Exploration and Assessment of Parametric and Non-Parametric Networks for 3D Point Cloud Classification

Ning-Yuan Li
University of Pennsylvania
Philadelphia, PA
ny0221@seas.upenn.edu

Jeffrey Li
University of Pennsylvania
Philadelphia, PA
lijeff@seas.upenn.edu

***Abstract*—Processing 3D point cloud data is important for applications in self-driving cars, robotics, and virtual and augmented reality. Qi et al. (2017) introduced the PointNet architecture which performs classification tasks after being trained on point cloud data [1]. The group based their design decisions on three properties of point clouds, namely permutation invariance, transformation invariance, and interactions among points [1]. On the other hand, Zhang et al. (2023) offers a newer non-parametric approach in solving the same problem. Non-parametric building blocks are stacked across multiple stages to construct a pyramid hierarchy [2]. We will be exploring and evaluating both models. Firstly, we will focus on implementing both networks and testing them on a smaller version of the ModelNet40 dataset [3]. Secondly, we perform a series of tests on robustness on PointNet, as we augment the data during inference. Lastly, we will visually assess the internal encodings of the multistep hierarchical layers inside the non-parametric encoder of the Point-NN to understand how it captures spatial representations for classification tasks.**

***Index Terms*—deep learning, computer vision, classification, PointNet, Point-NN, 3D Vision**

I. Introduction and Related Works

A volume (3D) image represents a physical quantity as a function of three spatial coordinates. Processing 3D image data is crucial, yet practical, with applications in fields including self-driving cars, robotics, and virtual and augmented reality. Various approaches have been designed to work with and learn 3D data. One pioneering approach involves using 3D convolutional neural networks (CNN) trained on voxelized shapes [1]. However, there are limitations to this method, notably high computational costs due to increased input sizes, data sparsity from having complex 3D shapes, and reduced image representation due to computational constraints [1], [4].

Alternatively, point clouds are 3D data structures, representing a collection of data points in a three-dimensional coordinate system (x, y, z) with no connectivity. Qi et al. (2017) introduced the PointNet architecture which inputs point clouds and performs computer vision tasks, namely classification and part and semantic segmentation [1]. The group based their design decisions on three properties of point clouds:

- Permutation Invariance: Point clouds are unordered and thus the data process must be invariant to different representations.
- Transformation Invariance: Predictions should not change if objects undergo certain transformations, including rotation and translation.
- Interactions Among Points: The interactions among neighboring points are meaningful, and thus the model needs to understand local structures [1].

The architecture of their PointNet is relatively straightforward and can be found in **Fig. 1**. For classification tasks, the network uses a shared multi-layer perceptron (MLP) to map the n input points from 3 dimensions to 64 dimension and again from 64 dimensions to 1024 dimensions [1]. The input data is first passed through an input transform layer which normalizes the input to remove global transformations and make the model transformation invariant [1]. The feature transform layer follows the first shared MLP layer and has a similar task but at feature level, capturing local transformations and learning local interactions [1]. Max-Pooling is used following the second MLP layer and creates a global feature vector. Lastly, a three-layer fully connected (FC) layer is used to output k classification scores [1].

Since the inception of PointNet, several advanced architectures have surpassed the model's capabilities. Networks achieve fractional improvements through adding advanced local operators and scaling up learnable parameters, leading to oversized models, complex network designs and increased computational resources required for training [2], [5]. Underlying most of these networks are a series of non-parametric components, all of which are learnable during the training stage [2], [5]. Deviating from this trend of increasing parameters, Zhang et al. (2023) introduced a Non-parametric Network, termed Point-NN, which achieves favorable performance on 3D vision tasks [2]. Point-NN consists of purely non-learnable components farthest point sampling (FPS), k-nearest Neighbors (k-NN), and pooling operations, with trigonometric functions. The architecture is composed of a non-parametric encoder and a point-memory bank, shown in **Fig. 2**. For classification tasks, the encoder summarizes the the high-dimensional global feature of a point cloud while the memory bank produces classification logits through similarity

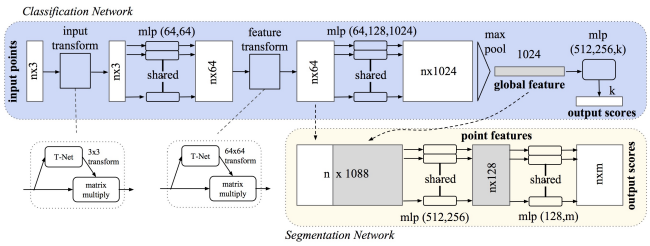

Fig. 1. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes [1].

matching [2].

In this project, we examined both PointNet and Point-NN networks. Firstly, we assessed the advantages and drawbacks of the PointNet architecture originally proposed by Qi et al. Some of the advantages of PointNet include being permutation-invariant, meaning it can process point clouds regardless of order of points, and having good generalization to real-world, noisy, and incomplete 3D image data [1], [4]. However, PointNet only extracts global features and cannot capture fine and complex local features. By not using the local structure of the data, learning may be difficult [6]. Secondly, we explored Point-NN with the goal of understanding how the network captures patterns and structural variations in point clouds. Zhang et al. (2023) claimed that PointNet utilizes trigonometric functions to extract spatial structures in 3D point clouds, which are later recognized by the point-memory blank [2].

