An Efficient Neural Network Architecture and Training Protocol for 3D Point Cloud Classification

Sneha Paul

A Thesis

in

The Department

of

Concordia Institute of Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

January 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Sneha Paul**

Entitled: **An Efficient Neural Network Architecture and Training Protocol for 3D Point Cloud Classification**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

________________ Chair and Examiner
Dr. Abdessamad Ben Hamza

________________ Examiner
Dr. Arash Mohammadi

________________ Supervisor
Dr. Zachary Patterson

________________ Co-supervisor
Dr. Nizar Bouguila

Approved by ________________
Abdessamad Ben Hamza, Chair
Department of Concordia Institute of Information Systems Engineering

________________ 2023

____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

An Efficient Neural Network Architecture and Training Protocol for 3D Point Cloud Classification

Sneha Paul

The point cloud is a set of data points in a 3D coordinate system with an irregular data format. As a result, they are needed to be transformed into a collection of images before being fed into models. This unnecessarily increases the volume of the data and increases complexities. The existing literature on point cloud uses a fixed number of points sampled from the whole point cloud as the input. However, with large point cloud data, it is important to consider more points as input to have a better understanding of the scene. The computational expense increases if the input number of points increases for existing networks. Our research contributes to the existing point cloud classification literature in two directions. First, we develop a training protocol for improved point cloud training accuracy on top of the existing PointNet [31] architecture over the ModelNet10 dataset. A few variations of encoder models have been proposed in this regard. Also, an extensive hyperparameter study and ablation study are done. These experiments achieve a 6.10% improvement over the baseline model. After that, we propose DualNet, a novel 3D point cloud network that resolves the trade-off between the number of input points and the computational expense of 3D data. The DualNet consists of two branches: DensetNet and SparseNet. The SparseNet is a comparatively large network in terms of number of parameters, that samples a small number of points from the whole point cloud. Whereas the DenseNet is a lightweight network that takes a large number of points as input. SparseNet is composed of more number of channels than DenseNet making it more computationally expensive than DenseNet. While the accuracy of the model shows good improvement when the number of points increases, the overall computational cost of DenseNet does not increase much in such settings. DualNet shows 0.81% and 0.45% increase in the SOTA results on

ModelNet40 and ScanObjectNN respectively. In respect of computational complexity, our model takes about 40% less time compared to SOTA.

Acknowledgments

First I would like to express my gratitude to my supervisors Dr. Zachary Patterson and Dr. Nizar Bouguila for providing me with the opportunity to complete my masters degree here at Concordia University. I am thankful for their support and contributions. Their knowledge, patience and guidance in my research work were key to my success.

I would like to thank the members of my defense committee, Dr. Arash Mohammadi and Dr. Abdessamad Ben Hamza for their valuable time and insightful feedback on this thesis. I am also thankful to my department, Concordia Institute of Information Systems Engineering (CIISE) and Mitacs for providing me with financial assistance throughout the completion of my masters degree.

Finally, I would like to thank my husband for his overall support towards my masters journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Point cloud, a set of points in the 3D geometric coordinate system with irregular data structure, is one of the central research areas of 3D Computer Vision [31]. It is commonly collected by sensors e.g. camera, LiDAR or radar scan, which perceives the 360-view information of the surrounding environment. Unlike 2D datasets that are composed of pixels of regular data format, point cloud data is a set of unordered points $P \in \mathbb{R}^{n\times 3}$. Each point in a point cloud data contains x, y, and z coordinates and sometimes some additional information such as surface normal and intensity. The nature of point cloud data prohibits it from being treated like a regular image to which image processing techniques, e.g. CNNs, can be applied directly. Furthermore, the analysis of point cloud data becomes even more difficult due to the sparsity of the dataset and the presence of noise (e.g. sensor error) [26].

The structure of point cloud data follows 3 unique properties in point sets $\mathbb{R}^n$. These three properties are summarized as follows:

- Unordered structure: Compared with 2D images or 3D voxels where the data structure is regular, point cloud is composed of a set of points of irregular data structure. As a result, if a model takes input N number of 3D point sets, it should be invariant of $N!$ permutation of data feeding order.

- Relation among the points: In point cloud, points are related to the neighbouring points. As a result, it is necessary to capture the local structure of the nearby points to understand the underlying information and learn the correct representation of the input point cloud.

- Invariant of transformation: Since point cloud is geometric data, certain transformations should not change its underlying geometric characteristics. For example, applying rotation and translation of points should not change its semantic information.

Since point cloud is generated from 3D scans, it contains detailed information about the physical environment, e.g. surface roughness of an object. It is primarily used for creating 3D models. Some major application of point cloud data includes self-driving vehicles and self-navigating robots. Some other industrial-level real-world use of point cloud includes creating 3D models of manufactured products for quality inspection, generating digital twin for analyzing and simulating any real-world scenario, meteorological analysis, visualization, rendering and animation of various custom application. In recent years, the use of neural networks in 3D point cloud analysis has shown promising performance and significantly improved different real-world point cloud applications. Success in point cloud applications is the result of various computer vision tasks, including 3D shape classification [**?**], 3D object detection [44] and segmentation [6] tasks, point registration [28], shape completion and so on.

3D object classification is one of the fundamental and challenging tasks of computer vision which involves a computer's understanding of a scene that can be generalized to many other related tasks, including object detection and segmentation. This makes 3D object classification one of the benchmark tasks for understanding the learning capabilities of 3D models and training recipes. As a result, new methods are often tested on some well-known object datasets on the object classification task.

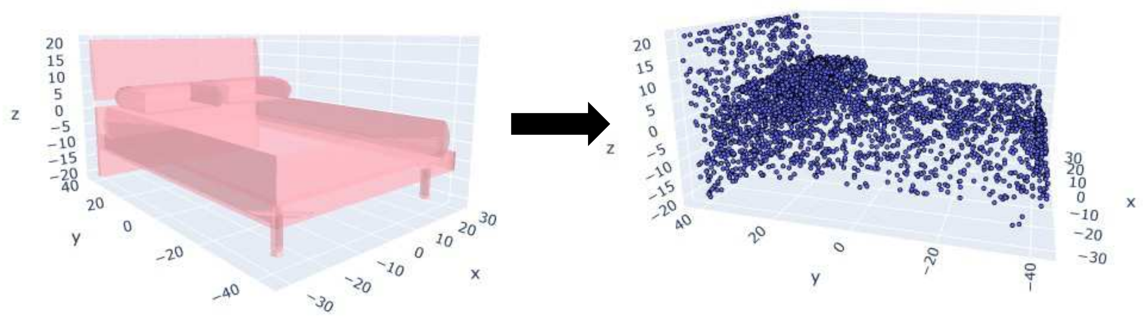


Figure 1.1: Example of 3D point cloud .

1.2 Research Motivation

Due to the special properties of point cloud data which make them different from 2D images, regular image processing techniques cannot be applied to point cloud data. Since convolution networks require a pre-defined structured data format, point cloud data or meshes must be transformed into regular 3D voxel grids or collections of images. This transformation makes the data voluminous and reduces the natural variance of data. Feeding the data directly to the model reduces the issues arising from data transformation.

PointNet [31] is one of the pioneering works in point cloud data analysis, which directly feeds point cloud data as input to the network. The encoder of PointNet [31] learns to select the most informative points from point cloud data and make predictions using them. However, one major limitation of the PointNet architecture is that it does not consider local structures in the metric space. This is an essential requirement for implementing convolutional neural network (CNN) based architectures that have shown tremendous success in image applications. CNNs expect input data to be defined on regular grid. PointNet++ [33] solves this problem by applying PointNet [31] recursively on a nested partition of point cloud data. Like CNNs, they capture the local patterns and finally group them to learn higher-level features.

Some other recent work on point cloud data analysis has also focused on learning local geometric patterns using local extractors such as Graphs [17] and attention mechanisms [11]. This class of local extractors has two major limitations. First, the local geometric extractors make models computationally expensive, compromising the inference latency of the model.

Second, the performance gain of such models has been saturated for the existing benchmark datasets [18]. To deal with the aforementioned limitations associated with existing feature extractors and further improve performance, PointMLP proposed a deep hierarchical residual MLP architecture [26]. Because of the MLP architecture, PointMLP is invariant to input permutation, which respects one of the major characteristics of point cloud data. Moreover, using residual MLPs facilitates the building of very deep networks. Since this model is free from local geometric extractors, repeating the feed-forward MLPs several times does not significantly increase the computational cost.

Even though PointMLP shows great performances, there are still limitations. First, it takes a fixed pre-defined number of points as input. Hence, PointMLP (as well as other existing methods) does not enjoy the advantages of more input points. Also, increasing the number of input points linearly increases the computational cost, as well as memory requirements, which makes it impossible to utilize high-density point cloud data.

Therefore the research gap in the existing literature on point cloud data classification can be divided into two stages:

- Developing an improved training protocol for the existing point cloud classification architectures, e.g. PointNet [31].
- Developing a model that can handle the increased number of points and learn rich features without a linear increase in complexity and memory demand.

1.3 Research Objective

Though deep learning is showing promising performance in 3D point cloud data analysis, it lacks a lightweight and computationally friendly network which can take benefit from the large input point cloud data. Several existing models are lightweight, e.g. PointNet [31], PointMLP [26] with fewer parameters showing better performance with fewer input points, for example, 1k. This thesis aims to meet the research gaps in existing point cloud data classification literature stated in section 1.2. The main objectives of this research can be stated as follows:

- To investigate the scope of further improvement of the existing SOTA point cloud data classification models' performance through empirical analysis of important training parameters and hyper-parameters.
- To build a lightweight point cloud model with less computational complexity while taking advantage of large point cloud data input. In other words, the objective is to increase the performance without increasing the computational cost.

To achieve the above-mentioned objectives, we propose a training protocol for improved point cloud data classification accuracy. We also propose a novel neural network which is less memory-demanding and captures the benefit of large input points. To measure the achievement of the objectives, we take the SOTA models as baseline and compare the performance of our proposed methods with them. Achieving higher accuracy than the SOTA can be stated as fulfilling the objectives.

1.4 Thesis Organization

The thesis is built on 5 chapters. Chapter 1 explains the background of this research work, the definition and properties of point cloud data, research motivation and research objectives. Chapter 2 describes the existing literature on point cloud data classification. The existing literature is divided into 3 main categories to highlight each of their contributions and limitations; (1) point cloud data with transformation techniques, (2) point cloud data without transformation, (3) point cloud data with local extractors. Also, three point cloud benchmark datasets are described, which are used for proposed methods in later chapters. To fulfill the research gaps indicated in section 1.2 and meet the research objectives stated in section 1.3, we propose two methods for point cloud data classification in chapter 3 and chapter 4. In chapter 3 we propose a training protocol for improving the classification accuracy of existing PointNet [31] architecture over the ModelNet10 dataset. Here, an intensive experiment is done on different model variants, augmentation, loss function and so on. In chapter 4, we propose a novel neural network architecture for point cloud data classification. To show the effectiveness of the proposed model, a detailed experimental study is done on two point cloud benchmark datasets; ModelNet40 and ScanObjectNN. Finally, our closing remarks, limitation and future direction of this research are given in chapter 5.

Chapter 2

Literature Review

2.1 Introduction

Point Clouds are 3D geometric data with x, y, and z positional information for each point. By computing the normals of local or global features, more dimensions can be added to this data. In the past, task-specific features extracted from Point Clouds were mostly handcrafted and encoded with various statistical properties of the points [31]. They were invariant of certain transformations and could be classified into two categories; intrinsic [1] and extrinsic [35]. We were also categorize them into local features and global features. Back then, finding the best feature combination was not necessary for a specific task.

Due to the unordered data structure of point cloud, general image analysis techniques can not be applied to them. As a result, transforming the unstructured point cloud into structured dataset (e.g. 2D image, 3D voxel) has taken a large area of literature in point cloud analysis. But this approach comes with the limitation of increased volume of data, and computational complexity. To solve these challenges imposed by point cloud transformation, researchers developed approaches to feed the point cloud dataset directly, without any transformation to the network. PointNet [31] is one of the pioneering works in this aspect. Nowadays, various local extractors (e.g. CNNs, GNNs, attention mechanisms) are showing promising performance in point cloud analysis.

In this chapter, we discuss various existing methods related to Point Cloud classification. The discussion is divided into three subsections; Point Cloud with transformation techniques (section

2.2), Point Cloud without transformations (section 2.3), and Point Cloud with sophisticated geometric extractors (section 2.4). The research gap indicated in section 1.2 and the discussion made on existing point cloud classification techniques has motivated the proposed methods in chapter 3 and chapter 4. Finally, a description of the popular point cloud benchmark datasets are discussed in section 2.5 which are used in the proposed method for point cloud classification in chapter 3 and chapter 4.

