Classification of Point Clouds using PointNet Architecture

Team

Manoj Gembali (A20527288)

Venkata Sai Neeharika Koniki (A20527105)

Rohith Reddy Bairi (A20526972)

ILLINOIS TECH

College of Computing

CS584- Machine Learning Spring 2024

Prof. Yan Yan

Classification of Point Clouds using PointNet Architecture

Team

Manoj Gembali (A20527288)

Venkata Sai Neeharika Koniki (A20527105)

Rohith Reddy Bairi (A20526972)

1. Introduction

The field of computer vision has undergone a significant transformation in recent years, evolving from traditional two-dimensional (2D) image analysis to incorporate the processing and understanding of three-dimensional (3D) data. Unlike conventional 2D computer vision techniques, which operate on flat images, 3D computer vision aims to extract spatial information and derive insights from three-dimensional environments. This emerging domain draws upon a diverse set of techniques, including point cloud processing, 3D reconstruction, and deep learning algorithms tailored to handle the complexities of 3D data.

The applications of 3D computer vision span across a wide range of industries and domains. In robotics and autonomous vehicles, it enables critical capabilities such as scene understanding, object recognition, and navigation in complex environments. Augmented reality (AR) applications leverage 3D computer vision to seamlessly integrate virtual elements into the real world, enhancing user experiences. Additionally, medical imaging has benefited from advancements in 3D computer vision, enabling more accurate diagnoses and treatment planning through the analysis of 3D medical scans.

While 3D computer vision presents numerous challenges, such as handling noisy sensor data, computational complexity, and the intricacies of representing and processing 3D information, its potential impact is immense. It holds the promise of revolutionizing industries and fostering innovation in a multitude of applications. Consequently, ongoing research and development efforts are dedicated to harnessing the capabilities of 3D computer vision and paving the way for its widespread adoption and real-world implementation.

In this project we will be implementing a CNN architecture to classify Modelnet10 Point Cloud data.

2. Problem statement

2.1 3D Data Representation Methods

Traditional approaches in 3D computer vision have relied heavily on extracting multiple 2D images from different viewpoints of a single model and employing multi-view convolutional neural network (CNN) architectures for classification tasks. While this technique has proven effective, it suffers from significant limitations in terms of computational resource requirements and scalability issues. As the number of viewpoints and models increases, the computational demands escalate rapidly, making these methods impractical for real-time applications or large-scale datasets.

Furthermore, traditional 3D convolution techniques, while powerful for processing voxelized 3D data, pose substantial computational burdens due to the inherent complexity of 3D convolutions. This computational overhead renders these methods impractical for real-time applications or

large-scale datasets, limiting their practical applicability.

Point cloud data, which represents geometric information as a set of unstructured 3D points, offers a promising alternative representation for 3D data. However, due to its irregular structure, researchers often convert point cloud data into regular voxel grids or sets of 2D image projections, which can lead to unnecessary data redundancy, increased storage requirements, and potential information loss.

The primary challenge lies in developing efficient and effective methods for directly processing and analyzing point cloud data in its native, unstructured format. Achieving accurate classification and semantic segmentation of point clouds without resorting to computationally expensive voxelization or projection techniques remains a significant obstacle. Overcoming this challenge is crucial for enabling real-time, scalable, and memory-efficient 3D computer vision applications across various domains, including robotics, autonomous vehicles, augmented reality, and medical imaging.

2.2 3D Geometric Object Data Representation

There are three main formats for representing 3D geometric objects: voxels, meshes, and point clouds.

Voxels:

Voxels are like 3D pixels, dividing space into small cubes. Each voxel contains information about the object's properties, like density or color. However, they pose challenges in machine learning:

Working with voxel data can be complex and memory-intensive due to its high dimensionality.

In areas with lots of empty space, voxels can create a sparse representation, making storage and processing less efficient.

Memory limitations may restrict the detail we can capture, leading to loss of fine-grained information.

Meshes:

Meshes consist of connected vertices, edges, and faces, forming the object's structure. Each vertex holds spatial coordinates, and edges connect vertices to create faces, defining the object's surface.

Point Clouds:

Point clouds are collections of 3D points, each representing a specific location or feature. They lack connectivity but retain precise spatial coordinates and additional attributes like color or intensity.

Point cloud data is straightforward to generate but often converted to regular voxel grids or 2D image sets to simplify its irregular structure.

