

Group 28: 3D Point Cloud Classification

Adrian Mai, Pryor Vo, Fangzhou Ai, Felix Fagan
Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093

a3mai@eng.ucsd.edu, prvo@eng.ucsd.edu, faai@eng.ucsd.edu, ffagan@eng.ucsd.edu
Github link: https://github.com/Fangzhou-Ai/Group28_19SP_ECE228

Abstract—3D objects detection and recognition become more and more popular nowadays with the power of deep learning. Points cloud is a particular subset of 3D representation of 3D objects. In this project, we wish to perform 3D classification on 3D points cloud dataset

I. INTRODUCTION

The goal of this project is to build a neural network model that could take raw point clouds as an input (sets of 3D coordinates) and calculate predicted object classes as an output. This is a problem subject to constraints that cause different issues to arise than in traditional neural network classification problems. To name a few examples, point clouds are unstructured, unordered sets of data distributed in space so this causes complications with conventionally using convolutional filters in that properties such as stride have arbitrary effect. In addition, the unordered property of point clouds means that different point cloud structures can represent the exact same point cloud, so the model must be invariant to permutation. Naturally, point clouds may also have varying numbers of points depending on the sensor and object so this must be accounted for as opposed to the problem of image classification, where set image sizes can be used. In addition, occlusion, noise, and rotational invariance must be accounted for because these are natural data perturbations in sensors such as lidars. This means the model should be able to be robust to missing points caused by the sensor being blocked as well as if the object was rotated. Despite these difficulties, this is an important problem with applications in computer vision, object detection, scene understanding, robotics, and machine learning.

II. RELATED WORK

Convolutional networks require the format of their inputs to have regularity in size, and thus these networks work best for data that can be formatted as tensors with the same dimensions. However, this proves troublesome for point cloud data, as they are by their nature an unordered collection of an arbitrary number of points. Previous papers have proposed various methods to allow point cloud data to be fed into a deep convolutional network.

Some methods analyze point cloud data to find statistical trends or other characteristics of the points in the data. These statistical trends are then used as features, which can then be used as an input to a CNN. Typically this involves finding

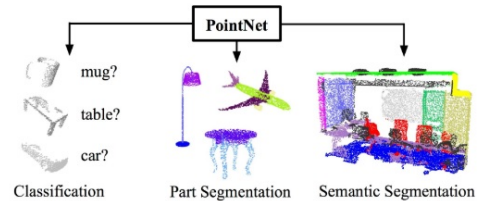


Fig. 1: **Applications of PointNet:** PointNet allows learning global and local features of raw point cloud data, allowing the use of convolutional networks to use point clouds for multiple purposes. Source: C. R. Qi et al. [1]

some feature descriptor that can be used to characterize a specific point relative to the rest of the points in the point cloud. One such feature descriptor is the Heat Kernel Signature (HKS), which is measured by modelling the heat diffusion across the surface of the 3D model as a function of time [3]. Another such descriptor is the Wave Kernel Signature (WKS), which uses a similar method of defining the feature descriptor using some natural phenomena, but instead measures the probability of finding a quantum mechanical particle at a specific point [4]. These or other feature descriptors can then be used to produce a shape signature which characterizes each point cloud as a whole and can be used as the input of the convolutional network [5]. Because the shape signature is more or less handcrafted, this method must essentially be tuned to the needs for the intended application of the convolutional network by choosing which features the user wants to emphasize. The issue with this method is that determining which features are needed can be a difficult proposition and requires prior knowledge and intuition to properly design the network. This method's inability for unsupervised learning defeats the purpose of using a deep convolutional network in the first place, so we want to implement a neural network that is capable of learning from labeled data without having to re-engineer it for each dataset.

Another alternate representation for point clouds is to represent them as two-dimensional images. While convolutional networks have been troublesome to design for three-dimensional data, there has been much success in implementing them to learn from two-dimensional image data. To take advantage of this, [10] describes a method used to project

point clouds onto 2D images from multiple viewpoints and different angles and then inputting those viewpoints into a regular image classifier. An issue with this method is that it is best suited almost exclusively for 3D object classification, and is troublesome to expand to other tasks in 3D like scene understanding or shape completion.

