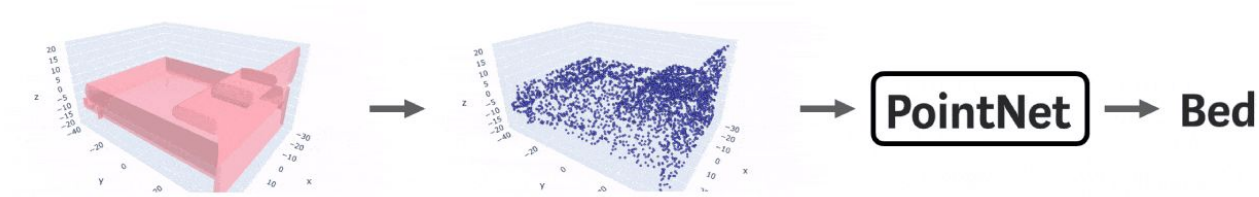
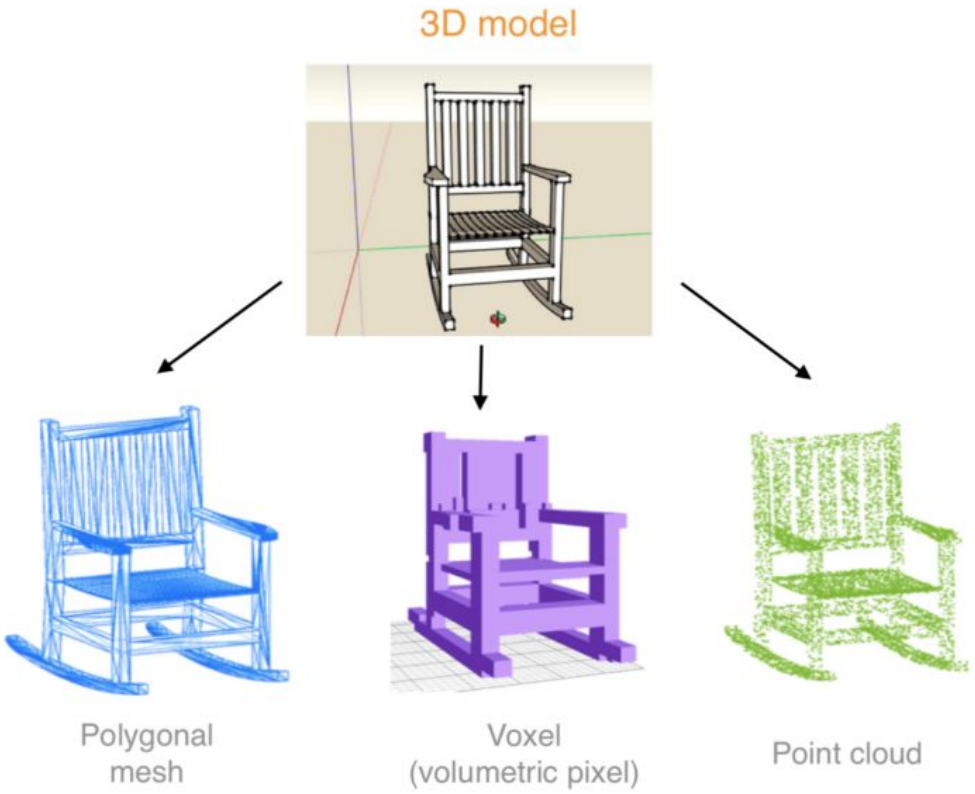
Point net

Pointnet is a deep neural architecture for point cloud object recognition.



