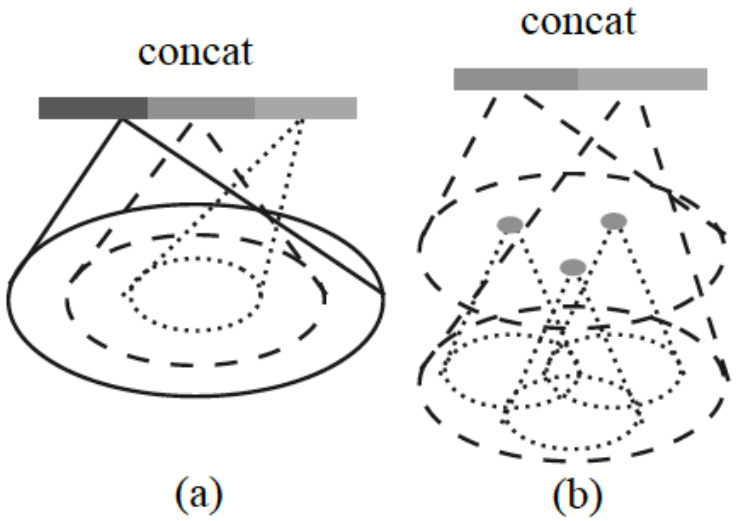
- In training, randomly dropping out points to simulate the non-uniform density:
  For each point cloud M_i, choose a random value θ_i
  uniformly from $[0, 0.95]$. For each $x \in M_i$, drop the point
  with probability θ_i

- Method (a) (MSG: Multi-Scale Grouping) :
  - For each center, concatenate features generated from different scales.
  - Problem: high cost



Non-Uniform Density

- Method (b) (MRG: Multi-Resolution Grouping):
  - Concatenate abstracted feature from the current level and the original level.
  - Features:
    - Original level guarantees density
    - The higher level guarantees finer details
    - Computational efficient
    - But less effective in practice than (a)



PointNet Proof Sketch (if necessary)

(Might drop it due to time constraint)

Experimental Setup

- Tasks: Classification & Semantic Scene Labeling

- Datasets: MNIST (2D images -> 2D point clouds). Baseline: MLP (for classification)
  ModelNet40 (3D rigid objects). Baseline: PointNet (for classification)
  SHREC15 (3D non-rigid objects). Baseline: Rigid classification
  ScanNet (real 3D scenes) Baseline: ScanNet (for scene labeling)

Classification Results

Metric: Accuracy

512 Points

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
Ours	0.51

Table 1: MNIST digit classification.

1024 Points

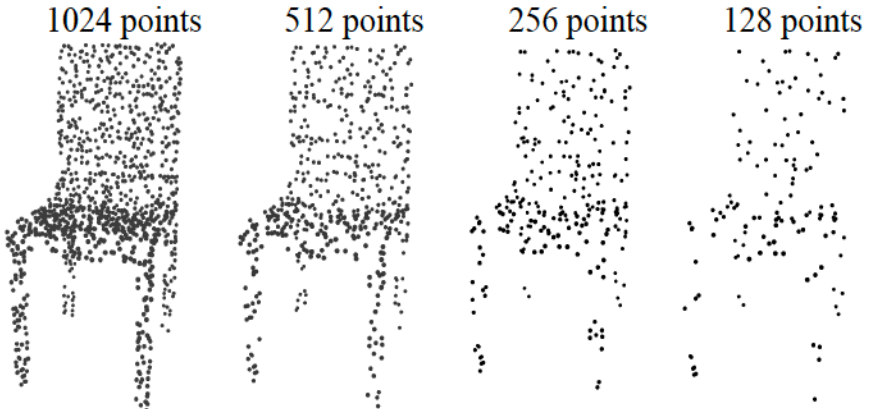
Method	Input	Accuracy (%)
Subvolume [21]	vox	89.2
MVCNN [26]	img	90.1
PointNet (vanilla) [20]	pc	87.2
PointNet [20]	pc	89.2
Ours	pc	90.7
Ours (with normal)	pc	**91.9**

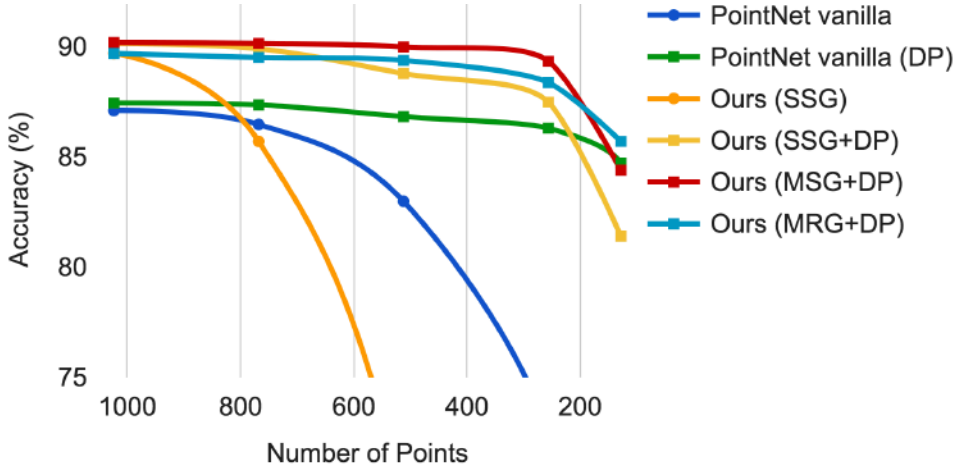
Table 2: ModelNet40 shape classification.

