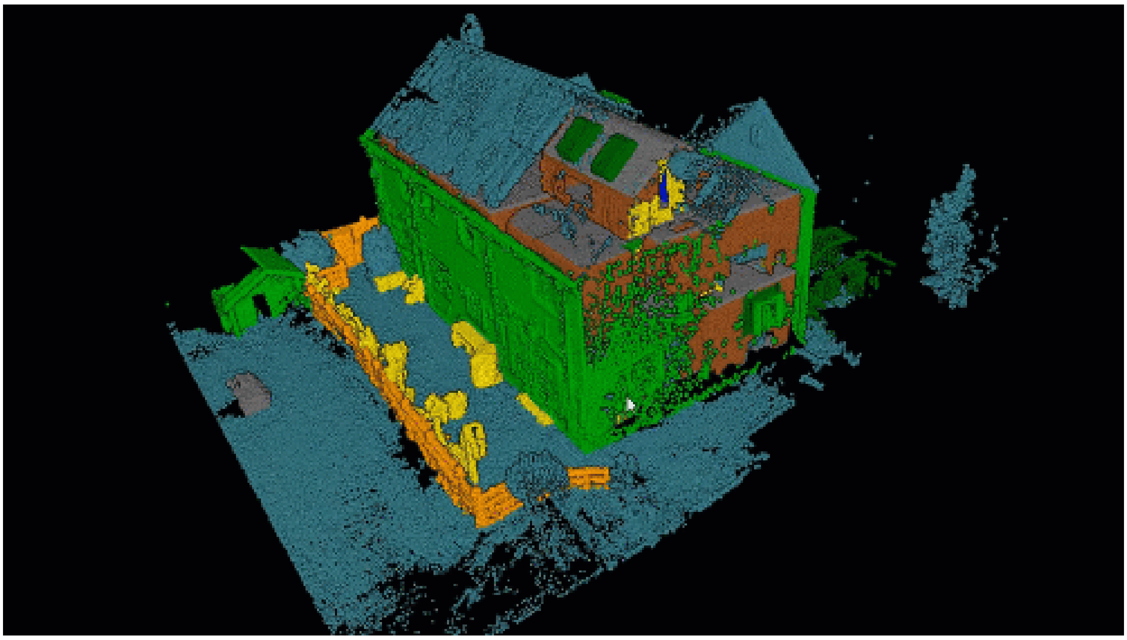
EE 046746 - Technion - Computer Vision

Dahlia Urbach

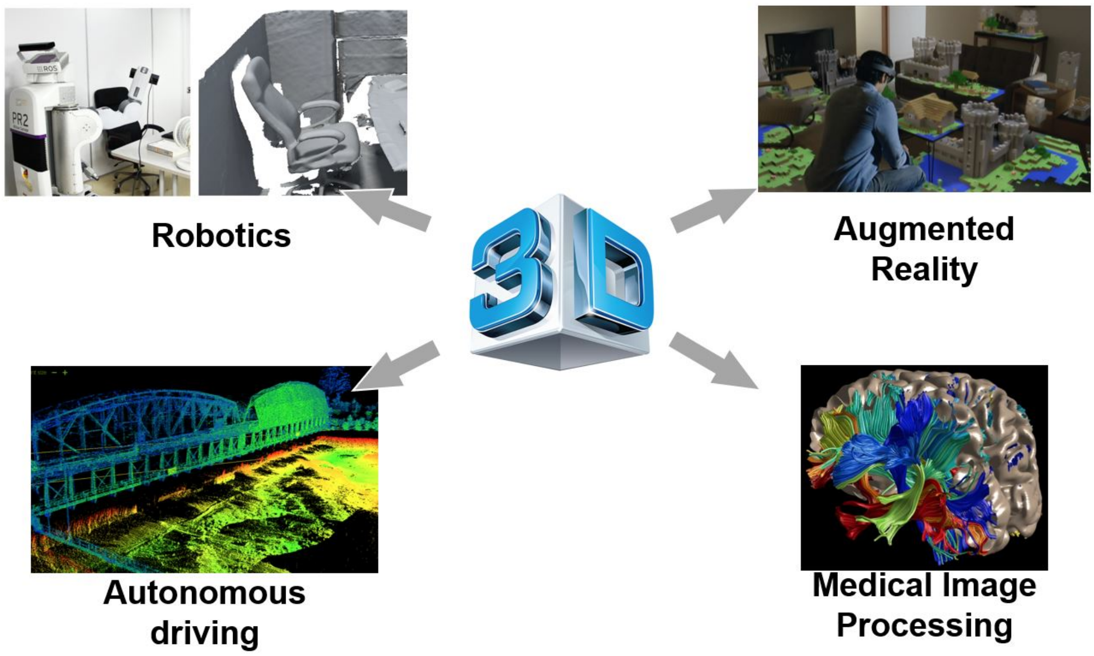
Tutorial 09 - Introduction to 3D Deep Learning



[Image source (https://towardsdatascience.com/the-future-of-3d-point-clouds-a-new-perspective-125b35b558b9)](https://towardsdatascience.com/the-future-of-3d-point-clouds-a-new-perspective-125b35b558b9)

[LiDAR point cloud support | Feature Highlight | Unreal Engine (https://www.youtube.com/watch?v=R-ZXdAEGbiw&feature=youtu.be)](https://www.youtube.com/watch?v=R-ZXdAEGbiw&feature=youtu.be)

Agenda




```
In [8]: # imports for the tutorial
        import numpy as np
        import matplotlib.pyplot as plt
        import time

        # pytorch imports
        import torch
        import torch.nn as nn
        import torch.nn.functional as F
        from torch.utils.data import DataLoader, Dataset, ConcatDataset
        # import torchvision
```

Depth Cameras

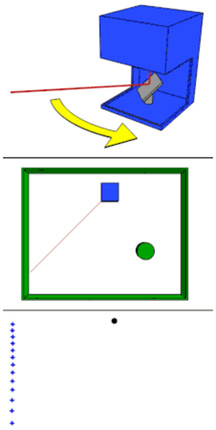
- Stereo Cameras - Next week
- Time of flight

Time of Flight Cameras

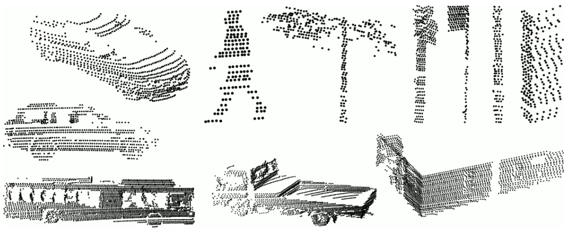
- Light travels at approximately a constant speed $c = 3 \times 10^8$ (meters per second).
- Measuring the time it takes for light to travel over a distance once can infer distance.
- Can be categorized into two types:
    1. Direct TOF - switch laser on and off rapidly.
    2. Indirect TOF - send out modulated light, then measure phase difference to infer depth.

1. Direct - TOF

- **Li**ght **D**etection **A**nd **R**anging (LiDAR) probably best example in computer vision and robotics.
- High-energy light pulses limit influence of background illumination.
- However, difficulty to generate short light pulses with fast rise and fall times.
- High-accuracy time measurement required.
- Prone to motion blur.
- Sparser as objects grow in distance.

