MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE GWALIOR

(A Govt. Aided UGC Autonomous Institute Affiliated to RGPV, Bhopal)

NAAC Accredited with A++ Grade

Project Report

on

Unsupervised Learning on Point Clouds

Submitted By:

Dhruve Kiyawat

(0901AD211011)

Faculty Mentor:

Prof. Arun Kumar

Assistant Professor, Centre for Artificial Intelligence

CENTRE FOR ARTIFICIAL INTELLIGENCE

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE

GWALIOR - 474005 (MP) est. 1957

July-Dec 2023

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE GWALIOR

(A Govt. Aided UGC Autonomous Institute Affiliated to RGPV, Bhopal)

NAAC Accredited with A++ Grade

CERTIFICATE

This is certified that **Dhruve Kiyawat** (0901AD211011) has submitted the project report titled **Unsupervised Learning on Point Clouds** under the mentorship of **Prof. Arun Kumar**, in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in **Artificial Intelligence and Data Science** from Madhav Institute of Technology and Science, Gwalior.

23 Nov 2023

Prof. Arun Kumar
Faculty Mentor
Assistant Professor
Centre for Artificial Intelligence

23/11/23

Dr. R. R. Singh
Coordinator
Centre for Artificial Intelligence

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE GWALIOR

(A Govt. Aided UGC Autonomous Institute Affiliated to RGPV, Bhopal)

NAAC Accredited with A++ Grade

DECLARATION

I hereby declare that the work being presented in this project report, for the partial fulfilment of requirement for the award of the degree of Bachelor of Technology in **Artificial Intelligence and Data Science** at Madhav Institute of Technology & Science, Gwalior is an authenticated and original record of my work under the mentorship of **Prof. Arun Kumar, Assistant Professor**, Centre for Artificial Intelligence

I declare that I have not submitted the matter embodied in this report for the award of any degree or diploma anywhere else.

Dhruve Kiyawat
0901AD211011
3rd Year
Centre for Artificial Intelligence

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE GWALIOR

(A Govt. Aided UGC Autonomous Institute Affiliated to RGPV, Bhopal)

NAAC Accredited with A++ Grade

ACKNOWLEDGEMENT

The full semester project has proved to be pivotal to my career. I am thankful to my institute, **Madhav Institute of Technology and Science** to allow me to continue my disciplinary/interdisciplinary project as a curriculum requirement, under the provisions of the Flexible Curriculum Scheme (based on the AICTE Model Curriculum 2018), approved by the Academic Council of the institute. I extend my gratitude to the Director of the institute, **Dr. R. K. Pandit** and Dean Academics, **Dr. Manjaree Pandit** for this.

I would sincerely like to thank my department, **Centre for Artificial Intelligence,** for allowing me to explore this project. I humbly thank **Dr. R. R. Singh**, Coordinator, Centre for Artificial Intelligence, for his continued support during the course of this engagement, which eased the process and formalities involved.

I am sincerely thankful to my faculty mentors. I am grateful to the guidance of **Arun Kumar**, Assistant Professor, Centre for Artificial Intelligence, for his continued support and guidance throughout the project. I am also very thankful to the faculty and staff of the department.

Dhruve Kiyawat
0901AD211011
3rd Year
Centre for Artificial Intelligence

ABSTRACT

Though a number of point cloud learning methods have been proposed to handle unordered points, most of them are supervised and require labels for training. By contrast, unsupervised learning of point cloud data has received much less attention to date. In this paper, we propose a simple yet effective approach for unsupervised point cloud learning. In particular, we identify a very useful transformation which generates a good contrastive version of an original point cloud. They make up a pair. After going through a shared encoder and a shared head network, the consistency between the output representations are maximized with introducing two variants of contrastive losses to respectively facilitate downstream classification and segmentation. To demonstrate the efficacy of our method, we conduct experiments on three downstream tasks which are 3D object classification (on ModelNet40 and ModelNet10), shape part segmentation (on ShapeNet Part dataset) as well as scene segmentation (on S3DIS). Comprehensive results show that our unsupervised contrastive representation learning enables impressive outcomes in object classification and semantic segmentation. It generally outperforms current unsupervised methods, and even achieves comparable performance to supervised methods.

