# Efficient Deep Learning Aspects for Classification of 3D Point Cloud Data

Nigam Katta
Georgia Institute of Technology
Atlanta, Georgia, USA
nkatta9@gatech.edu

Prasanth Bathala
Georgia Institute of Technology
Atlanta, Georgia, USA
pbathala3@gatech.edu

Janavi Khochare
Georgia Institute of Technology
Atlanta, Georgia, USA
janavikhochare@gatech.edu

Hemanth Tammana
Georgia Institute of Technology
Atlanta, Georgia, USA
htammana3@gatech.edu

## Abstract

*The aim of our project is to identify the aspects that make complex machine learning models more accurate and robust than basic models for real-time 3D point cloud classification. The paper discusses the potential applications of real-time 3D point cloud classification, such as autonomous driving, robotics, augmented reality, gaming, and virtual reality. The limitations of the basic model PointNet and the advantages of complex models such as DGCNN are also highlighted. The project reviews various aspects such as datasets, preprocessing techniques, model structure, and loss functions to evaluate the performance of the models. The results of the study could be useful for future researchers who aim to create new models with higher accuracy for 3D point classification*

## 1. Introduction/Background/Motivation

Real-time 3D point cloud classification has various potential uses in real-time applications such as autonomous driving, robotics, augmented reality, gaming, and virtual reality. In autonomous driving, real-time 3D point cloud classification is important for detecting and classifying objects on the road such as vehicles, pedestrians, and traffic signs. In robotics, it can be used for perception and interaction with the environment in real time. Real-time 3D point cloud classification can also create interactive augmented reality experiences, motion tracking, gesture recognition, and virtual object interaction in gaming and virtual reality applications. Overall, real-time 3D point cloud classification is an essential task for analyzing complex 3D scenes and structures in real-time, with practical uses in various applications.

Several machine learning models exist for 3D point cloud classification such as PointNet, a basic model that uses multi-layer perceptrons (MLPs) and max-pooling operations. It has some limitations, such as struggling with local features and missing out on the geometric relationships between points. On the other hand, complex models such as DGCNN use a dynamic graph CNN to capture local features and relationships between points, making it more robust and accurate. However, these complex models do not provide a proper justification or explanation of what aspects or additions to these models made them more robust and capable of achieving higher accuracy compared to the PointNet model. The aim of our project is to identify these aspects by checking the model performance before and after adding them to the model. It would be helpful for future researchers to make informed decisions while trying to create new models that provide higher accuracy for 3D point classification.

For this project, three different datasets were used i.e., ModelNet10, ModelNet40, and KITTI. The most important aspect of the dataset for this project is the distribution of categories, as it allows us to evaluate the performance of the models on a diverse range of objects. The code for this work is available at https://github.com/prasbathala/3D-Point-Cloud-Classification-Analysis

## 2. Related Works

In geometric data processing and analysis, tasks such as segmentation, classification, and matching require an understanding of local similarities between shapes. To establish this resemblance, feature descriptors representing local geometric structure are traditionally used. There are numerous publications in computer vision and graphics that propose local feature descriptors for point clouds, which are suitable for various problems and data structures.

Shape analysis has been an active area of research, and

in this regard, extrinsic and intrinsic descriptors have been widely used to represent 3D shapes. Extrinsic descriptors are usually derived from the shape's 3D coordinates and include shape context [2], spin images [12], integral features [19], distance-based descriptors [17], point feature histograms [30], and normal histograms [37], among others. Intrinsic descriptors, on the other hand, treat the shape as a manifold with a discretized mesh or graph and are invariant to isometric deformation. Spectral descriptors, such as global point signatures [29], heat, and wave kernel signatures [1], and their variants [5], are examples of intrinsic descriptors. Recently, some researchers have employed machine learning techniques to enhance the performance of standard descriptors, such as those proposed by [32].

While convolutional neural networks (CNNs) [14] have demonstrated remarkable success in the field of computer vision, adapting these methods to geometric data poses unique challenges due to the lack of an underlying grid structure. Researchers have therefore sought to develop new building blocks to replace the traditional convolution and pooling operations or to modify these operations to suit the gridless nature of geometric data. This review will provide an overview of the current state of research in this field, including the various approaches that have been proposed for deep learning on geometry.