In the next section, we detail our methods in exploring and assessing PointNet and Point-NN.

II. Dataset

For this project, we implemented and used parametric and non-parametric based architectures and trained the networks for point clouds classification tasks. Due to limited computing resources, we use a smaller version of ModelNet40 for our dataset, which is the same dataset used by the original authors, containing point cloud data for three household objects: chairs, vases, and lamps [3], [7]. In the training set, there are 4489 instances of chairs, 741 instances of vases, and 1554 instances of lamps. In the testing set, there are 617 instances of chairs, 102 instances of vases, and 234 instances of lamps. Each sample in the dataset has 10000 data points in three-dimensional coordinate system *(x, y, z)* format [8].

III. Methods and Motivations

The first portion of the project focused on implementing PointNet. We carefully followed the implementation guidelines based on the original paper as well as the supplementary resources provided on the group's GitHub page [9]. During the training process, we noticed very high training time and very high memory usage, which prompted us to make the following deviations from the original code. Firstly, we removed both transformation network (T-Net) components from our architecture. Qi at al. introduced T-Net as a mechanism

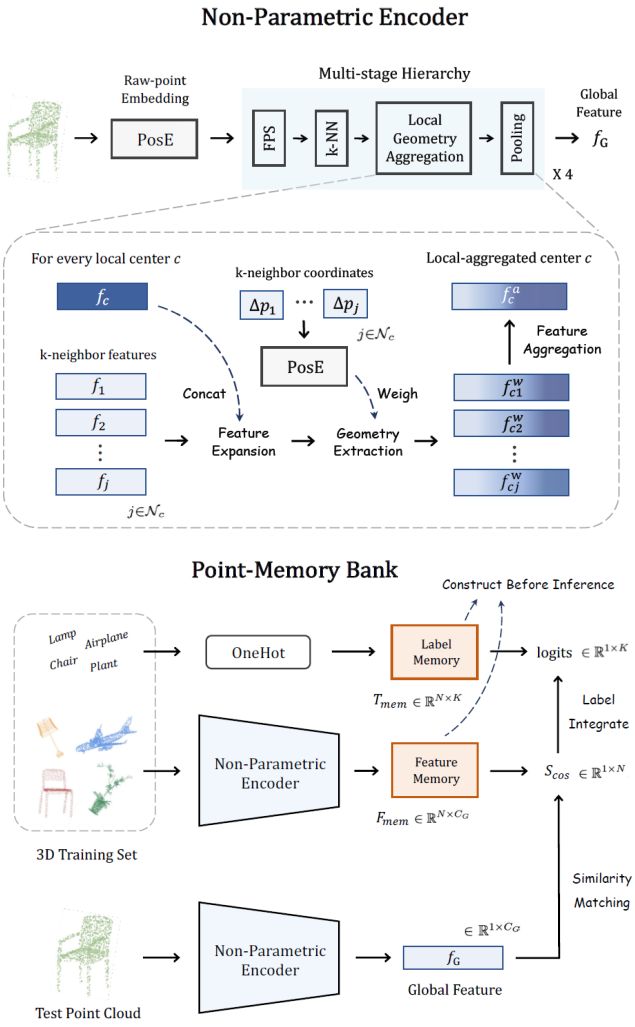

Fig. 2. **Point-NN Pipeline.** Non-Parametric Encoder (Above). Zhang et al. (2023) utilized trigonometric functions to encode raw Point Clouds points into high-dimensional vectors in PosE [2]. The vectors then pass through a series of hierarchical non-parametric operations, namely FPS, k-NN, local geometric aggregation, and pooling, where they will be encoded into a global feature f_G [2]. Point-Memory Bank (Below). The training set features are passed through the Point-Memory Bank outputting F_{mem}, which is later used to classify using similarity matching [2].

for PointNet to be invariant to input permutations and to improve its generalization across different spatial configurations of point clouds [1]. The group also compared network and computational complexity between PointNet with and without input and feature transformations, revealing a 4.3 times increase in parameters and nearly 3 times increase in FLOPS/sample [1]. Moreover, the most basic architecture already achieved reasonable results, and the addition of input transformation only yields a mere 0.8 percent performance boost [1]. By those reasons, we decided to remove the T-Net entirely. Secondly, we changed the batch-size from 32 to 8 as we encountered CUDA memory issues with training large batch-sizes on Google Colab and on local GPU.