Other methods have attempted to use the implement object classification with data that is three-dimensional in format. A common version of this is to represent point clouds as voxelized shapes (3D shapes discretized as regularly spaced samples on a Cartesian grid), as in [2]. Methods have been implemented that allow these voxelized shapes to be input directly into a convolutional network, such as VoxNet [7]. However, an issue with this approach is that the allowable volumetric resolution of the data used is limited by the computational complexity needed for the convolution of this data. The architecture needed to extract features from very large point clouds would need to be very complex and thus require very long computation times, both for training and forward propagation. This puts a practical limit on the size of the point clouds that can be used for convolutional networks, which is a problem when using point cloud data for whole scenes instead of just single objects.

In addition, little work has been done on tackling the unordered nature of point clouds. Previous research from Oriol Vinyals et al. [8] has provided a method for dealing with unsorted sets by utilizing a read-process-write network that can take unordered sets of data and find ways to sort them. The issue with this approach is that it focuses on generic unordered data and doesn't deal with geometry specifically. As such, it doesn't take into account the inherent characteristics that geometric data can hold and may be unoptimized for use with point clouds.

III. METHODS

We mainly use the PointNet package in this report to perform the 3D point cloud classification here. Before we start to introduce the packages, lets review the property of our dataset again. For classification, the two most important properties here is unordered-ness and invariance, which implies that the dataset is totally permutable and the model should have the same result under different rotation operations. We should bear these in mind because these would be the base rule of how people design the PointNet algorithm. Our implementation of PointNet is written in Python using Keras, based off Y.Li's own implementation [6]. The architecture of PointNet is in Figure 2.

As we're focusing solely on implementing just the classification part, we only need to focus of the blue region. We can treat the procedure largely as two parts: Fetch the features and train a network to classify the object based on these features. The second part is quite easy to be do, while determining how to select the features is most importance, since we have to maintain the properties we have mentioned above, which means the features should be invariant to the input order and rotation.

Usually people would try to train a recurrent neural network and treat the dataset as q series of signals, hoping that training would be independent of input order. However, this is not the truth, as demonstrated in [8]. Though the RNN has good performance on small unordered data, order does matter for datasets with thousands of data points, which is typically size of a point clouds data. In short, RNN is not a good choice here.

For PointNet, the authors of [1] employs a new function to represent the procedure of extracting the features. The function is shown below.

$$f(\{x_1, x_2 \dots x_n\}) \approx g(h(x_1), \dots, h(x_n)) \quad (1)$$

Function f is a mapping from 2^{R^N} to R, h is R^N to R and g is R^{nK} to R.

The author here approximate the h as a multi-layer perceptron and g as a composition of single-value function and max pooling. This framework is simple and effective in the experiment. T-Net in Figure 2 is a mini-transformation network, which include several convolution here. The first would be a convolution over each row of input data to extract the information within one point, then do a convolution for each point to detect those small features pixel-wisely. After extracting, we use max pooling to pick the most important features. This would explain why the PointNet would be irrelevant of the order. Then all the features would be connected to classification by a fully-connected network. In general, we have three main steps here:

1. Extract information pixel-wisely.
2. Pick the most extraordinary features.
3. Fully connected to classification.

This approximation runs pretty well on experiments (classification's accuracy is about 80%). We could do something to support this approximation from the theorem below:

Theorem : suppose $f : \chi \rightarrow R$ is a continuous set function w.r.t Hausdorff distance $d_H(\cdot, \cdot)$. $\forall \epsilon > 0, \exists$ a continuous function h and a symmetric function $g(x_1, x_2, \dots, x_n) = \gamma \cdot MAX$, such that for any $S \in \chi$ we have

$$|f(S) - \gamma(MAX\{h(x_i)\})| < \epsilon \quad (2)$$

where x_1, \dots, x_n is the full list of elements in S ordered arbitrarily, γ is a continuous function, and MAX is a vector max operator that takes n vectors as input and returns a new vector of the element-wise maximum.

