North East University Bangladesh

Department of Computer Science and Engineering

Towards the Creation and Spatio-temporal Representation of a Simulated Traffic Dataset of Sylhet

By

Md. Jahidul Islam
Reg. No: 200103020029
B.Sc. (Engg.) in CSE
4th year 2nd semester

Md. Ashrafuzzaman Sunny
Reg. No: 200203020002
B.Sc. (Engg.) in CSE
4th year 2nd semester

Supervised By

Ayon Dey
Lecturer
Dept. of CSE
North East University Bangladesh

January 21, 2024

Towards the Creation and Spatio-temporal Representation of a Simulated Traffic Dataset of Sylhet

A Thesis submitted to the Department of Computer Science and Engineering, North East University Bangladesh, in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering

By

Md. Jahidul Islam
Reg. No: 200103020029
B.Sc. (Engg.) in CSE
4th year 2nd semester

Md. Ashrafuzzaman Sunny
Reg. No: 200203020002
B.Sc. (Engg.) in CSE
4th year 2nd semester

Supervised By

Ayon Dey
Lecturer
Dept. of CSE
North East University Bangladesh

January 21, 2024

Recommendation Letter from Thesis Supervisor

These Students, *Md. Jahidul Islam and Md. Ashrafuzzaman Sunny*, whose thesis/project entitled *"Towards the Creation and Spatio-temporal Representation of a Simulated Traffic Dataset of Sylhet",* is under my supervision and agrees to submit for examination.

Signature of the Supervisor:

Ayon Dey
Lecturer
Dept. of CSE
North East University Bangladesh

Qualification Form of B.Sc. (Engg.) Degree

Student Name： **Md. Jahidul Islam, and Md. Ashrafuzzaman Sunny**.

Thesis Title： Towards the Creation and Spatio-temporal Representation of a Simulated Traffic Dataset of Sylhet.

This is to certify that the thesis was submitted by the student named above in January 2024. It is qualified and approved by the following persons and committees.

Head of the Dept.	**Supervisor**
Rathindra Chandra Gope	Ayon Dey
Associate Professor	Lecturer
Dept. of CSE	Dept. of CSE
North East University Bangladesh	North East University Bangladesh

Abstract

Traffic plays an important role in daily life. It can kill time and money. Accurate traffic prediction also plays an essential role in intelligent transportation systems, especially in Sylhet City. Nowadays this problem is very challenging due to their spatial and temporal dependencies on roads. Narrow roads, no lanes in roads, commercial areas, and residential areas cause different traffic in Bangladesh. Recently, a significant amount of research efforts has been devoted to this area, particularly deep learning methods. The purpose of this research is to mitigate traffic congestion in Sylhet City. Firstly, we first create a graph of the Sylhet City Corporation (SCC), showing the connection and distance between the major intersection points. Secondly, we create a numerically sound simulated dataset of the Sylhet city map. If we continue our research, we will create a Spatio-temporal representation based on the dataset and graph of the SCC.

Keywords: Spatio-temporal dependencies, sound simulated, deep learning,

Table of Contents

Abstract.....i

Table of Contents.....ii

List of Figures.....iv

List of Tables.....v

INTRODUCTION.....1

1.1 Challenges.....1

1.1.1 Lack of Available Dataset.....1

1.1.2 Complex and Spatial Dependencies.....1

1.1.3 External Factors.....2

1.2 Application Tasks.....2

1.2 Our Contributions.....2

BACKGROUND STUDY.....3

2.1 Survey on Visual Traffic Simulation.....3

2.2 Deep Learning Framework for traffic forecasting.....3

2.3 A new Framework for Spatio-Temporal Data forecasting.....3

2.4 Deep Learning on Traffic Prediction Methods.....3

2.5 You Only Look Once: Unified, Real-time object detection.....4

2.6 Traffic State Estimation on Highway.....4

2.7 Numerical Approximations of traffic flow model.....4

2.8 Traffic Flow Prediction for road transportation networks.....5

2.9 An Efficient Realization of deep learning for traffic data imputation.....5

2.10 Estimation of missing values in heterogeneous traffic Data.....5

PROPOSED METHOD.....6

3.1 Our Contribution.....6

3.1.1 Completed Graph.....6

3.1.2 Visual Representation of Graph.....10

3.1.3 Traffic flow dataset.....11

3.1.4 Proposed Methodology.....26

RESULTS AND DISCUSSION.....27

4.1 Graph Visualization.....27

4.2 City Points Distance.....27

4.3 YOLO Results.....28

4.4 Traffic flow Dataset.....29

4.5 Missing Data Imputation Result.....31

4.6 Comparing Results.....33

CONCLUSION.....35

5.1 Limitations.....35

5.2 Future Works.....35

References.....36

List of Figures

Figure 1: This Python code creates nodes for our graph……..17
Figure 2: Python code for creating our Graph class…….19
Figure 3: Python code to display a graph of Sylhet City……..21
Figure 4: Lamabazar at 9:30 AM (up) and 9:35 AM (down)……...22
Figure 5: Humayun Chottor at 9:10 AM (up) and 9:20 AM (down)……..……..23
Figure 6: Traffic flow data on a spreadsheet (Training)……...25
Figure 7: Traffic flow data on a spreadsheet (Testing)…….26
Figure 8: Training Images (Up) and Validation output (Down)……...29
Figure 9: Structure of DSAE…….29
Figure 10: Training Algorithms for the DSAE method……30
Figure 11: Upload and Create Dataframe…….31
Figure 12: Using fillna() to add null values……..32
Figure 13: Create Numpy array and add flatten……33
Figure 14: Randomly noise add……33
Figure 15: Create DSAE Model……34
Figure 16: Load Model……35
Figure 17: Test Set 1 Prediction……35
Figure 18: Test Set 1 Prediction Excel…….36
Figure 19: Test Set 2 Prediction……36
Figure 20: Test Set 2 Prediction Excel…….37
Figure 21: Visual Graph of Sylhet……38
Figure 22: Distance between neigh boring points…….39
Figure 23: Confusion Matrix (up) and Train Results (down)……...40
Figure 24: Traffic Flow Data from Training Dataset……42
Figure 25: Traffic Flow Data from Test Dataset……..42
Figure 26: Test Set 1 Prediction……43
Figure 27: Test Set 2 Prediction……...43

List of Tables

Table 1: Accuracy matrics for test data……44

Chapter 1

INTRODUCTION

Traffic Flow is a big issue in the populated areas of Sylhet, such as Lama-bazar, Amber-Khana, Zinda-bazar, Bondor, Uposhahor, etc. Accurate traffic prediction would greatly benefit Sylhet City. For example, mitigating traffic congestion; route planning for car-sharing services; emergency vehicle dispatching (ambulances, fire trucks, etc.). If traffic lights are implemented in the future, our dataset can be used to dynamically control traffic to mitigate congestion. The Denoising Stacked Auto-Encoder (DSAE) model presented in this paper can be used to simulate several months of traffic data for further prediction.

1.1 Challenges

There are many challenges when it comes to traffic prediction, particularly for Sylhet City. These are:
i) Lack of available dataset.
ii) Complex Spatial and Temporal dependencies.
iii) External factors.

Challenges we faced:
i) We requested access to traffic footage from ADC of Sylhet Metropolitan Police (SMP) Traffic; however, our request was denied.
ii) Due to strikes and elections during November and December, we could not go out and collect traffic data, so we did not gather as much as data as we wanted to.
iii) Due to lack of budget and volunteers, we could not gather much data as we wanted.

1.1.1 Lack of available datasets

A large amount of data is necessary to accurately perform machine learning tasks, however, there is no proper traffic dataset available for Sylhet City. Collecting a small amount of traffic data manually, and simulating the rest of the dataset, was a challenging task.

1.1.2 Complex Spatial and Temporal dependencies

The spatial correlation between an intersection points and its neighboring points is highly dynamic. The influence of the neighboring points varies with time. Dynamic temporal dependencies arise since traffic flow at a point will vary throughout the day, for example, traffic flow is high in the mornings when people

are going to work and school, it's very high in the afternoons and then decreases at night. These spatio-temporal dependencies need to be captured to make accurate traffic predictions.

1.1.3 External factors

External factors such as weather, accidents, strikes, and special events, all impact traffic and need to be accounted for to make accurate predictions.

1.2 Application Tasks

Traffic prediction involves various application tasks. Here, we list the main application tasks of the existing traffic prediction work, which are as follows:

Distance: The distance between two intersection points. Collected using Google Maps.
Flow: Traffic flow refers to the number of vehicles passing through a given point on the roadway in a certain period.
Demand: Demand is the popularity of a particular region, on a scale of 1-10. This will be predicted using the traffic flow in that region (the higher the flow, the higher the demand).

1.3 Our Contributions

To our knowledge, this is a novel work on creating a simulated traffic dataset for Sylhet City Corporation (SCC).

Our contributions are:
i) Firstly, we create a Graph of SCC, where nodes are intersection points and edges represent distance. We also create a visual representation of this graph.
ii) Create a Distance Table, showing the distances between all the major intersection points in SCC.
iii) Detect vehicles from pictures and count them, using YOLO model.
iv) Collect traffic flow information from the main intersection points of SCC, and summarize it in a spreadsheet.
v) Simulate the remaining data using Denoising Stacked Auto-encoder Model (DSAE).
vi) If time permits, we will create a Spatio-temporal representation of SCC, using the graph and the dataset.

Chapter 2

BACKGROUND STUDY

Simulation of traffic data is a rich and ongoing field of research. Many factors need to be considered such as vehicle speeds, weather conditions, direction/alignment of roads, population, etc. There are traditional methods of calculating traffic speed and flow using mathematical equations, as well as machine learning models like GANs that can generate missing traffic data. Likewise, representation learning is a relatively new ongoing area of research, since it is important to represent input data in such a way as to preserve all its features and attributes, before feeding it to machine learning models. We have studied several research papers based on simulation and representation learning, which we will outline below.