Unlike images, geometric data lacks an underlying grid, which requires the development of new building blocks to replace convolution and pooling or adaptation to a grid structure. One approach to address this is to place geometric data onto a grid through view-based and volumetric representations, as demonstrated in [36], [40] or their combination in [26]. Recently, PointNet [24] has exemplified a class of deep learning architectures for non-Euclidean data on graphs and manifolds, termed geometric deep learning [4]. This approach dates back to early methods for constructing neural networks on graphs [31] and has been improved with gated recurrent units and neural message passing. Alternative definitions of non-Euclidean convolution employ spatial rather than spectral filters, as shown in [21] through the Geodesic CNN (GCNN), which is a deep CNN on meshes that generalizes the notion of patches using local intrinsic parameterization. Follow-up work has proposed different local charting techniques using anisotropic diffusion [3] or Gaussian mixture models [23]. In other works, such as [11], a differentiable functional map layer was incorporated into a geometric deep neural network, allowing for intrinsic structured prediction of correspondence between nonrigid shapes.

By embedding the shape in a domain with shift-invariant structure, such as the sphere [34], torus [20], plane [6], sparse network lattice [35], or spline [8], the final class of geometric deep learning methods tries to undo a convolution operation.

Additionally, geometric generative models are an important aspect of non-Euclidean geometric deep learning. These models aim to generalize autoencoders, VAEs, and GANs to non-Euclidean settings, where there is a lack of canonical order between the input and output vertices. This necessitates the need to solve an input-output correspondence problem. In 3D mesh generation, the mesh is typically assumed to be given, and the vertices are canonically ordered [13] proposed SurfaceNets based on the extrinsic Dirac operator for mesh generation. For point clouds, various generative architectures have been suggested by [7], [15], and [42]. Intrinsic VAEs have been used by [18] and [28] for shape completion and 3D face synthesis, respectively.

## 3. Dataset

To evaluate the SOTA methods on different architecture used for 3D Point Cloud Classification, we evaluate two datasets namely, ModelNet10, ModelNet40

### 3.1. ModelNet10

ModelNet10[41] is a subset of the ModelNet40 dataset and is often used as a benchmark dataset for 3D object recognition tasks, especially in cases where computational resources are limited. ModelNet10 contains 4899 CAD models from 10 object categories: bathtubs, beds, chairs, desks, dressers, monitors, nightstands, sofas, tables, and toilets. Like ModelNet40, each object is represented as a 3D point cloud or a mesh, and the dataset is split into training and testing sets. The training set contains 3991 models and the testing set contains 908 models.

### 3.2. ModelNet40

ModelNet40 [41] is a widely-used dataset in computer vision research for 3D object recognition and classification. It was introduced in 2015 and has become a benchmark dataset for evaluating 3D deep learning algorithms. The dataset consists of 12,311 CAD models from 40 object categories, such as chairs, tables, cars, airplanes, etc. Each object is represented as a 3D point cloud or a mesh, and the dataset is split into training and testing sets. The training set contains 9,843 models and the testing set contains 2,468 models. The point cloud representation of each object is generated by uniformly sampling points on its surface. The point clouds are normalized to a unit sphere and represented as a set of (x, y, z) coordinates and associated (r, g, b) color values.

## 4. Methods:

### 4.1. PointNet

Point clouds, which are collections of 3D points reflecting the shape of an item or a scene, are processed using

the well-known deep learning architecture PointNet [24]. There are three main components in the PointNet Architecture: Feature Extraction, Global Feature Aggregation, and Classification. For feature extraction, Multi-layer Perceptron (MLP) is used to learn from each point in the data. Two sets of MLP are used in the architecture, the first set will have local features representation and the second set will extract a dense representation of the global features. The global features are then aggregated with max pooling across all the points. The aggregated vector is then sent to a fully connected layer for the task of classification. The implementation of the Pointnet is taken from this github repository. https://github.com/charlesq34/pointnet

### 4.2. Dynamic Graph Convolution Neural Network (DGCNN):

The current method being employed is inspired by the "Dynamic Graph CNN for Learning on Point Clouds" model [38]. It combines elements from PointNet, Convolution and graph neural networks to utilize local geometric structures through the creation of a local neighborhood K-nn graphs to process the point cloud for classificaion. The method computes an edge convolution on the edges linking neighboring points with the core point. Unlike conventional graph convolution networks, the graph gets dynamically updated after every layer of the EdgeCNN. This enables the diffusion of non-local information throughout the point cloud, enhancing the model's ability to capture complex patterns and relationships. The implementation of DGCNN base was taken from https://github.com/WangYueFt/dgcnn.