To prove this, the key idea is that in the worst case the network can learn to convert a point cloud into a volumetric representation, by partitioning the space into equal-sized voxels. In practice, however, the network learns a much smarter strategy to probe.

Proof: Define $K = [1/\delta_\epsilon]$ so that could cut [0,1] evenly into K intervals, we also need to define a auxiliary function that map the point to the low boundary of it's interval.

$$Auxiliary\ function : \sigma(x) = \frac{\lfloor Kx \rfloor}{K} \quad (3)$$

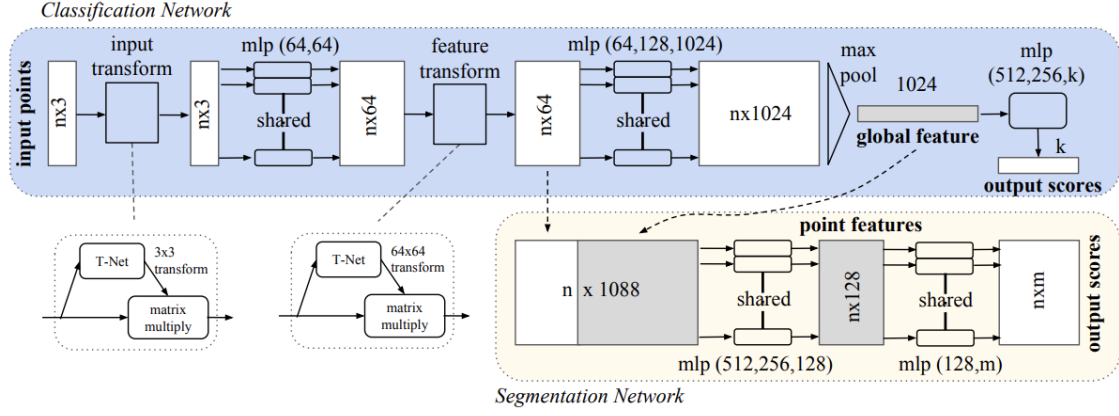


Fig. 2: **PointNet architecture.** The blue-highlighted region is the classification network, and the one below is the segmentation network. Our task was to implement just the classification network which takes n inputs, applies transformations, then aggregates point features by max pooling. In this figure, "mlp" is a multi-level perceptron. . Source: C. R. Qi et al. [1]

Notice $f(\text{Hausdorff distance})$ is continuous, and $d(S, \hat{S}) < \frac{1}{K} < \sigma_\epsilon$, therefore we have $|f(S) - f(\hat{S})| < \epsilon$, where $\hat{S} = \sigma(x) : x \in S$.

Let us write it into an exponential way below, this is a soft indicator function and $d(x, I)$ is the point to interval distance. So we have $h(x) = [h_1(x), h_2(x), \dots, h_K(x)]$ and h here is a $R \rightarrow R^K$ function.

$$h(x) = \exp(-d(x, [\frac{k-1}{K}, \frac{k}{K}])) \quad (4)$$

We define $v_j(x_1, \dots, x_n) = \max\{h_j(x_1), \dots, h_j(x_n)\}$ and $\mathbf{v} = [v_1; \dots; v_n]$, then we have $R \times \dots \times R \rightarrow 0, 1^K$. We also define $\tau : 0, 1^K \rightarrow \chi$ as $\tau(v) = \frac{k-1}{K} : v_k \geq 1$. This function could maps the occupancy vector to a set contains the low boundary of the occupied interval. Then we could easily show that

$$\tau(\mathbf{v}(x_1, \dots, x_n)) = \hat{S} \quad (5)$$

Notice x_1, \dots, x_n are the elements in an certain order.

Let $\gamma : R^K \rightarrow R$ be a continuous function such that $\gamma(\mathbf{v}) = f(\tau(\mathbf{v}))$ for $v \in 0, 1^K$. We have

$$|\gamma(\mathbf{v}(x_1, \dots, x_n)) - f(S)| = |f(\tau(\mathbf{v})) - f(S)| < \epsilon \quad (6)$$

We can decompose the $\gamma(\mathbf{v})$ as following

$$\begin{aligned} \gamma(\mathbf{v}) &= \gamma(\text{MAX}(h(x_1), \dots, h(x_n))) = \\ &(\gamma \times \text{MAX})(h(x_1), \dots, h(x_n)) \end{aligned} \quad (7)$$

It's obvious that $\gamma \times \text{MAX}$ is symmetrical and has a fixed output regardless the input order for a certain data set.

