



# Neural Network for 3D Object Classification

*Paper Link:* [http://cs231n.stanford.edu/reports/2016/pdfs/417\\_Report.pdf](http://cs231n.stanford.edu/reports/2016/pdfs/417_Report.pdf)

Team ID: 2

Ekta Gavas (2019701005)

Surabhi Gupta (2019701024)



# Outline

- Problem Statement
- Dataset
- Technical Approach
- VoxNet
- Spatial Transformer Network
- Experimental Setting
- Experimental Results
- References



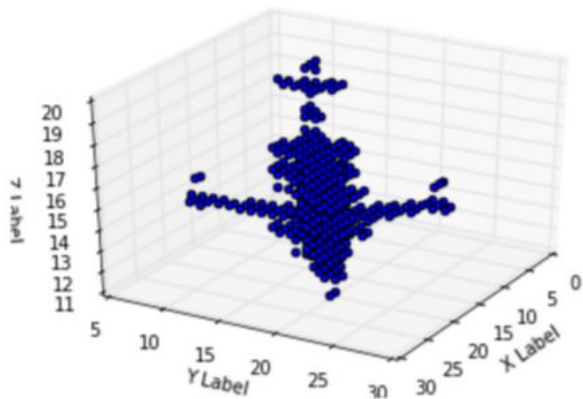
---

# Problem Statement

To solve 3D object classification on the 3D domain and evaluating our method by comparing with previous results.

Dealing with rotation invariance and difficulty of training CNN for each possible orientation of object

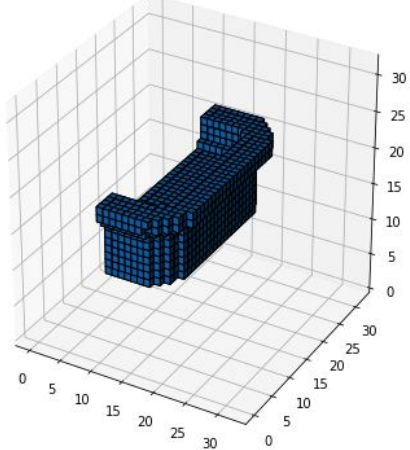
# Dataset



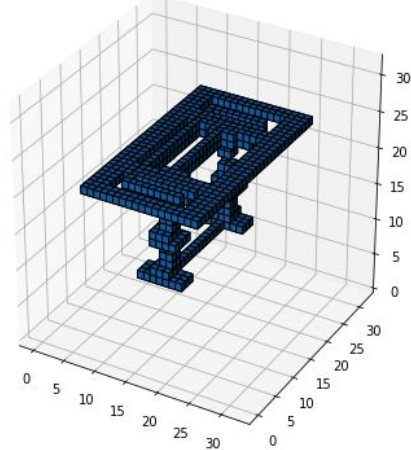
ModelNet contains CAD models used to train the deep network for 3D deep learning

- **ModelNet 40**
  - contains 12,311 shapes covering 40 common categories.
  - 9843 train + 2468 test samples
- **ModelNet 10**
  - contains 4899 shapes covering 10 common categories.
  - 3991 train + 908 test samples

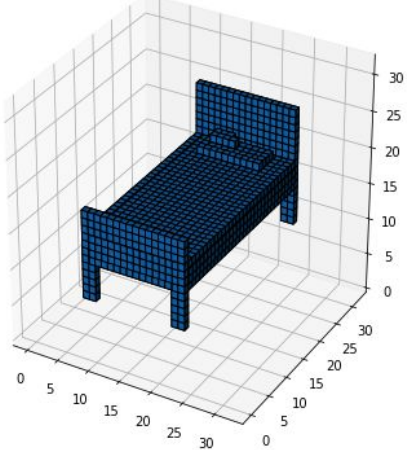
Sample #2, GT: sofa



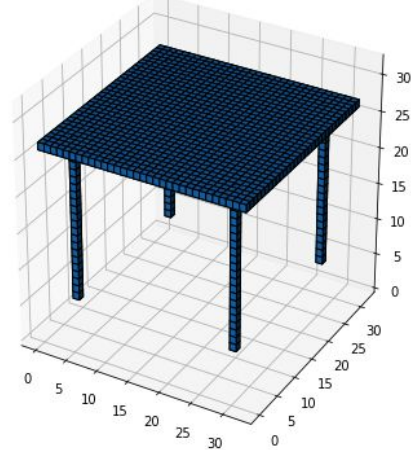
Sample #3, GT: table



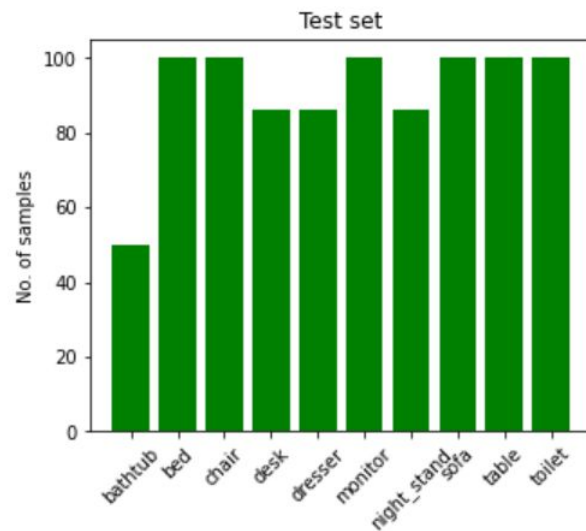
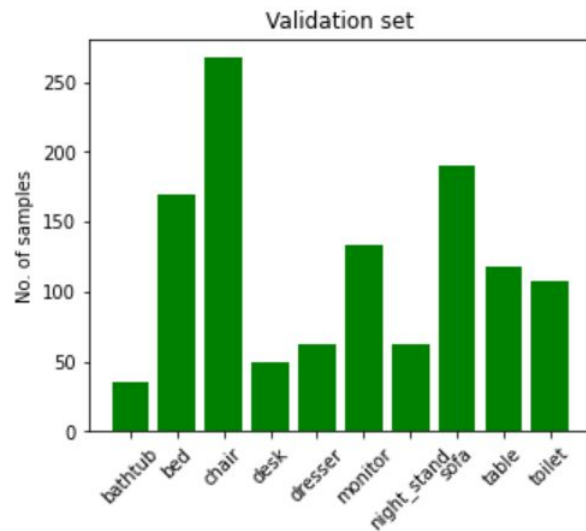
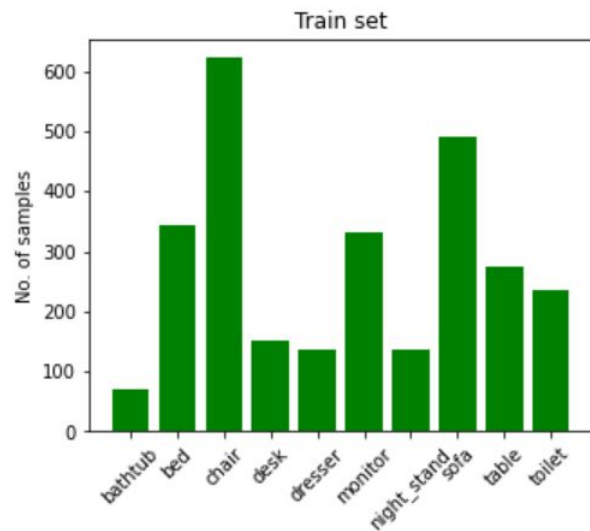
Sample #0, GT: bed



