

“TRAFFIC PREDICTION USING DEEP LEARNING”

Major Project Report

Submitted in Partial Fulfillment of the
Requirements for the degree of

BACHELOR OF TECHNOLOGY IN ELECTRONICS & COMMUNICATION ENGINEERING

By

Avantika Sharma (16BEC015)

Parth Modi (16BEC092)

Under the Guidance of
Dr. Ruchi Gajjar



**Department of Electronics and Communication Engineering
Institute of Technology, Nirma University
Ahmedabad 382 481
May 2020**

CERTIFICATE

This is to certify that the Major Project Report entitled “**Traffic Prediction using Deep Learning**” submitted by **Ms. Avantika Sharma (16BEC015)** and **Mr. Parth Modi (16BEC092)** towards the partial fulfillment of the requirements for the award of degree in Bachelor of Technology in the field of Electronics and Communication Engineering of Nirma University is the record of work carried out by them under our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for examination. The results embodied in this major project work to the best of our knowledge have not been submitted to any other University or Institution for award of any degree or diploma.

Date:

Institute Guide

Prof. Ruchi Gajjar

Department of Electronics and Communication Engineering.

Institute of Technology

Nirma University

Ahmedabad

Dr. Dhaval Pujara

Head of Department

Department of Electronics & Communication Engineering.

Institute of Technology

Nirma University

Ahmedabad

Dr. R.N. Patel

Director

Institute of Technology

Nirma University

Ahmedabad

Undertaking for Originality of the Work

We, **Avantika Sharma, Roll No. 16BEC015** and **Parth Modi, Roll No. 16BEC092**, give undertaking that the Major Project entitled “**Traffic Prediction using Deep Learning**” submitted by us, towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in **Electronics and Communication** of Nirma University, Ahmedabad 382 481, is the original work carried out by us and we give assurance that no attempt of plagiarism has been made. We understand that in the event of any similarity found subsequently with any other published work or any project report elsewhere; it will result in severe disciplinary action.

Signature of Student (16BEC015)

Signature of Student (16BEC092)

Date:

Place:

Endorsed by:

(Signature of internal guide)

Acknowledgement

We would like to take this opportunity to express our sincere gratitude and regards to **Prof. Ruchi Gajjar** for her timely guidance and continuous encouragement in carrying out this project work. She has imparted continual support and appreciation through various stages of the project. It would be impossible to complete the project without her advice and counselling. We are very fortunate to have her as our internal guide.

We would like to extend a deep sense of gratitude to **Dr. Dhaval Pujara**, HOD EC, for providing us with an opportunity to embark on this project. His guidance, encouragement and suggestions have contributed immensely to this project.

We are very much thankful to our peers, professors and lab officials for their continual support and help throughout the project.

Abstract

Deep Learning is one of the most trending cutting-edge technologies right now. It is a subset of Machine Learning and it involves algorithms powered by artificial neural networks ^[1]. Deep Learning(DL) and Machine Learning(ML) can be differentiated by the fact that ML algorithms require structured data in most cases while DL functions on a network of algorithms called Neural Networks ^[2].

The main purpose of this report is to showcase how a Deep Learning model can be used to predict the traffic, which is the topic of the project. Besides the millions that Traffic congestion costs the economy; it is one of the most persisting problems that the country faces. The main reason behind it is the increased number of vehicles due to an enormous population as well as the rapid development of the economy. In the project, we acquired traffic dataset from PEMS, California Department of Transportation and used that to train our model and predict upcoming traffic.

In another approach, we used a YOLO model that is trained on COCO dataset provided by Microsoft. We used object detection and bounding box generation algorithms to identify the vehicles that appear in the video. The bounding box moves along with the vehicles moving in the video. We calculated the speed of vehicles by assigning them coordinates and eventually got the average time taken by the vehicle to move out of the frame. Based on the speed of vehicles, we categorized the traffic as high or low. On implementing this technique, we were successfully able to categorize the different roads into congested roads and non-congested roads very accurately for both images and videos taken of the considered roads.

Contents

Acknowledgement	i
Abstract.....	ii
List of Figures.....	v
List of Tables	vi
Nomenclature	vii
INTRODUCTION.....	1
1.1 Introduction to Deep Learning.....	1
Neural networks	2
Convolutional Neural Network.....	3
1.2 Description Problem	4
1.3 Rationale for taking up the Project	5
1.4 Significance of the Project	5
1.5 Organization of the Report.....	6
Literature Review	7
2.1 Early Work.....	9
Traffic Prediction.....	14
3.1 Introduction.....	14
3.2 Different approaches to the problem.....	15
3.2.1 First Approach: Using regression algorithm.....	16
3.2.2 Second Approach: Using Object detection	17
3.3 Model Details.....	20
3.3.1 YOLO Algorithm.....	21
Experimental Results and Discussion	22
4.1 Dataset Details	22

4.1.1 PEMS Dataset	22
4.1.2 MSCOCO Dataset.....	24
4.2 Experimental Setup	25
4.3 Results.....	28
4.3.1 First Approach	28
4.3.2 Second Approach	29
4.4 Comparison with existing Approaches	33
4.5 Discussion	35
Conclusion and Future Scope	36
5.1 Future Scope	36
References	38
APPENDIX A	41
APPENDIX B	43

List of Figures

Figure 1: Deep Learning vs Machine Learning. ^[4]	1
Figure 2: A single node of a Neural Network. ^[4]	2
Figure 3: Diagram showing various layers of the convolutional neural networks ^[22]	4
Figure 4: Traffic Congestion caused in China. ^[6]	5
Figure 5: Result of Implementation of XOR Gate.	10
Figure 6: Dataset for Flower Classification.	10
Figure 7: Result of Flower Petal Classification.	11
Figure 8: Example of MNIST Dataset. ^[9]	12
Figure 9: Result of image classification using MNIST Dataset.	12
Figure 10: Traffic Congestion caused on a road. ^[10]	14
Figure 11: Project Block Diagram	16
Figure 12: Architecture of YOLO Algorithm. ^[12]	21
Figure 13: The final dataset used for the first approach.	24
Figure 14: MSCOCO dataset. ^[14]	25
Figure 15: Experimental setup for first approach.	26
Figure 16: Experimental setup for second approach.	27
Figure 17: Accuracy vs Epoch graph for the first approach.	28
Figure 18: Accuracy and loss metrics for first approach.	29
Figure 19: The algorithm giving an output showing the road is clear for the video overpass.mp4.	30
Figure 20: A still image from video overpass_output.avi.....	30
Figure 21: The output of the algorithm showing traffic congestion on the road.	31
Figure 22: A still image from the video traffic_output.avi.	31
Figure 23: The output of the algorithm for an image highway.jpg.....	32
Figure 24: The output of the algorithm for an image traffic.jpg.....	32

List of Tables

Table 1: Different Papers Referred.	7
Table 2: Truth Table of XOR Gate.	9
Table 3: Specifications of the model used for the first approach ^[11]	20
Table 4: Attributes provided in the PEMS dataset. ^[11]	23
Table 5: Comparison between all the three approaches.	34

Nomenclature

ML	Machine Learning
DL	Deep Learning
YOLO	You Only Look Once
COCO	Common Objects in Context
ReLU	Rectified Linear Unit
PEMS	Portable Emissions Measurement System
VOC	Visual Object Classes

Chapter 1

INTRODUCTION

1.1 Introduction to Deep Learning

Deep Learning is a subset of Machine Learning. It is a technique which can teach a computer to do what humans naturally can that is learn using examples. This technology is mainly used in self-driven cars, recognize images and control voices on consumer devices like TVs and phones. It is achieving results that weren't thought to be possible ^[3].

In the Deep Learning technique, a model is created on a computer which is trained in a way using datasets like text, image, sound or videos, so that it can directly perform the tasks desired from it. State-of-the-art accuracy can be achieved by these models, surprisingly even more than humans sometimes ^[3]. The models are trained using large datasets and neural networks. In Figure

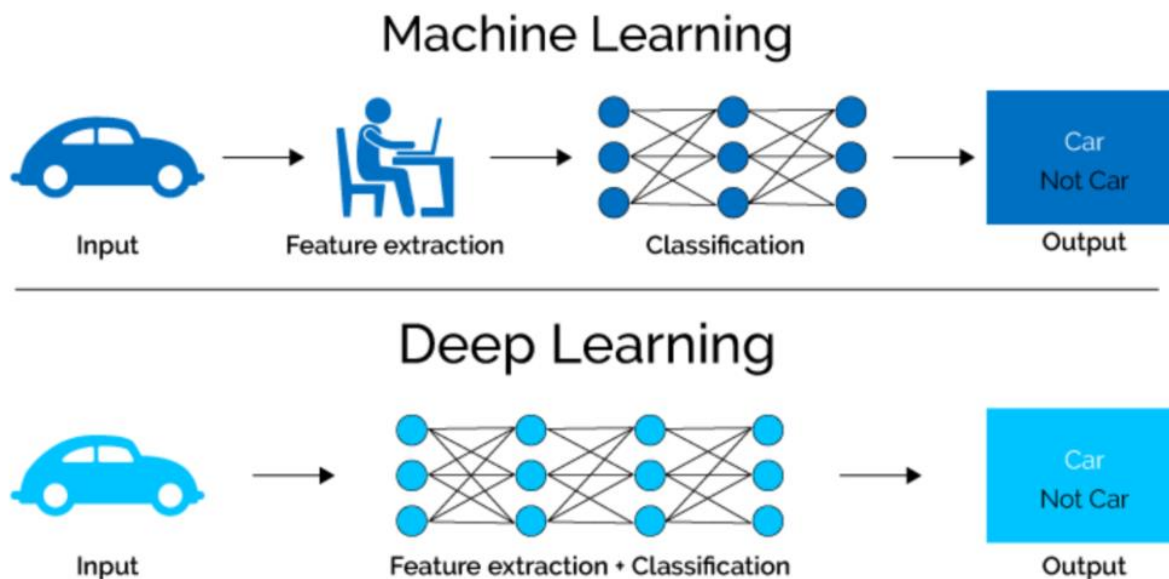


Figure 1: Deep Learning vs Machine Learning. ^[4]

1, it can be seen that in machine learning a feature extractor has to be created by the developer in accordance with the given data-type. After defining the different features of the data, the algorithm trains, itself from the data provided for training and does the classification of the

images. While in deep learning no manual feature extraction is required as the algorithm itself learns to extract a wide range of features from the data provided to it. Therefore, for deep learning algorithms different feature extractors are not required to be made for every problem.

Some of the spectacular applications powered by Deep Learning in real situations are:

1. Google Translate
2. Cancer Diagnosis
3. Self-driven cars
4. Identify faces
5. Read handwritten text
6. Speech recognition

Neural networks

Neural networks are used to train Deep Learning models. With the help of such models a computer learns to perform a given command by examining the training data. For instance, if we consider an object detection model, it will be trained on thousands of images of various objects and it will then visualize the pattern and recognize the class of object ^[5].

