Department of Artificial Intelligence and Machine Learning
Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)

Traffic Density Estimation using PointNet

REPORT

SUBMITTED BY

Name:Kalpitha S Naik	**USN:1MS22AI022**
Name:Khushi D K	**USN:1MS22AI023**
Name:Megha Prasad	**USN:1MS22AI027**
Name:Rajatha	**USN:1MS22AI047**

As part of the Course **Fundamentals of Data Science– AI51**

SUPERVISED BY
Faculty
Dr. Meeradevi

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE/ ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING
RAMAIAH INSTITUTE OF TECHNOLOGY
Oct 2024 –Jan 2025

RAMAIAH
Institute of Technology

CERTIFICATE

This is to certify that **Name:** Kalpitha S Naik(**USN:** 1MS22AI022), **Name:** Khushi DK(**USN:**1MS22AI023), **Name:** Megha Prasad(**USN:** 1MS22AI027), **Name:** Rajatha(**USN:** 1MS22AI047) have completed the **"Traffic Density Estimation using PointNet"** as a part of Fundamentals of Data Science Course

Submitted by Guided by

Name: Kalpitha S Naik	USN:1MS22AI022
Name: Khushi D K	USN:1MS22AI023
Name: Megha Prasad	USN:1MS22AI027
Name: Rajatha	USN:1MS22AI047

RAMAIAH
Institute of Technology

Evaluation Sheet

Sl. No	USN	Name	Demonstration of Project (10M)	Presentation & Report (10M)	Total Marks (20)
1.	1MS22AI022	Kalpitha S Naik			
2.	1MS22AI023	Khushi D K			
3.	1MS22AI027	Megha Prasad			
4.	1MS22AI047	Rajatha			

Evaluated By

Name: Dr. Meeradevi
Designation: Associate Professor
Signature:

1.INTRODUCTION

Urbanization and technological advancements have dramatically transformed transportation systems, leading to a surge in vehicle usage and traffic congestion. Managing traffic flow efficiently has become a major challenge for urban planners and policymakers. Accurate traffic density estimation is crucial for traffic management, route optimization, congestion control, and the development of autonomous driving technologies. Traditional traffic monitoring techniques often rely on camera systems and radar sensors, which are susceptible to challenges such as poor lighting, occlusions, and adverse weather conditions. To address these limitations, this project explores the use of LIDAR-based systems in conjunction with PointNet, a neural network architecture tailored for processing 3D point cloud data.

Point clouds, generated by LIDAR sensors, provide rich three-dimensional spatial data that offer significant advantages over traditional approaches. They are immune to lighting variations and can operate effectively in diverse environmental conditions. However, the unstructured nature of point cloud data poses unique computational challenges, necessitating advanced processing methods. PointNet emerges as an ideal solution, as it is specifically designed to handle unordered point cloud data while maintaining permutation invariance and leveraging both global and local geometric features.

The PointNet architecture efficiently processes raw point cloud data without requiring complex preprocessing steps such as voxelization. Its lightweight and scalable design make it suitable for real-time applications, which are essential for traffic monitoring and management. By extracting spatial features directly from raw data, PointNet can detect and classify vehicles, track their movements, and analyze traffic density patterns with remarkable accuracy. This capability enables smarter traffic management systems that dynamically adapt to real-time conditions, thereby improving urban mobility and reducing congestion.

The growing adoption of smart city infrastructure and autonomous vehicles underscores the need for robust traffic density estimation systems. PointNet's ability to process unstructured data directly aligns with this requirement, making it a powerful tool for modern transportation networks. The integration of LIDAR-based systems with PointNet provides an advanced framework capable of addressing key challenges in urban traffic management. It also facilitates data-driven decision-making processes that enhance efficiency and sustainability.

In this project, we aim to leverage the capabilities of PointNet to analyze LIDAR data for traffic density estimation. By classifying objects, segmenting scenes, and identifying spatial distributions, the system can provide detailed insights into traffic patterns. These insights can be used to optimize signal timings, plan alternate routes, and even support autonomous navigation systems. The proposed framework is not only scalable but also adaptable to varying traffic conditions, ensuring robustness and reliability.

Furthermore, this approach highlights the potential of neural networks in transforming traditional traffic monitoring methods. With PointNet's capacity to handle dynamic and unstructured environments, it opens avenues for applications beyond traffic management, such as crowd monitoring, infrastructure planning, and emergency response coordination. This project serves as a step forward in harnessing advanced AI techniques to create intelligent, responsive, and resilient urban transportation systems.