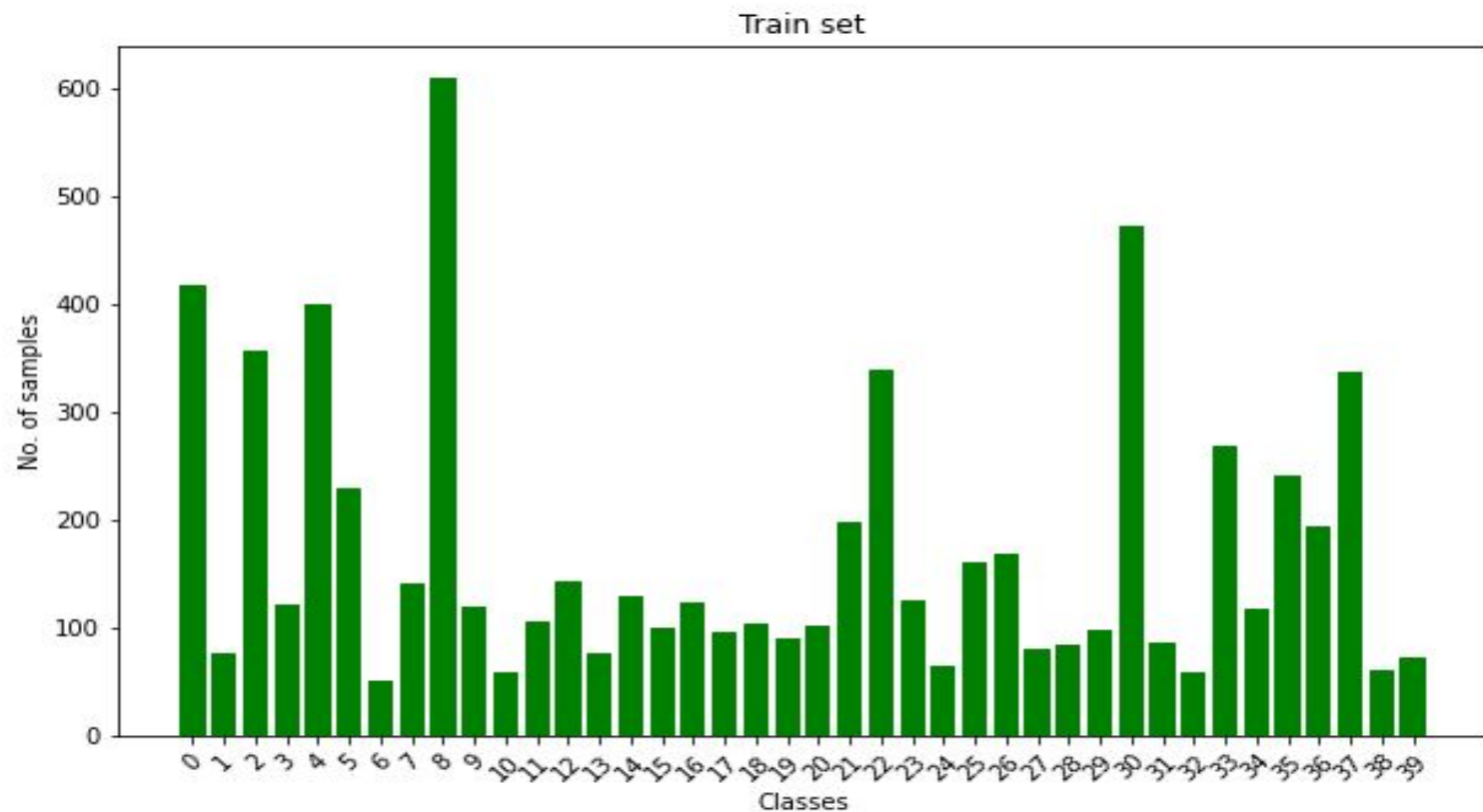
Sample #1, GT: table



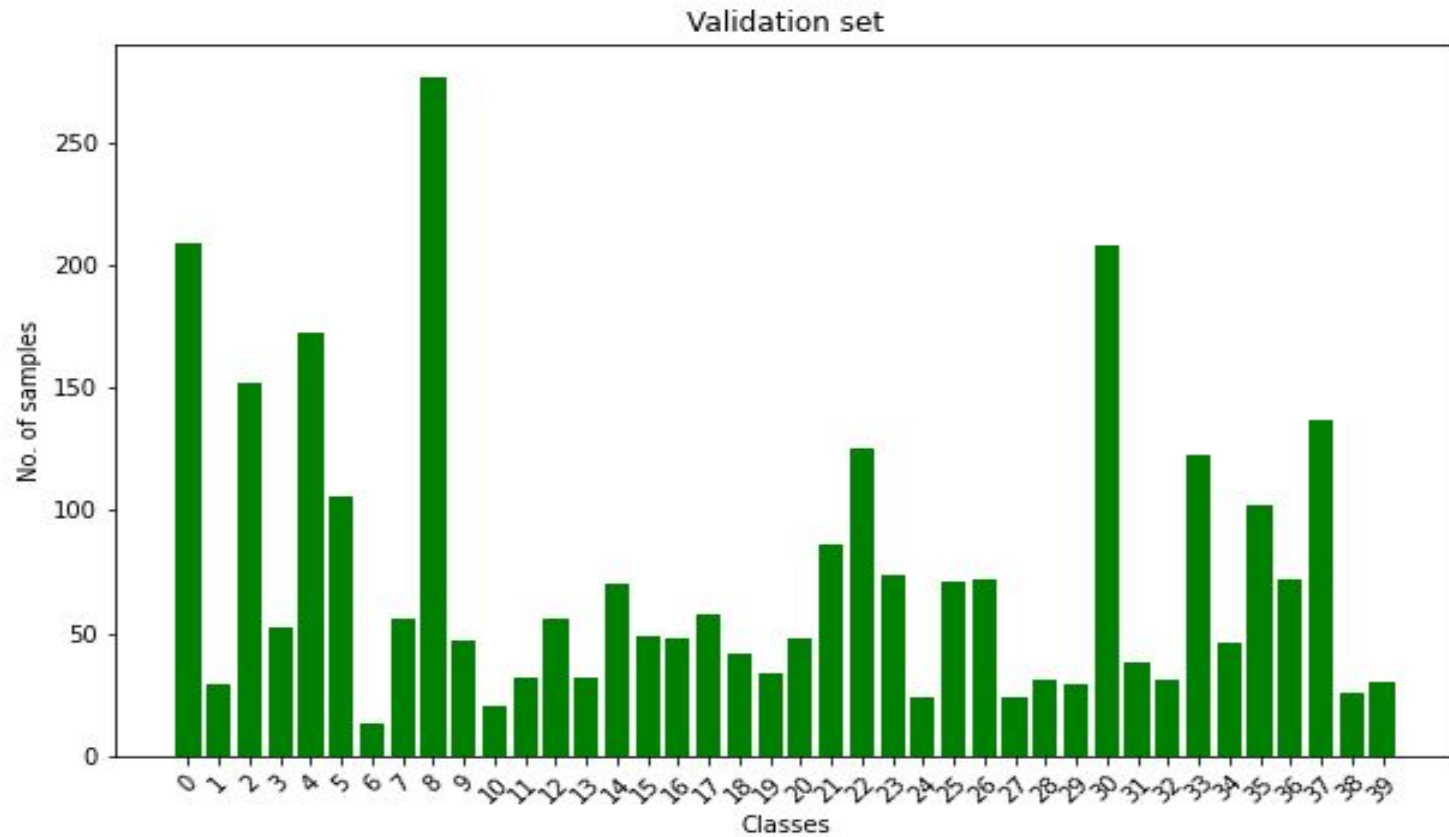
# Class Distribution for