5000 Points !

Classification Results

Robustness to Sampling Density





SSG: Single Scale Grouping
DP: Trained with random dropping points

Scene Labeling Results

Metric: Average Voxel Classification Accuracy

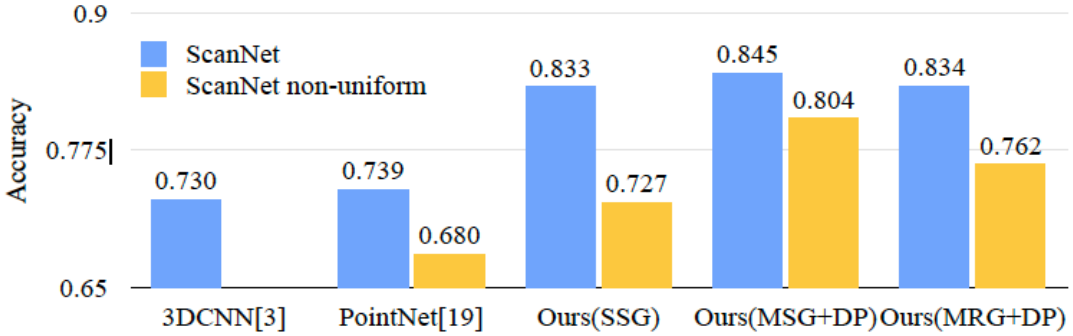
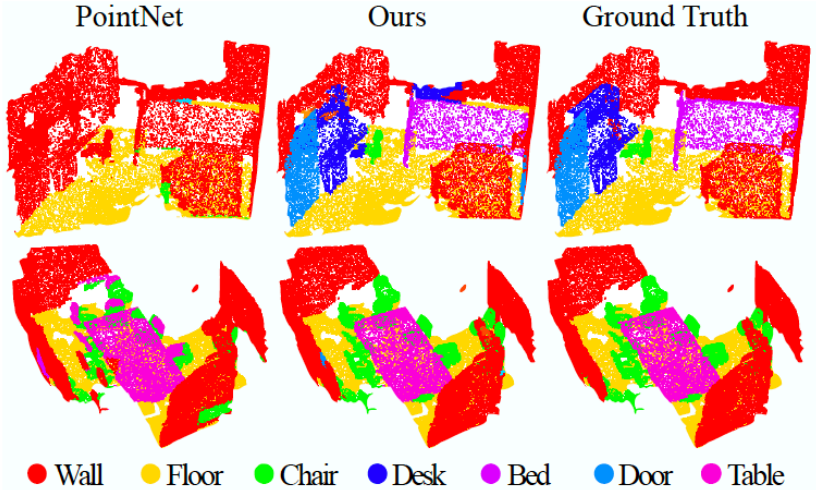


Figure 5: Scannet labeling accuracy.



(Baseline)

Non-Rigid Shape Classification

(a) Horse (b) Cat (c) Horse

Figure 7: An example of non-rigid shape classification.

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

Table 3: SHREC15 Non-rigid shape classification.

Non-Euclidean means using Geodesic distance for neighbouring

Discussion of Results

- PointNet++ is a **powerful framework** for deep learning in point clouds.

- By suitable training tricks, PointNet++ is **robust to non-uniform density**.

Are the stated conclusions fully backed by the results and references?

- PointNet++ indeed works well, **state-of-the-art performance** in standard benchmark (back then).

- Not convinced by robustness to sampling density. Should have PointNet++ (MSG/MRG) trained without random dropouts.

- **Didn't include computation complexity comparison for this algorithm!**

Critique / Limitations

1. PointNet++ could be expensive in practice (apply PointNet to each local neighborhood)

   —— not good for real time robotic vision

2. Didn’t fully exploit the neighbouring information (e.g. distance to neighbours)

3. When grouping in next level, can consider neighbors in feature space instead of just Euclidean.

(See DGCNN in the next presentation)

Extended Readings

1. Dynamic Graph CNN for Learning on Point Clouds. Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, Justin M. Solomon (2018)

2. PointCNN: Convolution On X-Transformed Points. Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, Baoquan Chen (2018)

3. ShellNet: Efficient Point Cloud Convolutional Neural Networks using Concentric Shells Statistics. Zhiyuan Zhang, Binh-Son Hua, Sai-Kit Yeung (2019)

4. A-cnn: Annularly convolutional neural networks on point clouds. Artem Komarichev, Zichun Zhong, Jing Hua (2019)

Summary

- Problem: 3D deep learning directly on point clouds
- Significance: 3D deep learning has extensive applications; point clouds are tricky
- Key limitation of prior work: Can't apply on point clouds; PointNet didn't exploit local information
- Key insight(s) of the proposed work:
  1. Use ball-query to define local neighborhood and exploit local feature
  2. Use different grouping techniques to deal with non-uniform density
- Achieve state of the art performance on benchmark dataset include ModelNet40, SHREC15, ScanNet