

Semantic Segmentation of 3D Point Clouds Using Deep Learning



By

Muhammad Hasnat Manzoor

FALL-2018-MSCS 00000277277 SEECS

Supervisor

Dr. Muhammad Shahzad

Department of Computing

School of Electrical Engineering & Computer Science (SEECS)

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

(July 2022)

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "Semantic Segmentation of 3D Point Clouds Using Deep Learning " written by MUHAMMAD HASNAT MANZOOR, (Registration No 00000277277), of SEecs has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: _____ *M. SHAHZAD* _____

Name of Advisor: _____ Dr. Muhammad Shahzad _____

Date: _____ 07-Jul-2022 _____

HoD/Associate Dean: _____

Date: _____

Signature (Dean/Principal): _____

Date: _____

Approval

It is certified that the contents and form of the thesis entitled "Semantic Segmentation of 3D Point Clouds Using Deep Learning " submitted by MUHAMMAD HASNAT MANZOOR have been found satisfactory for the requirement of the degree

Advisor : Dr. Muhammad Shahzad

Signature: M. SHAHZAD

Date: 07-Jul-2022

Committee Member 1: Dr. Muhammad Moazam Fraz

Signature: M. Moazam Fraz

Date: 08-Jul-2022

Committee Member 2: Dr. Qaiser Riaz

Signature: Qaiser Riaz

Date: 09-Jul-2022

Committee Member 3: Dr. Asif Ali

Signature: Asif Ali

Date: 09-Jul-2022

Dedication

I dedicate this report to my family and my teachers who always supported me and pushed me in any way possible to become what I am today. Their sacrifices seeded my success, especially my MOM DAD who always supported me and pushed me throughout this journey.

Certificate of Originality

I hereby declare that this submission titled "Semantic Segmentation of 3D Point Clouds Using Deep Learning " is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name: MUHAMMAD HASNAT MANZOOR

Student Signature: 

Acknowledgments

Glory be to Allah, the Creator, the Sustainer of the Universe. Who only has the power to honor whom He please, and to abase whom He, please? Verily no one can do anything without His will. From the day, I came to NUST till the day of my departure, He was the only one Who blessed me and opened ways for me, and showed me the path to success. There is nothing that can pay back for His bounties throughout my research period to complete it successfully. Firstly, I would like to express my sincere gratitude to my research supervisor, Dr. Muhammad Shahzad, for allowing me to do research and providing invaluable guidance throughout this research. It was a great privilege and honor to work and study under his guidance. I am extremely grateful for what he has offered me. Without his support and help, this work wouldn't have been completed. My sincere thanks also go to GEC members, Dr. Moazzam Faraz and Dr. Qaisar Riaz for their support and encouragement. Last, but not least, I am extremely grateful to my parents for their love, prayers, care, and sacrifices in educating and preparing me for my future. I am very much thankful to my siblings for their love, understanding, prayers, and continuing support to complete this research work.

Muhammad Hasnat Manzoor

Table of Contents

Introduction	1
1.1 Introduction	2
1.2 Problem Statement.....	4
1.2.1 Research Hypothesis.....	4
1.2.2 Research Questions.....	4
1.2.3 Overview of Proposed Approach	5
1.3 Main Contributions.....	6
1.4 Structure of Thesis.....	7
1.4.1 Chapter 2: Literature Review	7
1.4.2 Chapter 3: Design and Methodology.....	7
1.4.3 Chapter 4: Results and Discussion	8
1.4.4 Chapter 5: Conclusion	8
1.4.5 Chapter 6: Bibliography	8
Literature Review	9
2.1 Chapter Overview.....	10
2.2 Background.....	10
2.3 3D Scene Understanding	10
2.3.1 Point-based Networks.....	10
2.3.2 Sparse Convolution Networks	11
2.3.3 Pyramid Networks	12
2.3.4 Graph Neural Networks.....	13
2.4 3D Representation Learning.....	14
2.5 Data-Efficient Learning	15
Design And Methodology	18
3.1 Problem Definition	19
3.2 Architecture Design.....	19
3.2.1 PyraContrast as Pretext Task.....	20
Algorithm 1- PyraContrast	20
3.2.1.1 Input Module:	21
3.2.1.2 Transformation Module:.....	21
3.2.3 Fine-tuning on Downstream Tasks.....	28

Implementation	29
5.1 Data Pre-processing.....	30
5.2 PyraContrast as Pretext Task.....	31
5.2.1 PointInfoNCE Loss.....	31
5.2.2 Transformations.....	32
5.2.3 Batch Normalization.....	32
5.2.4 Activation Function.....	33
5.2.3 In-Convolution Block Implementation.....	34
5.2.4 Pyramid Module Implementation.....	34
5.2.5 Out-Convolution Block Implementation.....	35
Results and Discussion	36
5.1 Data.....	37
5.2 Evaluation Metrics.....	37
5.3 Training Setting.....	38
5.4 Pre-training Results.....	39
5.5 Fine-tuning on Semantic Segmentation.....	40
5.6 Fine-tuning on Instance Segmentation.....	42
Conclusion & Future Work	45
6.1 Concluding Remarks.....	45
6.2 Conclusion & Future Work.....	45
References.....	48
Appendix.....	51

List of Figures

Figure 1	Structure of the pipeline.	19
Figure 2	PyraContrast as Pretext Task.....	21
Figure 3	Pyramid Point Network (PPN).....	22
Figure 4	In-Conv Block.....	23
Figure 5	Inner Pyramid Module	24
Figure 6	Out-Convolution Block.....	25
Figure 7	Visualization of Scene Context.....	27
Figure 8	Visualization of ScanNetV2 Pair Dataset.	30
Figure 9	Visualization of Batch Normalization.....	33
Figure 10	Rectified Linear Unit (ReLU)	33

List Of Tables

Table 1	PyraContrast Algorithm	20
Table 2	Pre-training Loss.....	39
Table 3	Iteration Timing in Seconds	39
Table 4	Semantic Segmentation on S3DIS	40
Table 5	Stanford Area 5 Test (Fold 1).....	51
Table 6	Instance Segmentation on S3DIS Dataset.....	42
Table 7	AP on instance Segmentation.....	43
Table 8	Instance Segmentation on Stanford Area 5 Test [5]	51

Abstract

Arguably, one of the deep learning’s greatest achievements is transfer learning. It is proven that if we use a rich source dataset to pre-train a network (ImageNet in 2D) and after we use a smaller target dataset to be fine-tuned, it can help to boost performance. It has been used in many applications including language and vision. In 3D scene understanding, there have been few works using this method. Because annotating 3D data is difficult. In this work, we aim to further facilitate research on 3D representation learning. To achieve this goal, we select different datasets and downstream tasks to find the effectiveness of unsupervised pre-training on a large and rich source dataset of a 3D scene. The results we obtain are encouraging. we are using a unified backbone, source dataset, and contrastive objective for unsupervised pre-training, and further supervised downstream tasks are performed. Our method is achieving the almost same result in half time of recent works in both pre-training and downstream tasks like semantic segmentation, and instance segmentation.

Keywords: Unsupervised learning, Representation learning, 3D scene understanding.

Chapter 1

Introduction

1.1 Introduction

Point clouds are a large collection of tiny individual points plotted in 3D space. Each point has its coordinates X, Y, and Z axis along the other features like color value, which is in RGB format, intensity, depth, and many more. These features are helpful while analyzing. The points have a geometric structure such as triangles, lines, or curved surfaces which form a model.

Point clouds are created by scanning any physical object or structure which is completed by using a laser scanner or photogrammetry. Laser scanners send the light to the object and when light reaches the surface of the object it reflects and hits the scanner so by measuring the time, these are used to determine the exact location of points on an object and form a point cloud. Similarly, in photogrammetry, pictures are used to create measurements. By taking pictures of an object from different angles, we triangulate points on the object and plot these points in 3D space, and create a 3D model. There are many applications for the 3D point clouds across different industries. For example, in the architecture industry, as-built models are used which is helpful for the engineer to visualize the site without visiting it. 3D CAD (computer-aided design) models are used for manufacturing parts, quality inspection, metrology, animation, and rendering. It is also rapidly growing in computer vision such as robotics, remote sensing, self-driving cars, and medical care.

Let's talk about the different tasks in machine learning for better understanding. In machine learning, classification is a process in which we categorized the given into one or multiple classes. Data can be structured or unstructured. Another task is object localization in which an object is located in given data which can be an image or 3D scene. Typically, it is specified using a bounding box around the object. But object detection is a bit more complex than the previous two tasks. It combines the concept of classification and localization. In this task, a given data algorithm will return bounding boxes around all objects of interest and then assign a class to them. Image segmentation is another concept in which input is segmented so it can be processed with classification or object detection. It is

classified into different categories which are instance segmentation and semantic segmentation. Semantic segmentation links every pixel of an image with a class label such as a table, cup, etc. so it takes many objects of the same class as a single entity. But in instance segmentation, this processes multiple objects of the same class as separate single instances. These tasks can be performed using different learning methods which are supervised learning, unsupervised, and reinforcement learning.

In supervised learning, input has the labeled dataset which is helpful in classification problems and regression problems. But the unsupervised learning is the total opposite of supervised learning means there is no labeled dataset. It is self-learning and the main goal is to find the patterns and predict the output. In simple words, we gave the machine the data and it look for the hidden features and crusted the understandable data.

In computer vision, deep learning is getting a lot of attention for good reasons because it's a machine learning technique that helps the computer to understand like humans do which is to learn by example. In deep learning, the machine can learn 3D data that can provide more detail such as rich geometric, shape, and scale information as compared to 2D data. Point clouds don't lose the original geometric information in 3D space that's why it is preferred in many scenes for an understanding application like autonomous driving and robotics. So, if we talk about the driverless car, they should enable them to recognize stop signs and detect other cars, cycles, or pedestrians in real-time.

In 2D, unsupervised pre-training on a large rich dataset like ImageNet can be used to enhance the performance once we finetuned on a smaller dataset and it is successful in many applications [1]. In a 3D scene understanding task on the point clouds, training from scratch is time-consuming due to its sparse structure. In the past few years, unsupervised pre-training is gaining attention in 3D learning [2], [3].

In our thesis, a similar approach is used on the 3D point clouds. We kept our pretraining stage unsupervised. Which used the Scan-Net dataset and then perform the supervised downstream tasks (e.g., Semantic Segmentation, instance

segmentation) to boost the performance. The training and validation datasets, results and discussion will be covered next.