ModelNet10 dataset



# Class distribution for ModelNet40 dataset

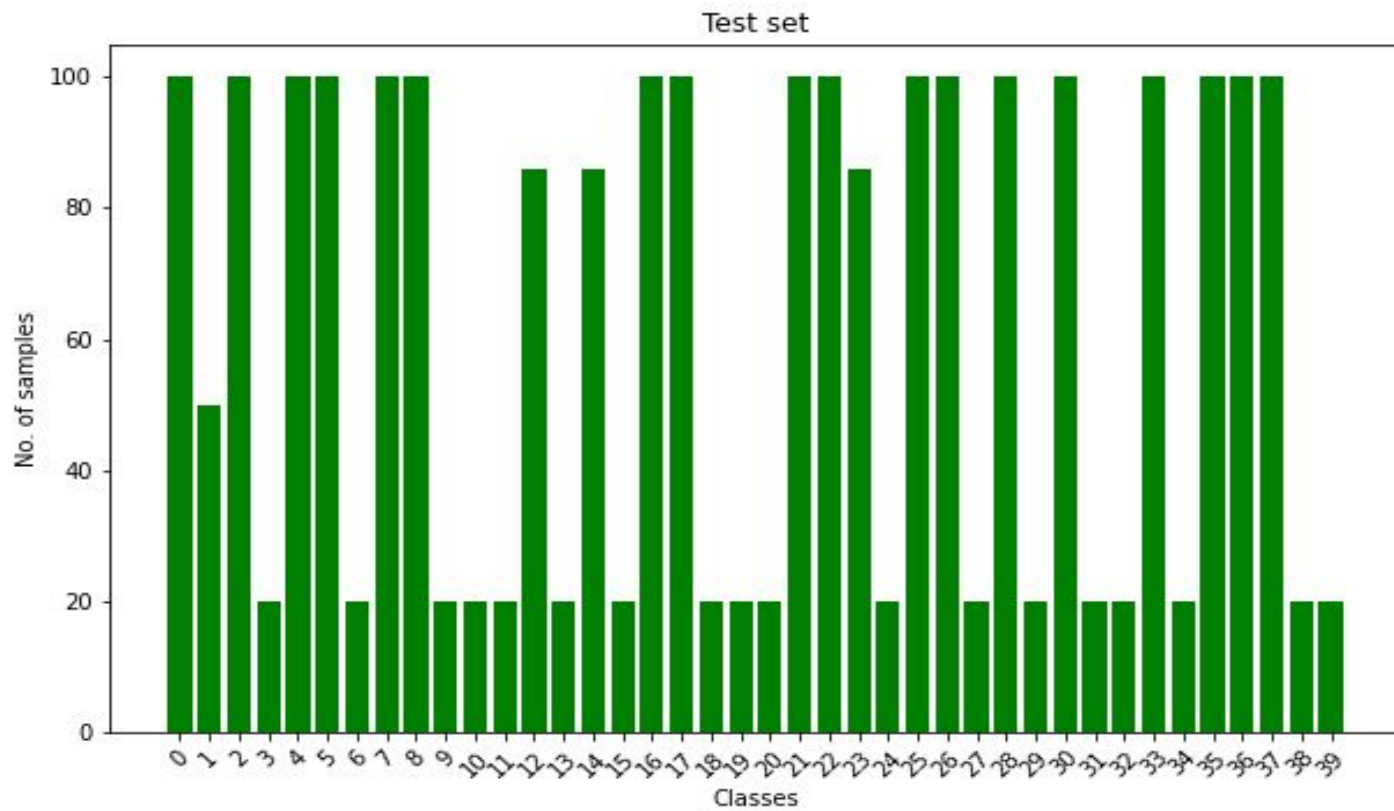



Continued...