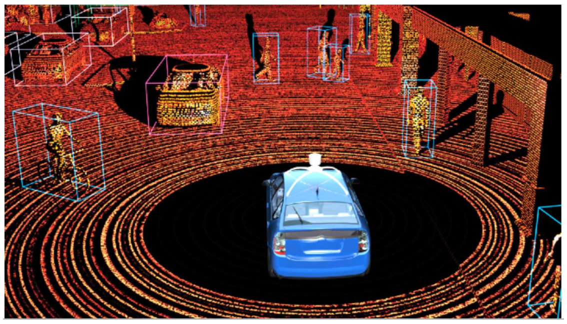


[Gif source - Wikipedia (https://en.wikipedia.org/wiki/Lidar)](https://en.wikipedia.org/wiki/Lidar)

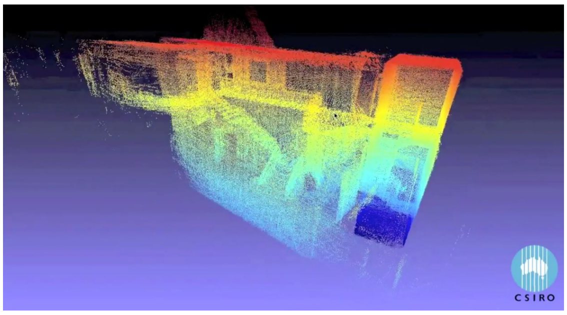


[Sydney Dataset (http://www.acfr.usyd.edu.au/papers/SydneyUrbanObjectsDataset.shtml)](http://www.acfr.usyd.edu.au/papers/SydneyUrbanObjectsDataset.shtml)

Autonomous Car - LiDAR



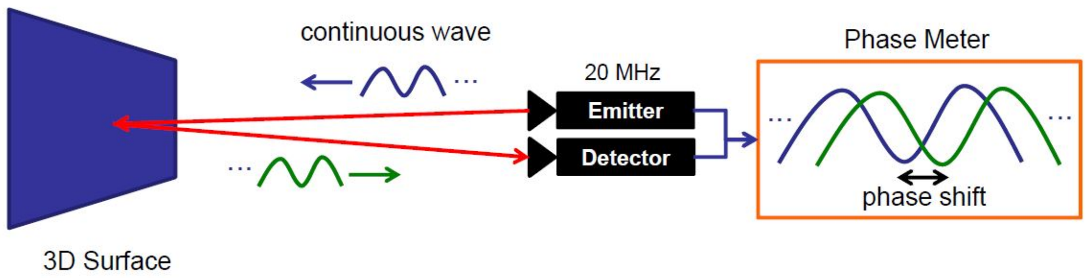
SLAM + LIDAT - Zebedee

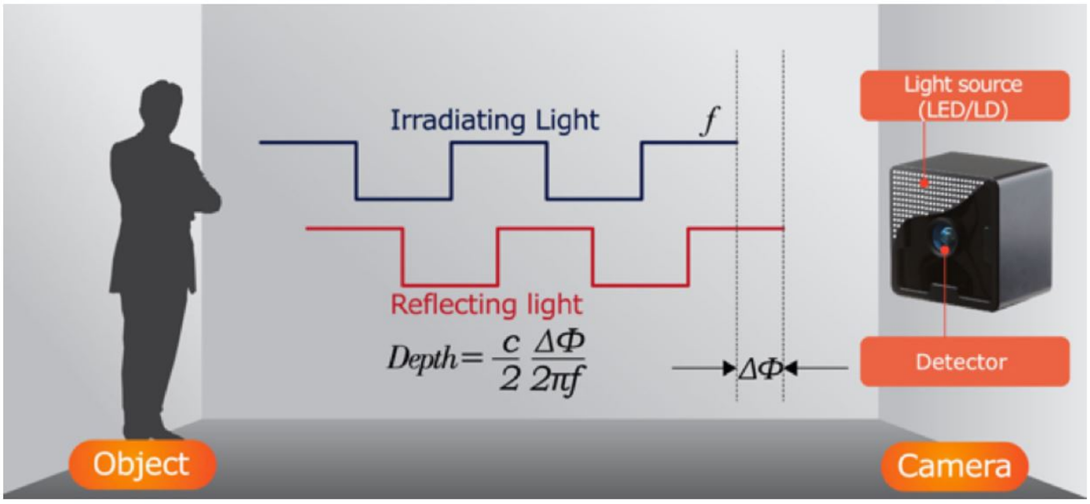


[zebedee (https://research.csiro.au/robotics/zebedee/)](https://research.csiro.au/robotics/zebedee/)

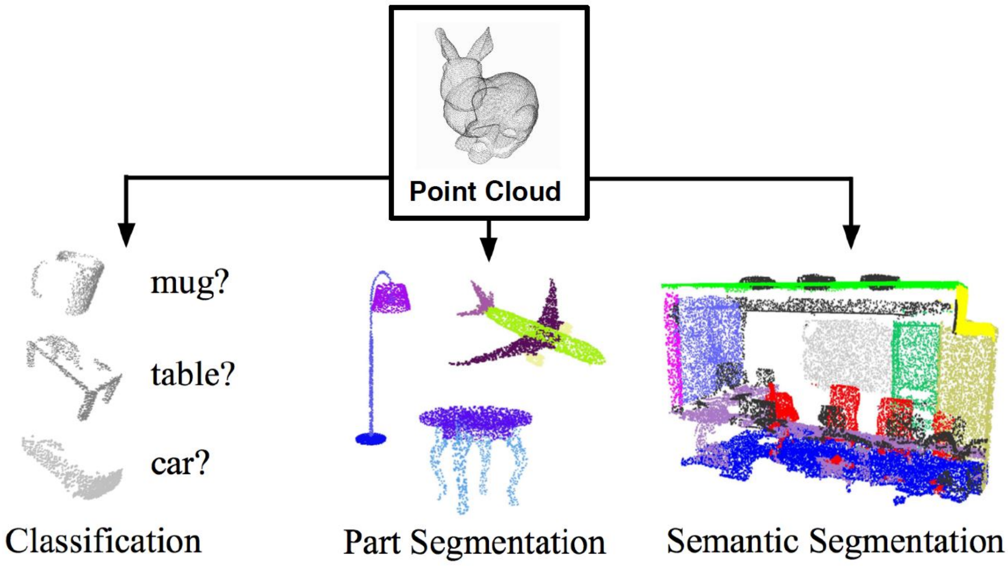
2. Indirect - TOF

- Continuous light waves instead of short light pulses.
- Modulation in terms of frequency of sinusoidal waves.
- Detected wave after reflection has shifted phase.
- Phase shift proportional to distance from reflecting surface.





Deep Learning on Point Clouds



- Calssification
- Semantic segmentation
- Part segmentation
  - Each point belongs to a specific part of the object
- ...

Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." CVPR. 2017.

Questions

- What are the differences between 2D image an a point cloud?
- Why it might be hard to feed a point cloud as neural network input?
- What are the benefits of using a point cloud?

