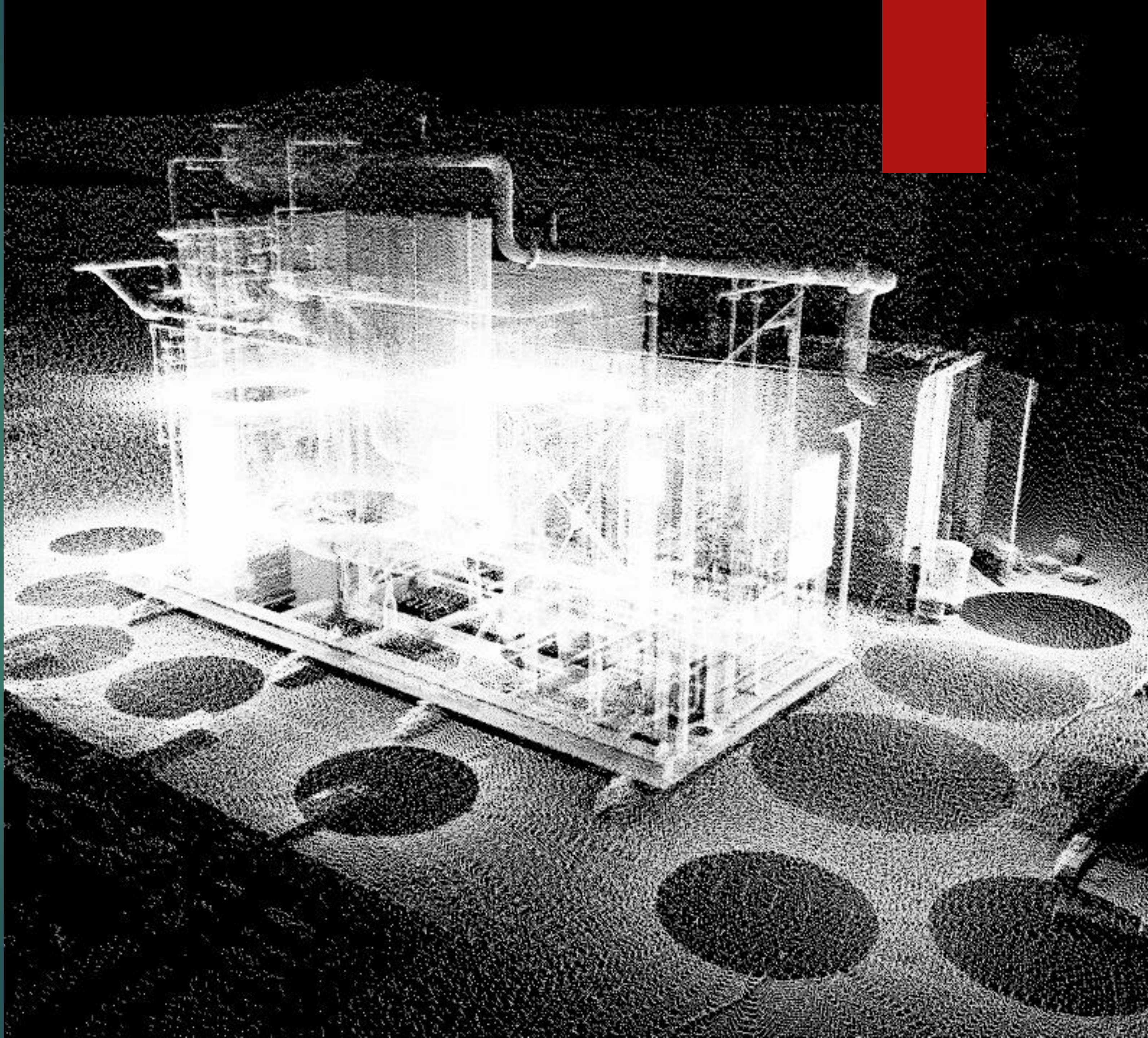
Point Clouds Processing Tutorial

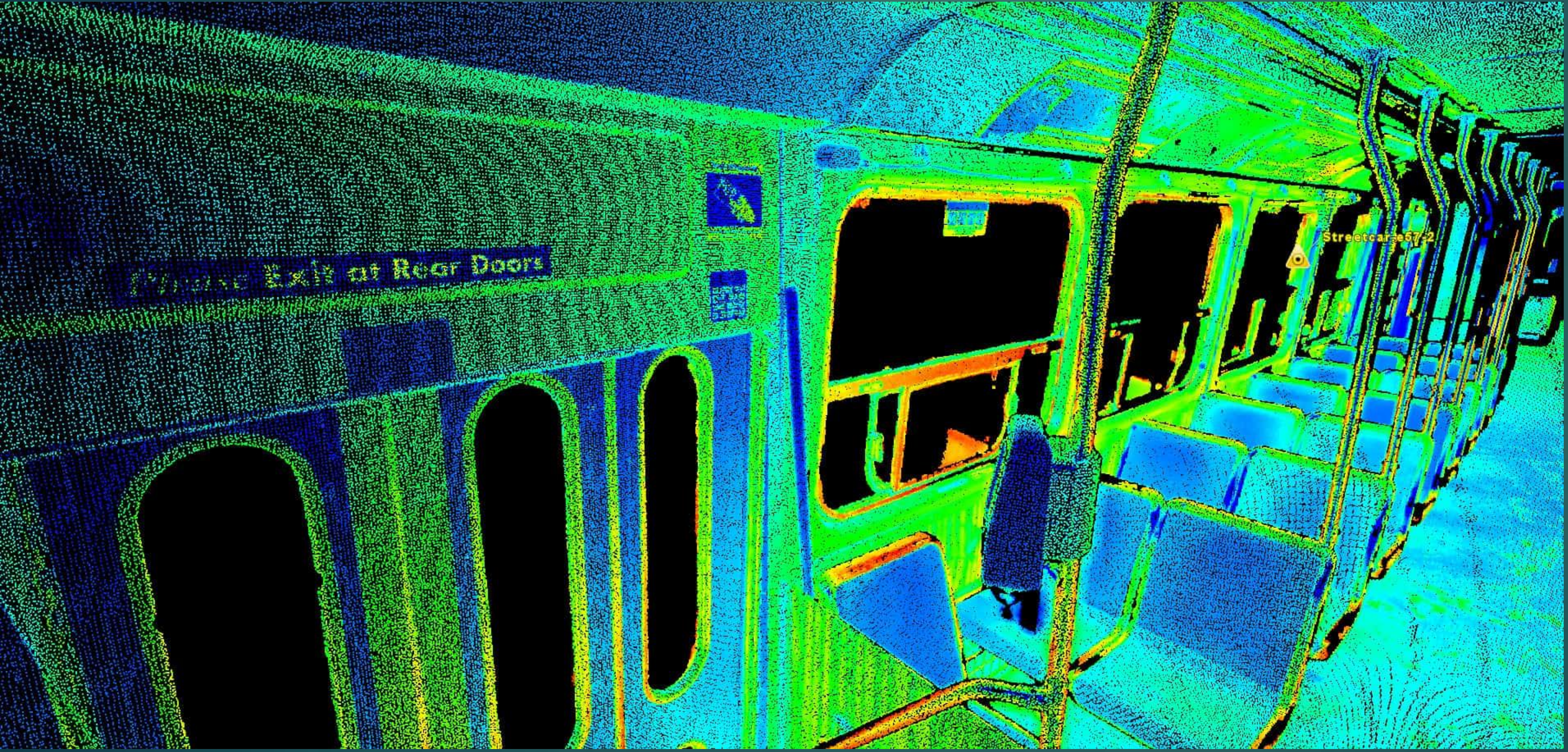
Beginner level Introduction & Building a point clouds classification model

Presenter: Zhoukatong Xia



What is Point Clouds?