2.PROBLEM STATEMENT

With the increasing adoption of autonomous vehicles and the growing reliance on sensor-based data, traffic density estimation has become an essential task for enhancing the efficiency and safety of transportation systems. One promising approach is the use of 3D point clouds generated by LiDAR sensors, which offer high-resolution spatial information about the environment. However, estimating traffic density from point clouds remains a challenging task due to the complexities in interpreting raw LiDAR data, handling occlusions, and distinguishing various objects in dynamic traffic scenarios.

This research aims to leverage the PointNet algorithm, a deep learning framework designed for processing unordered point clouds, to estimate traffic density in urban environments. Using the NuScenes dataset, which provides high-quality, annotated point cloud data captured from LiDAR sensors in real-world traffic scenarios, this study will focus on calculating the point cloud density as an indicator of traffic congestion.

The core objective of this project is to develop and train a model that can:

1. Efficiently process LiDAR point cloud data from the NuScenes dataset using PointNet.
2. Calculate the density of the point cloud in different areas of the scene to infer traffic congestion levels.
3. Train a machine learning model to correlate point cloud density with traffic flow and congestion.
4. Evaluate the accuracy of traffic density estimation and its potential applications for real-time traffic management and autonomous driving.

By addressing the challenge of traffic density estimation from point cloud data, this research could contribute to the development of intelligent transportation systems capable of dynamically adjusting to traffic conditions, improving vehicle routing, and enhancing safety measures.

3.METHODOLOGY

The methodology involves collecting 3D point cloud data from LIDAR sensors, utilizing datasets such as NuScenes or KITTI, which provide labeled annotations for training and evaluation. The raw data undergoes preprocessing, including normalization, noise removal, segmentation, and augmentation, ensuring robustness and consistency for model input.

The PointNet architecture is then implemented to process unordered point clouds, extracting relevant features and classifying traffic density levels. The model is trained using cross-entropy loss and optimized with the Adam optimizer, while its performance is validated through metrics such as accuracy, precision, recall, and confusion matrices. Finally, the trained model is deployed for real-time traffic density estimation, enabling the classification of objects and integration into traffic monitoring systems and autonomous vehicle frameworks. This streamlined approach leverages the efficiency of PointNet for scalable and accurate traffic density estimation using LIDAR data.

Algorithm:

Input:

- 3D LIDAR point cloud data.
- Pre-trained PointNet model.

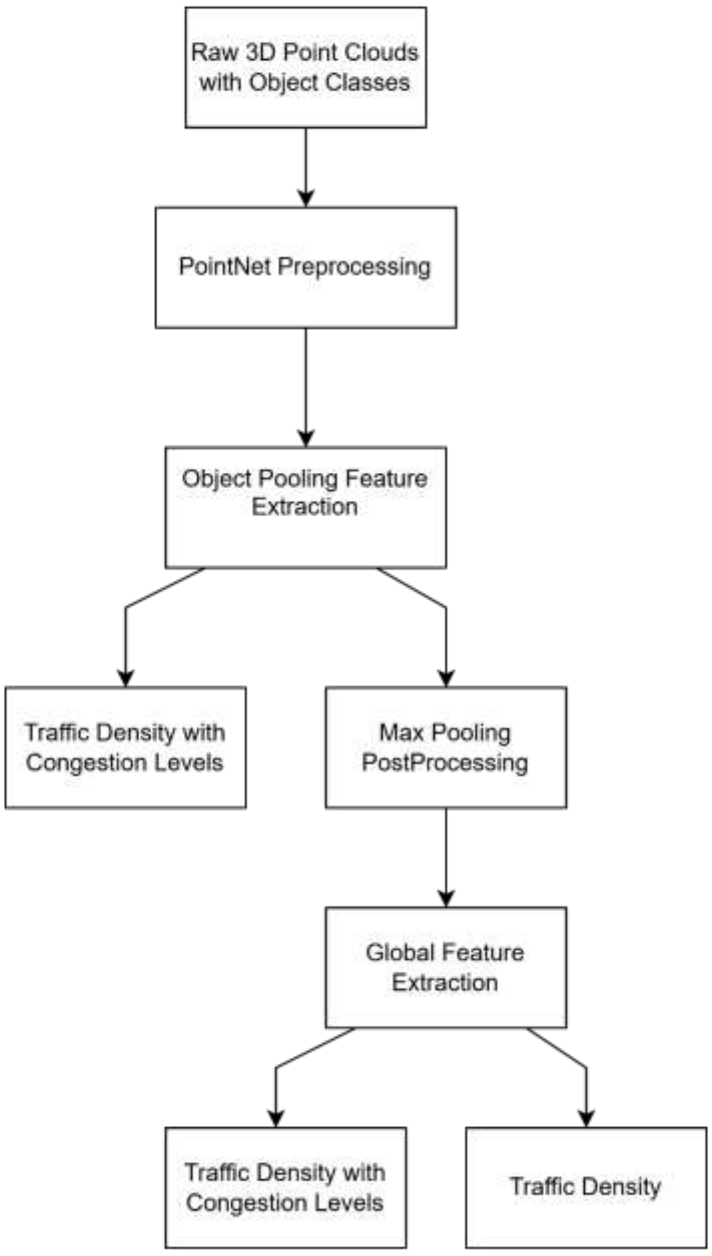
Output:

- Traffic density classification.
- Object detection and segmentation.

Steps:

1. **Data Preprocessing:**
    - Collect and clean LIDAR data.
    - Normalize coordinates and segment the region of interest.
2. **Feature Extraction with PointNet:**
    - Input raw point cloud data.
    - Extract local features using MLP layers.
    - Aggregate global features via max-pooling.
3. **Traffic Density Estimation:**
    - Detect and classify vehicles and objects.
    - Calculate density based on detected vehicles per unit area.
    - Analyze spatial clusters for density levels.
4. **Post-Processing and Output:**
    - Label density as Low, Medium, or High.
    - Visualize results and integrate with traffic systems.
5. **Model Evaluation:**
    - Test accuracy using metrics like precision and recall.
    - Optimize parameters for real-time performance.

Workflow Diagram:


Raw 3D Point Clouds with Object Classes
PointNet Preprocessing
Object Pooling Feature Extraction
Traffic Density with Congestion Levels
Max Pooling PostProcessing
Global Feature Extraction
Traffic Density with Congestion Levels
Traffic Density

This diagram illustrates a process for determining traffic density using 3D point cloud data. Raw point clouds are preprocessed using PointNet, followed by object-level feature extraction and max pooling. These features are then used to determine traffic density and congestion levels at both a local (object-based) and global scale.