A large number of nodes are present in the multi-layered neural networks. These densely connected nodes function as the processing units. To see how a neural network works, Figure 2 provides a brief view of a neural network:

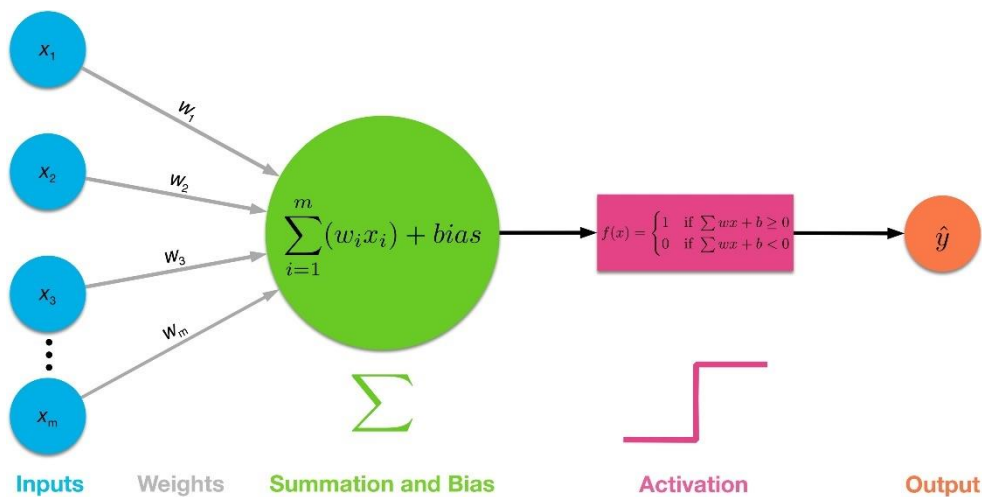


Figure 2: A single node of a Neural Network. ^[4]

The left side of the image has neurons containing the data input $z_1, z_2 \dots z_m$. These inputs are multiplied by the weights $a_1, a_2 \dots a_m$. So, the result becomes $z_1 * a_1, z_2 * a_2$ and so on. After the inputs are multiplied with the weights, we sum all of them up and add bias to it. Then we apply the activation function. Many types of activation functions are in existence such as sigmoid, tanh etc. Subsequently, the output is received ^[4].

Convolutional Neural Network

A convolutional neural network is a type of neural network which is very powerful and can use various filters to extract features of the data that is provided to it. Working with image data has been successfully accomplished by using neural networks such as CNN.

For a CNN, we need three basic components to define a network:

1. Convolutional layers
2. Pooling layers
3. Output layers

As shown in the Figure 3, Initially the input layer is provided with the input. On passing the input we get an output that is convoluted w.r.t to the input. The neural network extracts relevant features from the input by the filters used in the convolutional layer and passes it further. Each and every filter used helps in calculating different values to assist proper classification. It is important to retain the original size of the input and for that purpose zero padding is done. Sometimes it is advantageous to reduce the number of features I which case valid padding is used. For reduction in the parameter count, we further add pooling layers. Before making any predictions, addition of several convolutional and pooling layers takes place. If we want to extract more specific features, then we need to go into a deep network while generic features can be extracted with the help of a shallow network. A densely connected layer which has all of its nodes interconnected with all the nodes in the previous layer is used as the layer that provides the output. The flattened output from all the other layers is passed to the output layer. Here, it is transformed into the desired number of classes for the model. A loss function is then defined and various attributes like mean square error and error gradient are computed. After calculating the

error, it is then passed backwards to the other layers to update their weights and biasing values. This backward pass marks the completion of one training cycle^[21].

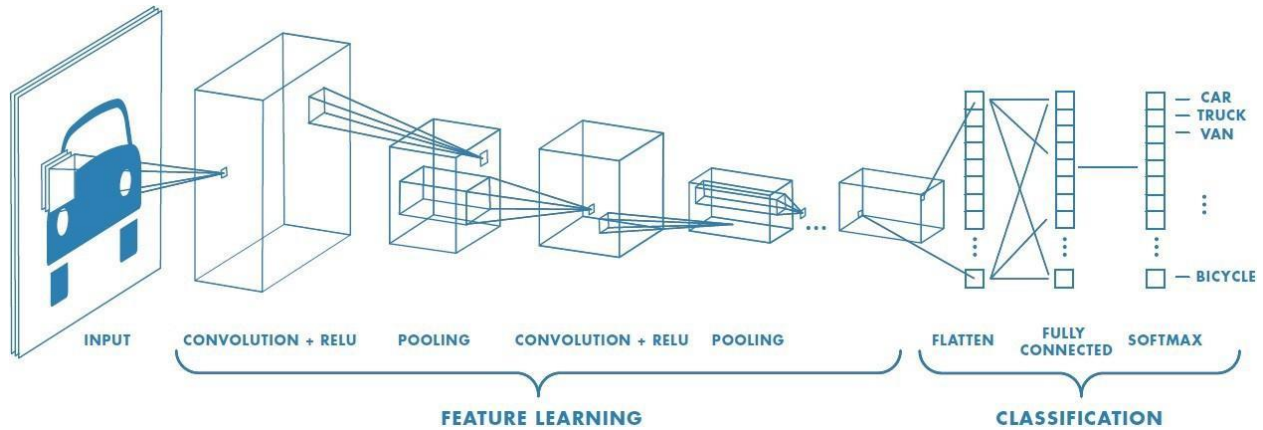


Figure 3: Diagram showing various layers of the convolutional neural networks ^[22].

Some of the challenges faced in implementing a Convolutional Neural Network are stated as follows:

1. They take up a lot of processing power
2. They need a huge amount of data to train the network
3. It is difficult to interpret since deep learning is a rapidly developing and changing field ^[23].

1.2 Description Problem

The objective of this project is to provide an analysis of road traffic data and from this analysis forecast the future trends of traffic using Deep Learning. Congestion of roads due to traffic is one of the persistent difficulties that the country is facing. It is majorly due to the enormous population and a rapid development of the economy. Global economy loses trillions of dollars in addition with the air pollution and the damage to public health and environment are caused due to the road congestion ^[11]. The problem of traffic congestion can be realized in Figure 4.



Figure 4: Traffic Congestion caused in China. ^[6]

1.3 Rationale for taking up the Project

The reason behind taking up this project was the challenges that Traffic congestion puts in front of the economy. Various such problems include:

- I. Global economy loses trillions of dollars due to road congestion.
- II. Air pollution is caused by road congestion which in turn plays a role in damaging the environment as well as the public health.
- III. Road congestion leads to more accidents and hence an increase in the medical costs.

The existing architecture of training models used to predict the traffic characteristics fall short in various aspects. Some of them are stated below.

- I. Only a small amount of data of the road traffic is used as the dataset.
- II. Datasets with limited scope are collected by researchers using their cameras.
- III. The configurations of networks based on deep learning are not discussed in detail by any models that exist now.

1.4 Significance of the Project

The importance of this project can be measured by the various purposes that it is capable of fulfilling. Traffic congestion is a persistent problem and it costs the nation's economy as well as a person his/her time. If one is able to know the traffic characteristic, the resident may take an

alternate route, avoiding the congested route and can save time while not adding to the congestion simultaneously.

The project may also serve to reduce travel delays by facilitating better utilization of available capacity and also help in decreasing traffic congestion.

Lastly, the prediction of the various characteristics of the road traffic can help in developing new traffic control strategies, varying existing road networks and modeling new road networks as well ^[11].

1.5 Organization of the Report

The literature that was used for reference in this project is reviewed in chapter 2. In chapter 3 information about the problem of traffic prediction is provided with mention to the different approaches used in this project and details about the models used. In chapter 4 the results found by implementation of the project are provided along with the details about the datasets used, the experimental setup, comparison of all the approaches used for traffic prediction and their discussion. Finally, the report is concluded in chapter 5 along with the future scope of the project. All the references used are mentioned in chapter 6 of the report.

Chapter 2

Literature Review

Congestion of roads due to traffic is one of the major difficulty in hand that costs any country billions and is a serious damage to the environment. This is the reason why various papers are being published and models being discovered to predict the traffic. We went through some research papers to get to know what kind of models for traffic prediction have already been discovered. Some of the papers that we studied for our project have been described below with respect to the dataset used and the model used in each paper.

Table 1: Different Papers Referred.

Paper Title	Year	Method Used	Mean Absolute Error	Dataset
Arterial Path-Level Travel-Time Estimation Using Machine-Learning Techniques	2017	k-nearest neighbour	8.90	Chennai
Smart Traffic Analysis using Machine Learning	2019	Random Forest	6.44	Bengaluru
Smarter Traffic Prediction Using Big Data, In-Memory Computing, Deep Learning and GPUs (Used in this project)	2019	CNN	16.25	California Transport

The first study that we referred to, introduced a methodology to forecast the travel time using the data provided by probe vehicles using global positioning system (GPS). The model used a combination of historic trends in addition to real time data to provide the expected travel time. As a result of this, we will get an accurate idea of the required time needed to travel on all the imaginable routes for any pair of destination and origin. This study provides some information about the need to split all the links into intersections and midlinks. The algorithm utilized in order to provide predictions at the midlinks was k-nearest neighbor. Moreover, at the intersection prediction was done by developing a random forest algorithm. Data acquired from the GPS of the buses used for public transport in Chennai, India was used to verify this model. The Mean Absolute Error as calculated in this approach turned out to be 8.90 ^[19].

The next study referred by us also used the random forest algorithm for traffic analysis. The paper worked in finding a method to help users find the easiest and the fastest path to their destination. The dataset used in this paper is generated by comparing to other datasets. The data contains details like node, time and distance for the traffic data. Random forest algorithm will be used to predict the delay in the nodes. This delay will help us choose a route with lesser delay. The TAM machine learning algorithm has been used in this paper for classifying the traffic whether it is low or high. The future work in the research includes the expansion of the places of traffic analysis and addition of reinforcement learning to the model. The dataset which is made and used in this paper is not provided online. Moreover, it is nowhere discussed in the paper how this dataset was created. The Mean Absolute Error calculated using the approach stated in this paper was found out to be 6.44 ^[20].

The final study we referred to us was very helpful for our project and was used thoroughly at various stages of the project. This study portrays a model based on deep learning which was used to forecast traffic. The current problems with the existing literature for models that are used for traffic prediction are also highlighted by this paper. These problems are stated below:

- I. Only a small amount of data of the road traffic is used as the dataset.
- II. Datasets with limited scope are collected by researchers using their cameras.
- III. The configurations of networks based on deep learning are not discussed in detail by any models that exist now.