1. Introduction

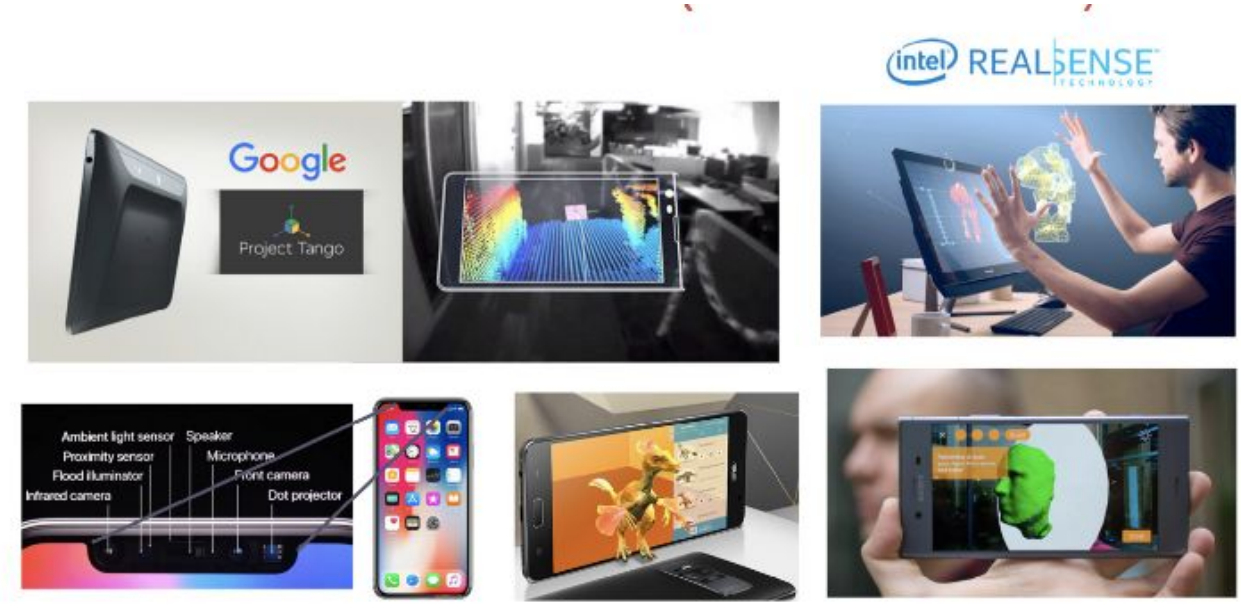
A set of data points in space create a point cloud. The 3D shape of the object can be represented with these point clouds. Every point is associated with three coordinates X, Y and Z coordinates.
3D points can be represented in different formats for example, polygonal mesh, volumetric pixel grid, point cloud and so on. As shown below: source



“90% of improvements in computer vision deals with only 2D images” : source

1.1 Point clouds

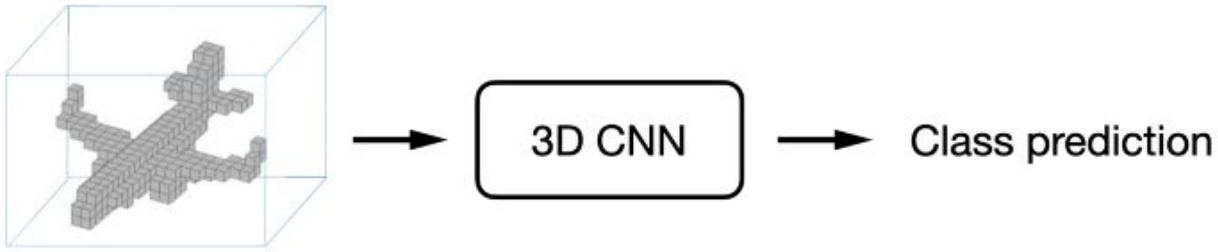
Point clouds can be generated using depth sensors such as LIDAR and RGB-D cameras.
It represents the points in the space and shows no connectivity between points.



Point cloud capturing devices: source

1.2 Deep learning on point clouds

We extend the idea of 2D convolution for 3D image grids.