Code:

```
import os
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader
from nuscenes.nuscenes import NuScenes
from nuscenes.utils.data_classes import LidarPointCloud
from Pointnet_Pointnet2_pytorch.models.pointnet_utils import PointNetEncoder,
feature_transform_reguliarzer
import open3d as o3d
from pyquaternion import Quaternion
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
NUM_CLASSES=5
class PointNetCls(nn.Module):
def __init__(self, k=NUM_CLASSES, normal_channel=True):
super(PointNetCls, self).__init__()
if normal_channel:
channel = 6
else:
channel = 3
self.feat = PointNetEncoder(global_feat=True, feature_transform=True, channel=channel)
self.fc1 = nn.Linear(1024, 512)
self.fc2 = nn.Linear(512, 256)
self.fc3 = nn.Linear(256, k)
self.dropout = nn.Dropout(p=0.4)
self.bn1 = nn.BatchNorm1d(512)
self.bn2 = nn.BatchNorm1d(256)
self.k = k

def forward(self, x):
x, trans, trans_feat = self.feat(x)
x = F.relu(self.bn1(self.fc1(x)))
x = F.relu(self.bn2(self.dropout(self.fc2(x))))
x = self.fc3(x)
x = F.log_softmax(x, dim=1)
return x, trans, trans_feat

class PointNetLoss(nn.Module):
def __init__(self, mat_diff_loss_scale=0.001):
super(PointNetLoss, self).__init__()
self.mat_diff_loss_scale = mat_diff_loss_scale

def forward(self, pred, target, trans_feat):
loss = F.nll_loss(pred, target)
mat_diff_loss = feature_transform_reguliarzer(trans_feat)
total_loss = loss + mat_diff_loss * self.mat_diff_loss_scale
return total_loss
In [6]:
class NuScenesTrafficDataset(Dataset):
def __init__(self, nusc, split='train', num_points=2048, normal_channel=True):
self.nusc = nusc
self.split = split
self.num_points = num_points
self.normal_channel = normal_channel
self.scene_tokens = [s['token'] for s in nusc.scene]
self.lidar_tokens = []
```

```
for scene_token in self.scene_tokens:
scene = nusc.get('scene', scene_token)
sample_token = scene['first_sample_token']

while sample_token:
sample = nusc.get('sample', sample_token)
lidar_token = sample['data']['LIDAR_TOP']
self.lidar_tokens.append(lidar_token)
sample_token = sample['next']
def __len__(self):
return len(self.lidar_tokens)

def get_traffic_density(self, points):
AREA_SIZE = 100 * 100

# Filter ground points (assuming z-axis is vertical)
# Only consider points between 0.1m and 3m height to focus on vehicles
vehicle_points = points[(points[:, 2] >= 0.1) & (points[:, 2] <= 3.0)]

point_density = len(vehicle_points) / AREA_SIZE

if point_density < 0.2:
return 0    # Very Low Traffic
elif point_density < 0.3:
return 1    # Low Traffic
elif point_density < 0.4:
return 2    # Medium Traffic
elif point_density < 0.5:
return 3    # High Traffic
```

```
else:
return 4    # Very High Traffic

def __getitem__(self, idx):
lidar_token = self.lidar_tokens[idx]
lidar_data = self.nusc.get('sample_data', lidar_token)
pcl_path = os.path.join(self.nusc.dataroot, lidar_data['filename'])

pc = LidarPointCloud.from_file(pcl_path)
points = pc.points.T

label = self.get_traffic_density(points)

points = points[:, :3]

if len(points) > self.num_points:
choice = np.random.choice(len(points), self.num_points, replace=False)
else:
choice = np.random.choice(len(points), self.num_points, replace=True)
points = points[choice, :]

points = points - np.mean(points, axis=0)
dist = np.max(np.sqrt(np.sum(points ** 2, axis=1)))
points = points / dist

if self.normal_channel:
normals = np.zeros((self.num_points, 3))
points = np.concatenate([points, normals], axis=1)

points = torch.FloatTensor(points)
```

```
label = torch.LongTensor([label])

return points.transpose(1, 0), label.squeeze()
In [7]:
training_losses = []
training_accuracies = []

def train_epoch(model, dataloader, criterion, optimizer, device):
model.train()
total_loss = 0
correct = 0
total = 0
batch_losses = []

for i, (points, target) in enumerate(dataloader):
assert torch.all(target < model.k), f"Invalid target values found: {target.unique()}"
points, target = points.to(device), target.to(device)

optimizer.zero_grad()
pred, trans, trans_feat = model(points)
loss = criterion(pred, target, trans_feat)

loss.backward()
optimizer.step()

pred_choice = pred.max(1)[1]
correct += pred_choice.eq(target.data).cpu().sum()
total += points.size(0)
total_loss += loss.item()
batch_losses.append(loss.item())
```

```
if i % 10 == 0:
print(f'Batch [{i}/{len(dataloader)}] Loss: {loss.item():.4f}')

accuracy = 100. * correct / total
avg_loss = total_loss / len(dataloader)

return avg_loss, accuracy, batch_losses
In [8]:
def evaluate(model, dataloader, device):
model.eval()
predictions = []
targets = []

with torch.no_grad():
for points, target in dataloader:
points, target = points.to(device), target.to(device)
pred, _, _ = model(points)
pred_choice = pred.max(1)[1]

predictions.extend(pred_choice.cpu().numpy())
targets.extend(target.cpu().numpy())

predictions = np.array(predictions)
targets = np.array(targets)

# Calculate overall accuracy
overall_accuracy = (predictions == targets).mean()

# Calculate per-class accuracies
```

```
class_accuracies = []
for class_idx in range(model.k):
mask = targets == class_idx
if mask.sum() > 0:
class_acc = (predictions[mask] == targets[mask]).mean()
class_accuracies.append(class_acc)
else:
class_accuracies.append(0.0)

return overall_accuracy, class_accuracies
In [17]:
def main():
global model, dataloader
BATCH_SIZE = 8
NUM_POINTS = 2048
NORMAL_CHANNEL = True
NUM_EPOCHS = 25
LEARNING_RATE = 0.0001

dataset = NuScenesTrafficDataset(nusc, num_points=NUM_POINTS,
normal_channel=NORMAL_CHANNEL)
dataloader = DataLoader(dataset, batch_size=BATCH_SIZE, shuffle=True, num_workers=4)

model = PointNetCls(k=NUM_CLASSES,
normal_channel=NORMAL_CHANNEL).to(device)
criterion = PointNetLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=LEARNING_RATE, betas=(0.9,
0.999))
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=20, gamma=0.5)

best_accuracy = 0
```

```
training_losses = []
training_accuracies = []

print("Starting training...")
for epoch in range(NUM_EPOCHS):
print(f'Epoch {epoch+1}/{NUM_EPOCHS}')

avg_loss, accuracy, batch_losses = train_epoch(model, dataloader, criterion, optimizer, device)
scheduler.step()

training_losses.append(avg_loss)
training_accuracies.append(accuracy)

print(f'Average loss: {avg_loss:.4f}')
print(f'Accuracy: {accuracy:.2f}%')

with open('training_metrics.txt', 'w') as f:
for epoch_idx, (loss, acc) in enumerate(zip(training_losses, training_accuracies)):
f.write(f"Epoch {epoch_idx+1}, Loss: {loss:.4f}, Accuracy: {acc:.2f}%\n")

print("\nTraining completed. Evaluating best model...")

if __name__ == "__main__":
main()
```