2.1 Survey on Visual Traffic Simulation

This paper comprises a comprehensive review of the state-of-the-art techniques for traffic simulation and animation. Discussion on three classes of traffic simulation models – microscopic, mesoscopic, macroscopic. Discuss how traffic simulations can benefit the training and testing of autonomous vehicles [1].

2.2 A deep learning framework for traffic forecasting

In this paper, the authors address the complex dependencies in traffic data by modeling it as a spatio-temporal graph. Utilizing graph and temporal convolutional layers to capture the spatial and temporal dependencies respectively. The authors proposed model can be achieved faster training and fewer parameters with flexibility and scalability [2].

2.3 A new framework for spatial-temporal network data forecasting

The STSGCN model can capture the localized spatial-temporal correlations effectively. Can take the heterogeneities in spatial-temporal data into consideration. The Author's model experiments show the model is superior to the existing other models [3].

2.4 Deep Learning on Traffic Prediction Methods

This paper authors summarize the existing traffic prediction methods and then gives a taxonomy. Then show the different approaches/techniques how to solve the traffic predictions in a city. Also, give an evaluation and analysis by conducting extensive experiments to compare the performance of different methods on a real-world public dataset [4].

2.5 You Only Look Once: Unified, Real-time object detection

The authors propose YOLO, a system designed to perform real-time object detection in images. YOLO frames object detection as a regression problem and predicts bounding boxes and class probabilities directly from the entire image in one forward pass. This approach allows YOLO to achieve high speed and efficiency.

Key features of YOLO are Unified Framework, Grid System, Single Forward Pass, Bounding Box Predictions, and Non-Maximum Suppression [5].

2.6 Traffic state estimation on highway

Traffic state estimation (TSE) is the process of inferring the traffic state variables (flow, density, speed, etc.) on road segments using partially observed traffic data.

TSE methods can be classified into three categories: model-driven, data-driven, and streaming-data-driven.

The challenges and future research directions for TSE on highways include the limited availability of traffic data, the noisy nature of traffic data, and the dynamic nature of traffic [6].

2.7 Numerical approximations of a traffic flow model on networks

The paper proposes a numerical method for solving a traffic flow model on networks. The model is based on the conservation laws of mass and momentum, and it is solved using conservative methods, such as the classical Godunov scheme and the more recent discrete velocities kinetic schemes.

The paper presents a numerical algorithm for solving the model at junctions. The algorithm is robust to perturbations in the initial conditions and to the choice of numerical parameters.

The paper applies the algorithm to some simple test cases and portions of urban networks. The results of the simulations show that the method can reproduce the main features of traffic flow on networks, such as the formation of traffic jams and the propagation of traffic waves [7].

2.8 Traffic Flow Prediction for Road Transportation Networks

The paper proposes a two-step approach for traffic flow prediction:

i) A dynamic traffic simulator is used to generate flows in all links using available traffic information, estimated demand, and historical traffic data available from links equipped with sensors.

ii) An auto-regressive model is used to predict the traffic flows on each link up to 30 minutes ahead.

The paper evaluates the proposed approach using a case study on a traffic network in San Francisco, CA, USA. The results show that the proposed approach can achieve good prediction accuracy even with limited traffic data.

The paper makes a significant contribution to the field of traffic flow prediction. The proposed two-step approach can achieve good prediction accuracy even with limited traffic data. The dynamic traffic simulator can capture the effects of traffic demand, traffic control, and road geometry on traffic flow. The auto-regressive model can capture the temporal correlation in traffic flow [8].

2.9 An efficient realization of deep learning for traffic data imputation

The DSAE model is a denoising stacked autoencoder model. It is trained on a dataset of traffic data that contains both corrupted and missing data points. The model learns to reconstruct the corrupted data points and impute the missing data points.

The DSAE model is more accurate than other traditional models for traffic data imputation.

The DSAE model can be trained efficiently using a hierarchical training approach [9].

2.10 Estimation of missing values in heterogeneous traffic data

The MMDL model is a multimodal deep learning model. It is trained on a dataset of traffic data that contains both corrupted and missing data points. The model learns to reconstruct the corrupted data points and impute the missing data points. The model is also able to capture the dependencies between different types of traffic data, such as speed, volume, and occupancy.

The MMDL model is more accurate than other traditional models for traffic data imputation.

The authors of the paper also proposed an efficient algorithm for training the MMDL model. The algorithm uses a hierarchical training approach, which allows the model to be trained more quickly. The authors also showed that the MMDL model can be used to impute traffic data from different periods and different spatial locations [10].

Chapter 3

PROPOSED METHOD

We have studied various research papers based on traffic data simulation and augmentation. From these papers, we have presented a methodology that best fits our dataset and allows us to simulate traffic flow data over 3-4 months.

3.1 Our contributions

Our contributions are as follows:
i) Completed Visualized Graph of the Sylhet City Corporation (using Python code).
ii) Distance Table, showing distance between all the major intersection points.
ii) Traffic flow dataset (7 days training set, 3 days testing set).
iv) Simulating remaining Dataset, using Denoising Stacked Auto-Encoder (DSAE) model.

3.1.1 Completed Graph

Create a complete graph based on Sylhet City Corporation (SCC). Nodes represent the major intersection points, and weighted edges represent the distance between points. The nodes can hold information such as Point Name, traffic flow information, etc. Many Machine Learning Models (Such as GCN) take input as a graph, which is why a graph representation is necessary.

The Python code shown below creates nodes, which holds unique id, node name, etc. Also creates our graph. Has methods to insert or delete nodes, insert and remove edges, see the degree of a node, and see a node's neighbors. This Graph can be used to store all the traffic flow data within the nodes, and the distance in the edges.

```python
class node:
    def __init__(self, id, name, demand):
        self.id = id
        self.name = name
        self.demand = demand


if __name__ == '__main__':
    n = int(input("how many nodes: "))
    L = []*n

    for i in range(n):
        s = input("enter id, name, demand ").split()
        id = int(s[0])
        name = s[1]
        demand = int(s[2])
        L.append(node(id, name, demand))

    for i in range(n):
        print(L[i].id, L[i].name, L[i].demand)
        print()
```

Figure 1: This Python code creates nodes for our graph [16].

```python
class Graph:
    def __init__(self, V, E):
        self.V = set(V)
        # self.E = set(frozenset((u,v) for u,v in E))
        self.E = set(E)
        self.adj = {}
        for v in self.V:
            self.addVertex(v)
        for u,v in self.E:
            self.addEdge(u,v)

    def addVertex(self, v):
        if v not in self.adj:
            self.adj[v] = set()
        self.V.add(v)

    def addEdge(self, u, v):
        self.addVertex(u)
        self.addVertex(v)
        self.adj[u].add(v)
        self.adj[v].add(u)
        self.E.add((u,v))

    def deg(self, v):
        return len(self.adj[v])
```

```
    @property
    def vertices(self):
        return len(self.V)

    def removeEdge(self, u, v):
        e = (u,v)
        if e in self.E:
            self.E.remove(e)
            self.adj[u].remove(v)
            self.adj[v].remove(u)

    def removeVertex(self, u):
        for v in self.neighbors(u):
            self.removeEdge(u,v)
        del self.adj[u]
        self.V.remove(u)

if __name__ == '__main__':
    G = Graph({1,2,3,4}, {(1,2), (2,3), (1,4), (3,4)})
    G.addEdge(1,5)

    G.removeVertex(3)
    print(G.V)
```

Figure 2: Python code for creating our Graph class [16].

3.1.2. Visual Representation of Graph

The code for displaying visualized Graph, depicting nodes and edges, is shown below. Firstly, we create a adjacency list with 26 nodes, where nodes are major intersection points in the SCC.

```
import networkx as nx
import matplotlib.pyplot as plt

# Create an example adjacency list with 26 nodes
adjacency_list = {
    'amber': {'subid': .9, 'eidgha': 1.1, 'bondor':1.5, 'chowhatta':.6},
    'bondor': {'amber': 1.5, 'jitu_mia': 1.1, 'bg_osmani': .6,
               'zinda':.4, 'mirza':.7, 'kodomtoli': 1.3},
    'chowhatta': {'zinda': .5, 'noya': .6, 'rikabi': .7, 'amber':.6},
    'zinda': {'bondor': .4, 'chowhatta': .5, 'mirza': .5, 'noya': .5},
    'lama': {'jitu_mia': .6, 'rikabi': .3, 'mirza': .3, 'somc': .9},
    'rikabi': {'chowhatta': .7, 'lama': .3, 'subid': .9, 'somc':.9},
    'jitu_mia': {'lama': .6, 'bondor': 1.1, 'kodomtoli': 1.9},
    'subid': {'amber': .9, 'rikabi': .9, 'pathantula': .7},
    'modina': {'pathantula': .7, 'somc': 1.3, 'temukhi': 2.4},
    'pathantula': {'subid': .7, 'modina': .7},
    'temukhi': {'modina': 2.4, 'tuker': 1.3},
    'tuker': {'temukhi': 1.3},
    'somc': {'rikabi': .9, 'lama': .9, 'modina': 1.3},
    'noya': {'chowhatta': .6, 'zinda': .5},
    'mirza': {'lama': .3, 'zinda': .5, 'bondor': .7},
    'bg_osmani': {'bondor': .6, 'upashohor': .9, 'shibganj': 1.5},
    'upashohor': {'bg_osmani': .9},
    'hm_cottor': {'kodomtoli': .4, 'condipul': 2.2},
    'tilagor': {'eidgha': 2.8, 'majortila': 2.6, 'shibganj': 1.6, 'madani': .6},
    'eidgha': {'tilagor': 2.8, 'amber': 1.8},
    'majortila': {'tilagor': 2.6},
    'shibganj': {'bg_osmani': 1.5, 'tilagor': 1.6},
    'condipul': {'hm_cottor': 2.2},
    'baluchor': {'madani': .9},
    'kodomtoli': {'hm_cottor': .4, 'bondor': 1.3, 'jitu_mia': 1.9},
    'madani': {'tilagor': .6, 'baluchor': .9}
```

```
# Create an empty graph object
G = nx.Graph()

G.add_nodes_from(adjacency_list.keys())

# Add edges to the graph with weights
for node, neighbors in adjacency_list.items():
    for neighbor, weight in neighbors.items():
        G.add_edge(node, neighbor, weight=weight)

# Set the figure size
plt.figure(figsize=(25, 10))

# Set positions for the nodes using the spring layout algorithm
pos = nx.spring_layout(G)

# Draw nodes
nx.draw_networkx_nodes(G, pos, node_color='lightblue', node_size=500)

nx.draw_networkx_edges(G, pos)

# Draw edge labels
labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)

nx.draw_networkx_labels(G, pos, font_weight='bold')

plt.axis('off')
plt.show()
```

Figure 3: Python code to display a graph of Sylhet City [16]. This displays name and position of all 26 nodes as well as the distance between each of them.

