# 3D DATA PROCESSING LAB 4

Giuseppe Labate mat.2095665

**Topic:** Deep 3D descriptors

**Goal:** Design a modified PointNet architecture that can extract 3D descriptors for matching. Replace the initial 3x3 T-Net with a rotation matrix extracted as in the Local Reference Frame of SHOT descriptors.

## Work description

### Task 1 – **PointCloudData**

For this task, the objective was to find anchor, positive and negative points and neighborhood.

To sample the anchor point, a random point belonging to pcd\_points is selected exploiting random.choice(pcd\_points).

To sample the positive point, the anchor\_pt nearest neighbor index in the noise point cloud is found by leveraging the KD-Tree.  
Next, the index is used to select the positive point in the noised point cloud.

To sample the negative point, a random point is selected from the noise point cloud and then it’s checked if its distance from the anchor\_pt is greater than the min\_dist.  
If it is so, the negative point is found.  
If it’s not, another point will be selected.

For each point, their neighborhood is found by picking all the point cloud points that are in a radius of self.radius

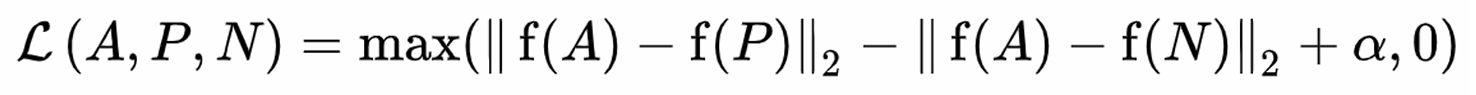
### Task 2 – **TinyPointNet**

In this task, it was requested to implement the TinyPointNet architecture:  
Immagine che contiene testo, diagramma, Piano, schermata

Descrizione generata automaticamente

To do so,

### Task 3 – Loss Function

In task 3 the objective was to insert the correct loss function.  
For this project, it was used a triplet loss as loss function:   
Where A, P and N stand respectively for anchor, positive and negative.

The loss is computed as the max between 0 and the difference of the L2 distances between anchor and positive and anchor and negative descriptors.

To implement this loss, tinypointnetloss = nn.TripletMarginLoss(margin = 1, p = 2)was used.  
Here margin = 1 is the α, margin between positive (A,P) and negative (A, N) pairs and p = 2 means that we’re using L2 distance.

## Encountered problems

The biggest problems I’ve encountered during this project are:

* The difficulty in writing and understanding NN pyhon code, because this was my first time
* The fact that I had to look every 5 minutes at the execution on colab because during training it constantly disconnected from the runtime.  
  I had to rerun multiple times the code because of these disconnections and this lead to wasted time.  
  Additionally, I had limited tries with colab GPU and often I had to use the much slower CPU

## Quantitative results

|  |  |  |
| --- | --- | --- |
|  | SVD RMSE | LM RMSE |
| Bunny | 0.00401621 | 0.00341366 |
| Dragon | 0.00568867 | 0.00564134 |
| Vase | 0.0162243 | 0.0162217 |

|  |  |  |
| --- | --- | --- |
|  | SVD ITERATIONS | LM ITERATIONS |
| Bunny | 23 | 21 |
| Dragon | 13 | 19 |
| Vase | 25 | 29 |

## Qualitative results

### 4.1)