Unsupervised Point-Cloud Reconstruction

Luca Barco *s276072@studenti.polito.it*
Stefano Bergia *s276124@studenti.polito.it*
Daniela De Angelis *s277493@studenti.polito.it*

Image: polito.it

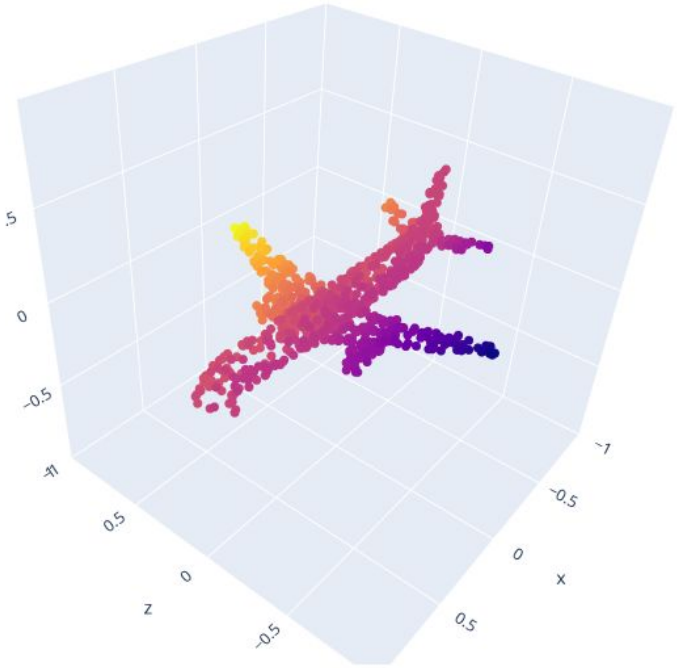
Politecnico di Torino
Machine Learning and Artificial Intelligence

A.Y. 2020/2021

AI ML

What is a **Point-Cloud**?

- Set of points
- Unordered and unstructured
- Represents an object
- Close to sensor data (e.g. LiDAR Camera)



Cannot directly use traditional CNN Models

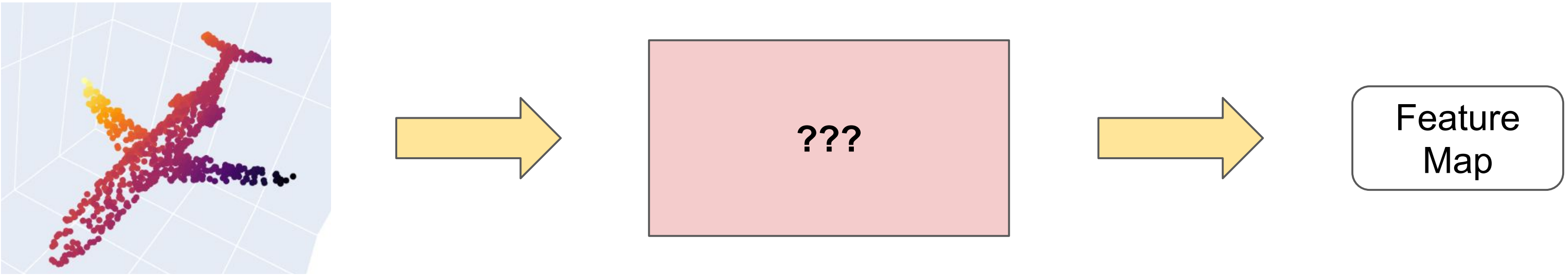
How deep learning methods deal with **Point-Clouds**?

To deal with Point-Clouds, deep learning methods must guarantee:

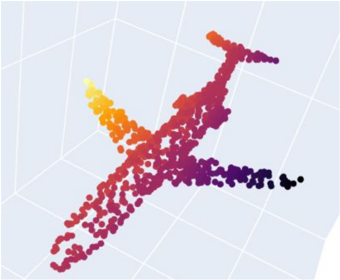
- **Permutation Invariance**: invariant to $N!$ permutations of the input.
- **Geometric Transformations Invariance**: Rigid transformation (e.g. rotation) applied to the input should not alter the performed task results.

Build a Point-Cloud Auto-Encoder

How to **extract information** from a Point-Cloud?



How to **extract information** from a Point-Cloud?



We need an **unsupervised** method in order to extract features…

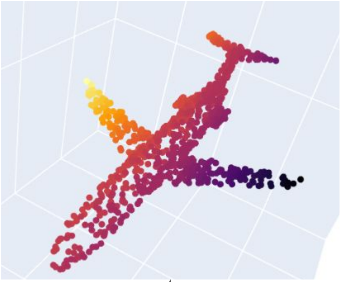
... we can train an **AUTO-ENCODER!**

???