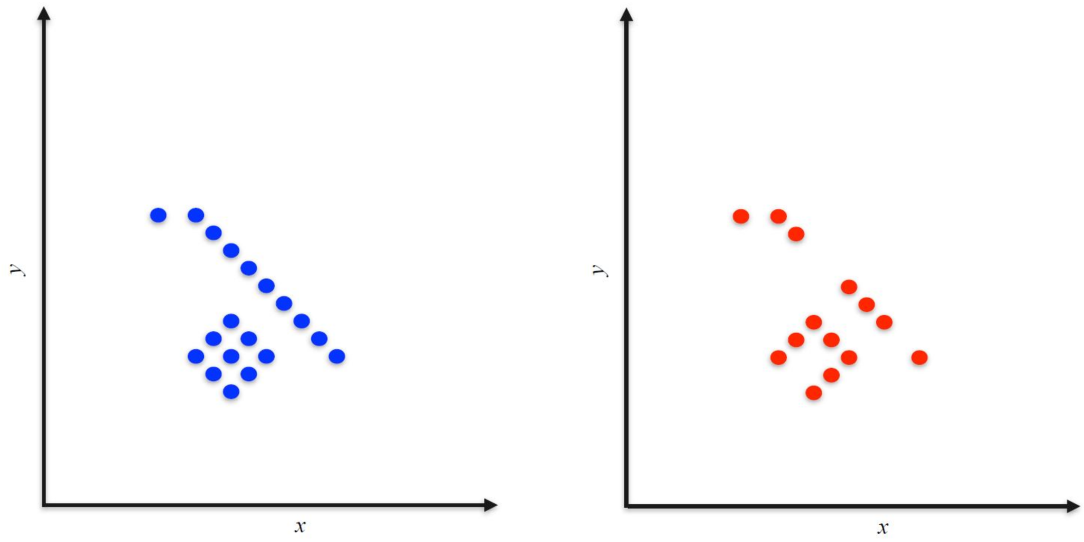
NA Point Clouds Problems

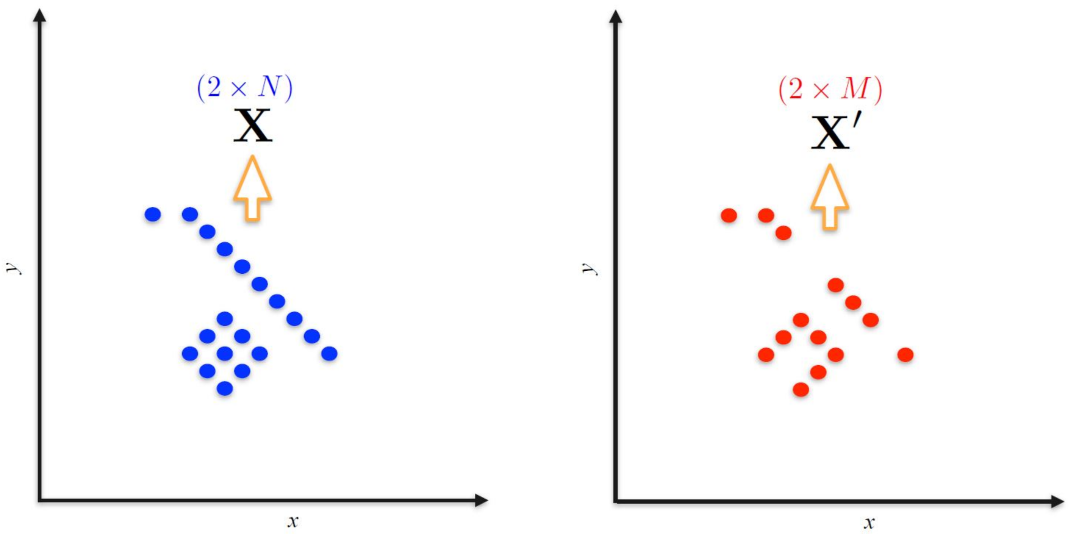
- Point Clouds Vary in Size (not constant)
- Unordered Input
  - Data is unstructured (no grid)
  - Data is invariant to point ordering (permutations)

Other Point Clouds Challenges

- Missing data
- Noise
- Rotations

Problem - Point Clouds Vary in Size

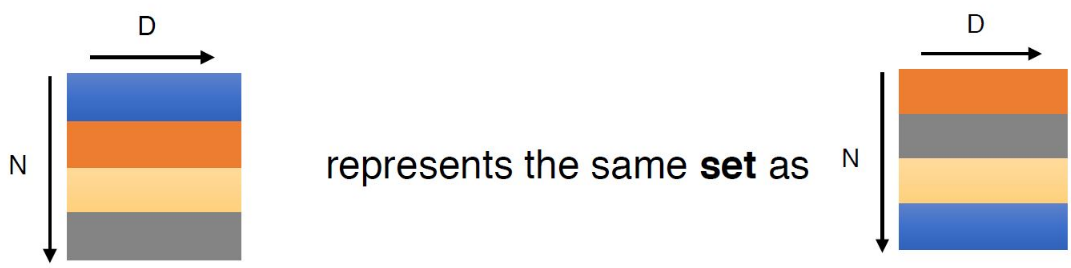




- Different point clouds represent the same object

Problem - Unordered Input

Point cloud: N **orderless** points, each represented by a D dim vector



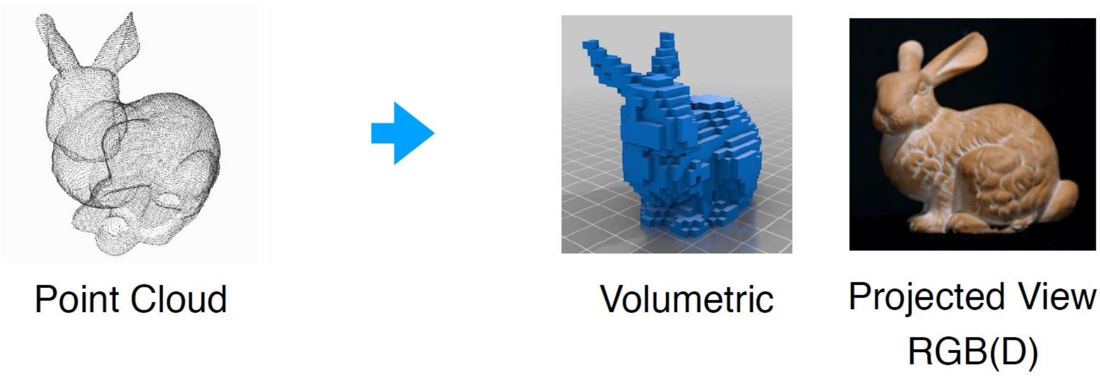
How many semi-equal representations?

Model needs to be invariant to $N!$ permutations

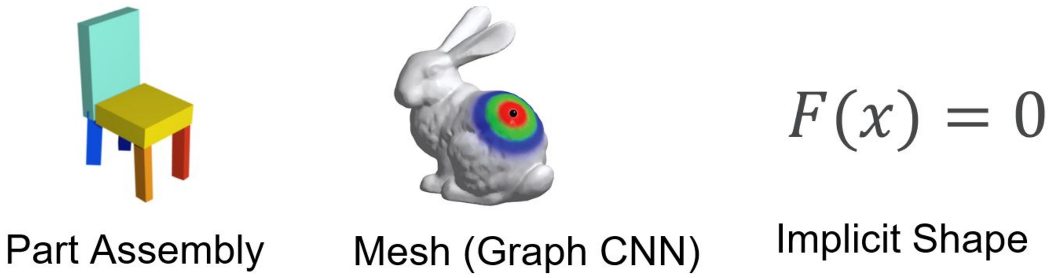
Alternate 3D Representations - Representations

Solution:

- Convert the raw point clouds into Voxels or multiple 2D RGB(D) images



Another 3D representation (not in this course):



Voxalization

Idea: generalize 2D convolutions to regular 3D grids

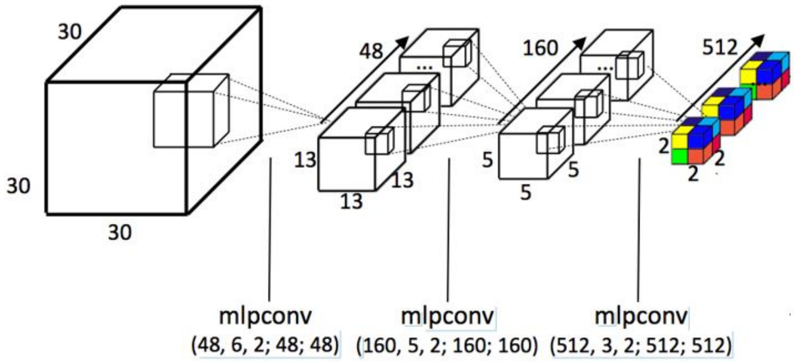
- The straightforward approach: transform the point clouds into a voxel grid by rasterizing and use 3D CNNs