The second portion of the project focused on qualitatively

and quantitatively analyzing PointNet through standard training and testing curves and robustness testing. Average loss and accuracy were recorded after every epoch of the training and testing loops. We discuss our findings in the next section. From the original paper by Qi et al., the group achieved high robustness with respect to perturbation and corruption [1]. For our project, we examined robustness in two ways. The first experiment is through data augmentation, in which we perform preset rotations on the point clouds by a set degree, notably (5 degrees, 30 degrees, 45 degrees, and 90 degrees) [8]. The second experiment is through data corruption, in which we deliberately restricted the point clouds to a set number of points [8]. Each point cloud has 10000 points and we randomly sample without replacement a fixed number of points to represent the object, notably 7500. 5000, 2500, 1000, 500, and 100 points. The purpose of these two experiments were to challenge the group's claim that the network learns to summarize a shape by a sparse set of key points [1]. Moreover, we also wanted to replicate real-life data, which may not always be ideal in nature.

Originally, the final portion of the project was to improve and assess our vanilla PointNet architecture with newer architectures, namely PointNet++, DGCNN, and/or Point Transformers, with the goal of borrowing and implementing key design decisions from those networks [6], [7], [10]. For example, PointNet++ achieves better understanding of spatial relationships within a point cloud through distinguishing between local and global features [6]. However, after consulting with Dr. Shi during our project check-in, we decided to diverge from our proposed path and study state-of-the-art non-parametric networks, namely Point-NN instead [2].

We divided the final portion of our project into two components. Firstly, we wanted to examine Point-NN's performance for classification tasks in comparison to the results from PointNet. In the original paper, Zhang et al. (2023) implemented a simple similarity matching technique for classifying point clouds without training, in which the cosine similarity between the encoded memory knowledge from training F_{mem} and the test feature vector f_G^t is computed using the equation:

$$S_{\cos} = \frac{f_G^t F_{\text{mem}}^T}{\|f_G^t\| \cdot \|F_{\text{mem}}^T\|} \in \mathbb{R}^{1\times N} \tag{1}$$

where N represents the number of training samples [2]. The label memory T_{mem} is one hot-encoded, matrix multiplied by S_{cos}, and later passed through an activation function $\phi(x) = \exp(-\gamma(1-x))$, as represented by

$$\text{logits} = \phi(S_{\cos} T_{\text{mem}}) \in \mathbb{R}^{1\times K} \tag{2}$$

where K represents the number of categories and the more similar feature memory of high score contributes greatly to the final logits [2]. We carefully followed the implementation guidelines based on the original paper as well as the supplementary resources provided on the group's GitHub page [11].

Lastly, we wanted to understand how the network encodes three-dimensional semantics through visualizing the internal representations at each hierarchical layer. The multi-stage hierarchy stages is depicted in **Fig. 2**. We saved the results of farthest point sampling (FPS), k-Nearest Neighbors, and geometric aggregations and performed visual comparisons of the high-dimensional encodings through portraying the first three-dimensions and/or performing t-distributed stochastic neighbor embedding (T-SNE) for dimensionality reduction.

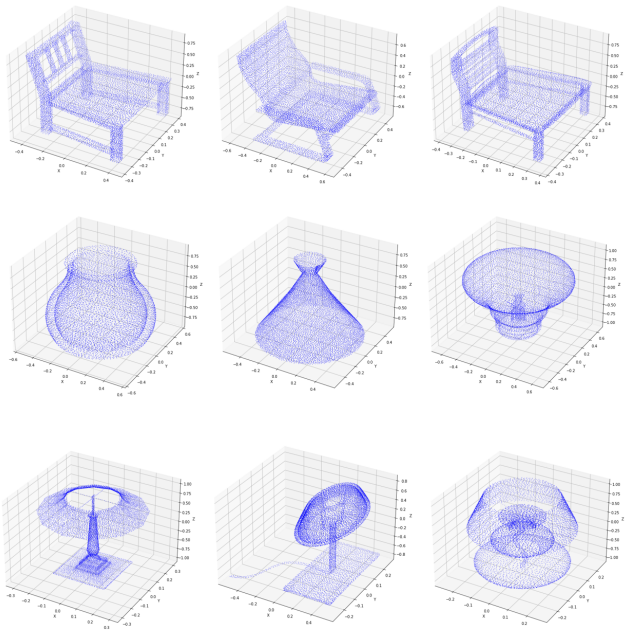

Fig. 3. Sample images from the ModelNet40 dataset depicting three household objects: chairs (top), vase (middle), and lamps (bottom).

IV. Results and Discussion

A. PointNet Training and Testing

From an initial glance of the training dataset, we see a class imbalance given the higher number of samples depicting chairs compared to lamps and vases. Some initial visualizations for the point cloud data have been plotted in **Fig. 3**. We trained our vanilla PointNet implementation for 250 epochs, with a batch-size of 8, Adam optimizer with $\beta = (0.9, 0.999)$, learning rate of 0.001 with no scheduler, and cross-entropy loss function.

The results from training and validation are depicted in **Fig. 4**. We see that both training and testing accuracies are very high, 0.996 and 0.977 respectively. On the other hand, we see the testing loss began to increase gradually after around 50 epochs. After examining the results, we found that all the chairs were classified correctly in the testing dataset and only vases and lamps were misclassified. To understand the result, we can see in **Fig. 3** that chairs have a more rigid and common geometry whereas lamps and vases are more roundish in nature and can take the form of various shapes. Moreover, the large number of chair instances compared to lamps and vases may have introduced bias in the model.