Keyword: unsupervised contrastive learning; point cloud; 3D object classification; semantic segmentation

सार:

यद्यपि अनुक्रमित बिंदुओं का सामान्यत: सुपरवाइज्ड तरीकों से सामना करने के लिए कई पॉइंट क्लाउड लर्निंग विधियां प्रस्तुत की गई हैं, उनमें से अधिकांश प्रशिक्षण के लिए प्रतीक्षा करती हैं। उसके बावजूद, अदृश्य बिंदु डेटा का अनुप्रयोग करने के लिए अनुप्रशिक्षित पॉइंट क्लाउड लर्निंग को आज तक कम ध्यान मिला है। इस लेख में, हम एक सरल लेकिन प्रभावी दृष्टिकोण का प्रस्तुतन करते हैं जो किसी मूल पॉइंट क्लाउड का एक अच्छा बहुलक्ष परिवर्तन उत्पन्न करता है। वे एक जोड़ी बनाते हैं। एक साझा एन्कोडर और साझा हेड नेटवर्क के माध्यम से जाने के बाद, आउटपुट प्रतिष्ठानों के बीच संघटन को अधिकतम किया जाता है, जिसमें नीचे वर्णित दो प्रकार के बहुलक्ष हानियां परिचित कराई जाती हैं, जिनमें नीचे वर्णित तीन डाउनस्ट्रीम कार्यों के लिए एक्सपीरिमेंट किया जाता है, जो कि 3डी ऑब्जेक्ट क्लासिफिकेशन (मॉडलनेट 40 और मॉडलनेट 10 पर), शेप पार्ट सेगमेंटेशन (शेपनेट पार्ट डेटासेट पर) साथ ही सीन सेगमेंटेशन (एस 3 डी आइएस पर)। सर्वांगीण परिणाम दिखाते हैं कि हमारी अनुप्रशिक्षित बहुलक्ष प्रतिस्थापन शिक्षा अनुक्रमण में अद्भुत परिणाम दिखाती है। यह सामान्यत: वर्तमान अनुप्रशिक्षित विधियों को पीछे छोड़ती है, और सुपरवाइज्ड विधियों के समतुल्य प्रदर्शन करती है।

TABLE OF CONTENTS

Chapter 1: INTRODUCTION

1.1 Introduction:

In this research, we focus on leveraging 3D point clouds for classification and segmentation tasks, crucial in fields like multimedia computing and robotics. While many existing methods for 3D point cloud analysis rely on annotated data for training, the annotation process is time-consuming and costly. Unsupervised learning offers a viable alternative.

Several approaches, such as Latent-GAN, FoldingNet, MAP-VAE, and 3D-PointCapsNet, have employed Autoencoders (AE) as the backbone for unsupervised learning. However, these methods often suffer from less quality representations, impacting downstream tasks like classification and segmentation.

Motivated by these challenges, we propose an unsupervised representation learning method aiming to enhance 3D object classification and semantic segmentation. The core idea is to maximize agreement or consistency between the representations of the original point cloud and its transformed version (contrastive version).

Our approach involves using a single transformation to generate the transformed point cloud version, pairing it with the original, and feeding them into a shared base encoder network (e.g., former part of PointNet with global features). A subsequent projection head network (e.g., latter part of PointNet with several MLP layers) imposes agreement maximization on its outputs. This enhances training efficiency and preserves rich representations from the encoder. Since there are no labels involved in training, it constitutes unsupervised representation learning for 3D point cloud data.

To validate our method, we conducted experiments on object classification tasks using ModelNet40 and ModelNet10, shape part segmentation tasks on ShapeNet Part dataset, and scene segmentation tasks on the S3DIS dataset. Results demonstrate that our unsupervised contrastive representative learning achieves impressive outcomes across these tasks, outperforming state-of-the-art unsupervised techniques and even being comparable to certain supervised counterparts.

- **Plotly** : Plotly is a versatile Python library for creating interactive and visually appealing plots. It supports a wide range of chart types and is particularly useful for data visualization in scientific computing, exploratory data analysis, and interactive dashboards.

- **Seaborn** : Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for creating aesthetically pleasing and informative statistical graphics. Seaborn simplifies the process of generating complex visualizations with concise code.

- **Pandas** : Pandas is a powerful data manipulation and analysis library for Python. It introduces data structures like DataFrames and Series, making it efficient to work with structured data. Pandas is widely used in data cleaning, exploration, and preparation tasks.

- **Matplotlib** : Matplotlib is a comprehensive 2D plotting library for Python. It enables the creation of static, animated, and interactive visualizations. Matplotlib is a foundational tool for data visualization and is extensively used in scientific computing and data analysis.