Visualized Graph is shown in the Results Section.

3.1.3 Traffic Flow Dataset

We have created a noisy, simulated dataset of the traffic flow of SCC. To collect traffic data, we sent volunteers to the main intersection points of SCC, and take pictures of the points every 5 minutes, over an hour. This was done at 2 different points during the day (morning and afternoon) to get a proper picture of the traffic flow of SCC. This was done for 1 day.

Remaining dataset of 9 days was collected by the thesis group members and juniors and some batch-mates.

We have divided the Dataset into 2 sets – **Training set (7 days Data)** and **Test Set (3 days data)**. Training Set is taken for each day of the week (Sunday to Saturday).

We have collected some traffic data over 1 day, which is shown in the Dataset section.

Figure 4: Lama-bazar at 9:30 AM (up) and 9:35 AM (down).

Figure 5: Humayun Chottor at 9:10 AM (up) and 9:20 AM (down).

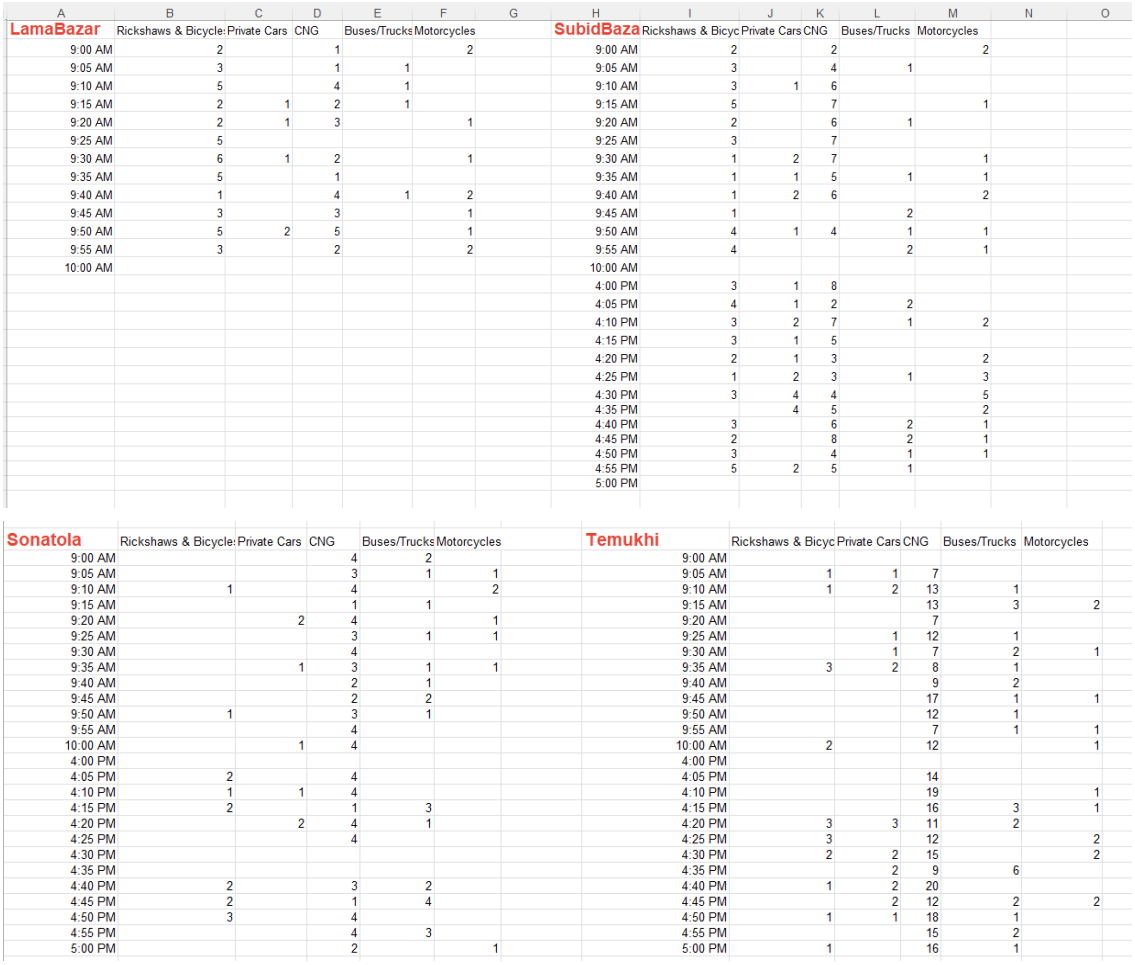
Some pictures were blurry, and it was difficult to manually count the vehicles from each image. This introduced some errors in the dataset. In the future, object detection will be used to count the vehicles from images.

Condipul	Rickshaws & Bicycle	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM			8	2	2
9:05 AM	1		7	2	
9:10 AM		1	6	6	
9:15 AM	1	1	5	1	1
9:20 AM		2	2	5	
9:25 AM	1	1	8	2	
9:30 AM		1	5	4	1
9:35 AM		1	7	4	4
9:40 AM		1	10	4	4
9:45 AM	1		11	1	2
9:50 AM			14	2	4
9:55 AM		1	5	1	2
10:00 AM					
4:00 PM	1	1	3	1	2
4:05 PM	2	1	4	1	1
4:10 PM	1		3	2	2
4:15 PM	1		13	3	2
4:20 PM	1	3	10	1	2
4:25 PM		5	5	2	4
4:30 PM		7	8	3	2
4:35 PM		4	4	1	1
4:40 PM	1	5	7	4	1
4:45 PM	1	4		3	2
4:50 PM	1	1	1	2	2
4:55 PM		5	10	1	1
5:00 PM					

Modina Market	Rickshaws & Bicyc	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	5	1	20		
9:05 AM	3		6	1	2
9:10 AM	4	1	8		1
9:15 AM	9		11		2
9:20 AM	5		9		1
9:25 AM	8		8		
9:30 AM	3	2	6		
9:35 AM	3	1	7		1
9:40 AM	4		9		
9:45 AM	2		7		
9:50 AM	5	3	5		2
9:55 AM	8	1	6	1	
10:00 AM	5		9		2
4:00 PM	2		15		3
4:05 PM	2		9	1	1
4:10 PM	5	3	4	1	
4:15 PM	6		4		
4:20 PM	4	1	2		3
4:25 PM	3	1	6		2
4:30 PM	4	2	5	1	
4:35 PM	7	1	6		2
4:40 PM	4	1	9		2
4:45 PM	1	1	7		2
4:50 PM	3	1	6		1
4:55 PM	3	1	11		
5:00 PM	3	2	3	1	1

Baluchor	Rickshaws & Bicycle	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM					
9:05 AM	1		2		
9:10 AM	1		3		
9:15 AM			2		
9:20 AM	1		2		1
9:25 AM			1		
9:30 AM	1			1	
9:35 AM	1			1	
9:40 AM			3		2
9:45 AM		2	1		
9:50 AM		2	2	1	
9:55 AM	1				3
10:00 AM					
4:00 PM	1	1	4		
4:05 PM		1	2		
4:10 PM	1	2	4		
4:15 PM	3		1	1	
4:20 PM	5		3		2
4:25 PM	2	1	5	1	
4:30 PM	1	3	1		1
4:35 PM	1	1	2		1
4:40 PM		1	2		
4:45 PM			5		1
4:50 PM	1	3			2
4:55 PM		2	4		2
5:00 PM	1		3	1	

Bondor	Rickshaws & Bicyc	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	2	3			1
9:05 AM	4	3	6	5	2
9:10 AM	3		5		3
9:15 AM	1		5	2	
9:20 AM	3		5		
9:25 AM	2	3	6	1	1
9:30 AM	3		8	1	
9:35 AM	4	2	8		2
9:40 AM	3		5	2	1
9:45 AM	4	3	7		
9:50 AM	3	2	10	3	2
9:55 AM	6	2	7	5	2
10:00 AM	7	2	5	6	3
4:00 PM					
4:05 PM					
4:10 PM	5	2	9	4	3
4:15 PM	6	4	12	5	2
4:20 PM	7	5	14	3	2
4:25 PM	6	3	15	4	3
4:30 PM	7	3	14	3	4
4:35 PM	6	1	13		2
4:40 PM	4	5	7	2	1
4:45 PM	3	5	10	3	2
4:50 PM	4	6	11	3	1
4:55 PM	3	5	13	3	3
5:00 PM	5	1	10	2	1



LamaBazar	Rickshaws & Bicycle	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	2		1		2
9:05 AM	3		1	1	
9:10 AM	5		4	1	
9:15 AM	2	1	2	1	
9:20 AM	2	1	3		1
9:25 AM	5				
9:30 AM	6	1	2		1
9:35 AM	5		1		
9:40 AM	1		4	1	2
9:45 AM	3		3		1
9:50 AM	5	2	5		1
9:55 AM	3		2		2
10:00 AM					