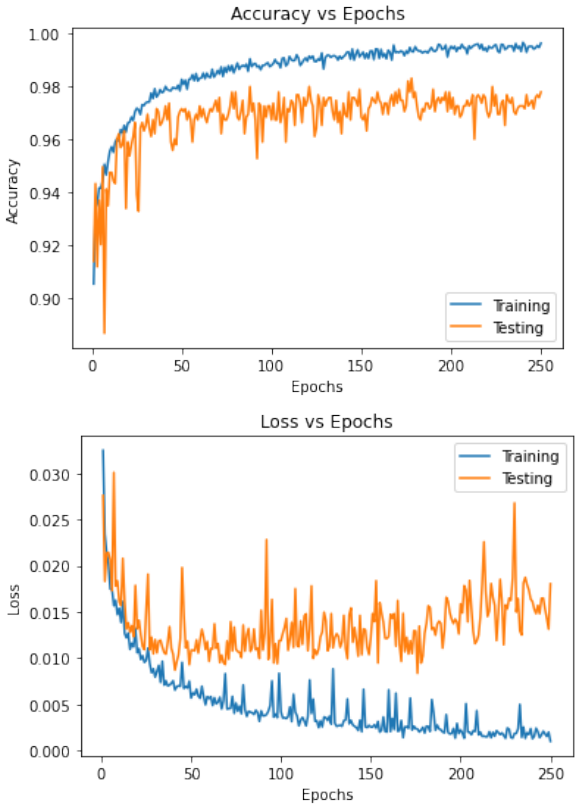

Fig. 4. PointNet Training and testing curves over 250 epochs.

The best model was selected as the set of model parameters that achieved the highest accuracy during training time. We performed robustness testing with the best model in the following section.

B. PointNet Robustness Testing: Data Corruption and Rotation

For data corruption, we experimented with sampling without replacement different number of points from each point clouds in the testing dataset. Each point cloud has 10000 points and we randomly sample without replacement a fixed number of points to represent the object, notably 7500. 5000, 2500, 1000, 500, and 100 points. We report our results in **TABLE I**. The best model's performance during inference only slightly decreases as the number of available points decreases. This result may be attributed to PointNet's ability to capture both global and local features, respectively. Even if the number of points change, the global geometry remains fairly similar allowing the model to still generate accurate classifications

Regarding data augmentation, we rotated each point cloud in the testing set by a preset degree, namely 5 degrees, 30 degrees, 45 degrees, and 90 degrees. We report our results in **TABLE I**. The best model's performance during inference decreases as the rotation of the point clouds increases. This may be attributed to PointNet's lack of appropriate mechanisms for handling geometric transformations such as rotation [1], [9]. The architecture is designed to be permutation invariant; however, the introduction of rotation may introduce changes to the point clouds' spatial arrangement and attribute to the model's sensitivity to objects in different orientations [1], [9].

Furthermore, we suspect the lack of the Transformation-Net components, namely input and feature transformation, has negatively contributed to the model's ability to handle variations in the input point cloud's orientation and position. The goal of the input transformation is to normalize and align the input point cloud, making it invariant to transformations like translation and rotation. On the other hand, the feature transformation allows the model to learn meaningful representation from the input point cloud [1].

Number of Samples	Accuracy	Degree of Rotation	Accuracy
10000	98.3%	0°	98.3%
7500	98.3%	5°	98.1%
5000	98.3%	30°	73.6%
2500	98.1%	45°	43.2%
1000	97.5%	90°	25.3%
500	97.0%		
100	94.6%		

TABLE I
RESULTS FROM ROBUSTNESS TESTING

Examples of point clouds sampled and rotated and their results can be found in the Appendix and in our code.

C. Point-NN Non-Parametric Classification

For the non-parametric classification of Point-NN, we achieved an accuracy of 0.908 when performing the steps mentioned in equations **(1)** and **(2)**. Despite achieving lower accuracy compared to the PointNet-based model, the Point-NN network showcased its ability to obtain satisfactory classification results with no training and substantially less computational resources [5].

D. Point-NN Non-Parametric Visualizing Internal Encodings

We obtained visualizations of our internal encodings in the multi-stage hierarchy to understand how each non-parametric operation captures features. Point-NN and image classification share similar data processing strategies. In deep neural networks for image classification problems, the spatial dimension of the feature maps decrease while the depth (number of channels) and size of the receptive field increases. Similarly, in Point-NN, we use FPS to downsample the point cloud's point density, resulting in smaller sizes at later stages while also performing feature expansion. In 2D image classification, CNNs use convolutional layers to capture information from adjacent pixels. Each pixel in a layer extracts information from a local neighborhood of the previous layer using these convolutional kernels. Point-NN adopts this concept for 3D point cloud analysis. Instead of convolutional kernels, Point-NN uses the k-Nearest Neighbors (k-NN) method. For each point in the downsampled point cloud (after FPS), k-NN identifies the nearest N points in the original point cloud (before FPS).