4.RESULTS AND DISCUSSIONS

2D Projections of 3D Point Cloud

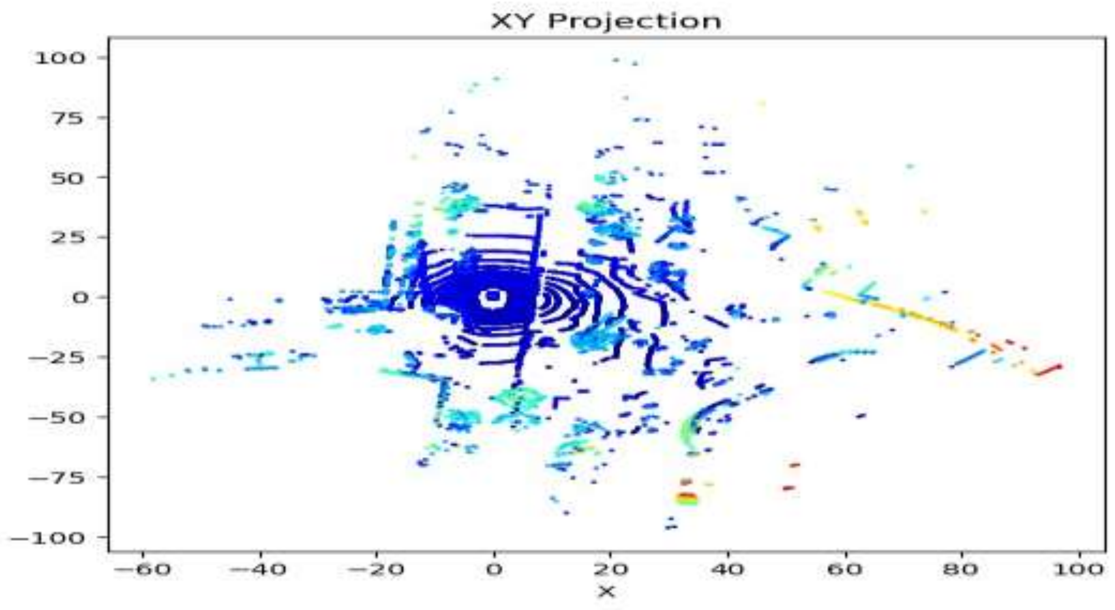


Fig 1

In Fig 1 XY projection of a 3D point cloud, where data points are visualized in a 2D plane by plotting their X and Y coordinates. It likely represents spatial distribution or scanned data, with color variations indicating intensity, height, or another parameter.

Visibility

In fig 2 and fig 3 visibility is defined as the fraction of pixels of a particular annotation that are visible over the 6 camera feeds, grouped into 4 bins.