#### 4.2.1 Edge Convolution:

Edge Convolution is a form of Convolution that performs a similar operation to Convolution on images. In the DGCNN model, a set of $n$ points in a point cloud space are chosen, where each point is represented by a directed graph with its nearest $K$ neighbors as nodes, and edges that connect each node to its neighbors, represented as an edge feature $E_{ij}$. The points are represented by their coordinates, which are used as features.

Let $X = \{x_1, x_2, ..., x_n\}$ represent the $n$ points, where each point has three features $(x, y, z)$ coordinates. A directed graph $G = (V, E)$ is created to capture the local structure of the point cloud, where $V = \{1, ..., n\}$ represents the vertices, and $E \subseteq V \times V$ represents the edges. A graph is created based on the k-nearest neighbor (k-NN) algorithm, where each node is connected to its k-nearest neighbors and a self-loop is included for each node.

The EdgeConv captures local geometric structure while maintaining permutation invariance. Edge features are computed based on a non-linear function $H(\theta)$, where $E_{ij} = H(x_i, x_j)$, and $H$ is parametrized by a set of learnable parameters. The output of EdgeConv at the i-th vertex is defined in (4). The EdgeConv applies a channel-wise symmetric aggregation operation on the edge features associated with all the edges emanating from each vertex, resulting in an aggregated feature vector for each vertex. The aggregation used in the above is max in (4). The performance and robustness of the model are dependent on the choice of $H$ and the aggregation operation.

In the current method, the employed H, edge feature extraction and final output of EdgeConv equations are:

$$H(x_i, x_j) = \overline{h}_\Theta(x_i, x_j - x_i) \tag{1}$$

$$e'_{ijm} = \text{ReLU}(\theta_m.(x_j - x_i) + \phi_m.x_i \tag{2}$$

$$x'_{im} = \max_{j:(i,j)}.e'_{ijm} \tag{3}$$

These equations can highly be effective in capturing both the global and local features of graph data. The equation 1 of $H(x_i, x_j)$, explicitly combines global shape structure with local neighborhood information. The equation 2 of $e'_{ijm}$, is a shared MLP that effectively extracts edge features. The equation 3, $x'_{im}$, captures the most salient features by taking the maximum over all edge features. Together, these equations make EdgeConv a highly effective technique for processing graph data.

## 5. Surveyed Approaches

This project aims to delve into the State-of-the-Art (SOTA) models for point cloud classification and identify the crucial factors that impact their performance. In particular, we seek to analyze the key ingredients for point cloud classification and understand what matters the most in achieving accurate results. To achieve this goal, we draw on the insights from a survey conducted by Ankit Goyal et al. in [9], which explores the impact of controlling factors independent of the network architecture on the point cloud classification task.

In this work, we aim to consider model-dependent and independent factors that would impact point cloud classification, including the choice of the dataset, preprocessing strategies, model structure, and loss functions. These factors play a critical role in determining the performance of point cloud classification models, and our analysis aims to shed light on the relative importance of each factor. We believe that this work will contribute to a better understanding of the key ingredients for achieving high performance in point cloud classification and help researchers and practitioners in developing better models for this task.

### 5.1. Dataset

The choice of the dataset is an important factor that can significantly impact the performance of point cloud classification models [10]. Some of the reasons are :

Table 1. Performance of various architectures on different variations (Numbers in Overall accuracy