1.2 Problem Statement

The problem is that it is hard to collect the 3D data and even harder to annotate it. More, it is time-consuming to train the model from scratch. In 3D deep learning, the technique of unsupervised pre-training on a rich dataset and after supervised fine-tuning is limited to numerous factors. Some of the factors are the absence of unified backbone architecture, large-scale and rich datasets, and lack of high-level tasks for validation.

To solve this challenge and improve the previous studies, our problem statement is as follows:

It is possible to Select a large dataset (like ImageNet in 2D, for 3D it's Scan-Netv2 [4] which will be used in unsupervised pre-training and select a backbone architecture that will be shared in different supervised downstream tasks for fine-tuning, evaluating unsupervised objectives in pre-training stage and create an evaluation system on different supervised downstream tasks.

1.2.1 Research Hypothesis

This research hypothesis is as follows:

Unsupervised pre-training on a large dataset can be used to boost the performance in fine-tuning stage on different supervised downstream tasks by eliminating limitations.

1.2.2 Research Questions

The research problem we want to solve, according to the above-mentioned hypothesis, is by eliminating limitations, the performance of the high-level tasks can be improved by unsupervised pre-training.

- **Research Question 1:** what is the benefit of the unsupervised pre-training stage?

Objective: Studies shows that high-level task like Semantic Segmentation performance can be Improved by pre-training as compared to training it from scratch.

- **Research Question 1:** which dataset will be used in the unsupervised pre-training stage?

Objective: The objective of this question is, that ScanNetV2 [4] is an RGB-D dataset containing 2.5 million views with 1500 Scans that will be used for this purpose.

- **Research Question 2:** What will be the backbone architecture?

- **Objective:** The goal of this question is, that Pyramid Network Architecture is the backbone model in unsupervised pre-training as well as in supervised fine-tuning in different downstream tasks.

1.2.3 Overview of Proposed Approach

We proposed that instead of training the 3D understanding task from scratch, which is time-consuming, we can use pre-training and then fine-tune our tasks. For this purpose, we have demonstrated a pipeline that shows that in the future, the scale will be preferred over the precise annotations due to unsupervised pre-training.

We select a rich source dataset which is ScanNetV2 [4], on which unsupervised pre-training will be done and for the backbone architecture, the Pyramid network [5] will be utilized in our unsupervised pre-training stage as well as downstream tasks and focus on point cloud learning on 3D data. we want to use our PyraContrast for the complex 3D scene understanding so most 3D scene understanding uses conventional U-Net structure which has encoder layers followed by decoder layers. There are generally connected by the similar size of encoders and decoders layers. this U-net structure, however, has some drawbacks and the results obtain from this structure find it difficult to segment fine details. This is because the structure which is using the max pooling and subsampling of feature layers, and this results in reduced feature maps resolution. So, in each encoder layer, the receptive fields are decreased and this makes the U-Net structure very challenging to segment the small object with a high level of precision and accuracy. But the Pyramid Network passed features in a dense pyramid structure and increases the feature map dimension sharply which helps for improving classification accuracy. So, our network can take a second look in layers simultaneously and allows the network to acquire various and different receptive field views. PointInfoNCE [2] is contrastive loss which is computed in our pre-training stage. Next, we select a downstream task for fine-tuning, which includes semantic segmentation and instance segmentation on S3DIS [6] dataset as our target dataset. Our supervised downstream task also utilized the Pyramid Network architecture built with Minkowski Engine [7]. So, we used an unsupervised deep learning model for pretraining on ScanNetV2 [4] and experiments are performed supervised finetuning for different downstream tasks. The proposed approach and results are discussed in further depth in chapters 3 and 4 of the current thesis, respectively

1.3 Main Contributions

The following major contributions were made during this master’s research project. In this thesis, we proposed a pipeline that will utilize a unified architecture

in the unsupervised pre-training stage on the 3D point cloud and then supervised fine-tuning for Semantic and instance segmentations as our downstream tasks.

- Pyramid Network, which is our unified architecture, is shared across all stages and demonstrated better performance.
- For unsupervised pre-training, ScanNetV2 [4] is used as our rich dataset.
- For fine-tuning, semantic segmentation and instance segmentation are done on S3DIS [6] dataset.

1.4 Structure of Thesis

The remainder of this thesis is laid out as follows:

1.4.1 Chapter 2: Literature Review

This includes a complete review of the literature. Other ways from many disciplines to handle a comparable research problem are summarized along with their flaws. The literature review chapter will explain that the suggested 3d scene understanding networks, 3d Representation learning, and unsupervised pretraining and supervised downstream tasks techniques are giving the same results but faster.

1.4.2 Chapter 3: Design and Methodology

In this chapter, the technique of the proposed deep learning model for unsupervised pre-training and supervised downstream tasks are explained with the help of figures and tables. The methodology starts with an overview of the model. There are stages for our pipeline. The first one is PyraContrast which is a pretext task. PyraContrast contains a sampling block, transformation block,

Pyramid Point Network (PPN), and contrastive loss block. The second stage is fine-tuning different downstream tasks.

1.4.3 Chapter 4: Results and Discussion

This chapter contains the details of the implementation and testing performed on our pipeline. The detail of the rich dataset used for the pre-training stage and the target dataset used for the fine-tuning stage is discussed in the first section. Evaluation metrics used for pre-training results and fine-tuning result analysis are also discussed in the pre-training setting. This chapter contains the tables and figures to show the evaluation outcomes and comparisons of different methods with our methodology.

1.4.4 Chapter 5: Conclusion

The second last section of this thesis covers two things, in particular, a conclusion and a summary of our contributions. The conclusion conveys the concluding remarks and future discussions regarding our work. Meanwhile, the summary of work briefly states the contributions of this thesis for the pre-training in 3D point clouds with fine-tuning.

1.4.5 Chapter 6: Bibliography

Finally, the last section of this thesis document contains the bibliography/references used in the thesis in IEEE style.

Chapter 2

Literature Review

2.1 Chapter Overview

We analyze the relevant literature in this chapter to highlight the existing work on unsupervised pretraining and supervised downstream tasks. The understanding of existing work also helps us in justifying the proposed solution later in this thesis.

2.2 Background

Training from scratch on the target data is still the dominant approach in the 3D understanding tasks. Few works have been done on the unsupervised pre-training and supervised downstream tasks.

2.3 3D Scene Understanding

The 3D sensors (Lidar, depth-sensing cameras) are growing and over the last few years, 3d scene understanding technology is much needed that can process the 3D data and then it can detect objects in a scene or predict the classes. A 3D scene often consists of objects of interest (e.g., cycles, cars, pedestrians), and such technology can help in autonomous driving and many more applications. It can also lead to new research areas. Different neural networks help in different 3D scene understanding tasks which are instance segmentation and object detection and semantic segmentation.

2.3.1 Point-based Networks

Most recently, Research in deep learning on 3D point clouds has been switched from synthetic, single object detection classification [8]–[10] to challenging large-

scale, real-world scene understanding. PointNet [8] and PointNet++ [9] open a new door to work directly on the raw point cloud.

- PointNet [8] is a unified architecture that takes the point clouds directly as raw and predicts the class label for the entire point cloud or it can predict the per point segment or part label for the given point of input. The architecture of this network is simple. Each point is processed identically, and they are no longer dependent on each other in the initial stages. For simplicity, each point is represented as (x, y, z) but it can contain other dimensions too (local or global features). It uses a symmetric function which is max pooling. The goal of the network is to learn a group of optimization functions that select the interested or informative points in a point cloud and encode it. In the end, there are fully connected layers of the network that will aggregate these optimal values of functions into the global descriptor of the entire shape (shape classification). It can also use to predict the per point labels which is shape segmentation.
- The problem in PointNet [8] was that it does not capture the local structures which were already there in the point cloud. It only gives the global feature of the point cloud and local structures are important for the success of convolution architectures. So, it was limited only to generalizing complex scenes. PointNet++ [9] solve this problem by using its hierarchical neural network which processes a set of points hierarchically given by a metric space. The idea is to first generate a set of points into the overlapping local regions by the distance metric in a given space. In the small neighborhoods, local features are captured which have fine geometric structures. Then these local features are further grouped into larger units and then high-level features are produced. This process is repeated until the features of whole point sets are obtained.

2.3.2 Sparse Convolution Networks

Recently, Sparse Convolution networks [11] [12] are showing more promised results in the deep learning for point clouds. These networks have computational efficiency and [11] [13] [14] show state-of-the-art performance for the 3D scene understanding tasks.

In the 3D dataset, high-dimensional perception is difficult and [11] adopts a Sparse tensor [7] which will be used in sparse convolution networks. The generalized sparse convolution encodes all different convolutions as a subclass which is important for high-dimensional perception. It takes less memory for computation and is fast.

- For the 3D instance segmentation, a bottom-up end-to-end network PointGroup [14] whose goal is to generate better groups of points. It extracts per-point semantic prediction and then performs an efficient point group to get the candidate's object instances. Descriptive features are extracted using a semantic segmentation backbone and output a semantic label for each point. Relative offset is learned using the offset branch which brings each point to its respective ground-truth instance center point. Given predicted semantic labels and offsets, these groups of points are converted to the cluster. For reference, coordination is taken from each point, making a group with the same label of nearby points and then the group is progressively expanded. To pick the candidate group and evaluation, ScoreNet [14] is used. In the end, non-maximum suppression is used to remove the duplicate predictions.

2.3.3 Pyramid Networks

Instead of a conventional “U” shape structure in semantic segmentation, a dense pyramid structure allows the network to simultaneously reviews all the layers for a second time. Creating the extra layers which have less noise, helps to increase the contextual information.

- Pyramid Point [5] uses this pyramid concept and feature fusion to solve the issue which is decreased feature map resolution. It transfers the features in a dense pyramid structure. It allows the ability to transfer to features between various units and extra layers which restrict them to define a U shape path. So, by restricting the U shape path, the network has a different receptive field view. The network has also many shallow feature layers which are not covered in the entire U shape structure. The goal of the decoding unit is to add the noise. The features that have passed more than one through the decoder units, so instead of conventional four, they are less chance to have segmentation errors and that is important for small objects. This network has a Feature Kernel Point Convolution (FKP Convolution) whose goal is to add an element of attention to the kernel. This is usually used kernel point convolution.