3. Data Collection and Preprocessing

3.1 Data Sources

The dataset central to this project is the ModelNet10 dataset, a comprehensive collection of 3D models curated by researchers at Princeton University. This widely recognized dataset has become a valuable resource for researchers and practitioners in the field of 3D computer vision and point cloud processing. The

ModelNet10 dataset can be accessed through the following link:

https://modelnet.cs.princeton.edu/

3.2 Data Description

The ModelNet10 dataset comprises a diverse range of 3D models, encompassing ten distinct categories: bathtubs, beds, chairs, desks, dressers, monitors, nightstands, sofas, tables, and toilets. Each category contains hundreds of high-quality 3D models, providing a rich and diverse representation of real-world objects.

The 3D models within the dataset are stored in the widely used Object File Format (OFF), a standardized format for representing 3D geometric data.

3.3 Preprocessing Steps

3.3.1 Data processing

The ModelNet10 dataset, containing hundreds of 3D models across ten categories, was used as the starting point for our project. To convert these models into a format suitable for PointNet, we employed the trimesh library to sample 2048 points from each 3D mesh, resulting in a point cloud representation.

Data augmentation plays a crucial role in improving the generalization capability of machine learning models and increasing the effective size of the training dataset. In our project, we applied augmentation techniques to the point cloud data to simulate real-world variations and introduce diversity.

The augmentation process involved two key steps:

1. **Jittering Points:**
   To simulate sensor noise and variations in point cloud data, we added random noise to each point in the point cloud. This was achieved by sampling from a uniform distribution within a small range (-0.005, 0.005) and adding the sampled values to the respective coordinates of each point.
2. **Shuffling Points:**
   Since PointNet is designed to be permutation-invariant, we shuffled the order of points within each point cloud. This step ensures that the model's performance is not affected by the order in which points are processed.

The augmented point cloud data was then combined with the original data to form the final training and testing datasets.

4. Implementation

4.1 PointNet Architecture:

PointNet is a deep learning architecture designed for effectively processing raw point cloud data while ensuring permutation and transformation invariance. Developed primarily for 3D vision tasks, PointNet has been successfully applied to tasks such as object recognition, classification, and segmentation in point cloud data.

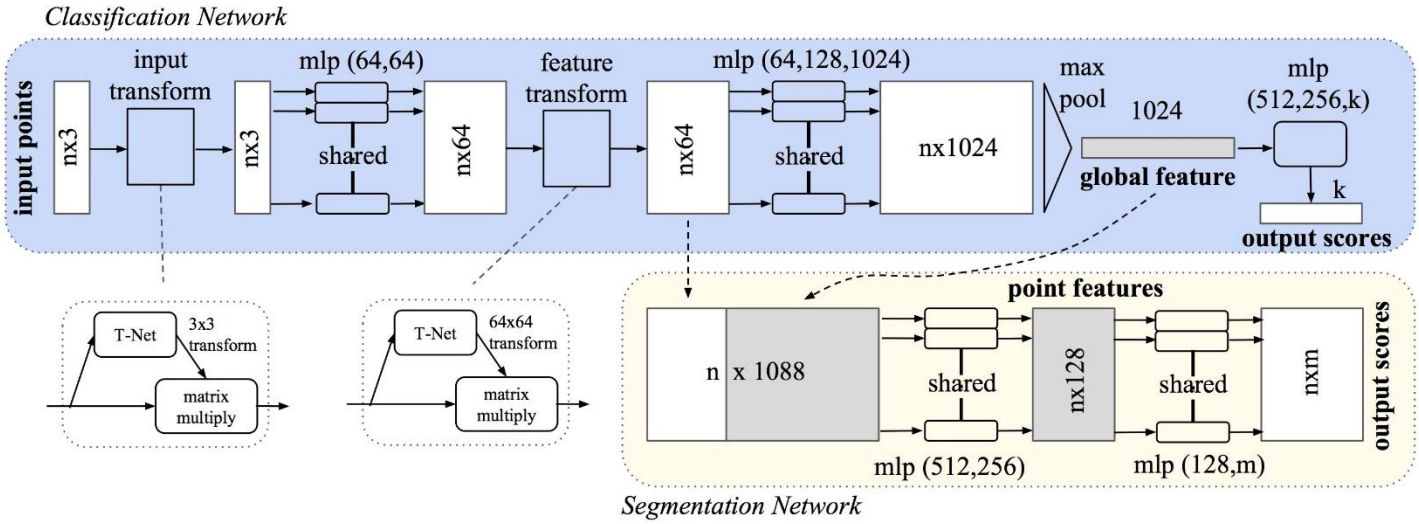
The architecture of PointNet comprises several key components, including T-Net layers, convolutional layers, global feature extraction, fully connected layers, and an output layer. These components work together to transform raw point cloud data into a canonical coordinate system, extract local and global features, and perform classification based on the learned features.

