COMPUTER VISION (SPRING-2020)

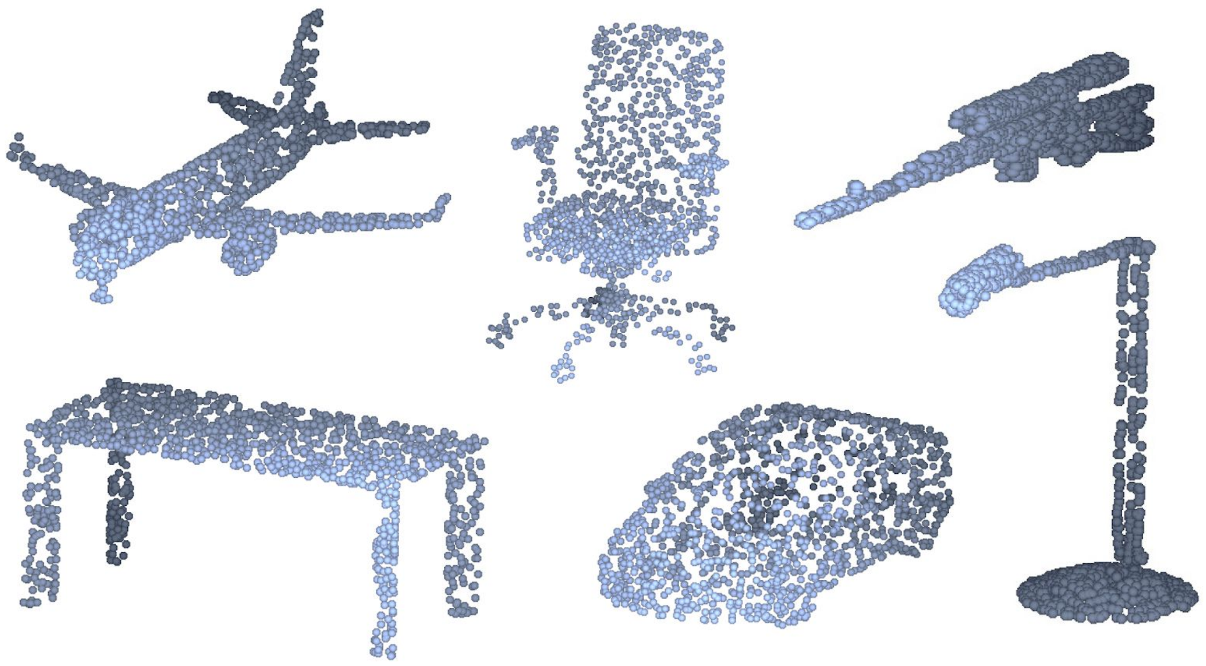
NEURAL NETWORK FOR 3D OBJECT CLASSIFICATION

Project Progress Report

Team ID: 2

Ekta Gavas (2019701005)

Surabhi Gupta (2019701024)



- **Problem Statement**

  To solve 3D object classification on the 3D domain and evaluating our method by comparing it with previous results. A convolutional neural network combining spatial transformation network is used to classify 3D objects in a subset of ModelNet. The spatial transformation network is an attempt to deal with the rotation invariance problem in 3D object classification.

- **Framework: Pytorch**

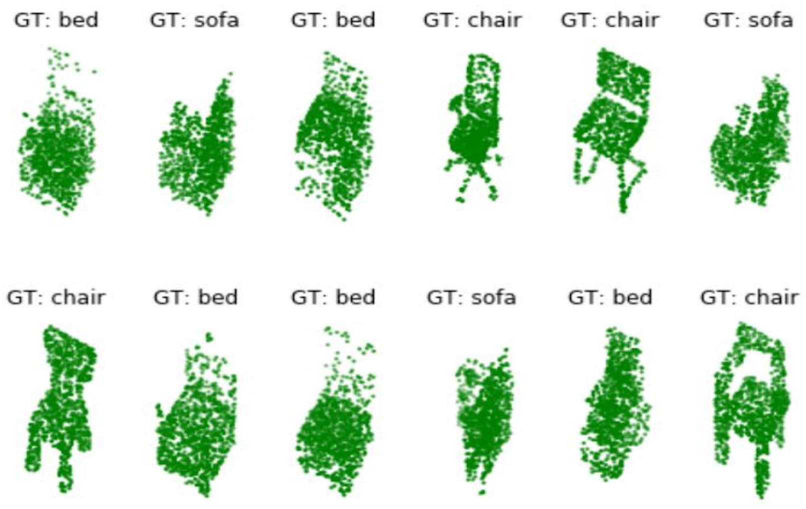
  The whole project so far is developed using Pytorch framework along with other libraries, including but not limited to numpy, matplotlib, sklearn, scipy, etc.