Continued...





# Problems to be addressed

- What is the best representation of 3D models?
  - Using complex representation
- What is a good deep learning model architecture to train 3D voxel data?
  - Using 3D Spatial Transformer Network



# Technical Approach

- We take 0-1 binary voxel representation of the all 3D models.
- We test the *VoxNet* as our *baseline* model.
- Complex Representation
- 3D Spatial Transform Network

# VoxNet



- 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

Architecture of VoxNet

# Complex representation



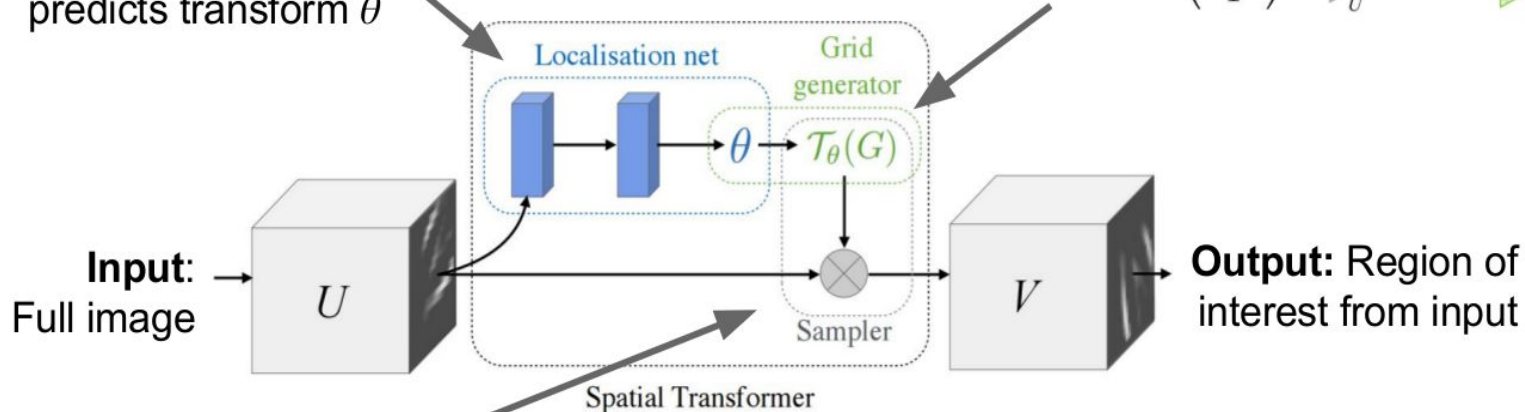
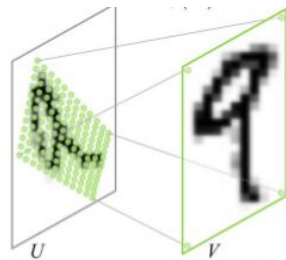
- For each voxel, we assign the **squared minimum distance to the surface of the 3D object**. In the representation, all the voxels in the object have value 0.
- We are calculating the euclidean distance transform for every object which converts it to the complex representation and we are using '***mahotas***' library to implement this.
- We are normalizing the values between 0 to 1 and then feeding it into the Voxnet model.
- Benefits of this approach:
  - **Solves the sparsity issue** of the simple 0-1 binary representation
  - **Rotation invariance**

# Spatial Transformer Network

A small  
**Localization network**  
predicts transform  $\theta$

**Grid generator** uses  $\theta$  to  
compute sampling grid

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$



**Sampler** uses  
bilinear interpolation  
to produce output

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

# Extended to 3D settings

- **Localization network:** We take same architecture of VoxNet, except the output layer has only 6 neurons which are the parameters  $\theta$  of the affine transformation
- **Parameterised Grid sampling**

$$\begin{pmatrix} x_i^s \\ y_i^s \\ z_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \begin{pmatrix} \theta_1 & \theta_{12} & \theta_{13} & \theta_{14} \\ \theta_{21} & \theta_{22} & \theta_{23} & \theta_{24} \\ \theta_{31} & \theta_{32} & \theta_{33} & \theta_{34} \end{pmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ z_i^t \\ 1 \end{pmatrix} \quad \mathcal{T}_\theta = \begin{pmatrix} \theta_1 & \theta_{12} & 0 & \theta_{14} \\ \theta_{21} & \theta_{22} & 0 & \theta_{24} \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

- **Differential Image sampling**

$$V_i = \sum_n^H \sum_m^W \sum_l^L U_{nml} \max(0, 1 - |x_i^s - m|) \\ \max(0, 1 - |y_i^s - n|) \max(0, 1 - |z_i^s - l|).$$

Layers	Parameters
fully connect	6
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

Architecture of localization network




# Architecture of 3D-SPN-VoxNet

---



The input is the 3D voxel data, and first go through 3D-SPN, then feed into VoxNet and get the output label



# Experimental Setting

We evaluated our model on ModelNets10 and ModelNets40.

- In the *training set*, we randomly rotated the data in  $[0-360^\circ]$  along the gravity direction .
- In the *validation set and test set*, we didn't apply any such rotations.