IV. DATASET AND FEATURES

The dataset used in [1] and in this paper is ModelNet40, which is a subset of ShapeNet [2]. This set was preprocessed into .h5 files that are designed to be easily read into NumPy arrays. The data is described by labeled point clouds which are unordered points in 3D space, as shown in Figure 3. They are loaded in by (batch, number of points, x-y-z coordinates)

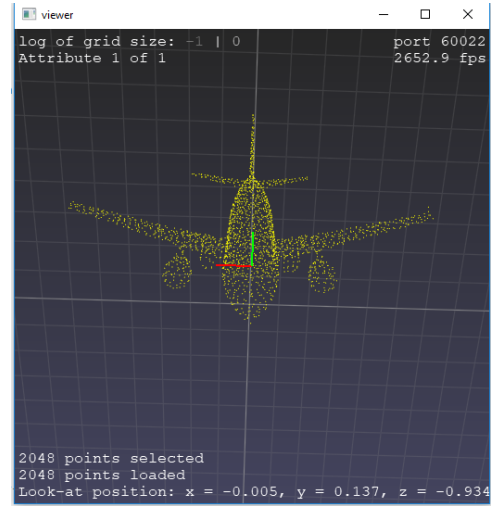


Fig. 3: Example of a point cloud dataset, in this case from the "airplane" class. Point clouds consist of an unordered collection of point defined in 3D space. The number of points in a given point cloud can be arbitrary.

along with a label among 40 classes. Additional examples of point clouds from the training set are shown in Figure 4.

No preprocessing of the dataset was done other than adding artificial noise through jittering. Some data augmentation was performed in adding random rotations to test the robustness of the model but these were not included in the training of the final model or in producing the final results. Feature extraction methods such as PCA and normalization would cause the data to lose its inherent spatial structure which is an essential property of point cloud representation. In addition, each point is already only represented by three dimensions so further dimensionality reduction would lose essential information. Although not performed in this project, further considerable augmentations could be data point deletion which would represent occlusion, possibly further increasing the robustness