2.2 Point Cloud with Transformation Techniques

Unlike sequence, image, and volume data, 3D point clouds are an unordered set of vectors. As a result, common operations for ordered data structures cannot be used on 3D point clouds. To deal with this problem, an attention mechanism with a read-process-write network was proposed that consumes the unordered point clouds and outputs an ordered vector [40]. But this approach could not store the geometric information captured by 3D point clouds. The deep learning methods used for point cloud transformation and analysis were categorized into 3 ways; Multiview CNNs, Spectral CNNs and Feature-based DNNs [31]. Each of them is described in the following sections.

2.2.1 Feature-based DNNs

This approach converted the Point Cloud into vectors and applied feature-based DNNs to extract traditional shape features and classify them [4]. Guo *et. al* proposed a 3D mesh labelling method to adapt with different 3D meshes using feature-based DNNs [10]. They non-linearly combined and hierarchically compressed the low-level features into a compact representation for the 3D mesh. Finally, a label vector was assigned by DNNs indicating their class probabilities. Fang *et. al* developed a deep shape descriptor for 3D object classification for various domains, e.g. engineering, medicine, biology and so on [4]. Despite various applications of this approach, it was constrained by the representation capability of the extracted features.

2.2.2 Multiview CNNs

The main concept behind Multiview CNN [37] was to render the 3D point cloud into 2D images and then apply 2D CNNs on the transformed 2D point cloud [36]. Due to highly improved CNNs, this method showed promising performance in shape classification and retrieval tasks. Though this approach had limitations for different 3D tasks, such as point classification and shape completion, it was advantageous for 3D shape classification tasks. RotationNet [15], a CNN-based approach, uses multi-view images from an object to predict its class and pose jointly. It extracts view-specific feature representations, essential for accurate object classification and pose estimation.

2.2.3 Volumetric CNNs

Due to the irregular format of Point Cloud, many researchers transformed them into regular 3D voxelized shapes and applied 3D convolutional neural networks (CNNs) to get the local information [32]. Wu *et. al* proposed 3D ShapeNets that learned complex 3D data distribution and their poses and found out hierarchical part representation, which showed promising performance for various tasks, such as shape completion from 2.5D depth maps, object recognition and so on [47]. VoxNet [27] used a volumetric Occupancy Grid representation with 3D CNN for 3D object recognition. Qi *et. al* proposed two 3D CNN-based architectures for Volumetric CNNs and Multi-view CNNs to improve the performance. For multi-view CNNs, they used 3D multi-resolution filtering [32]. However, this approach was costly due to the high volume of data and suffered from the curse of data sparsity. With higher dimensions, the number of data appearing in a dimension decreases by increasing the number of statistical analyses in an exponential order. As a result, many approaches, such as Vote3D [42], and FPNN [20] were proposed to deal with data sparsity but were not beneficial due to the high volume of points in Point Cloud.

2.3 Point Cloud without Transformation Techniques

One of the possible solutions to deal with the issues arising from transforming Point Clouds into 3D voxels or 2D images is to input Point Clouds directly into the model without any transformation. PointNet [31] is a pioneering work in this direction. At the primary stages, the points were processed independently and identically to keep the model architecture simple. Maxpool was used to learn a set of optimization functions for selecting informative points from the point cloud and encoded the reasons for selecting them. Finally, a fully connected layer was used in the final layer to aggregate all the information. To improve model performance, a rigid transformation network was applied to the input data to canonicalize them before feeding them into PointNet. Paul *et. al* [29] proposed an improved training setup of the existing PointNet model. Despite being a lightweight model, PointNet could not capture the hierarchical features at the local level.

PointNet++ [33] improved the PointNet [31] network. It used the K-nearest Neighbours algorithm to extract the overlapping points of the neighbourhoods and applied the PointNet network recursively to them. This process was continued until all the input points were processed. Here, the PointNet model was used like convolutional networks. Thus it successfully captured the local features.

Several works were developed upon PointNet [31] and PointNet++[33]. PointMLP [26] was one of them. It used the K-nearest Neighbours algorithm of PointNet++ [33] to extract the most useful points and apply residual MLPs to them. A lightweight geometric affine module was also used on the extracted points. Using only ResNet blocks kept the model lightweight and increased the model's inference speed. Wang et al. [41] proposed an MMI module which maximized the global and local representation by deploying the Jensen-Shannon mutual information (MI) estimator in both shallow and deep layers of the network. This novel network was combined with other existing models (e.g. PointNet, PointNet++, etc. of various tasks (e.g. classification, segmentation, object detection, etc.) to improve the model's accuracy.

2.4 Point Cloud with Local Extractors

The recent works on Point Cloud were concentrated on efficiently generating regional point extractors. For exploring the local feature extractors in a more precise way, we grouped the existing works into 3 families of approaches; convolution, graphs, and attention-based methods. PointConv [46] was a famous work among convolution-based methods. It used the MLP to approximate the continuous weights and density functions of convolutional filters and modified the filters dynamically for a new convolutional operation. DenX-Conv [16] used two connectivity techniques, neighbour connectivity and dense connectivity, to improve point cloud classification accuracy. For neighbour connectivity, CNNs extracted useful local geometric features by determining geometric hierarchy-based neighbourhood relationships. Dense connectivity was used to aggregate multiple layers. DenX-Conv expanded the PointCNN [19] into a deep neural network to reduce its limitation of vanishing gradient and learned stable feature learning in the deep layers.

Graph-based methods analyze the mutual co-relationship between different points in a graph. EdgeConv [45] was used for constructing a local graph by generating the relationship between a point with its neighbours by using the edge features. Wang et al. [43] proposed a graph-based point cloud classification architecture with spherical coordinate system. Using a spherical coordinate system rather than the traditional Cartesian coordinate system simplified the computation and representation of the point cloud. A pose-estimating auxiliary network was used to estimate the pose changes concerning rotation angles. To classify point cloud for the pose variation, Graph Convolutional Network (GCN) was used.

3D-GCN [22] used 3D Graph Convolution Networks for deriving 3D deformable kernels. The Attention-based methods were closely related to Graph-based methods and show promising performance in local feature extraction. For example, Guo *et al.* proposed PCT, a Transform network with farthest point sampling using nearest neighbour for better capturing local features [11]. Point Transformer [52] used self-attention layers and constructed a self-attention network for Point Cloud. Zhao et al. [51] proposed LGR-Net, a two-brunch network of MLP-based attention modules which fused the global topology rotation invariant (RI) features with local geometry RI feature to get the benefit from the two complementary RI responses.

Despite the success of these local extractors for extracting local features, the model's performance has been saturated and minimal improvement can be done to models by working on them. Moreover, these local extractors are heavy enough to demand high computational memory. In response to this situation, Ma *et al.* developed a simple and lightweight MLP-based network that used deep residual MLPs for Point Cloud analysis [26].

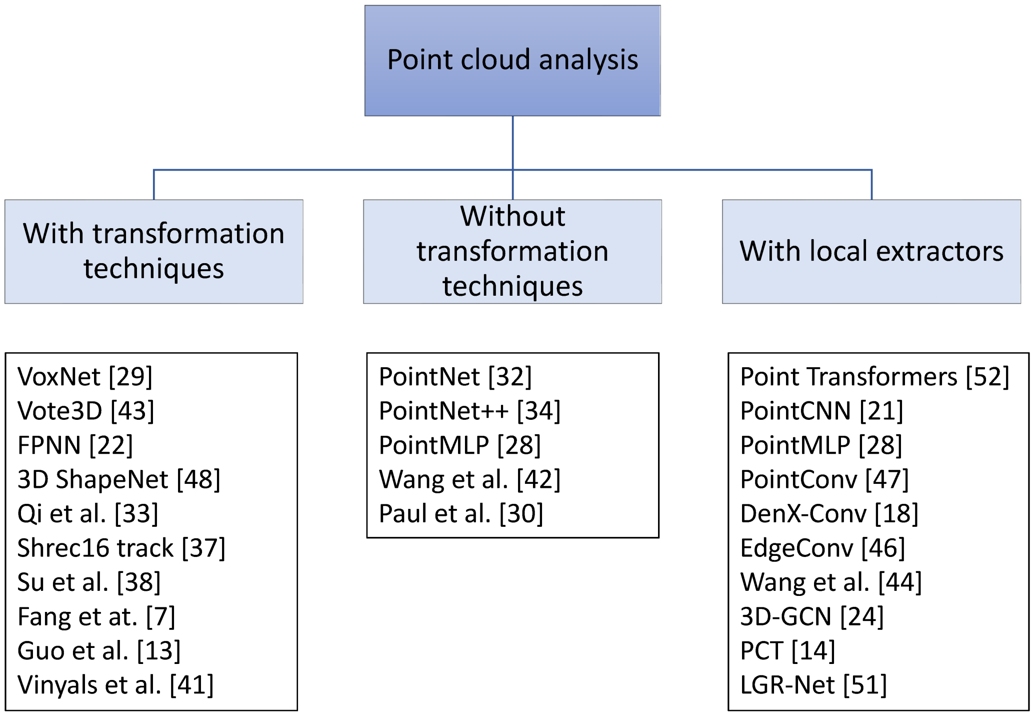


Figure 2.1: The summary framework of literature review

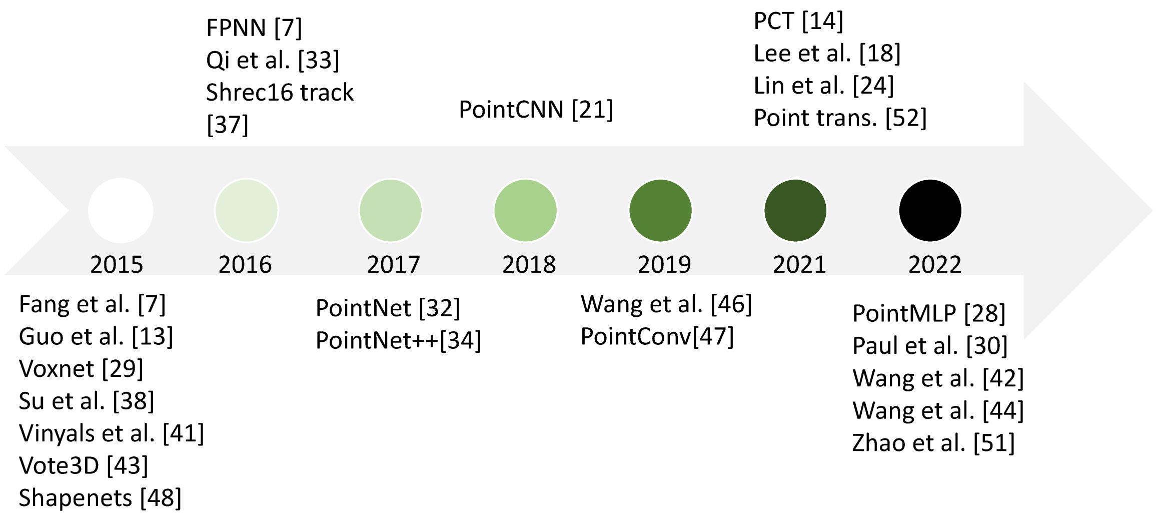


Figure 2.2: The summary timeline of literature review

2.5 Datasets Description

ModelNet10

ModelNet [47] is a popular 3D point could classification dataset. It has two versions with 40 and 10 classes respectively. ModelNet10 includes 10 categories of 3991 and 908 samples for training and testing partitions. The data is sparse coordinates of points, where each point $x \in \mathcal{R}^3$. Here, the 3 dimensions are the x, y, and z coordinates of a point. Each sample in the dataset contains N points, so the overall dimension of 3D data is $N \times 3$, where N is different for each sample. However, the input to the model needs to be the same size for each sample. So, we randomly sample n points from the set of N. So, the final input to the model is a vector $x \in \mathcal{R}^{n \times 3}$.

Since the dataset is already split into train and test data, no further splitting is required. The data needs to be normalized before giving this as input. No other pre-processing is required. The data is sparse, and no missing data problem exists. The output of the model is the probability distribution over the number of classes, in this case, 10.

ModelNet40

ModelNet40 [47] is a popular benchmark dataset for point cloud analysis. It is the larger version of the ModelNet dataset where 12,311 cad-generated sample objects belong to 40 different categories (e.g. airplane, car, lamp, etc.). The dataset is divided into 9,843 training samples and 2,468 testing samples. Like ModelNet10, each point in point cloud contains 3 dimensions; x,y and z. And each sample of point cloud contains N points. So the dimension of the dataset is $N \times 3$. Here, N varies for each sample. To keep the input size same for each sample, n points are randomly sampled from the set of N before feeding into the model.

ModelNet40 is a well-constructed dataset with no missing data and clean shapes. As a result, no further data pre-processing is required except normalization.