# Experiments carried out were...



- Modifying architecture
- Batch normalization
- Data Augmentation by rotating each object by some degree and adding to dataset
- Fine tuning with different batch-sizes, optimizer, number of epochs, loss functions etc
- Finally, random rotate transform was used for data augmentation and architecture was kept same as Voxnet network . Loss function used is Cross Entropy

# Experimental Results

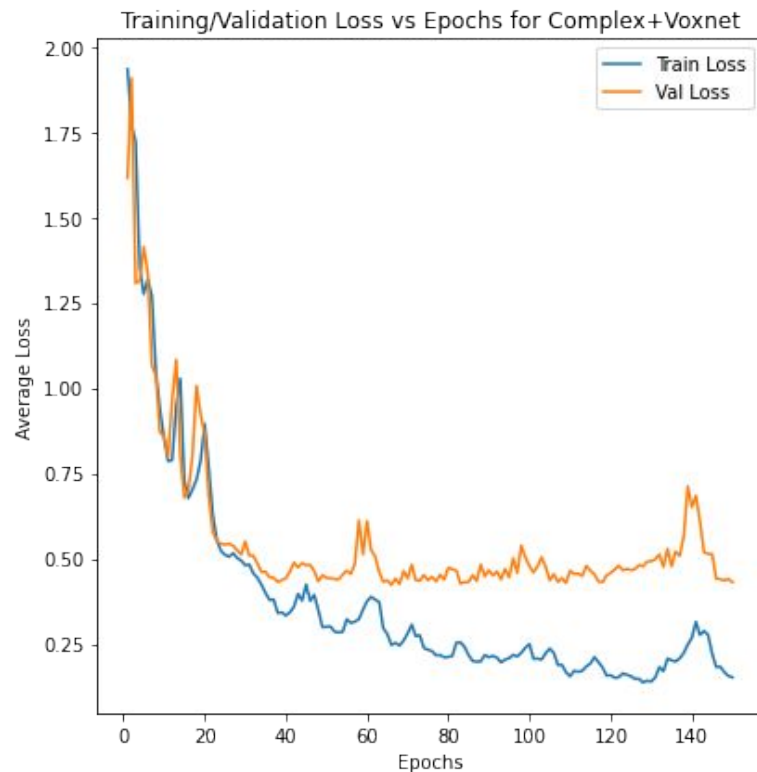
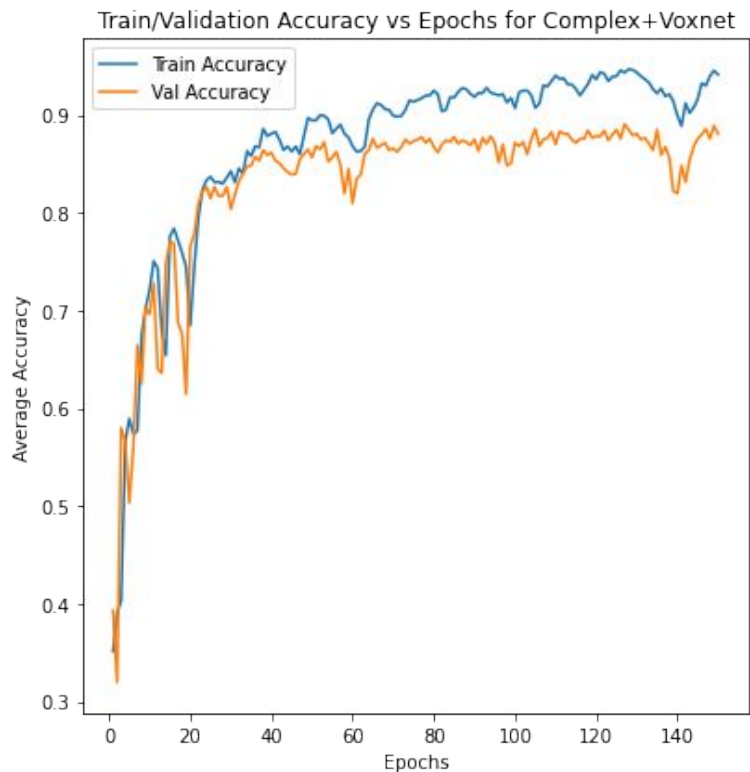
TABLE I  
CLASSIFICATION ACCURACY FOR MODELNET10

Algorithm	Training acc	Validation acc	Testing acc
Modelnet10+Baseline	91.55	89.01	<b>85.5</b>
Modelnet10+Complex	<b>94.13</b>	<b>89.06</b>	83.4
Modelnet10+3D SPN	90	88	54.4

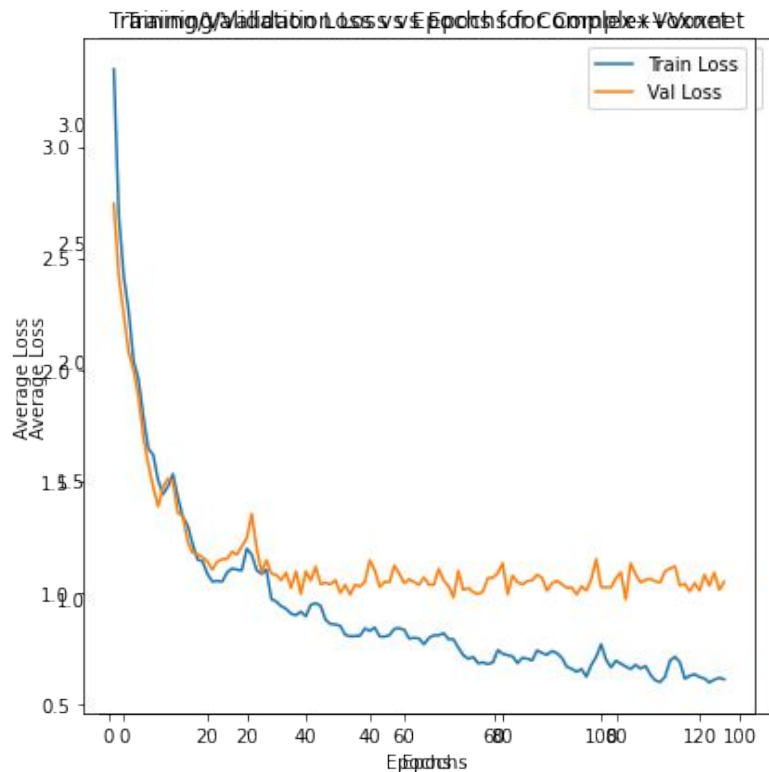
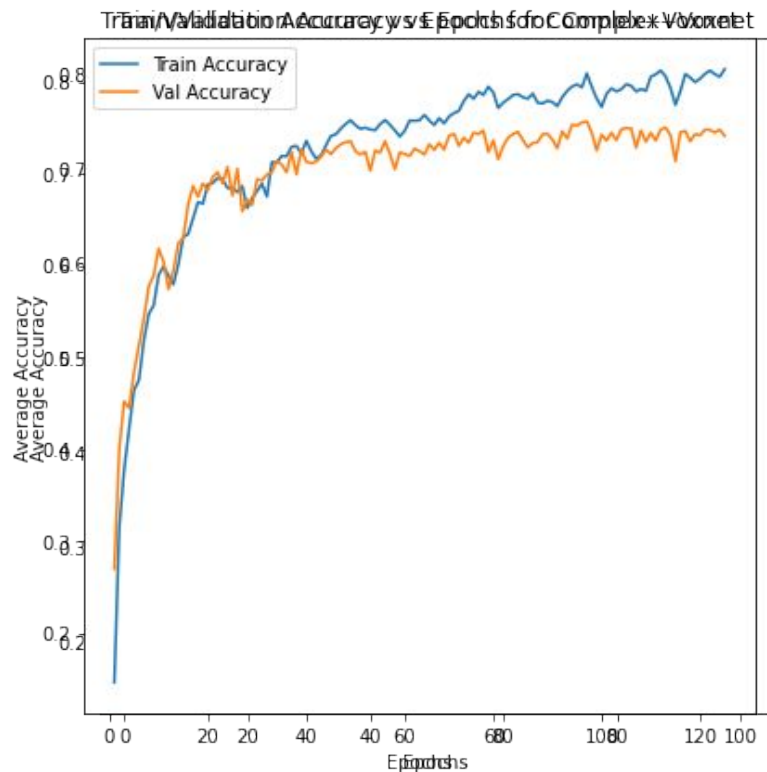
TABLE II  
CLASSIFICATION ACCURACY FOR MODELNET40