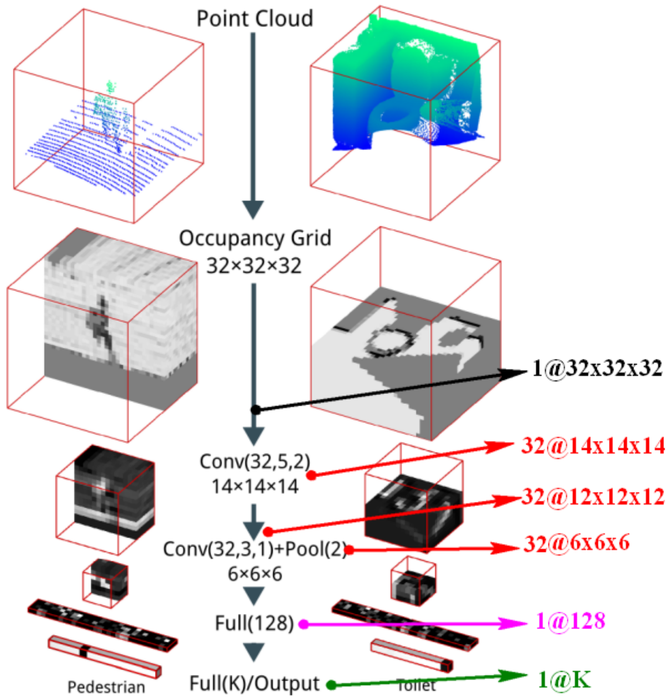
Voxel grid is a 3D grid of equal size volumes (voxels), can be occupied by:

- Binary 0/1 - Is there any point within the voxel?
- Weighted - The amount of point located within each voxel

Usually we use binary occupancy

3D convolution uses 4D kernels

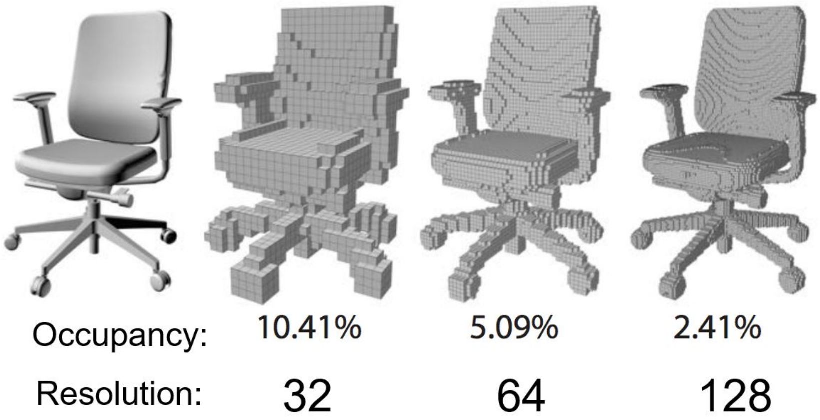




Maturana, Daniel, and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. IROS, 2015.

Voxalization Problems

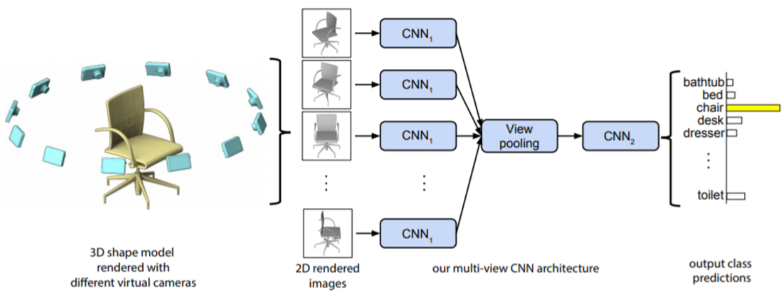
- Large memory cost
- Slow processing time
- Limited spatial resolution
- Quantization artifacts



Multi-View Approach

Idea: Transfrom the problem into a well known domain (3D→2D)

- The multi-view approach: project multiple views to 2D and use CNN to process
  - How many views do we need? (Another hyper parameter)



CNN_1 - We can use pre-trained networks to extract features followed by fine tune layers

H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multiviewconvolutional neural networks for 3d shape recognition. CVPR, 2015.

Apply Deep Learning Directly on 3D Point Clouds

Idea: Most of the raw 3D data are point clouds - Solve the problems!

Note: Point Clouds Problems:

- Point Clouds Vary in Size (not constant)
- Unordered Input
  - Data is unstructured (no grid)
  - Data is invariant to point ordering (permutations)

PointNet

Permutation Invariance: Symetric Function

$$f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), x_i \in R^D$$

π is a different permutation

Example:

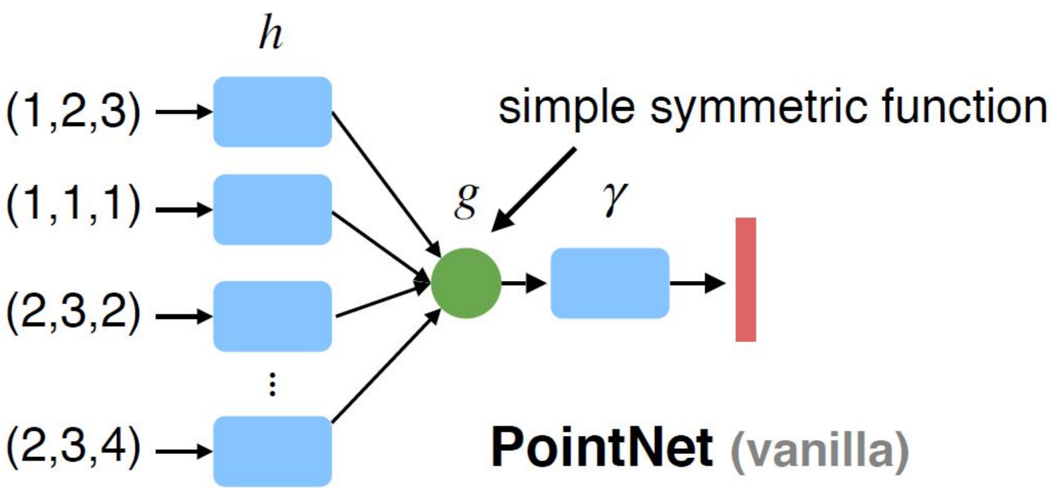
$$f(x_1, x_2, \ldots, x_n) = max\{x_1, x_2, \ldots, x_n\}$$
$$f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \cdots + x_n$$

- How can we construct a family of symmetric functions by neural networks?