The California Department of Transportation (Caltrans) has stored 11 years' worth of data from 2006-2017 which was used for the training of this model. The data collected for one whole day is divided into 288 five-minute interval traffic data. Various different information of the highways of California is provided by this dataset like flow of vehicles, speed, occupancy, stationID and timestamp. The deep neural network was developed in this paper for traffic prediction and this dataset was used to train that model. In this paper the dataset used was acquired from 26 vehicle detection systems installed on the I5-N highway of California. Evaluation metrics like mean absolute error (MAE), mean square error (MSE), and mean relative error (MRE) are computed. Out of which, Mean Absolute Error was found out to be 16.25 ^[11].

Some details about the referred papers is provided in Table 1.

2.1 Early Work

We started the project by delving into the world of Deep Learning and Neural Networks. Since we were new to the topic, the first task we did was to implement a simple XOR Gate with neural networks in python. Let us try to approach the problem by looking at the truth table designed for an XOR Gate that is shown in Table 2.

Table 2: Truth Table for an XOR Gate.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

A model with one layer for input, a hidden layer and one layer for output was designed by us. We initialized all weights to random numbers between 0 and 1. Sigmoid was utilized as the function for activation. Hence, it was a pretty simple algorithm. The process is to multiply the

input with the respective weights and then addition of bias takes place and finally the sigmoid activation function is added [7]. Figure 5 shows the prediction of the Machine Learning algorithm created to predict the output of an XOR Gate.

```

18 print("Initial hidden weights: ",end='')
19 print(*hidden_weights)
20 print("Final hidden bias: ",end='')
21 print(*hidden_bias)
22 print("Final output weights: ",end='')
23 print(*output_weights)
24 print("Final output bias: ",end='')
25 print(*output_bias)
26
27 for _ in range(20000):
28     hidden_layer_activation = np.dot(inputs,hidden_weights)
29     hidden_layer_activation += hidden_bias
30     hidden_layer_output = sigmoid(hidden_layer_activation)
31
32     output_layer_activation = np.dot(hidden_layer_output,output_weights)
33     output_layer_activation += output_bias
34     predicted_output = sigmoid(output_layer_activation)
35
36     error = expected_outputs - predicted_output
37     d_predicted_output = error * sigmoid_derivative(predicted_output)
38
39     error_hidden_layer = d_predicted_output.dot(output_weights.T)
40     d_hidden_layer = error_hidden_layer * sigmoid_derivative(hidden_layer_output)
41
42     output_weights += hidden_layer_output.T.dot(d_predicted_output) * 0.03
43     output_bias += np.sum(d_predicted_output,axis=0,keepdims=True) * 0.03
44     hidden_weights += inputs.T.dot(d_hidden_layer) * 0.03
45     hidden_bias += np.sum(d_hidden_layer,axis=0,keepdims=True) * 0.03
46
47 print("Final hidden weights: ",end='')
48 print(*hidden_weights)
49 print("Final hidden bias: ",end='')
50 print(*hidden_bias)
51 print("Final output weights: ",end='')
52 print(*output_weights)
53 print("Final output bias: ",end='')
54 print(*output_bias)
55
56 print("Input/output from neural network after 20000 epochs: ",end='')
57 print(*predicted_output)
58
59
60
61

```

Usage

Here you can get help of any object by pressing **Ctrl+H** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

Variable explorer | Help | Plots | Files

Console I/O

```

In [2]: runfile('C:/Users/Asus/.spyder-py3/xor.py', wdir='C:/Users/Asus/.spyder-py3')
Initial hidden weights: [0.9725655 0.97142935] [0.22454425 0.63642294]
Final hidden bias: [0.52831879 0.78914787]
Final output weights: [0.33954796] [0.46159399]
Final output bias: [0.8356893]
Final hidden weights: [0.9568826 0.37154876] [-2.75388683 5.06562059]
Final hidden bias: [-1.81446214 -0.99013093]
Final output weights: [-5.87995593] [7.05396752]
Final output bias: [-1.17362134]
Output from neural network after 20000 epochs: [0.01313865] [0.98385987] [0.49972452]
[0.50857945]
In [3]:

```

Python console | History

@ conda-base (Python 3.7.0) Line 59, Col 1 ASCII UTF-8 RAW Mem 69%

Figure 5: Result of Implementation of XOR Gate.

Next task on our project was Flower Classification. It involved classifying the flower petals based on the length and width of the petal. The dataset involved the details of length and width of petals and based on that the color of the petal was determined. An example of the dataset is shown below. The dataset provided for flower petal classification is shown in Figure 6









color									???
length	3	2	4	3	3.5	2	5.5	1	4.5
width	1.5	1	1.5	1	.5	.5	1	1	1

Figure 6: Dataset for Flower Classification.

Based on this, we had two axes, one for the width of the petal while the other for the length of the petal. Upon considering both the factors, the color (Red or Blue) of the petal can be determined. An output of the algorithm is displayed in Figure 7.

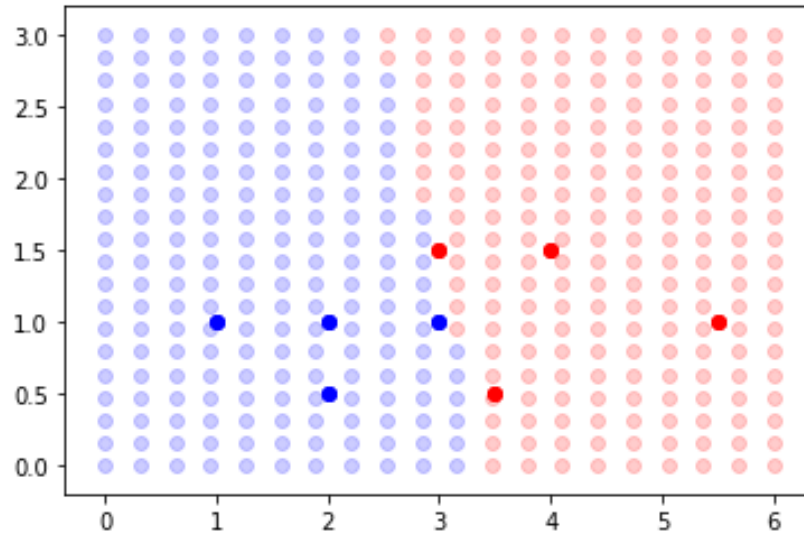


Figure 7: Result of Flower Petal Classification.

One of the capabilities that deep learning depicts is recognition of handwritten data. We developed a deep learning model for handwritten digit recognition. The dataset used to train the model was MNIST dataset. This dataset can be loaded in python using the Keras library. This dataset was developed by taking various images of digits from various scanned documents and normalizes them in size. The resolution of each image is 28 x 28-pixel square (total 784 pixels). The dataset contains 60000 images that are divided into training data and validation data. 10000 more images are used to test the model ^[8]. provides an idea about the MNIST dataset.

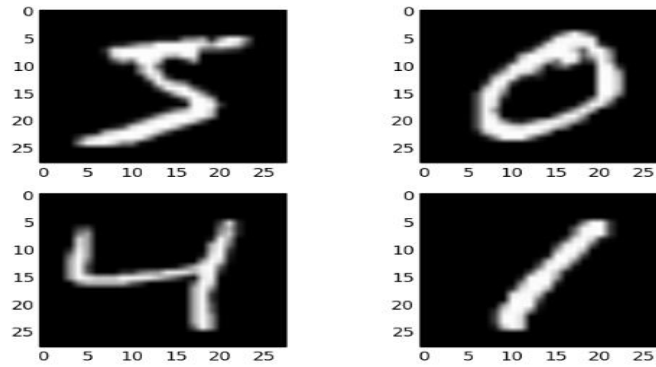


Figure 8: Example of MNIST Dataset. ^[9]

The data was divided into the two groups namely: train and test and the images and labels were

```

1 import tensorflow as tf
2 import numpy as np
3 mnist = tf.keras.datasets.mnist
4 (x_train, y_train), (x_test, y_test) = mnist.load_data()
5 import matplotlib.pyplot as plt
6
7 plt.imshow(x_train[0], cmap=plt.get_cmap('gray'))
8 from keras.models import Sequential
9 from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
10 x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
11 x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
12 input_shape = (28, 28, 1)
13 # Making sure that the values are float so that we can get decimal points after division
14 x_train = x_train.astype('float32')
15 x_test = x_test.astype('float32')
16 # Normalizing the RGB codes by dividing it to the max RGB value.
17 x_train /= 255
18 x_test /= 255
19 model = Sequential()
20 model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
21 model.add(MaxPooling2D(pool_size=(2,2)))
22 model.add(Flatten())
23 model.add(Dense(200, activation = tf.nn.relu))
24 model.add(Dropout(0.3))
25 model.add(Dense(10, activation=tf.nn.softmax))
26 model.compile(optimizer='adam',
27               loss='sparse_categorical_crossentropy',
28               metrics=['accuracy'])
29 model.fit(x=x_train, y=y_train, epochs=10)
30 z=model.evaluate(x_test, y_test)
31 print('test loss and accuracy:', z)

```

```

Please use 'rate' instead of 'keep_prob'. Rate should be set to 'rate = 1 - keep_prob'.
WARNING:tensorflow:From C:\Users\Aous\Anaconda3\lib\site-packages\tensorflow\python\ops\num_grad.py:1250: add_dispatch_support._locals.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Epoch 1/10
60000/60000 [=====] - 32s 533us/step - loss: 0.2808 - acc: 0.9382
Epoch 2/10
60000/60000 [=====] - 20s 336us/step - loss: 0.0858 - acc: 0.9737
Epoch 3/10
60000/60000 [=====] - 20s 337us/step - loss: 0.0588 - acc: 0.9815
Epoch 4/10
60000/60000 [=====] - 20s 334us/step - loss: 0.0452 - acc: 0.9856
Epoch 5/10
60000/60000 [=====] - 20s 335us/step - loss: 0.0388 - acc: 0.9871
Epoch 6/10
60000/60000 [=====] - 20s 337us/step - loss: 0.0308 - acc: 0.9898
Epoch 7/10
60000/60000 [=====] - 20s 335us/step - loss: 0.0278 - acc: 0.9906
Epoch 8/10
60000/60000 [=====] - 20s 336us/step - loss: 0.0242 - acc: 0.9921
Epoch 9/10
60000/60000 [=====] - 20s 335us/step - loss: 0.0234 - acc: 0.9929
Epoch 10/10
60000/60000 [=====] - 20s 335us/step - loss: 0.0221 - acc: 0.9929
10000/10000 [=====] - 1s 88us/step
test loss and accuracy: [0.053775641403728515, 0.9848]
In [4]:

```

Figure 9: Result of image classification using MNIST Dataset.

also separated. X_Train and X_Test are comprised of the RGB codes (0-255) and Y_Train and Y_Test are comprised of the labels from 0-9. We used matplotlib to help visualize the numbers. We used the shape attribute because we also need to know the shape of the dataset, besides the length and width, before channelizing it to the convolutional neural network. We imported the sequential model from Keras. The first dense layer has 10 neurons since we have 10 classes of digits ^[9]. The result was obtained with an accuracy of 98.48% with 10 epochs only and it is shown in Figure 9.