SubidBaza	Rickshaws & Bicyc	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	2		2		2
9:05 AM	3		4	1	
9:10 AM	3	1	6		
9:15 AM	5		7		1
9:20 AM	2		6	1	
9:25 AM	3		7		
9:30 AM	1	2	7		1
9:35 AM	1	1	5	1	1
9:40 AM	1	2	6		2
9:45 AM	1			2	
9:50 AM	4	1	4	1	1
9:55 AM	4			2	1
10:00 AM					
4:00 PM	3	1	8		
4:05 PM	4	1	2	2	
4:10 PM	3	2	7	1	2
4:15 PM	3	1	5		
4:20 PM	2	1	3		2
4:25 PM	1	2	3	1	3
4:30 PM	3	4	4		5
4:35 PM		4	5		2
4:40 PM	3		6	2	1
4:45 PM	2		8	2	1
4:50 PM	3		4	1	1
4:55 PM	5	2	5	1	
5:00 PM					

Sonatola	Rickshaws & Bicycle	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM			4	2	
9:05 AM			3	1	1
9:10 AM	1		4		2
9:15 AM			1	1	
9:20 AM		2	4		1
9:25 AM			3	1	1
9:30 AM			4		
9:35 AM		1	3	1	1
9:40 AM			2	1	
9:45 AM			2	2	
9:50 AM	1		3	1	
9:55 AM			4		
10:00 AM		1	4		
4:00 PM					
4:05 PM	2		4		
4:10 PM	1	1	4		
4:15 PM	2		1	3	
4:20 PM		2	4	1	
4:25 PM			4		
4:30 PM					
4:35 PM					
4:40 PM	2		3	2	
4:45 PM	2		1	4	
4:50 PM	3		4		
4:55 PM			4	3	
5:00 PM			2		1

Temukhi	Rickshaws & Bicyc	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM					
9:05 AM	1	1	7		
9:10 AM	1	2	13	1	
9:15 AM			13	3	2
9:20 AM			7		
9:25 AM		1	12	1	
9:30 AM		1	7	2	1
9:35 AM	3	2	8	1	
9:40 AM			9	2	
9:45 AM			17	1	1
9:50 AM			12	1	
9:55 AM			7	1	1
10:00 AM	2		12		1
4:00 PM					
4:05 PM			14		
4:10 PM			19		1
4:15 PM			16	3	1
4:20 PM	3	3	11	2	
4:25 PM	3		12		2
4:30 PM	2	2	15		2
4:35 PM		2	9	6	
4:40 PM	1	2	20		
4:45 PM		2	12	2	2
4:50 PM	1	1	18	1	
4:55 PM			15	2	
5:00 PM	1		16	1	

Figure 6: Traffic flow data on a spreadsheet (Training).

To gather this data, we gathered some volunteers and batch-mates to go 8 intersection points at 9 am and 4 pm. We took pictures of these points at 5-minute intervals, for one hour. The vehicles were manually counted from the images and input into a Google Spreadsheet shown above.

Baluchor	Rickshaws & E	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	1	1	2	1	4
9:05 AM	2		2		
9:10 AM	1		1	1	3
9:15 AM		1	2		
9:20 AM	1		2		1
9:25 AM			1		3
9:30 AM	-1	1		-1	
9:35 AM	1	1	3	-1	-1
9:40 AM			2	1	2
9:45 AM		1	1		1
9:50 AM		-1	-1	1	
9:55 AM	1				3
10:00 AM	2				
4:00 PM	1	1	5		
4:05 PM	-1	1	-1		2
4:10 PM	1	2	2		
4:15 PM	2	1	1	1	
4:20 PM	5		6		1
4:25 PM	-1	1	-1	1	
4:30 PM	4	1	1		1
4:35 PM	1	1	3		1
4:40 PM		-1	2	-1	
4:45 PM	2		5		1
4:50 PM	1	1	6		2
4:55 PM		-1	4	1	-1
5:00 PM	1		3	1	1

Bondor	Rickshaws & E	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	3	2	2	1	1
9:05 AM	2	-1	-1	2	2
9:10 AM	3	1	4		3
9:15 AM	2		3	2	
9:20 AM	3		4		1
9:25 AM	2	-1	-1	1	1
9:30 AM	3		7	1	
9:35 AM	1	2	6		2
9:40 AM	3	-1	5	-1	-1
9:45 AM	4	3	9		
9:50 AM	2	-1	-1	3	-1
9:55 AM	6	2	7	5	2
10:00 AM	7	3	-1	-1	1
4:00 PM					
4:05 PM	2	1	4		4
4:10 PM	5	2	9	2	3
4:15 PM	3	4	11	2	2
4:20 PM	7	-1	14	3	-1
4:25 PM	6	3	2	4	3
4:30 PM	7	3	14	1	3
4:35 PM	4	-1	11		-1
4:40 PM	4	5	7	2	2
4:45 PM	2	6	10	3	2
4:50 PM	4	6	11	3	1
4:55 PM	3	2	13	3	1
5:00 PM	3	-1	11	-1	3

Figure 7: Traffic flow data on a spreadsheet (Testing).

Footage:

We tried to acquire some traffic footage, from CCTV cameras placed at several roads and points across SCC. This footage is owned by Sylhet Metropolitan Police (SMP). We have submitted an application to the Additional Deputy Commissioner (ADC), Rakhi Rani Das, requesting to grant us access to the traffic footage. From these real-world traffic data, we would be able to simulate and generate the rest of the data, using simulation or data augmentation techniques. However, our application was denied and we did not receive the Traffic footage.

A simulation is necessary since a lot of traffic data is necessary to perform prediction using Machine Learning models. Moreover, simulation and data augmentation are rich fields of research, so we believe there is a lot of scope for us to contribute.

3.1.4 Methodology

YOLO Model:

After collecting our Dataset, we realized that manually counting vehicles from images is a long and error-prone task. We therefore decided to automate this process using the YOLO model.

YOLO is a real-time object detection system designed to identify and locate multiple objects in images or video frames.

How YOLO Works:

- YOLO divides the input image into a grid and assigns bounding boxes and class predictions to each grid cell.
- It predicts bounding box coordinates and class probabilities directly, making it a one-stage object detection model.
- The model predicts multiple bounding boxes per grid cell, allowing it to handle over-lapping objects efficiently.

Benefits of YOLO:

- **Speed:** YOLO is known for its real-time processing capability, making it suitable for applications where low latency is crucial.
- **Accuracy:** YOLO performs well in terms of accuracy, especially in detecting and localizing multiple objects in an image or video frame.
- **Simplicity:** YOLO's single pass architecture simplifies the object detection process compared to two-stage detectors.

We applied the YOLO model on a Dataset of Dhaka's roads and vehicles, available on Kaggle [10]. The results showed the YOLO model is able to detect and classify the vehicles with a high degree of accuracy. The results are shown in the results section.

Figure 8: Training images (Up) and Validation output (Down).

Denoising Stacked Auto-Encoder (DSAE) model:

We are studying research papers based on Traffic data simulation and augmentation/imputation. There are primarily 3 methods for simulating data:

i) **Model-driven method** – Takes into account vehicle velocity, traffic flow, density, etc to calculate traffic flow using statistical or physics equations.
ii) **Data-driven method** – Uses historical traffic data to generate traffic flow information using Machine Learning models
iii) **Streaming-data-driven method** – Uses live traffic data to make traffic predictions.

Model-driven methods usually require vehicle speeds, which we do not have the resources to calculate. Therefore, our preliminary approach is a **data-driven** model using Denoising Stacked Auto-Encoder (**DSAE**), which can be used to generate missing or corrupted traffic data.[9]

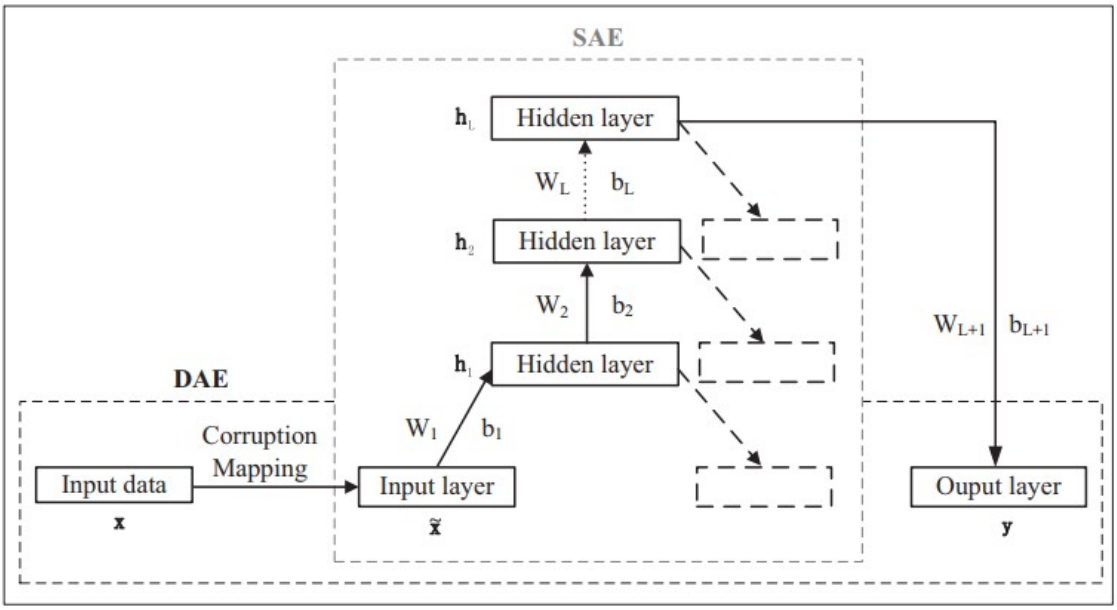


Figure 9: Structure of DSAE.