- **Sklearn** : scikit-learn is a machine learning library for Python that provides a simple and efficient toolkit for data analysis and modeling. It includes a variety of machine learning algorithms, tools for preprocessing data, and utilities for model evaluation and selection.

- **Tqdm** : It is a Python library that adds progress bars to loops and other iterable computations. It provides a visual representation of the progress, making it easier to monitor the execution of tasks and estimate completion times, especially in long-running processes.

1.3 Purpose and Scope:

Purpose:

This project aims to tackle the challenges associated with 3D point cloud analysis, focusing specifically on the tasks of 3D object classification and semantic segmentation. The central purpose is to develop and implement a novel unsupervised representation learning method tailored for 3D point clouds. The motivation for this project arises from the considerable time and cost involved in annotating data for training purposes, making unsupervised learning an attractive alternative. The goal is to significantly enhance the downstream tasks by maximizing the agreement between original point clouds and their transformed counterparts.

Scope:

The scope of this project is multifaceted, spanning the development, implementation, and validation of a novel unsupervised representation learning method. The method is designed to improve the efficiency and quality of training for 3D object classification and semantic segmentation tasks. The approach involves utilizing a shared base encoder network, specifically adopting the former part of the PointNet architecture with global features, and a subsequent projection head network, resembling the latter part of PointNet with multiple MLP layers.

The experimental validation encompasses diverse datasets to ensure a comprehensive evaluation. The datasets include ModelNet40 and ModelNet10 for object classification, the ShapeNet Part dataset for shape part segmentation, and the S3DIS dataset for scene segmentation. The effectiveness of the proposed method is assessed against state-of-the-art unsupervised techniques, and intriguingly, its performance is even compared favorably with certain supervised counterparts.

The project not only aims to contribute to the academic understanding of unsupervised representation learning for 3D point clouds but also strives to showcase practical applicability. By demonstrating superior outcomes in terms of 3D object classification and semantic segmentation, the proposed method has the potential to influence fields such as multimedia computing and robotics, where accurate and efficient 3D data analysis is paramount.

Chapter 2 : Dataset (ModelNet10)

2.1 Dataset Description:

ModelNet10 is a widely used benchmark dataset in the field of computer vision and 3D object recognition. Developed by researchers at Princeton University, it serves as a valuable resource for evaluating algorithms and models designed to recognize and classify 3D objects. The dataset encompasses a diverse collection of 3D CAD models, representing objects from ten distinct categories: bathtubs, beds, chairs, desks, dressers, monitors, nightstands, sofas, tables, and toilets.

Each category in ModelNet10 comprises a substantial number of 3D models, contributing to the dataset's richness and complexity. The objects within these categories vary significantly in shape, size, and structure, presenting a realistic and challenging set of examples for training and testing 3D object recognition systems. The dataset provides a balance between common everyday items, such as chairs and tables, and more complex objects like sofas and bathtubs.

The 3D models in ModelNet10 are annotated with category labels, enabling supervised learning approaches for training machine learning models. This facilitates the development and evaluation of algorithms that can generalize across diverse 3D shapes and accurately classify objects into their respective categories.

Researchers and practitioners often use ModelNet10 to assess the robustness and generalization capabilities of 3D object recognition models, making it a fundamental resource for advancing the state-of-the-art in this domain. The dataset's widespread adoption in the research community has contributed to the development of more effective algorithms for applications ranging from robotics and augmented reality to computer-aided design and manufacturing.

2.2 Dataset Visualization:

These visualization methods are crucial for researchers and practitioners working with 3D data, such as those dealing with the ModelNet10 dataset. Visualizing the dataset helps in gaining insights into

the geometric properties of objects, identifying patterns, and verifying the integrity of the data. The dataset consists of 3D CAD models representing various object categories. Here's an explanation of the visualization methods:

```
def read_off(file_path):
    with open(file_path, 'r') as file:
        if 'OFF' != file.readline().strip():
            raise('Not a valid OFF header')
        n_verts, n_faces, __ = tuple([int(s) for s in file.readline().strip().split(' ')])
        verts = np.array([[float(s) for s in file.readline().strip().split(' ')] for i_vert in range(n_verts)])
        faces = np.array([[int(s) for s in file.readline().strip().split(' ')][1:] for i_face in range(n_faces)])
        return verts, faces

file_path = r"/content/ModelNet10/table/test/table_0301.off"
mesh = read_off(file_path)
```