2.3.4 Graph Neural Networks

A graph neural network aggregates the features along the edges and then iteratively updates its vertex. This aggregation is somehow the same as in set-based deep learning, near the edges, GNN can learn more complex features. There is no need to be sampled and group vertices every time. [15] [16] [17] used the graph neural network in part segmentation, classification, and object detection respectively on a point cloud.

- Point-GNN [17] uses the input of the point cloud as raw. The goal of this network is to output the category and bounding box of the objects to which every vertex of the graph belongs. It is a single-stage detector, and it can detect many objects in a single shot. There is translation variance in the point cloud so to make the Point-GNN translation invariance, an auto registration method is used which helps the point to align the coordinates based on the features. A boxing merging and scoring operation are used to combine the detection results from multiple vertices accurately.

- DG-CNN [15] is inspired by PointNet [8] and convolution operations. PointNet works on the individual points, but DG-CNN [15] created a local neighborhood graph to use the local geometric features and then apply the operation similar to convolution on the edges which are connecting neighboring pairs of points. EdgeConv creates an edge feature which is the relationship between a given point and its neighbors. It is invariant to the ordering of neighbors in a point cloud, so it is permutation invariant. This graph is not fixed, and it is dynamic which updated the layers of the network dynamically. From layer to layer of the network, the set of K-nearest neighbors of a point change is computed from the sequence of embedding. Proximity in the feature space is different from the proximity in input, which leads to nonlocal diffusion of information in the whole point cloud.

2.4 3D Representation Learning

In 2D representation learning, transferring learning from a big dataset to a smaller target dataset has become dominant. But in 3D representation learning, it is not widely used.

- PointContrast [2] initiate the work on unsupervised pre-training on the rich dataset and supervised fine-tuning on the smaller dataset by selecting a large ScanNetV2 [4] which will be used in the pre-training. It has a full convolution design along the point-level matrix learning. The Idea of full convolution comes from FCN [18] and FCGF [19] which take the input point cloud as a whole and find the corresponding without cropping the scene; this way from a very large number of neighboring points, a local descriptor can aggregate large information and gives the output full - resolution (*i.e. for the P Points, the network gives the P corresponding vectors as output*) due to sparse residual U-Net architecture. For metric learning, positive/negative pairs are defined on the point level. Select the

two views x_1 and x_2 from given a distribution-sampled point cloud and the source dataset is ScanNetV2 [4]. These two views x_1 and x_2 are selected from every fixed frame which are in the same world coordinates. There is a need to compute the similarity of these two views where the similarity between these two views should be a minimum of 30% so a mapping function is applied over the pair of view. Then two transformation functions are applied on the pair, by which the point cloud will be further transform to two different and distinct views. This transformation makes the work more challenging because we need to find an equivariance between these two transformed views. There is need to compute the point features for these two views, so U-Net architecture was chosen due to sparse residual. For the pre-training objectives, two different contrastive losses were chosen, Hard-contrastive loss [19], and PointInfoNCE loss [19]. Which is the updated version of InfoNCE [20] loss and used in 2D vision pretraining. So contrastive loss is applied over two views, if these two views are matched then it will minimize the distance between them and if two views are unmatched then distance will be maximized. After the pre-training stage on the ScanNetv2 [] dataset with the U-Net architecture, a huge collection of target datasets and the downstream task are selected. Downstream tasks are semantic segmentations on S3DIS [6], ScanNetV2 [4], Synthia4D [21] and ShapeNetPart [22]; and object detection on the ScanNetV2 [4] and SUN RGB-D [23]. Backbone architecture remains the same for both pre-training and fine-tuning. So instead of training from scratch, PointContrast [2] first uses unsupervised pretraining on a large dataset and then supervised fine-tuning by using the pre-trained weights on different downstream tasks and datasets which helps to gain performance.

2.5 Data-Efficient Learning

The data-efficient learning's main goal is to learn with limited available training datasets or labels because Collecting and annotating 3D point clouds is an expensive job. In contrast, training a deep neural network is usually not data efficient because it relies on a large or rich source of the annotated dataset.

- [3] uses the concept of PointContrast [2] but also gives a new way of learning in 3D scene understanding with limited data or supervision. There are two main settings for data-efficient: (1) *limited scene reconstruction (LR)* and (2) *limited annotation (LA)*. In the 1st setting, it gives the possibility that there can be a bottleneck in the number of scenes that can be scanned and reconstructed. In the 2nd setting, there can be a case where in each scene, labeling is done on a small set of points. The pre-training objective of PointContrast [2] is to find the equivariance of points on which random geometric transformations are applied. Given a pair of points, a simple contrastive loss is applied over the point features. Which objective is to minimize the distance if points are matched (positive pairs) and maximize the distance if points are unmatched (negative pairs) but the simple contrastive learning in PointContrast [2] only applies to point-level matching, and it completely misses the spatial configuration and context in the scene and does not capture any spatial information: the unmatched pairs (negative pairs) could be sampled from random locations in many scenes in given mini-batches. The spatial context is the relative pose, distance, and direction and this information could be important for complex 3D tasks such as instance segmentation. So, to capture the spatial information in the pre-training objective, [3] takes the idea from ShapeContext local descriptor which helps in shape matching. The shape context descriptor [24] first partitions the given space into the spatially inhomogeneous cells and then, it has the spatial context about the shape at each point which it encodes in each cell. This spatial context information is obtained by computing a histogram over the number of neighboring points. At a high level, this its objective is also to capture the distribution over relative locations in a scene. So, a scene of the point cloud is partitioned into multiple regions, instead of computing a contrastive loss

for the whole point set sampled in a mini-batch, now contrastive learning is applied over multiple partitioned regions separately and, in the end, the loss is aggregated.

Chapter 3

Design And Methodology

3.1 Problem Definition

Because useful representation learning has been successful in the 2D domain, we introduce our pipeline for unsupervised pre-training on a large dataset and supervised fine-tuning on a small dataset for 3D point clouds. We adopt a unified backbone architecture [5] which is faster than state-of-the-art.

Specifically, there are two stages in our pipeline. The first stage is PyraContrast’s unsupervised pre-training as a pretext task. This stage uses our unified architecture PyraNet on a rich ScanNetv2 [4] dataset. The second stage is supervised fine-tuning tasks (i.e. Semantic Segmentation, Instance Segmentation) on the target S3DIS [6] dataset. Instead of random initialization of weights, the already pre-trained weights are used and then improve the performance of the different downstream tasks. With this pipeline, our network leads to good performance in terms of time in downstream tasks.

3.2 Architecture Design

In the proposed pipeline as shown in figure 1 below, the first stage is unsupervised pretraining with *PyraContrast*.

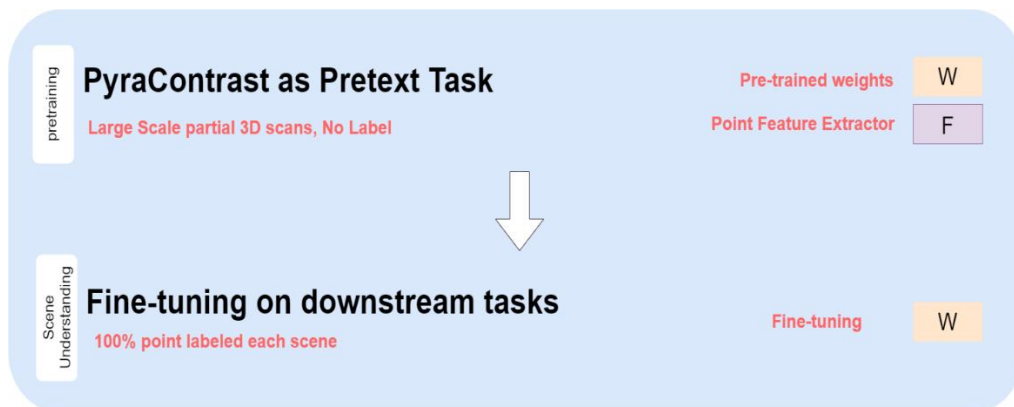


Figure 1 Structure of the pipeline. This is the pipeline of work. The pre-training stage is PyraContrast which will output the pre-trained weights. The second stage is scene understanding and it will use the pre-trained weights instead of random initialization for semantic segmentation and instance segmentation.

And the goal is to learn the weights and pre-trained networks. After the pre-training, different supervised downstream tasks are performed in which instead of random initialization of weights for the network for fine-tuning, the pre-trained weights \mathbf{W} are used.

3.2.1 PyraContrast as Pretext Task

A good pretext task, on the other hand, aims to learn the weights of the network that can be transferred universally and are beneficial on many high-level 3D scenes understanding tasks. In terms of *architecture*, there is a concern about inference speed in registration tasks. we used Pyramid Point Network which is lightweight and faster. In terms of the dataset, we used the ScanNetV2 [4], and finally, in terms of loss design, we used contrastive loss.

Algorithm 1- PyraContrast
<p>Input: Neural Network Architecture, $\mathbf{P} = \{p_i \in \mathbb{R}^{N \times 3}\}$ as Dataset, D as Point feature Dimension;</p> <p>Output: Neural Network Pre-trained weights</p> <p>for p in P do</p> <ul style="list-style-type: none"> • From p, creates a pair of views x^1 and x^2. • Compute the correspondence M between pairs x^1 and x^2 • Sample two transformations T_1 and T_2 • Compute point features $f^1, f^2 \in \mathbb{R}^{N \times D}$ by $f^1 = \text{NN}(T_1(x^1))$ and $f^2 = \text{NN}(T_2(x^2))$ • Convert the input to different partitions • Compute the contrastive loss $L_c(f^1, f^2)$ for each partition • Aggregate the total L_c • Backpropagation to update NN with aggregated contrastive loss $L_c(f^1, f^2)$ on positive/matched points <p>end</p>

Table 1 PyraContrast Algorithm. This is the algorithm of our pre-training stage which takes the raw point cloud as input and outputs the pre-trained weights.