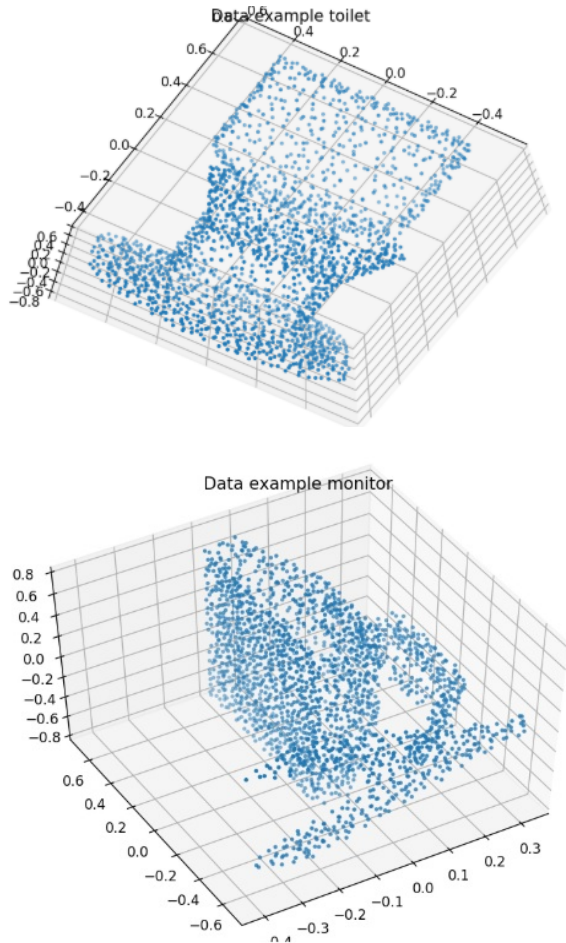


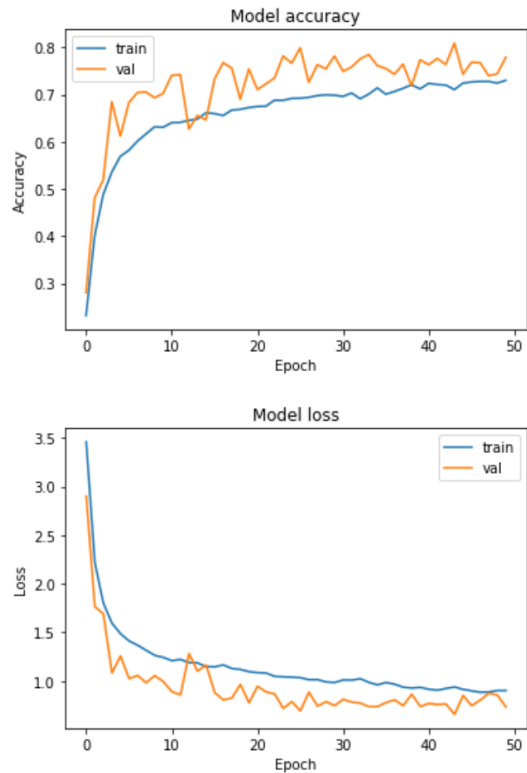
Fig. 4: Sample point clouds from ModelNet40 training set. Top: "toilet" class; Bottom: "monitor" class

of the model. However, it was decided that this was also effectively done with the high dropout rates of the dropout layers which was intended to resist data point redundancy. The training set consists of 9840 samples of 2048 sets of 3 dimensional points, and the test set consists of 2468 samples of 2048 sets of 3 dimensional points.

V. EXPERIMENTS-RESULTS-DISCUSSION

As mentined earlier in this paper, T-net a mini-transformation function while not demonstrated clearly, representing a series of 1D convolutions to extract the features pixel-wisely, which would be used by max pooling and fully connected network for classification. While convolution could be a good choice here, picking another function could be better. The intrinsic property of T-net is more like a N to 1 mapping and it should be sensitive enough to the minor changes of points position, we guess that designing a more computationally efficient function would accelerate the whole frame work.

An interesting note about the architecture is that it appears to effectively deal with the issues of the point cloud data being unordered. As previously stated in the introduction, unordered



Test set accuracy with dropout 0.7: 76.2%

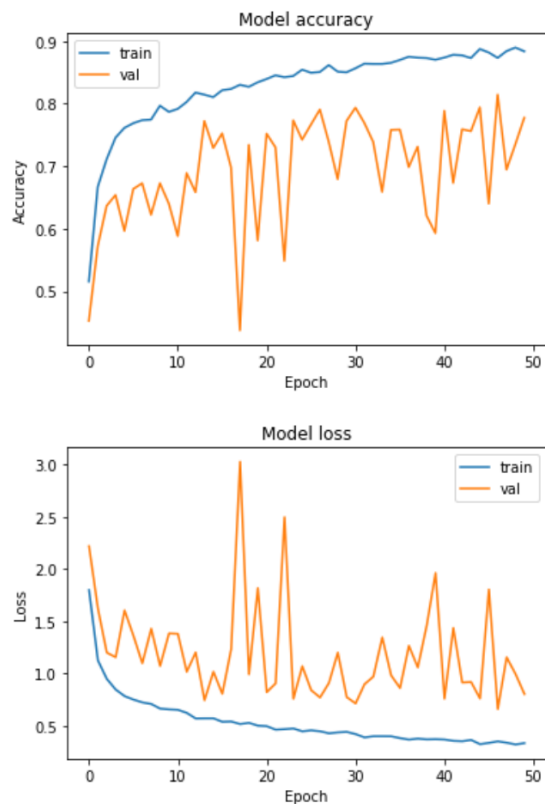
Fig. 5: Train and test accuracy and loss as a function of training epoch for dropout rate set to 0.7. Test set accuracy for the final model is 76.2%

data was not usable for certain convolution methods. As T-net uses 1D convolutional layers in our implementation, it does not share the same issue and is able to train and predict successfully from unordered point cloud data..