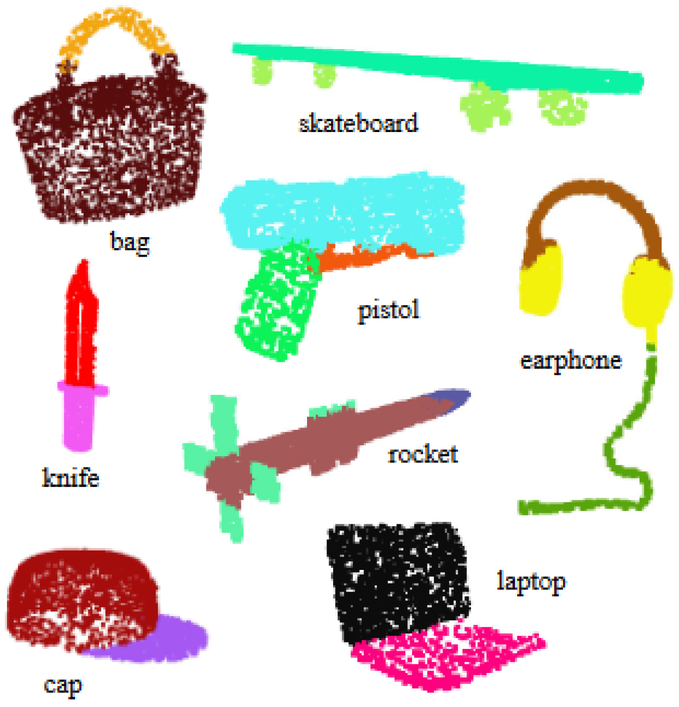


Figure 2.3: Example of 3D point cloud data from ModelNet10. The figure is collected from [47].

ScanObjectNN

ScanObjectNN [39] is a real-world, benchmark dataset of 3D point cloud analysis. It contains 15,000 real-world scanned indoor objects. The objects belong to 15 different categories and 2902 unique object instances. Each raw object contains N points and each point has global and local coordinates, colour attributes, normals and semantic labels. So, the overall dimension of 3D data is $N \times 9$.

Because of the cluttered background and occlusion of the real-world scanned samples, the dataset is challenging for existing classification techniques.

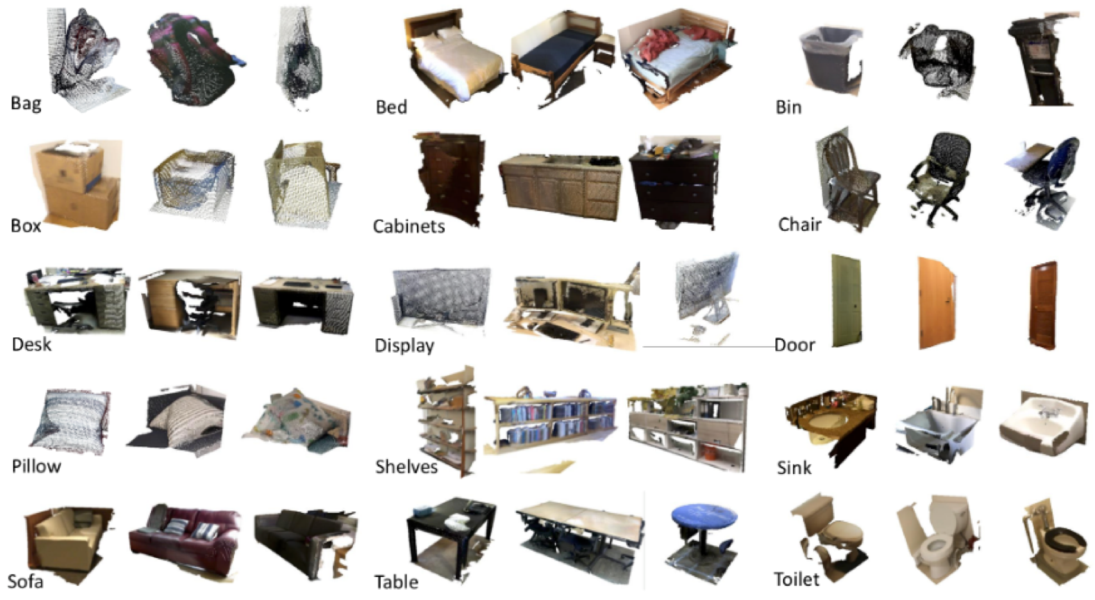


Figure 2.4: Example of 3D point cloud data from ScanObjectNN. The figure is collected from [39].

2.6 Summary

In this chapter, we investigated the existing literature on point cloud analysis from three different aspects and highlighted their limitations and existing research gaps in the arena of point cloud classification. Finally, three point cloud benchmark datasets are described which are going to be used in the proposed methods in chapter 3 and chapter 4. The findings in this chapter can be summarized as follow:

- Previously, the unstructured point cloud dataset was used to be transformed into structured dataset (e.g. 3D voxels, 2D images) to perform common operations for ordered data structures.
- Point cloud analysis with transformed dataset suffered from various limitations (e.g. increasing the volume of the data) making the computational process complex and costly.
- Feeding point cloud directly to the model, without any transformation, decreases such limitations.

- PointNet [31] is one of the pioneering works for point cloud analysis without any transformation.
- Nowadays various local extractors (e.g. CNNs, GNNs, attention mechanisms) are used showing promising performance.
- Despite the promising performance of these local extractors, their performance has been saturated leaving minimal room for improvement.
- Also, these local extractors are memory-demanding.
- Developing a simple and lightweight MLP-based network would be beneficial from the computational point of view and its generic network structure [26].

Chapter 3

Improved Training for 3D point cloud data Classification

3.1 Introduction

As we have stated before, PointNet is a pioneering work in point cloud data analysis because it feeds the point cloud data directly into the model without any transformation. As a result, it avoids the complex and costly data transformation task, which increases the volume of the data. In this chapter, we have worked on top of the existing PointNet [31] architecture, shown in figure 3.1. This research aims to explore the possibility of further improvement in the prediction accuracy of the existing PointNet architecture. Due to its simple and generic network architecture, the PointNet model is less memory-demanding with generalization capability.

The prediction accuracy of the existing PointNet architecture is 86.1% which we have taken as the baseline accuracy of this research. Our target is to achieve higher accuracy than the baseline. We have proposed a few variations of the encoder model and conducted extensive sensitivity and ablation studies on the model. We use the ModelNet10 dataset in this research. The experiments in this research achieve a 6.10% improvement over the baseline model. The contributions made in this research are summarized as follows:

- We propose a variant of 1D CNN and RNN models that improved the original PointNet on

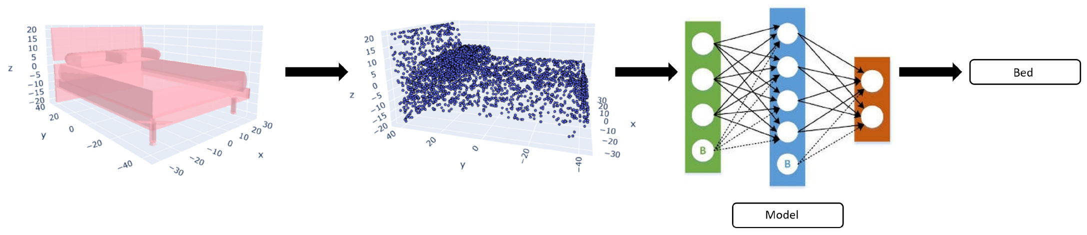


Figure 3.1: Overview of the PoineNet architecture.

the MobileNet10 dataset.

- We also introduce different loss functions to get better results for the model.
- Extensive hyper-parameter and ablation studies help to find the best settings for training the model.

3.2 Methodology

3.2.1 Preliminaries

The 3D point cloud is a two-dimensional dataset representing the points in the 3D mesh, which require a special form of the model to learn important features from the data and respect its underlying properties (e.g. invariant of transformation, interaction among neighbouring point). PointNet is one of the popular models that is specialized in 3D data handling [31]. To respect the invariant property of point cloud data and aggregate the information from all the points, maxpooling layer is used as a symmetric function. Convolutional neural networks (CNN) with $1x1$ kernel are the primary building block of the PointNet architecture. The original PointNet model consists of two transformation blocks, namely Tnet and Transform.

Tnet

The motivation behind using Tnet is to make the model invariant of geometric transformations (e.g. rigid transformation). Tnet is used to predict an affine transformation matrix directly applied to the coordinates of the input point cloud. Tnet is composed of maxpool, feature extraction and fully connected layers. Convolutional neural networks (CNN) with $1x1$ kernel have been used for

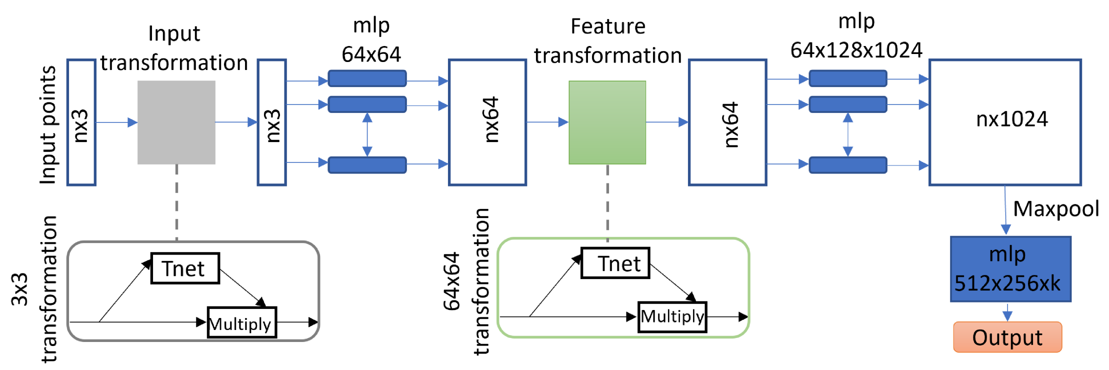


Figure 3.2: Overview of the PoineNet architecture.

feature extraction layers. ReLU, as an activation function and batch normalization, is used in every layer except the layer. The output of Tnet is a $3x3$ identity matrix.

Transform

Transform network is used to predict a feature transformation matrix that aligns the feature space of various input point cloud. Like Tnet, Transform network is composed of maxpool, feature extraction ($1x1$ CNNs) and fully connected layers. However, the higher dimension of the feature transformation matrix increases the complexity of optimization. As a result, a regularization term is used with the softmax training loss function. The output of Transform is a $64x64$ identity matrix.

Finally, the PointNet is just a composition of linear layers on top of the transformation blocks. An overview of the PointNet model is shown in Figure 3.2.

3.2.2 Explored Model variants

We explore different variants and combinations of CNNs for Tnet and Transformer and analyze their effects on the overall model's accuracy. The following model variations have been explored:

- The variation of CNNs has been made concerning kernel size and stride. For the kernel size, 3 and 5 have been used. For the stride, 2 and 3 have been used for both Tnet and Transformer blocks of the original PointNet architecture.

- Variants of Tnet and Transform block with RNNs are explored. More specifically, we use LSTM [9, 14], GRU [2], and vanilla RNNs in place of 1D CNN. For this, we use a hidden

Experiment	Best parameter	Accuracy (%)
Dropout	0.3	86.1
Epoch	20	86.67
Weight decay	0.00	86.1
Batch size	32	86.1
Learning rate	0.0001	87.78

Table 3.1: Summary of the hyper-parameter study.

size of 1024 and 2 hidden layers for all three models. Finally, a different number of linear layers are also explored in this experiment.

Long Short Term Memory (LSTM) is a popular recurrent neural network (RNN) composed of input gate, output gate and forget gate. It is a well-known network for analyzing very long sequence data which is suffered from vanishing gradient problem in vanilla RNNs [9, 14]. Like LSTM, Gated Recurrent Units (GRU) is another type of RNN. It comprises only input and forgets gate and has fewer parameters than LSTM [2].

3.2.3 Loss Functions

Original PointNet proposed to use negative log-likelihood in combination with a regularization softmax loss applied on the intermediate output of the Tnet and Transform block.

We experiment with different loss functions to train the model; for example, negative log-likelihood and categorical cross-entropy functions are used. Negative log-likelihood is used for multi-class classification tasks. This loss function returns a negative value for the log of each number in an interval of 0 and 1. As such, an additional negative sign is added to convert it to a positive number. At 0 the function returns ∞, $(log(0) = \infty)$ and at 1 returns 0 $(log(1) = 0)$. negative log-likelihood is formulated as follows:

$$Loss_{NLL} = -\sum_{i=1}^{c} y_i \; log(\hat{y_i}) \tag{1}$$

Like negative log-likelihood, categorical cross entropy is also used for multi-class classification problems. This loss function distinguishes two discrete probability distributions from each other.