Chapter 3

Traffic Prediction

3.1 Introduction

Traffic is one of the largest chokepoints when it comes to efficiency in production and transportation. Any goods being transported via airways, seaways and of course roadways have to be first transported to and from the necessary ports by road. Almost, 80% of all passenger transportation in India takes place via road transportation and it also accounts for 65% of all goods transportation ^[15]. Therefore, a large amount of money, time and fuel is lost every day due to congestion of cars on roads. A congestion free road network can be a very valuable asset for transportation of people and goods in a country. A big congestion of vehicles on the roads also



Figure 10: Traffic Congestion caused on a road. ^[10]

plays a part in the increase of pollution in those areas. A less congested road is a safer road to travel as considered by the government. Traffic congestions take place due to many reasons some of which are accidents, less efficient timing of the signals, roads being repaired, bad weather. But, the main cause of increase in traffic is due to a very dense number of vehicles on a narrow road. Traffic congestion caused on a road is shown in Figure 10.

Due to this traffic prediction is a very important tool for the development of a country. It is also true that avoiding traffic is not always possible but if we are able to show the trends of traffic on the roads of the country, then using these predictions a lot of money can be saved in terms of time and fuel and can also make the roads safer. These forecasts can also help the government to identify the bottlenecks on the road networks and upgrade those areas and in some cases to even make adjacent roads to reduce the dependency on that part of the road network in order to increase the capacity of those areas to have a decrease in congestion. The common people commuting by roads or using these roads for travelling can use these predictions to better plan their route to avoid as much traffic as possible and save more time. The time of commuting from point A to point B can be fairly reduced if accurate forecasts of traffic congestion can be produced and made available to people in real time. In this project two different approaches have been used to provide an accurate prediction of traffic. Both these approaches are discussed in detail in the next sections.

3.2 Different approaches to the problem

A couple of approaches were used in this project so that this problem of real-time traffic prediction can be solved. Through Figure 11, we can see the block diagram representing the entire project. It can be seen in the block diagram that both the approach uses different methods to for data collection and model training and testing. However, at the end the output of either model can be used for classification and to obtain the final output of whether there is traffic congestion present on the road under consideration or not. Both the approaches are explained in detail in the coming sections.

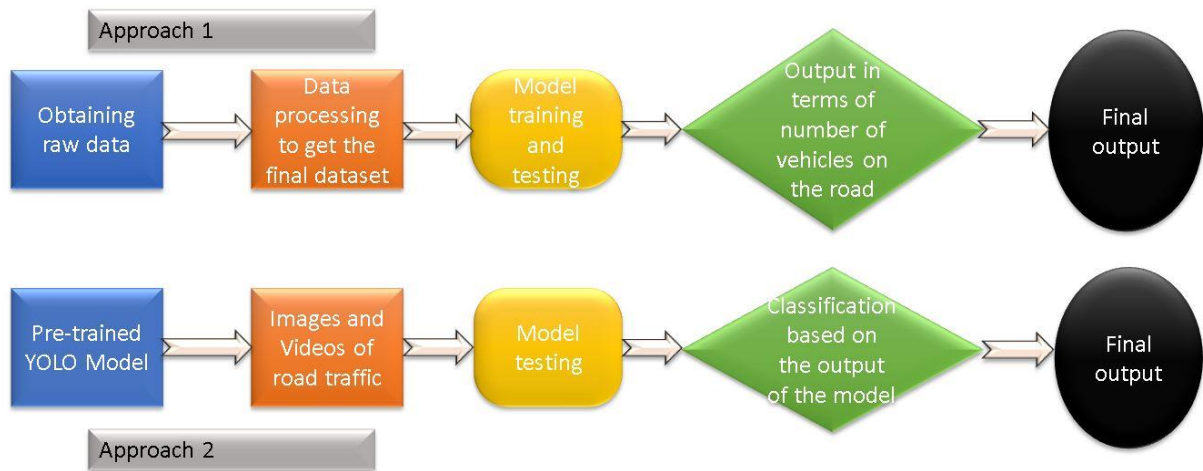


Figure 11: Project Block Diagram

3.2.1 First Approach: Using regression algorithm

In this approach, data of the numbers of cars on a given highway at a given time of the day was collected. This dataset comprised of a month of data collected from the PEMS website which has been collecting such data for more than a decade in the state of California. The state of California has installed stations at pre-decided points on all the highways in the state that collect this data. The dataset contains data like number of cars crossing a station in every five minutes, the average speed of the traffic crossing the station, the density of the traffic on the highway where the station is installed. The dataset divides the whole day into 288 parts of five-minute intervals and provides the numbers the number of cars, average traffic speed and density for each part of the day. This can help in determining the trends on the roads at peak traffic hours and non-peak traffic hours. As the data is over a decade long it can also predict the changes in the trend on weekends, holidays and different seasons. Putting this whole dataset in a regression-based model can estimate the number of all the vehicles that may travel on the considered highway at any hours of any day in any month of any year. This seems to be a simple and elegant solution to the problem of traffic prediction. However, there are some complications and disadvantages while utilizing this approach at traffic prediction. First of all, as stated above the dataset is over a decade long and hence a very powerful computer system is needed to get the trends of traffic for such a huge dataset. Due to the non-availability of such a powerful computer system the size of the dataset was reduced by us to a month worth of data. The data chosen was of the month of

July 2015. Due to the reduction of the size of the dataset there was a steep decrease in the accuracy of the model and also some of the characteristics of the trends like trends for holidays and different seasons were lost. Secondly, the dataset available was for the state of California. Not all states in the United States store such data for their road networks let alone the other countries of the world. Such a dataset for the road networks of a country like India are almost non-existent and to create such a dataset in India needs a lot of infrastructure in place, which will be both costly and time consuming. Lastly, this model does not consider the effects of accidents and repairs taking place on the roads and as the frequency of such phenomenon on the road is very less the predictions will not be much affected by these numbers even if they took place in the past and were recorded by the dataset. Due to these complications' and limitations this model not able to very accurately predict the trends in traffic all year round as is required in the problem. For this reason, a different approach was implemented in order to resolve the problem of traffic prediction.

3.2.2 Second Approach: Using Object detection

In this approach YOLO object detection algorithm was applied to predict traffic taking place on the roads accurately. This approach resolved many of the limitations that were experienced in the previous approach. In this approach the photos and videos of the roads using cameras installed on the roads by the government were used in order to detect the number of vehicles on the road at a given point of time. This infrastructure is available in almost all the cities in India where the problem of traffic congestion is faced. Also, this approach can accurately show the congestion on the roads in real-time. In this approach, the videos and photos collected are fed in the YOLO object detection algorithm using the OpenCV library available for python. On feeding the dataset in the YOLO algorithm, it detects objects that are in those photos and videos. The YOLO object detection algorithm that is used in this project has been pre-trained on the COCO dataset provided by Microsoft. This dataset contains more than 1,00,000 images of objects divided in 80 classes. These classes include objects like person, airplane, car, truck, bus, motorcycle, bicycle and many others. The dataset is more than 18 GB in size, out of which 12 GB is data for training and the rest is testing data. On feeding the videos and images of the roads in the algorithm, it detects all the objects that are included in the COCO dataset and also appearing in that image or video. Due to this, the algorithm was trained using custom dataset which comprised of only 5

classes that were car, bicycle, motorcycle, bus and truck. In this way any unwanted detections were excluded from consideration. After the detection of the required objects was done there were two methods to predict traffic.

1. By finding out the total number of vehicles detected by the algorithm:

By this method we can find out the total density of vehicles present on the highway when the detection is done. This can be useful to forecast the traffic as according to common knowledge the greater number of cars detected in unit time i.e. More vehicles lead to higher congestion levels on a highway. However, this method does not account for fast moving traffic without congestion. In this situation the density of vehicles on the road will come out to be large in number, but in real sense there is no traffic jam on the road despite of large number of vehicles. Therefore, deciding a number of vehicles as threshold for this method can be difficult and can vary for each and every road of different widths and number of lanes. Therefore, it can be argued that although this method may predict traffic accurately it may provide some incorrect predictions. Consequently, for more accuracy another method was employed for better predictions.

2. By determining the speed of the traffic:

The YOLO object detection algorithm can detect a required object as well as create a bounding box rectangular in shape around the detected object. This bounding box is employed so that we can determine the average speed of the vehicles. This algorithm utilizes the coordinates at which the object is present in the frame of the video or the photo. These coordinates help us in identifying the speed at which a detected object that is essentially a vehicle on the road, is moving. If we define a starting coordinate and an ending coordinate point and get the time taken for a detected object to reach from the starting coordinate to the ending coordinate, we can simply get the speed of that object by the formula of speed which can be written as in the equation given below.

$$\text{Speed of the detected object} = \frac{\text{Ending coordinate} - \text{Starting coordinate}}{\text{The time required by the object}} \quad \text{---(1)}$$

The outcome of this calculation will be the speed of the object not in meters/second but in coordinates/second. It is not the actual speed of the object but we don't need the actual

speed and can work with this value to make accurate predictions. The value of speed we get from this formula is the speed of individual vehicles. In this way, we can find out the speeds of all the detections made by the algorithm in terms of coordinates/second. On taking average of all the speeds calculated by the above formula we get the average speed of all the vehicles which can also be considered as the speed at which the traffic is moving. In this method also it is difficult to find out the threshold value for the speed of traffic. But we know in this case that the lower the speed is the greater is the congestion. Also, in most of the cases of congestion the vehicles crossing the initial coordinate point are not able to cross the final coordinate point and on tweaking the code a little more we can see that vehicles in such situation get a speed of 0 coordinates/second and as soon as the code detects a vehicle with a speed of zero, it predicts a congestion on the road. On the roads sometimes the vehicles stop for some reasons other than congestion. This may cause a problem in prediction as these vehicles will get a speed of zero and the code will predict a congestion which will be an incorrect prediction to tackle this problem the number of vehicles with a speed of zero needed to get a prediction of congestion can be increased with a simple 'if' statement.

In this approach we don't need to worry about the different times of the day or different seasons because this approach shows a real-time traffic situation on the roads and also the infrastructure required is mostly the cameras required on the roads where congestion is a problem. Indian government has already installed cameras in almost all the cities at various locations to tackle the problem of theft, crime, traffic violations and these cameras can also be efficiently used for traffic prediction. The problem faced in the previous approach regarding accidents and repair work on the roads is also insignificant here as real-time feed of the traffic at desired locations is created by this approach. The only limitation that can be faced in this approach is that there is no possible way of predicting traffic where cameras are not installed and also each and every road of the country cannot be monitored in this manner. However, this limitation can be solved by installing cameras efficiently which can provide a better idea of traffic to the common people and also the government. Moreover, if the government creates a database for

frequency of traffic congestion at various roads, they can find out which areas need an upgrade on the road networks.