- **ModelNet Dataset**

- The dataset contains files in .off format which is not directly read by Python. For this, a utility named 'binvox' is used. It is a straight-forward program that reads a 3D model file, rasterizes it into a binary 3D voxel grid, and writes the resulting voxel file.

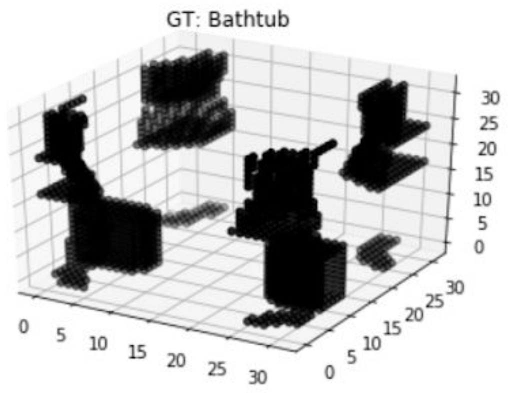
- It is used to convert .off files into numpy arrays and then the arrays are stored in a file that can later be loaded with numpy in Python.

- We also explored the interconversions of this data from triangular meshes to point clouds and to voxels by visualizations using Nvidia's library Kaolin. We noticed that the voxel representations were not desirable as the 3D object was flipped and broken randomly as compared to its corresponding point cloud representations.