Reading OFF Files : The read_off function reads a given OFF file, a standard file format for representing 3D geometric objects. It starts by checking if the file has a valid OFF header. If not, it raises an exception. The function then extracts the number of vertices (n_verts), faces (n_faces), and reads the corresponding vertex and face information from the file. The result is a tuple containing the vertex and face arrays.

```
def visualize_mesh(mesh):
    verts, faces = mesh
    import plotly.graph_objs as go
    # Create a 3D mesh plot
    mesh = go.Mesh3d(x = verts[:, 0], y = verts[:, 1], z = verts[:, 2],
                     i = faces[:, 0], j = faces[:, 1], k = faces[:, 2], opacity=0.5)
    layout = go.Layout(scene=dict(aspectmode="data"))
    fig = go.Figure(data=[mesh], layout=layout)
    fig.show()
```

Mesh Visualization : The visualize_mesh function takes a mesh (vertices and faces) as input and uses the Plotly library to create an interactive 3D mesh plot. It represents the 3D object by connecting vertices based on the face information. The resulting visualization provides a clear representation of the object's structure, allowing for an interactive exploration of its shape and form.

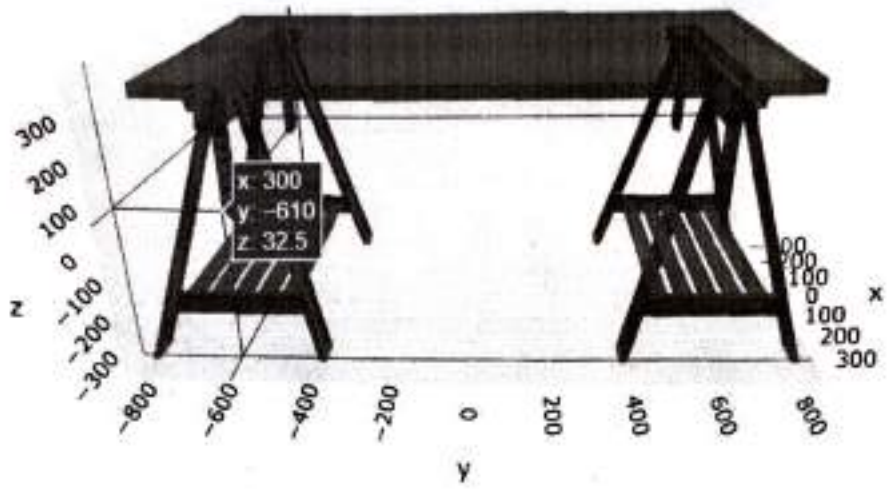


Figure : Visualization of a 3D Mesh

```
def visualize_pointcloud(pointcloud):
    import plotly.graph_objects as go
    pointcloud = go.Scatter3d(x = pointcloud[:, 0], y = pointcloud[:, 1], z = pointcloud[:, 2],
                              mode='markers', marker=dict(size= 1, opacity=0.5))
    layout = go.Layout(scene=dict(aspectmode="data"))
    fig = go.Figure(data=[pointcloud], layout = layout)
    fig.show()
```

Point Cloud Visualization : The visualize_pointcloud function is designed to visualize a 3D point cloud. It takes a point cloud array as input and uses Plotly to create a 3D scatter plot. Each point in the cloud is represented as a marker, and the overall visualization provides an overview of the spatial distribution of points in 3D space. This is particularly useful for understanding the point cloud's structure, density, and spatial arrangement.

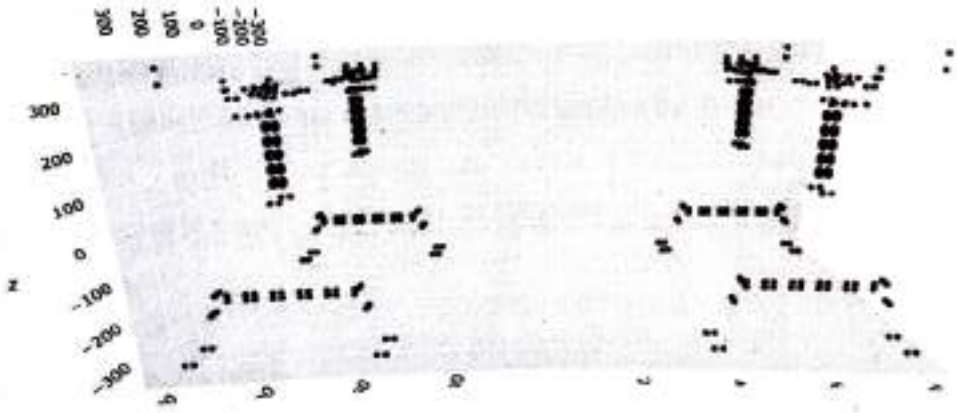


Figure : Visualization of 3D point cloud.