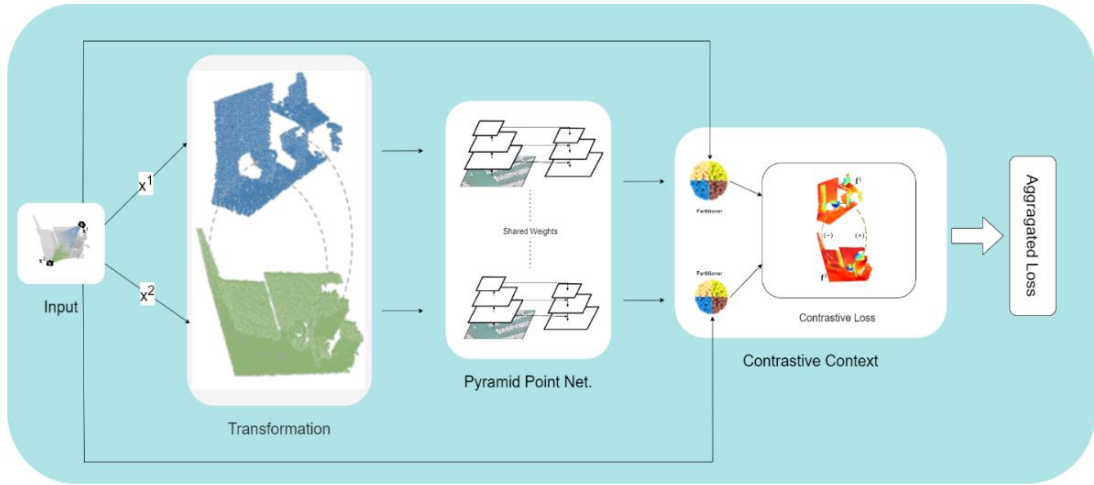


Figure 2 PyraContrast as a Pretext Task. this is the visualization of our pre-training stage. **Input:** takes the point cloud scene as input, and generates two views. **Transformation:** this module applies the transformation to two views. **Pyramid Point Network:** it is a feature extractor using a pyramid structure. **Contrastive Scene:** this will divide the point cloud into different partitions, calculate the contrastive loss for each partition and aggregate all the loss at the end.

3.2.1.1 Input Module:

The two views x^1 and x^2 are selected from a 3D scene in our ScanNetV2 [4]. So many pairs are generated from a single scene. The main goal of this block is to select only those pairs of views that are partially overlapping with each other.

3.2.1.2 Transformation Module:

After two views are generated, transformation needs to be done on the selected pair, so this module applies the rotation and translation transformation to them separately. The detail of transformations will be explained in Chapter 4 which is Implementation.

3.2.1.3 Pyramid Point Network (PPN) Module

This is the feature extraction module of our PyraContrast. This is our unified network which will be used both in unsupervised pre-training and supervised fine-tuning on different downstream tasks. Fig 3 illustrates our network architecture.

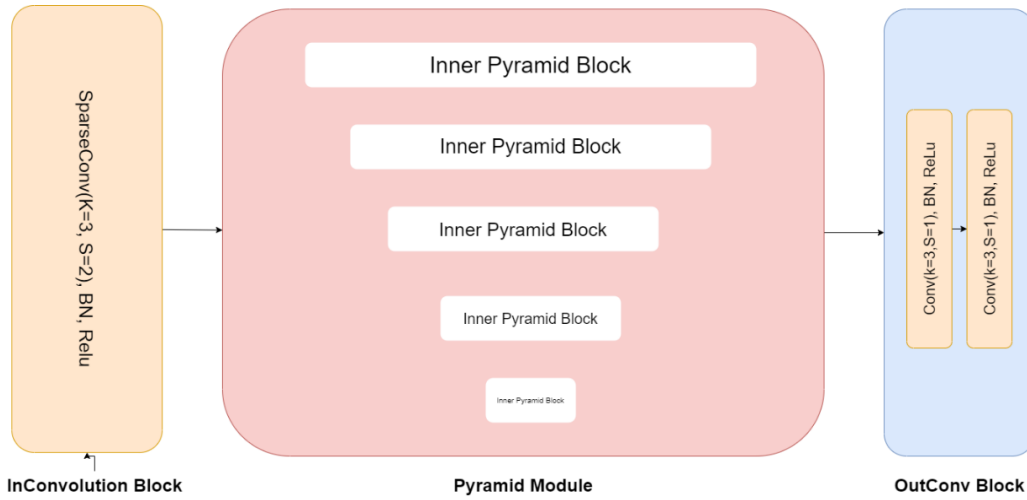


Figure 3 Pyramid Point Network (PPN). **InConvolution Block:** this will do the sparse convolution on our input. **PyramidModule:** this module will create many inner pyramid blocks that will allow the network to transfer features at many levels. **OutConv Block:** this will perform sparse convolution to get the required feature map dimension.

In feature extraction, the features are extracted densely and the most common representation of this kind of data are tensors, matrices, and vectors. But if we talk about the 3D data or higher dimensional features, we cannot use traditional representations due to the sparsity of 3D data. So, to solve this problem we can only use that part of space that is non-empty as its coordinated and associated features. Unlike the traditional sparse tensor, we will use the [11] sparse tensor. This tensor has the coordinates as (x, y, z) along with the batch indices to differentiate the point the points that will occupy the same coordinate in different batches. We can show a set of 4D coordinates as $C = \{(x_i, y_i, z_i, t_i)\}$ as a Matrix

C. There is also another Matrix F which has a set of associated features $F = \{f_i\}$. The whole sparse tensor can be represented as

$$C = \begin{bmatrix} x_1 & y_1 & z_1 & t_1 & b_1 \\ & & \vdots & & \\ x_N & y_N & z_N & t_N & b_N \end{bmatrix}, \quad F = \begin{bmatrix} f_1^T \\ \vdots \\ f_N^T \end{bmatrix}$$

In-Convolution Block: This layer will use the Sparse Convolution. After the transformation module, our inputs are fed to our Pyramid Point Network for feature extraction. The first block is the convolution block which is a sparse convolution.

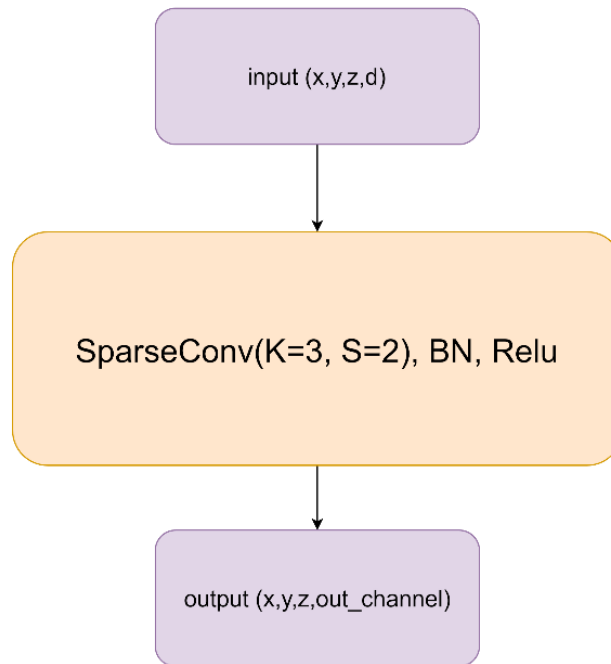


Figure 4 *In-Conv Block. This will apply only a single sparse convolution before feeding it to the Pyramid Module.*

This block will take the input of the point cloud to perform a sparse convolution and will create a required feature map. This feature map will be further used in

the Pyramid block. The detail of the In-Convolution block will be discussed in Chapter 4 which is Implementation.

Pyramid Module: This is our main block and this block has two main modules which are the encoder and decoder. The encoder will take the input and it will create a high-dimensional feature map by performing a sparse convolution. This will reduce the size of the input, but it will create the feature dimension. This feature dimension will further pass to the Inner Pyramid module. The Inner Pyramid Module is itself a pyramid Module and the network will further perform sparse convolution until we will reach the depth of the pyramid module.

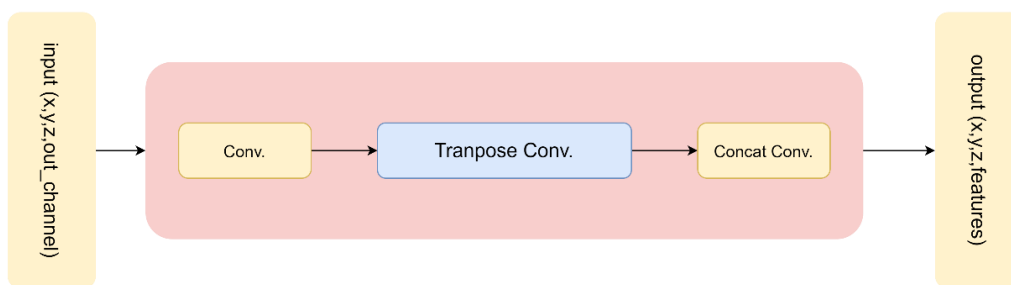


Figure 5 Inner Pyramid Module. The encoder will downsample the input but it will increase the feature dimension. The decoder will use the Transpose convolution to get the original size of input with a higher feature dimension map.

The output of each inner pyramid module is used as input for the next one. At the last depth of our pyramid block, the decoder will start it will first perform sparse transpose convolution and then add the lower resolution feature map will the high-resolution feature map for the current module. This will go up until we perform this for every pyramid module. The output of this whole Pyramid module will be used as input for our final Out-Convolution block.