Feature Map

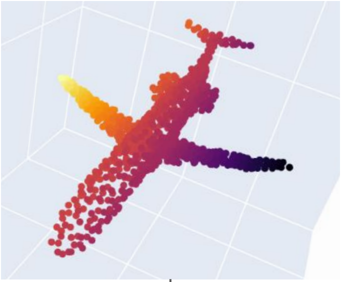
Our problem: train a Point-Cloud Auto-Encoder

Input Point-Cloud



AUTO - ENCODER

Reconstructed Point-Cloud



The reconstructed output should be as close as possible to the original input

Starting point: PointNet and DGCNN

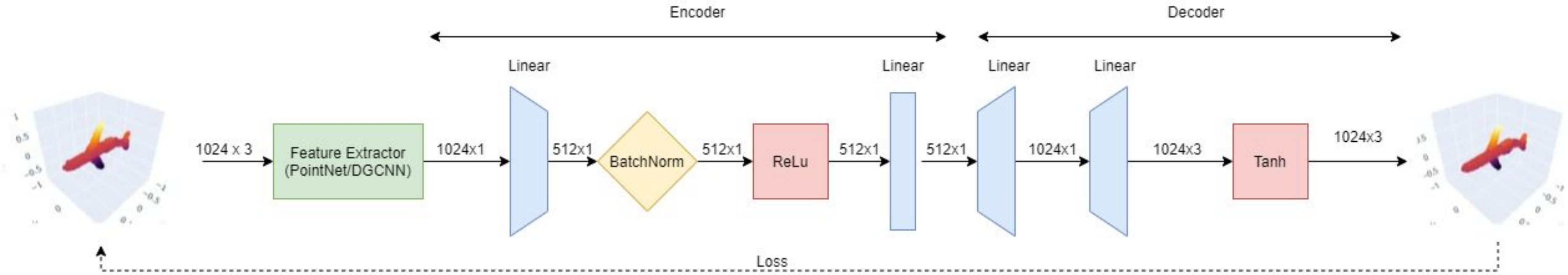
PointNet[1] and DGCNN[2] are **supervised** deep learning method for Point-Clouds classification and segmentation.

They extract the main features of a Point-Cloud

Use their features extractors in an **unsupervised** Auto-Encoder

[1] Qi et Al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, in CVPR 2017
[2] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. CoRR, abs/1801.07829, 2018.

Auto-Encoder’s Architecture



Main components:

- **Feature Extractor** (based on PointNet[1] or DGCNN[2])
- **Encoder**
- **Fully Connected Decoder**
- Loss function based on **Chamfer Distance**

[1] Qi et Al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, in CVPR 2017
[2] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. CoRR, abs/1801.07829, 2018.

Auto-Encoder's Architecture: PointNet[1] Feature Extractor

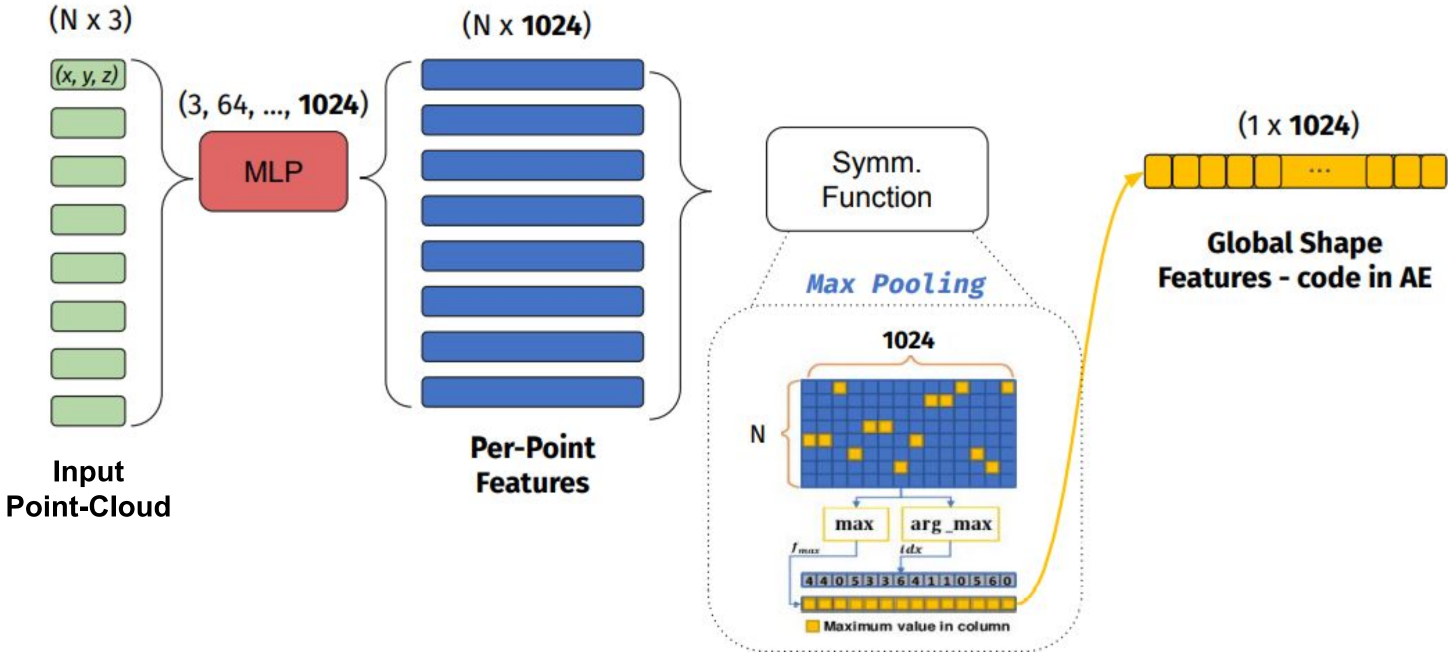
PointNet[1]: deep learning method for Point-Clouds classification and segmentation.