Point cloud representations

Voxel representation generated with Kaolin

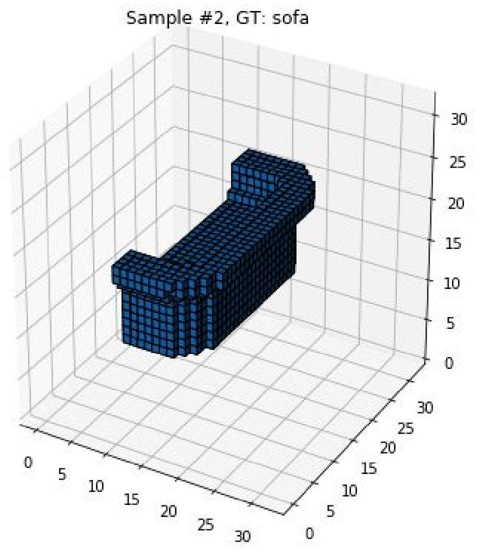


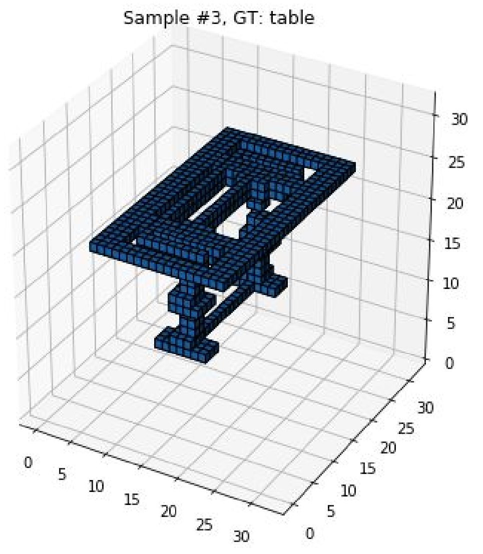
- **Visualizing the Dataset**

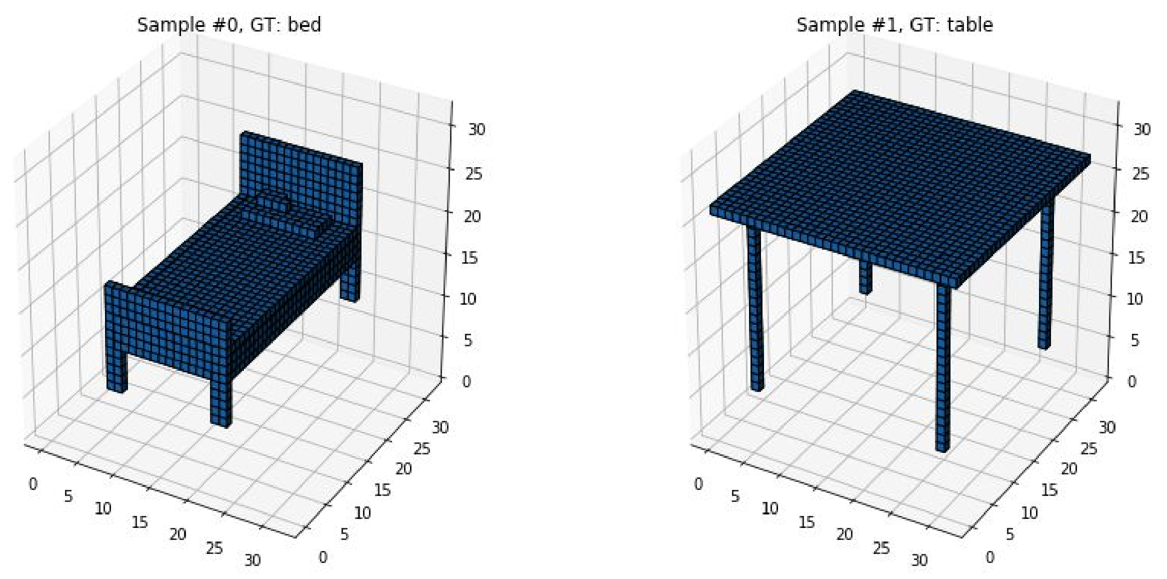
  ModelNet contains 3D CAD models used to train the deep network for 3D deep learning

  - **ModelNet 10**
    - *contains 4899 shapes covering 10 common categories.*
    - *3991 train + 908 test samples*

Voxelized representations







- We define the ***ModelNetDataset*** class which loads the data into numpy array from files and contains methods to get a sample from the dataset and also visualize the voxel grid.

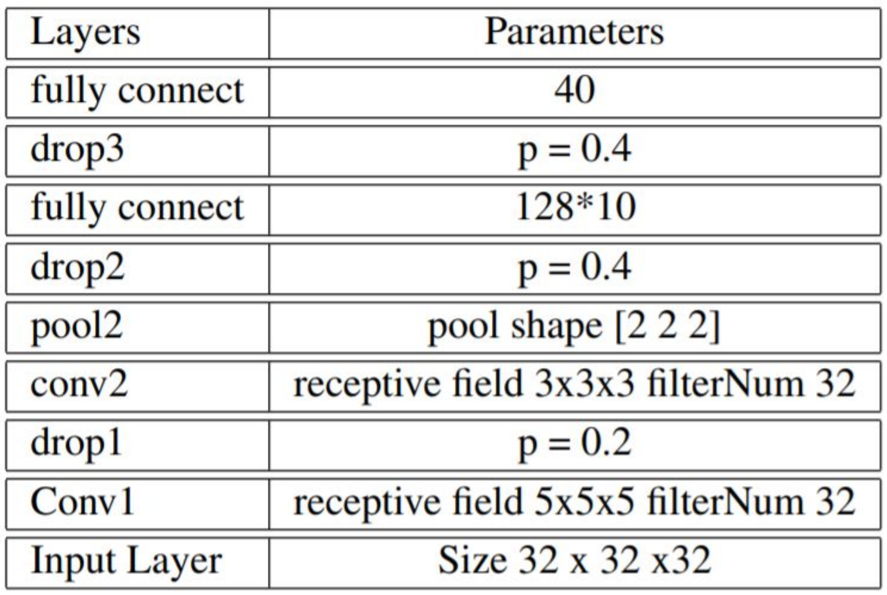
- ## Framework: Pytorch

- The whole project so far is developed using Pytorch framework along with other libraries, including but not limited to numpy, matplotlib, sklearn, scipy, etc.