Out-Convolution Block: This is the final block of our feature extraction. It has two sparse convolutions layers. The first sparse convolution layer will take

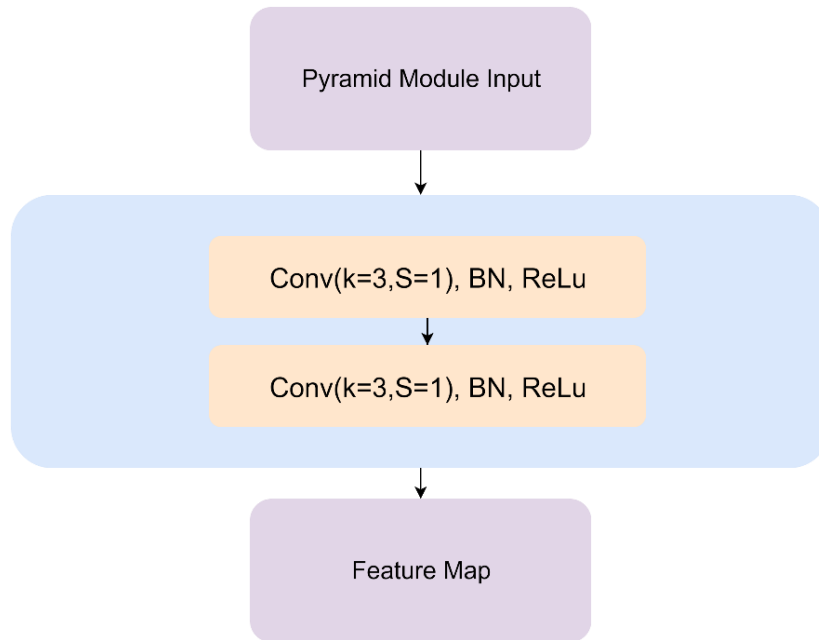


Figure 6 Out-Convolution Block. This is the final block of our feature extractor. This will take input from the previous block and perform two sparse convolutions for the final feature map.

the output of the whole pyramid module and perform convolution to generate the required feature map followed by Batch normalization and ReLU as activation function. The detail of this block will be also discussed in chapter 4 of Implementation. The output of the final convolution layer will be the out channels.

3.2.1.4 Contrastive Context Module

In the 3D point cloud, the spatial context such as relative pose, direction, and distance is crucial for difficult 3D tasks such as instance segmentation and our objective is to include this information in our pre-training. To achieve this goal, the idea came from the ShapeContext [24] local descriptor for shape matching. This module takes the input and divides that scene in a point cloud into various partitions. Thus, rather than having a single scene, now we can have multiple partitions of a scene. Before the contrastive loss was working for a single scene. there are multiple partitions now so rather than having a single contrastive loss

which is PointInfoNCE [3]. We perform the contrastive loss for each partition within a mini-batch separately for the whole point cloud sampled in a mini-batch. In the end, we aggregate our total loss.

Specifically for our work, we have two partial frame point clouds that are from the same scene and are x and y . The correspondence mapping is present which is $(i, j) \in M_{xy}$. Where the index i in frame x corresponds to the $x_i \in \mathbf{R}^3$ and the index j in frame y corresponds to a matched point $y_j \in \mathbf{R}^3$. we sample the positive points as matched, and the negative points as unmatched. The space is subdivided into multiple regions for each anchor point x_i , and other the points are assigned to distinct partition/regions. Based on their spatial information which includes relative angles along the distance to i , this allocation is made.

At the anchor point x_i , the following formula determines the distance along the angle information for the scene partition.

$$D_{ij} = \sqrt{\sum_{d=1}^3 (x_i^d - x_j^d)^2} \quad (1)$$

$$A_{ij} = \arctan2(D_{ij}) + 2\pi \quad (2)$$

In equation (1), the D is the matrix of the relative distance and the D_{ij} keeps the distance between the points i and j . The d represents the 3D dimensions. while in equation (2), the A_{ij} contains the relative angle between the points i and j . Now that we have the D and A , it is simple to construct a spatial partitioning function similar to so a ShapeContext- spatial.

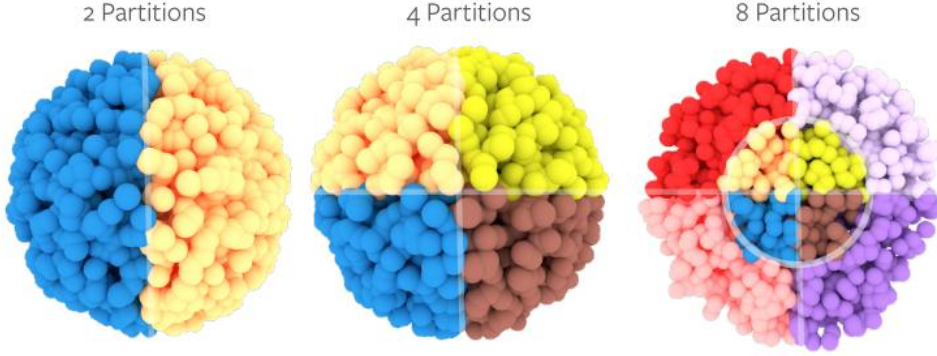


Figure 7 Visualization of Scene Context. This shows that a scene can be partitioned into different regions. It can be either 2 parts or 4 parts.

In Fig 7, we show a visualization of the scene context. This demonstrates how regions/partitions can be created from space. If we need to compute the 2 or 4 partitions for our space, cutting the space only needs to be done following the angle that is based on A . If there is a need for the cutting of space into more than eight partitions, then it's necessary to determine the size of inner regions using the D 's distance matrix. Space is always divided evenly along the distance and relative angles. *To the anchor point i* , keep in mind that the portioning is relative.

PointInfoNCE Loss: If we express the functions of spatial partition as $par_p(\cdot)$ and there are the P partitions/regions, where the $p \in \{1, \dots, P\}$. The anchor point i is the input for the function of the partition function $par_p(\cdot)$, its output of this function is a list of negative points. As there are many partitions so, for each partition, a PointInfoNCE loss L_p is independently calculated.

$$L_p = - \sum_{(i,j) \in M} \log \frac{\exp(f_j^1 \cdot f_j^2 / \tau)}{\sum_{(.,k) \in M, k \in par_p(i)} \exp(f_j^1 \cdot f_j^2 / \tau)} \quad (3)$$

Equation 3 specifics and other implementations will be covered in Chapter 4 on Implementation. Aggregating all partitions results in the determination of the final total loss $L = \frac{1}{|P|} \sum_p L_p$.

3.2.3 Fine-tuning on Downstream Tasks

Learning the features of the provided data is the most crucial and fundamental motivation for representation learning, and it can adapt well to various and unique downstream tasks. The learned representation usefulness can be measured by different evaluation methods. For instance, the study [25] uses a linear classifier to probe, and the [26] study evaluates a semi-supervised approach. The pre-trained weights that we obtain from the pretext task are used as the initialization of network weights in the *supervised fine-tuning* technique, then further fine-tuned on the intended downstream tasks. This approach is potentially the most useful way to assess transferability. With this method, we can no doubt get the performance gain in downstream tasks with the good, learned features.

So, in this stage, we will use our pre-train weights which we get from our PyraContrast as pretext task on multiple downstream tasks, and S3DIS [6] dataset as our intended dataset for fine-tuning. We discuss a variety of tasks of different natures including instance segmentation and semantic segmentation. Pyramid Point Network servers as the same backbone architecture for all our experiments. For the pre-training using the PointInfoNCE objectives on the proposed source ScanNetV2 pair dataset. In Chapter 5 which is titled Experiments and results, pre-training, semantic segmentation, and instance segmentation are further discussed in detail with results.

Chapter 4

Implementation

In this section of our thesis, we discuss the detail of our implementation for the unsupervised pre-training as well as for the fine-tuned downstream tasks.

5.1 Data Pre-processing

Following the PointContrast [2], Every 25 frames, we subsample ScanNetV2 [4] scene’s partial frame. We identify the pairs of frames in each scene by calculating the overlap.

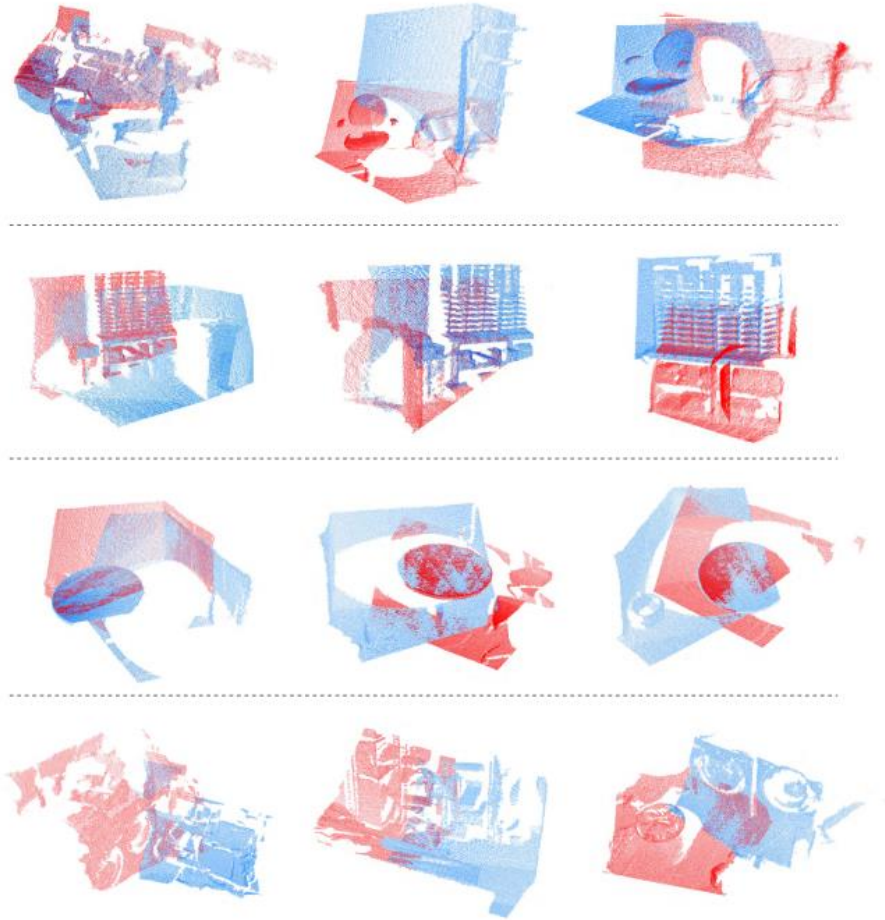


Figure 8 Visualization of ScanNetV2 Pair Dataset. Each row is randomly sampled from the scene. The first two columns show the pairs from that scene. The last column shows the two views in a single frame.

The overlap should be a minimum of 30%. As we dive further, each frame is converted to world coordinates. To calculate and determine how many points are overlapping, we iterate through each pair of the frame. The threshold of overlapping is 2.5cm.

For instance, if we find another point in frame B for each point in frame A that is within the 2.5 cm voxel size which is converted into a world coordinates system, then only those two points are considered to be a correspondence pair. When there is overlapping of 30% for points of 2 frames, those frames are saved for the training. The XYZ coordinates and RGB colors both are saved and used for unsupervised pre-training.