- In recent years, there is a "new" way to sense the world, detect our familiar surrounding but gives an unfamiliar result -- the Point Clouds



- Even though, Point Clouds and 3d sensors(like Lidar), has existed for decades, many people start to know it recently.
- So what makes point cloud so popular? We should have a deeper look at our smart phones, cars and VR/AR devices.

Ref: Point Cloud Conversion to 3D Model [1]
Source: https://www.cadcam.org/3d-scanning/converting-point-clouds-into-cad/

Point Clouds vs. Smart Phone

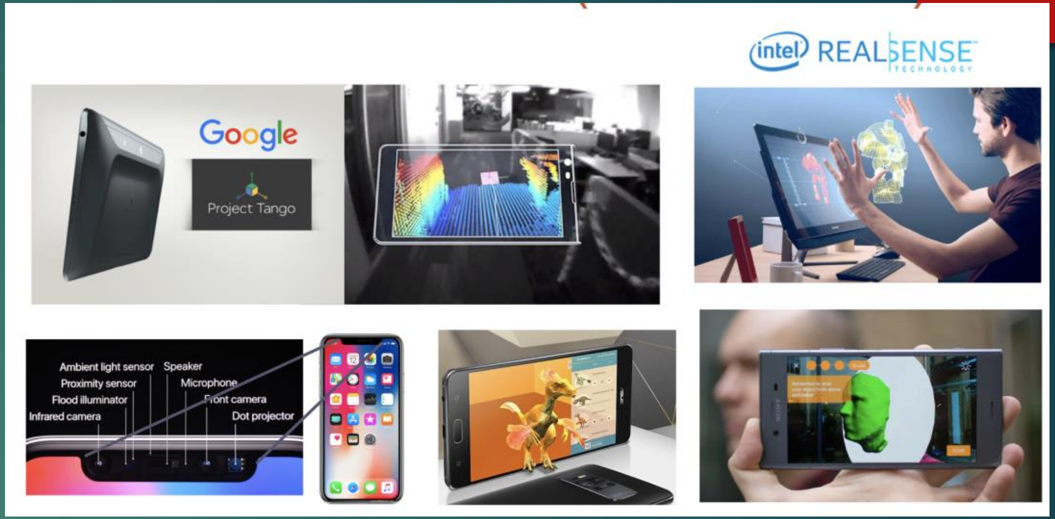
More and more smart phone can capture Point Clouds (like Iphone11/12, Asus Zenfone AR, Sony Xperia XZ1…)

And many applications are based on point clouds data, like:

Apple Face ID use a dot projector to gather around 3000 points to identify user's face.

Newly released Iphone12 Pro owns currently(2020) the best Lidar scanner, which could build 3d model for your room, detect obstacle for people with visual impairment, and do auto-focus and subject detection at night.

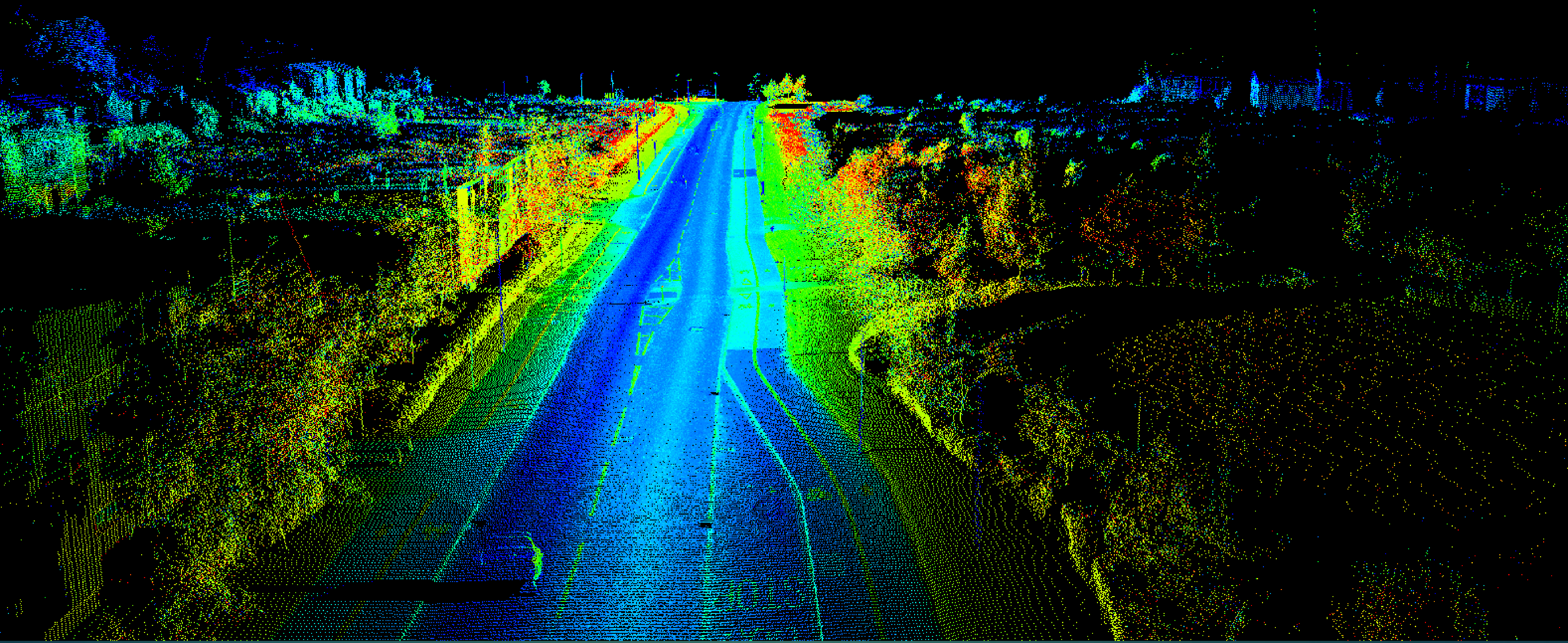
However, all device has a limiting scanning range(1m-5m), Sometimes, we want to see the world, not just my room.



Devices that can capture Point Clouds

Ref: Deep Learning on Point clouds: Implementing PointNet in Google Colab [1]

[3] Source: https://www.roadtovr.com/apple-iphone-12-pro-max-lidar-instant-ar-depth-mapping/



Here is the world in point clouds format.

We can imagine a vehicle with Lidar was traveling forward and generate this 3d model.

But to do so, better 3d scanner with longer range and better resolution is needed.