3.3 Model Details

The model created and used for the first approach used Keras and Tensorflow libraries in order to execute the model at the back end. The model consisted to 6 layers including the input layer and the output layer. There were 4 hidden layers in the model. The number of input parameters was eighteen while there were twelve output parameters. The layer wise hidden units were 85, 425, 425, 85 respectively. The batch size used for this model was 500 and 5000. Number of epochs at which the accuracy of the model was checked were 100, 500 and 1000. The activation function used for the model was ReLU ^[11]. The model specifications are as depicted in Table 3.

The model used in the second approach is the YOLO algorithm that is discussed in the next section.

Table 3: Specifications of the model used for the first approach ^[11].

Number	Description	Values
1	Input parameters	18
2	Output parameters	12
3	Hidden layers	4
4	Layer wise hidden units	85, 425, 425, 85 respectively
5	Batch Size	500, 5000
6	Epochs	100, 500, 1000
7	Activation Function	ReLU

3.3.1 YOLO Algorithm

This algorithm for object detection was designed by Joseph Redmon et al ^[12]. A Convolutional

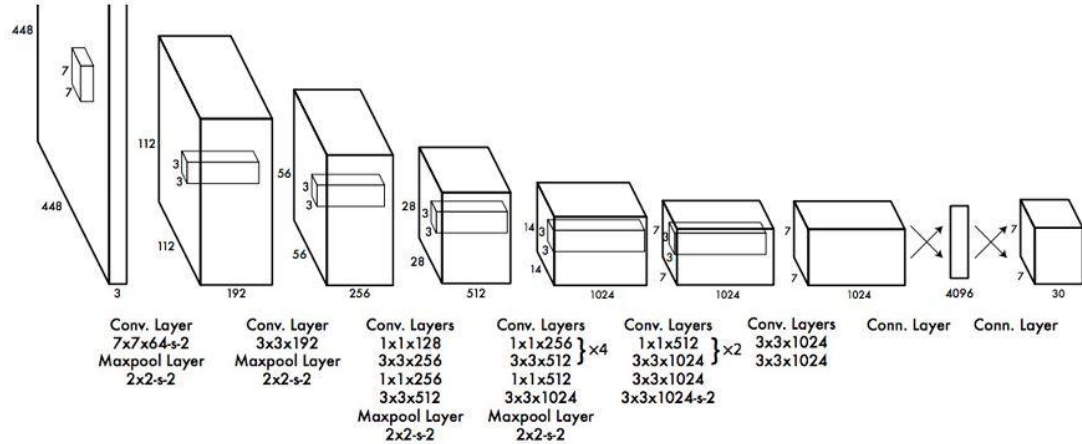


Figure 12: Architecture of YOLO Algorithm. ^[12]

Neural network trained on PASCAL VOC dataset is used in this model ^[13]. The convolutional layers in the initial part take on the task of extracting the different features from the image and the output probabilities and coordinates are forecasted by the layers that are fully connected, at the end.

The neural network is designed with the help of the model provided by GoogLeNet for image classification ^[12]. There are 24 convolutional layers at the beginning of the YOLO model and 2 layers that are fully connected at the end. GoogLeNet uses inception modules instead of which we use 3×3 convolutional layers after using 1×1 reduction layers. The network architecture as explained above is shown in Figure 12: Architecture of YOLO Algorithm. ^[12].

Chapter 4

Experimental Results and Discussion

The results found from the experiment conducted in this project are discussed in this section.

4.1 Dataset Details

As discussed above two approaches were followed to solve the problem at hand and two different models were used in both the approaches. While in the first approach the model was trained by a dataset extracted from the PEMS website for the state of California in the United States, in the latter approach a YOLO algorithm was used which was pre-trained on the COCO dataset provided by Microsoft.

Let us discuss both these datasets separately.

4.1.1 PEMS Dataset

The dataset extracted from the PEMS website ^[13] is basically the data stored by the different stations installed on the highways of California. This dataset consists of 38 attributes which includes the numbers of vehicle on the different lanes of the road at a given time. This dataset not only consists of the number of cars on the road, but also the average occupancy, average speed of the traffic. This dataset is created for a road of 8 lanes and the above stated attributes are provided for all the different lanes. Along with this the total number of vehicle flow, average occupancy and average speed at the stations are also observed and stored. These many attributes are not needed by us for the traffic prediction and therefore this dataset is first needed to be parsed with the help of a data parsing code. This data parsing code splits the timestamp attribute present in the raw data into hours, date and day of the week. Station ID is also appended by the code into the final dataset. Finally, the number of vehicles that are collected for a given day are split into 288 5-minute intervals with the help of the timestamp which helps us understand the trend of day, night, peak hours and non-peak hours. The actual dataset that is extracted after data parsing is specified in Table 4.

Table 4: Attributes provided in the PEMS dataset. ^[11]

Number	Attribute	Details
1	Station identification	This attribute provides information about the identity of the station that are considered in this project.
2	Day of Month	This attribute provides information about the date at which the corresponding data was collected from the corresponding station on the considered highway.
3	Month	The number assigned to the months of the year are displayed here. The month in which the data was collected is depicted in this column
4	Year	Information about the year is provided in this column.
5	Time of the day	The time of the day is indicated by this column, the precise time at which the data was registered by the station is indicated in this column. It varies from 0 to 23 to span all the hours of the day
6	Week Day	The information about which day fell on the corresponding date according to the calendar is provided in this column. This information will help us clearly understand the difference in the trends for week days and weekends. This will help in increasing the accuracy of the proposed model
7-18	Flow (flow00, flow05, flow10, ..., flow55)	The data collected from the website provides information about the number of vehicles, speed and occupancy in respect with the whole day. This provides us with daily trends in place of the required hourly trends. Therefore, in these columns the data with respect to the whole day is divided into data for every 5-minute interval and hence more microscopic trends of the road traffic is understood

The dataset provided by PEMS provides data about each and every highway in the state of California and the hundreds of stations placed on those highways. For experiment purposes we considered only one highway which was I5-N and the 26 stations present on that highway. The dataset contained of a large number of entries and therefore only one highway was considered.

The data was collected for one month for the 26 stations for each and every 5-minute interval of all the 31 days of the month. The final dataset and all the required parameters can be seen in

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	stationid	day_of_m-month	year	hour	day_of_w	flow_0	flow_5	flow_10	flow_15	flow_20	flow_25	flow_30	flow_35	flow_40	flow_45	flow_50	flow_55	sum					
2	715918	1	7	2015	0	3	76	66	70	70	68	63	57	46	63	60	43	60	742				
3	715918	1	7	2015	1	3	60	63	60	70	57	46	48	48	48	63	36	117	716				
4	715918	1	7	2015	2	3	33	36	147	48	122	313	429	864	165	106	131	155	2549				
5	715918	1	7	2015	3	3	123	91	126	99	112	123	86	195	191	96	236	91	1569				
6	715918	1	7	2015	4	3	50	70	333	191	76	66	81	84	96	170	231	112	1560				
7	715918	1	7	2015	5	3	94	106	106	150	183	197	195	210	246	242	244	203	2176				
8	715918	1	7	2015	6	3	226	221	236	260	215	215	165	180	208	203	188	210	2527				
9	715918	1	7	2015	7	3	180	197	226	195	224	188	231	213	200	215	221	221	2511				
10	715918	1	7	2015	8	3	193	208	183	231	203	210	228	218	244	180	208	246	2552				
11	715918	1	7	2015	9	3	197	206	158	158	191	180	208	210	188	178	173	180	2227				
12	715918	1	7	2015	10	3	188	200	170	144	147	155	165	228	170	158	203	173	2101				
13	715918	1	7	2015	11	3	159	183	158	162	188	158	165	178	165	147	141	208	2012				
14	715918	1	7	2015	12	3	144	197	155	162	178	140	173	178	210	158	158	254	2107				
15	715918	1	7	2015	13	3	168	197	203	344	864	471	313	295	165	180	170	173	3543				
16	715918	1	7	2015	14	3	165	165	147	155	162	178	152	140	158	168	176	186	1952				
17	715918	1	7	2015	15	3	158	170	178	178	147	170	186	147	170	165	159	155	1983				
18	715918	1	7	2015	16	3	170	162	176	186	168	170	137	173	162	144	170	195	2013				
19	715918	1	7	2015	17	3	152	176	158	170	140	159	152	168	191	165	168	176	1975				
20	715918	1	7	2015	18	3	170	140	173	152	165	131	152	158	141	141	158	152	1833				
21	715918	1	7	2015	19	3	131	159	150	159	140	144	131	120	150	180	165	170	1799				
22	715918	1	7	2015	20	3	170	147	141	168	168	150	150	165	170	147	99	122	1797				
23	715918	1	7	2015	21	3	112	120	99	126	96	129	122	117	140	114	109	137	1421				
24	715918	1	7	2015	22	3	99	129	109	126	94	104	114	106	102	99	76	96	1254				
25	715918	1	7	2015	23	3	102	96	104	91	91	99	86	66	60	60	63	78	996				
26	715918	2	7	2015	0	4	81	81	66	63	76	68	66	63	48	81	50	78	821				
27	715918	2	7	2015	1	4	63	46	48	55	48	43	48	55	46	40	66	63	621				
28	715918	2	7	2015	2	4	48	36	48	40	55	63	38	48	48	43	57	40	564				
29	715918	2	7	2015	3	4	60	57	60	43	50	68	52	57	55	68	63	63	696				
30	715918	2	7	2015	4	4	55	57	55	57	68	81	76	84	81	112	96	91	913				
31	715918	2	7	2015	5	4	109	131	144	155	162	200	191	191	188	168	140	150	1929				
				</																			

Figure 13: The final dataset used for the first approach.

Figure 13.

4.1.2 MSCOCO Dataset

MSCOCO dataset ^[16] was introduced by Microsoft. It was introduced in the year 2014 and is updated every year with a greater number of images and increased number of classes. In this project we have used the original COCO dataset of 2014.