Representations can be found in the Appendix and in our code.

V. Conclusion

Point cloud processing is important in several fields. In this project, we examined and assessed two different methods for classifying point clouds on a point cloud dataset. The PointNet architecture provides a way to learn global and local point features while also achieving permutation invariance, whereas the Point-NN network offers a way to capture features, like spatial patterns and structures, through a non-parametric approach. Through our experiments, we built and implemented both networks to understand the different mechanisms that allow them to perform classification tasks. Though PointNet achieves higher accuracy when evaluated on the test dataset, we acknowledge the ability for Point-NN to achieve reasonable accuracy with no training and substantially less computational resources. Additionally, we performed robustness tests, specifically data corruption and rotations, in order to evaluate PointNet's ability to classify objects despite changes to orientation and structure. From our tests, we found that PointNet is capable of recognizing global structures of objects despite missing points, but it suffers when large rotations are applied to the point clouds. Lastly, we visually analyzed the internal representations of the multistep hierarchical layers inside the non-parametric encoder to understand how Point-NN captures meaningful representations in point cloud data.

We have several directions in which we can expand upon our existing work. Firstly, there are several relevant and more advanced architecture capable of processing point cloud data, while also performing other vision tasks, such as segmentation and object detection, that we can further explore. PointNet is a stepping stone to more exciting networks, such as PointNet++, DGCNN, and Point Transformers, in which several changes and design decisions have been made to account for the drawbacks of the original PointNet [6], [7], [10]. Secondly, Point-NN's utility can be expanded upon as a plug-and-play module to boost existing learnable 3D models without further training [2]. Applying Point-NN's ability to capture spatial representations will enhance the shortcomings to baseline models such as PointNet. Thirdly, acquiring access to greater computing resources will allow us to build upon our existing work through being able to keep and run the Transformation-Network as well as the full ModelNet40 point cloud dataset.

References

[1] 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 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [Online]. Available: https://arxiv.org/abs/1612.00593

[2] R. Zhang, L. Wang, Z. Guo, Y. Wang, P. Gao, H. Li, and J. Shi. Parameter is not all you need: Starting from non-parametric networks for 3d point cloud analysis. [Online]. Available: https://arxiv.org/abs/2303.08134

[3] Princeton modelnet. Available at https://modelnet.cs.princeton.edu/. Princeton University.

[4] N. Karaev. Deep learning on point clouds: Implementing pointnet in google colab. Available at https://towardsdatascience.com/deep-learning-on-point-clouds-implementing-pointnet-in-google-colab-1fd65cd3a263.

[5] R. Zhang, L. Wang, Z. Guo, and J. Shi. Nearest neighbors meet deep neural networks for point cloud analysis. [Online]. Available: https://arxiv.org/abs/2303.00703

[6] C. R. Qi, K. Mo, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Conference on Neural Information Processing Systems (NIPS) 2017*. [Online]. Available: https://arxiv.org/abs/1706.02413

[7] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," in *Computer Vision and Pattern Recognition*. [Online]. Available: https://arxiv.org/abs/1612.00593

[8] S. Tulsiani, H. Mittal, and H. Wu. assignment5. Available at https://github.com/learning3d/assignment5/tree/main.

[9] C. R. Qi. pointnet. Available at https://github.com/charlesq34/pointnet/tree/master.

[10] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun, "Point transformer," in *Computer Vision and Pattern Recognition*. [Online]. Available: https://arxiv.org/abs/2012.09164

[11] R. Zhang. Point-nn. Available at https://github.com/ZrrSkywalker/Point-NN.