[4] Ref: https://techcrunch.com/2017/02/12/wtf-is-lidar/

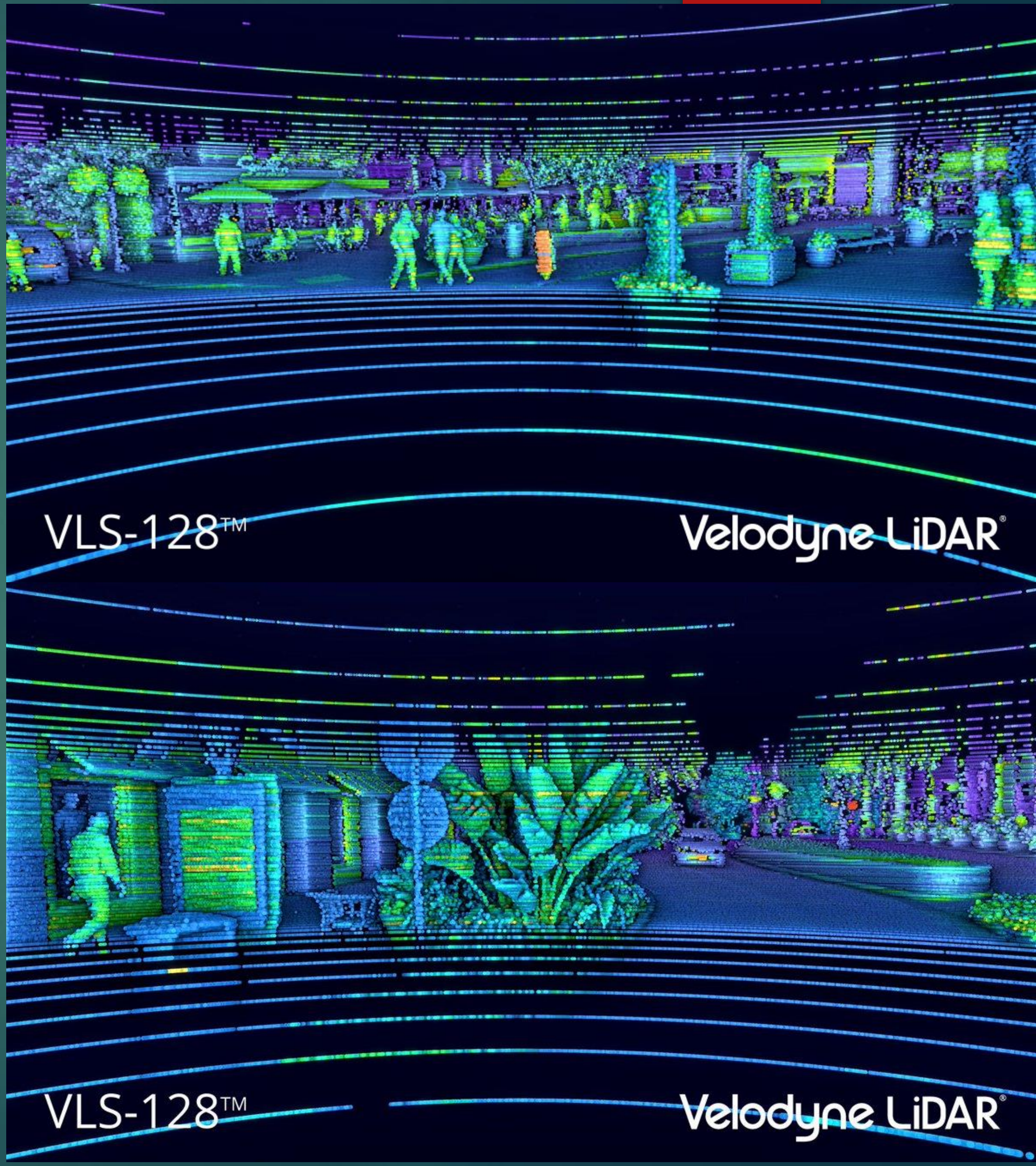
Sensor

Lidar Scanner

Currently, Lidar used on autonomous vehicles have a detecting range from 100m to 200m,
And much higher resolutions to distinguish people from other objects.

Because of the development of Lidar industry, we have plenty of resources of point clouds

[4] Velodyne Lidar
Source: https://velodynelidar.com/



Principle & Feature of Lidar

- As currently the most powerful and reliable source of point cloud, many people could ask:
- Is that a good idea to use Lidar to sense(understand) the world? Why not use camera, or radar, which are obviously much popular.
- So, If your AIM is to have abundant and accurate spatial information.
- The answer is yes.
- In the next Slide, we can have a brief compare of the difference between data gathered by camera, radar and Lidar.

Source

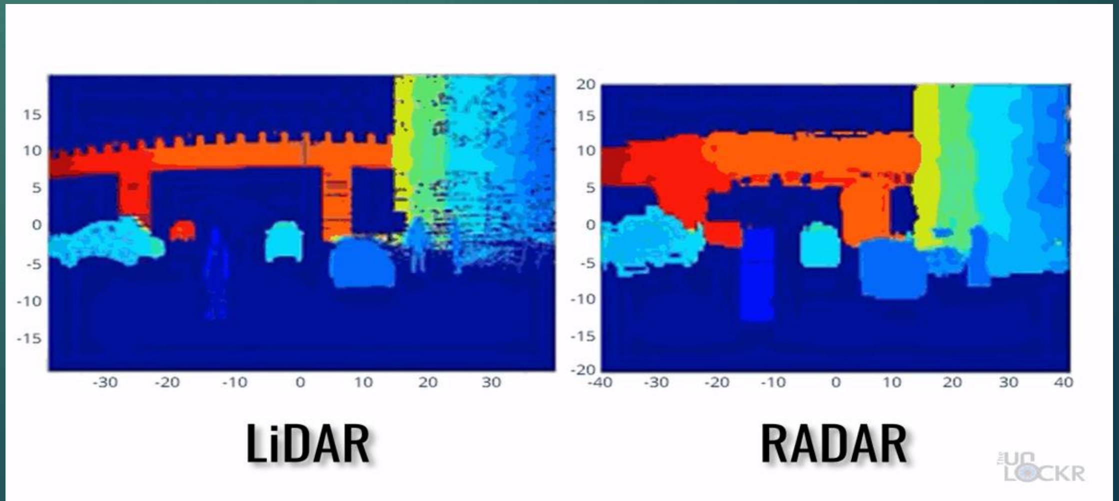
Property of Lidar

- Traditionally, we use radar to detect range, and use camera to observe texture.
- And Lidar is kind of the middle of radar and camera in function
- Here are the advantage of each device

Camera can generate high resolution, rich texture images

Lidar can detect range. As it can emit electromagnetic beam for detection, working in complete darkness is also possible.
It have better resolution than radar, making object classification Possible.

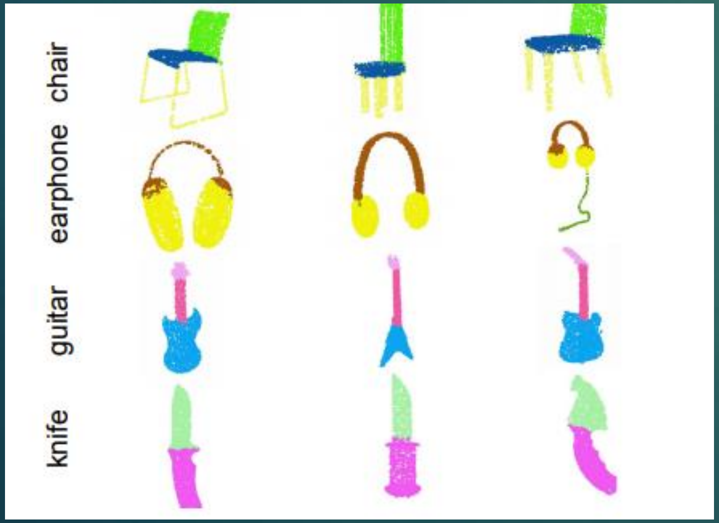
Radar also emit electromagnetic beam to detect object but will get a blur feedback.