MSCOCO 2014 consisted of 80 classes of objects. However, it defined 91 classes but of those 91 classes 80 were only used. The dataset also has a split of test, train and validation images to better improve the accuracy of the YOLO algorithm. COCO contains object details, annotations, instances, captions, segments and labels. The total download size of COCO dataset is over 37 GB. It includes 82,783 images for training, 40,775 images for testing along with 40,504 images for validation. The images in the train and the validation splits are labeled while the images in the test split are not labeled. There is a panoptic version of MSCOCO available for download. This version defines about 200 classes but uses only 133 classes. The dataset also includes the coordinate details of the objects present in the images. This helps the YOLO algorithm to spot

the objects and make a bounding box around it. Coordinates of the objects are shown as (centreX, centreY, height, width), where centreX and centreY coordinates represent the central x and y points of an image and height and width show the height and width of the image respectively. By using these four coordinate points, the bounding boxes around the object can be created accurately. In the PASCAL VOC dataset ^[13], the images are stored in a rectangular form this causes more than one object to be in the image but in MSCOCO the images are stored in such a way that there is at most one instance in every detection. Some examples of the MSCOCO



Figure 14: MSCOCO dataset.^[14]

dataset are provided by Figure 14.

4.2 Experimental Setup

The project consists of two different approaches to the same problem. However, the setup of the experiment is almost identical for both the approaches.

As shown in Figure 15, In the first approach the data is collected from the stations created on the highways and is in form of texts and numbers. The stations use Sensys Network Wireless Technology system. This system is created by Sensys Networks. This system uses Grind Resistance Sensors to detect speed, flow and occupancy of vehicles on the road ^[24]. This data is then put into a parser code and the required fields of the data are extracted. The raw data contains

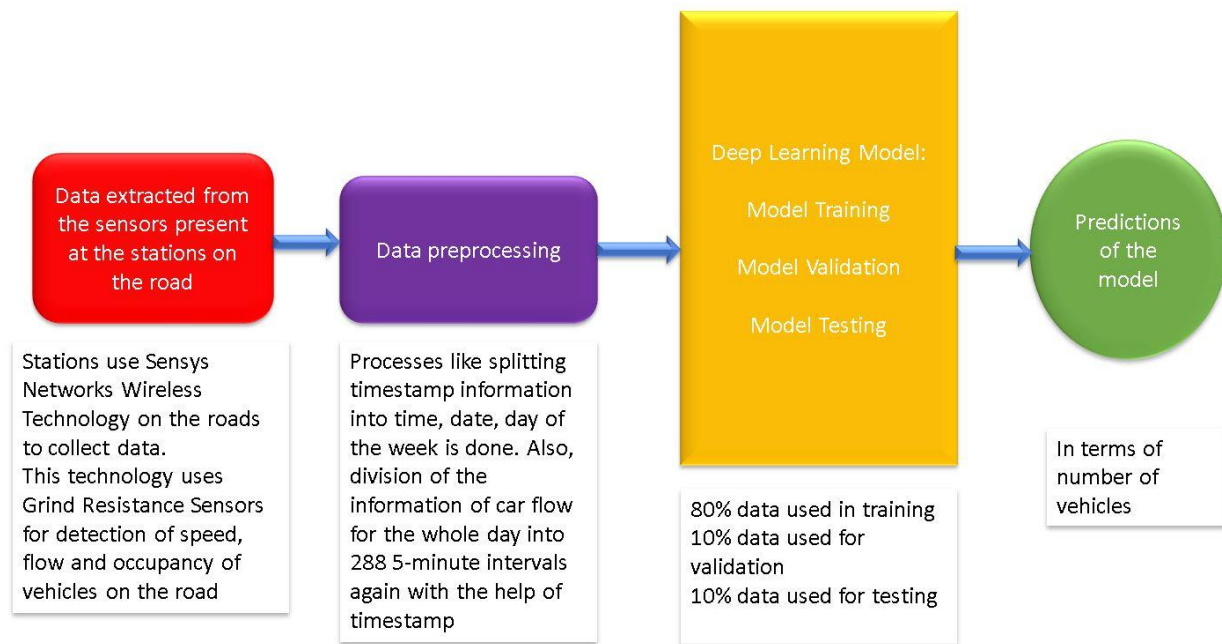


Figure 15: Experimental setup for first approach.

fields like timestamp and separate data of number of cars on roads up to 8 lanes. Data parsing code splits the timestamp and extracts details like hours, date, day, month, year. The code also separates the data of number of vehicles for a day into 288 5-minute intervals, so that better track of peak hours and non-peak hours can be kept. This extracted data works as the final dataset that will be utilized by the deep learning model. This model is trained with the help of this dataset. The dataset is divided into training, validation for better accuracy and also in testing data which provides the predictions of the model that can be compared with the test data itself. 80% of the dataset is used to train the model, while 10% of the dataset is used as validation data and the remaining 10% data is reserved for testing. The model after getting the data crunches out number of cars, which are used as outputs for different times of the day. This output can be used to realize the trends of traffic for any specific time of any day. These trends can be displayed at the

preceding stations to provide knowledge to the people of what kind of traffic can be met at the next station.

As stated above the experimental setup for both the approaches are almost identical. Although, in the second approach the dataset is the MSCOCO dataset and the YOLO algorithm used is pretrained on this dataset. However, for custom object detection the YOLO algorithm can again be trained using the COCO dataset only, by making some necessary changes in the config file of the YOLO algorithm. As input to the algorithm for object detection some images and videos of traffic on roads are used. These input videos and mages can be extracted from the cameras already installed on many of the roads. The objects in these videos and images are detected thereafter bounding boxes are created surrounding the detected objects using the coordinates generated by the algorithm. The coordinates generated by the algorithm are also used to calculate the speed of the traffic in the video in terms of coordinates/second. This equation formulated by us, helps us to classify whether the videos and images are congested by traffic or not. This output can be directly used to provide information to the people in form of display boards put at convenient locations or by developing an app that can provide this information to the people,

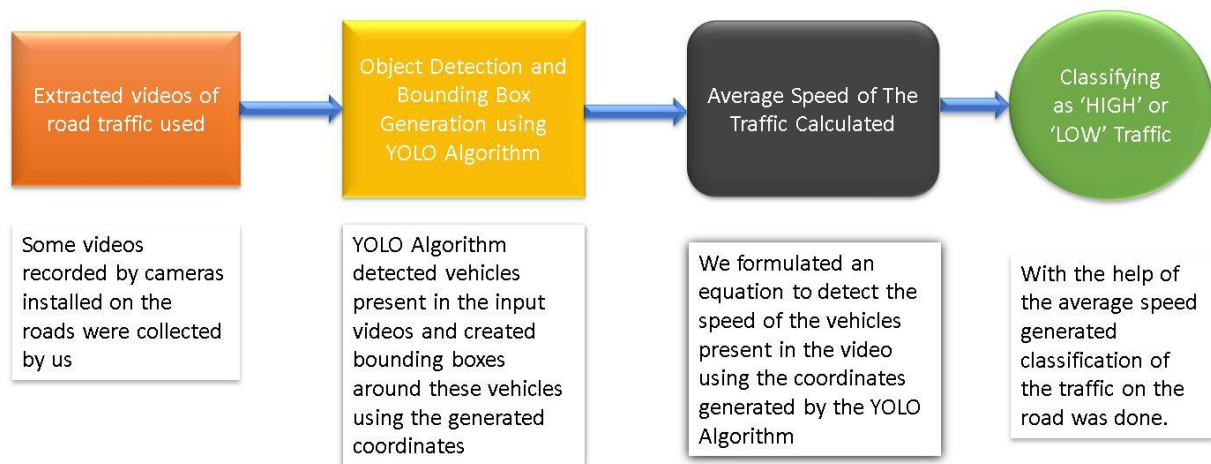


Figure 16: Experimental setup for second approach.

also can store this information for government verification in order to upgrade the roads. The experimental setup for the second approach as explained above can be understood by Figure 16.

In the following section the results that were observed from both the approaches are discussed.

4.3 Results

The results obtained after implementing both the above-mentioned approaches are discussed in this section.

4.3.1 First Approach

Figure 17 shows the graph of accuracy vs epochs from which we can observe how the accuracy of the model increases with respect to the number of epochs. Figure 18 provides the value of the accuracy and the loss metrics of the implemented model. As discussed earlier this approach provides the results depending upon the data provided to the model. The output of this approach is in the form of the number of cars that may pass through a given station as predicted by the model.

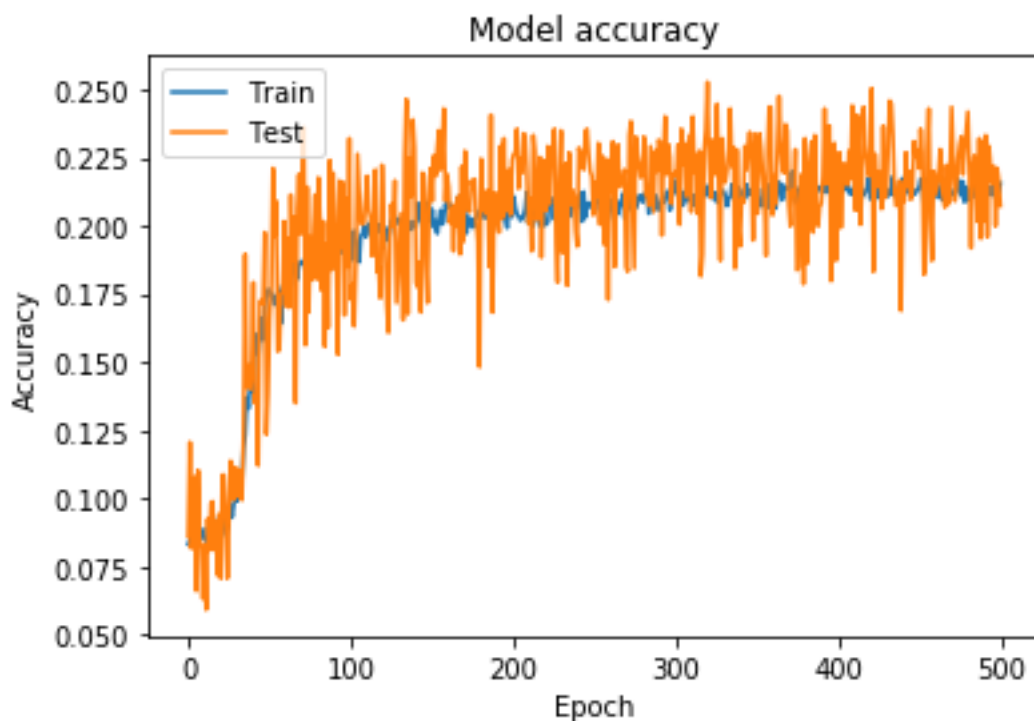


Figure 17: Accuracy vs Epoch graph for the first approach.