- ## Voxnet Architecture

- VoxNet is a 3D CNN architecture for efficient and accurate object detection.

- VoxNet achieves accuracy beyond the state of the art while labeling hundreds of instances per second.

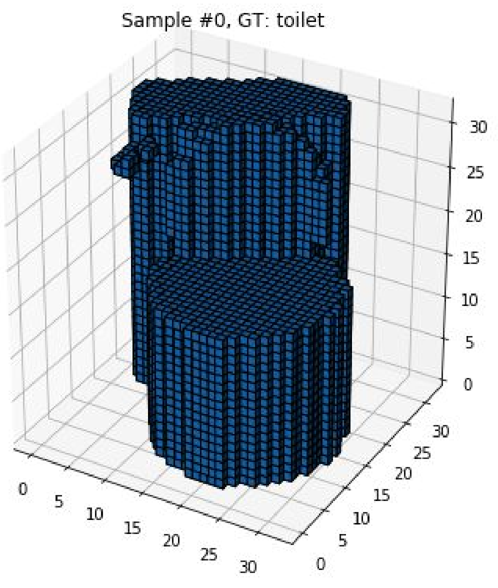


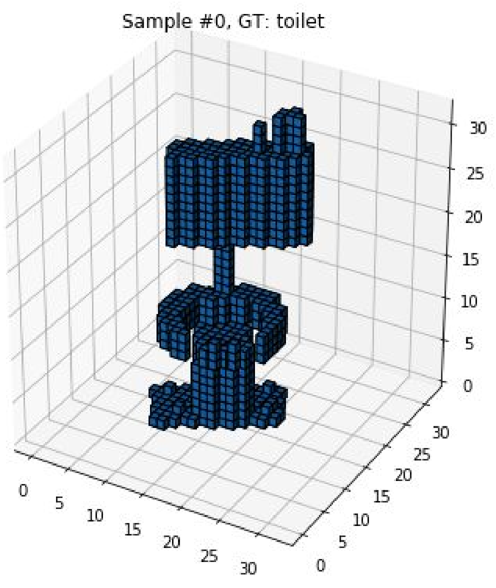
Layers	Parameters
fully connect	40
drop3	p = 0.4
fully connect	128*10
drop2	p = 0.4
pool2	pool shape [2 2 2]
conv2	receptive field 3x3x3 filterNum 32
drop1	p = 0.2
Conv1	receptive field 5x5x5 filterNum 32
Input Layer	Size 32 x 32 x32

- We define ***VoxNet*** class with the mentioned architecture and implement the forward function for the neural network to calculate the weights. As there are 10 categories, the output layer contains 10 units.
- We also define ***Rotate*** class which implements a caller for transforming a sample by rotating it by given angle and along a specified axis. multi_rotate argument is useful for data augmentation where each object is rotated n number of times to cover 360-degree rotation. This comes into the picture while loading the data using data loader which applies transform and converts it to Tensor.
- ***ToTensor*** class is just a utility class for converting ndarrays to numpy and can also be used as a transform class. Pytorch by default converts the data to tensor with data loader so, it is not currently used.

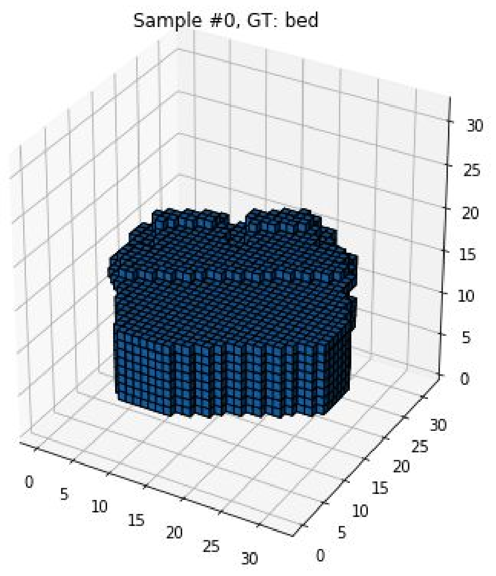
● Baseline Voxnet (0/1 representation)