Experiment	Accuracy (%)
PointNet baseline	**86.1**
Baseline w/o dropout and normalization	85.90
Tnet; 1D Conv (k=3, s=2)	84.14
Tnet; 1D Conv (k=3, s=3)	86.12
Tnet; 1D Conv (k=5, s=3)	86.67
Tnet; Linear 2 layer	84.91
Transform; 1D Conv (k=3, s=2)	**87.00**
Transform; 1D Conv (k=3, s=3)	84.36
Transform; 1D Conv (k=5, s=3)	83.70
Tnet & Transform; 1D Conv (k=3, s=2)	69.82
Tnet & Transform; 1D Conv (k=3, s=3)	53.96
Tnet & Transform; 1D Conv (k=5, s=3)	49.23
LSTM	83.26
GRU	79.96
RNN	77.43

Table 3.2: Experiments on different model variants.

When the similarity between two probability distributions increases, the minus sign makes the loss smaller [50].

$$Loss = -\sum_{i=1}^{c} p_i \, log(\hat{p_i}) \tag{2}$$

Here, $\hat{p_i}$ = $i-th$ probability of the output, p_i= probability of the $i-th$ class, c= number of classes.

The ModelNet10 dataset is not a balanced dataset with an equal amount of samples per class. So, we adopted a popular loss function called Focal Loss [21], which is specialized in handling class imbalanced datasets. The formula for focal loss is represented as follows:

$$Loss_{focal} = -\sum_{i=1}^{c} p_i \, (1-\hat{p_i})^{\gamma} \, log(\hat{p_i}) \tag{3}$$

here, $\hat{p_i}$ is the output probability from the model, $\gamma \in [0, 5]$ is a hyper-parameter that controls the penalty. If a prediction with a high probability $\hat{p_i}$ is wrong, the loss factor will be very high.

3.2.4 Implementation details

Augmentations

It is widely known for computer vision applications that applying augmentations can generate more data for training which in turn can reduce overfitting in the training. For point cloud data, there are not as many augmentation options as in the image. In this work, we adopted the 'rotation along the z-axis' and 'random noise' as the augmentations. Here, rotation along the z-axis changes the view of the angle for the object, which does not change the semantics of the data. The random noise augmentation adds some random noise to each of the points in the data point in point clouds.

Transfer learning

Transfer learning is a widely used method for computer vision tasks, where a model pre-trained on a task is used for another similar task. Using a pre-trained model on a different but similar task is called fine-tuning. The pretrained model is a starting point for fine-tuning the model on a different task. This approach prevents the model from the random initialization of weights, and the knowledge learned by the model in the pretraining stage reduces the overfitting tendency of models. Thus it increases the accuracy of the model.

Training setups

The model is trained with Adam optimizer with a constant learning rate of 0.001. The model is trained for 15 epochs by default with a batch size of 32. The training is done on Google Colab, which takes around 10 hours to run the default 15 epochs training setup.

3.3 Experimental Results

For this thesis, we take PointNet [31] as the baseline model. First, we run the model with the original PointNet on the MobileNet10 dataset. We get an accuracy of 86.1% on PointNet. Note that the original PointNet used ModelNet40 only; however, a separate paper [7] implemented the PoinNet on ModelNet10 and got an accuracy of 77.6%, which is far worse than the baseline results.

We do one more experiment on the original PointNet by removing the Batch-Normalization and the dropout components, which gave an accuracy of 85.9%. Considering this implementation of the PointNet on ModelNet10 as a baseline, the aim of the rest of the experiments in this research is to improve this accuracy (86.1%).

3.3.1 Hyper-parameter Sensitivity

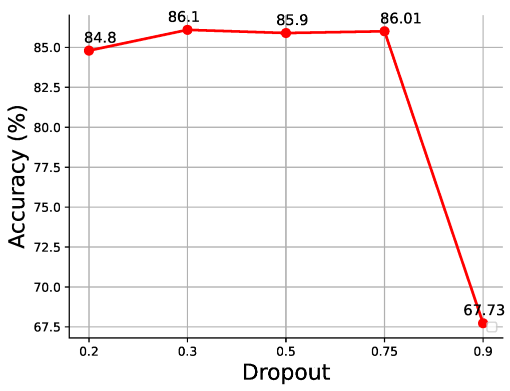
We first start with the hyper-parameter sensitivity study on the baseline method. The sensitivity study results are summarized in Table 3.1, which shows the effect of different dropout settings, the number of epochs, weight decay, batch size, and learning rate. More details of the results are presented in figure 3.3.

We experiment with the effect of different dropout values on the model accuracy (figure 3.3a). We experiment with dropout for 0.2, 0.5, 0.75 and 0.9. Among them, the model gives the highest accuracy, 86.01% for a 0.75 dropout value. A trend can also be seen in the model for different dropout values. The model accuracy increases by increasing the dropout rate to 0.75. After that, the model accuracy falls to 67.73% for a 0.9 dropout rate. Because the higher value of dropout means the higher the number of perceptrons of the model is avoided to reduce over-fitting of the model. Higher penalization reduces the model's accuracy.

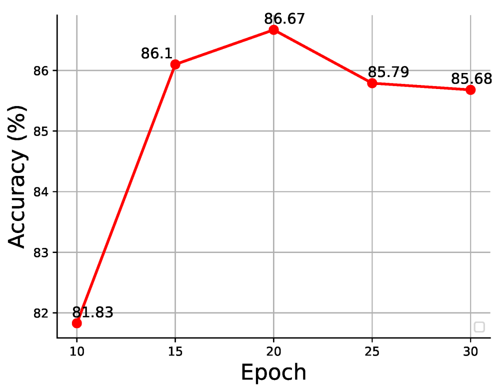
For the number of epochs (figure 3.3b), the highest accuracy is 86.67%, which is found in epoch 20. This is higher than the baseline accuracy of the model. The model's accuracy decreases for increasing the number of epochs by more than 20. Because training the model for a large number of epochs increases the overfitting of the model and reduces the model's generalization ability.

Weight decay is a regularization factor that penalizes by adding a penalty term to the cost function of a model, which shrinks the weights during back-propagation and thus reduces over-fitting. In figure 3.3c, we can see an increasing trend in the model's accuracy for decreasing values of weight decay. The highest accuracy of the model is 86.1% when no weight decay is imposed. So, for further experiments, weight decay can be avoided.

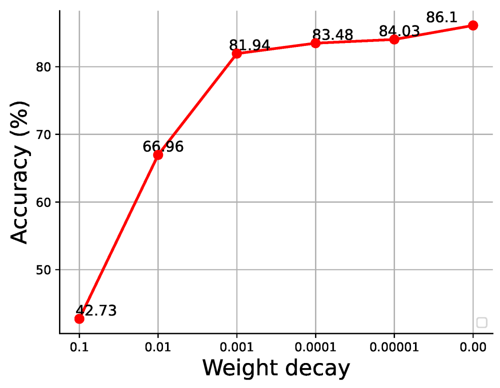
For the batch size (figure 3.3d), the highest accuracy of the model is found for 32, which is 86.1%. This is similar to the baseline accuracy. In the beginning, the model's accuracy increases as the batch size increases to 32; after that, the model's accuracy begins to decrease. Finally, the



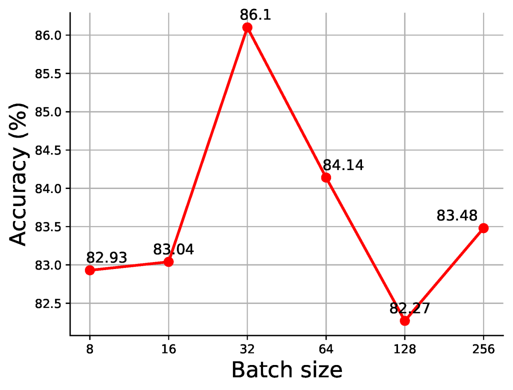
(a) Accuracy for different values of dropout.



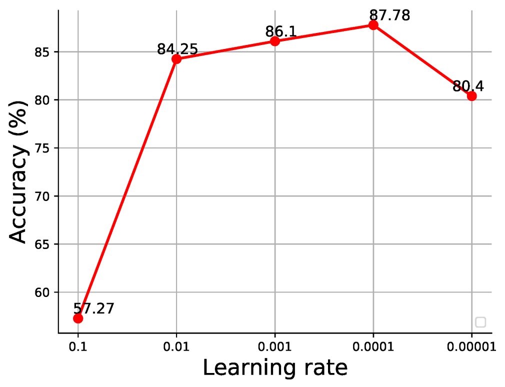
(b) Accuracy for different values of epoch.



(c) Accuracy for different Weight decay.



(d) Accuracy for different values of batch size.



(e) Accuracy for different values of Learning rate.

Figure 3.3: Sensitivity study on different parameters on the methods on all the datasets.

model's accuracy increases for batch size 256, which is 83.48%.

The learning rate controls the convergence rate of a model. Smaller learning rates tend to smaller update steps to the model's convergence, thus requiring a large number of epochs. A larger learning rate tends to larger update steps to the model's convergence, thus causing overshooting problems and delaying the model's convergence. In figure 3.3e, an increasing trend in the model's accuracy can be seen in decreasing the learning rate to 0.0001. The highest accuracy of the model is 87.78% for the 0.0001 learning rate, which is higher than the baseline accuracy. After that, the model's accuracy began to decrease as the smaller learning rate made the model's convergence speed slower.

3.3.2 Results for proposed Model Variants

The results for different model variants are presented in table 3.2. As we can see from the table, replacing the 1D CNN of kernel size 1 in the original Tnet block of PointNet with higher kernel size and strides shows mixed results. For instance, k=3 and s=2 hurts the model and get worse results, but there is a slight improvement for the same k with s=3. And there is a good amount of improvement for k=5 and s=3 compared to the baseline result. However, the best improvement is achieved when the 1D CNN of the 'Transform' block is changed with k=5 and s=2. It shows a 0.9% improvement over the baseline. An interesting result is seen when both the Tnet and Transform blocks of the PointNet are changed to a higher kernel size, in which case the results drop catastrophically. This indicates that the sparse point structure of the point cloud data needs to be processed with kernel size 1 (no relation comes into consideration from the neighbouring points) in at least one of the pre-processing blocks.

As mentioned in the method, we replace the 1D CNNs in the Tnet with RNNs, more specifically, LSTM, GRU, and vanilla RNN. The results of these variants of the model do not perform as well as the 1D CNNs. Here the best results are achieved by LSTM of 83.26%.

3.3.3 Results for explored Loss Functions

As mentioned in the method, we explore different variants of loss functions in the research. The results for different loss functions are summarized in table 3.3. First, we remove the regularization part of PointNet, leaving the Negative log-likelihood (NLL) only, which shows about a 0.5% drop in

Experiment	Accuracy (%)
PointNet	86.1
Negative log-likelihood	85.68
Categorical cross-entropy	**87.11**
Focal loss	86.78

Table 3.3: Effect of different loss functions.

Experiment	Accuracy (%)
Step LR sc. (step size= 4, gamma= 0.1)	88.11
Step LR sc. (step size= 6, gamma= 0.1)	87.33
Step LR sc. (step size= 4, gamma= 0.5)	**89.43**
Step LR sc. (step size= 6, gamma= 0.5)	88.33
Plateau sc.	87.67

Table 3.4: Summary of learning rate scheduler study.

accuracy. Next, we replace NLL with Categorical Cross-entropy loss, which shows a 1% improvement in the result compared to the original PointNet. The focal loss also shows 0.7% improvement compared to the baseline.

3.3.4 Explored Training protocol

The learning rate scheduler is used in models to adjust the learning rate between epochs as the training progresses. Here we have used two different scheduler methods to adjust the learning rate. One is learning rate scheduler, and another is plateau scheduler. From table 3.4 we can see that the plateau scheduler gets an accuracy of 87.67%, which is better than the contestant LR used in the baseline model. However, better results are achieved with Step LR. For the learning rate scheduler, four different combinations are used for two different parameters. They are step size and gamma. Step size is the number of epochs after which the learning rate will be updated. Gamma is the amount by which the learning rate will be changed. We have used the combination of step sizes 4 and 6 with gamma 0.1 and 0.5. The highest accuracy for step size 4 with gamma 0.5, which is **89.43**%.

Experiment	Accuracy (%)
Baseline	86.1
Without random noise	86.78
Without z axis rotation	**91.52**
Without any of them	90.42

Table 3.5: Effect of augmentation.