Algorithm	Training acc	Validation acc	Testing acc
Modelnet40+Baseline	80	<b>77.6</b>	<b>77.5</b>
Modelnet40+Complex	<b>81.36</b>	75.66	72.97
Modelnet40+3D SPN	75.6	76	53.4

# Complex Representation *(ModelNet 10)*



# Complex Representation *(ModelNet 40)*



## Other Evaluation metrics...

TABLE III  
PRECISION, RECALL AND F1-SCORE FOR COMPLEX+MODELNET10

Class	Precision	Recall	F1 Score	Support
0	0.9	0.54	0.675	50
1	0.847	0.94	0.891	100
2	0.809	0.89	0.848	100
3	0.686	0.558	0.615	86
4	0.8	0.791	0.795	86
5	0.970	0.96	0.965	100
6	0.783	0.628	0.627	86
7	0.905	0.95	0.927	100
8	0.677	0.86	0.758	100
9	0.951	0.97	0.96	100
Avg/Total	0.833	0.830	0.826	908

# Complex Repr with ModelNet40

- Here are the metrics for first few classes
- We see that the precision is less for classes with less samples

	precision	recall	f1-score	support
0.0	0.971	1.000	0.985	100
1.0	0.667	0.440	0.530	50
2.0	0.816	0.800	0.808	100
3.0	0.171	0.350	0.230	20
4.0	0.643	0.740	0.688	100
5.0	0.847	0.940	0.891	100
6.0	0.824	0.700	0.757	20
7.0	0.835	0.960	0.893	100
8.0	0.721	0.880	0.793	100
9.0	0.917	0.550	0.687	20
10.0	0.423	0.550	0.478	20
11.0	0.147	0.700	0.243	20
12.0	0.506	0.488	0.497	86
13.0	0.583	0.700	0.636	20
14.0	0.560	0.547	0.553	86
15.0	0.312	0.250	0.278	20
16.0	0.690	0.870	0.770	100
17.0	0.944	0.840	0.889	100
18.0	0.654	0.850	0.739	20
19.0	0.440	0.550	0.489	20
20.0	0.810	0.850	0.829	20
21.0	0.852	0.750	0.798	100
22.0	0.828	0.960	0.889	100
23.0	0.667	0.558	0.608	86
24.0	0.700	0.700	0.700	20
25.0	0.822	0.740	0.779	100
26.0	0.761	0.540	0.632	100
27.0	0.444	0.200	0.276	20
28.0	0.975	0.770	0.860	100
29.0	0.000	0.000	0.000	20



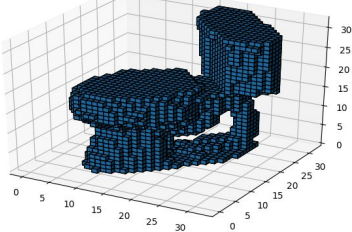
# 3D SPN



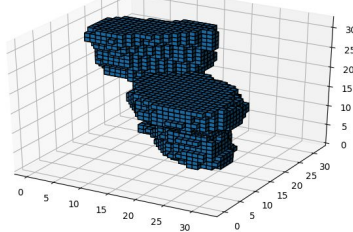
- After training is completed, we take the intermediate output from 3D SPN and plot the generated transformations.
- We noticed that the voxelizations trying to align themselves along one direction of cube surface.
- According to the paper, SPN actually has the effect of alignment which is also observed in our model upto some extent.