Observe:

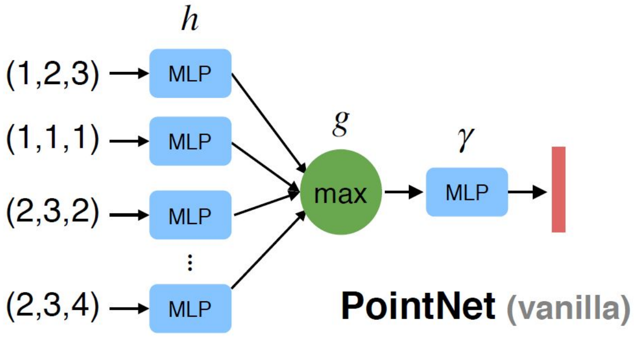
$$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$$

is symmetric if g is symmetric



Basic PointNet Architecture

Empirically, we use multi-layer perceptron (MLP) and max pooling:



Input MLP:

$$h(x_i) : R^3 \to R^D$$

We can look at it as D functions $\{h_k\}_{k=1}^{D}$ operate on each point where

$$h_k(x_i) : R^3 \to R^1$$

Pooling layer:

$$g(h(x_1), \ldots, h(x_n)) : R^{N \times D} \to R^D$$

We apply the pooling over all points for each function h_k.

$$g(h_k(x_1), \ldots, h_k(x_n)) : R^{N \times 1} \to R^1$$

Classification MLP:

$$\gamma \circ g(h(x_1), \ldots, h(x_n)) : R^D \to R^{D_{NumClasses}}$$

Shared MLP implementation "trick":

- Use conv layers : Number of filters C_{out}, each filter size is $1 \times C_{in}$.
- Input: $R^{N \times C_{in}}$
- Output: $R^{N \times C_{out}}$

MLP:

$$h : R^{C_{in}} \to R^{C_1} \to \cdots \to R^{C_{out}}$$

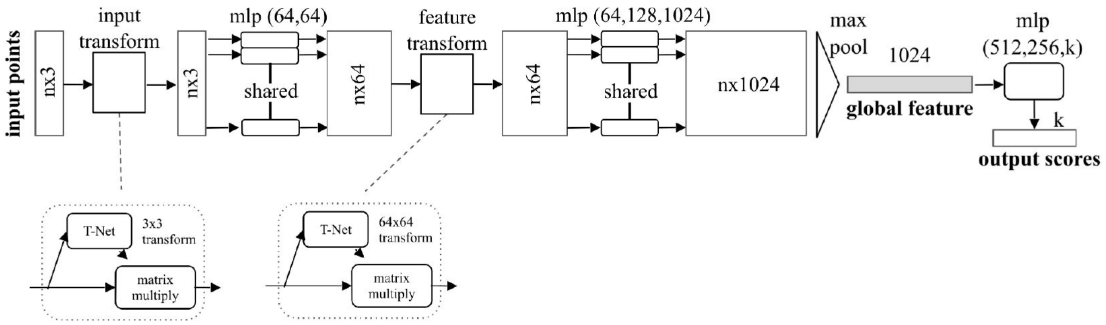
In [7]:
```
class Tnet(nn.Module):
    def __init__(self, k=3):
        super().__init__()
        self.k=k
        self.conv1 = nn.Conv1d(k,64,1)
        self.conv2 = nn.Conv1d(64,128,1)
        self.conv3 = nn.Conv1d(128,1024,1)
        self.fc1 = nn.Linear(1024,512)
        self.fc2 = nn.Linear(512,256)
        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)))

        #initialize as identity
        init = torch.eye(self.k, requires_grad=True).repeat(bs,1,1)
        if xb.is_cuda:
        init=init.cuda()
        matrix = self.fc3(xb).view(-1,self.k,self.k) + init
        return matrix


class Transform(nn.Module):
    def __init__(self):
        super().__init__()
        self.input_transform = Tnet(k=3)
        self.feature_transform = Tnet(k=64)
        self.conv1 = nn.Conv1d(3,64,1)

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


        self.bn1 = nn.BatchNorm1d(64)
        self.bn2 = nn.BatchNorm1d(128)
        self.bn3 = nn.BatchNorm1d(1024)

    def forward(self, input):
        matrix3x3 = self.input_transform(input)
        # batch matrix multiplication
        xb = torch.bmm(torch.transpose(input,1,2), matrix3x3).transpose(1,2)

        xb = F.relu(self.bn1(self.conv1(xb)))

        matrix64x64 = self.feature_transform(xb)
        xb = torch.bmm(torch.transpose(xb,1,2), matrix64x64).transpose(1,2)

        xb = F.relu(self.bn2(self.conv2(xb)))
        xb = self.bn3(self.conv3(xb))
        xb = nn.MaxPool1d(xb.size(-1))(xb)
        output = nn.Flatten(1)(xb)
        return output, matrix3x3, matrix64x64

class PointNet(nn.Module):
    def __init__(self, classes = 10):
        super().__init__()
        self.transform = Transform()
        self.fc1 = nn.Linear(1024, 512)
        self.fc2 = nn.Linear(512, 256)
        self.fc3 = nn.Linear(256, classes)


        self.bn1 = nn.BatchNorm1d(512)
        self.bn2 = nn.BatchNorm1d(256)
        self.dropout = nn.Dropout(p=0.3)
        self.logsoftmax = nn.LogSoftmax(dim=1)

    def forward(self, input):
        xb, matrix3x3, matrix64x64 = self.transform(input)
        xb = F.relu(self.bn1(self.fc1(xb)))
        xb = F.relu(self.bn2(self.dropout(self.fc2(xb))))
```

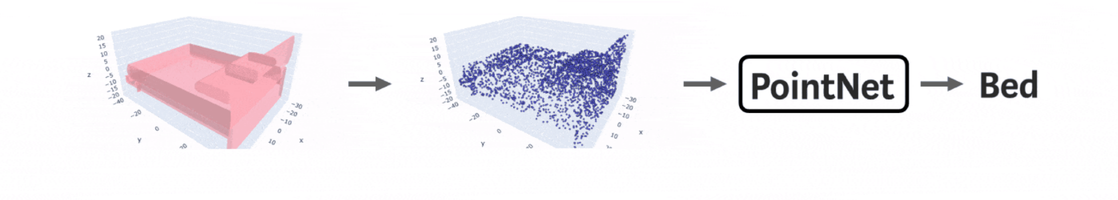
```
        output = self.fc3(xb)
        return self.logsoftmax(output), matrix3x3, matrix64x64
```

[Code source (https://github.com/nikitakaraevv/pointnet/blob/master/nbs/PointNetClass.ipynb)](https://github.com/nikitakaraevv/pointnet/blob/master/nbs/PointNetClass.ipynb) - Full implementation of PointNet Classification (can be opened in Colab)

[Deep Learning on Point clouds: Implementing PointNet in Google Colab (https://towardsdatascience.com/deep-learning-on-point-clouds-implementing-pointnet-in-google-colab-1fd65cd3a263)](https://towardsdatascience.com/deep-learning-on-point-clouds-implementing-pointnet-in-google-colab-1fd65cd3a263) - Nikita Karaev

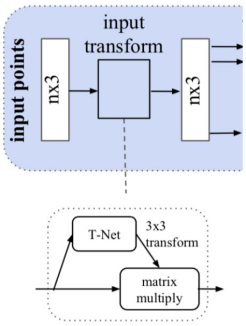
PointNet Classification Network





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

Transformation Invariance



Learn transformation matrix to improve task performance. We want our network to be invariante to rigid transformation of the object.

- Practically, more augmentation over the training dataset also solved the transformation invariance

[Image source (https://medium.com/@luis_gonzales/an-in-depth-look-at-pointnet-111d7efdaa1a)](https://medium.com/@luis_gonzales/an-in-depth-look-at-pointnet-111d7efdaa1a)

Qi, Charles R., et al. "PointNet: Deep learning on point sets for 3d classification and segmentation." CVPR 2017.