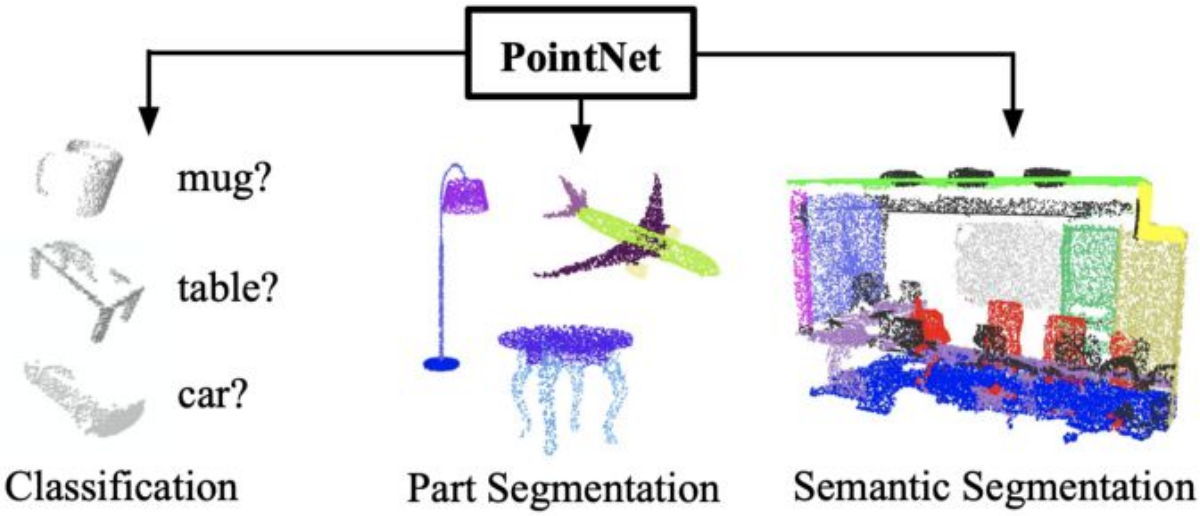
Source

1.3 Pointnet

Three main problem with the point clouds are as follows:

- Point clouds are not in order. So the algorithm needs to be efficient for changing permutation of the input sets.
- It needs to be invariant to rigid transformation
- Interaction between different points needs to be captured.

Pointnet is able to solve these problems and manage to perform classification and segmentation.



source

2. Implementation

Re implementation of classification model from the paper in Google Colab using Pytorch.

The source code and implementation can be found at:

2.1 Dataset

The author of the Pointnet has used a ModelNet40 dataset. The dataset is composed of 12,311 models from 40 object categories containing 9843 training data and 2468 testing data.

To reduce the computation time a small subset of the Modelnet40 is named as ModelNet10.

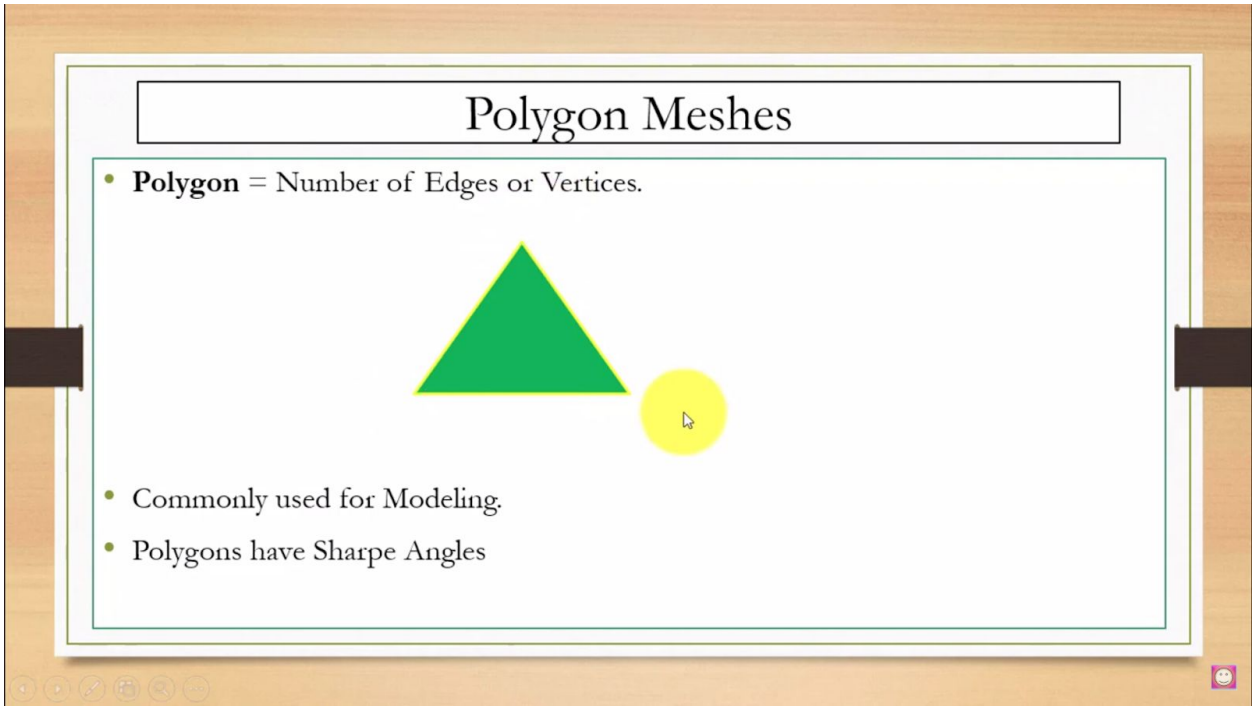
```
import numpy as np
import random
import math
!pip install path.py;
from path import Path
```