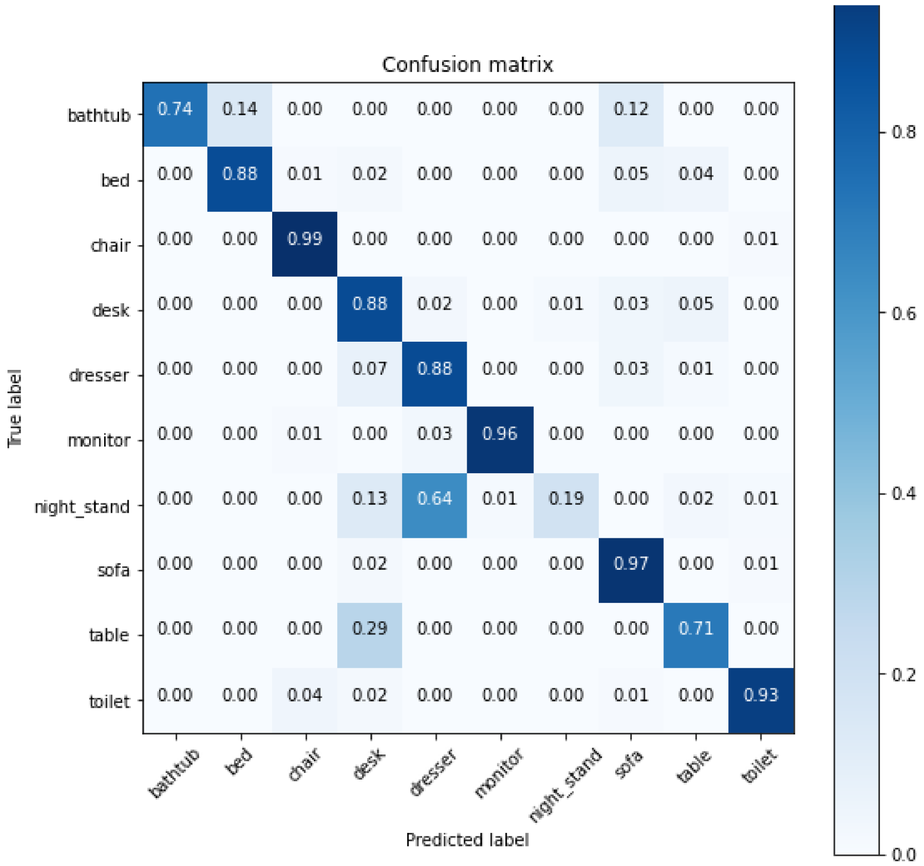
3.3.5 Effect of augmentations

In this thesis, we adopt two commonly known augmentations and consider them with the baseline. However, we do some ablation studies by removing them one at a time (table 3.5). Surprisingly, all the ablation results get better accuracy than the baseline including all. The best result here (and overall) is achieved by removing the rotation along the z-axis. It gets an accuracy of 91.52%. We suspect that the use of augmentation may require a bigger model with some other training settings to get the best out of augmentation. With the current setup, the best accuracy achieved in this research is 91.52% which is a significant **5.42%** improvement over the PointNet baseline.

3.3.6 Transfer Learning

In the transfer learning setting, we first pre-train the model on the ModelNet40 dataset. The weights learned by the model are used to fine-tune the model on the ModelNet10 dataset. A learning rate of 0.0001 is used while fine-tuning. The accuracy of the model after fine-tuning is **92.20%** which is the highest accuracy achieved by the proposed model. This is a significant **6.10%** improvement over the PointNet baseline.

3.3.7 Confusion Matrix



True label \ Predicted label	bathtub	bed	chair	desk	dresser	monitor	night_stand	sofa	table	toilet
bathtub	0.74	0.14	0.00	0.00	0.00	0.00	0.00	0.12	0.00	0.00
bed	0.00	0.88	0.01	0.02	0.00	0.00	0.00	0.05	0.04	0.00
chair	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.01
desk	0.00	0.00	0.00	0.88	0.02	0.00	0.01	0.03	0.05	0.00
dresser	0.00	0.00	0.00	0.07	0.88	0.00	0.00	0.03	0.01	0.00
monitor	0.00	0.00	0.01	0.00	0.03	0.96	0.00	0.00	0.00	0.00
night_stand	0.00	0.00	0.00	0.13	0.64	0.01	0.19	0.00	0.02	0.01
sofa	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.97	0.00	0.01
table	0.00	0.00	0.00	0.29	0.00	0.00	0.00	0.00	0.71	0.00
toilet	0.00	0.00	0.04	0.02	0.00	0.00	0.00	0.01	0.00	0.93

Figure 3.4: Confusion matrix.

Figure 3.4 represents the comparison between the true label and the predicted label of the model. As we see from the figure, the model performs reasonably well for most classes. Some of the class with higher error rate includes bathtub, nightstand, and table. For 'bathtub', the highest amount of confusion is observed for 'bed', followed by 'sofa'. Another high miss-classification observed is for 'night stand', which is mostly confused with 'dresser'. The 'nightstand' also misclassifies 'desk' sometimes. Class 'table' is often confused with the class 'desk'.

3.4 Conclusions and Future works

In this chapter, we used PointNet as a baseline to improve the accuracy of 3D point cloud data classification. We did extensive experiments on hyper-parameters and proposed different variants of the model. We also did an ablation study on different components of the model, including augmentations and loss functions. Overall our proposed training protocol achieved a 6.10% improvement over the baseline results of PointNet.

Chapter 4

DualNet: An Efficient Two-stream Fusion Network for 3D Point Cloud Data Classification

4.1 Introduction

In section 1.2, we discussed two major research gaps in the existing point cloud classification literature, which motivated us to propose a novel neural network architecture in this chapter. Also, the training protocol that we proposed in chapter 3 for the PointNet [31] model, is used as a default training protocol for the model we propose in this chapter. We aim to build a lightweight model that gives higher performance and less memory demand compared with the current local extractors.

We propose DualNet, a lightweight yet efficient residual MLP-based model that can take a larger amount of points as input without any significant increase in the computational cost. The proposed DualNet is composed of two parallel networks: DenseNet and SparseNet. Both DenseNet and SparseNet are composed of residual MLP blocks (inspired from PointMLP [26]). The DenseNet takes a large number of points as input but is comparatively more efficient than the SparseNet since the number of channels is low. As input, SparseNet takes a small subset of the input to the DenseNet. For training the model with fewer input points, SparseNet is composed of more channels, making the model computationally more expensive than the DenseNet. Combining more channels in SparseNet and fewer channels in DenseNet makes the overall DualNet lightweight. The SparseNet is also slightly less computationally expensive than the original PointMLP. More specifically, DenseNet contains 8 times fewer parameters than the SparseNet. Thus, DualNet can enjoy the advantages of more input points keeping the computational cost low simultaneously. To combine the two intermediate networks; SparseNet and DenseNet into DualNet, different types of lateral connections is used both in the final layer and the intermediates layers. An intensive ablation study is done on different lateral connections in the last layer (e.g. concatenation, multiplication, etc.), showing that the model's performance on the different lateral connections is data-dependent.

We test DualNet on two benchmark datasets: ModelNet40 [47] and ScanObjectNN [39]. On both datasets, DualNet achieves better accuracy than the state-of-the-art methods on the point cloud data classification task.

On the ModelNet40 dataset, DualNet achieves a 0.81% improvement over the state-of-the-art and on the ScanObjectNN dataset, our model achieves a 0.45% improvement over the state-of-the-art. Regarding computational complexity, our model takes about 38% less time compared to PointMLP. For example, on the ScanObjectNN dataset, PointMLP with 1024 points takes 25 hours to train an NVidia-100 GPU, while the DualNet with 128 points for SparseNet and 2048 points for DenseNet takes only 18 hours to train.

The main contributions of this research can be summarized as follows:

- We propose a new architecture for 3D point cloud data that can utilize a large number of points from the point cloud as the input to the model without drastically increasing the computation overhead.

- The proposed method is faster than the state-of-the-art methods and achieves similar or better accuracy.

- Extensive experiments on two benchmark datasets show the effectiveness of the proposed method and set new state-of-the-art accuracy on them

The rest of the chapter is organized as follows: In section 4.2, we present the details of the DualNet architecture. The experimental results are discussed in section 4.3. Finally, we draw the

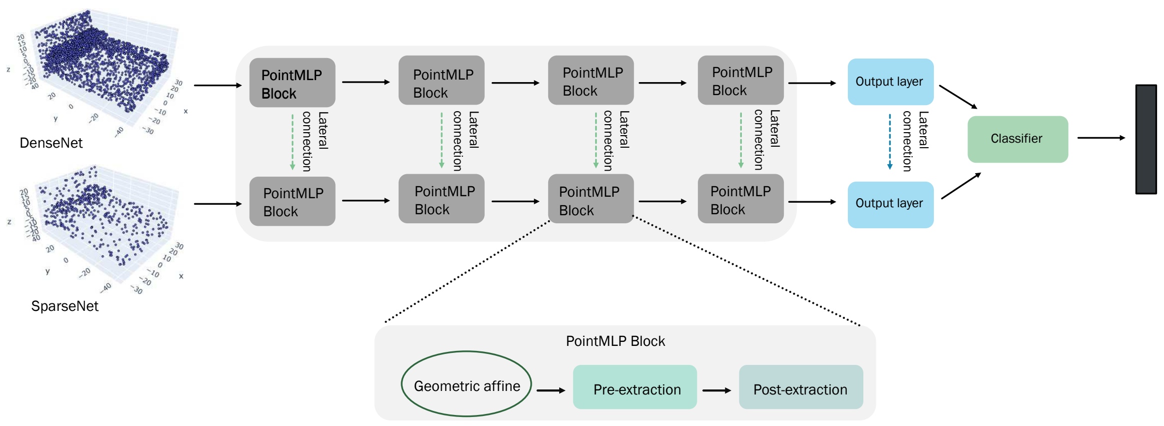


Figure 4.1: Architecture of DualNet. An upper branch of DualNet is a Dense network that takes a large number of points from point cloud as input, whereas a lower branch is a sparse network that takes a small number of points as input. Various lateral connection has been used both in the intermediate layers and output layer to fuse the following brunches into DualNet.

closing remarks and comment on the future direction of this research in section 4.4.

4.2 DualNet Networks

In this section, we describe the details of the proposed DualNet architecture. First, we present the preliminaries on some of the important concepts related to DualNet followed by a description of the proposed method.

4.2.1 Preliminaries

Here we describe the preliminaries on the building blocks of the proposed DualNet. The main block of DualNet utilizes the concept introduced in PointMLP [26]. PointMLP is developed upon PointNet++[33], which hierarchically learns local features. PointNet++ [33] extracts a partition of input data by using the K-nearest Neighbours algorithm and then uses the original PointNet [31] model on the nested partition of the input data in a recursive fashion. The following three sections describe the building blocks of PointMLP.

K-nearest Neighbours

PointNet++ [33] partitions the point cloud data into a set of overlapping points which shares common geometric structures among two consecutive sets of points. As a result, weights can be shared between different sets of points like the convolutional networks. PointNet++ [33] captures the local geometry of small neighbourhoods like CNNs. Due to robustness, the PointNet architecture is used as a local feature extractor. These local features are combined into larger units for extracting higher-level features. PointNet++ repeats this process upon the nested partitions of the input point set until features for the whole point set are obtained.

Pre-extractions and Post-extraction Blocks

As we have mentioned earlier, geometric local extractors are memory-demanding and also increase the model's computational complexities. To deal with this issue PointMLP [26] architecture uses simple residual MLPs. The model architecture is composed of two residual MLP blocks, namely, Pre-extraction and Post-extraction blocks, doing two different jobs (figure 4.2). The pre-extraction block learns local-regional shared weights while the post-extraction block extracts deep aggregated local features. FC, normalization layer, activation layers, and max-pooling (for aggregation) are used along with the two residual MLP blocks to build PointMLP architecture. To make the model deep and capture hierarchical features, this operation is repeated recursively for s stages. To make the model robust, a Geometric Affine Module (described in the next subsection) is used, which normalizes the input points. Finally, the K-nearest Neighbours algorithm is used for selecting the neighbours.

Geometric Affine Module

Due to the irregularity in the data format of point cloud data the geometric structures are also sparse and irregular in different local regions. This demands diverse local extractors for extracting different geometric features. Shared residual MLPs can not capture them. Thus decreasing the accuracy of the model. To deal with this issue, a geometric affine module has been introduced in PointMLP that normalizes the input points while keeping its original geometric properties intact.