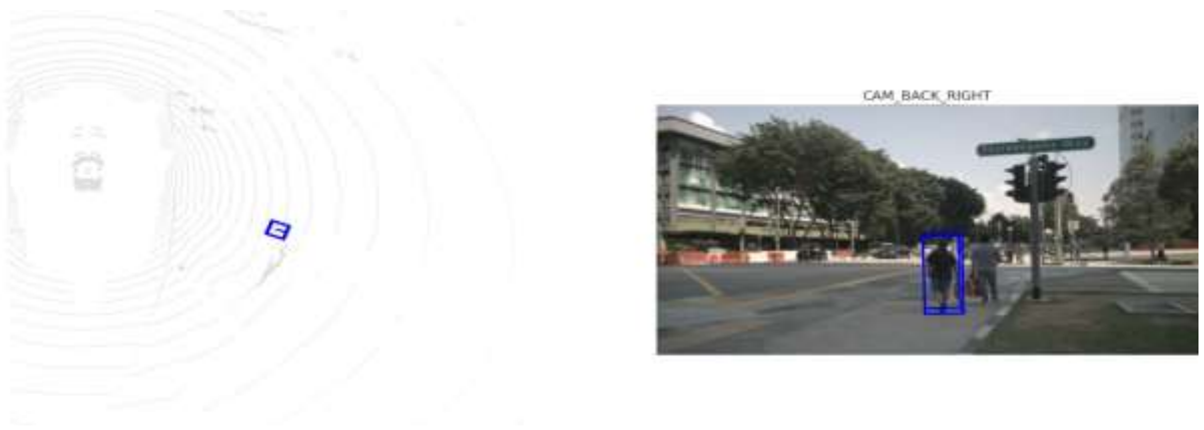


Fig 2

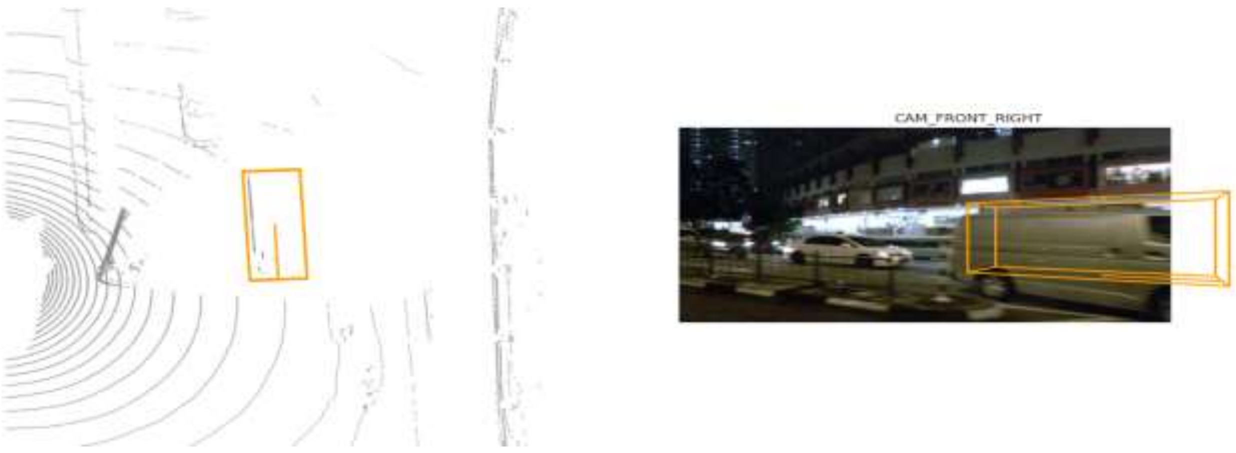


Fig 3

Training Loss and Accuracy

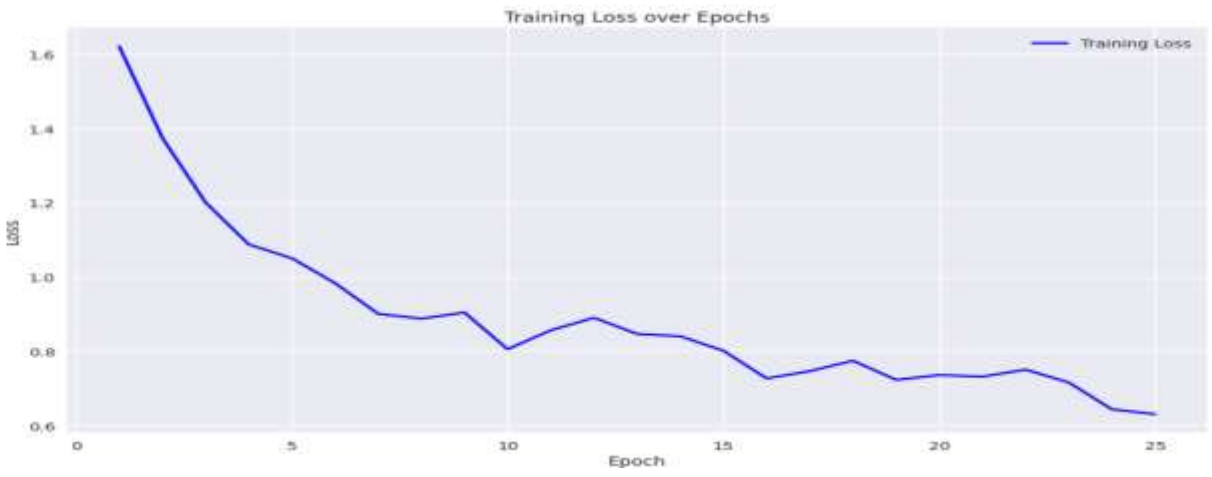


Fig 4

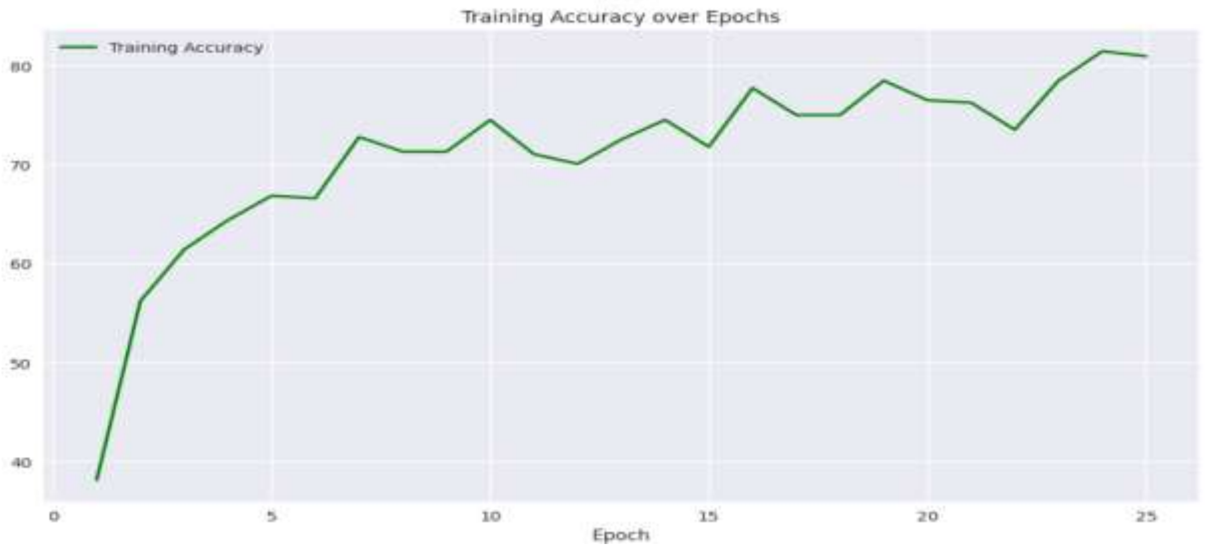


Fig 5

In Fig 4 graph shows the training loss over epochs during the training of a machine learning model. The decreasing trend in loss indicates that the model is learning and improving its performance as training progresses.

In the Fig 5 the graph shows the training accuracy of a model over 25 epochs. The accuracy increases rapidly in the initial epochs and then fluctuates before generally rising again towards the end. This suggests the model is learning, but may benefit from further training or adjustments to prevent overfitting.

Final Loss and Accuracy

Metric	Value
Final Loss	0.6305
Final Accuracy	80.94%
Best Accuracy	81.44%
Lowest Loss	0.6305