| Dataset | | Data Augmentation | Model Selection | | Loss Function | | Architectural Changes | Architecture | |
|---|---|---|---|---|---|---|---|---|---|
| ModelNet10 | ModelNet40 | | Final | Best Test | CE | Smooth Loss | | PointNet | DGCNN |
| | ✓ | Rotation, Jitter, Shuffle | ✓ | | ✓ | | None | 49.18 | |
| | | | | | | | Addition of two conv1D layers | | |
| | ✓ | Rotation, Jitter, Shuffle | ✓ | | ✓ | | Remove Batch Normalization | 40.12 | |
| | ✓ | Jitter, Shuffle | ✓ | | ✓ | | Dropout rate to 0.5 | 49.56 | |
| | ✓ | Rotation, Jitter, Shuffle | | ✓ | | ✓ | removal of two conv2D layers | 49.78 | |
| ✓ | | Jitter, Shuffle | ✓ | | ✓ | | None | 81.23 | |
| ✓ | | Rotation, Shuffle | | ✓ | | ✓ | Dropout to 0.5 | 87.53 | |
| | | | | | | | Adding two convolutional layers | | |
| ✓ | | Jitter, Shuffle | | ✓ | | ✓ | Dropout to 0.3 | 88.32 | |
| | ✓ | Translate, Jitter | | ✓ | | ✓ | k = 10 points | | 82.75 |
| | ✓ | Shuffle, Jitter | ✓ | | ✓ | | k = 15 points | | 81.76 |
| ✓ | | Translate, Jitter | | ✓ | | ✓ | k = 10 points, add edge conv | | 93.53 |
| | ✓ | Translate, Jitter | | ✓ | | ✓ | k = 20 points | | 85.64 |
| ✓ | | Translate, Shuffle, Jitter | | ✓ | | ✓ | k = 10 points | | 94.53 |

1. **Generalization:** A good dataset for point cloud classification should be representative of real-world scenarios and contain a wide variety of object shapes, sizes, and orientations. The model should be able to generalize to new and unseen data, and this ability is highly dependent on the quality and diversity, the model might not be able to learn the necessary features to accurately classify the new data.

2. **Bias:** The Bias in the dataset can significantly improve the performance of the model, especially if the training data is not representative of the target population. For example, if the training dataset contains only a few specific object types or is biased towards certain viewpoints, the model may not be able to classify objects accurately from different viewpoints or of different types.

3. **Data Augmentation:** Augmenting the dataset by applying transformations such as rotation, scaling, and translation can increase the diversity of the training data and improve the model's performance. Therefore, the choice of the dataset should also consider whether it is amenable to data augmentation.

In this work, we will evaluate two datasets ModelNet10, and ModelNet40 [41]

## 5.2. Processing

**Input Points:** The number of input points is an important factor in point cloud classification. If the number of points is too low, then the model may not have enough information to accurately classify the object, and if it is too high, then the model may become computationally expensive and may overfit to the training data. It is required to select the optimal number of points. PointNet [24] uses a fixed 1024 points per object to train the network. This fixed point strategy simplifies the point cloud classification and helps reduce the computation time required to train the network. On the other side, DGCNN [38] uses a resampled points strategy during the training. Unlike using a fixed set of points, DGCNN randomly samples a subset of points from the input point cloud during each epoch. This approach exposes the model to more than 1024 points per object during the training process, which can improve the model's ability to handle varying point cloud densities and noise levels.

**Data Augmentation:** For the point cloud classification, there are various data augmentation strategies like jittering, random translation, random scaling, and random rotation along the y-axis. Different methods use different combinations of these augmentations. PointNet uses all the above mentioned augmentations. However, considering that the objects in ModelNet40 are aligned, random rotation along the y-axis will severely penalize the performance of the model. So, the DGCNN on the other hand will not employ random rotation. DGCNN only uses random translation and rotation.

**Loss Function:** The Cross-Entropy (CE) loss function is commonly used in point cloud classification tasks. However, DGCNN uses a modified version of CE called the smooth-loss function, where the ground truth labels are smoothed out before calculating the CE. This smoothing technique helps to prevent overfitting by reducing the impact of noisy or mislabeled data points. The use of smooth loss has been shown to improve the performance of all network architectures in point cloud classification tasks. This is because it provides a more robust way of training the model by reducing the sensitivity to individual data points. Smooth-loss is particularly useful in scenarios where the dataset contains noisy or mislabeled data.

**Selected Model for Testing:** Different methods employ different ways of selecting the final model for testing. PointNet uses the final converged model to evaluate the test set, and they create a validation set from the training set to tune the number of epochs. The model is retrained on complete training set to the tuned number of epochs. This strategy is called *fixed model selection*. On the other hand, DGCNN

evaluates the model on the test set after every epoch during the training and uses the best test performance as the final performance. This strategy is called *Best Model Selection*. Both strategies have their advantages and disadvantages. Final Model Selection is more computationally efficient since it requires training only the model once. However, it may result in suboptimal performance if the number of epochs is not tuned correctly. Best Test Model selection requires more computational resources since the model is evaluated on the test set after every epoch, but it ensures that the final performance is the best possible.