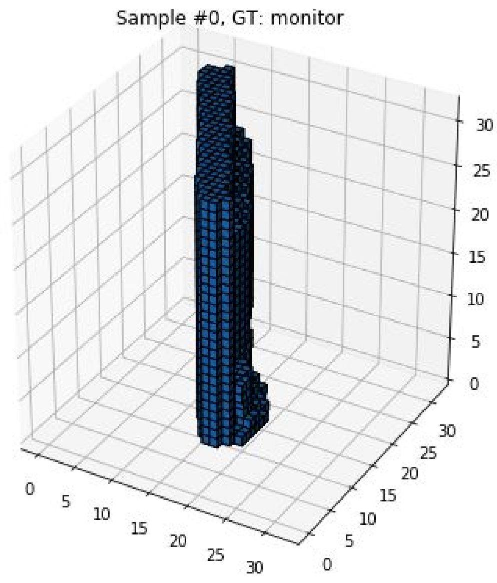
Training Set *(augmented by rotating every sample by 30 degrees)*





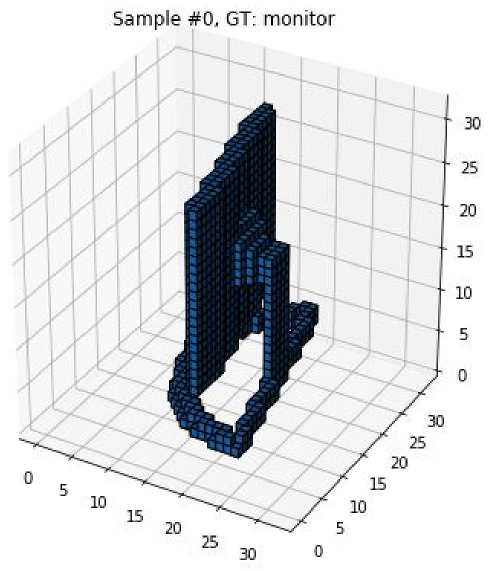
Testing Set-1 *(augmented by rotating every sample by 30 degrees)*