- **Permutation Invariance** : guaranteed using a symmetric function to aggregate information from each of the points (e.g. Max Pooling).
- **Geometric Transformations Invariance**: guaranteed using Transformer Networks (T-Net) in euclidean space.
- It aggregates **global features** into a unique vector

[1] Qi et Al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, in CVPR 2017

Auto-Encoder's Architecture: PointNet[1] Feature Extractor

PointNet[1] Architecture: Features extraction



[1] Qi et Al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, in CVPR 2017

Auto-Encoder's Architecture: DGCNN[2] Feature Extractor

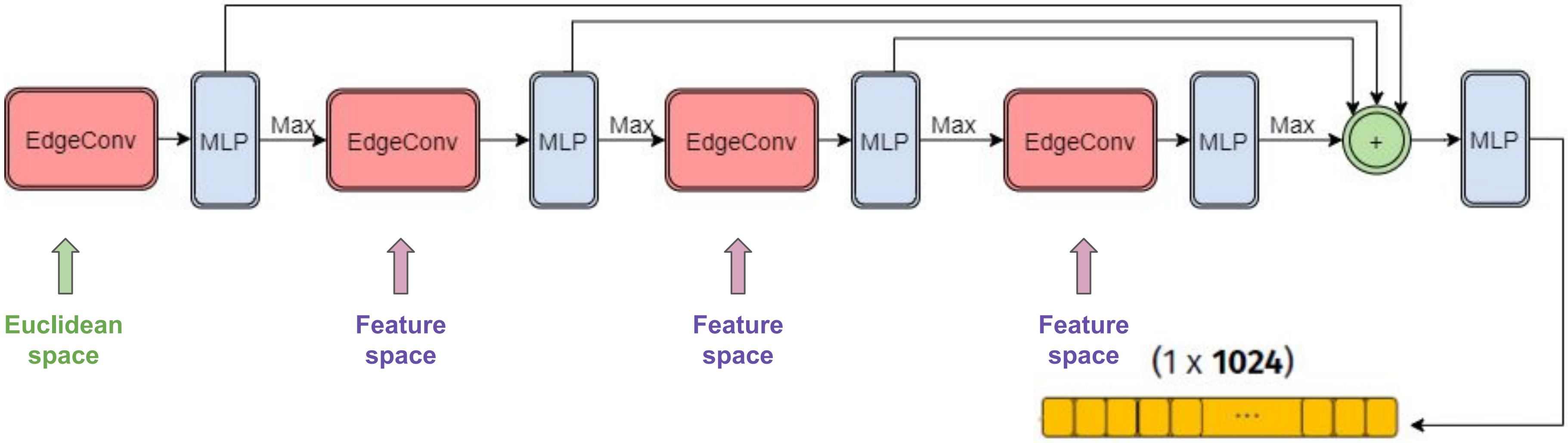
DGCNN[2]: deep learning method for Point-Clouds classification and segmentation based on computation of direct graph representing local structures.

- **Permutation Invariance** : guaranteed using a symmetric function (e.g. Max Pooling) to aggregate information at each layer.
- **Geometric Transformations Invariance**: guaranteed using the neighbourhood graph.
- It aggregates both **global and local** features into a unique vector

[2] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. CoRR, abs/1801.07829, 2018.

Auto-Encoder's Architecture: DGCNN[2] Feature Extractor

- **Dynamic Graph:** the graph is updated at each layer
- **EdgeConv**: convolutional layer working on the dynamic graph



[2] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. CoRR, abs/1801.07829, 2018.

Auto-Encoder's Architecture: DGCNN[2] Graph Computation

How is the graph computed?

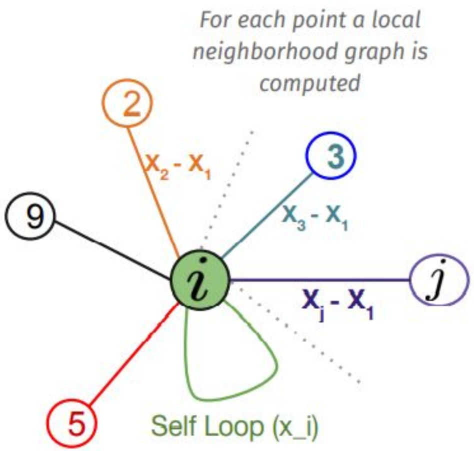
- Using an **Edge function:** asymmetric function that considers each point x_i and its neighbourhood (x_j-x_i)

$$h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j) = \bar{h}_{\Theta}(\boxed{\mathbf{x}_i}, \boxed{\mathbf{x}_j - \mathbf{x}_i}).$$

Θ : Learnable Parameters

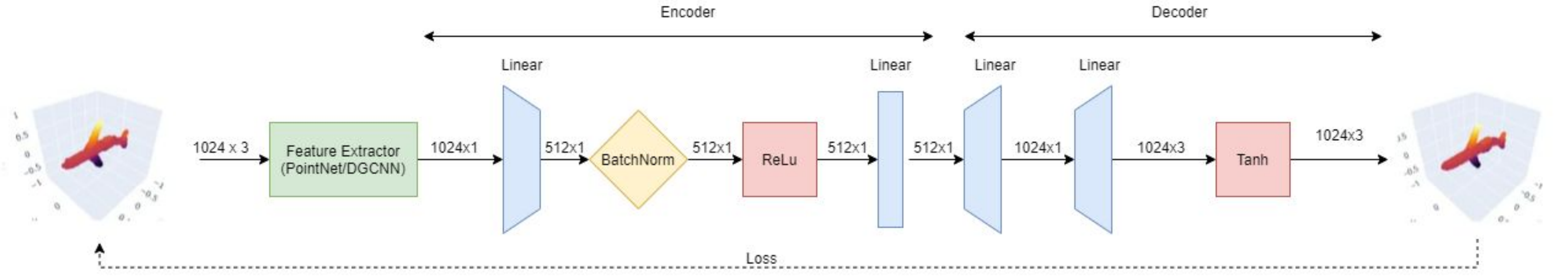
Global Shape

Local Neighborhood Structure



[2] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. CoRR, abs/1801.07829, 2018.