The encoder of an **AE** (auto-encoder) maps an input vector, **x**, into a hidden representation, which is a non-linear transformation.
A **DAE** (Denoising Auto Encoder) is very similar to AE, except it accounts for corrupted traffic information, by taking in a corrupted version of the original input, mapped onto the input layer. Through encoding and decoding, the output, a reconstructed vector, **y**, is obtained.

The reconstruction error is minimized by solving the following optimization problem:

$$(\theta, \theta') = \arg\min_{(\theta,\theta')} L(X, Y),$$

Algorithm 1. Training an autoencoder

Require: data set $X = \{\mathbf{x}\}, \mathbf{x} \in \mathbb{R}^N$, the number of hidden nodes n_h, the iterations T
1: **Initialization:** initialize the weights and biases randomly: $\mathbf{W}(n_h \times N), \mathbf{W}'(N \times n_h), \mathbf{b}(1 \times n_h), \mathbf{b}'(1 \times N)$;
2: **for** i = 1 to T **do**
3. Perform forward propagation to compute Y
4. Compute output error: $X - Y$
5. Perform backward propagation to compute $\Delta\theta, \Delta\theta'$
6: Update $\theta, \theta' : \theta = \theta + \Delta\theta, \theta' = \theta' + \Delta\theta'$
7: **end for**

Algorithm 2. Training denoising stacked autoencoders

Require: data set $X = \{\mathbf{x}\}, \mathbf{x} \in \mathbb{R}^N$, the number of hidden layers L, the number of hidden nodes $n_l, l = 1, 2, \ldots, L$, the pretraining iterations T, the fine-tuning iterations T'
1: **Stochastic mapping:** map X into $\widetilde{X}, \widetilde{X} = \{\widetilde{\mathbf{x}}\}, \widetilde{\mathbf{x}} \in \mathbb{R}^N$;
Step 1: Pretraining
2: $\widetilde{X}$ is fed to the input layer;
3: Train the first AE using Algorithm 1;
4: **for** l = 2 to L **do**
5: H_{l-1} is fed to the lth AE
6: Train the lth AE using Algorithm 1
7: **end for**
Step 2: Fine-tuning
8: **Initialization:** initialize $\mathbf{W}_{L+1}(N \times n_L), \mathbf{b}(1 \times N)$ randomly;
9: **for**i = 1 to T' **do**
10: Perform forward propagation to compute Y
11: Compute output error: $X - Y$
12: Perform backward propagation to compute $\Delta\Theta$
13: Update $\Theta : \Theta = \Theta + \Delta\Theta$
14: **end for**

Figure 10: Training Algorithms for the DSAE method [9].

Firstly, random noise is added to the Input Data for corruption. The corrupted Data is passed into the DSAE model, and Original Data is given as output. The DSAE model trains itself to map the corrupted input into the original input.

Implementation of DSAE:

```
## Upload files

from google.colab import files
uploaded = files.upload()
```

```
## Create pandas dataframe

import numpy as np
import pandas as pd

df = pd.read_excel('traffic.xlsx')
df1 = pd.read_excel('Day1_new.xlsx')
df2 = pd.read_excel('Day2_new.xlsx')
df3 = pd.read_excel('Day3_new.xlsx')
df4 = pd.read_excel('Day4_new.xlsx')
df5 = pd.read_excel('Day5_new.xlsx')
df6 = pd.read_excel('Day6_new.xlsx')
df7 = pd.read_excel('Day7_new.xlsx')
df8 = pd.read_excel('Day8_new.xlsx')
df_test1 = pd.read_excel('Test1_new.xlsx')
df_test1_or = pd.read_excel('Test1_original.xlsx')
```

Figure 11: Upload and Create data-frame.

All the Training and Test Dataset of Traffic flow are loaded into Pandas Dataframes, from Excel files. Each Dataset has **5 rows** and **219 columns**. Total **1095** data values in each Dataset. There are **10 total Datasets**, equaling **10,950** data values.

```
[ ]  ## Imputed null value in missing values

     df = df.fillna(0)
     df1 = df1.fillna(0)
     df2 = df2.fillna(0)
     df3 = df3.fillna(0)
     df4 = df4.fillna(0)
     df5 = df5.fillna(0)
     df6 = df6.fillna(0)
     df7 = df7.fillna(0)
     df8 = df8.fillna(0)
     df_test1 = df_test1.fillna(0)
     df_test1_or = df_test1_or.fillna(0)
     df8[:10]
```

	2	2.1	Unnamed: 2	Unnamed: 3	1
0	2.0	0.0	2.0	1.0	3.0
1	3.0	0.0	4.0	1.0	4.0
2	2.0	2.0	2.0	1.0	0.0
3	4.0	1.0	1.0	0.0	1.0
4	3.0	0.0	5.0	0.0	0.0

Figure 12: Using fillna () to add null values.

The null values represent no vehicles present at that moment in time. The null values are replaced with 0.

```
## Create an numpy array

arr = np.array(df)
arr1 = np.array(df1)
arr2 = np.array(df2)
arr3 = np.array(df3)
arr4 = np.array(df4)
arr5 = np.array(df5)
arr6 = np.array(df6)
arr7 = np.array(df7)
arr8 = np.array(df8)
arr_test1 = np.array(df_test1)
arr_test1_or = np.array(df_test1_or)
```

```
[ ] ## Add flatten

flat = arr.flatten('C')
flat1 = arr1.flatten('C')
flat2 = arr2.flatten('C')
flat3 = arr3.flatten('C')
flat4 = arr4.flatten('C')
flat5 = arr5.flatten('C')
flat6 = arr6.flatten('C')
flat7 = arr7.flatten('C')
flat8 = arr8.flatten('C')
```

Figure 13: Create Numpy array and add flatten.

The dataframes are in a 2-dimensional format, but the input needs to be 1-dimensional array. So, the array is flattened using numpy.

```
## Randomly noise add
# 30% of the data will be given random noise
# 328 data is noisy

import random
sz = output.shape[0]
print(sz)

x = int(sz*0.3)
# print(x)

noisy_indexes = random.sample(range(sz), x)
print(noisy_indexes)
```

```
1095
[151, 1048, 82, 658, 184, 118, 242, 145, 874, 249, 1044, 856, 167, 270, 478, 78, 971, 598, 990, 544, 552, 323, 1012, 437, 22,
```

Figure 14: Randomly Noise add

The input data needs to be corrupted before passing into the DSAE model. Corruption mapping is done by randomly selecting **30%** of the data (**328**) and changing them to **-1**. This is done using the random module in python.

```
#Create model

model = Sequential()
model.add(Dense(2000, input_dim=wrappedinput.shape[1], activation='relu'))
model.add(Dense(1000, activation='relu'))
model.add(Dense(500, activation='relu'))
model.add(Dense(1095))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(wrappedinput,wrappedoutput,verbose=0,batch_size=len(wrappedinput),epochs=10000)

pred = model.predict(wrappedinput)

print("Actual")
print(wrappedoutput)

print("Pred")
print(pred)
```

Figure 15: Create DSAE Model.

The **DSAE** model consists of Stacked Autoencoders, and a Decoder as output. There are 3 hidden layers. 1st layer has **2000** neurons, 2nd layer has **1000** neurons, 3rd layer has **500**. Output layer has same number as neurons as dataset (**1095**). Activation function is **ReLu**(Rectified Linear Unit). Optimizer is **Adam**. Training done with **10,000** epochs.

```
[ ] # Connect to google drive

    from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
    ## Load model
    import pickle
    model = pickle.load(open('/content/drive/MyDrive/Thesis Output/model.pkl', 'rb'))
    model1 = pickle.load(open('/content/drive/MyDrive/Thesis Output/model1.pkl', 'rb'))
    model2 = pickle.load(open('/content/drive/MyDrive/Thesis Output/model2.pkl', 'rb'))
    model3 = pickle.load(open('/content/drive/MyDrive/Thesis Output/model3.pkl', 'rb'))
    model4 = pickle.load(open('/content/drive/MyDrive/Thesis Output/model4.pkl', 'rb'))
    model6 = pickle.load(open('/content/drive/MyDrive/Thesis Output/model6.pkl', 'rb'))
    model7 = pickle.load(open('/content/drive/MyDrive/Thesis Output/model7.pkl', 'rb'))
```

```
[ ] import pickle
    pickle.dump(model, open('model.pkl', 'wb'))
    !cp model.pkl /content/drive/MyDrive/Thesis-Output
```

Figure 16: Load Model.

7 models were Trained using the 7 Training Datasets. The trained models were saved to Google Drive, and then loaded again for prediction.

```
[ ] #Using model7 to predict test set wrappedinput8

    pred8 = model7.predict(wrappedinput8)

    1/1 [==============================] - 0s 172ms/step
```

```
[ ] ## Measure Accuracy

    from sklearn import metrics
    score = np.sqrt(metrics.mean_squared_error(pred8, wrappedoutput8))
    print("Score (RMSE): {}".format(score))

    score1 = np.sqrt(metrics.mean_absolute_error(pred8, wrappedoutput8))
    print("Score (MAE): {}".format(score1))

    Score (RMSE): 2.1015376094167473
    Score (MAE): 1.161046982609224
```

Figure 17: Test Set 1 Prediction.

Test Set 1 was predicted using Model 7. The Table above shows the Root Mean Squared Error (RMSE) and Mean Average Error (MAE) compared with Original Data.

```
excelout = pd.DataFrame(pred8, columns=['Rickshaw/Bicycle', 'Private Car', 'CNG', 'Bus/Truck','Motorcycle'])
excelout
```

	Rickshaw/Bicycle	Private Car	CNG	Bus/Truck	Motorcycle
0	1.0	0.0	1.0	0.0	1.0
1	2.0	0.0	2.0	0.0	2.0
2	0.0	1.0	2.0	1.0	0.0
3	1.0	1.0	1.0	0.0	1.0
4	2.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...
214	0.0	0.0	3.0	1.0	0.0
215	1.0	0.0	0.0	0.0	0.0
216	2.0	1.0	1.0	0.0	1.0
217	0.0	1.0	1.0	0.0	1.0
218	2.0	0.0	2.0	1.0	0.0

219 rows × 5 columns

Figure 18: Test Set 1 Prediction Excel.