Application

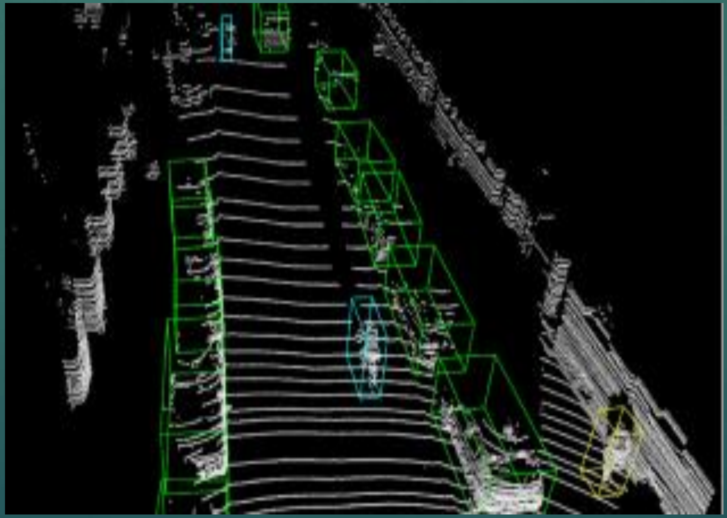
- As we have mentioned, point clouds can help Face ID check our identity, help autonomous cars detect and identify people on the road. What else can point clouds do?
- Here is the three main area:

Point clouds classification



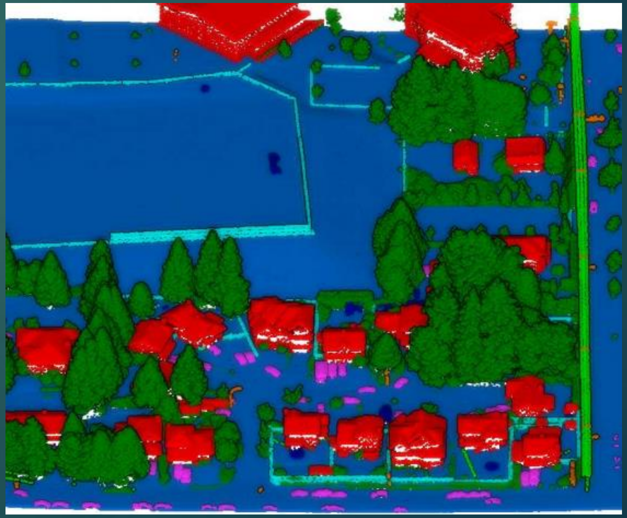
Ref: PointNet: deep learning on point sets for 3D classification and segmentation https://arxiv.org/abs/1612.00593

3D object detection



Ref: VoxelNet: end-to-end learning for point cloud based 3D object detection https://arxiv.org/abs/1711.06396

Point clouds segmentation



Ref: Learning object bounding boxes for 3D instance segmentation on point clouds https://arxiv.org/abs/1906.01140

How to get started

- If you are interested to working on point cloud processing, the following slides are just for you.
- First, let me introduce some resources of point cloud data to train you model

1. ModelNet [6],
2. ScanObjectNN [7],
3. ShapeNet [8],
4. PartNet [9],
5. S3DIS [10],
6. ScanNet [11],
7. Semantic3D [12],
8. ApolloCar3D[13],

- Then, we can dive into a deep learning model for point clouds classification from PointNet

Point cloud Processing: -- Design Principles and Implementation

- **First, Let us find some training data:**
- **Source: PointNet**

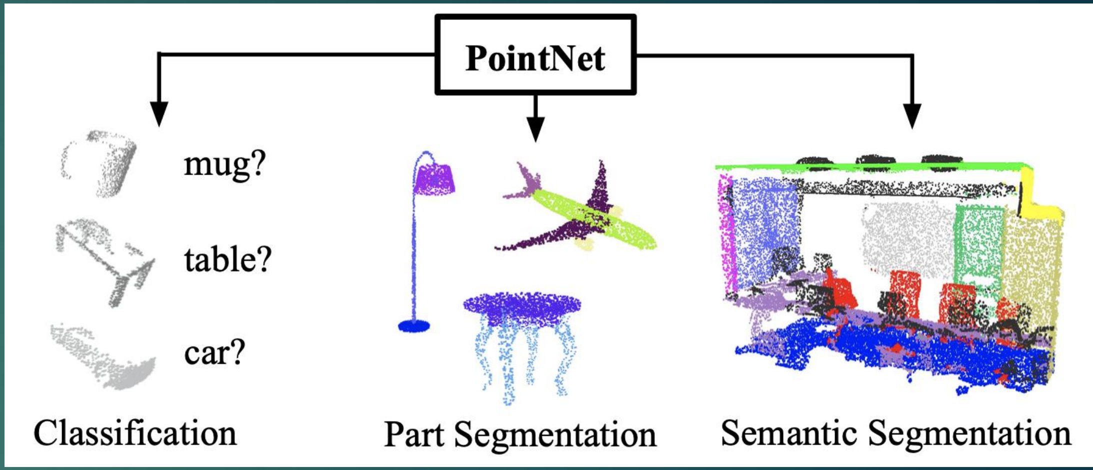
  Provide data for point cloud classification and segmentation
- **DataSet: ModelNet10**

  Data Type: mesh (a collection of vertices and triangular faces that defines the shape)

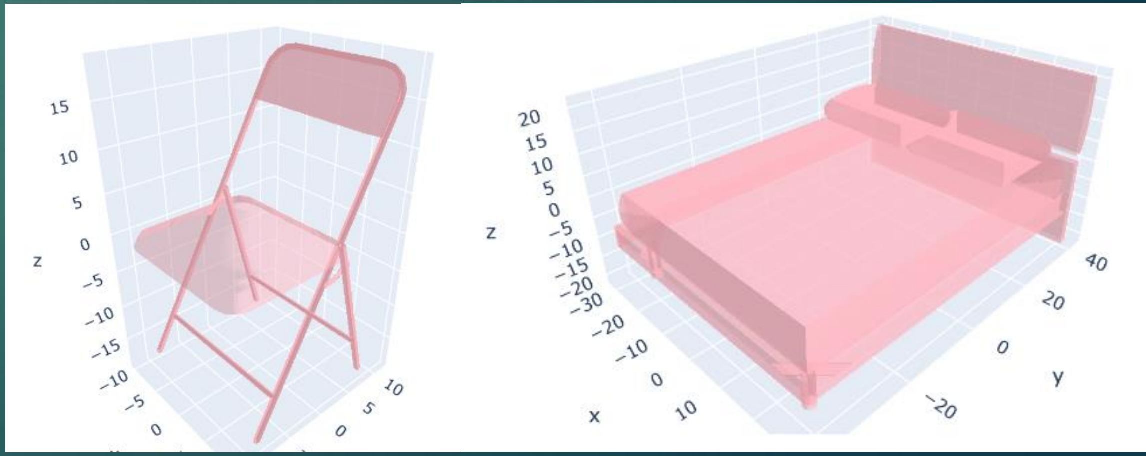
  Class Number: 10 categories;

  Content: 3,991 models for training, 908 models for testing;
- **Other Sources for point cloud:**

  ModelNet [6], ScanObjectNN [7], ShapeNet [8], PartNet [9], S3DIS [10], ScanNet [11], Semantic3D [12], ApolloCar3D[13]



PointNet application



Meshes in ModelNet10

Ref: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation
Source: https://arxiv.org/pdf/1612.00593.pdf

Data preprocessing

Data Sampling:

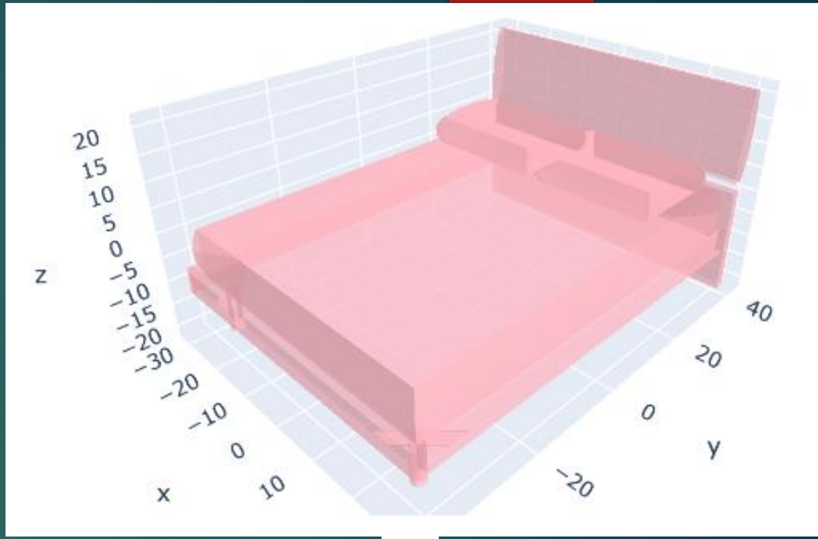
- As points are not uniformly distributed in the raw data, we need:
- Uniformly sample points

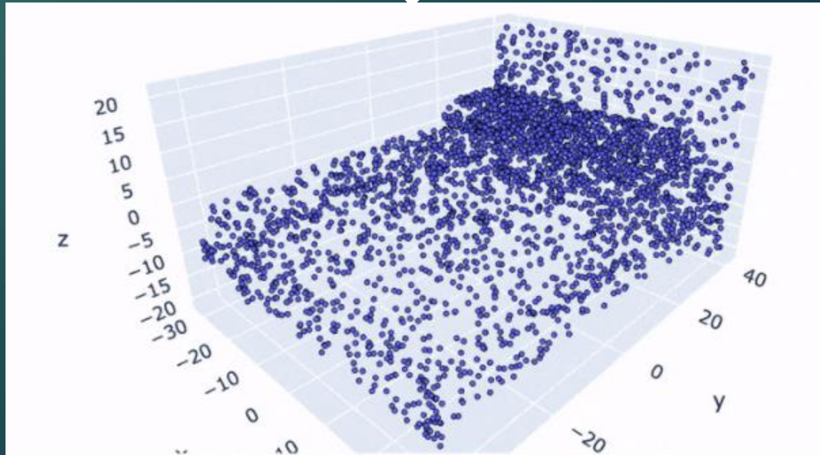
```
## Uniformly sample points
verts, faces = mesh
areas = np.zeros((len(faces)))
verts = np.array(verts)

def triangle_area(pt1, pt2, pt3):
    side_a = np.linalg.norm(pt1 - pt2)
    side_b = np.linalg.norm(pt2 - pt3)
    side_c = np.linalg.norm(pt3 - pt1)
    s = 0.5 * ( side_a + side_b + side_c)
    return max(s * (s - side_a) * (s - side_b) * (s -
side_c), 0)**0.5

for i in range(len(areas)):
    areas[i] = (triangle_area(verts[faces[i][0]],
                              verts[faces[i][1]],
                              verts[faces[i][2]]))
```

Result:





- As the input size of our neural network is fixed, we need
- sample a fixed number of points as input from previous distribution.

```
## sample a fixed number (k) of points
k = 3000
sampled_faces = (random.choices(faces, weights=areas,k=k))
def sample_point(pt1, pt2, pt3):
    s, t = sorted([random.random(), random.random()])
    f = lambda i: s * pt1[i] + (t-s) * pt2[i] + (1-t) * pt3[i]
    return (f(0), f(1), f(2))

pointcloud = np.zeros((k, 3))

for i in range(len(sampled_faces)):
    pointcloud[i] = (sample_point(verts[sampled_faces[i][0]],
                                  verts[sampled_faces[i][1]],
                                  verts[sampled_faces[i][2]]))
```

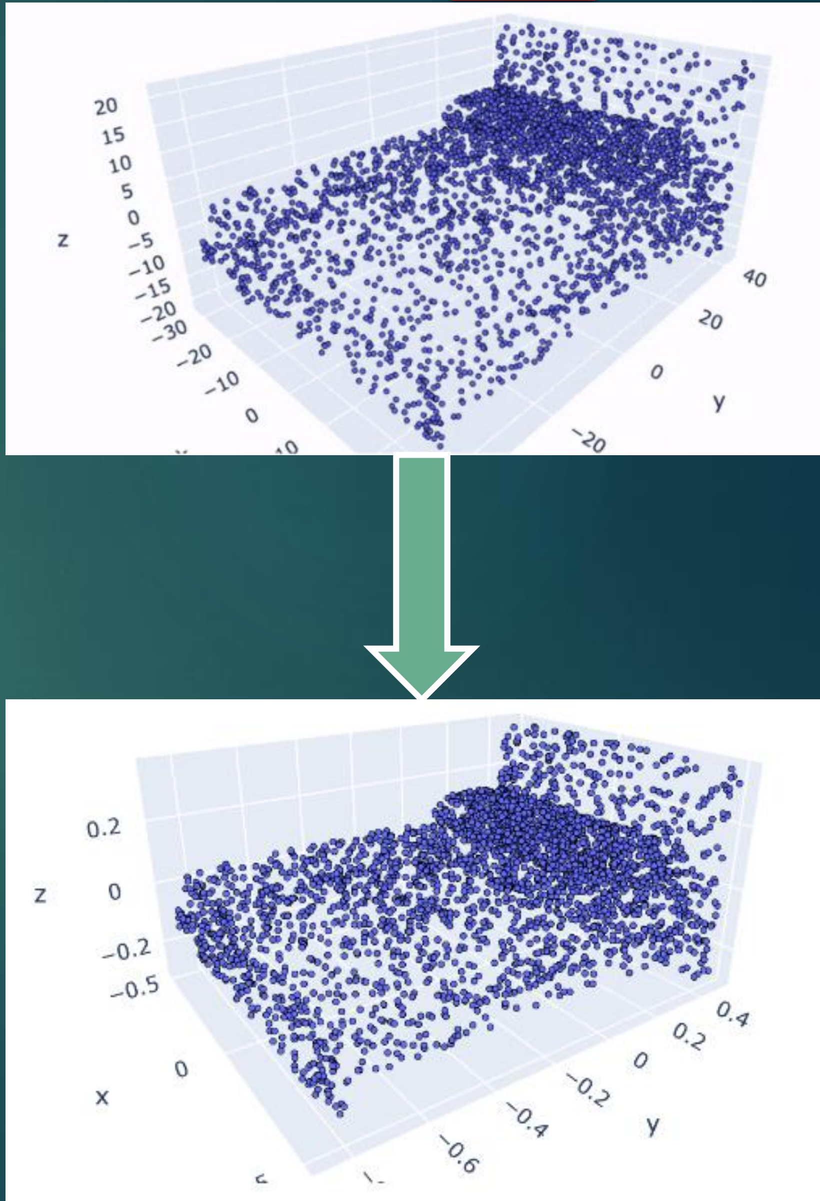
Ref: visit PointNet for more code
Source: https://github.com/nikitakaraevv/pointnet/blob/master/nbs/PointNetClass.ipynb

Normalizing by shape:

- Problem:

  Identical **objects can have different sizes** and can be placed **in different parts of coordinate system**.