Auto-Encoder's Architecture: Encoder and Decoder



- **Encoder:** Further refinement of features of the input Point-Cloud
- **Decoder:** Fully-Connected layers to reconstruct the input Point-Cloud

Auto-Encoder's Architecture: Loss Function

Distance metric: **Chamfer Distance**[3]

Given two point sets P_1, P_2 →

$$d_{CD}(P_1, P_2) = \underbrace{\sum_{x \in P_1} \min_{y \in P_2} ||x - y||_2^2}_{P_1 \to P_2} + \underbrace{\sum_{y \in P_2} \min_{x \in P_1} ||x - y||_2^2}_{P_2 \to P_1}$$

- *"pseudo-symmetrical" behaviour:*
  - For each point in P_1, compute the squared distance with respect to the nearest point in P_2
  - The same for each point in P_2
  - Sum these two components

Loss Function: average of the Chamfer Distance over N points of Point-clouds P_1 and P_2 multiplied by a *w* factor (in our work w=100)

$$L(P_1, P_2) = \frac{w}{2N} \cdot d_{CD}(P_1, P_2)$$

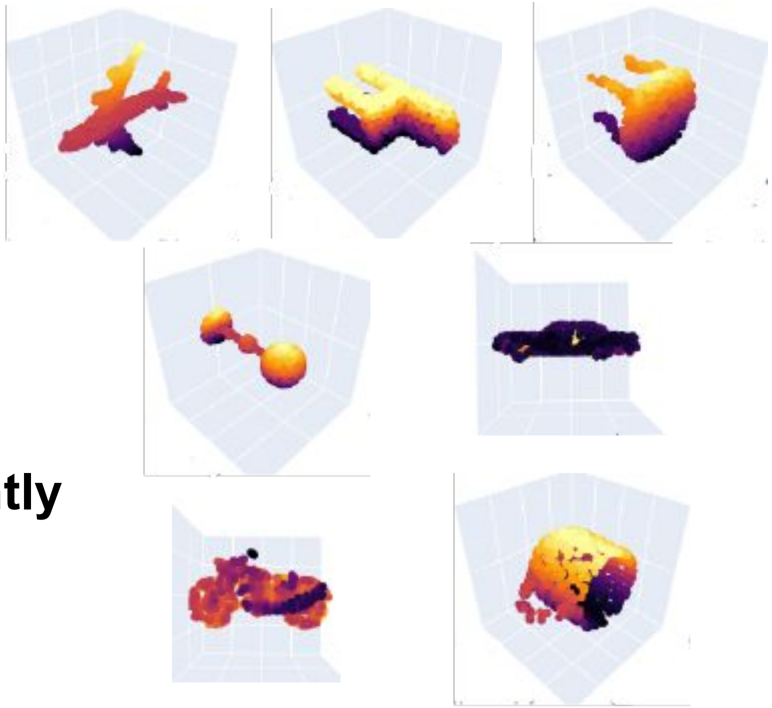
[3] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. CoRR, abs/1612.00603, 2016.

Auto-Encoder: Training

- Training Dataset: **ShapeNet[4]**
- Training on seven classes of data:

  Airplane, Chair, Table, Lamp,

  Car, Motorbike, Mug.
- **Two scenarios:**
  - **Training on all seven classes jointly**
  - **Training on single classes**



[4] X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. CoRR, abs/1512.03012, 2015

Auto-Encoder: Training on all 7-classes vs single

- Two Architectures:
  - **PNet_AE_512** : PointNet based AE, Encoder lower level size: 512
  - **DGCNN_AE_512**: DGCNN based AE, Encoder lower level size: 512

Training on all seven classes jointly:
Testing loss results

Category	PNet_AE_512	DGCNN_AE_512
Airplane	**0.100**	0.105
Chair	0.191	**0.190**
Table	**0.207**	0.219
Lamp	0.301	**0.272**
Car	**0.245**	0.256
Motorbike	**0.186**	0.193
Mug	0.381	**0.364**
Avg	*0.230*	***0.228***

Training on single classes:
Testing loss results

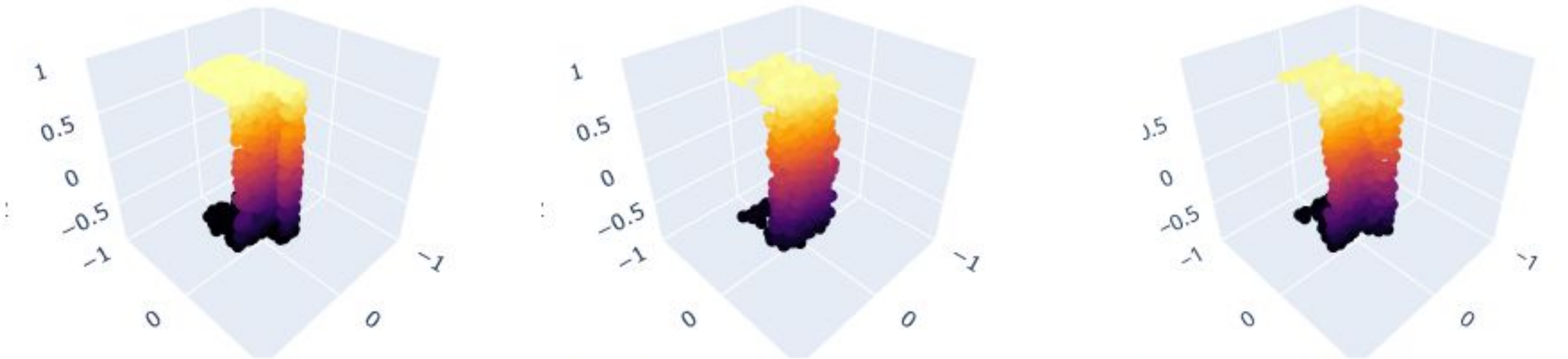
Category	PtNet_AE_512	DGCNN_AE_512
Airplane	0,113	**0,099**
Chair	0,221	**0,197**
Table	**0,227**	0,228
Lamp	**0,387**	0,499
Car	0,294	**0,254**
Motorbike	**0,225**	0,236
Mug	**0,582**	0,585
Avg	***0,293***	*0,300*