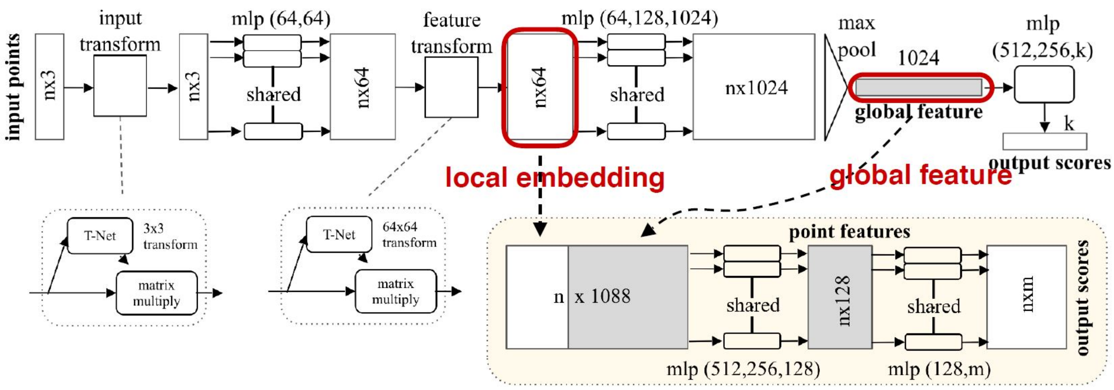
Results - Classification

3D CNNs

	input	#views	accuracy avg. class	accuracy overall
SPH [12]	mesh	-	68.2	
3DShapeNets [29]	volume	1	77.3	84.7
VoxNet [18]	volume	12	83.0	85.9
Subvolume [19]	volume	20	86.0	**89.2**
LFD [29]	image	10	75.5	-
MVCNN [24]	image	80	**90.1**	-
Ours baseline	point	-	72.6	77.4
Ours PointNet	point	1	86.2	**89.2**

Qi, Charles R., et al. "PointNet: Deep learning on point sets for 3d classification and segmentation." CVPR 2017.

PointNet Segmentation Network



- Extract local features - Describes each point seperatly
- Extract global feature - Describes the entire point cloud
- Concatenate the local and global features and feed it into a shared MLP - The MLP learns to process the point feature acording to a condition. The condition is described by the global feature vectore.

[Code (https://github.com/nikitakaraevv/pointnet/blob/master/nbs/PointNetSeg.ipynb)](https://github.com/nikitakaraevv/pointnet/blob/master/nbs/PointNetSeg.ipynb) - Full implementation of PointNet Segmentation (can be opened in Colab)

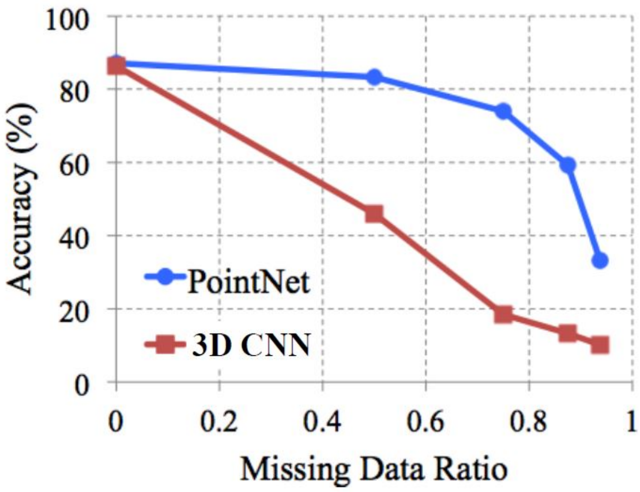
Qi, Charles R., et al. "PointNet: Deep learning on point sets for 3d classification and segmentation." CVPR 2017.

Semantic Scene Parsing

Qi, Charles R., et al. "PointNet: Deep learning on point sets for 3d classification and segmentation." CVPR 2017.

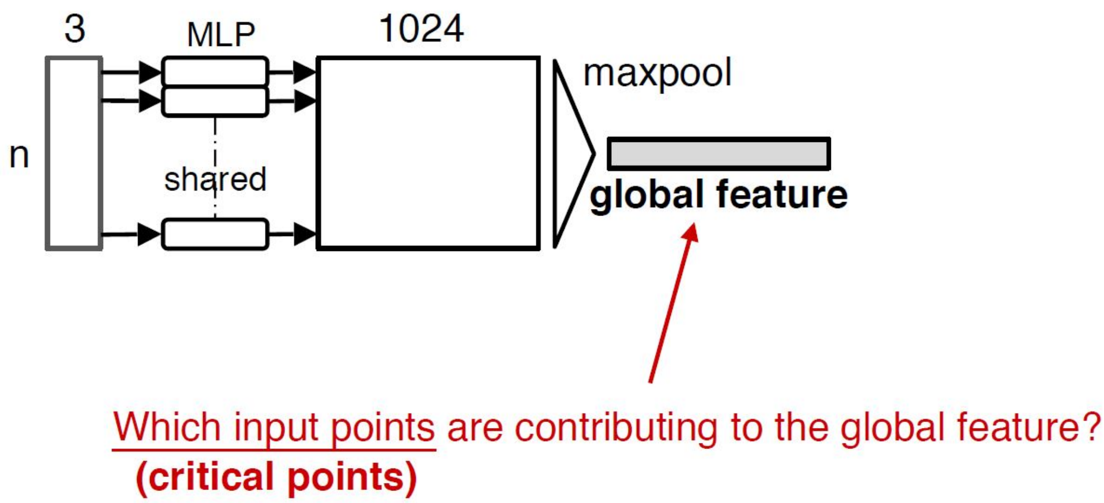
Results - Robustness to Missing Data (Classification example)

- Why is PointNet so robust to missing data?



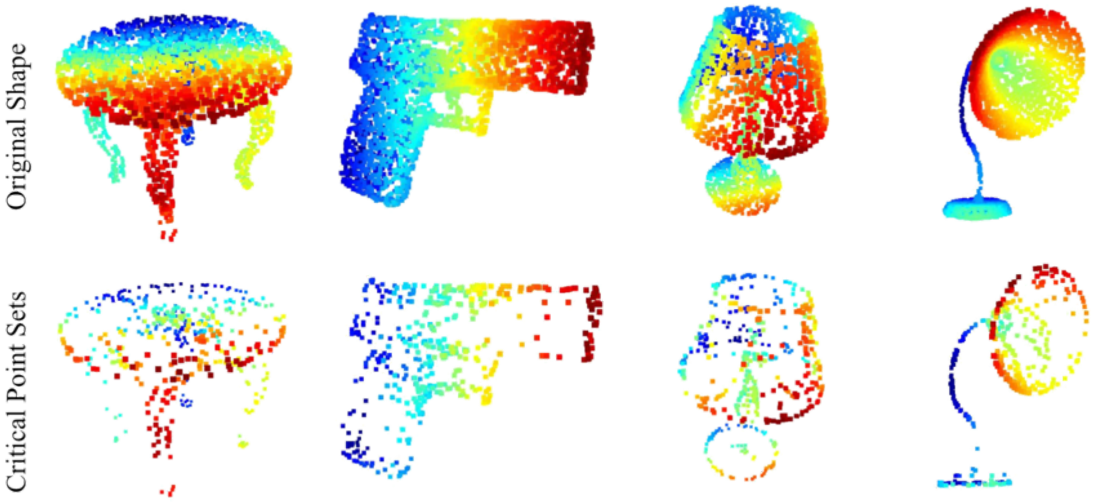
Qi, Charles R., et al. "PointNet: Deep learning on point sets for 3d classification and segmentation." CVPR 2017.

Visualizing Global Point Cloud Features



Visualize What is Learned by Reconstruction

- Salient points are discovered!