As seen in the above figure the accuracy metrics for the model used in the first approach is around 21% for training data after 500 epochs and around 20% for the validation data. The

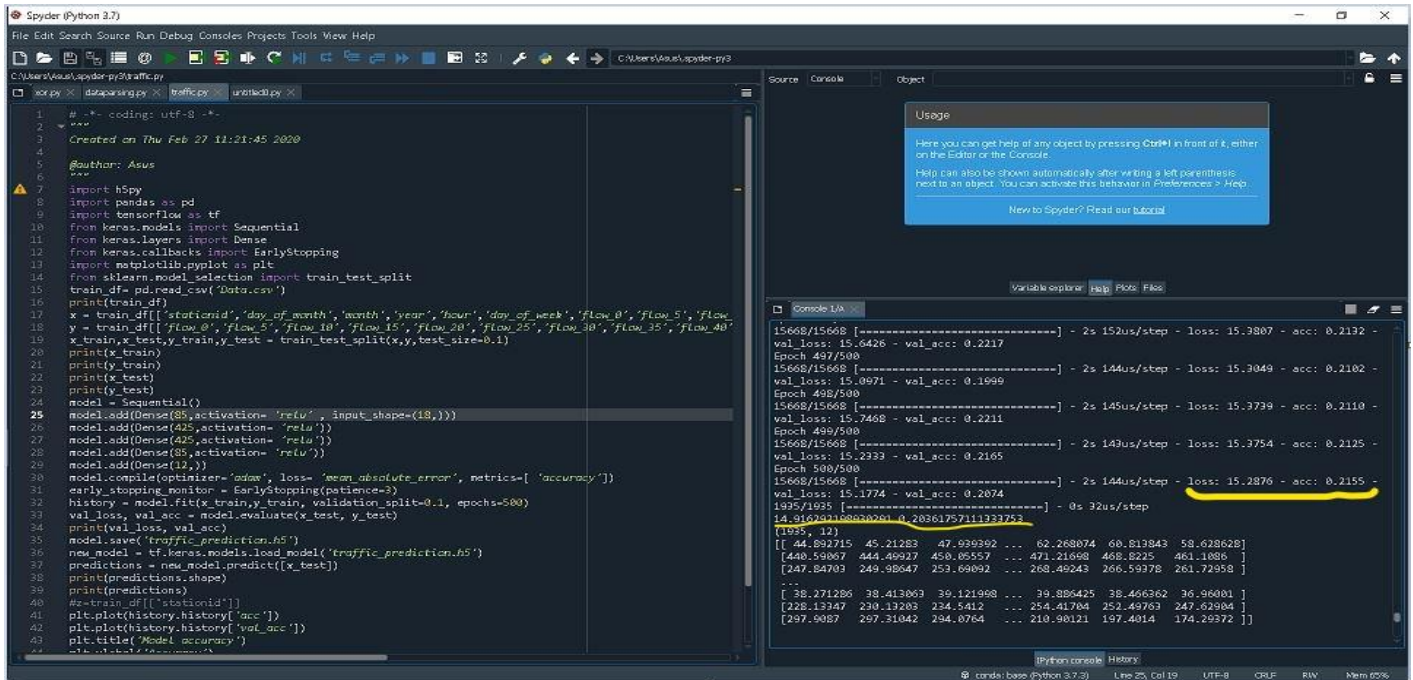


Figure 18: Accuracy and loss metrics for first approach.

accuracy in this range can be considered as very low and therefore this approach does not provide reliable results and therefore at such an accuracy level this approach cannot be implemented for traffic prediction.

4.3.2 Second Approach

In the second approach the YOLO algorithm is applied to detect the vehicles present on the road in the videos and the images provided to the algorithm as input. Bounding boxes are drawn surrounding the detected vehicles. And in videos on the basis of the average speed of the detected objects and in images on the basis of number of vehicles detected the algorithm classifies if there is congestion or a clear road. In Figure 19 we can observe that the algorithm shows that a clear road can be seen in the input video named overpass.mp4. In Figure 20 we can see a still from the video overpass_output.avi which is the output video generated by the algorithm.

```
Anaconda Prompt - C:\Users\Asus\Anaconda3\Scripts\activate.bat C:\Users\Asus\Anaconda3
C:\Users\Asus>python yolo_video.py --input videos/overpass.mp4 --output output/overpass_output.avi --yolo yolo-coco
[INFO] loading YOLO from disk...
[INFO] 812 total frames in video
[INFO] single frame took 0.8105 seconds
[INFO] estimated total time to finish: 658.1530
[INFO] cleaning up...
The Road is clear
Average Speed x is: 0
Average Speed y is: 437.87272178059413
C:\Users\Asus>
```

Figure 20: The algorithm giving an output showing the road is clear for the video overpass.mp4.

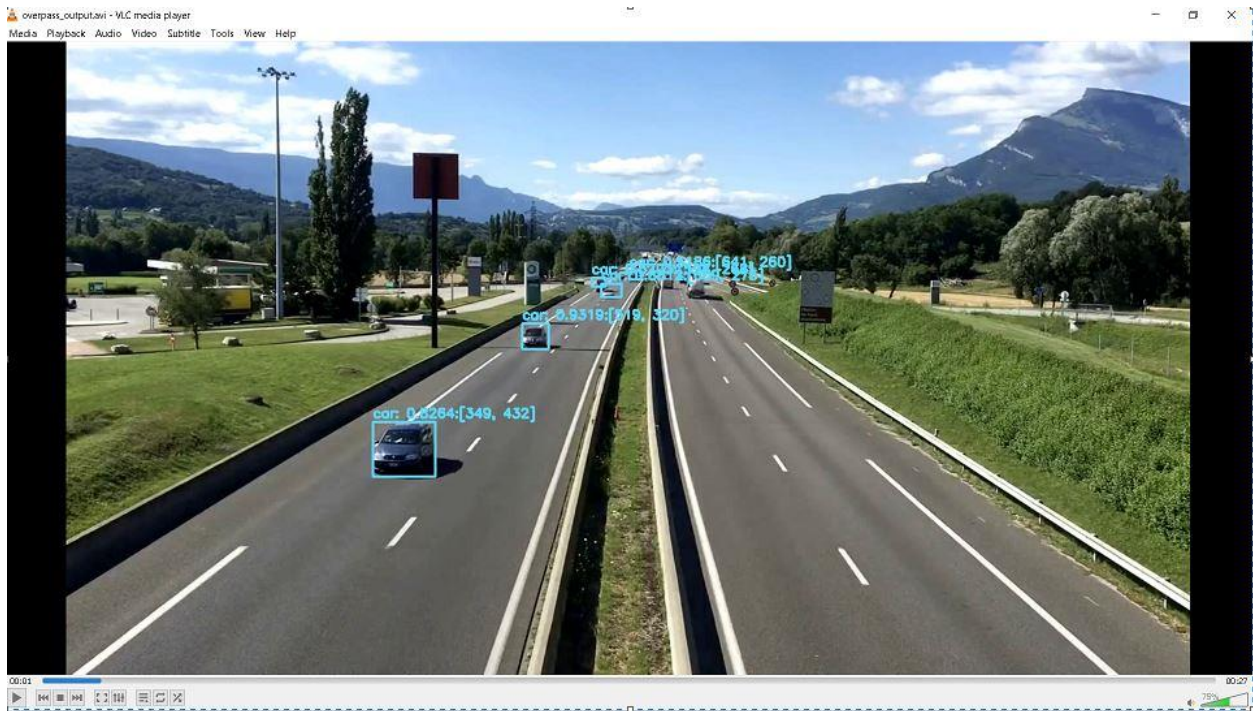


Figure 19: A still image from video overpass_output.avi.

In Figure 21 we can observe that the algorithm shows that there is a congestion on the road that

```
Anaconda Prompt - C:\Users\Asus\Anaconda3\Scripts\activate.bat C:\Users\Asus\Anaconda3

C:\Users\Asus>python yolo_video.py --input videos/traffic.mp4 --output output/traffic_output.avi --yolo yolo-coco
[INFO] loading YOLO from disk...
[INFO] 72 total frames in video
[INFO] single frame took -1588063166.8590 seconds
[INFO] estimated total time to finish: -114340548013.8485
[INFO] cleaning up...
There is traffic congestion
Average Speed x is: 0
Average Speed y is: 0
C:\Users\Asus>
```

Figure 22: The output of the algorithm showing traffic congestion on the road.

can be seen in a video named traffic.mp4. Figure 22 shows the still from video named

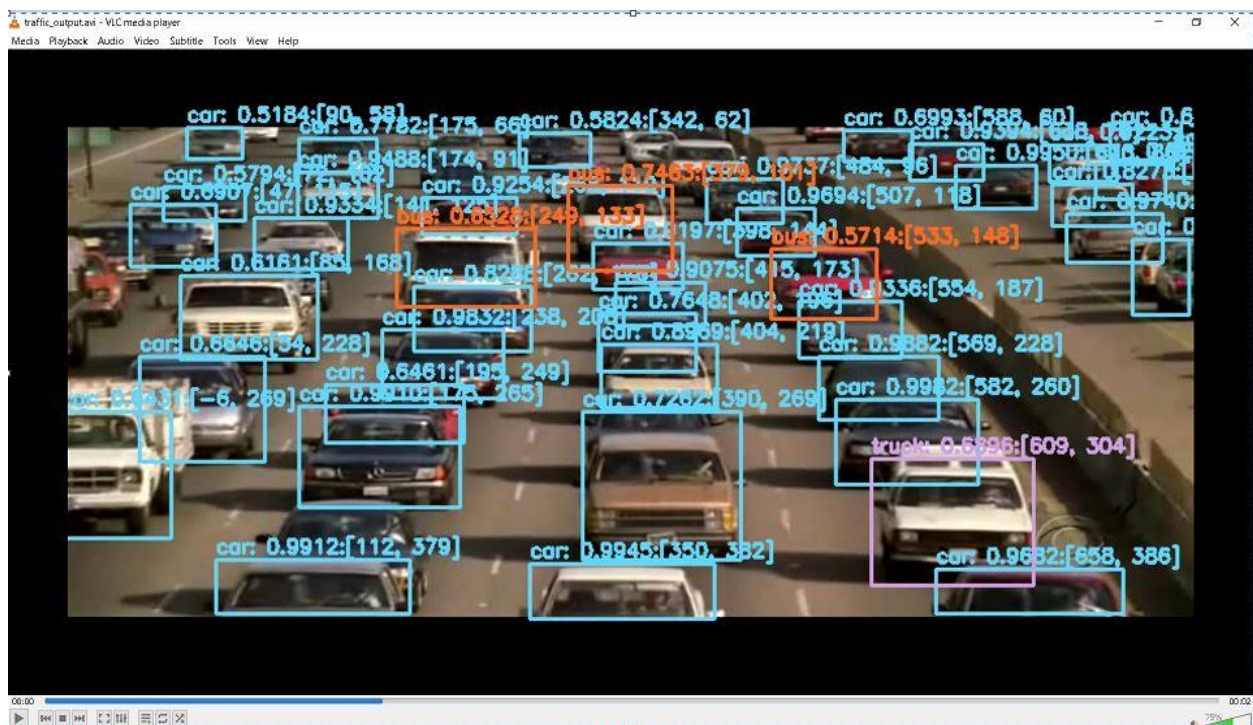


Figure 21: A still image from the video traffic_output.avi.

classifies both the images into congested and not congested respectively.