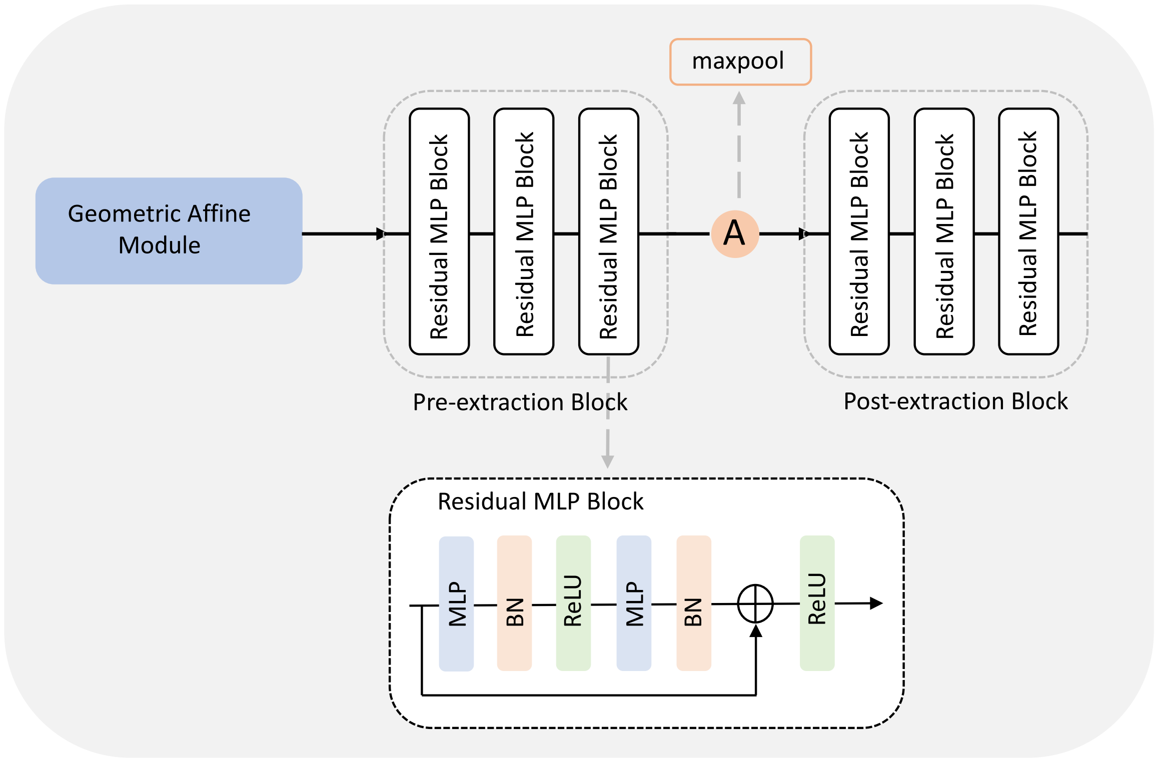


Figure 4.2: Overview of PointMLP block.

This block is shown to help to solve the aforementioned problem.

4.2.2 DualNet

Figure 4.1 shows the detail architecture of our proposed model, DualNet. DualNet is a further improvement upon PointMLP [26]. We have used the PointMLP blocks to build our dual network; SparseNet and DenseNet. By fusing these two networks into our final model, DualNet gives state-of-the-art accuracy. For better extraction of the local features, the farthest point sampling is used with K-nearest Neighbour algorithms developed in the PintNet++ [33].

The main idea behind DualNet architecture is a parallel network of two sub-networks: DenseNet and SparseNet. The intuition of the DenseNet is to be a computationally efficient network and yet capture the high-level abstraction about the input object from a large number of points. On the other hand, SparseNet is a deep network that captures detailed features of the object from a small number of points. This type of model is shown to be useful for video processing [5]. We have introduced a few key design choices to achieve the desired property of the mentioned network.

First, to learn the high-level features from a large number of points while being computationally efficient, we designed the DenseNet to have a reduced number of channels. Second, the SparseNet is a deep network that captures detailed features. This network contains a similar number of parameters as PointMLP. To reduce the computational cost we reduce the number of neighbours in this model. This is also motivated by the fact that the number of inputs to this model is also very low. Finally, we introduce the intermediate and final lateral connection between the DenseNet and the SparseNet. The intermediate lateral connection sends the learned intermediate representation to learn from DenseNet to the SparseNet. That way, SparseNet gets the information from a large number of points that DenseNet processes.

As the main building block on DualNet is adopted from PointMLP, it gets some of the important properties of PointMLP. Since the PointMLP blocks are residual MLP blocks, DualNet is simple yet generic. Also, using residual MLPs allows the model to be less memory-demanding and decreases computational complexities. Before feeding the input points to the networks, the points are normalized by the geometric affine module.

SparseNet

As mentioned above, the SparseNet comprises residual MLP blocks from the PointMLP. The main distinction of SparseNet from the PointMLP comes from the input number of point N_s and the number of neighbours n_s. Given the total number of sample points in the point cloud data is N, the input to the SparseNet is $N_s = N/f_i$, where f_i is an efficiency factor that reduced the number of input points to the SparseNet. For a typical value of f, the input to the SparseNet is $N_s \ll N$. Since the input points are lower, we take this opportunity to reduce the computational cost by reducing the number of neighbours n_s in the nearest neighbour calculation. A typical value of $n_s = n_d/2$, where n_d is the number of neighbours in the DenseNet. The number of channels in the SparseNet C_s is similar to that of the PointMLP. Overall, the computational cost of SparseNet is slightly less than that of PonitMLP [26] but contains the same number of parameters.

DenseNet

DenseNet is one of the key elements of the proposed DualNet that process a high number of points while being computationally efficient. This model also has two important design parameters: (a) input number of points N_d, and (b) number of channels C_d. Note that, we do not change the value of the neighbour selecting factor (k) in the K-nearest Neighbours algorithm from the typical implementation of PointMLP or PointNet++. Here, to reduce the computational cost and learn the high-level features from a large number of points, we desire to make a lightweight model. The channel number C_s is reduced to $C_d = C_s/f_c$, where f_c is the channel reduction factor. Here, we use a high number of points as input, with the maximum being $N_d = N$. The intermediate representation learned from this DenseNet is passed to SparseNet to help the SparseNet learn from the high number of input points. Next, we describe the lateral connections that help with this objective.

Lateral Connections

Finally, we describe the details of the lateral connection of the proposed DualNet. As mentioned before, there are two types of connection: (a) intermediate connection, and (b) final connection.

The intermediate connection ensures the information learned from a high number of points in the DenseNet is passed to the SparseNet. However, the size of the intermediate representations is different as these two networks are not the same. More specifically, the representation generated by DenseNet has fewer channels but a higher embedding dimension. To deal with this issue, we propose a simple transformation to adjust the output size. First, we apply a maxpool layer with kernel size f_i, which reduces the embedding size to that of SparseNet at that particular layer. Then to adjust the channel dimension, we repeat this at the channel dimension f_c times. This operation makes the intermediate representation of the DenseNet same as that of SparseNet. Next, we multiply the output of that layer from the SparseNet with the transformed output of DenseNet. This lateral connection is repeated in the s stages of the network.

The final connection is a more straightforward implementation. This connection is added on the final embedding from the DenseNet to SparseNet. At this layer, the output dimension is the same for both networks making it an easy connection. Here we explore concatenation, multiplication, and addition as the lateral connection. As we will discuss in the result section, this lateral connection has a crucial role in the performance of the proposed DualNet model.

Instantiating

The proposed DualNet introduces a few model parameter choices that are crucial to the design of this network. We show a detailed study on the effect of some of the parameters in the results (e.g. input size to the SparseNet and DenseNet). Here, we describe the default values for all the model-specific parameters mentioned in the methods.

The input to the DenseNet, N_d, by default is 2048, and the efficiency factor f_i is tested for values ranging from 2 to 16. The number of channels is $c_s = 32$, and $c_d = 16$ has been used respectively for the SparseNet and DenseNet. For selecting neighbours, the K-nearest Neighbours algorithm has been used with $n_d = 24$ for DenseNet and $n_s = 12$ for SparseNet. For the intermediate lateral connection, DualNet uses $s = 4$ intermediate connections and a final connection.

4.2.3 Implementation Details

We have trained the model for 300 epochs with batch size 32 and an initial learning rate of 0.01. To adjust the learning rate, cosine annealing scheduler has been used. We have trained our model with synchronous SGD [8], 0.9 of Nestorov Momentum, and 0.0002 of weight decay. The number of input points is randomly initialized for every experiment we have performed for both the SparseNet and DenseNet. For the augmentation strategy for the input points, we have followed the recipe in [31]. We ran experiments for 300 epochs with the Standard SGD optimizer and reported the overall accuracy (OA) of our model compared with the state-of-the-art.

4.3 Experiment and Results

In this section, we present intensive experiments on two popular datasets, namely, ModelNet40 and ScanObjectNN. First, we present the experimental setups, followed by results on ModelNet40 and ScanObjectNN. Finally, we show a sensitivity study on various important training parameters

Table 4.1: Classification result on ModelNet40 dataset

Method	Parameter	Input	Accuracy
PointNet [31]	-	1K	89.2
PointNet++ [33]	1.4M	1K	90.7
Improved PNet [29]	-	1K	92.2
PointCNN [19]	-	1K	92.5
PointConv [46]	18.6M	1K	92.5
KPConv [38]	15.2M	7K	92.9
DGCNN [30]	-	1K	92.9
RS-CNN [24]	-	1K	92.9
DensePoint [23]	-	1K	93.2
PointASNL [49]	-	1K	92.9
PosPool [25]	-	5K	93.2
MLMSPT [13]	-	1K	92.9
LGR-Net [51]	-	-	91.1
Cartesian GCN [43]	-	-	73.0
PointMLP[26]	13.2M	1K	92.9
DualNet (ours)	14.3M	0.5k/1k	**93.7**

for our proposed DualNet model.

4.3.1 Experiment setups

ModelNet40 [47] is a point cloud dataset containing 9,843 training data samples and 2,468 test samples of 3D CAD objects classified into 40 categories. As a synthetic dataset, there is no noise and missing data. So, no pre-processing is required. The ScanObjectNN [39] is a recently published benchmark dataset that contains 15,000 objects belonging to 15 categories and 2,902 unique real-world object instances. As a real-world dataset, there exists background, missing data, and noise, which makes the dataset challenging for 3D point cloud data classification. Since ModelNet40 is a synthetic dataset, point cloud data analysis might be comparatively easy and may not pose the challenges that a real-world dataset does. To this end, we present the results for both ModelNet40 and ScanObjectNN datasets to fully evaluate the effectiveness of the proposed model.

4.3.2 Results on ModelNet40

In this section, we present the results for DualNet on the ModelNet40 dataset. First, we show the comparative study with the state-of-the-art methods on this dataset. Then we show the performance on different input points and ablation study on the proposed lateral connection.

Comparison to SOTA methods

In table 4.1, we present the final results of DualNet and its comparison with previous methods on ModelNet40 datasets. As we see from this table, DualNet achieves state-of-the-art results with 93.7% accuracy on this dataset. This is a 0.81% improvement over the current state-of-the-art PointMLP method. Note that, the PointMLP achieves 92.91% accuracy on the ModelNet40 dataset in our experiments from the available code of the original paper. DualNet achieves this accuracy for only 512 point inputs in the SparseNet and 1,024 points on DenseNet. Here, the DenseNet with 1024 model has a *very* low computational cost compared to the SparseNet. From the perspective of model parameters, PointMLP had a total parameter of 13.2 million, and DualNet slightly increased it to 14.3 million.

To further understand the performance of DualNet and its robustness in a low number of points, we present a direct comparison with PointMLP in table 4.2. Here, 'points' shows the input point to the SparseNet. Since DenseNet comes with negligible extra cost with more points, we do not report the input to DenseNet in this study. From an overview of this table, we see that DualNet shows significant improvement compared to PointMLP for any number of input points. The maximum improvement of **2.15%** is achieved for 64 points. For 128 and 256 points, the improvements are 1.53% and 1.17%, respectively. The overall highest accuracy **93.72%** is achieved from 512 points as the input. In this setting, the comparative improvement is 1.05%. For a higher number of input points (e.g. 1024), DualNet gets a 0.53% improvement over PointMLP. This concludes that the proposed DualNet is more effective with fewer points. However, it still shows improvements with a larger number of points too.

Table 4.2: Comparison of DualNet with state-of-the-art methods on ModelNet40.

Points	PointMLP	DualNet	Improvement
64	90.92	93.07	**2.15**
128	91.78	93.31	1.53
256	92.14	93.31	1.17
512	92.67	**93.72**	1.05
1024	92.91	93.44	0.53

Sensitivity and Ablation study on DualNet Architecture

There are two main components of the proposed DualNet architecture: (1) the number of input points in SparseNet and DenseNet, and (2) the lateral connection. Here, we show an in-depth analysis of them with the ModelNet40 dataset.