APPENDIX A
POINTNET ROBUSTNESS TESTING: ROTATION

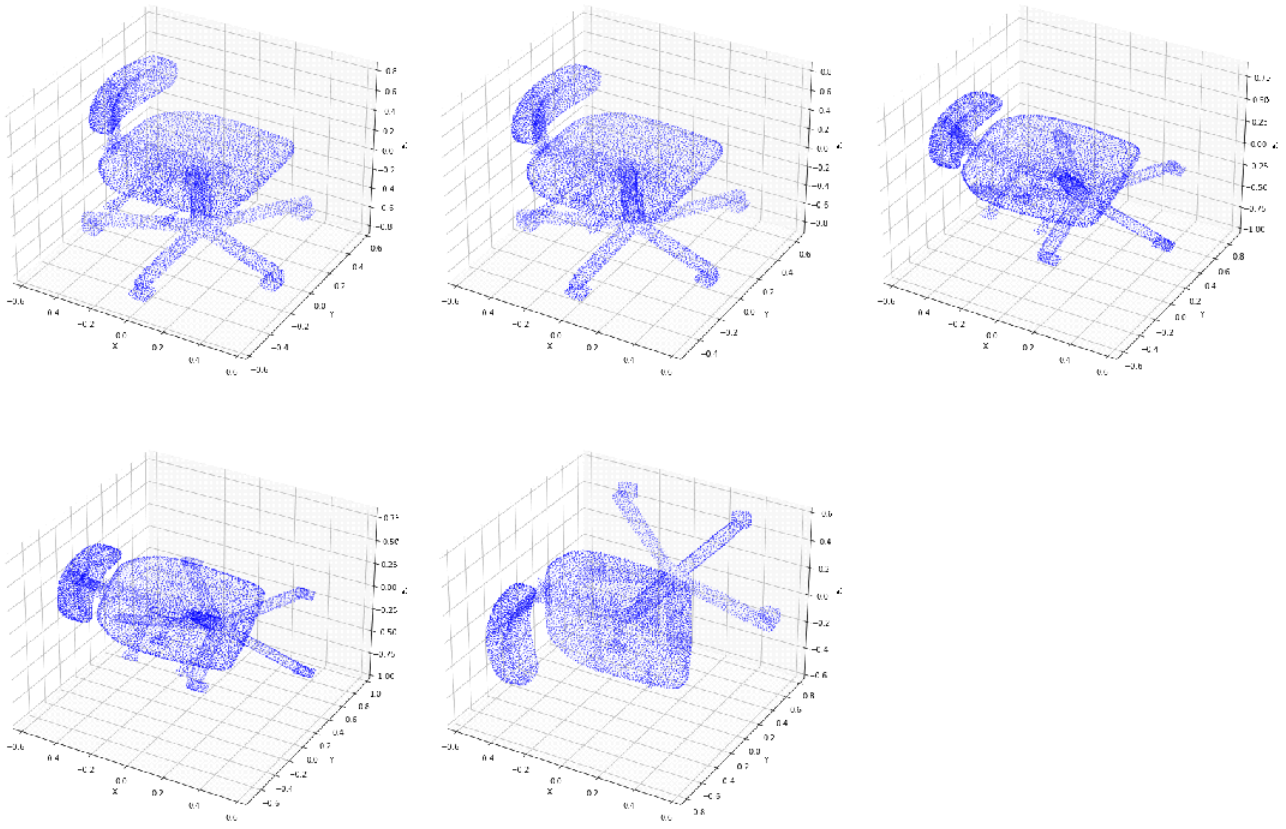

Fig. 5. Sample point cloud demonstrating a chair undergoing rotations (left to right): 0 degrees, 5 degrees, 30 degrees, 45 degrees, 90 degrees. The model incorrectly classifies the chair starting at the 30 degrees rotation.