The visualization of ScanNetV2 pair data is shown in Fig 8 which is used for the pre-training. There are different pairs of point clouds in each column which are sampled from the same scene. Whereas the different color shows that two different views (partial scans) are corresponding to each other.

5.2 PyraContrast as Pretext Task

The implementation and detail of different modules which are used in PyraContrast are discussed below.

5.2.1 PointInfoNCE Loss

Here, we describe the PointInfoNCE [3] in detail and use it as our contrastive loss (Equation 3 of our thesis)

$$L_p = - \sum_{(i,j) \in M} \log \frac{\exp (f_j^1 \cdot f_j^2 / \tau)}{\sum_{(.,k) \in M, k \in \text{par}_p(i)} \exp (f_j^1 \cdot f_j^2 / \tau)}$$

In the above equation, M represents the set of all corresponding matches from the two frames which we discussed in data pre-processing. We denoted the point features from frame 1 as f^1 and frame 2 as f^2 . The points that have at least one match are used as negative points in the formula above. We eliminated any non-matching points. The point feature f_i^1 , acts as query, and the point feature f_j^2 acts positive key for the matched pair in $(i, j) \in M$. The point feature f_k^2 contains the sets of negative keys where $\exists (., k) \in M, k \in \text{par}_p(i)$ and we used $k \neq j$. A sample of a subset of match pairs from M is used for the training.

5.2.2 Transformations

We used the transformation on the partial views. In our experiments, on the two different views \mathbf{x}^1 and \mathbf{x}^2 , we applied the transformations \mathbf{T}_1 and \mathbf{T}_2 . These are the random rotation (0 to 360 °) along the arbitrary axis. On both views, these are applied independently. On both views, we also apply scale augmentation (0.8× to 1.2× of the input scale). We have noticed that with the other augmentation like point coordinates jittering, point dropout, and translation, there was no noticeable difference in fine-tuning performance.

5.2.3 Batch Normalization

Mostly deep neural networks consist of tens of layers and millions of neurons, and they are sensitive to the weights and configurations of the learning model. Batch normalization is a technique that is used in very deep neural networks like ours where we have millions of neurons. It standardizes the input to a layer of each mini-batch. So, it standardized the input to a network. By using this technique in all layers of the network except the last layer, it can accelerate the training, do some regularization, and overall, reduces the generalization error.

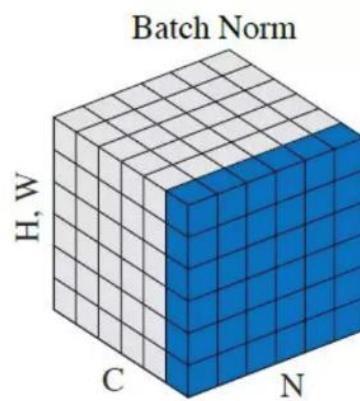


Figure 9 Visualization of Batch Normalization.

5.2.4 Activation Function

As we know our network consists of a lot of layers and neurons. But we must activate our neurons to work, and an activation function defines the output of weight or neuron is given the input. In simple words, they decide whether the network should activate the neuron or not.

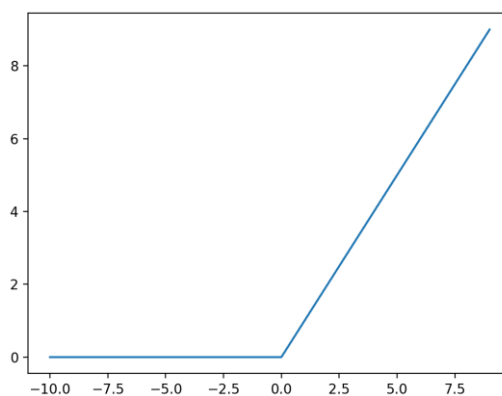


Figure 10 Rectified Linear Unit (ReLU). The graph represents that this linear function only outputs if the input is positive.

In our case, we use the Rectifies Linear Unit (ReLU) activation for our layers which is the most used one. Because it is easier to train and mostly gives better results. The formula for this activation is

$$A(x) = \max(0, x) \quad (4)$$

As written in the formula, it is a piecewise linear function that will take the input x and output it directly if it is a positive value. For the negative value of x , it will output as zero.

5.2.3 In-Convolution Block Implementation

For specifically our work we used the input channel of 3 for this layer and the output channel is 32 for both unsupervised pre-training and supervised downstream tasks. The kernel size is 3 and the dilation size is 1. We use the stride size of 3. After the convolution, we used the Batch Normalization and at last, we used the ReLU as the activation function.

5.2.4 Pyramid Module Implementation

For the sparse convolution used in the encoder and the inner pyramid modules, we use the input channel as provided by the previous layer and outputs defined for the select module. The kernel size we used is 3 along the stride of size 1 and dilation 1. Same Batch Normalization technique used for In-Conv and ReLU as activation function. But in the decoder, with a kernel size of three and dilation size of one, we carried out the sparse transposed convolution, but this time we used the stride size 2. For the concatenation of the inner feature and original feature dimension of that block, we also use the sparse convolution with the kernel size 1, stride size 1, and dilation size 1.

5.2.5 Out-Convolution Block Implementation

This is the last block of our feature extractor. The first layer of this block takes the input of the previous pyramid module and performs the sparse convolution with kernel size 3, stride size 1, and dilation 1. This layer uses the Batch normalization and ReLU activation function. The last layer of this block output the 32-feature dimension for the pre-training. For the semantic segmentation and instance segmentation, we use the feature map of dimensions according to their classes which is 13. The last layer doesn't use any Batch Normalization or activation function.

Chapter 5

Results and Discussion

5.1 Data

In this section, we utilized ScanNetv2 as a source dataset for our PyraContrast as Pretext task, aiming to address the scale issue. The RGB-D video dataset ScanNetV2 [4] has 2.5 Million views across more than 1500 scans and is annotated with surface reconstruction, instance-level semantic segmentation, and 3D camera poses. Right now, it is the biggest of its kind. There are 20 classes of annotated 3D objects in this dataset. We have evaluated our PyraContrast around 10 scenes which generate around 20k pairs, and this was sufficient for good pre-training.

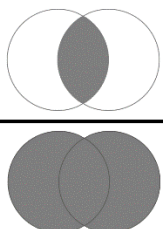
For the fine-tuning downstream tasks which are semantic segmentation and instance segmentation, we have used the target dataset of Stanford Large-Scale 3D indoor spaces S3DIS [6] for both semantic segmentation and instance segmentation for the benchmark. It consists of a 3D point cloud of 6 different areas consisting of 271 rooms. Each point cloud has a medium size room, whereas each area includes approximately 50 rooms. The number of points in all rooms ranges from half a million to two and half million. Each point has an annotation of a semantic label from 13 different classes. Further, each point has the 3D coordinates along with the RGB information. We have evaluated our downstream tasks in Area 5 which is 1-Fold validation. For benchmarking, we follow standard train/test splits.

5.2 Evaluation Metrics

To evaluate the performance of PyraContrast, we have used PointInfoNCE loss. We also evaluate the iteration time both for pre-training and downstream tasks because in the large-scale dataset training time also matters.

To evaluate the performance of downstream tasks semantic segmentation and instance semantic segmentation, we have used the Mean Intersection Over Union

(mIOU) strategy. Intersection Over Union is computed for the prediction score and the ground truth of the methodology.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


The mean value mIOU is calculated for the intersection of union for all classes. Due to the imbalance of different classes, we also used the mean Average Precision (mAP) and mean Average Class Accuracy (mAcc) of all classes. Hence, for all areas of the target S3DIS dataset, we have computed the mean intersection over union (mIOU) on the all-predicted results in Area 5. We choose Area 5 for cross-validation since the dataset we have used S3DIS, has a total of 6 areas.

5.3 Training Setting

We train our pipeline on a single Nvidia Titan X 12-GB GPU. Our batch size for the unsupervised pre-training is 6. The pre-training iteration depends on the given number of pairs. For the 26k pairs, our pre-training runs for 4200 iterations. The learning rate starts from 0.1 with a decay of $1e^{-6}$.

To fine-tune downstream tasks such as semantic segmentation, we used the same single GPU as pre-training. We used a batch size of 16 with a total number of iterations of 60,000. The learning rate starts from 0.1 with a decay of 0.0001.

For the instance segmentation, training is also done on a single GPU with a batch size of 16. There is a total of 15,000 iterations for instance segmentation with a learning rate of 0.1. the decay rate is 0.0001. we implemented the code in Python and PyTorch library in Linux.

5.4 Pre-training Results

We use the ScanNetV2 dataset for our pre-training and we compare it with two different methods [2][3]. The loss comparison is below.

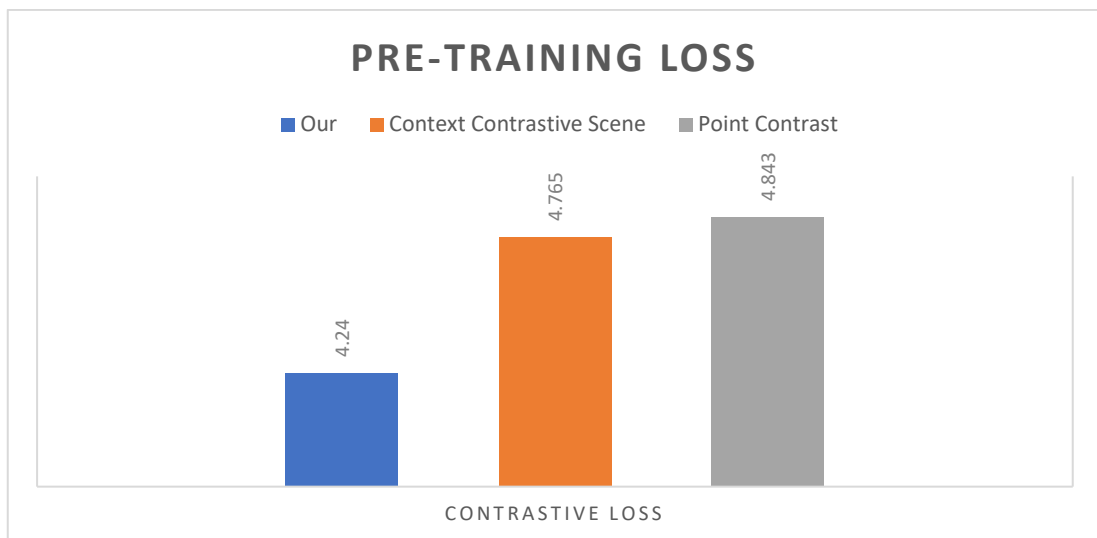


Table 2 Pre-training Loss. This is the semantic loss at the end of our pre-training stage.