### 5.3. Model Structure

CNNs are particularly well-suited for processing 3D data such as point clouds [16] because they can learn hierarchical representations that capture spatial relationships and geometric structures in the data. By using a deep learning model for point cloud classification, we can leverage its ability to extract relevant features from the input data and produce accurate predictions. Different model structures and hyperparameters to optimize the performance of the model were tested such as adjusting the number of convolutional layers, the filter sizes, the activation functions, the learning rate, and the batch size to find the optimal configuration that produces the best results.

1. **Modifying layers:** The original DGCNN architecture for classification utilizes four EdgeConv layers with LeakyReLU activation and batch normalization. To investigate the effect of increasing depth and changing activation functions on performance, we modified the architecture by adding an additional EdgeConv layer and replacing LeakyReLU with ReLU activation. Similarly, PointNet was modified by adding and removing the Convolution layers from the model architecture. Our modifications aim to explore the impact of network depth and activation functions on achieving high accuracy in fewer epochs.

2. **Modifying Hyperparameters:** The last two fully-connected layers have a dropout rate of 0.5. The value of k, the number of nearest neighbors, is chosen through a validation set. The model is first trained on 80% of the training data while using the remaining 20% for validation to search for the optimal k value. The original PointNet model is implemented using a dropout rate

## 6. Experiments and Results

**Implementation Details:**

Our study involves the implementation of various models and their variations, and we have utilized PyTorch and Tensorflow frameworks for this purpose. Whenever possible, we have reused the official code for these frameworks. Additionally, we have employed the official version of DGCNN, and since PointNet is officially available in Tensorflow, we have used this framework for any modifications made to it in our study. In order to analyse the performance of the models, we have computed the Mean Accuracy and Overall accuracy for the classification task.

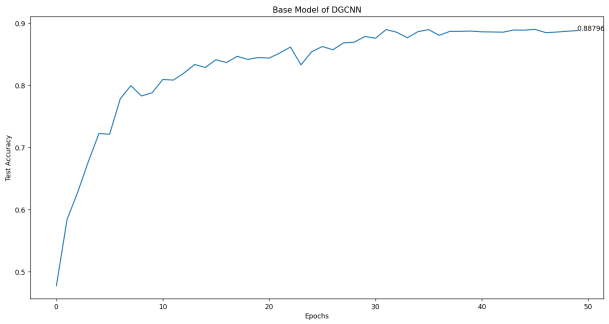


Figure 1. DGCNN Base Model

### 6.1. Architecture Depended Factors:

In this study, we aimed to evaluate the impact of architectural changes on the performance of our model. Specifically, we investigated the effect of different activation functions and network depths on the model's robustness. To this end, we conducted experiments where we varied the depth of the model while using either LeakyRelu or Relu activation functions. We plotted the accuracy vs epochs curve to analyze the behavior of the model for all the cases considered. All the experimental configurations underwent training for 50 epochs. The K-value (20), the Data Augmentations(Translatem Jitter) and Loss Functions(Smooth Loss) utilized were selected based on the results presented in table(1). To sample each test point cloud, we used 1024 points, and we employed SGD as the optimizer. The learning rate for SGD was initially set to 0.1 and gradually reduced using cosine annealing until it reached 0.001.

#### 6.1.1 Case 1: Base model

In this case, the original model from the DGCNN paper is considered. The plot of Accuracy vs Epochs can be seen in figure 1. From the table 2, the original model implemented achieved an accuracy of around 88.9% accuracy.

#### 6.1.2 Case 2: Base model with Relu Activation function

In this case, the original model with RELU as activation is considered instead of LeakyRelu from the DGCNN paper is considered. The plot of Accuracy vs Epochs can be seen in figure 2. When the ReLU activation function is introduced

