Molla Hafizur Rahman (MEEG)
ID: 010832493

Project Report
Fall 2021

1 Introduction

1.1 Objective

In recent years, deep learning has gained great success in image and object detection. With the increasing number of 3D CAD models, object detection with these 3D data is ubiquitous in different fields such as self-driving car, garments, architecture, etc. Similarly in the mechanical field, various research has been done for 3D mechanical shape classification and retrieval. Marini et al. [1] propose a technique where they define 3D shape prototype for the classification of 3D content by applying graph transformation. Jayanti et al. [2] apply several clustering methods on 3D models to navigate them. Several works have been conducted by applying machine learning techniques. Several works have been conducted by applying machine learning techniques. For example, Ip et al. [3] utilize decision tree and reinforcement learning for the automated categorization of solid models such as wheels, sockets and housing models. Using the deep learning technique, Qin et al [4] proposed a model that is able to mimic the main phase of a manual classification process. Wang et al.[5] propose a novel method of retrieval 3D CAD model based on convolutional neural network. This model process the information gained from sketch-based and shape-based methods. Rucco et. al [6] proposed a supervised classification process that utilizes engineering knowledge and adjacent parts' class and contact type. Several other works classify 3D models by measuring similarities between them. For example, Zethaban et al. [7] used cosine similarity based on CAD features such as descriptors, image, signature, CAD file, etc. Cicirello and Regli [8] used manufacturing features for similarity assessment and classification. These methods, however, are time-consuming and do not provide accurate results.

Therefore, models are required that can efficiently detect different mechanical parts. Identifying and exploring the shape of mechanical components is vital for computer vision and manufacturing applications. Therefore, the **objective** of this project is to the identification of appropriate data representation and classification of 3D mechanical components. There are several ways to represent 3D data such as Descriptors[9], projection data[10], volumetric data [11], multiview data[12], point cloud [13], 3D meshes, and graphs, [14] etc. However, in this project, I choose volumetric data and point clouds as these data types are popular and have great potential to further extend the deep learning models based on these data. Particularly, VoxNet [15] based on volumetric data and PointNet [16] based on point cloud data is developed. The models are evaluated with different metrics such as precision, recall, F1-score, and accuracy.

1.2 Significance and contribution

The significance of classifying mechanical components can be described in several aspects. **First**, It enables process automation in the industry setting. The mechanical components are vital for both Engineers and manufacturers as they encounter different mechanical components every day during work. Automatic detection of these components can help to accelerate the production process. **Second**, easy retrieving and reusing existing CAD models can speed up the development of new products. **Third**, the identification of mechanical parts may greatly enhance the mechanism recognition in large assembly models. **Finally**, classifying those parts is the first step towards enabling virtual reality in the industry environment.

From the contribution point of view, the deep learning models developed in this project are general and applicable to any 3D CAD model. Additionally, the weights can be used as a Transfer learning for classifying other mechanical parts or segmentation.