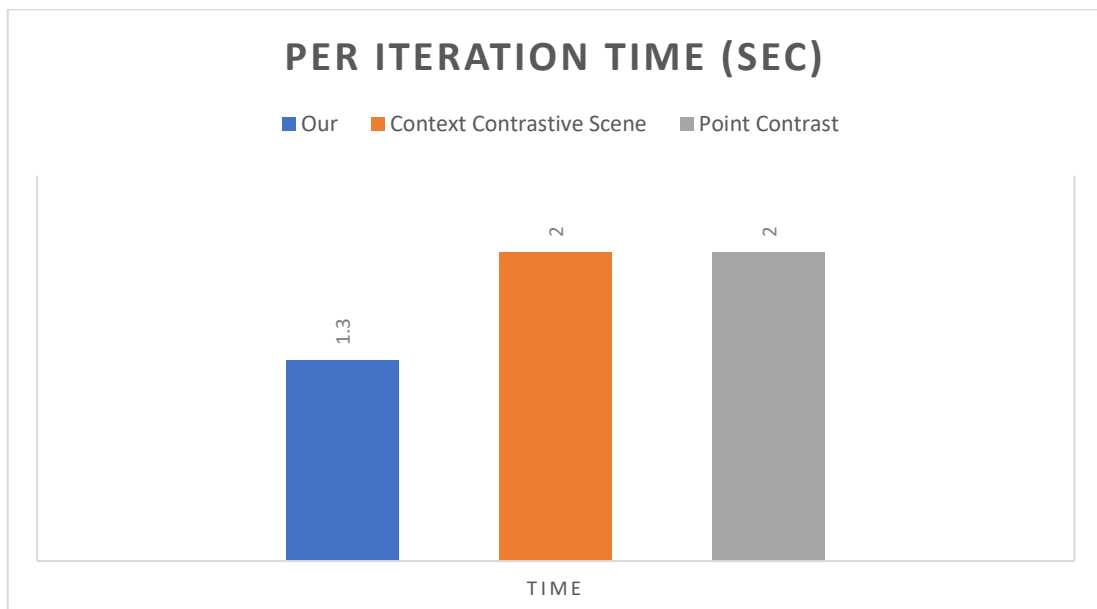


Table 3 Iteration Timing in Seconds. This shows that our model is performing better in terms of speed which is 1.3 sec per iteration.

As we can see from the results, our method is performing better in terms of loss and iteration time. Our method computes the loss in 1.3 seconds which is **35%** faster than both Contrastive Scene Context [3] and Point Contrast [2]. The less iteration time will help us to reduce the training time to half. In our work, it takes 4200 iterations to complete the pre-training. The computation is faster due to our unified Pyramid Point Network which uses a pyramid structure instead of a U structure. We already have discussed the shortcomings of the U structure which is max pooling and subsampling of feature layers. The pyramid structure does not have any residual block and it can view the feature at different levels.

5.5 Fine-tuning on Semantic Segmentation

The performance of our pipeline on the S3DIS dataset is compared with another state-of-the-art method [3]. We used the same training setting for both pipelines due to a lack of resources. We have quantitatively presented this comparison in table 4.

Method	mIOU	mAP	mAcc	Precision Score	Iteration Time
Contrastive Scene Context [3]	68.429	82.202	74.721	82.552	4.7(Sec)
Our	68.076	82.402	74.514	87.718 (+4.36)	2.2(Sec) (+53.1%)

Table 4 Semantic Segmentation on S3DIS. Our model is achieving the same result as the state-of-the-art method but 53% faster.

If we look at the results in Table 4 then in terms of time, our network is 53.1% faster in a single iteration which is a huge improvement in the training time, and

this is half of the training time as well as in validation compared to the state-of-the-art method [3]. We achieve the same results but faster. The detail of semantic segmentation class-wise is shown in Appendix. This improvement will save a lot of training time in the supervised fine-tuning semantic segmentation by using S3DIS [6] large-scale dataset, our pipeline performs better in Precision Score which is the ratio between true positive and true negative., We perform the validation on Area 5 which is the Fold 1 Test and if look at the per-category IOU performance, our method is performing better than most of the state-of-the-art methods.

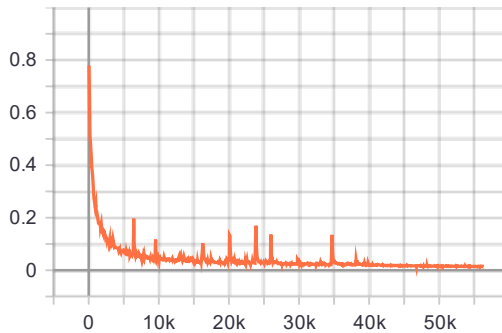


Figure 11 Semantic Loss. This is our semantic loss in training. This figure shows that the training loss curve gets smoother after the 40k iteration.

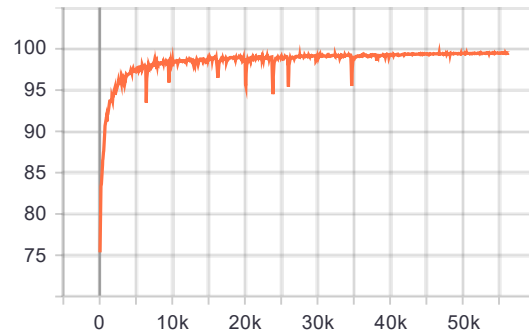


Figure 12 Precision score. This graph represents the precision score in training.

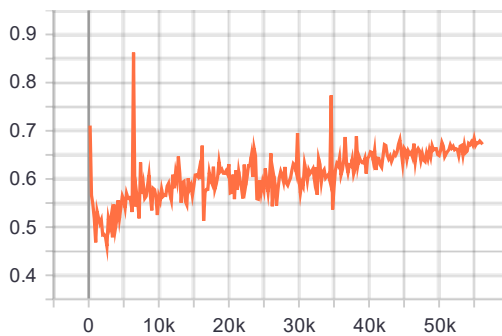


Figure 13 Validation Loss. This is our validation loss and this also gets smoother after 40k iterations

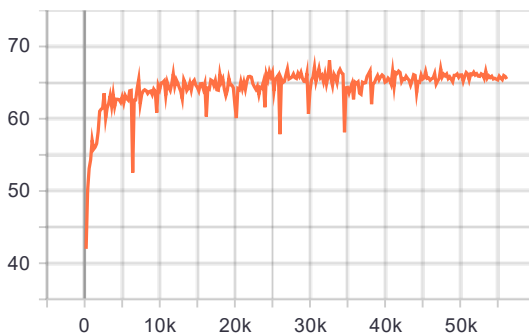


Figure 14 mIOU. This graph manifests the mIOU in validation.

The graphs of semantic loss, precision score, validation loss, and mIOU are below which shows that training needs a total of 60k iteration and at 40k the curve gets smooth. In the original paper on Contrastive Scene Context [3], they train their

model on 8 GPU with 48 batch size and they get the 72.2% mIOU. But due to a lack of hardware, we have to train on a single GPU with a 16-batch size so we can compare the efficiency of our work.

5.6 Fine-tuning on Instance Segmentation

We also perform instance segmentation in our work and the performance of our pipeline on the S3DIS dataset is compared with another state-of-the-art method [3]. We used the same training setting for both pipelines for comparison. We have quantitatively presented this comparison in table 6. In the instance segmentation, our method is once again time efficient and reduces the training time to more than half. The original paper on Contrastive Scene Context [3] achieves the 63.4% mAP50 using the 8 GPU and 48 batch size in instance segmentation. For the comparison, we train both models using a single GPU with 16 batch size. If we look at Figure 15, we can see that a total of 15k iterations are enough for the fine-tuning of instance segmentation.

Method	mIOU	mAP	mAcc	Precision Score	Iteration Time
Contrastive Scene Context [3]	69.461	84.912	76.444	89.886	4.4
Our	67.552	83.808	74.878	89.436	2.1 (+53.3%)

Table 5 Instance Segmentation on S3DIS Dataset. This table compares our method with state-of-the-art.

The average precision (AP) is also compared with the Contrastive Scene Context (CSC) [3] with ours. Our method is surprisingly time efficient in all cases. We

Method	AP	AP@50%	AP@25%
CSC[3]	0.457	0.589	0.679
Our	0.418	0.578	0.678

Table 6 Average Precision in Instance Segmentation on S3DIS [6] dataset.

performs the AP50 as well as AP25. The AP50 means that we are calculating the AP with the IoU threshold of 50%, and the mAP is the average of all AP.

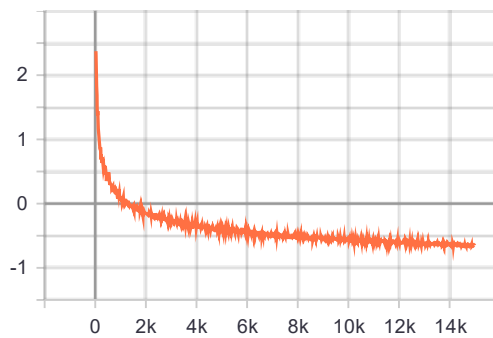


Figure 15 Training Loss. This is our training loss for instance segmentation and a total of 15k iterations are enough for good training.