We used the following settings and parameters for training loop: the learning rate was set to 0.001, the decay rate was 0.7, the batch size was 32, and we ran training for 50 epochs. The training-validation dataset split was 70% to 20%. For stochastic optimization we used the Adam algorithm.

The metric we used for accuracy was simply the overall classification accuracy (number of correct class predictions over the total number of predictions). The loss metric used for training was defined as the categorical cross-entropy between the prediction and the ground-truth labels.

The loss and accuracy graphs in Figure 5 were produced with a dropout rate of 0.7. The high dropout rate was necessary to account for point redundancy within point clouds which we initially assumed from other implementation and intuition to be a source of overfitting and error. However, it was brought to our attention that the validation accuracy was higher than the training accuracy which was pointed out to supposedly be an error in our model or datasets. Upon further research and experimentation, it was discovered that these results were



Test set accuracy with dropout 0.2: 72.4%

Fig. 6: Train and test accuracy and loss as a function of training epoch for dropout rate set to 0.2. Test set accuracy for the final model is 72.4%

due to the high dropout rate. This makes sense because while the dropout layer is applied to the training set in order to fight overfitting and allow the model to generalize better with less information, dropout is not applied to the validation set. This results in lower loss and higher accuracy calculations on unseen data. The following graphs generated at a dropout rate of 0.2 in Figure 6 support this point.

It is worth noting that when the dropout rate is low at 0.2, overfitting appears to be a significant issue which is shown by the large disparity between the validation and training curves (Figure 6). In comparison, plots at the dropout rate at 0.7 have a significantly smaller difference. Also considering the evaluation results on the testing set, we can see that the accuracy drops from 0.76 to 0.72 when the dropout rate is lowered. This small experiment demonstrates that point redundancy leading to overfitting in the training set is inherently a significant issue to this classification problem, which can be helped with high dropout rates.

The plots in Figure 7 are randomly generated point cloud samples from the test set, along with their respective class predictions and probabilities from the model. As the results show, the point clouds clearly show what their ground-truth

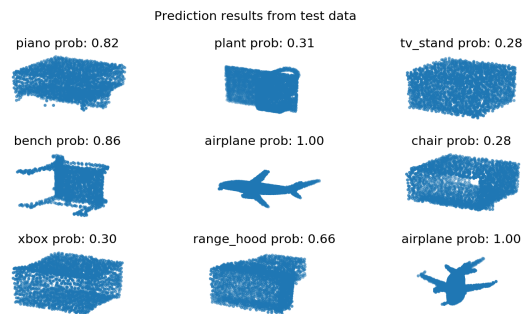


Fig. 7: Sample point clouds from the ModelNet40 test set and their respective class predictions and confidence scores.

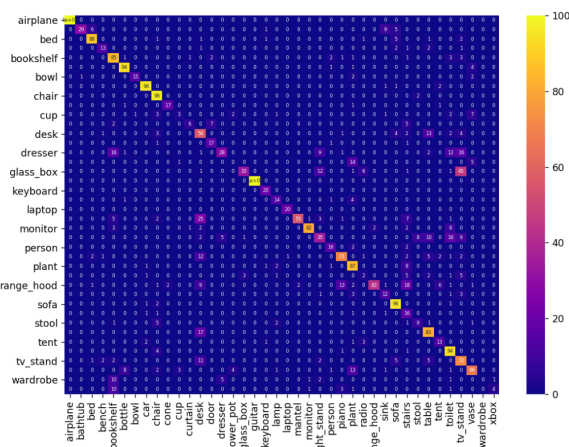


Fig. 8: Confusion matrix of test data. Blue squares indicate no confusion (0%) between classes and yellow squares indicate total confusion (100%) between classes.