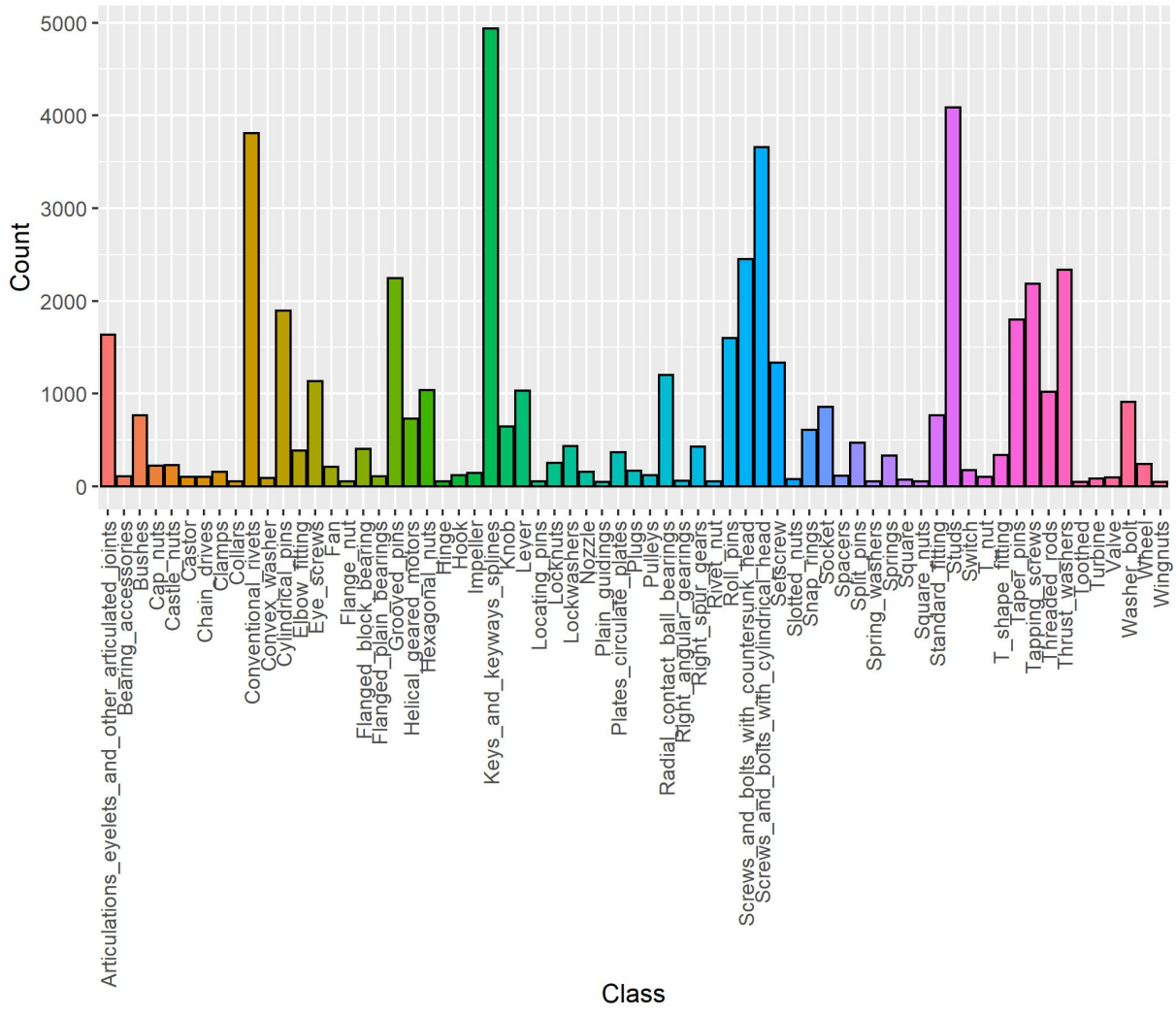


Figure 1: Count of 3D Mechanical CAD models of different classes

2 Methodology

2.1 Data Collection

The data is collected from an open-source platform that contains 3D CAD model of mechanical components of 67 classes. The dataset contains a total of 51,000 different parts of CAD models. The figure 1shows the total number of CAD models of different classes. These mechanical parts are differentiated by their detail shape while other common 3D objects are identified by their general shape. Also, a small detail of a mechanical part can distinguish it from the other parts. For example, Figure 2 shows different 3D CAD models of mechanical parts. Here, the thrust washer is very similar to the lock washer. Only a small split features of the lock washer differentiate it from a thrust washer. Similarly, hexagonal nut shares a similar hexagonal shape with lock nut. However, the additional circular features of the lock nut differentiate it from the hexagonal nut. In another application, a spur gear is comparable to a pulley. But small details in teeth and design distinguish them from each other. In common objects like chairs, changing the leg size does not change its function and class. Therefore, on one hand, the classification of these mechanical parts is essential, on the other hand, identification of the deep learning to classify is also necessary.