Auto-Encoder: Training on all 7-classes vs single

'image: Flaticon.com'

Regardless of the architecture, results look better when training on all 7-classes jointly!

'image: Flaticon.com'



Original | Training on single classes | Training on 7-classes jointly

Point-Cloud contains uncommon details → Loss of details: the model learns an *average shape* → Features learned on one category can be used to reconstruct others

Auto-Encoder: Testing on novel categories

Testing the models trained on all seven classes jointly on novel unseen categories:

- **Similar**: *Basket, Bicycle, Bowl, Helmet, Microphone, Rifle, Watercraft*
- **Dissimilar**: *Bookshelf, Bottle, clock, Microwave, Pianoforte, Telephone*

Testing on novel categories: Testing loss results

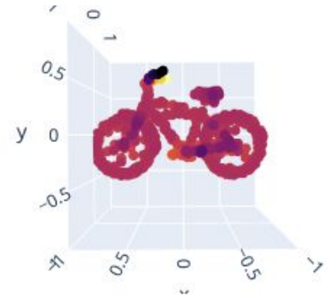
	Category	**PNet_AE_512**	**DGCNN_AE_512**
Similar	Basket	0.711	**0.606**
	Bicycle	0.408	**0.399**
	Bowl	1.072	**0.747**
	Helmet	0.902	**0.732**
	Microphone	1.635	**0.694**
	Rifle	0.201	**0.197**
	Watercraft	0.261	**0.259**
	Avg	*0.741*	***0.516***
Dissimilar	Bookshelf	0.576	**0.551**
	Bottle	0.330	**0.307**
	Clock	0.702	**0.562**
	Microwave	**0.494**	0.517
	Pianoforte	0.732	**0.631**
	Telephone	0.494	**0.424**
	Avg	*0.584*	***0.499***

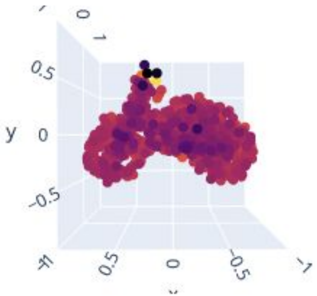
'image: Flaticon.com'

Not much differences between similar and dissimilar categories!

Auto-Encoder: Examples of outputs

Similar:

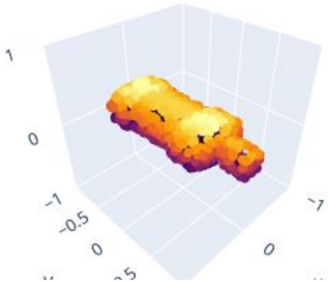


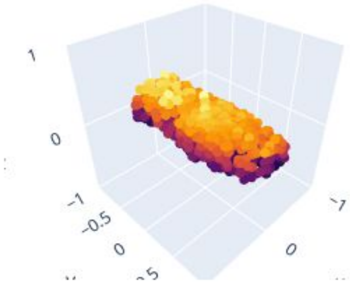


'image: Flaticon.com'

Uses Motorbike features!

Dissimilar:





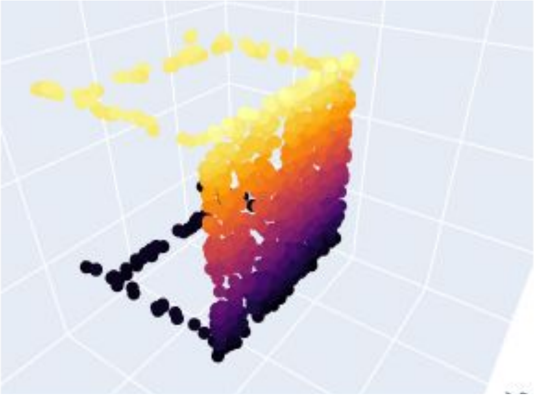
'image: Flaticon.com'

Uses Car Features!