2.3 Data Fetching:

```
import torch
from torch.utils.data import Dataset, DataLoader
from torch_geometric.data import Data
import glob

class CustomPointCloudDataset(Dataset):
    def __init__(self, file_paths, transform=None):
        self.file_paths = file_paths
        self.transform = transform

    def __len__(self):
        return len(self.file_paths)

    def __getitem__(self, idx):
        file_path = self.file_paths[idx]
        mesh = read_off(file_path)
        pointcloud = mesh_to_pointcloud(mesh, 3000)
        pointcloud = scale_pointcloud(pointcloud)
        aug_pointcloud = rotate_pointcloud(pointcloud)
        if self.transform:
            pointcloud = self.transform(pointcloud)
        return torch.tensor((pointcloud, aug_pointcloud))

batch_size = 8

file_paths = glob.glob("/content/ModelNet10/**/train/**.off")
file_paths = file_paths[:100]
truncated_length = (len(file_paths) // batch_size) * batch_size #dataset_size // batch_size
file_paths = file_paths[:truncated_length]


dataset = CustomPointCloudDataset(file_paths, transform=None)
dataloader = DataLoader(dataset, batch_size=batch_size, shuffle=True)
#dataset -> ((pointcloud, aug_pointcloud), batch_size, n_points, coordinates)
```

Dataset Class (CustomPointCloudDataset):

- The dataset is designed to handle 3D object representations from the ModelNet10 dataset, stored in the OFF file format.
- The __init__ method initializes the dataset with a list of file paths and an optional data transformation function.
- The __len__ method returns the total number of samples in the dataset.
- The __getitem__ method loads a single data sample at the given index. It reads the OFF file, converts the mesh to a point cloud, scales and augments the point cloud, and applies an optional transformation.

Data Loading and Processing:

- The code collects file paths for training OFF files in the ModelNet10 dataset using glob.glob.
- The dataset is instantiated with the collected file paths and a None transform.

- The DataLoader is initialized with the dataset, specifying a batch size of 8 and enabling data shuffling during training.

Expected Output:

- The DataLoader produces batches, each containing a pair of point clouds: the original and its augmented version.
- The output tensor has dimensions (batch_size, 2, n_points, coordinates), representing batches of point clouds. n_points is the number of points in the cloud, and coordinates represent spatial coordinates (e.g., x, y, z).

The code establishes a pipeline for creating batches of point cloud data from ModelNet10, crucial for training neural networks. The dataset class encapsulates the logic for reading, processing, and augmenting point clouds, while the DataLoader efficiently batches this data for training, streamlining the integration of point cloud data into PyTorch-based deep learning models.