“.off” files that contain meshes represented by vertices and triangular faces. Vertices are just points in a 3D space and each triangle is formed by 3 vertex indices.

Polygon mesh:

A polygon mesh is a collection of vertices, edges and faces that defines the shape of a polyhedral object.

In the below triangle we have three vertices, three edges and one face.



A 3D polygon mesh consists of one or more polygons.

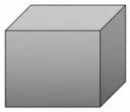
Polygon mesh made of points(vertices), edges and faces.

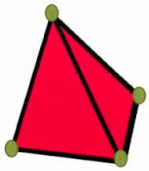
Polygon Meshes

- **Polygon Mesh =** 3D Object [Composed of One or more Polygons]
- Polygon Mesh Components: Points, Edges and Faces.

Polygon Meshes

- **Polygon Mesh =** Collection of Edges, Vertices and Faces [that define SHAPE of Polyhedral Object].
- Oldest form of Geometry Representation in Computer Graphics.
- **Polyhedral Object** is a 3D Shape with STRAIGHT Edges, FLAT Polygon Faces and SHARP Vertices.



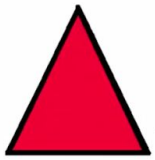


Polygon Meshes

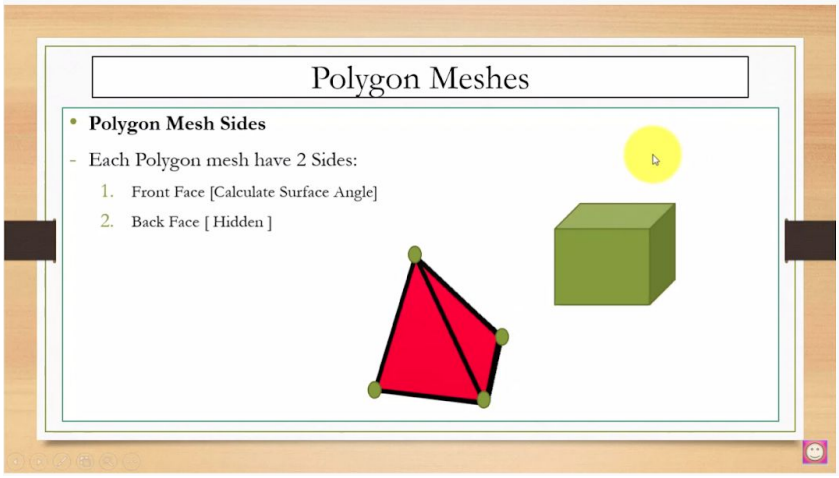
- **Polygon Mesh =** Collection of Edges, Vertices and Faces [that define SHAPE of Polyhedral Object].
- **Polyhedral Object** is a 3D Shape with STRAIGHT Edges, FLAT Polygon Faces and SHARP Vertices.

Vertices

Edges



Faces



Point sampling

Since the points are not uniformly distributed on the object surfaces it will be difficult for point nets to recognize them.
To solve this problem a approach to uniformly sample points on the object surface is carried out.
Different faces have different areas and probability of choosing a particular face is proportional to its area.