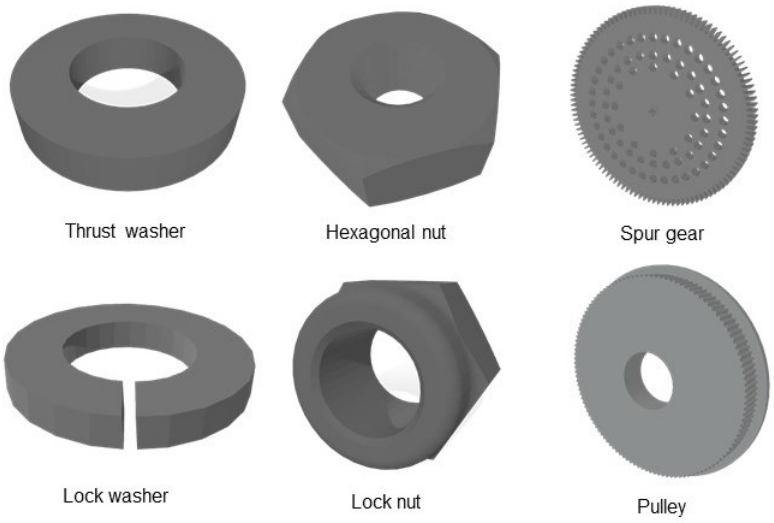


Figure 2: Example of 3D CAD models of mechanical parts

2.2 Data processing

The CAD models are in the obj format. However, for implementing the 3D model into the deep learning model, the obj format needs to be converted according to the deep learning models. For example, in this project, I use VoxNet and Pointnet for the classification task. For the VoxNet, the 3D CAD models are voxelized while for the Pointnet, the models are converted into point clouds. Binvox [17] an open source software and script is used to voxelized the 3D obj format. Using the Binvox, each of the CAD models is converted into $32 \times 32 \times 32$ voxels.

Unlike 3D voxel arrays, point clouds data is uninstructed which means data is simply a collection of the point captured from the 3D object. Point clouds have three main properties which differ from other 3D data. The first one is its permutation invariance. That is mean a network that takes N 3D data sets must be invariants to N! permutations of the input set in data feeding order. The second one is transformation invariance. The output of the learned representation should be unchanged if the object undergoes certain transformations such as rotation and translation. Finally, there should be interaction among points. In the point cloud, neighboring points often carry meaningful information. Therefore, the model needs to be able to capture local structures from nearby points. In this study, in order to obtain the point cloud, the Trimesh [18] python library is used. The figure 3a shows the voxel grid of a 3D CAD model of Elbow fit, while 3bshows the model with point clouds.

2.3 Model Architecture

In this project, two deep learning models including VoxNet and Pointnet are used to classify the 3D CAD models of mechanical parts. Each of the models is described below:

2.3.1 VoxNet

VoxNet is built based on the 3D convolutional neural network (CNN). Bellow each of the components of the VoxNet are described:

2.3.1.1 3D Convolutional layer The convolutional layer is the core building block of a CNN. A feature detector, known as kernel or filter is used to extract the feature. This filter moves across the respective fields of the image, checking if the feature is present. This process is known as convolution.

Unlike 2D convolution which moves along 2 directions (x and y), 3D convolution uses a 3-dimensional filter on the 3D dataset. This 3D filter moves in 3 directions (x,y,z) and calculates the low-level features and representation. As the input of the 3D convolutional layer is 3 dimensional ($I \times J \times K$), the output will also be a cube or cuboid-like. As mentioned earlier, in this study, I use I = J = K = 30, a fixed size voxel. It is mentionworthy that the convolutional layer takes a four-dimensional input volume in which three of the dimension are spatial, and the fourth contains the feature map. This layer creates f features maps by convolving the input with f learned filters of shape $d \times d \times d \times f^{'}$, where d is the spatial dimension and $f^{'}$ is the number of features map. A stride s is also applied to cover the 3D data. The output of the convolution layer is passed through rectified linear unit (ReLU) activation function.

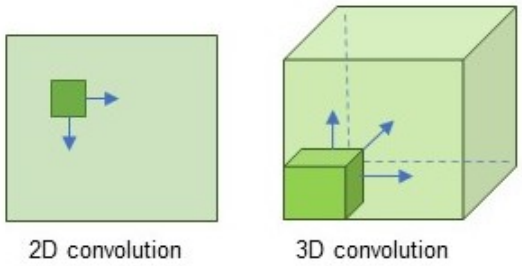
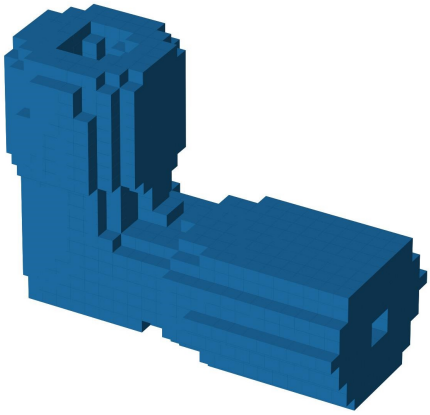