# Output of 3D SPN

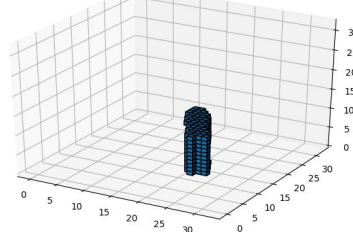
Sample #0, GT: toilet



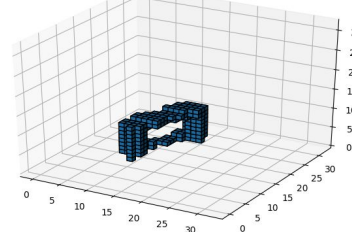
STN output



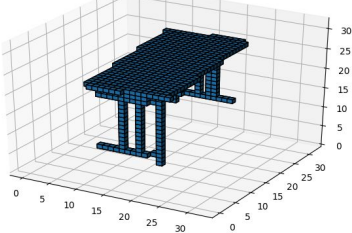
Sample #0, GT: mantel



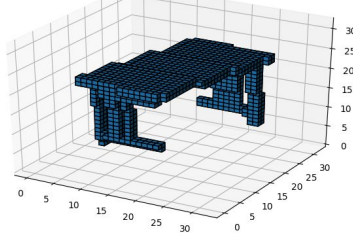
STN output



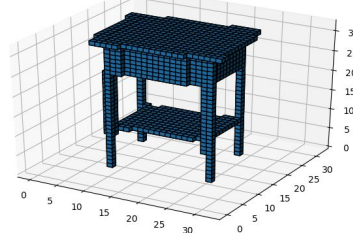
Sample #1, GT: table



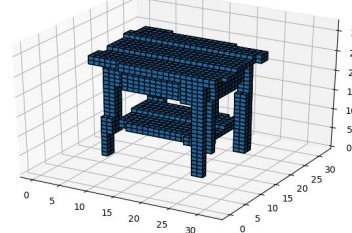
STN output



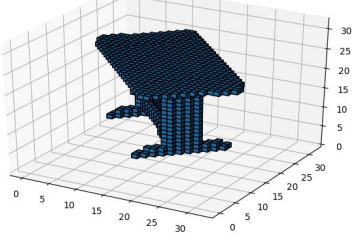
Sample #1, GT: night\_stand



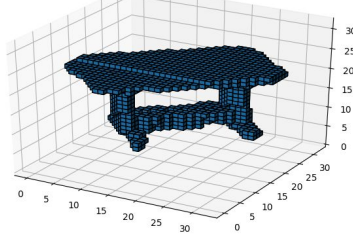
STN output



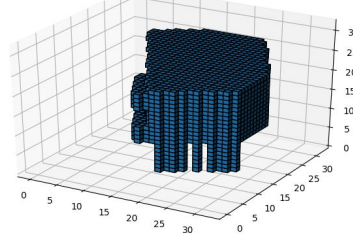
Sample #2, GT: table



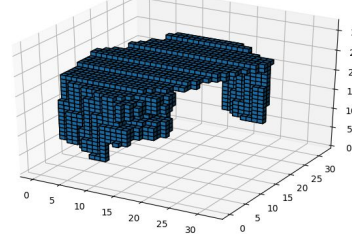
STN output



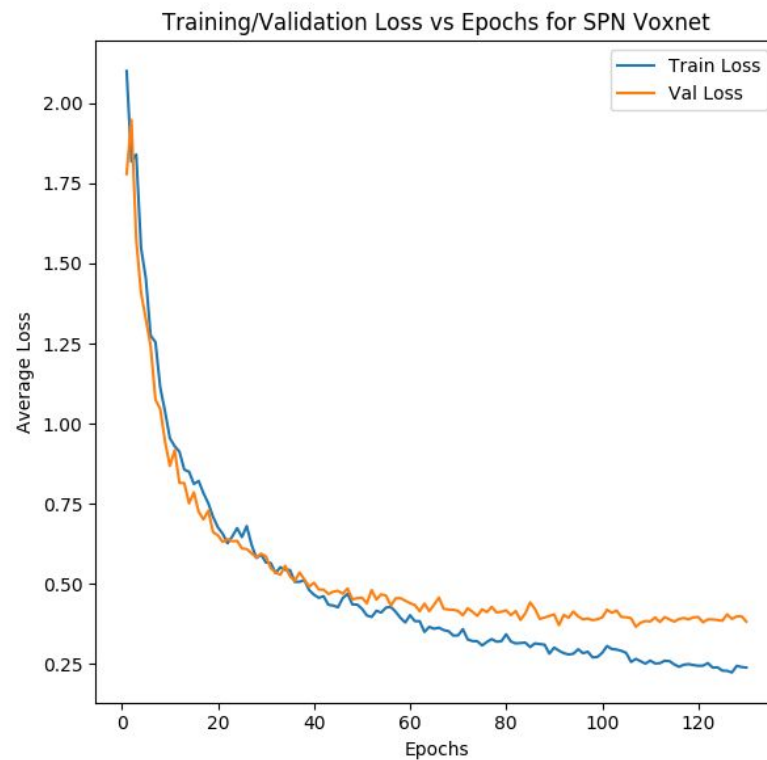
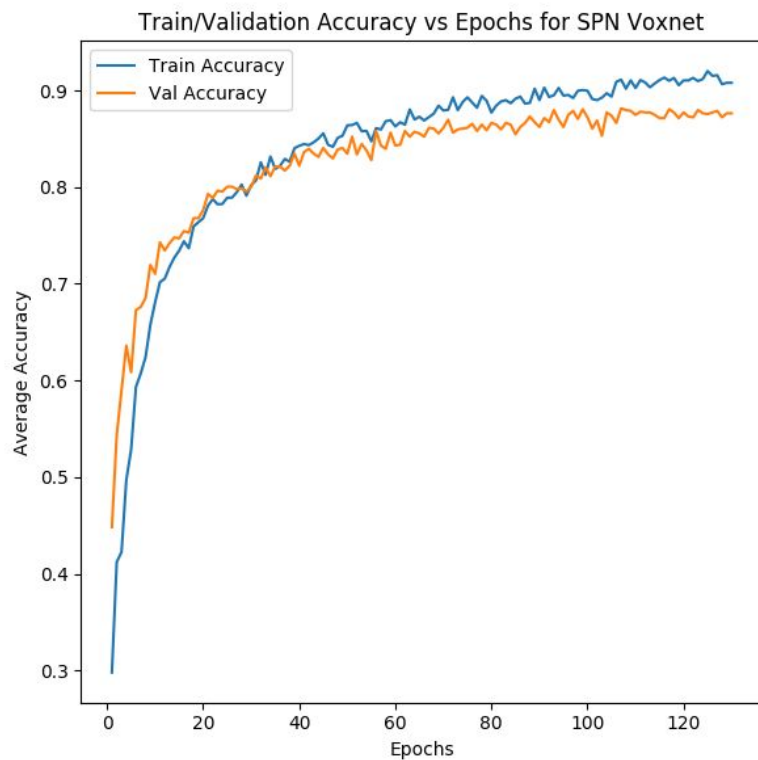
Sample #2, GT: desk