During training, PointNet learns to extract discriminative features from point clouds by optimizing a suitable loss function, typically using gradient-based optimization methods such as stochastic gradient descent (SGD) or its variants. The network parameters, including weights and biases, are iteratively adjusted to minimize the discrepancy between predicted and ground truth labels.

Through its robust architecture and training procedure, PointNet demonstrates remarkable resilience to variations in object orientation, position, and permutation of points within the point cloud. This makes it suitable for a wide range of 3D vision tasks, where traditional 2D convolutional neural networks (CNNs) may struggle due to the irregular and unstructured nature of point cloud data.

In practice, PointNet can be implemented using deep learning frameworks such as TensorFlow or PyTorch, providing researchers and practitioners with flexible and efficient tools for developing and deploying state-of-the-art solutions for 3D

perception tasks. We have implemented it using the tensorflow framework.



4.1.1 T-Net Layers:

- **Transformation Parameters**: T-Net layers learn transformation parameters (such as rotation and translation matrices) that map the input point cloud into a canonical coordinate system. These parameters are learned during training through backpropagation.
- **Learnable Parameters**: T-Net layers contain learnable parameters that are optimized to minimize the loss function, ensuring that the transformed point cloud retains essential geometric information while being invariant to transformations.
- **Normalization**: T-Net layers may incorporate normalization techniques,

such as batch normalization, to stabilize the learning process and accelerate convergence.

Implementation of T-Net Layer:

The T-net layers consist of a series of convolutional layers with 1D convolutions. We use kernel sizes of 1 to capture individual points and their features within the point cloud.

The initial convolutional layer has 32 filters to extract low-level features, followed by subsequent layers with 64 and 512 filters to capture more complex patterns and structures within the point cloud.

After the convolutional layers, a global max pooling operation is applied to aggregate the features across all points in the point cloud. This operation helps in capturing the most salient features from the entire point cloud representation.

Following the global max pooling, dense fully connected layers are employed to further process the extracted features. These layers consist of 256 and 128 neurons, respectively, facilitating the extraction of high-level representations from the global feature vector.

An Orthogonal Regularizer is utilized to enforce orthogonality constraints on the learned transformation matrix. This regularization term penalizes deviations from the identity matrix, ensuring that the learned transformation maintains the orthogonality property.

4.1.2 Convolution Layers

- **Local Context Extraction**: 1D convolutions are applied to capture local spatial relationships between neighboring points within the point cloud. This helps the network discern fine-grained features and patterns.
- **Kernel Size**: The size of convolutional kernels determines the receptive field of each convolutional operation, influencing the scale of features captured. PointNet may employ kernels of varying sizes to capture features at multiple scales.
- **Hierarchical Feature Extraction**: Convolutional layers in PointNet may be stacked to form hierarchical feature extraction pipelines, where higher layers capture increasingly abstract and complex features.

  Implementation:

  The input shape for the model is defined as (2048, 3), indicating a point cloud with 2048 points, each represented by 3 coordinates (x, y, z).

  The previously define T-net layers are integrated into the convolutional neural network architecture as functional components. They are applied at the beginning and middle of the network to perform point cloud transformation and feature extraction.

  Dropout layers with a dropout rate of 0.3 are inserted after the dense layers to prevent overfitting by randomly dropping neurons during training.

  The output layer consists of a dense layer with softmax activation, producing class probabilities for the input point cloud data. The number of output units is determined by the number of classes in the classification task (NUM_CLASSES).

4.1.3 Global feature Extraction

- **Pooling Operations**: In addition to global max pooling, PointNet may utilize other pooling operations, such as average pooling or adaptive pooling, to aggregate information from the entire point cloud. Different pooling strategies

can impact the network's ability to capture global context.

- **Feature Dimensionality Reduction**: Global pooling operations are often followed by operations that reduce the dimensionality of the global feature vector. Techniques such as feature-wise max or average pooling may be employed to condense the feature representation.

  Implementation:

  Global max pooling is performed using the layers.GlobalMaxPooling1D() function, which aggregates the features across all points in the point cloud, resulting in a global feature vector.