The predicted values are saved in an Excel Spreadsheet.

```
#Using model to predict test set Test1_new

pred9 = model7.predict(output_test1)
```

```
1/1 [==============================] - 0s 39ms/step
```

```
from sklearn import metrics
score = np.sqrt(metrics.mean_squared_error(pred9, output_test1_or))
print("Score (RMSE): {}".format(score))

score1 = np.sqrt(metrics.mean_absolute_error(pred9, output_test1_or))
print("Score (MAE): {}".format(score1))
```

```
Score (RMSE): 2.343664449963408
Score (MAE): 1.266831539952977
```

Figure 19: Test Set 2 Prediction.

Test Set 2 was predicted using Model 7. The Table above shows the Root Mean Squared Error (RMSE) and Mean Average Error (MAE) compared with Original Data.

```
excelout1 = pd.DataFrame(pred9, columns=['Rickshaw/Bicycle', 'Private Car', 'CNG', 'Bus/Truck','Motorcycle'])
excelout1
```

	Rickshaw/Bicycle	Private Car	CNG	Bus/Truck	Motorcycle
0	1.0	0.0	1.0	0.0	1.0
1	1.0	0.0	2.0	0.0	2.0
2	1.0	0.0	2.0	1.0	0.0
3	1.0	1.0	1.0	0.0	2.0
4	1.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...
214	0.0	0.0	2.0	0.0	0.0
215	1.0	0.0	0.0	1.0	0.0
216	2.0	0.0	1.0	0.0	1.0
217	0.0	1.0	1.0	1.0	1.0
218	2.0	0.0	2.0	1.0	0.0

219 rows × 5 columns

Figure 20: Test Set 2 Prediction Excel.

The predicted values are saved in an Excel Spreadsheet.

Chapter 4

RESULTS AND DISCUSSION

Data Collection and Results so far:

i) Graph Visualization.

ii) City Points Distance.

iii) YOLO results.

iv) Traffic Flow Data.

v) Results of Missing Data Imputation using DSAE.

vi) Comparing Results from Test Set 1 and Test Set 2.

4.1 Graph Visualization

Using the code from Fig 3, the visualized graph below shows major intersection points and the distance between them.

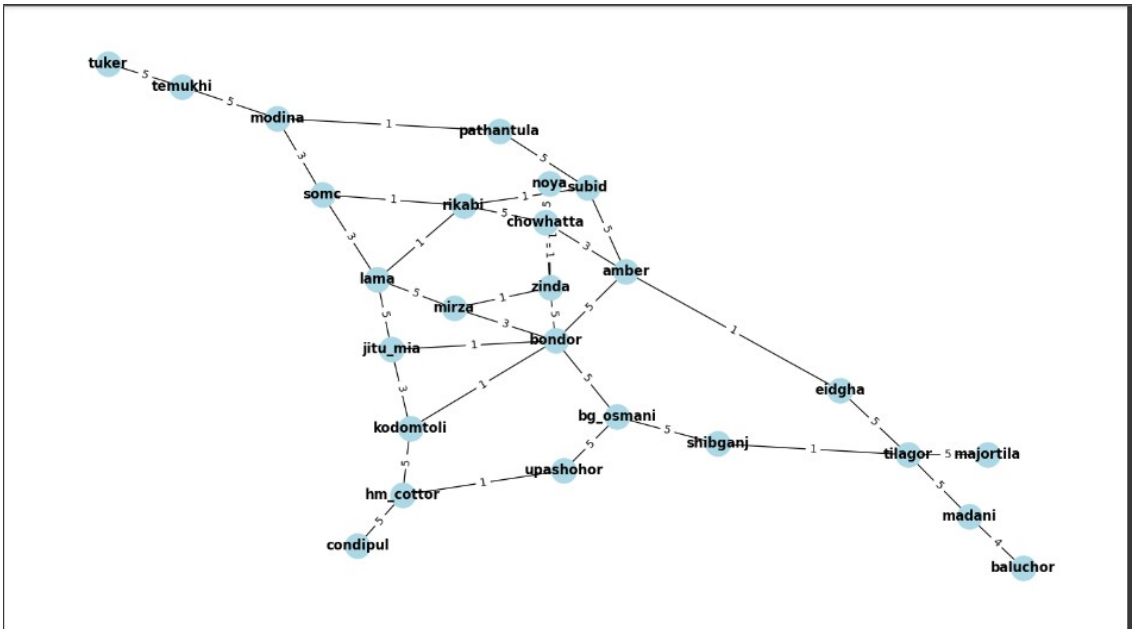


Figure 21: Visual Graph of Sylhet City [16].

Here, Nodes represent the major intersection points, and weighted edges represent the distance between major intersection points.

4.2 City Points Distance

Using Google Maps, records the distance between the major intersection points in the SCC is shown below:

	Amberkhan	Baluchor	Bondor	Bongobir Osmani	Chowhatta	Chowkidekhi	Eidgah	Jitu Miar Point	Kodomtoli	LamaBazar	Somc	Mirzajangal	Modina-Market
Amberkhana	0		1450		640								
Baluchor		0											
Bondor	1450		0	599				1070				683	
Bongobir Osmani			599	0									
Chowhatta	640				0								
Eidgah							0						
Jitu Miar Point			1070					0	1910	636			
Kodomtoli			1320					1910	0				
LamaBazar								636		0	908	340	
Somc										908	0		1280
Mirzajangal			683							340		0	
Modina-Market											1280		0
Noyasarak					604								
Pathantula													662
RikabiBazar					651					342	925		
Shibgonj													
Subid Bazar	946												
Condipol													
Temukhi													2400
Tilaghor													
Tukerbazar													
Upashohor				897									
ZindaBazar			368		493							462	
Majortila													
Humayun Cottor									414				

Somc	Mirzajangal	Modina-Market	Noyasarak	Pathantula	RikabiBazar	Shibgonj	Subid Bazar	Condipol	Temukhi	Tilaghor	Tukerbazar	Upashohor	ZindaBazar	Majortila	Humayun Cottor
							946								
	683												368		
						1490						897			
			604		651								480		
															414
908	340				342										
0		1280			925										
	0												462		
1280		0		662					2400						
			0										510		
		662		0			747								
925					0		918								
						0									
				747	918		0								
								0							2170
		2400							0		1260				
						977				0					
									1260		0				
												0			
	462		510										0		
								2170							
														0	

Figure 22: Distance between neigh-boring points [16].

4.3 YOLO Results

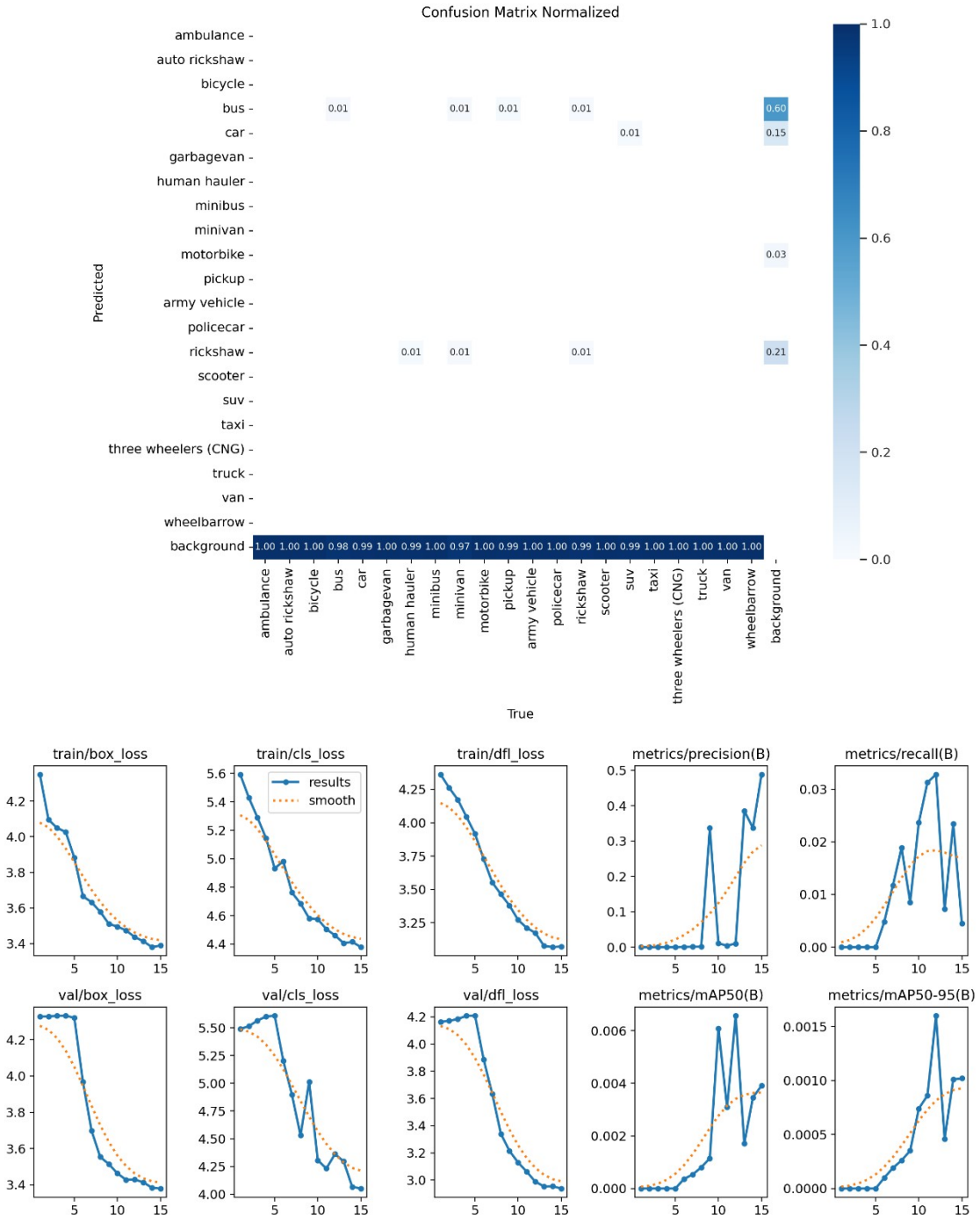


Figure 23: Confusion Matrix (Up) and Training Results (Down).