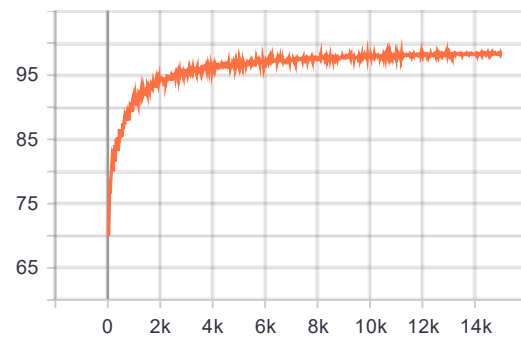


Figure 16 Precision Score

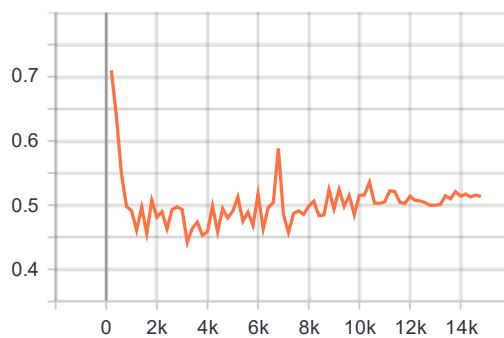


Figure 17 Validation Loss. This is the validation loss for a total of 15k iterations

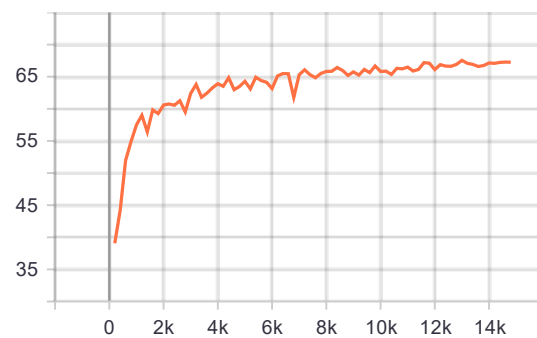


Figure 18 mIOU. This curve represents the mean Intersection over Union (mIOU).

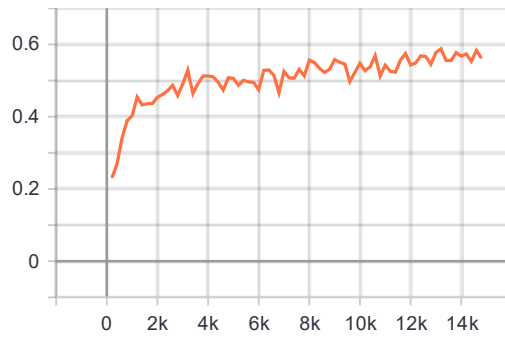


Figure 19 mAP 50 This represents the mAP50 of our model and which shows the AP value at 50% IoU threshold

Chapter 6

Conclusion & Future Work

This chapter summarizes the contribution of this research and discusses the possible future work for the unsupervised pre-training with the supervised downstream tasks.

6.1 Concluding Remarks

Unsupervised pre-training and supervised fine-tuning on 3D point clouds is still new and challenging due to the scale and annotation of the large 3D data. And this can be a real obstacle for transfer learning to boost performance. Few works have been done to tackle this problem. However, it is important that to boost the performance, good feature learning is important along with the spatial context of 3D data because the spatial context is important in segmentation tasks. we took a step forward and provide a pipeline and our method has successfully provided an architecture of unsupervised pre-training which encodes the spatial information too and further downstream fine-tuned tasks which are faster than the recent works. This implementation is better than many existing works. The results are provided in tabular form. There is also a comparison with similar approaches. The results shown in chapter 5 show that our pipeline has performed well and fast both in the ScanNetV2 [4] source dataset used in pre-training and the S3DIS [6] target dataset used in downstream tasks (semantic segmentation and instance segmentation). Our method almost reduced the pre-training time and fine-tuning on different downstream tasks to half achieving the same results.

6.2 Future Work Recommendations

In the future, this methodology can be improved and optimized by updating the Minkowski Engine [7] which is being used for sparse convolution further for the real-time application for faster 3D scene understanding without losing important scene information. Another possible future work is that can adopt efficient scene learning. As we know that annotating the point cloud is expensive work. So

instead of using all scenes, downstream tasks like instance segmentation and semantic segmentation can use a few points or few labels with the maximum performance. The existing real-world applications that are based on semantic segmentation and instance segmentation can benefit from this methodology to improve their performance.

References

- [1] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, no. M1m, pp. 4171–4186, 2019.
- [2] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, “PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12348 LNCS, pp. 574–591, 2020, doi: 10.1007/978-3-030-58580-8_34.
- [3] J. Hou, B. Graham, M. Nießner, and S. Xie, “Exploring data-efficient 3D scene understanding with contrastive scene contexts,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 15582–15592, 2021, doi: 10.1109/CVPR46437.2021.01533.
- [4] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “ScanNet: Richly-annotated 3D reconstructions of indoor scenes,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2432–2443, 2017, doi: 10.1109/CVPR.2017.261.
- [5] N. Varney, V. K. Asari, and Q. Graehling, “Pyramid Point: A Multi-Level Focusing Network for Revisiting Feature Layers,” 2020, [Online]. Available: <http://arxiv.org/abs/2011.08692>
- [6] I. Armeni *et al.*, “3D Semantic Parsing of Large-Scale Indoor Spaces (a) Raw Point Cloud (b) Space Parsing and Alignment in Canonical 3D Space (c) Building Element Detection,” pp. 1534–1543, 2016, doi: 10.1109/CVPR.2016.170.
- [7] C. Choy, J. Gwak, and S. Savarese, “Minkowski convolutional neural networks,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 3070–3079, 2019.
- [8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 77–85, 2017, doi: 10.1109/CVPR.2017.16.
- [9] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, pp. 5100–5109,

2017.

- [10] C. R. Qi, H. Su, M. Niebner, A. Dai, M. Yan, and L. J. Guibas, “Volumetric and multi-view CNNs for object classification on 3D data,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 5648–5656, 2016, doi: 10.1109/CVPR.2016.609.
- [11] C. Choy, J. Gwak, and S. Savarese, “4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 3070–3079, Apr. 2019, doi: 10.48550/arxiv.1904.08755.
- [12] B. Graham, M. Engelcke, and L. Van Der Maaten, “3D Semantic Segmentation with Submanifold Sparse Convolutional Networks,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 9224–9232, Nov. 2017, doi: 10.48550/arxiv.1711.10275.
- [13] L. Han, T. Zheng, L. Xu, and L. Fang, “OccuSeg: Occupancy-aware 3D Instance Segmentation,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2937–2946, Mar. 2020, doi: 10.48550/arxiv.2003.06537.
- [14] L. Jiang, H. Zhao, S. Shi, S. Liu, C. W. Fu, and J. Jia, “PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4866–4875, Apr. 2020, doi: 10.48550/arxiv.2004.01658.
- [15] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph Cnn for learning on point clouds,” *ACM Trans. Graph.*, vol. 38, no. 5, 2019, doi: 10.1145/3326362.
- [16] Y. Guo and X. Tong, “View-volume network for semantic scene completion from a single depth image,” *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2018-July, pp. 726–732, 2018, doi: 10.24963/ijcai.2018/101.
- [17] W. Shi and R. Rajkumar, “Point-GNN: Graph neural network for 3D object detection in a point cloud,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1708–1716, 2020, doi: 10.1109/CVPR42600.2020.00178.
- [18] E. Shelhamer, J. Long, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, 2017, doi: 10.1109/TPAMI.2016.2572683.

- [19] C. Choy, “Fully_Convolutional_Geometric_Features_ICCV_2019_paper.pdf,” *Iccv*, 2019.
- [20] A. van den Oord, Y. Li, and O. Vinyals, “Representation Learning with Contrastive Predictive Coding,” Jul. 2018, doi: 10.48550/arxiv.1807.03748.
- [21] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, “The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 3234–3243, Dec. 2016, doi: 10.1109/CVPR.2016.352.
- [22] L. Yi *et al.*, “A scalable active framework for region annotation in 3D shape collections,” *ACM Trans. Graph.*, vol. 35, no. 6, 2016, doi: 10.1145/2980179.2980238.
- [23] S. Song, S. P. Lichtenberg, and J. Xiao, “SUN RGB-D: A RGB-D scene understanding benchmark suite,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June-2015, pp. 567–576, Oct. 2015, doi: 10.1109/CVPR.2015.7298655.
- [24] S. Xie, S. Liu, Z. Chen, and Z. Tu, “Attentional ShapeContextNet for Point Cloud Recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4606–4615, 2018, doi: 10.1109/CVPR.2018.00484.
- [25] P. Goyal, D. Mahajan, A. Gupta, and I. Misra, “Scaling and benchmarking self-supervised visual representation learning,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-October, pp. 6390–6399, 2019, doi: 10.1109/ICCV.2019.00649.
- [26] O. J. Henaff *et al.*, “Data-Efficient Image Recognition with Contrastive Predictive Coding,” *37th Int. Conf. Mach. Learn. ICML 2020*, vol. PartF168147-6, pp. 4130–4140, May 2019, doi: 10.48550/arxiv.1905.09272.
- [27] S. Qiu, S. Anwar, and N. Barnes, “Semantic Segmentation for Real Point Cloud Scenes via Bilateral Augmentation and Adaptive Fusion,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1757–1767, 2021, doi: 10.1109/CVPR46437.2021.00180.

Appendix A

Appendix

Method	Clutter	Beam	board	bkcase	Ceiling	Chair	Clmn	Door	floor	sofa	Tbl	wall	windw	mIOU
PointNet [8]	33.22	0.05	26.38	40.28	88.80	52.6	3.92	10.7	97.3	5.85	58.93	69.8	46.26	41.09
PointCNN [16]	56.74	0.00	62.05	66.67	92.31	80.5	17.6	62.0	98.2	31.6	74.39	79.4	22.77	65.39
PC[2]	62.32	0.11	81.49	74.66	93.26	91.5	45.9	67.8	98.6	78.2	80.09	85.5	54.41	70.32
BAAFNet[27]	57.20	0.00	68.70	70.70	92.90	87.5	23.1	69.4	97.9	61.4	78.50	82.3	65.50	65.40
CSC [3]	58.70	0.00	81.09	72.33	94.38	88.0	27.3	76.8	98.6	73.2	81.31	85.3	52.31	68.42
Our	57.94	0.000	77.21	72.59	94.39	89.2	35.6	71.3	98.5	69.2	78.82	84.1	55.86	68.07

Table 7 *Stanford Area 5 Test (Fold 1). Per-category IOU performance over 13 classes*

Method	board	bkcase	Ceiling	Chair	Clmn	Door	floor	sofa	Table	wall	window	Avg
CSC	89.4	31.5	59.6	90.1	35.3	77.8	80.4	72.7	31.5	69.1	69.0	58.9
Our	82.4	33.9	53.0	84.2	37.3	73.7	92.8	63.6	37.1	60.9	74.2	57.8

Table 8 *Instance Segmentation on Stanford Area 5 Test [6] mAP@0.5 over 11 classes.*