  The global feature vector obtained from global max pooling is then passed through dense fully connected layers for further processing. These layers consist of 256 and 128 neurons, respectively, facilitating the extraction of high-level representations from the global feature vector.

4.1.4 Fully connected Layers

- **Depth and Width**: The depth (number of layers) and width (number of neurons) of fully connected layers can vary depending on the complexity of the classification task and the size of the input point cloud.
- **Regularization Techniques**: Fully connected layers may incorporate regularization techniques such as dropout or weight decay to prevent overfitting and improve generalization performance.
- **Non-linear Activation**: ReLU activation functions introduce non-linearities into the network, allowing it to learn complex mappings between input and output spaces.

  Implementation:

  Dense fully connected layers are implemented using the layers.Dense() function. These layers have 256 and 128 neurons, respectively.

  Dropout layers with a dropout rate of 0.3 are inserted after each dense layer to prevent overfitting.

  ReLU activation functions are applied to the output of each dense layer using the layers.Activation("relu") function.

4.1.5 Output Layer

- **Loss Function**: The output layer computes class probabilities based on the learned features and applies a suitable loss function (e.g., cross-entropy loss) to measure the disparity between predicted and ground truth labels during training.
- **Training Objective**: The objective during training is to minimize the loss function, thereby optimizing the network's parameters to make accurate predictions on unseen data.
- **Evaluation Metrics**: In addition to the loss function, evaluation metrics such as accuracy, precision, recall, and F1 score may be used to assess the performance of PointNet on classification tasks.

  Implementation:

  The output layer consists of a dense layer with softmax activation, implemented using the layers.Dense(NUM_CLASSES, activation="softmax") function. This layer produces class probabilities for the input point cloud data.

  Cross-entropy loss is commonly used as the loss function for classification tasks and can be applied during model compilation using the appropriate loss function in TensorFlow's compile() function.

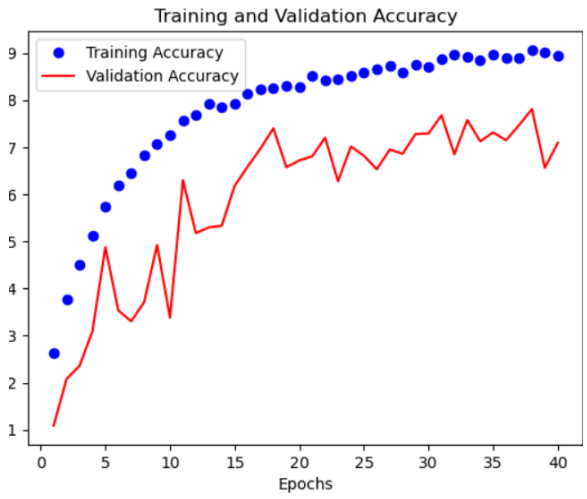
  During training, the objective is to minimize the cross-entropy loss

function, optimizing the network's parameters for accurate predictions on unseen data. Evaluation metrics such as accuracy can be computed using TensorFlow's built-in metrics functions during model evaluation.

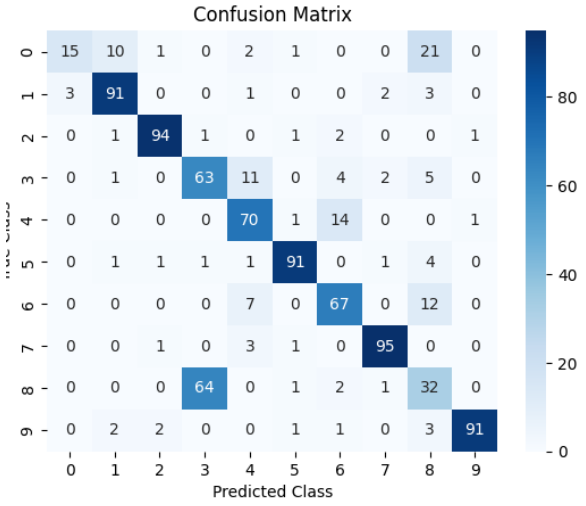
4.2 Results and Evaluation

To assess the performance of our PointNet model and gain insights into its behavior, we trained the model for 40 epochs and monitored various evaluation metrics throughout the training process. The model's weights and training history were saved at each iteration, allowing us to analyze and compare the results across different epochs.