Here, Figure 12 shows the classification of the vehicles into different categories (Rickshaw, CNG, bus, car, etc.).

Figure 13 shows the graphs during training and validation, showing loss and accuracy metrics.

4.4 Traffic Flow Dataset

Pictures of several intersection points were taken on [date]. The points covered are Lamabazar, Bondor, Zindabazar, Amberkhana, Temukhi, Sonatala, Condipul, Baluchor, Subidbazar. 12 pictures were taken at 5-minute intervals, from 9 am to 10 am, and another 12 pictures from 4 pm to 5 pm. From the pictures, the number of vehicles at each period was calculated, for all the points. The results are tabulated below in the spreadsheet.

Total 10 days Data collected. 1 day's Dataset was collected by volunteers (juniors from CSE Department). The rest of Dataset was collected by the thesis members and some juniors and batch-mates.
Dataset divided into Training Set (7 days) and Test Set (3 days). The Test set has several values missing to allow the DSAE Model to perform Data Imputation on it.

Baluchor	Rickshaws & Bicycle	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM					
9:05 AM	1		2		
9:10 AM	1		3		
9:15 AM			2		
9:20 AM	1		2		1
9:25 AM			1		
9:30 AM	1			1	
9:35 AM	1			1	
9:40 AM			3		2
9:45 AM		2	1		
9:50 AM		2	2	1	
9:55 AM	1				3
10:00 AM					
4:00 PM	1	1	4		
4:05 PM		1	2		
4:10 PM	1	2	4		
4:15 PM	3		1	1	
4:20 PM	5		3		2
4:25 PM	2	1	5	1	
4:30 PM	1	3	1		1
4:35 PM	1	1	2		1
4:40 PM		1	2		
4:45 PM			5		1
4:50 PM	1	3			2
4:55 PM		2	4		2
5:00 PM	1		3	1	

Bondor	Rickshaws & Bicyc	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	2	3			1
9:05 AM	4	3	6	5	2
9:10 AM	3		5		3
9:15 AM	1		5	2	
9:20 AM	3		5		
9:25 AM	2	3	6	1	1
9:30 AM	3		8	1	
9:35 AM	4	2	8		2
9:40 AM	3		5	2	1
9:45 AM	4	3	7		
9:50 AM	3	2	10	3	2
9:55 AM	6	2	7	5	2
10:00 AM	7	2	5	6	3
4:00 PM					
4:05 PM					
4:10 PM	5	2	9	4	3
4:15 PM	6	4	12	5	2
4:20 PM	7	5	14	3	2
4:25 PM	6	3	15	4	3
4:30 PM	7	3	14	3	4
4:35 PM	6	1	13		2
4:40 PM	4	5	7	2	1
4:45 PM	3	5	10	3	2
4:50 PM	4	6	11	3	1
4:55 PM	3	5	13	3	3
5:00 PM	5	1	10	2	1

Condipul	Rickshaws & Bicycle	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM			8	2	2
9:05 AM	1		7	2	
9:10 AM		1	6	6	
9:15 AM	1	1	5	1	1
9:20 AM		2	2	5	
9:25 AM	1	1	8	2	
9:30 AM		1	5	4	1
9:35 AM		1	7	4	4
9:40 AM		1	10	4	4
9:45 AM	1		11	1	2
9:50 AM			14	2	4
9:55 AM		1	5	1	2
10:00 AM					
4:00 PM	1	1	3	1	2
4:05 PM	2	1	4	1	1
4:10 PM	1		3	2	2
4:15 PM	1		13	3	2
4:20 PM	1	3	10	1	2
4:25 PM		5	5	2	4
4:30 PM		7	8	3	2
4:35 PM		4	4	1	1
4:40 PM	1	5	7	4	1
4:45 PM	1	4		3	2
4:50 PM	1	1	1	2	2
4:55 PM		5	10	1	1
5:00 PM					

Modina Market	Rickshaws & Bicyc	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	5	1	20		
9:05 AM	3		6	1	2
9:10 AM	4	1	8		1
9:15 AM	9		11		2
9:20 AM	5		9		1
9:25 AM	8		8		
9:30 AM	3	2	6		
9:35 AM	3	1	7		1
9:40 AM	4		9		
9:45 AM	2		7		
9:50 AM	5	3	5		2
9:55 AM	8	1	6	1	
10:00 AM	5		9		2
4:00 PM	2		15		3
4:05 PM	2		9	1	1
4:10 PM	5	3	4	1	
4:15 PM	6		4		
4:20 PM	4	1	2		3
4:25 PM	3	1	6		2
4:30 PM	4	2	5	1	
4:35 PM	7	1	6		2
4:40 PM	4	1	9		2
4:45 PM	1	1	7		2
4:50 PM	3	1	6		1
4:55 PM	3	1	11		
5:00 PM	3	2	3	1	1

Sonatola	Rickshaws & Bicycle	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM			4	2	
9:05 AM			3	1	1
9:10 AM	1		4		2
9:15 AM			1	1	
9:20 AM		2	4		1
9:25 AM			3	1	1
9:30 AM			4		
9:35 AM		1	3	1	1
9:40 AM			2	1	
9:45 AM			2	2	
9:50 AM	1		3	1	
9:55 AM			4		
10:00 AM		1	4		
4:00 PM					
4:05 PM	2		4		
4:10 PM	1	1	4		
4:15 PM	2		1	3	
4:20 PM		2	4	1	
4:25 PM			4		
4:30 PM					
4:35 PM					
4:40 PM	2		3	2	
4:45 PM	2		1	4	
4:50 PM	3		4		
4:55 PM			4	3	
5:00 PM			2		1

Temukhi	Rickshaws & Bicyc	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM					
9:05 AM	1	1	7		
9:10 AM	1	2	13	1	
9:15 AM			13	3	2
9:20 AM			7		
9:25 AM		1	12	1	
9:30 AM		1	7	2	1
9:35 AM	3	2	8	1	
9:40 AM			9	2	
9:45 AM			17	1	1
9:50 AM			12	1	
9:55 AM			7	1	1
10:00 AM	2		12		1
4:00 PM					
4:05 PM			14		
4:10 PM			19		1
4:15 PM			16	3	1
4:20 PM	3	3	11	2	
4:25 PM	3		12		2
4:30 PM	2	2	15		2
4:35 PM		2	9	6	
4:40 PM	1	2	20		
4:45 PM		2	12	2	2
4:50 PM	1	1	18	1	
4:55 PM			15	2	
5:00 PM	1		16	1	

LamaBazar	Rickshaws & Bicycle	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	2		1		2
9:05 AM	3		1	1	
9:10 AM	5		4	1	
9:15 AM	2	1	2	1	
9:20 AM	2	1	3		1
9:25 AM	5				
9:30 AM	6	1	2		1
9:35 AM	5		1		
9:40 AM	1		4	1	2
9:45 AM	3		3		1
9:50 AM	5	2	5		1
9:55 AM	3		2		2
10:00 AM					

SubidBaza	Rickshaws & Bicyc	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	2		2		2
9:05 AM	3		4	1	
9:10 AM	3	1	6		
9:15 AM	5		7		1
9:20 AM	2		6	1	
9:25 AM	3		7		
9:30 AM	1	2	7		1
9:35 AM	1	1	5	1	1
9:40 AM	1	2	6		2
9:45 AM	1			2	
9:50 AM	4	1	4	1	1
9:55 AM	4			2	1
10:00 AM					
4:00 PM	3	1	8		
4:05 PM	4	1	2	2	
4:10 PM	3	2	7	1	2
4:15 PM	3	1	5		
4:20 PM	2	1	3		2
4:25 PM	1	2	3	1	3
4:30 PM	3	4	4		5
4:35 PM		4	5		2
4:40 PM	3		6	2	1
4:45 PM	2		8	2	1
4:50 PM	3		4	1	1
4:55 PM	5	2	5	1	
5:00 PM					

Figure 24: Traffic flow data from Training Dataset.

Baluchor	Rickshaws & E	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	1	1	2	1	4
9:05 AM	2		2		
9:10 AM	1		1	1	3
9:15 AM		1	2		
9:20 AM	1		2		1
9:25 AM			1		3
9:30 AM	-1	1		-1	
9:35 AM	1	1	3	-1	-1
9:40 AM			2	1	2
9:45 AM		1	1		1
9:50 AM		-1	-1	1	
9:55 AM	1				3
10:00 AM	2				
4:00 PM	1	1	5		
4:05 PM	-1	1	-1		2
4:10 PM	1	2	2		
4:15 PM	2	1	1	1	
4:20 PM	5		6		1
4:25 PM	-1	1	-1	1	
4:30 PM	4	1	1		1
4:35 PM	1	1	3		1
4:40 PM		-1	2	-1	
4:45 PM	2		5		1
4:50 PM	1	1	6		2
4:55 PM		-1	4	1	-1
5:00 PM	1		3	1	1

Bondor	Rickshaws & E	Private Cars	CNG	Buses/Trucks	Motorcycles
9:00 AM	3	2	2	1	1
9:05 AM	2	-1	-1	2	2
9:10 AM	3	1	4		3
9:15 AM	2		3	2	
9:20 AM	3		4		1
9:25 AM	2	-1	-1	1	1
9:30 AM	3		7	1	
9:35 AM	1	2	6		2
9:40 AM	3	-1	5	-1	-1
9:45 AM	4	3	9		
9:50 AM	2	-1	-1	3	-1
9:55 AM	6	2	7	5	2
10:00 AM	7	3	-1	-1	1
4:00 PM					
4:05 PM	2	1	4		4
4:10 PM	5	2	9	2	3
4:15 PM	3	4	11	2	2
4:20 PM	7	-1	14	3	-1
4:25 PM	6	3	2	4	3
4:30 PM	7	3	14	1	3
4:35 PM	4	-1	11		-1
4:40 PM	4	5	7	2	2
4:45 PM	2	6	10	3	2
4:50 PM	4	6	11	3	1
4:55 PM	3	2	13	3	1
5:00 PM	3	-1	11	-1	3

Figure 25: Traffic flow data from Test Dataset.