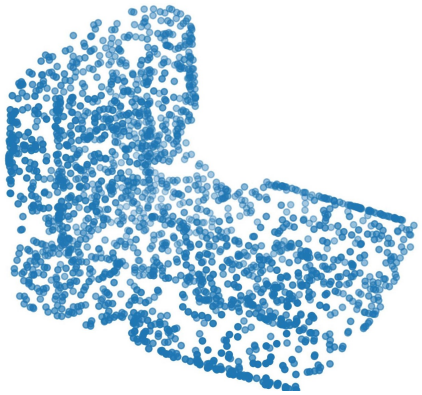
Figure 4: Comparison of 2D CNN vs. 3D CNN

2.3.1.2 Pooling layers This layer is also known as downsampling and is used to reduce the number of parameters in the input. Similar to the convolutional layer, in this layer, a filter $m \times m \times m$ sweeps over the entire input. However, this filter does not have any weight. Rather, the filter applies an aggregation function to the values of the specific volume of the voxel.

2.3.1.3 Fully connected layers Fully connected layers have n output neurons, where n is the total number of classes. Based on the linear combination of the previous layer output, these neuron provides the outputs of each class. A non-linear function 'softmax' is used to provide the probabilistic outputs.



(a) Voxelization of Elbow fit



(b) Point clouds of Elbow fit

Figure 3: Voxel grid and point cloud of 3D CAD model of mechanical parts

Name	VoxNet-Base	VoxNet-Deep (Own model)
Layer 1	3D Conv 32F-6K-S2	3D Conv 32F-5K-S1
Layer 2	3D Conv 64F -5K-S2	Batch normalization + Leaky ReLU
Layer 3	3D Conv 64F -5K-S2	Max Pool 2 x 2 S2
Layer 4	FC- 67	3D Conv 64F-3K-S1
Layer 5		Batch normalization + Leaky ReLU
Layer 6		Max Pool 2 x 2 S2
Layer 7		3D Conv 128F-3K-S1
Layer 8		Batch normalization + Leaky ReLU
Layer 9		Max Pool 2 x 2 S2
Layer 10		FC 256
Layer 11		FC - 67

Table 1: Network architectures of VoxNet-Base and VoxNet-Deep.

2.3.1.4 Proposed architecture for VoxNet With these layers and hyperparameters, different architectures are possible. Comparing each of the architecture is time and resource-consuming. Therefore, I start from a basic architecture (VoxNet-Base) and improve it gradually (VoxNet-Deep). In the baseline model, the first layer of Convolution 3D, i use 16 3D filters with a size of $6 \times 6 \times 6$. In the second and third layers, I use 64 kernels with the size of $5 \times 5 \times 5$. In all the convolution layers, I used a stride size of 2, and the 'ReLU' activation function is used here. All the features obtained from the previous layers are vectorized by a Flatten layer. Finally, a dense layer with 'softmax' activation function is used to identify the class. In the VoxNet-Dense architecture, I use Batch normalization [19] and Leaky ReLU [20] as activation functions in addition to Convolution 3D. Also, I used max pool layers. Three blocks of this combination are used where the size of the filters are 32, 64, and 128 respectively. For the classification, I use two fully connected layers which include 256 and 67 (total class number) neurons. Table 1 shows the comparison of different layers between VoxNet-Base and VoxNet-Deep. F, K, and S refer to filter numbers, kernel size, and strides respectively.FC indicates fully connected layers

2.3.2 PointNet

PointNet consists of two main components which include shared multi-layer perceptron (MLP) and transformer net (T-net). Each of the components is discussed below:

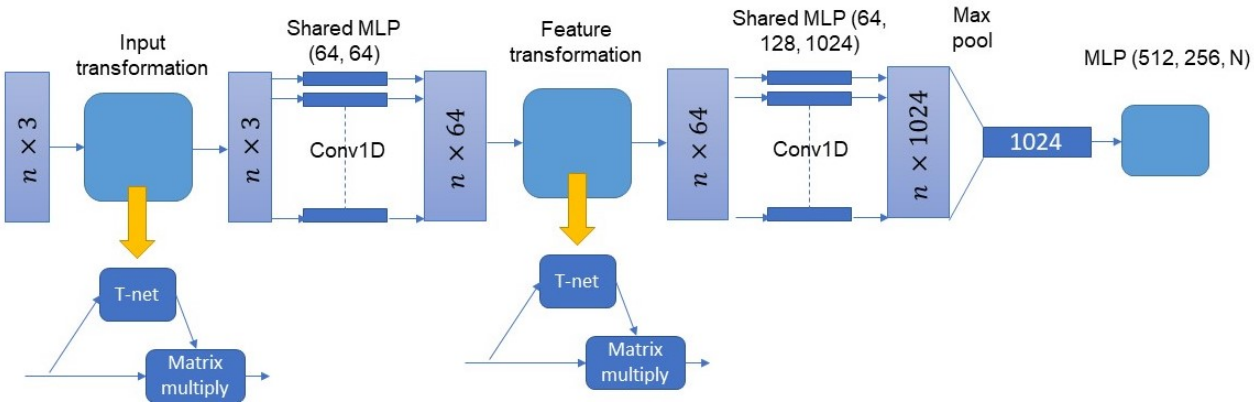


Figure 5: PointNet architecture

Hyperparameters and parameters	Value		
	VoxNet-base	VoxNet-Deep	PointNet
Voxel/Point Size	$32 \times 32 \times 32$	$32 \times 32 \times 32$	2048
Conv activation function	ReLU	Leaky ReLU	ReLU
FC activation function		ReLU	
Loss function	Cross-entropy	Cross-entropy	Cross-entropy
Number of epochs (Early stopping enabled)		100	
Learning rate		0.001	
Optimizer		Adam	
Batch size		32	
Trainable parameters	646,265	4,225,283	3,486,988

Table 2: Hyperparameter settings for different models

2.3.2.1 Shared MLP The shared MLP map each of the n points from three-dimension (x, y, z) to 64 dimensions. It is noted that this shared MLP can be obtained by using a 1D convolutional neural network. This procedure is repeated several times with 64 dimensions, 128 dimensions, and finally 1024 dimensions to map the n points. After that, a max-pooling layer is used to create the feature vector in R^{1024}. Here, max pool layer works as a symmetric. As mentioned earlier, the point cloud data need to be permutation invariants. In order to make PointNet permutation invariants, other symmetric functions such as LSTM, average pooling also can be used. However, maxpooling works best. Finally, a three-layer fully connected network with 512, 126, and n neurons are used to identify the final class. Here n is the number of classes of the dataset. Figure 5 shows the overview of the architecture of the PoinNet.

2.3.2.2 Tranformer network (T-net) T-net is applied to confirm the transformation invariance of the PointNet. T-net aligns all input set to a canonical space before feature extraction. Two sets of the T-net are used in PointNet architecture. The first one is for input transformation. The operation for the T-net are similar to the high-level architecture of PointNet. 1D convolution is used to map the input points to the higher dimensional space. After that max-pooling is used to encode a global feature vector whose dimensionality is reduced to R^{256} with fully connected layers.The features at the final FC layers are combined with weight and biases which results in a 3×3 transformation matrix. Figure 6 shows the architecture of the T-net. The second T-net is used for feature transformation which is extended to 64-dimensional embedding space. This T-net is nearly similar to the previous one expect the trainable weight and biases. Here the weights become 256 and the biases become 4096 and resulting a 64. To reduce the overfitting due to this increased parameter, a regularization term is used to the loss function. The equation below shows the regularization term.