labels should be, and the corresponding predictions are accurate. However, some classes seem to produce more confident predictions than others. Our hypothesis is that these low class probabilities occur for classes with shapes very similar to other ones. For example, classes with more distinct shapes like "airplane" (~100% confidence) and "piano" (~70% confidence) had very high class probabilities, but classes with much lower scores, like "xbox" (~30% confidence) and "tv_stand" (~28% confidence), are very similar in shape. This can be seen in the confusion matrix for the test data in Figure 8, where the class-pairs with higher confusion probabilities are all ones with similar box-like shapes. For example, "glass box" and "tv_stand" have a very high confusion rating together (45%), and "desk" has high ratings with multiple classes, including "mantel" (25%), "table" (17%), and "tv_stand" (11%). Each of these groups of classes have point clouds that are very similar in shape, and it can be predicted that this similarity leads to a lack of distinct features that PointNet can use to differentiate the classes during training. Because of this, predictions for these labels are much less confident, which may lead to potential misclassifications for similar classes.

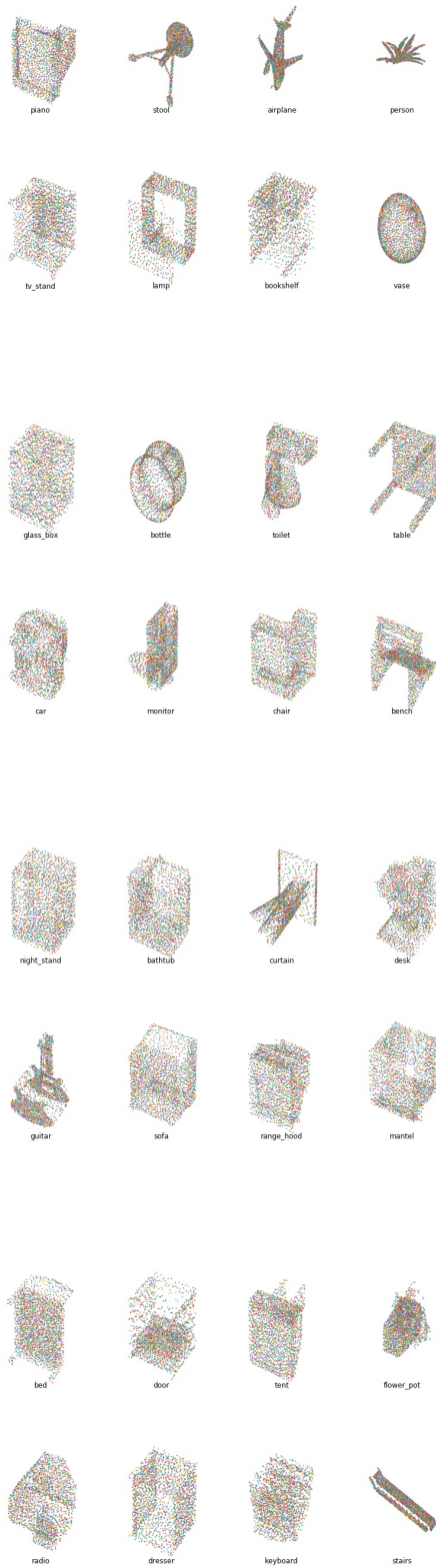


Fig. 9: Additional test set sample and predictions. The predicted label for each sample is printed alongside their respective point cloud.

Figure 9 shows additional predictions from the test set. The point clouds are plotted along with their predicted labels. We obtained an overall accuracy of 80.55% on the test set, which is a very good result. We suspect that the misclassifications are due to similarity in the geometric shape of the different classes. Some objects such as TVs and toilets have very similar shapes, making it hard for the model to differentiate features and learn. Also, the data set may not be perfect, after jittering it may cause distortion and result in further misclassification.

VI. CONCLUSION AND FUTURE WORKS

T-net here is a mini-transformation function while not demonstrated clearly in this paper. After looking through the code, the T-net in fact represents a series of 1D convolution to extract the features pixel-wisely, which would be used by max pooling and fully connected network for classification. While convolution could be a good choice here, picking another function could be better. The intrinsic property of T-net is more like a N to 1 mapping and it should be sensitive enough to the minor changes of points position, we guess that designing a more computationally efficient function would accelerate the whole framework.