4.5 Missing Data Imputation Result

	Predicted Values					Actual Values				
	Rickshaw/Bicycle	Private Car	CNG	Bus/Truck	Motorcycle	Rickshaw/Bicycle	Private Car	CNG	Bus/Truck	Motorcycle
0	2	0	2	1	2	2	2			1
1	3	0	4	0	4	2		2	1	3
2	1	1	4	1	0	3		4	1	4
3	2	2	3	0	3	2	2	2	1	
4	3	0	0	0	0	4	1	1		1
5	5	2	2	0	1	3		5		
6	6	2	1	1	3	3	1	3		1
7	3	1	5	1	5	1	2	1		2
8	3	0	3	0	2	1		6	1	6
9	1	2	3	0	1		2	3		1
10	3	0	2	0	3	2	2	3		1
11	1	1	1	0	1	3		3	2	2
12	1	1	3	0	3	1	1	2		1
13	3	0	4	1	0	2	1	2	1	3
14	2	1	5	0	1	3	1	5	1	
15	5	0	7	0	1	2	1	7		2
16	3	0	8	1	0	5		7		1
17	3	0	7	0	1	2	2	6	1	

Figure 26: Test Set 1 Prediction.

	Predicted Values					Actual Values				
	Rickshaw/Bicycle	Private Car	CNG	Bus/Truck	Motorcycle	Rickshaw/Bicycle	Private Car	CNG	Bus/Truck	Motorcycle
0	1	0	1	0	1	2	2			1
1	1	0	2	0	2	2		2	1	3
2	1	0	2	1	0	3		4	1	4
3	1	1	1	0	2	4	2	3	1	
4	1	0	0	0	0	4	1	1		1
5	2	1	1	0	0	3		5		
6	3	1	0	0	2	2	1	3		1
7	2	0	3	0	2	1	2	1		2
8	2	0	1	0	1	1		6	1	6
9	0	1	2	0	0		2	3		1
10	1	0	1	0	1	2	1	3		3
11	1	1	0	0	1	3		3	2	2
12	1	0	2	0	2	1	2	2		3
13	1	0	2	0	0	2	1	2	1	3
14	1	0	3	0	0	2	1	3	1	
15	3	0	4	0	1	2	1	7		2
16	1	0	5	1	0	5		7		1
17	2	0	3	0	1	2	2	6	1	
18	1	1	3	0	1	3		7		
19	2	1	3	0	0	1	3	8		1

Figure 27: Test Set 2 Prediction.

The two tables above show the Predicted Traffic flow (left) and Actual Traffic Flow (right). The values are for different intersection points taken at 9 am and 4 pm.

4.6 Comparing Results

Test Set 1 had RMSE of 2.10 and MAE of 1.16. This represents a deviation of around 2 vehicles at each point on average. Meaning that, the results are accurate to predicting the traffic flow within 2 vehicles at each intersection point.

Test Set 2 had RMSE of 2.34 and MAE of 1.26. This is slightly higher error than Test Set 1, however, the deviation is still around 2 vehicles at each point on average.

For both Test sets, the predictions are fairly accurate. The model is able to take an input that has missing or corrupted values, and predict the traffic flow within a reasonable accuracy.

Collected further data for each day will improve the accuracy.

Error Metrics	Test Set 1	Test Set 2
RMSE	2.10	2.34
MAE	1.16	1.26

Table 1: Accuracy Metrics for Test Data

Chapter 5

CONCLUSION

Our work involved collecting Spatio-Temporal traffic data, which is a novel work that has not been done before in Sylhet. Our simulated dataset can be used for traffic prediction tasks, as well as other downstream tasks such as route planning, emergency vehicle dispatching, etc.

Firstly, we created a Graph of Sylhet City Corporation, showing major intersection points and distance between them. We also tabulated the distance between each point in a separate table. We showed a visual representation of Sylhet City Corporation graph.

We have collected 10 days Data and used a DSAE model to simulate further Data from missing or corrupted values. This model can be used to simulate Data for further Data Predictions, with a reasonable accuracy.

5.1 Limitations

i) Small Dataset. Around 6 months traffic data, throughout the whole day, needs to be collected to generate a large Dataset.
ii) Traffic footage is available from CCTV cameras at some points in SCC. If SMP grants access to the footage, a very accurate Dataset can be generated.
iii) Manually counting vehicles from a large dataset of images or videos will be very time consuming and error-prone. Here, YOLO model can be used for automatic vehicle detection.

5.2 Future Works

We collected 10 days of traffic data, at 2 periods, 9 am and 4 pm. In the future, around 6 months data needs to be collected to generate a large, accurate traffic Dataset of SCC. YOLO model can then be used to easily detect and count vehicles from the images and videos. The DSAE model presented in this paper provided very good results, and it can be improved further, by adding more Auto-Encoder layers. To create a Spatio-Temporal representation, other parameters such as vehicle speed, demand, road size needs to be calculated.

References

[1] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi, and B. Yin, "Deep Learning on Traffic Prediction: Methods, Analysis and Future Directions' Transactions on Intelligent Transportation Systems (Volume: 23, Issue: 6, June 2022). https://arxiv.org/pdf/2004.08555

[2] Q. Chao, H. Bi, W. Li, T. Mao, Z. Wang, Ming C. Lin, and Z. Deng, "A Survey on Visual Traffic Simulation: Models, Evaluations, and Applications in Autonomous Driving, "COMPUTER GRAPHICS forum Vol – 39 (2020), number 1 pp. 287–308. https://huikunbi.github.io/research/2019%20EG_STAR___A_Survey_on_Visual_Traffic_Simulation_and_Animation.pdf

[3] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," Procs of the 27th International Joint Conference on Artificial Intelligence. https://arxiv.org/pdf/1709.04875

[4] C Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," 34th AAAI Conference (AAAI-20). https://ojs.aaai.org/index.php/AAAI/article/view/5438/5294

[5] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", https://pjreddie.com/darknet/yolo/

[6] T. Seo, A. M. Bayen, T. Kusakabe, and Y. Asakura "Traffic state estimation on highway: A comprehensive survey", Elsevier, Vol 43, 2017.https://www.researchgate.net/profile/Toru-Seo/publication/315784652_Traffic_state_estimation_on_highway_A_comprehensive_survey/links/59cd7605aca272b0ec14fe60/Traffic-state-estimation-on-highway-A-comprehensive-survey.pdf

[7] G. Bretti, R. Natalini, and B. Piccoli, "Numerical Approximations of a traffic flow model on networks" Networks and Heterogeneous Media, 2006. https://www.researchgate.net/profile/Gabriella-Bretti/publication/220519874_Numerical_approximations_of_a_traffic_flow_model_on_networks/links/0912f50e81798823bd000000/Numerical-approximations-of-a-traffic-flow-model-on-networks.pdf

[8] A. Abadi, T. Rajabioun, and P. A. Ioannou “Traffic Flow Prediction for Road Transportation Networks with Limited Traffic Data”, IEEE Transactions on intelligent transportation systems, vol. 16, no. 2, April 2015. https://sci-hub.ru/10.1109/TITS.2014.2337238

[9] Y. Duan, Y. Lv, Y. Liu, F. Wang “An efficient realization of deep learning for traffic data imputation”, Transportation Research Part C: Emerging Technologies in 2016. https://sci-hub.ru/10.1016/j.trc.2016.09.015

[10] L. Li, B. Du, Y. Wang, L. Q. e, H. Tan, “Estimation of missing values in heterogeneous traffic data: Application of multimodal deep learning model”, Elsevier, Knowledge-Based Systems, Vol 194, 22 April 2020. https://sci-hub.ru/10.1016/j.knosys.2020.105592

[11] T. Nishi, K. Otaki, K. Hayakawa and T. Yoshimura, “Traffic Signal Control Based on Reinforcement Learning with Graph Convolutional Neural Nets’’, 2018 21st International Conference on Intelligent Transportation Systems (ITSC) Maui, Hawaii, USA, November 4-7, 2018. https://sci-hub.se/10.1109/itsc.2018.8569301

[12] Z. xie, W. Lv, S. Huang, Z. Lu, B. Du, and R. Huang, “Sequential Graph Neural Network for Urban Road Traffic Speed Prediction”, IEEE Access Vol:8, 2019. https://sci-hub.se/10.1109/access.2019.2915364

[13] F. Kocayusufoglu, A. Silva, A. Singh, “FlowGEN: A Generative Model for Flow Graphs”, Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery, August 2022, Pages 813-823. https://doi.org/10.1145/3534678.3539406

[14] Y. Xue, Y. Zhang, D. Fan, P. Zhang, H. He, “An extended macroscopic model for traffic flow on curved road and its numerical simulation’’, Springer Nature B.V. 2019. https://sci-hub.se/10.1007/s11071-018-04756-y

[15] J. Zhao, Y. Nie, S. Ni, and X. Sun, “Traffic Data Imputation and Prediction: An Efficient Realization of Deep Learning”, IEEE Access, Vol:8. https://sci-hub.se/10.1109/access.2020.2978530

[16] https://github.com/Jahid234/Towards-the-Creation-and-Spatio-temporal-Representation-of-a-Simulated-Traffic-Dataset-of-Sylhet