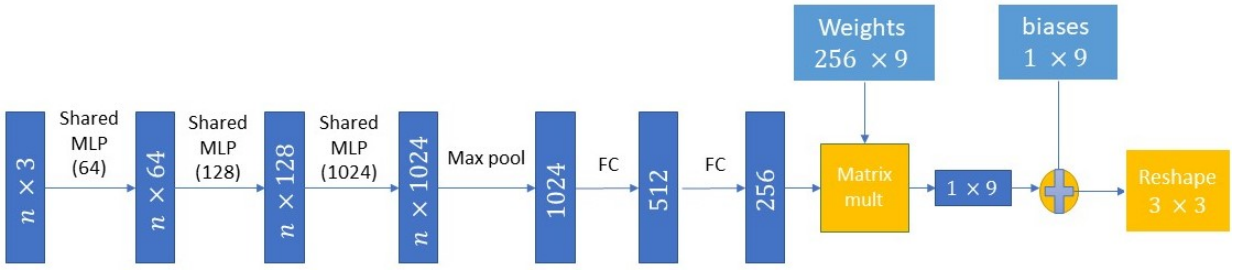
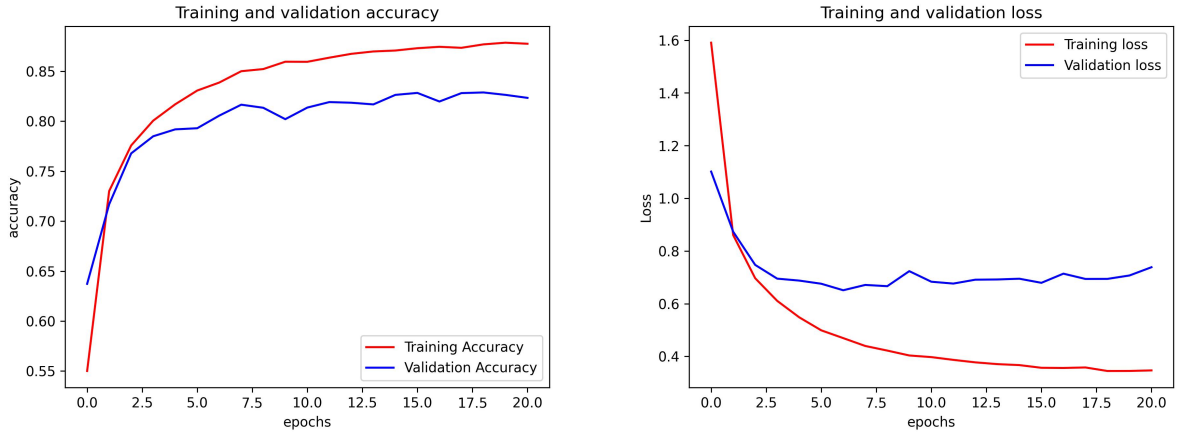


Figure 6: T-net architecture



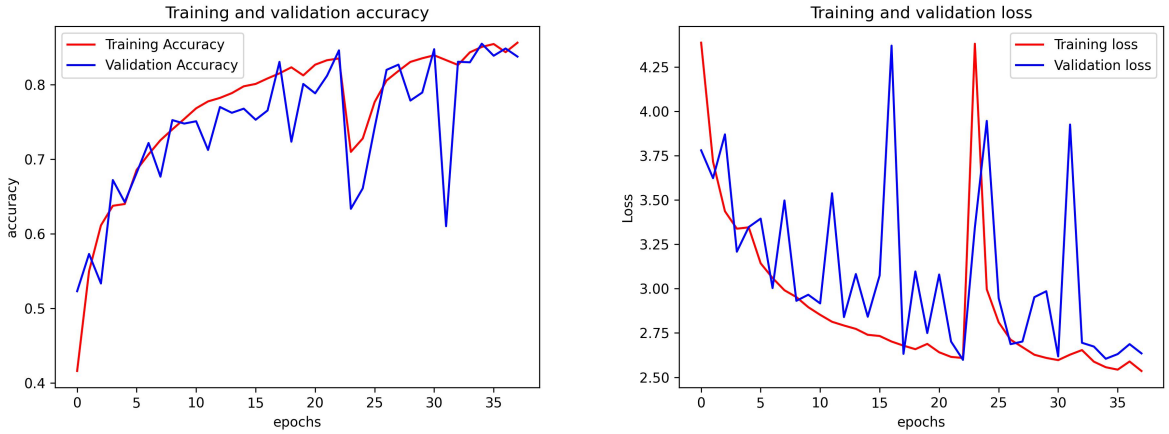
(a) Accuracy vs. Epoch

(b) Loss vs. Epoch

Figure 7: Training and Validation loss and accuracy of VoxNet

$$L_{reg} = \|I - AA^T\|$$

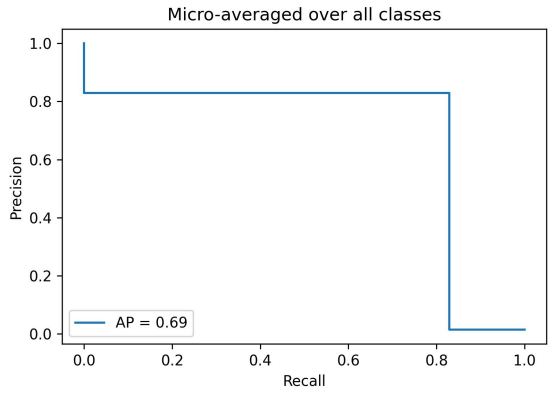
Both VoxNet and PointNet are implemented through Keras deep learning library [21]. The data set is split into train, validation, and testing with a ratio of 70 %, 20 % and 10 % respectively. For both of the models, a batch size of 32 and a "categorical-crossentropy" loss function is used. The models are trained with 100 epochs and Adam optimizer is used with a learning rate of 0.001 (see Table 2 for overall hyperparameter settings). Figure 9 shows the VoxNet accuracy over epochs. For early stopping the training ends at 20 epochs. For PointNet, it ends at 35 epochs (Figure 8b). The training accuracy of both models reaches about 84%. However, the validation accuracy of the VoxNet (79%) is lower than the PointNet (83%).