Given the vertices we can calculate the area using the **Heron's formula.**

Heron's formula states that the area of a triangle whose sides have lengths a, b, and c is

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

Where s is the semi perimeter of the triangle

$$s = \frac{a+b+c}{2}$$

All the mesh areas are calculated using these formulas.

We have a dense layer in network architecture that is why we need a fixed number of points in a point cloud.
We can sample from the constructed distribution then we choose one point per chosen face.

Return a list with 14 items.
The list should contain a randomly selection of the values from a specified list, and there should be 10 times higher possibility to select "apple" than the other two:

```
mylist = ["apple", "banana", "cherry"]

print(random.choices(mylist, weights = [10, 1, 1], k = 14))
```

In our case the weights are the areas and K is the total number of data points sampled.

We use barycentric coordinates on the circle to sample points. https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-rendering-a-triangle/barycentric-coordinates#:~:text=Barycentric%20coordinates%20are%20also%20known,A%2C%20B%2C%20C).

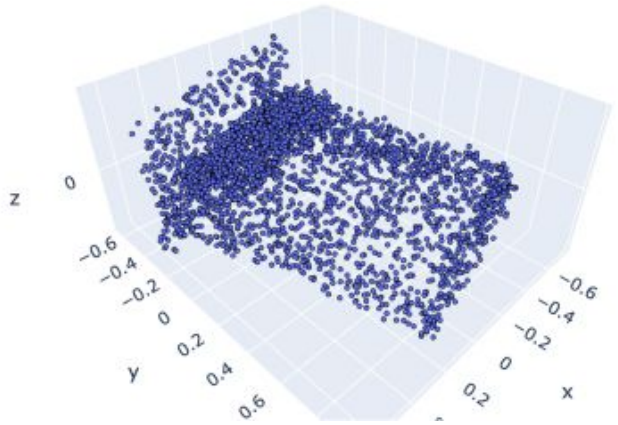
https://mathworld.wolfram.com/BarycentricCoordinates.html

https://www.youtube.com/watch?v=G_8e5f8iNBw&ab_channel=InsightsintoMathematics

Some faces can have more than one sampled point while others can not have points at all.

Data augmentation

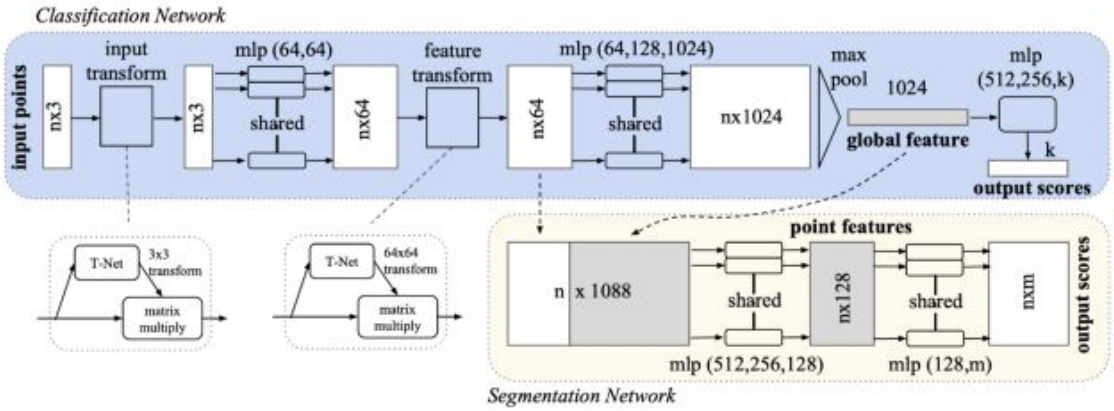
Objects can have different sizes and can be placed at different coordinates.
So, let's translate the object to the origin by subtracting mean from all its points and normalizing its points into a unit sphere.
To augment the data during training, we randomly rotate objects around Z-axis and add Gaussian noise as described in the paper



Model

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

The result needs to be invariant to input points permutations and rigid transformation.



References:

[1]https://towardsdatascience.com/deep-learning-on-point-clouds-implementing-pointnet-in-google-colab-1fd65cd3a263
[2]