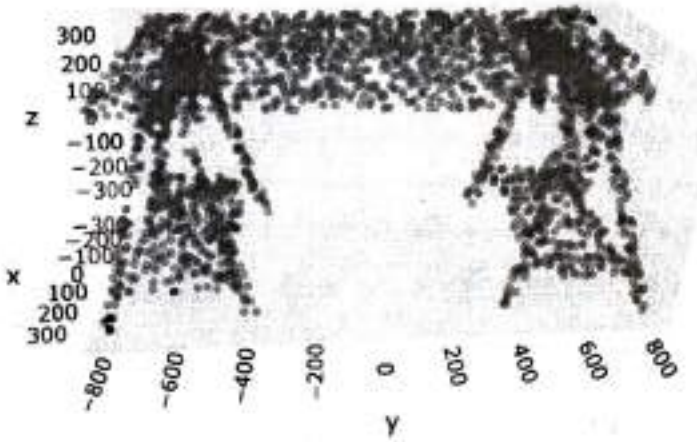


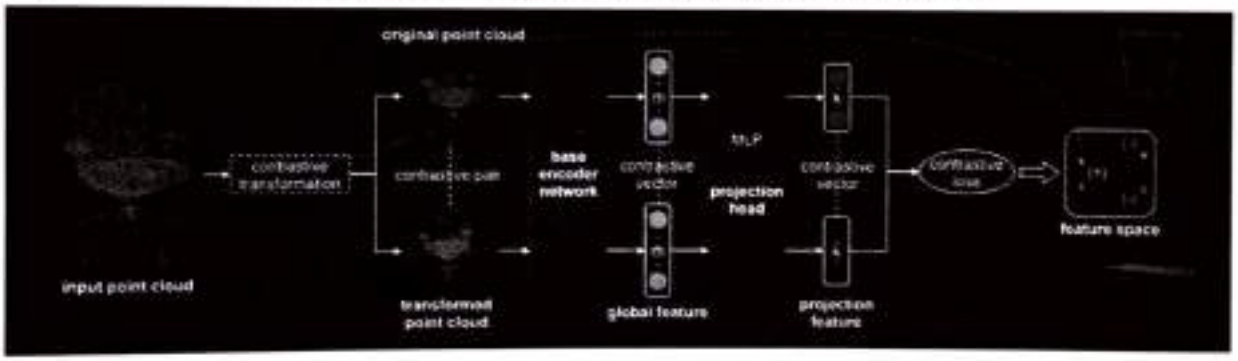
Figure : Visualization of a Sampled PointCloud

Chapter 3 : Methodology

3.1 Proposed Methodology:

Contrastive Transformation:

In contrast to 2D images, point cloud data is characterized by an irregular distribution in 3D space, presenting a complex degree of freedom. Constructing effective contrastive pairs for point clouds is challenging due to these unique characteristics. While SimCLR utilizes various transformations for a single image (e.g., cropping, rotation) to generate two transformed versions, we opt for simplicity. We employ a single transformation, pairing the original point cloud with its transformed counterpart. Common transformations in 3D space, such as rotation, cutout, crop, scaling, and smoothing, are considered for data augmentation. Heavy noise corruption, which can distort object shapes, is excluded from the transformations. Jittering, akin to light noise, is used for data augmentation following the protocol of state-of-the-art point-based methods.



Base Encoder Network:

Point-based networks like PointNet, DGCNN, and Pointfilter often incorporate a pooling layer to output global features for an input point cloud. The portion of a point-based network preceding this layer (inclusive) naturally serves as a base encoder in our framework, encoding the input point cloud into a latent representation vector (i.e., the global feature). In this work, we select state-of-the-art point-based networks, such as PointNet and DGCNN, and extract their former parts as our base encoder. Interestingly, encoders involving T-Net (transformation net) hinder the learning of unsupervised contrastive representations. We deduce that T-Net introduces various rotated point cloud augmentations, degrading the ability to capture a significant contrast between input pairs.

Projection Head Network:

Point-based networks typically incorporate fully connected layers to connect the global feature with the final k-class vector. Similar to the encoder, we can straightforwardly extract the latter part of a point-based network as the projection head. Alternatively, it is possible to customize a projection head network by designing more or fewer fully connected layers.

3.2 Architecture Implementation:

```
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.encoder1 = BaseEncoder()
        self.encoder2 = BaseEncoder()
        self.head1 = ProjectionHead()
        self.head2 = ProjectionHead()

    def forward(self, input):
        pcd1, pcd2 = input[0].permute(0, 2, 1).float(), input[1].permute(0, 2, 1).float()
        feature1 = self.encoder1(pcd1)
        feature2 = self.encoder2(pcd2)

        z1 = self.head1(feature1)
        z2 = self.head2(feature2)

        return feature1, feature2, z1, z2
```

BaseEncoder:

- *Purpose*: This component serves as the base encoder network responsible for extracting features from the input point clouds.
- *Architecture*:
- Three 1D convolutional layers (conv1, conv2, conv3) process the input point cloud data with increasing channel dimensions (3 channels to 64, 64 to 128, and 128 to 1024).
- Batch normalization is applied after each convolutional layer (bn1, bn2, bn3) to stabilize and normalize the activations.
- ReLU activation functions are used to introduce non-linearity after each batch normalization.
- The final global feature is obtained by applying max-pooling across the last dimension and flattening the result

ProjectionHead:

- *Purpose*: This component further refines the features obtained from the base encoder into a lower-dimensional representation.

- *Architecture*:
- Three fully connected layers (fc1, fc2, fc3) reduce the dimensionality of the input features from 1024 to the desired output dimension (default is 10 classes).
- Batch normalization (bn1, bn2) is applied after the first two fully connected layers.
- ReLU activation functions introduce non-linearity after the first two fully connected layers.
- Dropout is applied with a probability of 0.3 after the second fully connected layer.

Model:

- *Purpose*: The Model class integrates the base encoder and projection head to create a complete model for unsupervised contrastive learning.