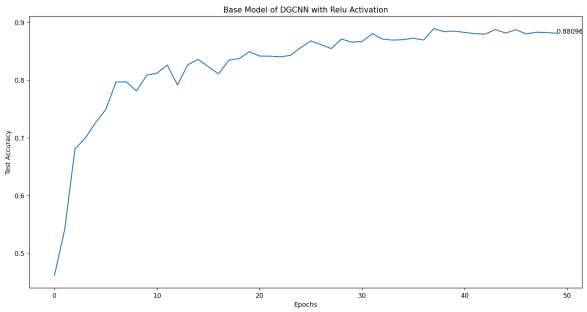


Figure 2. DGCNN with Relu Activation function

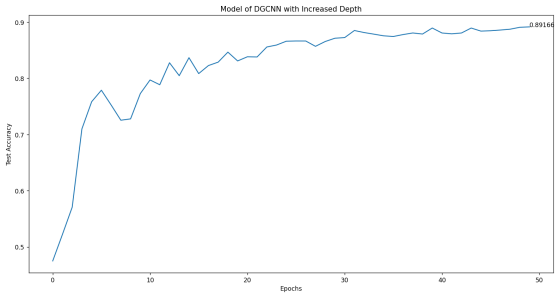


Figure 3. Modified DGCNN with increased depth

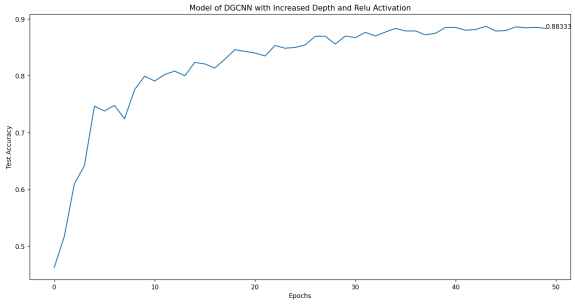


Figure 4. Modified DGCNN with increased depth and Relu Activation function

into the model, The accuracy was around 83.45%, where we can see not much improvement in the model.

#### 6.1.3 Case 3: Modified model with increased Depth

In this case, the modified DGCNN model where an extra EdgeConv is added is considered. The plot of Accuracy vs Epochs can be seen in Figure 3.

#### 6.1.4 Case 4: model with increased Depth and Relu activation function

In this case, the modified DGCNN model where an extra EdgeConv is added along with modification of LeakyRelu to Relu is considered. The plot of Accuracy vs Epochs can be seen in Figure 4. when the depth of the model was increased, we can see that there is a significant improvement in the model.

From the above results, it can be seen that Case 3 out of all cases gave the best accuracy when tested on ModelNet40 test data with 89.16 % being the highest Mean -accuracy. As expected the model with increased depth gave better results, due to its ability to further learn the global representation features better.

Also, we have analyzed the modified case 3 model with the PointNet model and other widely used models for the classification of point cloud data.

Table 2. **Classification results on ModelNet40**

| **Method** | **Accuracy** |
|---|---|
| 3DShapeNets [41] | 84.7 |
| VoxNet [22] | 85.9 |
| Subvolume [25] | 89.2 |
| ECC [33] | 87.4 |
| PointNet [24] | 89.2 |
| PointNet++ [27] | 90.7 |
| DGCNN Base Model [38] | 88.9 |
| DGCNN with Relu Activation Function | 90.7 |
| Modified DGCNN with increased depth | 92.13 |
| Modified DGCNN with increased depth and Relu Activation function | 91.9 |

From table 1 it is clear that the Smooth loss functions provide better performance compared to the cross entropy loss function which holds true with respect to our hypothesis that it is capable of handling noisy and mislabeled data. However, adding more convolution layers or removing the layers did not show any significant impact on the model performance based on the obtained results. Similarly, data augmentation techniques such as rotation, and jitter showed a minute improvement in the model performance but are not significant enough to compensate for the additional computation it adds to the model.

The findings from table 2 indicate that the model's accuracy showed a slight improvement compared to the baseline when the modifications were made. These results confirm our hypothesis that the performance and accuracy of models are significantly impacted by changes in the activation function and architecture.

## 7. Conclusion

In conclusion, this project aimed to investigate the impact of different factors on the performance of 3D point cloud classification models. Specifically, we focused on the effects of the dataset, preprocessing, model structure, and loss function on model accuracy. Our results demon-

strated that the choice of the dataset can have a significant impact on model performance, with larger and more diverse datasets generally leading to better accuracy. Preprocessing techniques, such as data normalization and augmentation, can also improve model performance by increasing the amount of training data and reducing the impact of noise. Additionally, model architecture and loss function were found to be not adding more value to the model in achieving high accuracy. Specifically, the use of 1D convolutional layers was tested. Overall, our findings suggest that a combination of these factors is necessary for achieving high accuracy in 3D point cloud classification, and further research is needed to identify optimal combinations.