Performance for a different number of input points. We show the effect of different numbers of inputs to the SparseNet and DenseNet in Table 4.3. Although we show a direct comparison to PointMLP for different input points to SparseNet, we further show the effects of different numbers of points on the DenseNet in this experiment. We fix the input point to the SparsNet and show the results for different values (higher than the input to SparseNet) for DenseNet. Overall, we observe that, with the increment of input points in DenseNet, the accuracy of DualNet increases. Note that the computational cost does not increase significantly as long as the input point to SparseNet is the same. This trend continues up to 1024 points. However, With the further increment of points (e.g. 2048), the accuracy of the model declines in most of the cases except that for 128 input points in SparseNet. The highest accuracy of **93.72%** is obtained from 512 and 1024 input points for SparseNet and DenseNet, respectively.

Ablation study on lateral connection. In table 4.4, we do a detailed ablation study on different types of lateral connections on the last layer of DualNet to show the effect of lateral connections on the performance of DualNet. We test the model with 3 types of lateral connections: concatenation, multiply, and addition. We also show another study without any lateral connection on the last layer, represented by ”none”. Similar to the previous experiment, here, ’points’ indicate the input to the SparseNet. For 64, 256, and 512 input points, the ”addition” type of lateral connection provides higher accuracy. For 128 input points, we get the highest accuracy without any lateral connection at the last layer. However, the accuracy for ’addition’ is also *very* close for this setting. It can

Table 4.3: Accuracy with more points on DenseNet

SparseNet	DenseNet	Accuracy (%)
64	128	91.01
64	256	91.73
64	512	92.42
64	1024	92.87
64	2048	91.98
128	256	91.65
128	512	92.91
128	1024	93.07
128	2048	*93.31*
256	512	92.26
256	1024	93.23
256	2048	92.18
512	1024	**93.72**
512	2048	92.83
1024	2048	91.05

Table 4.4: Ablation study on different connection on last layer

Points	Concatenation	Multiply	Addition	None
64	92.63	92.42	92.87	92.59
128	93.03	92.87	93.07	93.11
256	92.46	92.95	*93.23*	93.19
512	91.73	93.03	**93.72**	92.30

be concluded that for the ModelNet40 dataset, the ”addition” type of lateral connection shows a promising result.

4.3.3 Results on ScanObjectNN

In this section, we present a similar set of experiments to that of ModelNet40 for the ScanObjectNN. Since this is a more complex and real-world dataset, it shows a difference in observations from that of the previous section.

Table 4.5: Classification result on ScanOjectNN dataset

Method	Parameter	Accuracy
PointNet [31]	-	63.0
PointNet++ [33]	1.4M	77.9
SpiderCNN [48]	-	68.2
PointCNN [19]	-	78.1
DGCNN [30]	-	78.1
DRNet [34]	-	80.3
PRANet [3]	-	82.1
MVTN [12]	-	82.8
PointMLP[26]	13.2M	85.9
DualNet (ours)	14.3M	**86.4**

Table 4.6: Comparison of DualNet with state of the art

Points	PointMLP	DualNet	Improvement
64	75.95	84.94	**8.99**
128	81.09	**86.36**	5.27
256	82.62	**86.36**	3.74
512	84.91	85.67	0.76

Comparison to SOTA methods

In table 4.5, we present the results for the DualNet and its comparison to previous methods over the ScanObjectNN dataset. As we see from this table, DualNet beats the state-of-the-art (PointMLP) by 0.5%. As shown before, the previous SOTA, PointMLP had 13.2 million parameters, and DualNet achieved the SOTA of 86.4% accuracy with the parameters slightly increased to 14.3 million. This accuracy is significantly higher than all other methods in this table.

For analyzing the performance of DualNet on the ScanObjectNN dataset, we do a direct comparison of DualNet with the SOTA in table 4.6. Similar to previous experiments, we report the number of input points only for the SparseNet due to the negligible computation cost of DenseNet. We can see that a significant improvement of **8.99%** is achieved by DualNet over the PointMLP for 64 input points in the SparseNet. For 128 and 256, the improvement of DualNet over SOTA is 5.27% and 3.74%, respectively. The lowest improvement is achieved from 512 input points which are 0.76%. Hence, similar to the observation from ModelNet40, it can be concluded that DualNet can achieve better improvement over the SOTA with fewer input points in the SparseNet. Moreover, the highest accuracy of the DualNet over ScanObjectNN dataset of **86.36%** is achieved for both 128 and 256 input points in the SparseNet. However, we can say the performance of DualNet for 128 input points on SparseNet is more significant than 256 as the computation cost with the former point is lower. More specifically, the run-time for 128 input points is 18.2 hours compared with 256 input points which is 22.9 hours.

Sensitivity and Ablation study on DualNet Architecture

Similar to the previous ablation and sensitivity study on ModelNet40, we perform the study on (1) the sensitivity toward the number of input points on SparseNet and DenseNet, and (2) the ablation study on the lateral connection of the last layer of DualNet.

Performance for a different number of input points.

In table 4.7, we show the performance of DualNet on different numbers of input points on the DenseNet. Previously we showed the performance of DualNet on a different number of input points on the SparseNet only. Here, we fix the number of input points for the SparseNet and show the accuracy of varying the input points for DenseNet. As we can see from this table, DualNet achieves the highest accuracy of **86.36%** for both 128 and 256 input points in the SparseNet. In both cases, the number of input points in DenseNet is 2048. The second highest accuracy of *85.67%* is achieved for 512 input points in the SparseNet and 2048 input points in the DenseNet. So, it can be concluded that DualNet shows higher performance for many input points in DenseNet. As we mentioned before, DenseNet is a lightweight network with less number of parameters. Using a higher number of points as input does not increase the computational cost of DualNet. Therefore, DualNet enjoys the advantage of a higher number of input points.

Ablation study on lateral connection.

In table 4.8, we compare the performance of DualNet for different types of lateral connections on the last layer for the ScanObjectNN dataset. As before, We consider 3 different types of lateral connections: (1) concatenation, (2) multiply, and (3) addition. We also perform another analysis by removing all the lateral connections at the last layer, described as ”none”. As we observe from this

Table 4.7: Accuracy with more points on DenseNet

SparseNet	DenseNet	Accuracy (%)
64	128	75.75
64	256	78.77
64	512	82.55
64	1024	84.25
64	2048	84.94
128	256	79.11
128	512	82.27
128	1024	83.94
128	2048	**86.36**
256	512	83.03
256	1024	84.53
256	2048	**86.36**
512	1024	84.63
512	2048	*85.67*
1024	2048	84.94

table, the performance varies with the changes in the number of input points. The highest accuracy **86.36%** is achieved for both multiply and concatenation types of lateral connection. Note that, for the ”multiply” type of lateral connection, this accuracy is achieved for only 128 input points. The concatenation type of lateral connection requires 256 input points to achieve this accuracy. The second highest accuracy *85.81%* is achieved with the ”addition” type of lateral connection at the last layer. This study shows that the performance of ScanObjectNN with lateral connection is sensitive to the number of points. We suspect that the variance and distortion present in the ScanObjectNN dataset are responsible for such deviation in the results compared to ModelNet40 (where ’addition’ was the best lateral connection).

4.3.4 Run Time Analysis

We present a run time analysis in Table 4.9. This table shows the training time for PointMLP and DualNet on the ScanObjectNN dataset. This particular instance shows the runtime of PointMLP with 1024 points, which is 25.1 hours. While the proposed ScanObjectNN with 2048 points to

Table 4.8: Ablation study on different connection on last layer

Points	Concatenation	Multiply	Addition	None
64	84.49	84.11	84.87	84.94
128	85.29	**86.36**	84.84	84.56
256	**86.36**	84.66	85.81	84.35
512	84.70	84.84	85.29	85.67

Table 4.9: Run time comparison on ScanObjectNN dataset

Method	Input	Accuracy (%)	Runtime (hr)
PointMLP	1024	85.91	25.1
DualNet	128/2048	86.36	18.2

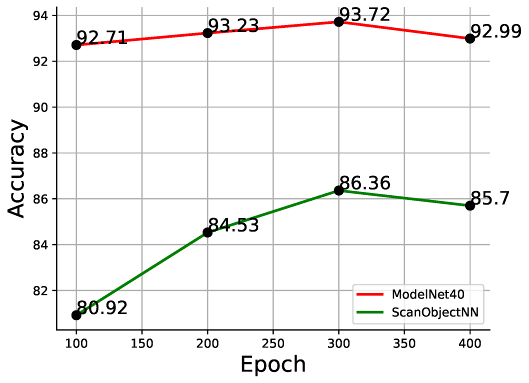
DenseNet and 128 points on SparseNet takes only 18.2 hours to train to make it 38% faster than PointMLP. At the same time, DualNet achieves 0.45% improvements over PointMLP. This shows both the effectiveness and improvements of the DualNet over PointMLP.

4.3.5 Hyper-parameter Sensitivity Analysis

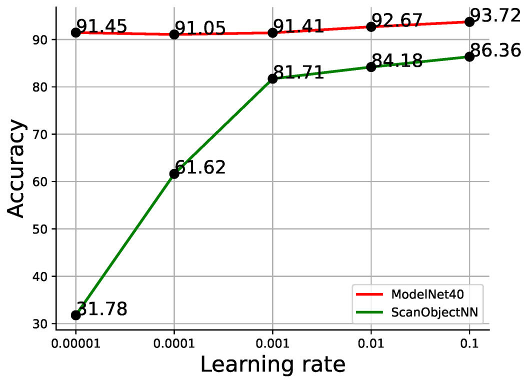
In this section, we present the hyper-parameters study on different important parameters for training the models, including (a) the total number of training epochs, (b) learning rate, (c) weight-decay, and (d) batch size (figure 4.3). The findings from this study are summarized below:

Performance for different training epochs. Figure 4.3a presents the results for the accuracy against the different number of epochs. As we observe from this table, for both ModelNet40 and ScanObjectNN datasets, the best accuracy is observed for 300 epochs. Training lower than 300 lacks does not learn and generalize well. However, training longer also shows reduced accuracy. This is caused by over-fitting to the training set. Other observations from this figure indicate that ScanObjectNN is more sensitive to the number of epochs than ModelNet40. More specifically, ModelNet40 has about 1% deviation with different numbers of epochs, while the ScanObjectNN can have up to almost 6% deviation. For the rest of the experiments in this paper, we set 300 epochs as the default for training.

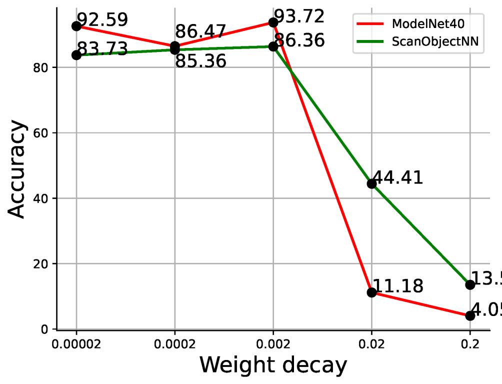
Performance for different learning rate Figure 4.3b demonstrates the performance of DualNet for different learning rate values. As we observe from this figure, ModelNet40 is comparatively



(a) Accuracy for different values of epoch.



(b) Accuracy for different learning rates.



(c) Accuracy for different Weight decay.

Figure 4.3: Sensitivity study on different parameters on the DualNet on ModelNet40 and ScanObjectNN datasets.

stable against different learning rates. However, the ScanObjectNN is highly sensitive to the values of the learning rate. On ScanObjectNN, the highest accuracy **86.36%** is achieved for 0.1. After that, the model's accuracy starts declining exponentially. For 0.0001 and 0.00001 learning rates, the model's accuracy is 61.62% and 31.78%, respectively. This indicates that the model cannot learn while training for 0.0001 and 0.00001 learning rates on the ScanObjectNN dataset. On the ModelNet40 dataset, the highest accuracy **93.72%** is achieved for a 0.1 learning rate. After that, the model's accuracy also declines. Lowering more does not hurt too much, but it does not help either for ModelNet40.

Performance for different weight decay values. We present the results for different values of

Table 4.10: Accuracy for different batch size on ModelNet40 and ScanObjectNN

Batch size	ModelNet40	ScanObjectNN
8	13.21	85.5
16	92.18	85.64
32	**93.72**	**86.36**
64	92.67	84.07

weight decay to train the DualNet in figure 4.3c. The best accuracy for ModelNet40 and ScanObjectNN is achieved at 0.002. Overall we observe that both ModelNet40 and ScanObjectNN are sensitive to the value of weight decay. For instance, using larger values than 0.002 hurts the performance significantly. For ScanObjectNN, the accuracy drop from 86.36% to 44.41% and on ModelNet40, it drops from 93.73% to 11.18%. These trends continue for even higher values where the model almost learns nothing, and accuracy drops to 13.52% and 4.05% for ScanObjectNN and ModelNet40, respectively. We also do not benefit from reducing the learning rate from 0.002.