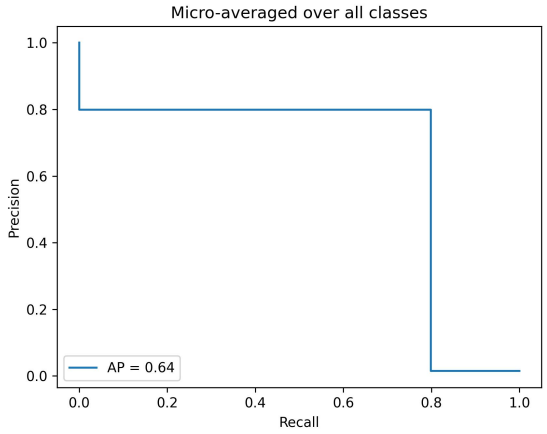
(a) Accuracy vs. Epoch

(b) Loss vs. Epoch

Figure 8: Training and Validation loss and accuracy of PointNet



(a) Micro average precision recall curve for PointNet



(b) Micro average precision recall curve for VoxNet

Figure 9: Precision Recall curve of PointNet and VoxNet-deep

2.4 Result and Discussion

Once the models are trained, it is tested on the testing dataset. From the testing result, I obtained the predicted labels and it is compared with the actual labels. Table 3 shows the Precision, Recall, F1-score, and Accuracy for different models. It is mention-worthy that all the metrics are computed in a micro average setting. As mentioned earlier, the data set is imbalanced. Therefore, the micro average is chosen over the macro average. In multiclass classification setting, micro average precision and recall are computed by the following equation:

$$Precision = \frac{\sum_c(TP)}{\sum_c(TP) + \sum_c(FP)}$$

$$Recall = \frac{\sum_c(TP)}{\sum_c(TP) + \sum_c(FN)}$$

The baseline VoxNet achieves the lowest performance for the metrics. My own model VoxNet-Deep achieves higher 2% higher scores in all the metrics compared to VoxNet-baseline model. I believe adding extra max-pooling layers improves the results a bit. For precision, VoxNet-Deep achieves a 0.84 score which is much higher than the baseline model. PointNet achieves the overall highest score in all the metrics. It achieves highest accuracy which is 0.83.

The precision recall curve also shows that the area under this curve is higher for PointNet compare to VoxNet. Figure 9a and 9bplots the Micro average precision recall curve for PointNet and VoxNet respectively. The area under the precision recall curve for PointNet is about 0.69 while the area under the precision recall curve for VoxNet is about 0.64.

Metric (Micro)	VoxNet-Base	VoxNet-Deep	PointNet
Precision	0.80	0.84	0.84
Recall	0.79	0.81	0.83
F1-score	0.79	0.80	0.83
Accuracy	0.79	0.81	0.83