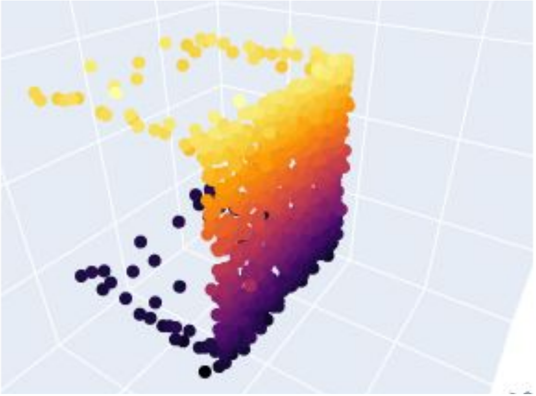
Auto-Encoder Loss Function issue: local density

Problem: it doesn't preserve local density → Loss of finer details

Input Point-Cloud



Reconstructed Point-Cloud



Lower densities for legs and bars, higher density for the plane → **Errors in low density regions are penalized much more** → **Few points are assigned to low density regions**

Auto-Encoder: Pros, cons and improvements

PROS

- ✓ Good at reconstructing general shape of the object
- ✓ Good at “transfer” local features learned for an object to reconstruct another one
- ✓ Small embedding size

CONS

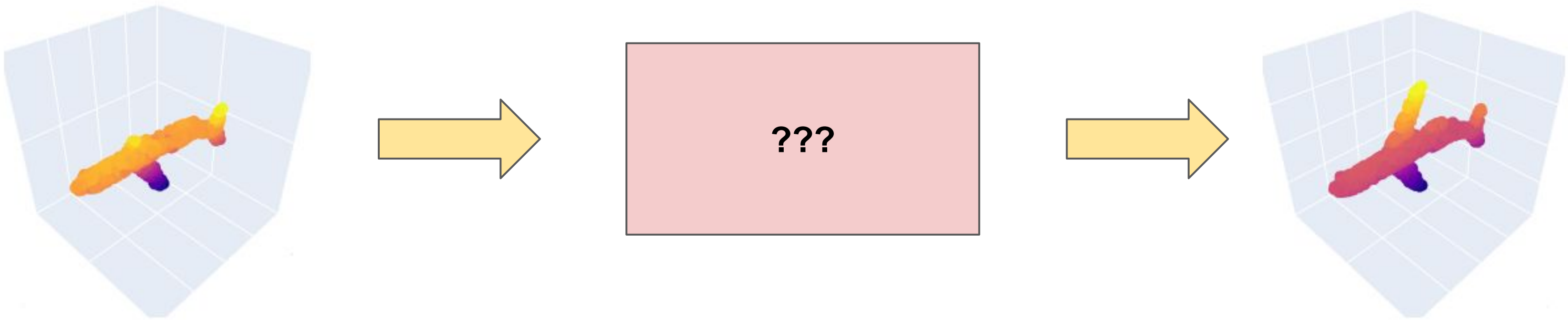
- ❌ Bad at reconstructing finer details
- ❌ Bad at preserving local density

IMPROVEMENTS

- 💡 Adjust the loss function to preserve density and better reconstruct finer details

Point-Cloud Completion: Complete a cropped Point-Cloud

How to **complete** a cropped Point-Cloud?



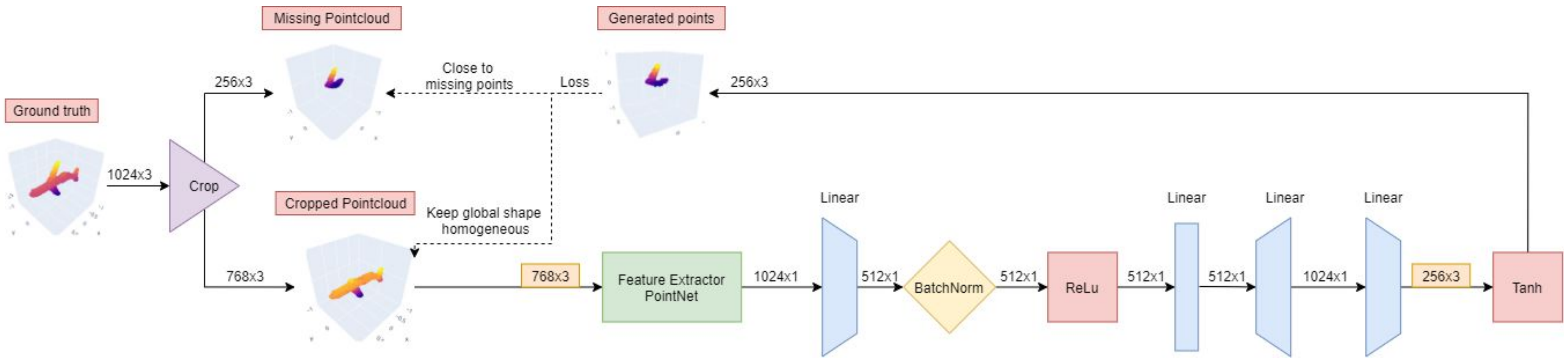
Starting point: PF-Net[4]

PF-Net[4] is a deep learning method to reconstruct 3-D objects considering only the missing part

Use their idea with **our Auto-Encoder**

[4] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le. Pf-net: Point fractal network for 3d point cloud completion, 2020.