APPENDIX B
POINTNET ROBUSTNESS TESTING: DATA CORRUPTION

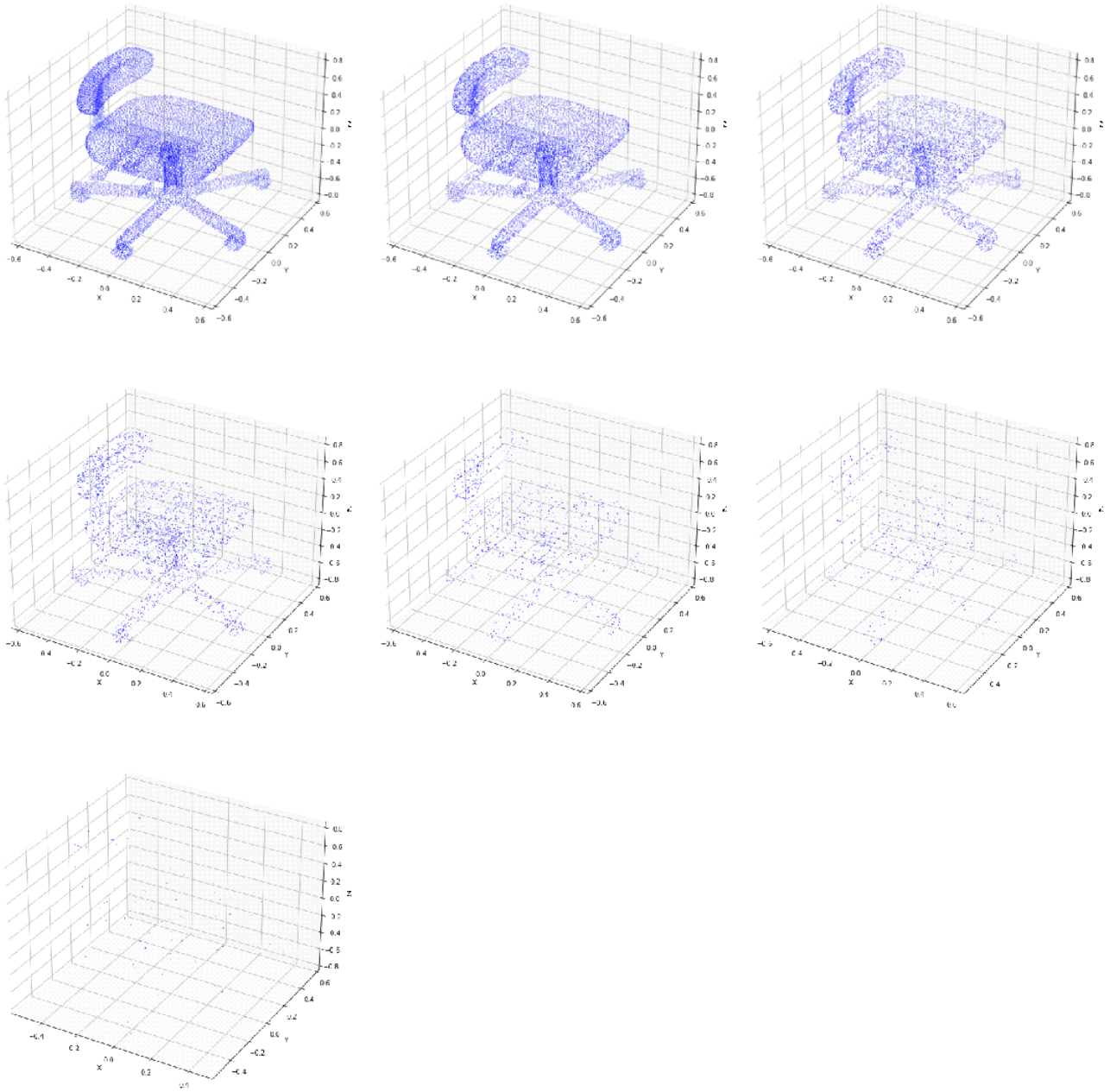

Fig. 6. Sample point cloud demonstrating a chair undergoing sampling (left to right): 10000 points, 7500 points, 5000 points, 2500 points, 1000 points, 500 points, 100 points. The model correctly classifies the chair at all different samplings.

APPENDIX C
FARTHEST POINT SAMPLING (FPS) ENCODING

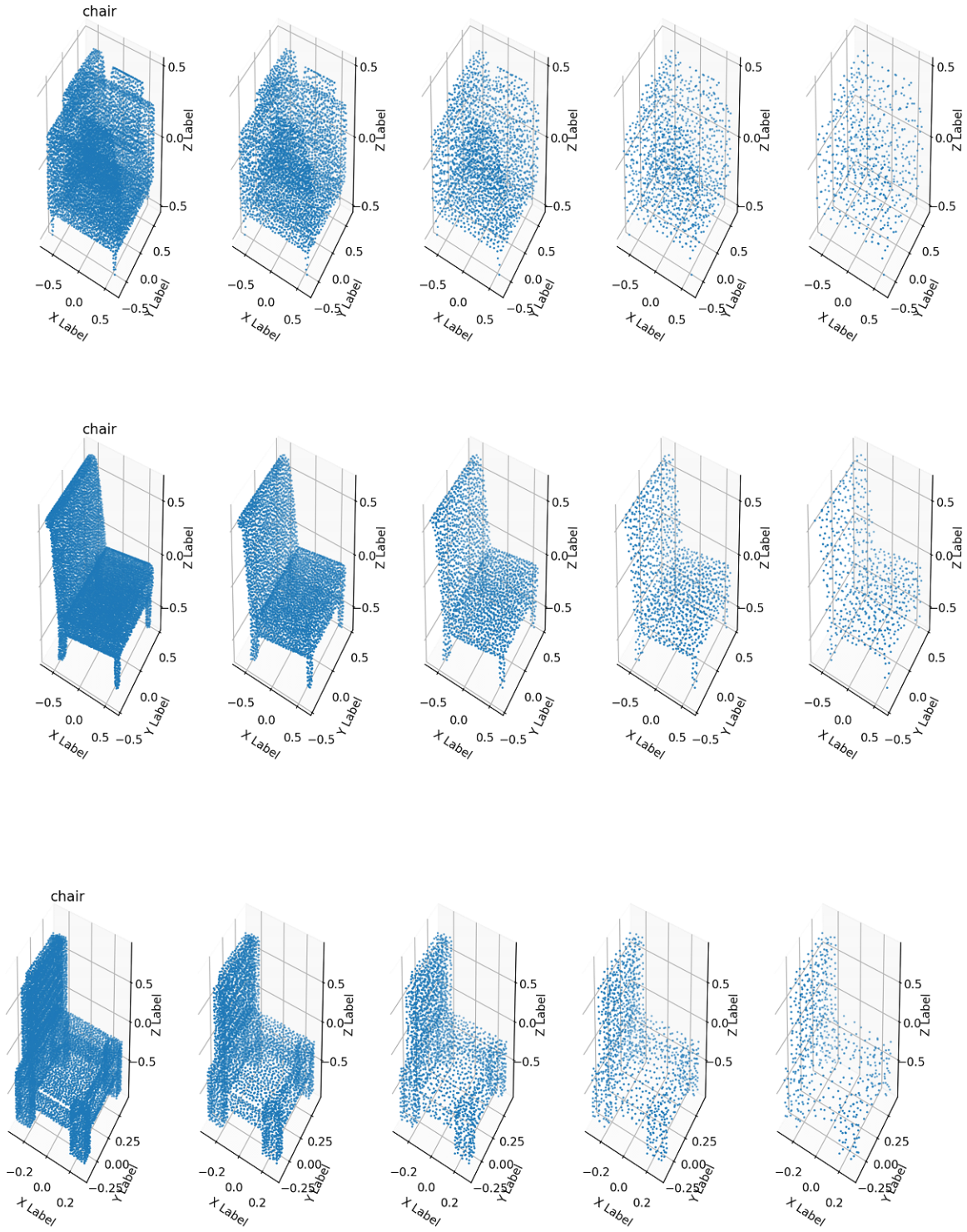

Fig. 7. Sample point clouds undergoing four FPS iterations (left to right): 10000 points, 5000 points, 2500 points, 1250 points, 625 points.