## 8. Future Works

This study can be extended by analyzing more aspects and different techniques under each of these aspects. It can be achieved by analyzing more complex models such as the RS-CNN, 3D2SeqViews, and MLVCNN which are proven to be having higher accuracy for 3D point cloud classification. While the current report only focussed on CNN architecture, this work can be extended to identify the impact of other architectures on model performance.

## References


[1] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 1626–1633. IEEE, 2011. 2

[2] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context: A new descriptor for shape matching and object recognition. *Advances in neural information processing systems*, 13, 2000. 2

[3] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. *Advances in neural information processing systems*, 29, 2016. 2

[4] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 2

[5] Michael M Bronstein and Iasonas Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 1704–1711. IEEE, 2010. 2

[6] Danielle Ezuz, Justin Solomon, Vladimir G Kim, and Mirela Ben-Chen. Gwcnn: A metric alignment layer for deep shape analysis. In *Computer Graphics Forum*, volume 36, pages 49–57. Wiley Online Library, 2017. 2

[7] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 2

[8] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877, 2018. 2

[9] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline, 2021. 3

[10] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey, 2020. 3

[11] Oshri Halimi, Or Litany, Emanuele Rodola, Alex Bronstein, and Ron Kimmel. Self-supervised learning of dense shape correspondence. *arXiv preprint arXiv:1812.02415*, 2018. 2

[12] Andrew E Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999. 2

[13] Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Joan Bruna. Surface networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2540–2548, 2018. 2

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 2

[15] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018. 2

[16] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on $\mathcal{X}$-transformed points, 2018. 5

[17] Haibin Ling and David W Jacobs. Shape classification using the inner-distance. *IEEE transactions on pattern analysis and machine intelligence*, 29(2):286–299, 2007. 2

[18] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1886–1895, 2018. 2

[19] Siddharth Manay, Daniel Cremers, Byung-Woo Hong, Anthony J Yezzi, and Stefano Soatto. Integral invariants for shape matching. *IEEE Transactions on pattern analysis and machine intelligence*, 28(10):1602–1618, 2006. 2

[20] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.*, 36(4):71–1, 2017. 2

[21] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. 2

[22] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE, 2015. 6

[23] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017. 2

[24] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 2, 3, 4, 6, 10

[25] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 6

[26] Charles R Qi, Hao Su, Matthias Nießner Angela Dai Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data supplementary material. 2

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 6

[28] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European conference on computer vision (ECCV)*, pages 704–720, 2018. 2

[29] Raif M Rustamov et al. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Symposium on geometry processing*, volume 257, pages 225–233, 2007. 2

[30] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009. 2

[31] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. 2

[32] Syed Afaq Ali Shah, Mohammed Bennamoun, Farid Boussaid, and Amar A El-Sallam. 3d-div: A novel local surface descriptor for feature matching and pairwise range image registration. In *2013 IEEE International Conference on Image Processing*, pages 2934–2938. IEEE, 2013. 2

[33] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017. 6

[34] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, pages 223–240. Springer, 2016. 2

[35] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2530–2539, 2018. 2

[36] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 2

[37] Federico Tombari, Samuele Salti, and Luigi Di Stefano. A combined texture-shape descriptor for enhanced 3d feature matching. In *2011 18th IEEE international conference on image processing*, pages 809–812. IEEE, 2011. 2