How to **complete** a cropped Point-Cloud?
Our proposal



Key concepts:

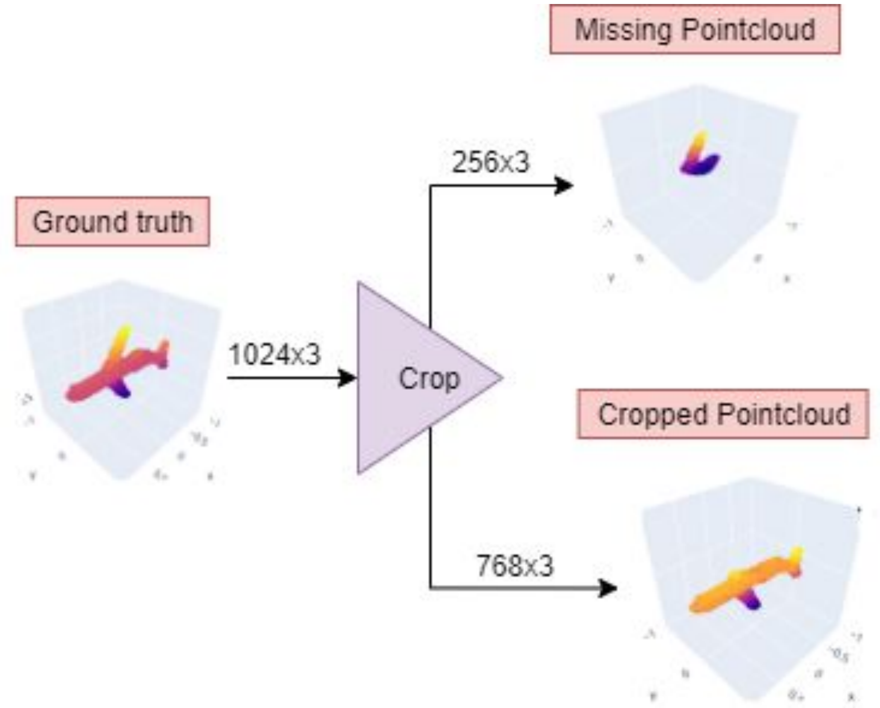
- **Point-Cloud cropping** → reconstruct only the Missing Part
- **PointNet based Auto-Encoder**
  - **input**: Cropped Point-Cloud
  - **output**: Reconstructed Missing Part Point-Cloud
- **Final result:** Reconstructed Missing Part + Cropped → **complete object Point-Cloud**
- **Objective**: good reconstruction of Missing Part + keep global object shape homogeneous

Point-cloud completion: Point-Cloud cropping

- Set of **viewpoints**
- **For each one**

  Remove the 256 nearest points

1) **Cropped Point-Cloud**
2) **Missing Part Point-Cloud**



Point-cloud completion: Loss function

- **P_{gt}**: Ground Truth Point-Cloud
- **P_m**: Missing-Part Point-Cloud
- **P_c**: Cropped Point-Cloud
- **P_{rm}**: Reconstructed Missing-Part Point-Cloud
- **$P_r = P_c + P_{rm}$**: Reconstructed Point-Cloud given by concatenation of Prm and Pc.
- **$L(P_1; P_2)$** as the mean Chamfer Distance between Point-Clouds P1 and P2.
- **r**: weighting parameters to balance the two terms

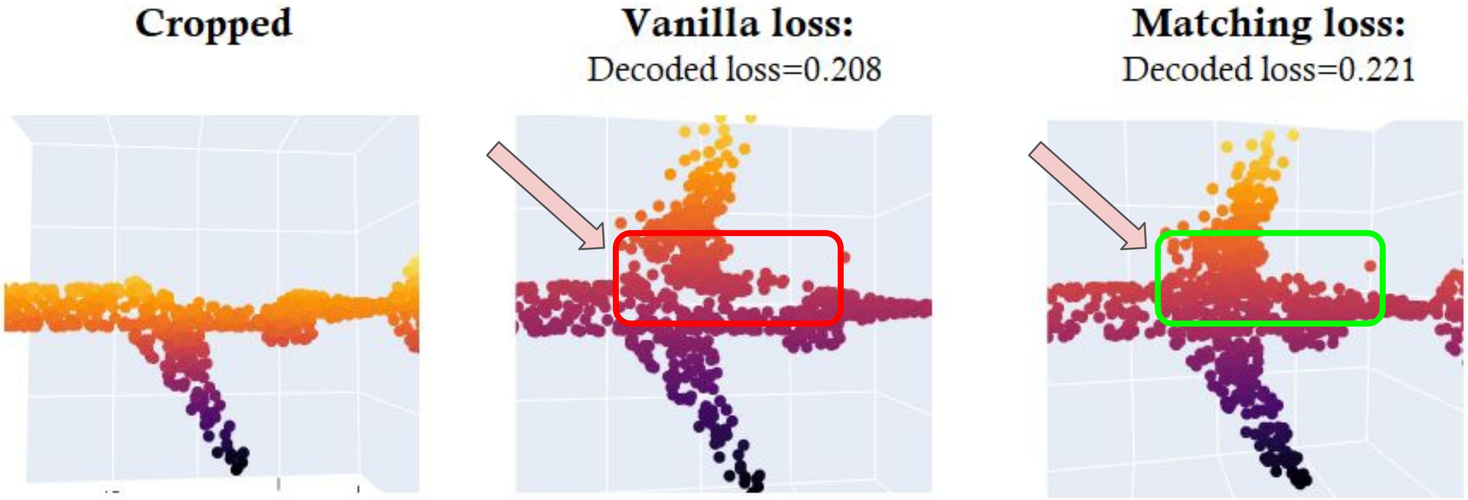
→ **Matching Loss function**

$$L_{PCC} = L(P_{rm}, P_m) + r \cdot (L(P_r, P_{gt}))$$

good reconstruction of missing part (also called **Vanilla Loss Function**)