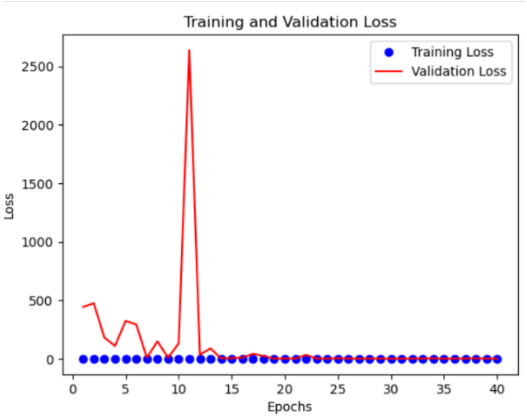
Training and Validation Accuracy: The training and validation accuracy curves provide valuable information about the model's learning capability and generalization performance. As shown in the image below, the training accuracy consistently improved over the course of the training process, reaching a maximum value of approximately 92%.



Confusion Matrix: The confusion matrix offers a detailed breakdown of the model's performance across different classes. It highlights the classes that the model accurately classified and identifies any potential areas of confusion or misclassification. The confusion matrix for our PointNet model is presented in the following image:



Training and Validation Loss: Monitoring the training and validation loss is crucial for understanding the model's convergence and potential overfitting or underfitting issues. The training and validation loss curves for our PointNet model are shown below:



After evaluating the results across multiple epochs, we determined that the 38th iteration achieved the best performance. Consequently, we selected the model weights from this iteration as our final model.

Final Results:

Test Loss: 4.78445
Test Accuracy: 0.7808

The test accuracy of 0.7808 (78.08%) indicates that our PointNet model was able to correctly classify approximately 78% of the instances in the test dataset. While this performance is promising, there is still room for improvement through further fine-tuning, hyperparameter optimization, or incorporating additional techniques such as data augmentation or ensemble methods.

These evaluation metrics and visualizations provided valuable insights into the model's behavior and performance, enabling us to identify areas for potential improvement and make informed decisions regarding the final model selection.

5. Applications/ Conclusion:

In conclusion, the integration of 3D point cloud machine learning heralds a new era of data-driven innovation, offering boundless opportunities across diverse domains. From urban planning to healthcare, and from robotics to Virtual Reality (VR), the transformative impact of 3D point cloud ML resonates across industries, reshaping the way we perceive, analyze, and interact with three-dimensional data.

In urban planning and city modeling, 3D point cloud ML enables precise semantic segmentation of urban environments, facilitating informed decision-making for infrastructure development and disaster response planning. Moreover, in cultural heritage preservation and archaeology, the reconstruction capabilities of point cloud ML empower researchers to digitize and preserve historical artifacts and architectural marvels, unlocking insights into our shared past.

In the realm of robotics and autonomous systems, the applications are equally profound. Object classification and scene understanding empower autonomous vehicles to navigate complex environments safely, while 3D object detection and localization facilitate object manipulation tasks in industrial automation, enhancing efficiency and productivity.

Furthermore, in healthcare and biomedical research, 3D point cloud ML holds promise for advancing medical imaging and diagnostics. By leveraging point cloud data from medical scans, researchers can develop innovative diagnostic tools and treatment planning algorithms, ushering in a new era of personalized medicine and patient care.

In the realm of Virtual Reality (VR), 3D point cloud ML enables immersive experiences that blur the lines between the virtual and physical worlds. By leveraging point cloud data to create realistic 3D environments, VR applications can offer users unparalleled immersion and interactivity, revolutionizing entertainment, education, and training.

As we look to the future, it is evident that the potential of 3D point cloud machine learning knows no bounds. By harnessing the synergies between advanced ML techniques and rich spatial data, we can address pressing challenges, drive innovation, and forge a path towards a more connected, intelligent, and sustainable world.

6. References

1. Charles, R. Qi; Su, Hao; Kaichun, Mo; Guibas, Leonidas J. (2017). *[IEEE 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - Honolulu, HI (2017.7.21-2017.7.26)] 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. , (), 77–85.* doi:10.1109/cvpr.2017.16

2. CS492(A) Machine Learning for 3D Data (KAIST, Spring 2022)
3. Modelnet10 dataset
4. A. Garcia-Garcia, F. Gomez-Donoso, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla and J. Azorin-Lopez, "PointNet: A 3D Convolutional Neural Network for real-time object class recognition," *2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Canada, 2016, pp. 1578-1584, doi: 10.1109/IJCNN.2016.7727386.
5. https://towardsdatascience.com/point-net-for-semantic-segmentation-3eea48715a62

GitHub Link:
https://github.com/Manoj99Q/ML_Project_PointCLoud.git