Performance for different batch-size. Finally, we present the sensitivity study for different batch-size for both datasets in table 4.10. We immediately observe that the best accuracy for both datasets is obtained at the batch size of 32. Both increasing and reducing the batch size has a large shift in the accuracy. For instance, with a learning rate of 8, the accuracy of ModelNet40 dropped significantly (only 13.21%). Also, note that batch-size larger than 64 did not fit in our GPU.

Overall, the sensitivity study from this section shows that the DualNet is somewhat hyper-parameter sensitive. However, a good set of parameters can achieve good results. We can also conclude that the parameters are not dataset sensitive, as both datasets got the best results for the same parameter setting in all the studies above.

4.4 Discussion and Conclusion

In this chapter, we proposed DualNet, a novel architecture for point cloud data analysis. DualNet is a simple residual MLP-based model with the fusion of two sub-networks: SparseNet and DenseNet. This model solves one important limitation of the existing literature where the input number of points to the network is not scalable. That means, increasing the number of points linearly increases the computational cost and can not take advantage of more points available in most of the point cloud data. DualNet provides three major contributions to the 3D point cloud data literature: (1) It can grasp the benefit of a large number of points as input from the point cloud data without a significant increase in computational complexity; (2) DualNet is faster than the existing method resulting in better accuracy with less training time. (3) It sets a new SOTA accuracy on two benchmark datasets (ModelNet40 and ScanObjectNN) for point cloud data classification. The comparison of Dualnet with existing literature in table 4.1 depicts the efficiency and improvement of our model. DualNet achieves 0.81% improvement over the SOTA method with only 512 and 1024 input points for DenseNet and SparseNet on the ModelNet40 dataset. For the ScanObjectNN dataset, DualNet achieves 0.45% improvement over the SOTA for only 128 input points in the DenseNet (table 4.5). To analyze the computational cost of DualNet, we perform runtime analysis on the ScanObjectNN dataset in table 4.9. For 128 and 2048 input points in Densenet and SparseNet, respectively DualNet 86.36% accuracy in only 18.2, making it 38% faster than the SOTA. Therefore, it proves the computational efficacy of DualNet while taking the benefit of a larger number of input points.

Chapter 5

Conclusion

5.1 Research Summary and Contribution

This chapter presents a summary and discussion on the future research direction of our research. We have generally focused on 3D point cloud data classification from the architectural improvement and computational efficiency point of view. The overall research can be summarized as follows:

- In chapter 2, we have done an extensive literature review on existing point cloud data classification literature and pointed out the research gap, which worked as our research motivation. We have also described three popular point cloud benchmark datasets: ModelNet10, ModelNet40 and ScanObjectNN, commonly used for training and evaluating the models. In chapter 3 and chapter 4, we presented our proposed solution to deal with two of the research problem described in this chapter.

- In chapter 3, we propose an improved training protocol for the point cloud data classification model [31] and show that there is still room for improvement in PointNet classification accuracy. The baseline accuracy of PointNet was 86.01%, on which we show considerable improvement. We also show an extensive hyper-parameter study and explore different model variants, such as 1D CNN with different strides and kernel sizes, and variants of RNNs e.g. LSTM, GRU and Vanilla RNN. Three different loss functions have been explored. Among them, the categorical cross-entropy achieved the highest accuracy of 87.11%. Our sensitivity study includes training parameters, including learning rate, batch size, dropout, weight decay and epoch. Finally, The highest accuracy, 91.52%, came from augmentation without z-axis rotation. Overall, our proposed training protocol achieved a 6.10% improvement over the PointNet baseline.

- In chapter 4, we proposed DualNet, a novel two-stream fusion network for point cloud data classification. DualNet is a residual MLP-based model which is lightweight, computationally efficient and can enjoy the benefit from large point cloud data input points. DualNet is composed of two parallel networks, SparseNet and DenseNet. The difference between the two parallel networks is the number of input points and channels. The combination of SparseNet and DenseNet makes the DualNet lightweight. We described the proposed training protocol in chapter 3 along with its default training protocol for DualNet. The model is tested on two point cloud benchmark datasets: MdelNet40 and ScanObjectNN. We did an extensive study on different numbers of input points on both of the datasets, where DualNet achieves better accuracy than SOTA. Also, a detailed ablation study on the lateral connection of the last layer shows that the model's performance is data-dependent. Finally, we do a run-time analysis of DualNet which depicts the computational efficiency of the model. DualNet is 40% faster than SOTA. While being computationally efficient DualNet increases the SOTA for ModelNet40 and ScanObjectNN by 0.81% and 0.45%. DualNet is also scalable to the number of input points, making it a feasible choice for high-density point clouds.

5.2 Limitation and Future Work

The improved training protocol for 3D point cloud data classification was proposed in chapter 3. One limitation and future work lie in the fact that the hyper-parameters and other ablation studies are done on one component at a time. Due to the time and computational requirements, all possible combinations are not explored, which has the potential to further improve the result.

In chapter 4, the use of residual MLP as the building block of DualNet provides a simple and generic structure making it feasible to use in other applications of computer vision and pattern recognition tasks. Such an efficient model will also be helpful for future research on point cloud data, including object detection, segmentation, pose detection, and so on. Since the irregular data format and large dataset size make point cloud data analysis challenging, our work focuses on making a lightweight, faster model yet gaining the benefit of a larger number of input points. We did not explore the effectiveness of this model in other applications (e.g. object detection, or semantic segmentation). Future work can also extend the model for other point cloud data-related domains.

Bibliography

[1] BRUNA, J., ZAREMBA, W., SZLAM, A., AND LECUN, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[2] CHUNG, J., GULCEHRE, C., CHO, K., AND BENGIO, Y. Gated feedback recurrent neural networks. In *International conference on machine learning* (2015), PMLR, pp. 2067–2075.

[3] FAN, D.-P., JI, G.-P., ZHOU, T., CHEN, G., FU, H., SHEN, J., AND SHAO, L. Pranet: Parallel reverse attention network for polyp segmentation. In *International conference on medical image computing and computer-assisted intervention* (2020), Springer, pp. 263–273.

[4] FANG, Y., XIE, J., DAI, G., WANG, M., ZHU, F., XU, T., AND WONG, E. 3d deep shape descriptor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 2319–2328.

[5] FEICHTENHOFER, C., FAN, H., MALIK, J., AND HE, K. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 6202–6211.

[6] FENG, M., ZHANG, L., LIN, X., GILANI, S. Z., AND MIAN, A. Point attention network for semantic segmentation of 3d point clouds. *Pattern Recognition 107* (2020), 107446.

[7] GARCIA-GARCIA, A., GOMEZ-DONOSO, F., GARCIA-RODRIGUEZ, J., ORTS-ESCOLANO, S., CAZORLA, M., AND AZORIN-LOPEZ, J. Pointnet: A 3d convolutional neural network for real-time object class recognition. In *2016 International joint conference on neural networks (IJCNN)* (2016), IEEE, pp. 1578–1584.

[8] GOYAL, P., DOLLÁR, P., GIRSHICK, R., NOORDHUIS, P., WESOLOWSKI, L., KYROLA, A., TULLOCH, A., JIA, Y., AND HE, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* (2017).

[9] GRAVES, A. Long short-term memory. *Supervised sequence labelling with recurrent neural networks* (2012), 37–45.

[10] GUO, K., ZOU, D., AND CHEN, X. 3d mesh labeling via deep convolutional neural networks. *ACM Transactions on Graphics (TOG) 35*, 1 (2015), 1–12.

[11] GUO, M.-H., CAI, J.-X., LIU, Z.-N., MU, T.-J., MARTIN, R. R., AND HU, S.-M. Pct: Point cloud transformer. *Computational Visual Media 7*, 2 (2021), 187–199.

[12] HAMDI, A., GIANCOLA, S., AND GHANEM, B. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 1–11.

[13] HAN, X.-F., KUANG, Y.-J., AND XIAO, G.-Q. Point cloud learning with transformer. *arXiv preprint arXiv:2104.13636* (2021).

[14] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[15] KANEZAKI, A., MATSUSHITA, Y., AND NISHIDA, Y. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 5010–5019.

[16] LEE, J., CHEON, S.-U., AND YANG, J. Connectivity-based convolutional neural network for classifying point clouds. *Pattern Recognition 112* (2021), 107708.

[17] LI, G., MÜLLER, M., QIAN, G., PEREZ, I. C. D., ABUALSHOUR, A., THABET, A. K., AND GHANEM, B. Deepgcns: Making gcns go as deep as cnns. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[18] Li, L., Zhu, S., Fu, H., Tan, P., and Tai, C.-L. End-to-end learning local multi-view descriptors for 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 1919–1928.

[19] Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems 31* (2018).

[20] Li, Y., Pirk, S., Su, H., Qi, C. R., and Guibas, L. J. Fpnn: Field probing neural networks for 3d data. *Advances in neural information processing systems 29* (2016).

[21] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2980–2988.

[22] Lin, Z.-H., Huang, S.-Y., and Wang, Y.-C. F. Learning of 3d graph convolution networks for point cloud analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, 8 (2021), 4212–4224.

[23] Liu, Y., Fan, B., Meng, G., Lu, J., Xiang, S., and Pan, C. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 5239–5248.

[24] Liu, Y., Fan, B., Xiang, S., and Pan, C. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 8895–8904.

[25] Liu, Z., Hu, H., Cao, Y., Zhang, Z., and Tong, X. A closer look at local aggregation operators in point cloud analysis. In *European Conference on Computer Vision* (2020), Springer, pp. 326–342.

[26] Ma, X., Qin, C., You, H., Ran, H., and Fu, Y. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123* (2022).

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

[28] Pais, G. D., Ramalingam, S., Govindu, V. M., Nascimento, J. C., Chellappa, R., and Miraldo, P. 3dregnet: A deep neural network for 3d point registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 7193–7203.

[29] Paul, S., Patterson, Z., and Bouguila, N. Improved training for 3d point cloud classification. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)* (2022), Springer, pp. 253–263.

[30] Phan, A. V., Le Nguyen, M., Nguyen, Y. L. H., and Bui, L. T. Dgcnn: A convolutional neural network over large-scale labeled graphs. *Neural Networks 108* (2018), 533–543.

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

[32] Qi, C. R., Su, H., Niessner, M., Dai, A., Yan, M., and Guibas, L. J. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 5648–5656.

[33] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems 30* (2017).

[34] Qiu, S., Anwar, S., and Barnes, N. Dense-resolution network for point cloud classification and segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2021), pp. 3813–3822.

[35] Rusu, R. B., Blodow, N., and Beetz, M. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation* (2009), IEEE, pp. 3212–3217.

[36] Savva, M., Yu, F., Su, H., Aono, M., Chen, B., Cohen-Or, D., Deng, W., Su, H., Bai, S., Bai, X., et al. Shrec16 track: largescale 3d shape retrieval from shapenet core55. In *Proceedings of the eurographics workshop on 3D object retrieval* (2016), vol. 10.

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

[38] Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 6411–6420.

[39] Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, T., and Yeung, S.-K. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 1588–1597.

[40] Vinyals, O., Bengio, S., and Kudlur, M. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391* (2015).

[41] Wang, D., Tang, L., Wang, X., Luo, L., and Yang, Z.-X. Improving deep learning on point cloud by maximizing mutual information across layers. *Pattern Recognition 131* (2022), 108892.

[42] Wang, D. Z., and Posner, I. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems* (2015), vol. 1, Rome, Italy, pp. 10–15.

[43] Wang, H., Zhang, Y., Liu, W., Gu, X., Jing, X., and Liu, Z. A novel gcn-based point cloud classification model robust to pose variances. *Pattern Recognition 121* (2022), 108251.

[44] Wang, Q., Chen, J., Deng, J., and Zhang, X. 3d-centernet: 3d object detection network for point clouds with center estimation priority. *Pattern Recognition 115* (2021), 107884.

[45] WANG, Y., SUN, Y., LIU, Z., SARMA, S. E., BRONSTEIN, M. M., AND SOLOMON, J. M. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog) 38*, 5 (2019), 1–12.

[46] WU, W., QI, Z., AND FUXIN, L. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 9621–9630.

[47] WU, Z., SONG, S., KHOSLA, A., YU, F., ZHANG, L., TANG, X., AND XIAO, J. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1912–1920.

[48] XU, Y., FAN, T., XU, M., ZENG, L., AND QIAO, Y. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 87–102.

[49] YAN, X., ZHENG, C., LI, Z., WANG, S., AND CUI, S. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 5589–5598.

[50] ZHANG, Z., AND SABUNCU, M. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems 31* (2018).

[51] ZHAO, C., YANG, J., XIONG, X., ZHU, A., CAO, Z., AND LI, X. Rotation invariant point cloud analysis: Where local geometry meets global topology. *Pattern Recognition 127* (2022), 108626.

[52] ZHAO, H., JIANG, L., JIA, J., TORR, P. H., AND KOLTUN, V. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 16259–16268.