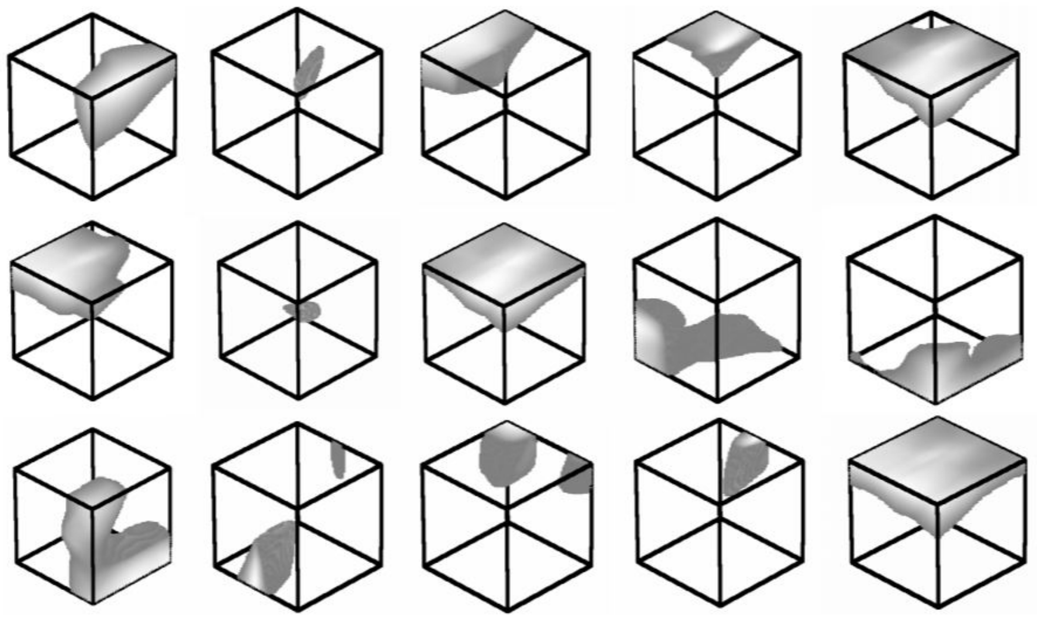
The "critical points" are those who influenced the global feature vector, a.k.a the pooling layer. The "critical" object's geometry structured is reserved.

Qi, Charles R., et al. "PointNet: Deep learning on point sets for 3d classification and segmentation." CVPR 2017.

Point function visualization

For each per-point function h (MLP), calculate the values of $h(p)$ for all the points p in the cube.

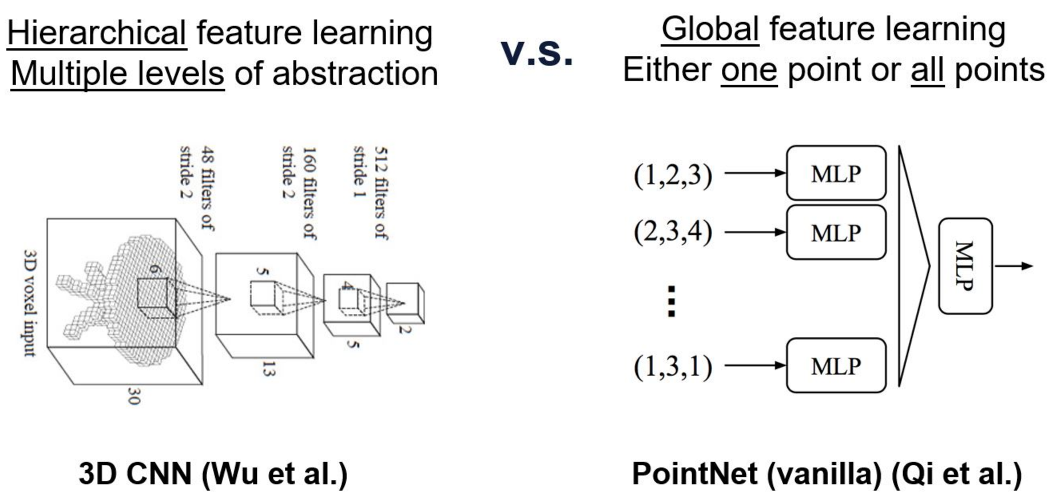
Random 15 function out of the 1024 learned functions:



- Semi-equivalent to filter response in CNNs

Qi, Charles R., et al. "PointNet: Deep learning on point sets for 3d classification and segmentation." CVPR 2017.

NA Limitations of PointNet

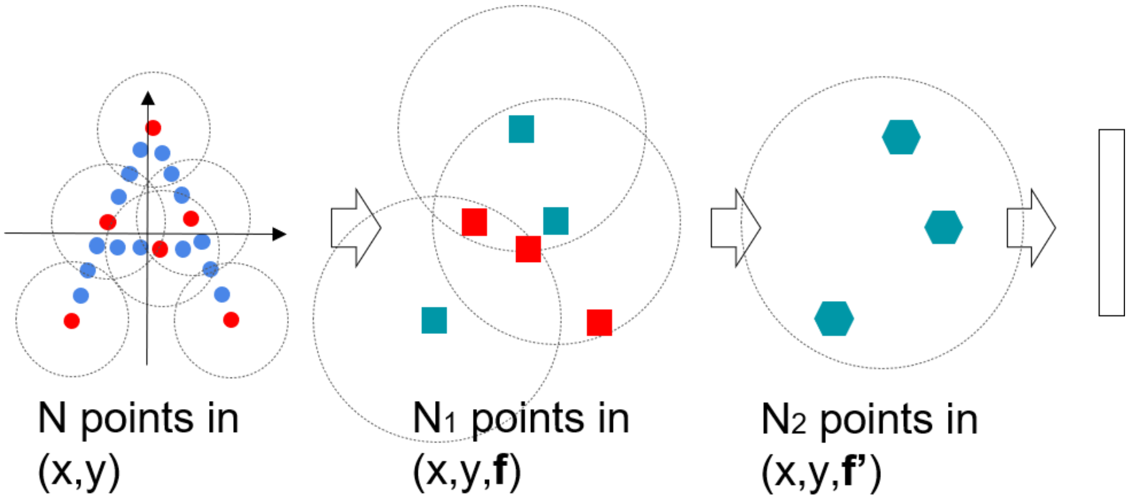


- No local context for each point
- Global feature depends on **absolute** coordinate. Hard to generalize to unseen scene configurations

Points in Metric Space

- Learn "kernels" in 3D space and conduct convolution
- Kernels have compact spatial support
- For convolution, we need to find neighboring points
- Possible strategies for range query
  - Ball query (results in more stable features)
  - k-NN query (faster)

PointNet v2.0: Multi-Scale PointNet



Repeated layers:

- Sample anchor points
- Find neighborhood of anchor points
- Apply PointNet in each neighborhood to mimic convolution

Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." Advances in neural information processing systems. 2017.

More Point Clouds DL solutions:

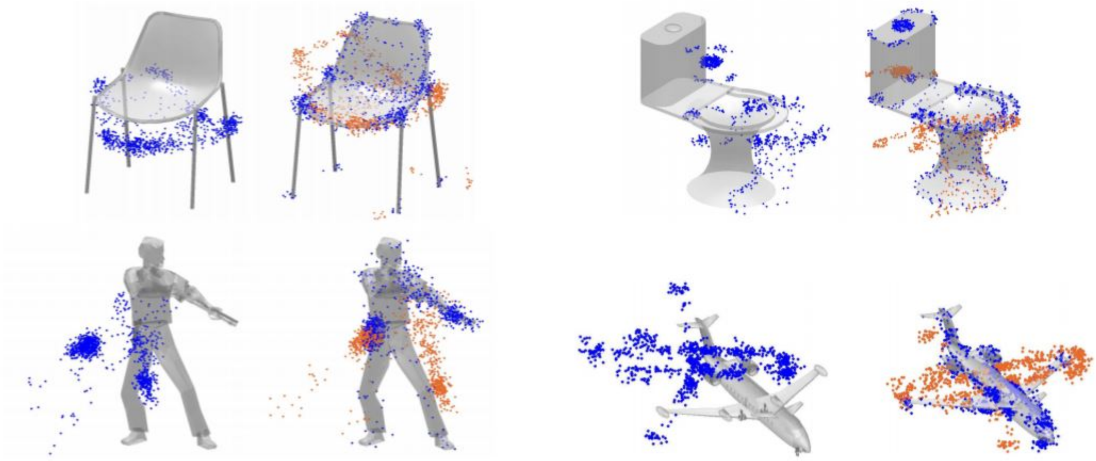
- 3DmFV
- Dynamic Graph CNN
- PCNN
- PointCNN
- KPConv