It appears that to take care of the issue of the point clouds being unordered and therefore not applicable to certain convolution methods involving stride, the model uses a series of 1D convolutional layers which does not share the issue. Dropout layers with high dropout rates are used to prevent overfitting of the training data as some point clouds may have many redundant points which results in higher performance on unseen data.

We are able to make prediction based on the model up to 80% of accuracy on the test set. Given that the data used consists of raw point cloud with no projections or other preprocessing, these results are fairly impressive.

If there was more time, an interesting application for PointNet we wanted to pursue is expanding the model base for PointNet for use with 3D object detection. The authors for [1] have worked on a newer version of this model known as Frustum PointNet [9] that can perform object detection from point cloud data of entire scenes. The authors accomplished this functionality and displayed its performance using point cloud data taken from RGB-D data from both indoor and outdoor scenes, as shown in Figure 10. Implementing 3D scene reconstruction and object detection using this model was our original goal for this paper, but due to time constraints we decided to scale our efforts back to 3D object classification.

Possible further development from this model includes point cloud segmentation of 3D scenes, as well 3D point cloud reconstruction using additional data.

VII. CONTRIBUTIONS

Team member contributions, as reported by them:

Adrian Mai was coordinator for all of the meetings. He makes sure that everyone is doing the fair amount of work for the project. He trained the network and the visualization

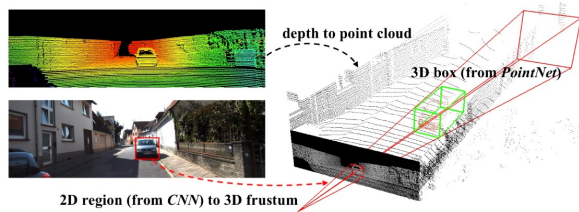


Fig. 10: Frustum PointNet. Using RGB-D data, region proposals for objects are first generated on the 2D RGB image. These regions are then projected onto 3D space, defining a volume known as a *3D viewing frustum*, within which 3D depth information is sampled Source: C. R. Qi et al. [9]

to obtain the 40 objects classification figures. In addition, he also write up his part on the report.

Fangzhou Ai was in charge of theory part. He do the derivation part to finish the poster and report's method part. He also wrapped the input T-net and feature T-net as a module employed in our code.

Pryor Vo was the Vice Chancellor to the coordinator of scheduling all meetings. He worked on producing quantitative results and the analysis that in this report in addition to doing the experiments on dropout. He also trained each model with the new hyperparameters accordingly for those experiment . He also produced visualizations including the loss and accuracy graphs pictured, the random prediction results with scores, and the confusion matrix.

Felix Fagan did the research into other methods and related papers, and wrote the related works section. In addition, he worked with Pryor on the analysis and results sections, and helped with visualizations for the confusion matrix and model predictions with class prediction scores. He was also responsible for formatting and editing the final report.

REFERENCES

- [1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, arXiv:1612.00593[cs.CV], April 10, 2017.
- [2] A. Khosla, S. Song, X. Tang, Z. Wu, J. Xiao, F. Yu, and L. Zhang, 3D ShapeNets: A Deep Representation for Volumetric Shapes, in *Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)*, 7-12 Jun. 2015, Boston, MA, USA [Online]. Available: <https://modelnet.cs.princeton.edu/>. [Accessed: Jun. 3, 2019]
- [3] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 13831392. Wiley Online Library, 2009.
- [4] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to Shape analysis. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 16261633. IEEE, 2011.
- [5] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *Transactions on Graphics (TOG)*, 21(4):807832, 2002.
- [6] Y. Li, Keras implementation for Pointnet, (2018), GitHub repository, Jan. 2018. [Online]. Available: <https://github.com/garyli1019/pointnet-keras>. [Accessed: Jun. 3, 2019].
- [7] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2015.
- [8] O. Vinyals, S. Bengio, and M. Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

- [9] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guiba, Frustum PointNets for 3D Object Detection from RGB-D Data arXiv:1711.08488v2 [cs.CV], April 13, 2018.
- [10] C. R. Qi, H. Su, M. Niener, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view cnns for object classification on3d data. In *Proc. Computer Vision and Pattern Recognition(CVPR)*, IEEE, 2016.