APPENDIX D
K-NEAREST NEIGHBOR (K-NN) ENCODING

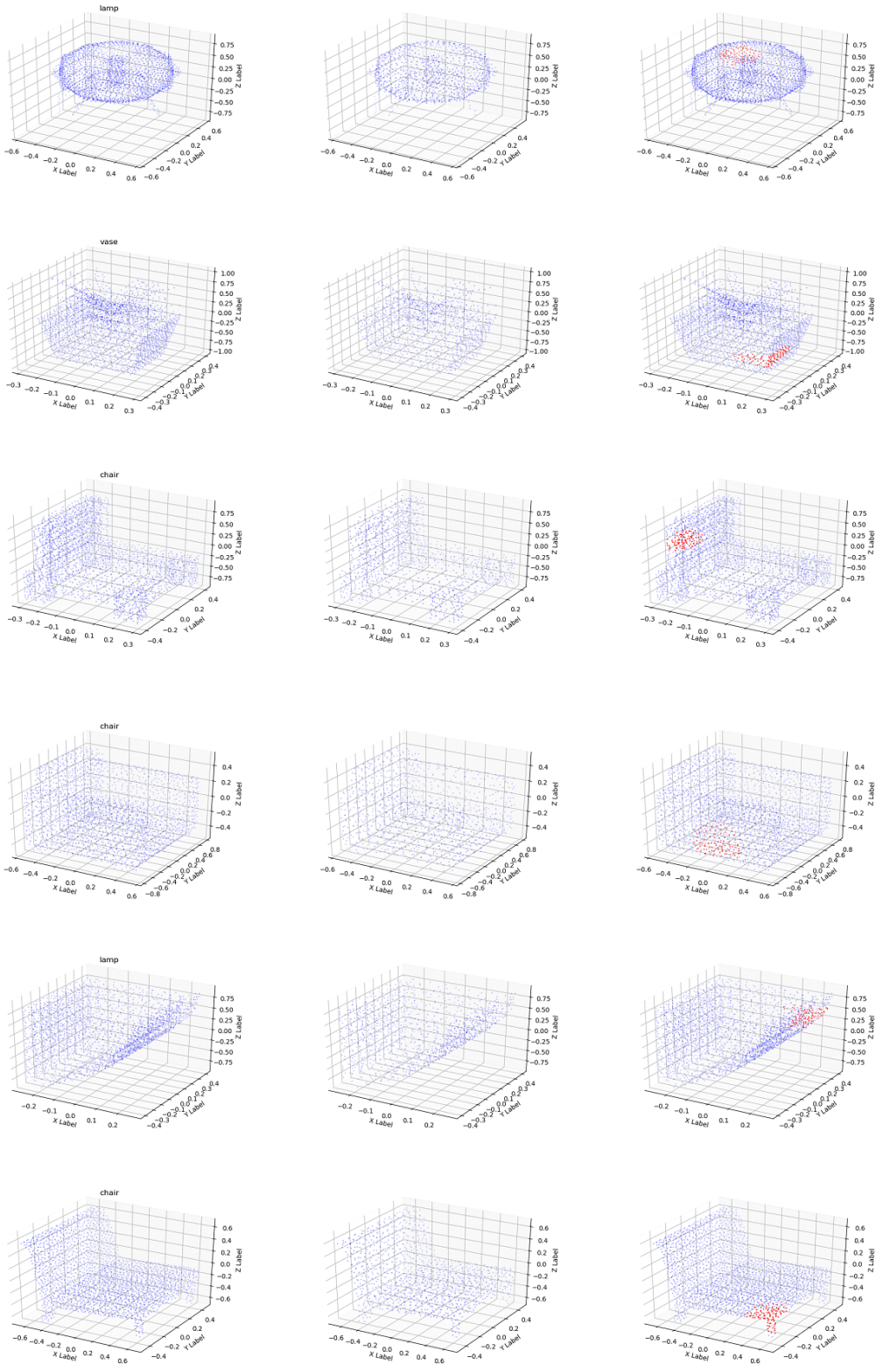

Fig. 8. Sample point clouds in the nth stage of the multistage hierarchy undergoing FPS and k-NN with k = 90 (from left to right): the point cloud before FPS, the point cloud after FPS, and k-NN where red indicates the nearest neighbors (cluster) of a selected point in the cloud (after FPS).