Table 3: Precision, Recall, F1-score and Accuracy for different models

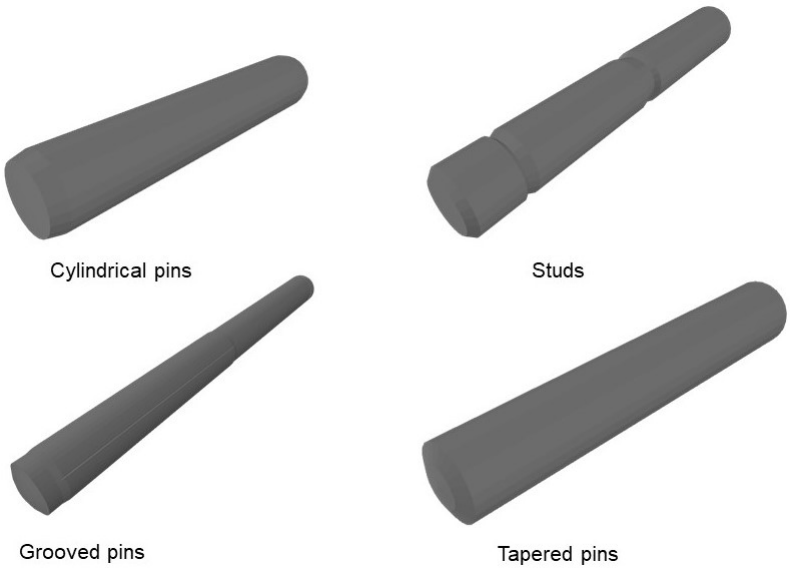


Figure 10: Example of objects that cause miss classification

As mentioned earlier, the dataset contains several components that closely resemble each other. Therefore, even though the parts are detail enough, the deep learning models fails to recognize them. For example, the confusion matrix indicates that miss classification occurred among Cylindrical pins, Grooved pins, Threaded rods, Studs and Tapered pins. This objects are nearly similar to each other. Figure 10 shows the example of those objects.

2.5 Conclusion

In this project, a comparative study of different deep learning models has been conducted to classify 3D CAD models of mechanical components. Two types deep learning models are developed. One is PointNet based on point cloud and the other is VoxNet based on Voxel grid. Two different VoxNet architectures are implied which include a baseline model (VoxNet-base) and my own model (VoxNet-Dense). Results indicate that VoxNet-Dense achieves higher accuracy (81%) than the baseline VoxNet. However, it does not surpass PointNet which is one of the state-of-the-art models for 3D object classification. The next step of this project is to apply a generative adversarial neural network (GAN) on the whole dataset or on the same functional objects. By applying the GAN model, my envision is to generate some novel objects that could be useful for multi-purpose problems.

References

[1] S. Marini, M. Spagnuolo, and B. Falcidieno, "Structural shape prototypes for the automatic classification of 3d objects," *IEEE Computer Graphics and Applications*, vol. 27, no. 4, pp. 28–37, 2007.

[2] S. Jayanti, Y. Kalyanaraman, and K. Ramani, "Shape-based clustering for 3d cad objects: A comparative study of effectiveness," *Computer-Aided Design*, vol. 41, no. 12, pp. 999–1007, 2009.

[3] C. Y. Ip, W. C. Regli, L. Sieger, and A. Shokoufandeh, "Automated learning of model classifications," in *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, SM '03, (New York, NY, USA), p. 322327, Association for Computing Machinery, 2003.

[4] F.-w. Qin, L.-y. Li, S.-m. Gao, X.-l. Yang, and X. Chen, "A deep learning approach to the classification of 3d cad models," *Journal of Zhejiang University SCIENCE C*, vol. 15, no. 2, pp. 91–106, 2014.

[5] F. Wang, L. Kang, and Y. Li, "Sketch-based 3d shape retrieval using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1875–1883, 2015.

[6] M. Rucco, F. Giannini, K. Lupinetti, and M. Monti, "A methodology for part classification with supervised machine learning," *AI EDAM*, vol. 33, no. 1, pp. 100–113, 2019.

[7] L. Zehtaban, O. Elazhary, and D. Roller, "A framework for similarity recognition of cad models," *Journal of Computational Design and Engineering*, vol. 3, no. 3, pp. 274–285, 2016.

[8] V. A. Cicirello and W. C. Regli, "An approach to a feature-based comparison of solid models of machined parts," *AI EDAM*, vol. 16, no. 5, pp. 385–399, 2002.

[9] I. K. Kazmi, L. You, and J. J. Zhang, "A survey of 2d and 3d shape descriptors," in *2013 10th International Conference Computer Graphics, Imaging and Visualization*, pp. 1–10, IEEE, 2013.

[10] Z. Cao, Q. Huang, and R. Karthik, "3d object classification via spherical projections," in *2017 international conference on 3D Vision (3DV)*, pp. 566–574, IEEE, 2017.

[11] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1903–1911, 2015.

[12] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

[13] X.-F. Han, J. S. Jin, M.-J. Wang, and W. Jiang, "Guided 3d point cloud filtering," *Multimedia Tools and Applications*, vol. 77, no. 13, pp. 17397–17411, 2018.

[14] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[15] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, IEEE, 2015.

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[17] P. Min, "Binvox voxel file format specification," 2012.

[18] Dawson-Haggerty et al., "trimesh."

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[20] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[21] F. Chollet *et al.*, "Keras," 2015.