Confusion Matrix for Classification

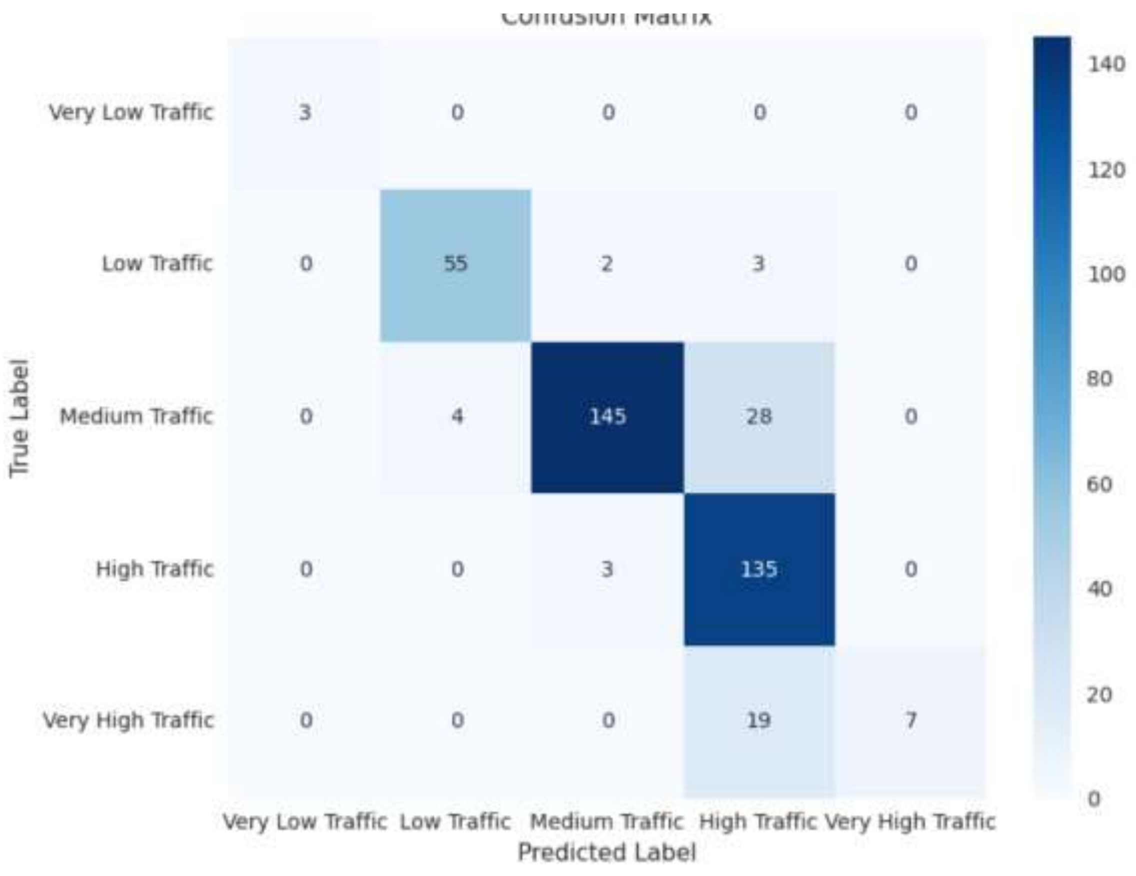


Fig 6

In fig 6 this confusion matrix evaluates a traffic classification model, showing its performance across different traffic levels. While the model accurately classifies most traffic (especially "Low," "Medium," and "High"), it struggles to differentiate between "Medium" and "High" traffic and has some difficulty with "Very High" traffic.

Object-wise Precision, Recall, and F1-Score

Class	**Precision**	**Recall**	**F1 Score**
Very Low Traffic	1.0000	1.0000	1.0000
Low Traffic	0.9322	0.9167	0.9244
Medium Traffic	0.9667	0.8192	0.8869
High Traffic	0.7297	0.9783	0.8359
Very High Traffic	1.0000	0.2692	0.4242

5.CONCLUSION

The use of the PointNet algorithm for traffic density estimation from LIDAR-based point clouds offers a promising approach to tackle real-world challenges in traffic management, especially in the context of autonomous vehicles and intelligent transportation systems. By utilizing the NuScenes dataset, which provides annotated 3D point cloud data captured from urban traffic scenarios, this research seeks to establish a robust method for estimating traffic congestion levels and analyzing point cloud density.

PointNet, with its ability to process raw, unordered 3D data directly, proves effective for calculating point cloud density in various areas of the scene. This density serves as an indicator of traffic congestion, which is crucial for dynamic traffic management and optimizing vehicle routing.

By training a machine learning model to correlate point cloud density with traffic flow, this study contributes to the understanding of how traffic density impacts real-time management systems, improving efficiency and safety in transportation networks.

PointNet excels in tasks like vehicle detection, trajectory prediction, and incident detection, which can significantly enhance traffic monitoring systems. The ability to extract both global and local features from point clouds enables the detection of complex patterns in traffic scenarios, such as congestion or accidents.

The model's ability to classify traffic density (e.g., high vs. low) has real-time applications for traffic flow optimization, incident detection, and route adjustments. This can further contribute to the development of autonomous driving systems capable of adapting to dynamic traffic conditions.

PointNet, combined with LIDAR sensors, is highly complementary in smart city applications, providing detailed 3D spatial information crucial for safe navigation. The use of point clouds over traditional camera or radar data makes it particularly useful in low-light and complex environments where other sensors might fail.

Ultimately, this research demonstrates the potential of leveraging PointNet and LIDAR point cloud data for traffic density classification and broader traffic flow management in autonomous systems. It represents a step toward more intelligent, adaptive transportation systems that improve safety, efficiency, and real-time traffic monitoring.