- *Architecture*:
- Two instances of the BaseEncoder (encoder1 and encoder2) process two input point clouds independently.
- Two instances of the ProjectionHead (head1 and head2) refine the features obtained from the respective encoders.
- The forward method takes a pair of input point clouds, processes them through their respective encoders, refines the features with the projection heads, and returns the resulting features (feature1 and feature2) along with the final outputs (z1 and z2).

3.3 Underlying Transformation and Loss:

Transformation

- **normalize_pointcloud** : This operation centers the point cloud around its mean and scales it, ensuring that the distribution of points is uniform. The normalization enhances the stability and convergence of learning algorithms. By centering the cloud, it helps mitigate biases in the data, making the model less sensitive to initial conditions.

- **rotx_pointcloud** : Rotates the point cloud around the x-axis by a randomly generated angle. Introducing variations in the x-axis orientation ensures the model learns to recognize objects from different perspectives. This rotation augmentation promotes diversity in the dataset, enhancing the model's ability to generalize to unseen orientations.

- **roty_pointcloud** : Rotates the point cloud around the y-axis by a randomly generated angle. Similar to 'rotx_pointcloud', this transformation introduces diversity in the y-axis orientation, expanding the range of perspectives the model encounters during training.

- **rotz_pointcloud** : Rotates the point cloud around the z-axis by a randomly generated angle. Introducing diversity in the z-axis orientation ensures comprehensive coverage of rotational variations, contributing to the overall transformation diversity in the dataset.

- **noisy_pointcloud** : Adds random noise drawn from a normal distribution to each point in the cloud. Simulating real-world noise conditions, this transformation improves the model's robustness to noisy input, helping it generalize better in environments with inherent data imperfections.

- **scale_pointcloud** : Combines normalization and noise addition by first normalizing and then adding noise to the point cloud. This transformation offers a balanced approach to data augmentation, normalizing the point cloud while introducing random perturbations through noise. It ensures a comprehensive yet controlled variation in the dataset.

- **rotate_pointcloud** : Sequentially applies rotations around all three axes (x, y, z). Generating diverse rotations allows the model to learn invariant features under various orientations. This

comprehensive transformation enriches the dataset, enabling the model to handle a wide range of spatial configurations.

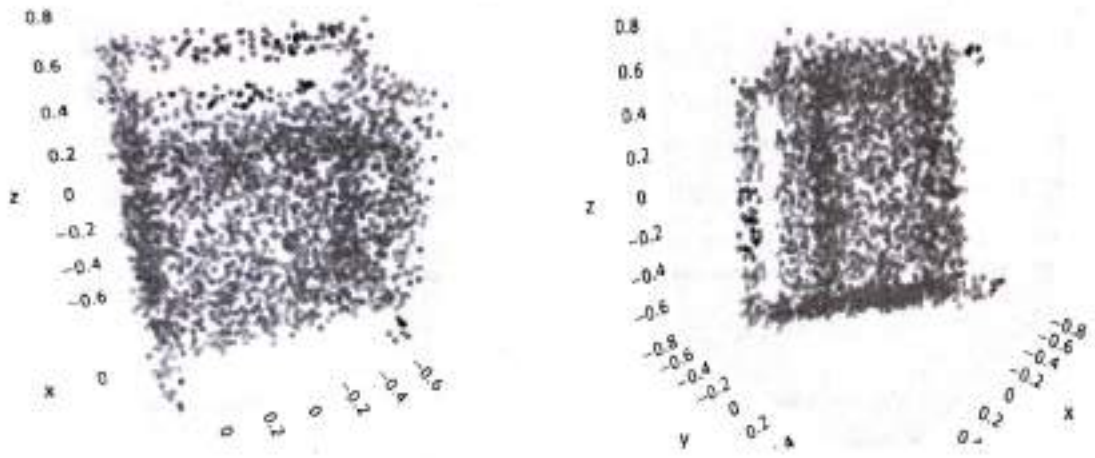


Figure : Tranformed Point Clouds

Loss

The custom loss criterion for unsupervised contrastive learning, a technique commonly employed in training neural networks for tasks like Siamese networks or SimCLR. The primary objective of contrastive learning is to encourage similar embeddings for positive pairs (pairs of similar samples) and dissimilar embeddings for negative pairs (pairs of dissimilar samples). The loss function incorporates a temperature parameter to adjust the sharpness of the probability distribution during similarity calculations. Utilizing a custom logic reminiscent of a triplet loss, the code accumulates positive and negative similarities for each anchor in the batch and computes the loss based on the negative log of the ratio between these similarities. Normalization ensures that the loss is independent of the batch size. In essence, the custom loss aims to guide the learning process by penalizing differences in similarity for positive and negative pairs, fostering the development of discriminative features within the embeddings for effective unsupervised representation learning.