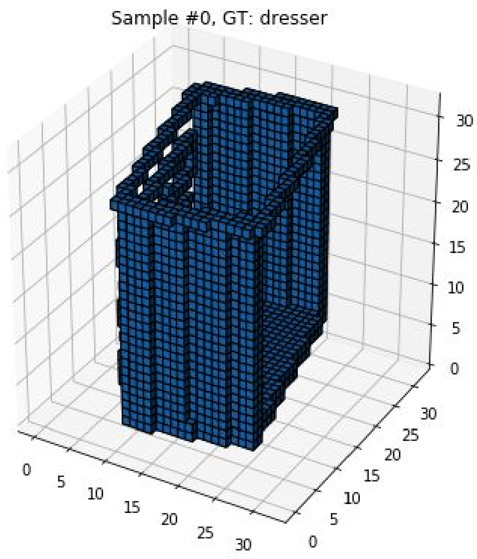


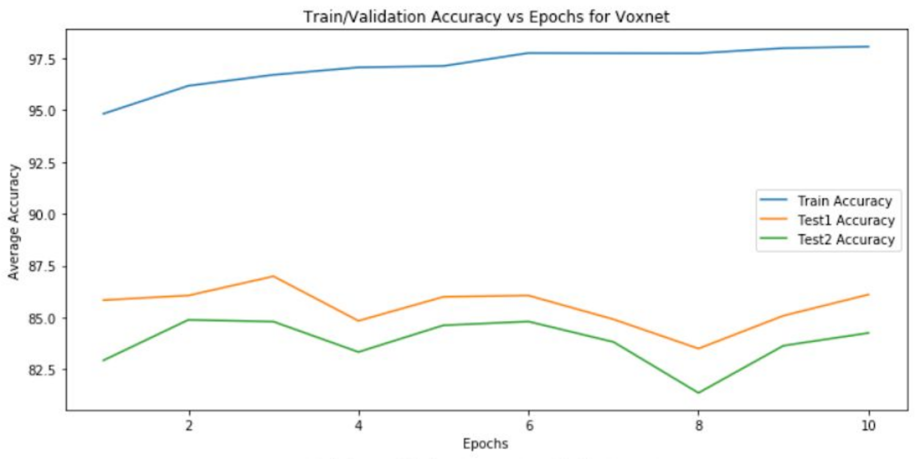


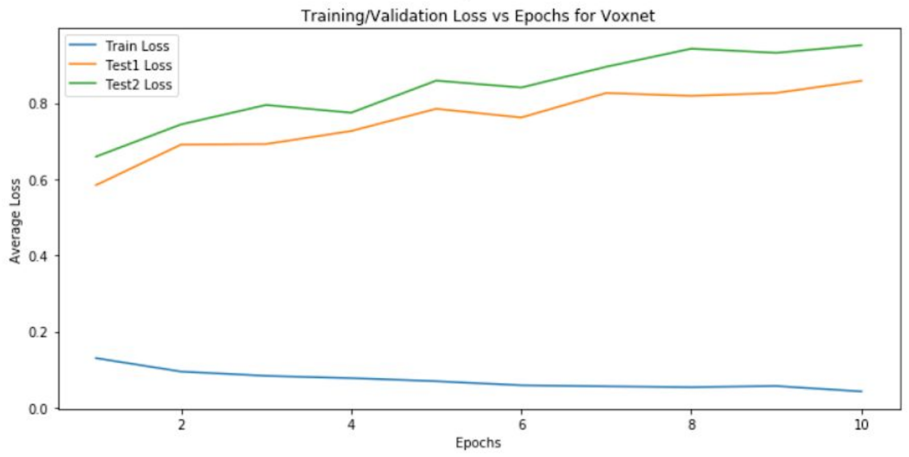
- We used CrossEntropy loss and Adam optimizer with a learning rate of 0.001.

Testing Set-2 *(augmented by rotating every sample by 12 degrees)*



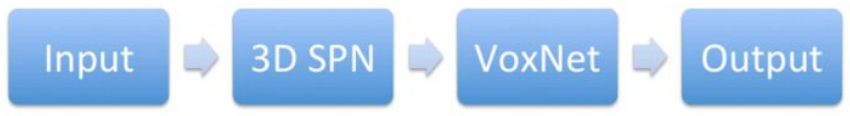






- **Observations:**
- Train accuracy on ModelNet10-30 deg: 98% approx
- Test accuracy on ModelNet10-30 deg: 86% approx
- Test accuracy on ModelNet10-30 deg: 85% approx

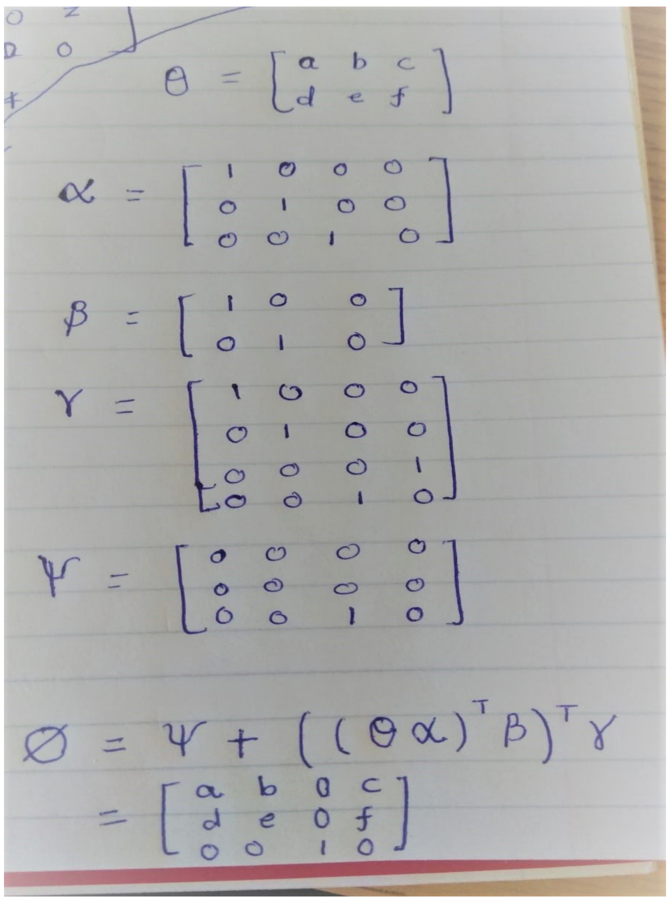
- ## 3D-SPN-VoxNet



- To achieve rotation invariance, a 3D spatial transformer network is built with VoxNet.
- The basic idea is to consider the spatial transform as a regression problem as we can specify the transformation parameters
- The localization of 3D-SPN has the same architecture as VoxNet.
- As we are considering the transformation only along gravity direction. The output layer having 6 units which are the transformation parameters.