[38] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019. 3, 4, 6

[39] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 10, 11

[40] Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1544–1553, 2016. 2

[41] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes, 2015. 2, 4, 6, 10

[42] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215, 2018. 2

# 9. Appendix

## 9.1. Contribution Table

## 9.2. PointNet Architecture

## 9.3. DGCNN Architecture

| Student Name | Contributed Aspects | Details |
|---|---|---|
| Nigam Katta | ModelNet40 dataset [41] and DGCNN [39] | Analyzing the dataset, conducting a literature review, fine-tuning hyperparameters, designing and implementing the DGCNN architecture, and running multiple experiments to evaluate its effectiveness. Contributed to report writing. |
| Prasanth Bathala | ModelNet10 dataset [41] and DGCNN [39] | Investigating the dataset, performing a survey of relevant literature, optimizing hyperparameters, designing and implementing the DGCNN architecture, and conducting a series of experiments to evaluate its performance on 3D object classification.Assisted in the writing of the report. |
| Janavi Khochare | ModelNet40 [41] and PointNet [24] | Exploring the dataset, conducting a literature review, tuning hyperparameters, comprehending and implementing the PointNet architecture, and performing various experiments to determine the significant preprocessing and activation functions that enhance the accuracy of 3D object detection. Contributed to report writing |
| Hemanth Tammana | ModelNet10 [41] and PointNet [24] | Analyzing the dataset, reviewing relevant literature, fine-tuning hyperparameters, grasping the PointNet architecture, and executing multiple experiments to identify the crucial preprocessing and activation functions that contribute to improving the precision of 3D object detection. Assisted in the writing of the report. |

Table 3. contributions of team members.

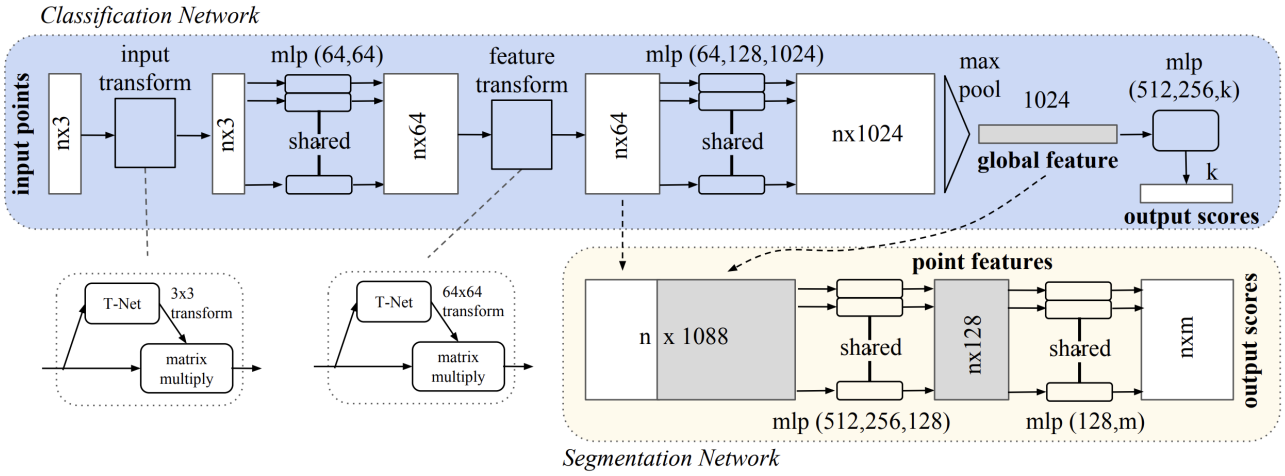


Figure 5. PointNet Architecture [24]

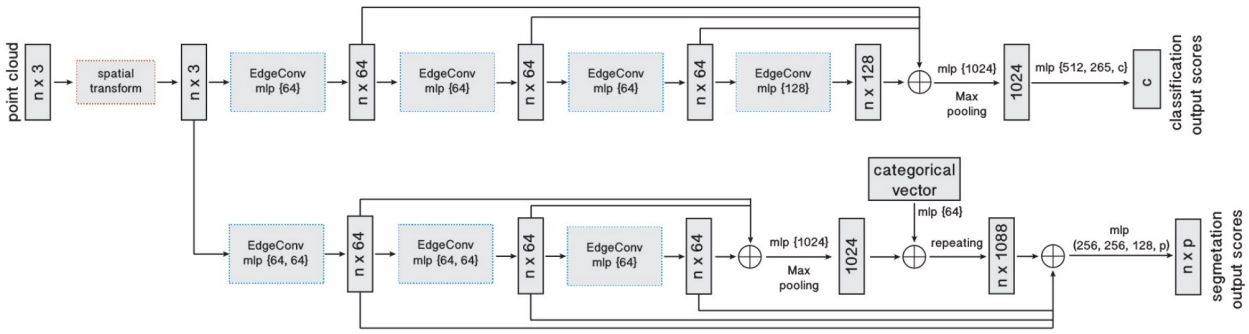


Figure 6. DGCNN Architecture [39]