keep global shape homogeneous

Point-cloud completion: Vanilla vs Matching loss



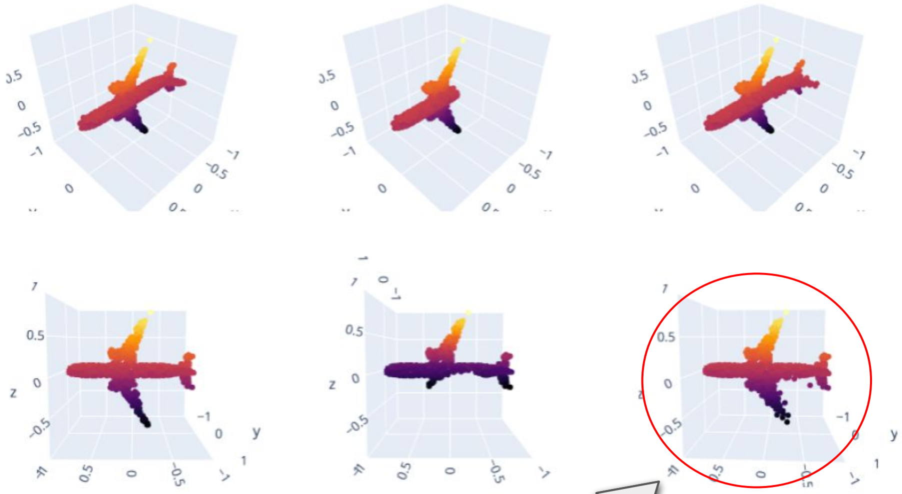
Point-cloud completion: Training

- **Training on all seven classes jointly**
  - *Airplane, Chair, Table, Lamp, Car, Motorbike, Mug.*

	Category	PNet_AE_512	
		Vanilla Loss Model	Matching Loss Model
Known	Airplane	0.103	**0.102**
	Chair	0.181	**0.167**
	Table	**0.180**	0.191
	Lamp	**0.592**	0.639
	Car	**0.181**	0.249
	Motorbike	**0.130**	0.372
	Mug	**0.283**	0.422
	Avg	***0.236***	*0.306*

Table . Testing Loss results for Point-Cloud Completion (training on all 7 known classes jointly using Vanilla Loss and Matching Loss). Testing Loss: mean Chamfer Distance only on the missing part.

Outputs for Matching Loss model



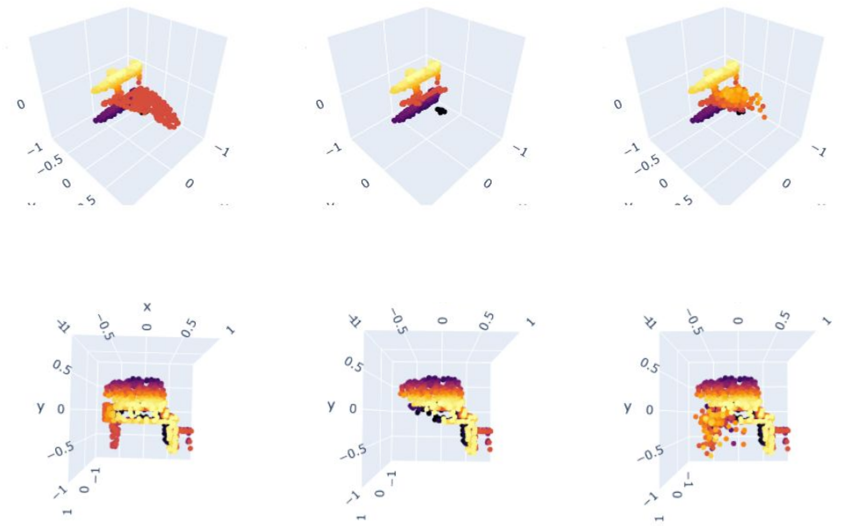
Wings symmetry?

Point-cloud completion: Testing on novel categories

	Category	PNet_AE_512	
		Vanilla Loss Model	**Matching Loss Model**
Similar	Basket	0.264	**0.249**
	Bicycle	**0.155**	0.159
	Bowl	0.474	**0.416**
	Helmet	**0.287**	**0.287**
	Microphone	2.426	**2.168**
	Rifle	0.152	**0.141**
	Watercraft	**0.087**	0.094
	Avg	*0,549*	***0.502***
Dissimilar	Bookshelf	0.239	**0.226**
	Bottle	**0.174**	0.184
	Clock	**0.257**	0.281
	Microwave	0.251	**0.242**
	Pianoforte	**0.168**	0.188
	Telephone	0.252	**0.219**
	Avg	***0,223***	*0.224*

Table . Testing Loss results for Point-Cloud Completion (training on all 7 known classes jointly using Vanilla Loss and Matching Loss). Testing Loss: mean Chamfer Distance only on the missing part.

Outputs for Matching Loss model



Point-cloud completion: Pros, cons and improvements

PROS	**CONS**
✓ Good at reconstructing general shape of the object	❌ Bad at reconstructing finer details
✓ Good at “transfer” local features learned for an object to reconstruct another one	❌ Bad at preserving symmetry
✓ Small embedding size	

IMPROVEMENTS

💡 Adjust the loss function to preserve symmetry and better reconstruct finer details

Thanks for your attention!

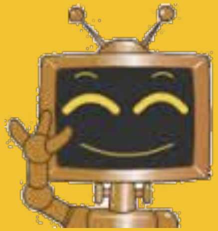


Image ‘Smile - Statistical Machine Intelligence and Learning Engine (haifengl.github.io)’

Luca Barco *s276072@studenti.polito.it*
Stefano Bergia *s276124@studenti.polito.it*
Daniela De Angelis *s277493@studenti.polito.it*

Politecnico di Torino
Machine Learning and Artificial Intelligence
A.Y. 2020/2021

AIML

Image: polito.it