$$\mathcal{T}_\theta = \begin{pmatrix} \theta_1 & \theta_{12} & 0 & \theta_{14} \\ \theta_{21} & \theta_{22} & 0 & \theta_{24} \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

- As Pytorch works on computational graphs, we need to convert the 6 regressed parameters into a 3 x 4 matrix to apply affine transformation in grid sampling. This we have achieved by transforming the 3 x 2 matrix with other matrices, alpha, beta, gamma, psai as follows.



$$\theta = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

$$\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\beta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\gamma = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

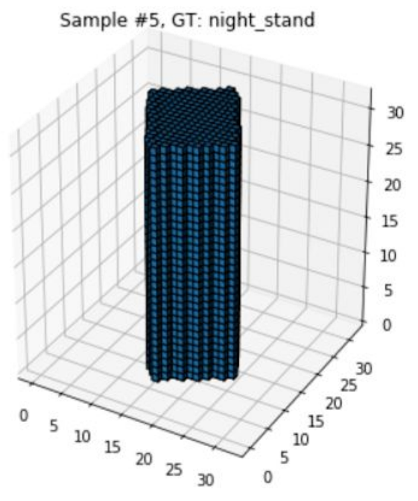
$$\psi = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

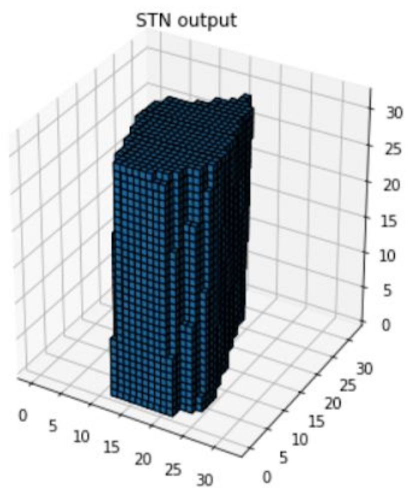
$$\phi = \psi + \left((\theta\alpha)^T \beta \right)^T \gamma$$

$$= \begin{bmatrix} a & b & 0 & c \\ d & e & 0 & f \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Theta contains the regressed parameters and phi is the resultant matrix used to transform the 3D object.

- We defined the **SPN_VoxNet** class that integrates the VoxNet with the SPN module. We defined the localization network the same as VoxNet architecture and implemented the forward method.
- We are training the model with CrossEntropy loss and Adagrad optimizer and learning rate 0.001 currently.
- We are further fine-tuning the model, experimenting with various hyperparameters, losses, etc to achieve the reported accuracy. We noticed that the model might by suffering from vanishing gradients problems as the spatial transformer is not efficiently rotating the 3D object when we visualized the objects with the trained model. We also used batch normalization layers to solve the problem but the results did not improve much.
- Now, we would be debugging the network and understanding the performance at the micro-level by looking at the gradients, plotting curves and tuning the hyperparameters.
- Accuracies (of the best model so far): Similar to the baseline 0/1 model
- Following the output of a data sample from the ‘nightstand’ class. STN tries to align it to one of the axis based on the error backpropagated from VoxNet.





References:

- Original Project: http://cs231n.stanford.edu/reports/2016/pdfs/417_Report.pdf
- ModelNet dataset: https://modelnet.cs.princeton.edu/
- Binvox: https://www.patrickmin.com/binvox/
- Pytorch: https://pytorch.org/tutorials/beginner/data_loading_tutorial.html#iterating-through-the-dataset
- VoxNet: https://ieeexplore.ieee.org/abstract/document/7353481
- Spatial Transformer Networks (2D) in Pytorch: https://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html