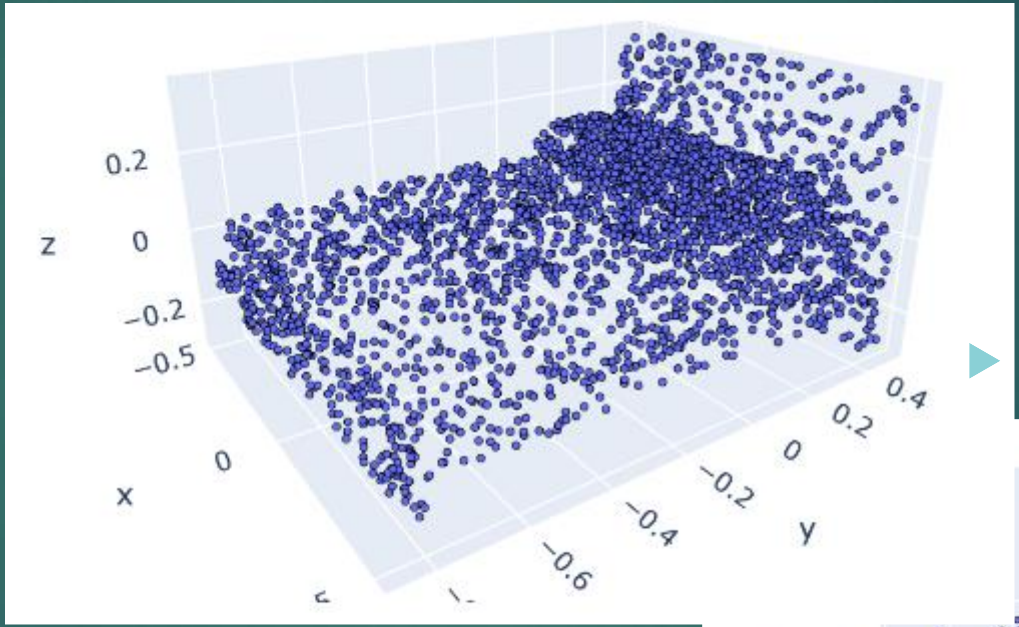
- Solution:
- 1. **Translate** the object **to the origin** by subtracting mean from all its points
- 2. divide some number to make all points lies in range 0~1



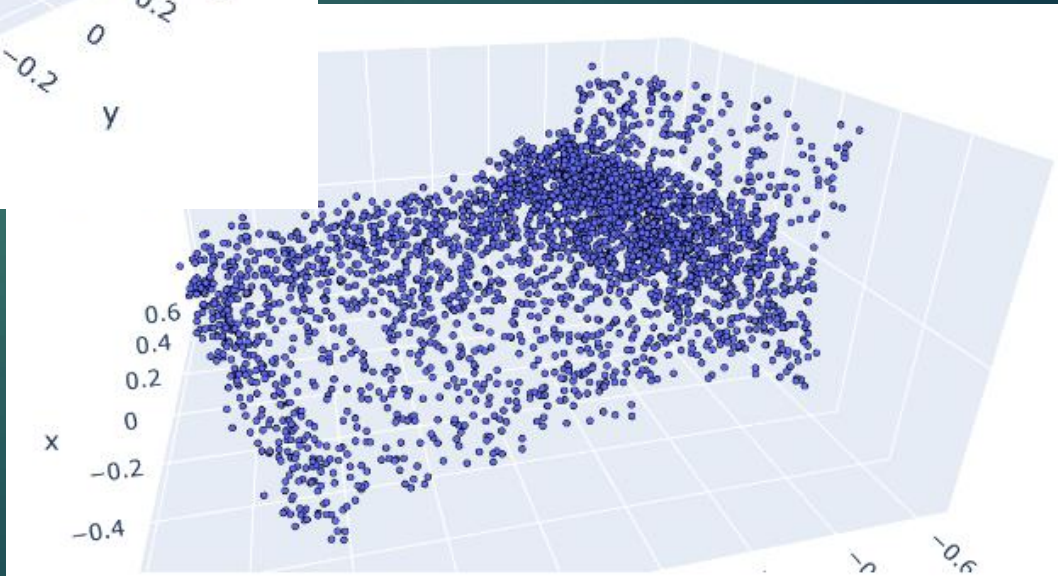
Rotate and add noise:

- **1. randomly rotate** objects for some degree for all training data
- 2. add a random noise (range 0~0.02) to each point

- **Before adding noise**



- **After adding noise**



Model Design:

- The model for point cloud classification should have the following properties:
- 1. Permutation Invariance
- 2. Transformation Invariance
- So, in the next few slides, we will introduce:
- 1. how to use max pool to generate global feature, which is the signature of a model.
- 2. how to use T-Net to transform input data, which can achieve Transformation Invariance
- 3. the final model structure

Permutation Invariance

- Problem:

  As point clouds are inherently unstructured data, the order of point so not say anything,

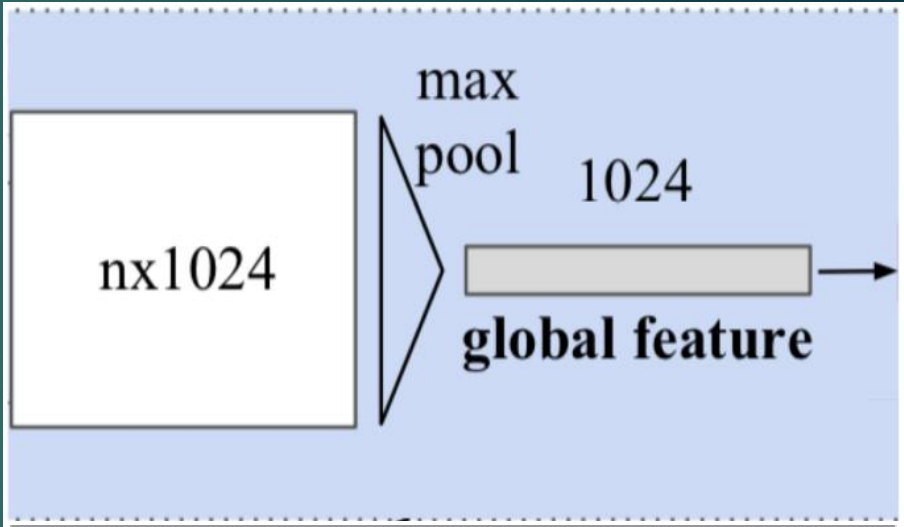
  If we have N data points, there are *N*! permutations all indicating the exact same object.

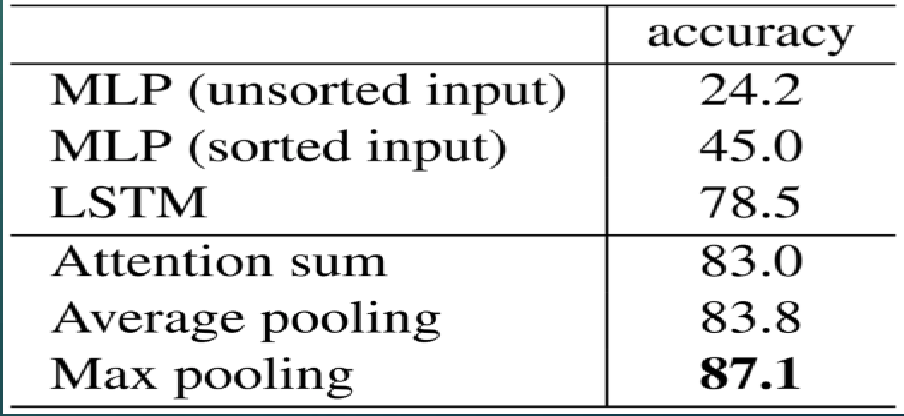
- Solution:

  Use a symmetric function (which will return the same output regardless of the order of the input arguments)

  To implement a symmetric functions:

  1. we map n points to higher dimension(1024)
  2. use max pool to capture the global feature
  3. This global feature is used directly for classification





	accuracy
MLP (unsorted input)	24.2
MLP (sorted input)	45.0
LSTM	78.5
Attention sum	83.0
Average pooling	83.8
Max pooling	**87.1**

Ref: Empirical testing of symmetric functions (source)

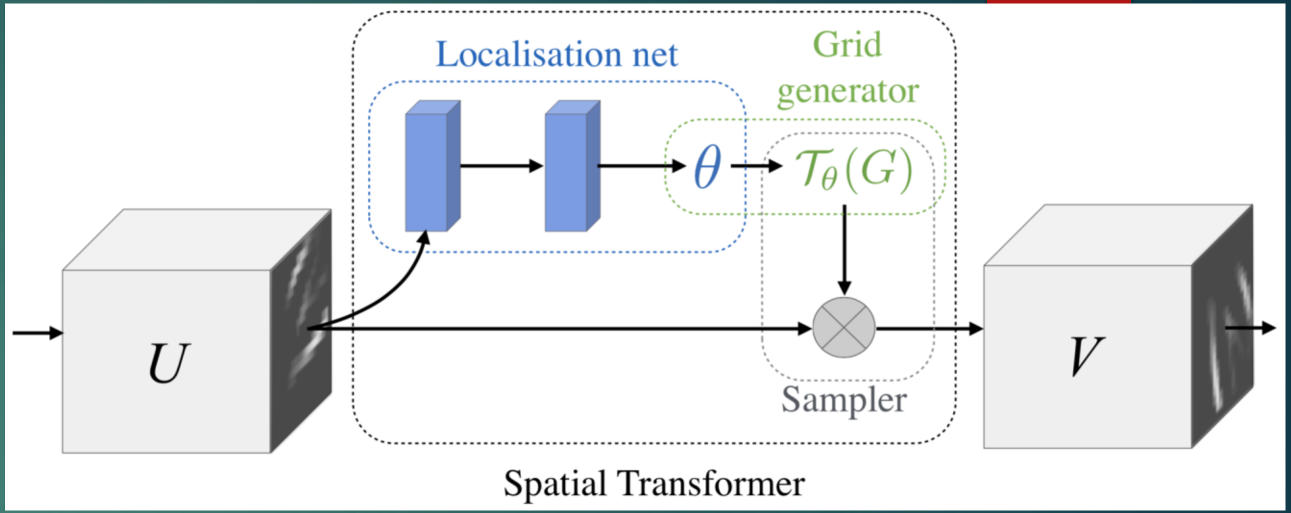
Transformation Invariance

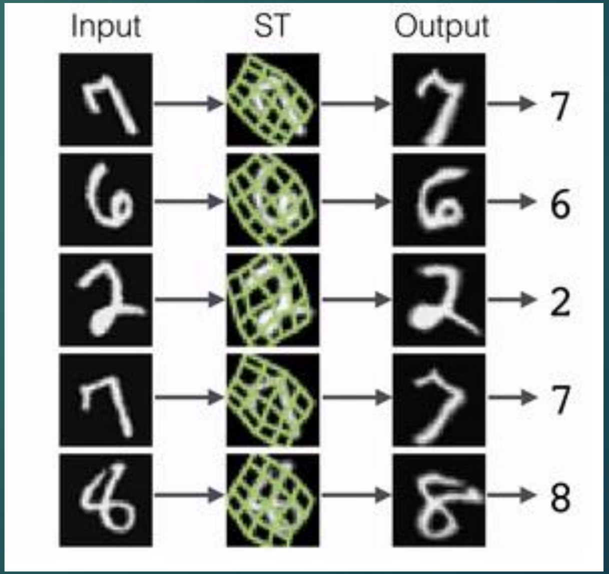
- Problem: our models should identify the same object in pose(angle).
- Idea: as we already have Spatial Transformer Networks (STN) for 2d object, as shown on the right, can we process 3d points in the same way? Answer is yes.
- Principle of STN:

  generate a transformation angle θ *to normalize the original input U.*

  Thus as shown on the right picture, input vector of different poses are normalized vertically.

- *Next: how to use this principle to 3d points*



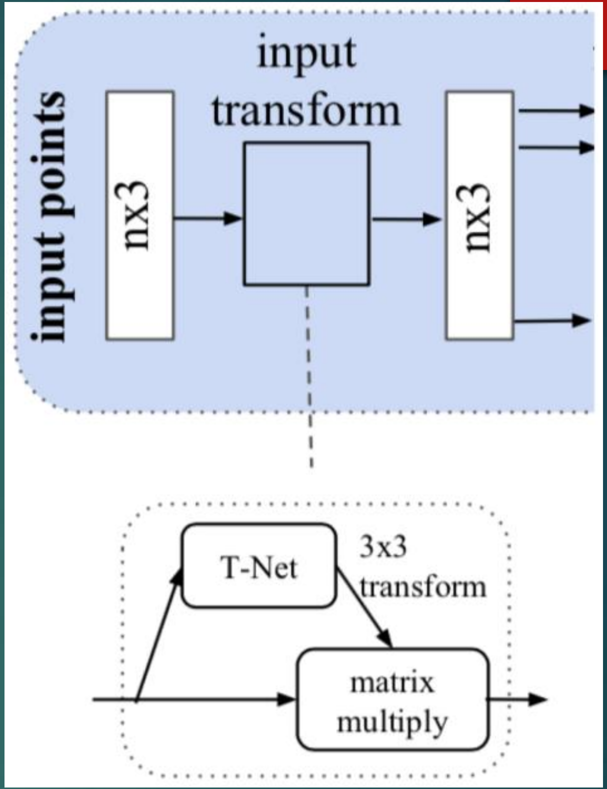


Ref: Review: STN — Spatial Transformer Network (Image Classification)
Source: https://towardsdatascience.com/review-stn-spatial-transformer-network-image-classification-d3cbd98a70aa

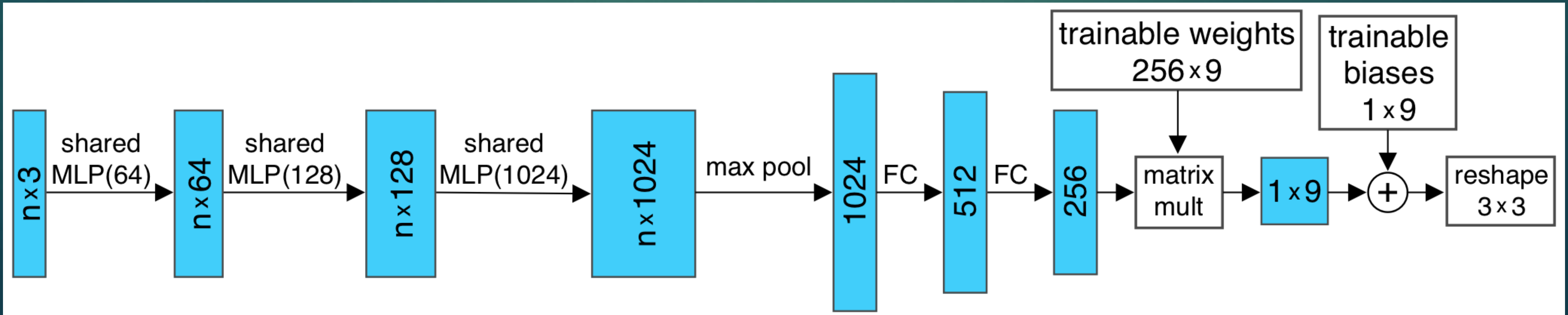
Transformation Invariance

- The similar structure of STN is called T-Net
- Principle of T-Net:
- Based on the whole model, T-Net will generate an transformation matrix of 3*3 shape
- Then, it will take each point in this model as input, and output a transformed point, as shown in the right picture
- Structure of T-Net are shown below:

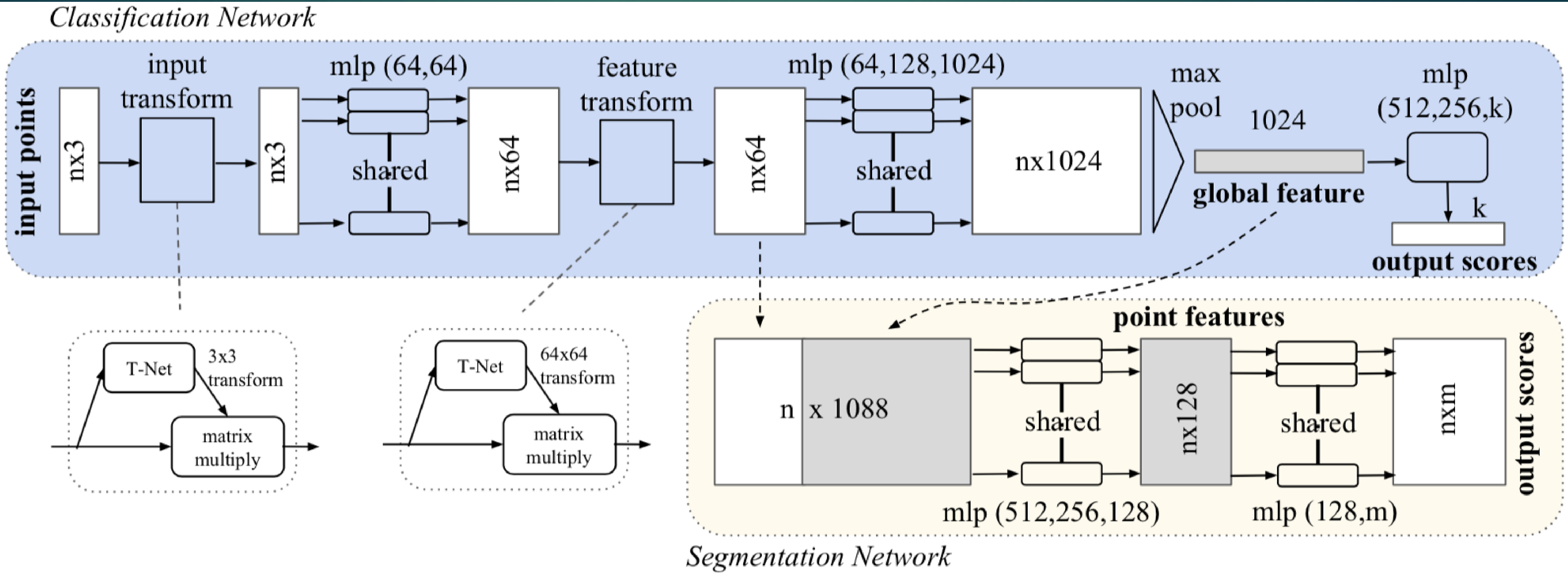
  Input points(n*3) → encoded global vector(256) → transform matrix(3*3) ; and for the last step, we use some trainable weights to help



Ref: Review: STN — Spatial Transformer Network (Image Classification)



Model structure for: Classification & Segmentation



Ref : https://medium.com/@luis_gonzales/an-in-depth-look-at-pointnet-111d7efdaa1a

Model implementation:

```
## Layers:
import torch
import torch.nn as nn
import torch.nn.functional as F

class Tnet(nn.Module):
   def __init__(self, k=3):
      super().__init__()
      self.k=k

      self.conv1 = nn.Conv1d(k,64,1)
      self.conv2 = nn.Conv1d(64,128,1)
      self.conv3 = nn.Conv1d(128,1024,1)

      self.fc1 = nn.Linear(1024,512)
      self.fc2 = nn.Linear(512,256)
      self.fc3 = nn.Linear(256,k*k)

      self.bn1 = nn.BatchNorm1d(64)
      self.bn2 = nn.BatchNorm1d(128)
      self.bn3 = nn.BatchNorm1d(1024)
      self.bn4 = nn.BatchNorm1d(512)
      self.bn5 = nn.BatchNorm1d(256)
```

```
def forward(self, input):
    # input.shape == (bs,n,3)
    bs = input.size(0)
    xb = F.relu(self.bn1(self.conv1(input)))
    xb = F.relu(self.bn2(self.conv2(xb)))
    xb = F.relu(self.bn3(self.conv3(xb)))
    pool = nn.MaxPool1d(xb.size(-1))(xb)
    flat = nn.Flatten(1)(pool)
    xb = F.relu(self.bn4(self.fc1(flat)))
    xb = F.relu(self.bn5(self.fc2(xb)))
    init = torch.eye(self.k, requires_grad=True).repeat(bs,1,1)
    if xb.is_cuda:
      init=init.cuda()
    # add identity to the output
    matrix = self.fc3(xb).view(-1,self.k,self.k) + init
    return matrix
```

Ref : https://medium.com/@luis_gonzales/an-in-depth-look-at-pointnet-111d7efdaa1a

Reference

- [1] Point Cloud Conversion to 3D Model; Source: https://www.cadcam.org/3d-scanning/converting-point-clouds-into-cad/
- [2] Deep Learning on Point clouds: Implementing PointNet in Google Colab; Source: https://towardsdatascience.com/deep-learning-on-point-clouds-implementing-pointnet-in-google-colab-1fd65cd3a263
- [3] Source: https://www.roadtovr.com/apple-iphone-12-pro-max-lidar-instant-ar-depth-mapping/
- [4] Velodyne Lidar; Source: https://velodynelidar.com/
- [5] PointNet: deep learning on point sets for 3D classification and segmentation; https://arxiv.org/abs/1612.00593
- [6] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao,
- “3D shapeNets: A deep representation for volumetric shapes,” in
- CVPR, 2015.
- [7] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung,
- “Revisiting point cloud classification: A new benchmark dataset
- and classification model on real-world data,” in ICCV, 2019.
- [8] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang,
- Z. Li, S. Savarese, M. Savva, S. Song, and H. Su, “ShapeNet:
- An information-rich 3D model repository,” arXiv preprint
- arXiv:1512.03012, 2015.
- [9] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and
- H. Su, “PartNet: A large-scale benchmark for fine-grained and
- hierarchical part-level 3D object understanding,” in CVPR, 2019.
- [10] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer,
- and S. Savarese, “3D semantic parsing of large-scale indoor
- spaces,” in CVPR, 2016.
- [11] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and
- M. Nießner, “ScanNet: Richly-annotated 3D reconstructions of
- indoor scenes,” in CVPR, 2017.
- [12] T. Hackel, N. Savinov, L. Ladicky, J. Wegner, K. Schindler, and
- M. Pollefeys, “Semantic3D.net: A new large-scale point cloud
- classification benchmark,” ISPRS, 2017.
- [13] X. Song, P. Wang, D. Zhou, R. Zhu, C. Guan, Y. Dai, H. Su,
- H. Li, and R. Yang, “Apollocar3D: A large 3D car instance
- understanding benchmark for autonomous driving,” in CVPR,
- 2019.