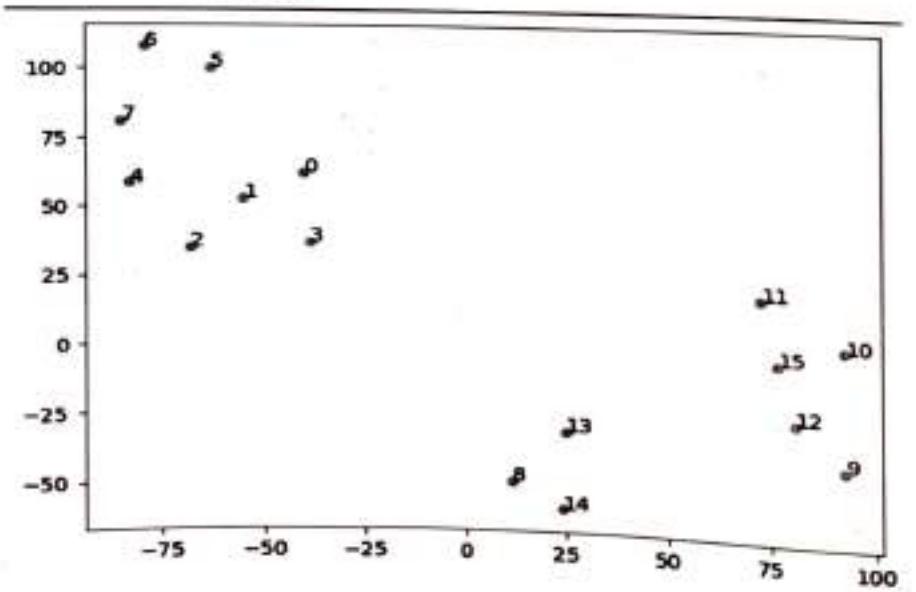
Chapter 4 : Results

4.1 Result:

The visualization of point cloud embeddings using t-Distributed Stochastic Neighbor Embedding (t-SNE), a dimensionality reduction technique. The process begins by fetching a sample batch from the data loader and obtaining representations of the point clouds through a pre-trained model. The high-dimensional embeddings are then concatenated and converted to a NumPy array. Subsequently, t-SNE is applied to reduce the dimensionality of the embeddings to 2D, facilitating visualization while preserving pairwise similarities.

The resulting 2D t-SNE embeddings are depicted in a scatter plot, where each point corresponds to a data point in the original high-dimensional space. The positions of points in the 2D space are indicative of their similarities. Transparency is applied to the points for clearer visualization, and a distinctive color palette is utilized.

To aid identification, annotations representing the indices of the data points are added to the plot. This allows users to associate specific points on the scatter plot with corresponding entries in the original dataset. The visual representation offers insights into the structure, clustering, and relationships within the point cloud dataset. Points that are close in the t-SNE space are expected to share similarities, providing a valuable tool for exploring and understanding the underlying patterns in the high-dimensional embeddings.



Conclusion

The introduced method is an unsupervised representation learning approach tailored for 3D point cloud data. The core of the approach involves identifying rotation as a highly beneficial transformation for generating a contrastive version of an original point cloud. Through this method, it aims to learn unsupervised representations by maximizing the correspondence between paired point clouds, where each pair consists of an original point cloud and its contrastive version. The implementation of the method is designed to be straightforward and operates without the need for resource-intensive computing.

The evaluation of the unsupervised representations spans various downstream tasks, including 3D object classification, shape part segmentation, and scene segmentation. The experimental results illuminate the remarkable effectiveness of the approach in delivering impressive performance across these tasks.

Looking ahead, future endeavors involve delving into semi-supervised techniques to further enhance the performance of the method. Additionally, there is an aspiration to extend the applicability of the approach to diverse domains, including but not limited to 3D object detection. Through ongoing research and exploration, the aim is to contribute to the advancement of unsupervised learning methodologies for point cloud data, unlocking new possibilities and applications in the realm of three-dimensional representation learning.

References

Here are some references that can be useful for learning more about the project:

- https://colab.research.google.com/drive/1w3Sve2DFADuY2I2zcHRc4JvaAlFywvFl#scrollTo=c5JlUCLQ6pCh
- **https://arxiv.org/abs/2110.06632**
- **https://modelnet.cs.princeton.edu/**