3D Deep Learning Applications

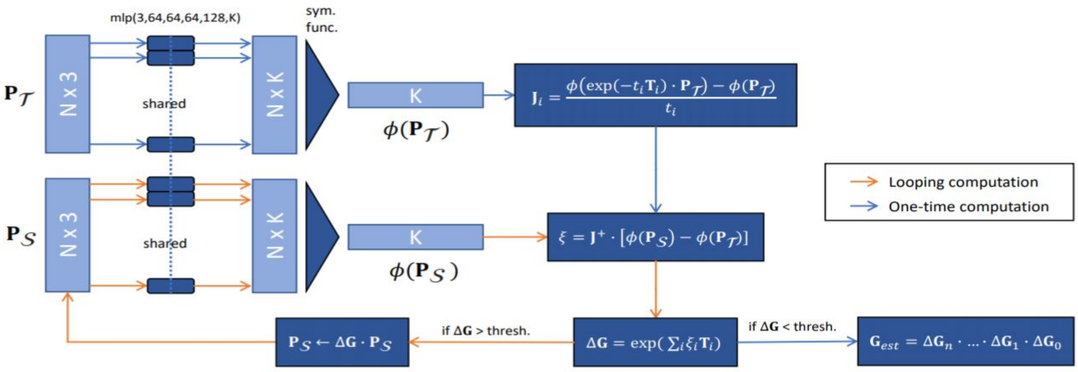
- Calssification (V)
- Semantic segmentation (V)
- Part segmentation
- Object detection (Upcoming)
- Reconstraction
- Generation (Upcoming)
- Registration (Upcoming)
- Sampling - Downsampling, Upsampling
- SLAM
- Normal Estimation
- and many more...

Registration:

Problem statment: Find the rotation and translation transformation between objects



- PointNetLK (blue) - Deep Learning, based on Lucas–Kanade method (Tracking lecture)
    - Comparing 2 point clouds using PointNet features
- ICP (orange) - Classic registration method



Both inputs (target and source) are being processed by PointNet architecture

Aoki, Yasuhiro, et al. "PointNetLK: Robust & efficient point cloud registration using PointNet." CVPR 2019.

Generation

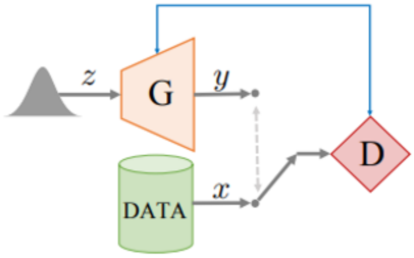
Conditional generation

Free generation

How would you build a point cloud GAN?

Learning Representations and Generative Models for 3D Point Clouds (Achlioptas et al.)

- FC layer as generator
- PointNet as discriminator



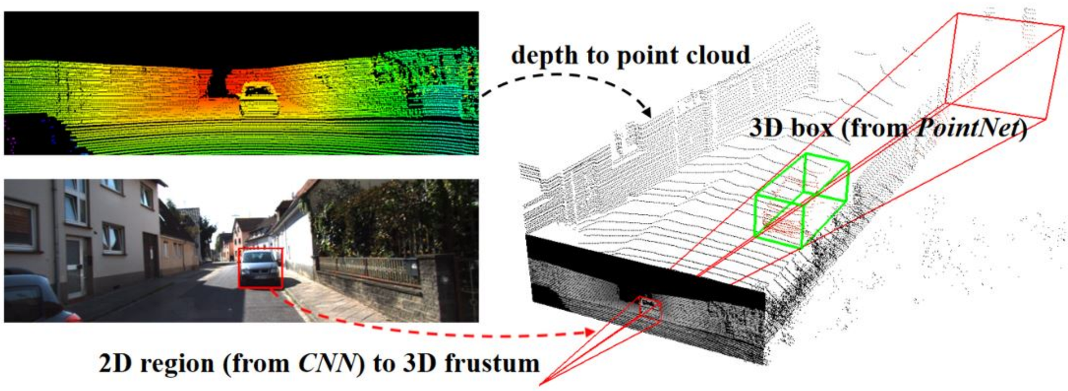
Achlioptas et al., “Learning Representations and Generative Models for 3D Point Clouds”, ICML 2018

More generation methods:

- AtlasNet
- FoldingNet
- PointFlow
- OccupancyNetworks
- DeepSDF
- ...

Detection:

- Generate object proposals from a view (e.g., using SSD)
- Recognize using PointNet



Qi et al., “Frustum PointNets for 3D Object Detection from RGB-D Data”, CVPR 2018

Questions

- What are the differences between 2D image an a point cloud?
  - Unstructured
  - Vary number of points
  - Unordered
- Why it might be hard to feed a point cloud as neural network (NN) input?
  - Does not rely on a grid
  - Does not has a fix size
  - Different permutation represent the same point cloud

All three diffrences influence directly the abbility of using NN!

- What are the benefits of using a point cloud?
  - Most sensors raw outputs are point clouds (LiDAR)
  - Very efficient representation of 3D data (no empty voxels)
  - Reserve geometric details (no quantization)

Recommended Tools

Python:

- Open3D
- trimesh
- Ipyvolume - Visualization for Notebooks

Deep Learning:

- Python3D
- Kaolin (Pytorch)
- TensorFlow Graphics

Visualize Tools (drop and view):

- CloudCompare
- MeshLab

For more 3D deep learnig frameworks and datasets:

- awesome-point-cloud-analysis (https://github.com/Yochengliu/awesome-point-cloud-analysis)
- 3D-Machine-Learning#datasets (https://github.com/timzhang642/3D-Machine-Learning#datasets)

Datasets:

- ModelNet
- ShapeNet
- PartNet
- Sydney Urban Opject DAtaset
- Stanford 3D
- KITTI
- ...

Recommended Videos

Warning!

- These videos do not replace the lectures and tutorials.
- Please use these to get a better understanding of the material, and not as an alternative to the written material.

Video By Subject

- 3D Deep Learning
    - General (Both highly recomanded):
        - 3D Deep Learning Tutorial from SU lab at UCSD (https://www.youtube.com/watch?time_continue=6&v=vfL6uJYFrp4&feature=emb_logo) - Hao Su
        - Geometric deep learning (https://www.youtube.com/watch?v=wLU4YsC_4NY_o) - Micahel Bronstein
    - PointNet (https://www.youtube.com/watch?v=Cge-hot0Oc0&t=24s)
    - 3DmFV (https://www.youtube.com/watch?v=HIUGOKSLTcE)

Credits

- Slides - Yizhak (Itzik) Ben-Shabat (http://www.itzikbs.com/category/research-blog), Simon Lucey (CMU) (https://ci2cv.net/people/simon-lucey/), Hao Su, Jiayuan Gu and Minghua Liu(UCSanDiego) (https://cseweb.ucsd.edu/~haosu/)
- Multiple View Geometry in Computer Vision - Hartley and Zisserman - Sections 9,10
- Computer Vision: Algorithms and Applications (https://www.springer.com/gp/book/9781848829343) - Richard Szeliski - Sections 11,12
- Icons from Icon8.com (https://icons8.com/) - https://icons8.com (https://icons8.com)