STN output

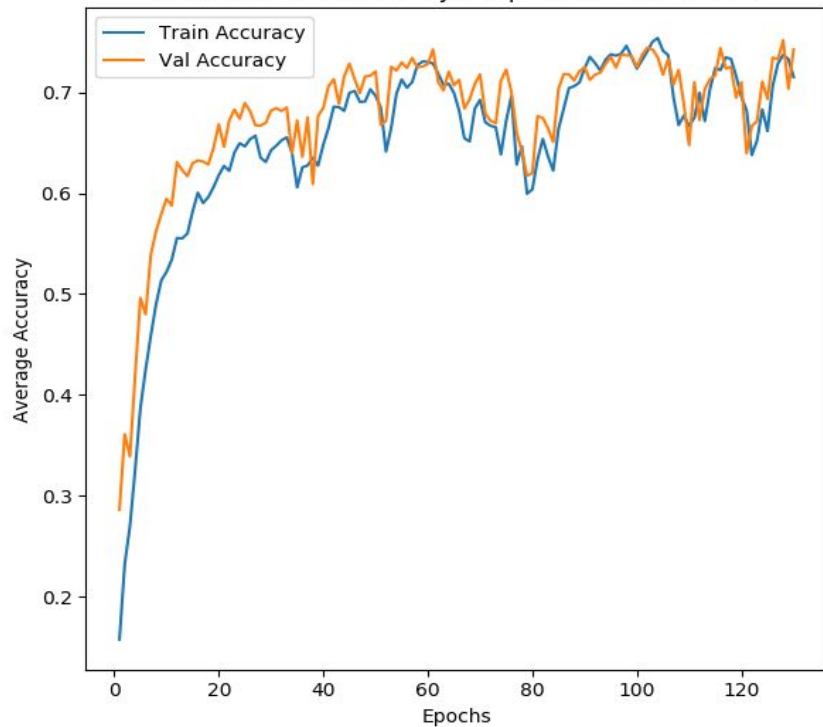


# 3D SPN *(ModelNet 10)*

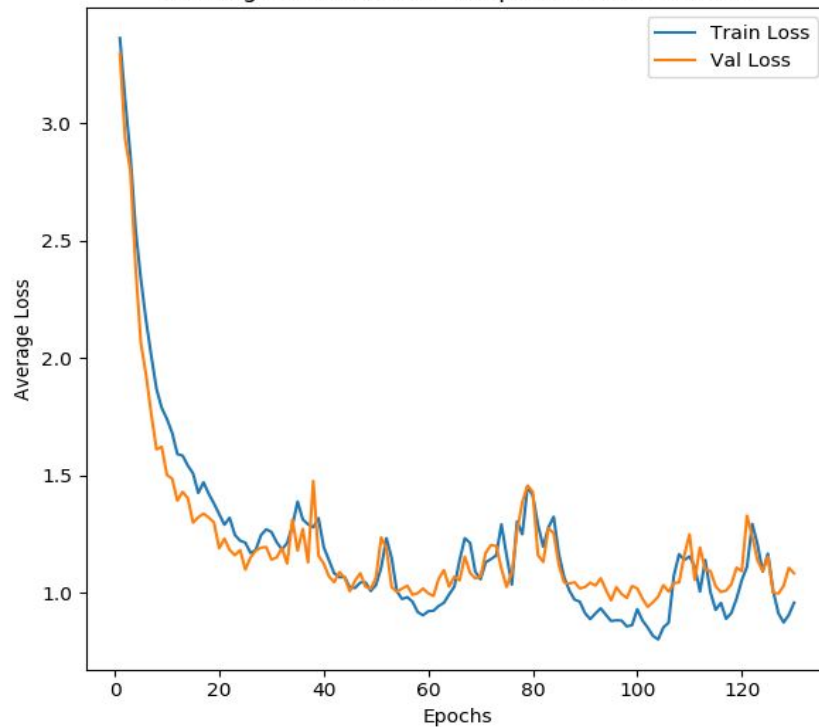


# 3D SPN *(ModelNet 40)*

Train/Validation Accuracy vs Epochs for SPN Voxnet



Training/Validation Loss vs Epochs for SPN Voxnet



## Other Evaluation metrics...

TABLE IV  
CLASSIFICATION ACCURACY FOR 3D SPN+MODELNET10

Class	Precision	Recall	F1 Score	Support
0	0.208	0.100	0.135	50
1	0.737	0.870	0.798	100
2	0.598	0.730	0.658	100
3	0.417	0.291	0.342	86
4	0.542	0.523	0.533	86
5	0.680	0.680	0.680	100
6	0.357	0.233	0.282	86
7	0.600	0.870	0.710	100
8	0.453	0.340	0.389	100
9	0.536	0.670	0.596	100
Avg/Total	0.533	0.563	0.539	908

	precision	recall	f1-score	support
0.0	0.821	0.960	0.885	100
1.0	0.537	0.440	0.484	50
2.0	0.690	0.580	0.630	100
3.0	0.140	0.350	0.200	20
4.0	0.516	0.660	0.579	100
5.0	0.814	0.790	0.802	100
6.0	0.444	0.600	0.511	20
7.0	0.727	0.480	0.578	100
8.0	0.672	0.820	0.739	100
9.0	0.304	0.350	0.326	20
10.0	0.333	0.350	0.341	20
11.0	0.278	0.250	0.263	20
12.0	0.531	0.302	0.385	86
13.0	0.464	0.650	0.542	20
14.0	0.333	0.256	0.289	86
15.0	0.087	0.200	0.121	20
16.0	0.351	0.200	0.255	100
17.0	0.899	0.710	0.793	100
18.0	0.354	0.850	0.500	20
19.0	0.333	0.500	0.400	20
20.0	0.562	0.900	0.692	20
21.0	0.383	0.180	0.245	100
22.0	0.597	0.740	0.661	100
23.0	0.258	0.267	0.263	86
24.0	0.560	0.700	0.622	20
25.0	0.458	0.380	0.415	100
26.0	0.459	0.280	0.348	100

## 3D SPN with ModelNet40

- Here are the metrics for first few classes
- We see that the precision is less for classes with less samples

# Possible improvements



- Mixing all the dataset samples and again splitting may improve the performance
- Efficiently detecting the noisy samples and removing them is also one possibility
- We can also address the class-imbalance problem by augmentation or weighted sampling.



# References

[Project Paper](#)

[Spatial Transformer Network](#)

[VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition](#)

[3D ShapeNets: A deep representation for volumetric shapes](#)

[Debugging neural networks](#)





THANK YOU