4.4 Comparison with existing Approaches

In general, there are three main approaches to solve the problem of traffic prediction. First approach is to predict the trends in traffic by providing the data collected over a large period of time from the roads in consideration. This approach is used by TomTom app for their traffic prediction algorithm. In the second approach images and videos collected from the roads are used for detection of the objects using YOLO object detection algorithm. This approach was not found to be used by many traffic prediction algorithms. Lastly, the third approach to traffic prediction is by collecting location data from GPS of the devices in real time and by calculating the number of active GPSs in a certain area and the average speed at which they are travelling to predict traffic in that area. Google uses this third approach in their traffic prediction algorithm. Out of these three approaches, two of them are implemented by us in this project

In the first approach to train a model to predict accurate trends in traffic a very large dataset is required. Therefore, sometimes this approach is aid to be using Big Data. This approach uses historical data to predict the traffic on a road. Therefore, it is not real-time and there is a very huge chance of error. Although, this approach considers the change in traffic trends for different days of the week and different time of the year. It does not account for the accidents or repair work on the considered road. Hence, this can cause errors in prediction.

The second approach works in real-time as the input videos and images can be real-time videos and images collected from the cameras installed at proper locations by the government. The only drawback of this approach is that it is a very complex system and therefore it needs a computer with very high computing power and this computer has to be maintained from time to time as any lag in the calculation can cause an error in traffic prediction. Therefore, this approach can be very costly in comparison with the other approaches.

In the third approach a user's location data and average speed is collected from the GPS of the user's device. This approach is used by companies like Google and Waze. The current data collected from the GPS is then clubbed with the historical data collected over a large period of time are used to predict the traffic in the considered area. It can be said that the larger the number of users using the app provided by these companies at a point of time provides a better prediction

of traffic. This is very good approach to traffic prediction but still it is vulnerable to errors. As seen recently when a lot of phones are brought in a certain area with Google Maps open on them and in steady state then the Google Maps app predicts a congestion in that area. This prediction will be erroneous.

The comparison of all the three approaches can be made based on metrics like accuracy, precision, recall and F1 score. The accuracy for the first model was found out to be 21% this was the only metric observed for the first approach. For the second approach the accuracy for object detection by YOLO algorithm was found out to be 97% ^[17]. However, the mean average precision for this algorithm with a threshold of 0.5 IoU turned out to be 61.1 ^[12] (IoU is the ratio of area of overlap between the label of the considered image and the prediction for that image made by the algorithm to the area of union between the same). As, for the algorithm used by Google maps these metrics are accuracy of 74.9%, precision was 71.9 ^[18]. It should be noted that the accuracy and precision metrics for the Google Maps algorithm is only for the model that uses the historical data of the number of cars that is provided by the dataset used by them. It is this way because the current data from the GPS of the mobile devices is clubbed with the output of the said model therefore, the accuracy and precision of the complete algorithm is not known and was not found.

Table 5: Comparison between all the three approaches.

Approach	First	Second	Third
Data Collection Method	Stations installed on the roads	Cameras installed on the roads	GPS of mobile devices of users
Data Time Frame	Historical	Current data	Historical & Current data
Cost Effectiveness	Low	High	Medium
Error Probability	Very High	Low	Low
Accuracy	21%	97%	74.9%
Precision	N.A.	61.1	71.9

Therefore, by discussing three different approaches we can say that none of them is a full proof approach. All the three approaches have some room for error. The differences between the three approaches can be properly understood by the Table 5.

4.5 Discussion

The experiments implemented in this project describes two approaches that can be taken to solve the problem of traffic prediction. The first approach is used commercially but the second approach is not used commercially as per our research. On implementing both the approaches it was found that the first approach was prone to more errors in comparison to the second approach. Also, the first approach could provide predictions in real time it was not essentially a real-time system. However, the second approach was found to be a real-time system. The first approach had a major flaw as it did not consider some parameters which can prove to be fatal in terms of accurate prediction. While the chances of error in the second approach were less the computation power it used was much larger in comparison to the first approach. The first approach also required high computation power but it was required only for the training of the model. The model in the first approach will have to be trained with new data after some specific period of time. But the second approach needed to work with high computation power at all times as it dealt with videos and images.

Chapter 5

Conclusion and Future Scope

After implementing two approaches for “Traffic Prediction using Deep Learning”. It was seen that the first approach used data in terms of number of vehicles, average speed of the traffic and density of the vehicles on the road, which was collected from stations installed on the roads was used to predict future trends in these parameters on those roads. While the second approach used images and videos collected by cameras installed on the roads and processed these videos and images to predict traffic. It was observed that for the first approach to provide accurate predictions the dataset used had to be very large. Even after using a dataset which spanned the duration of one month and over 19000 entries, the accuracy of the model turned to be as low as 21%. In the second approach the YOLO algorithm used for object detection was already trained on MSCOCO dataset and therefore training it again was not required. The measured accuracy for this pre-trained model for detecting objects in images and videos is over 97% ^[12] with a precision of over 61.1 for a threshold IoU of 0.5 ^[17]. Therefore, simply based on the accuracy of both the approach it can be clearly seen that the second approach can produce more accurate results. However, as video processing was involved in the second approach therefore the computation power required was very high. A 30 second video used in this approach was divided by the algorithm in 812 frames and on the computer used by us each frame needed an average computing time of 0.8105 seconds, thus a mere 30-second video required a time of over 11 minutes to be processed and produce an output. For real-time prediction of traffic this time requirement is very large and thus much more graphically powerful computer would be required for this approach to work successfully and in real-time.

5.1 Future Scope

Traffic prediction using the second approach can give out very accurate results. The results produced by this approach are very promising and can be used to show the condition of any roads on which this system is active, to the people with the help of an app. Another crucial requirement of this approach is that it needs input such as videos and images from each and every road that is to be considered for traffic prediction. So, instead of installing cameras on each

and every road of the country we can just try to get these inputs with the help of the IRNSS satellite system of India. The object detection approach needs a lot of resources and infrastructure this makes it a very costly approach, but still it can provide very accurate results.

Chapter 6

References

- [1]. Medium. 2020. Introducing Deep Learning And Neural Networks — Deep Learning For Rookies (1). [online] Available at: <<https://towardsdatascience.com/introducing-deep-learning-and-neural-networks-deep-learning-for-rookies-1-bd68f9cf5883>> [Accessed 29 April 2020].
- [2]. Parsers.me. 2020. Deep Learning & Machine Learning: What’S The Difference? – Parsers. [online] Available at: <<https://parsers.me/deep-learning-machine-learning-whats-the-difference/>> [Accessed 29 April 2020].
- [3]. Mathworks.com. 2020. What Is Deep Learning? | How It Works, Techniques & Applications. [online] Available at: <<https://www.mathworks.com/discovery/deep-learning.html>> [Accessed 29 April 2020].
- [4]. Medium. 2020. Parallel And Distributed Deep Learning: A Survey. [online] Available at: <<https://towardsdatascience.com/parallel-and-distributed-deep-learning-a-survey-97137ff94e4c?source=rss-b2d5b3af50d7-----3>> [Accessed 29 April 2020].
- [5]. MIT News. 2020. Explained: Neural Networks. [online] Available at: <<http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>> [Accessed 29 April 2020].
- [6]. En.wikipedia.org. 2020. Traffic Congestion. [online] Available at: <https://en.wikipedia.org/wiki/Traffic_congestion> [Accessed 29 April 2020].
- [7]. Theshybulb.com. 2020. Neural Network Implementation Of An XOR Gate. [online] Available at: <<http://theshybulb.com/2017/09/27/xor-neural-network.html>> [Accessed 29 April 2020].
- [8]. Brownlee, J., 2020. Handwritten Digit Recognition Using Convolutional Neural Networks In Python With Keras. [online] Machine Learning Mastery. Available at: <<https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/>> [Accessed 29 April 2020].

- [9]. Medium. 2020. Image Classification In 10 Minutes With MNIST Dataset. [online] Available at: <<https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d>> [Accessed 29 April 2020].
- [10]. Medium. 2020. YOLO — You Only Look Once, Real Time Object Detection Explained. [online] Available at: <<https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006>> [Accessed 28 April 2020].
- [11]. Aqib, M., Mehmood, R., Alzahrani, A., Katib, I., Albeshri, A. and Altowaijri, S., 2020. Smarter Traffic Prediction Using Big Data, In-Memory Computing, Deep Learning And Gpus. [Accessed 28 April 2020]
- [12]. Redmon, J., 2015. [online] Pjreddie.com. Available at: <<https://pjreddie.com/media/files/papers/yolo.pdf>> [Accessed 28 April 2020]
- [13]. M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015. [Accessed 28 April 2020]
- [14]. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. [Accessed 28 April 2020]
- [15]. Praveen, “The world's biggest road networks,” *Verdict Traffic*, 28-Jan-2020. [Online]. Available: <https://www.roadtraffic-technology.com/features/featurethe-worlds-biggest-road-networks-4159235/>. [Accessed: 04-May-2020].
- [16]. “Common Objects in Context,” *COCO*. [Online]. Available: <http://cocodataset.org/>. [Accessed: 04-May-2020].
- [17]. R. Khan and R. Debnath, “Multi Class Fruit Classification Using Efficient Object Detection and Recognition Techniques,” *International Journal of Image, Graphics and Signal Processing*, vol. 11, no. 8, pp. 1–18, Aug. 2019.
- [18]. “Intelligent Systems and Applications,” *Google Books*. [Online]. Available: <https://books.google.co.in/books?id=WuN3DwAAQBAJ&pg=PA1152&lpg=PA1152&dq=what+is+the+F1+score+of+the+model+used+by+google+maps+for+traffic+prediction&source=bl&ots=QTaa6kYYet&sig=ACfU3U1qHCOIfiEw51iaUaWDRkTfJ2Ewhg&hl=en&sa=X&ved=2ahUKEwi886H6pprpAhWVheYKHcb7ASYQ6AEwAnoECAoQAQ#v=onepage&q&f=false>. [Accessed: 05-May-2020].

- [19]. H. Bahuleyan and L. D. Vanajakshi, "Arterial Path-Level Travel-Time Estimation Using Machine-Learning Techniques," *Journal of Computing in Civil Engineering*, vol. 31, no. 3, p. 04016070, 2017.
- [20]. A. Krishna K.V.S, A. K, A. Swaraj, S. D. Patil, and G. K. Shyam, "Smart Traffic Analysis using Machine Learning," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 8, no. 2249, May 2019.
- [21]. "Convolutional Neural Network (CNN) for Image recognition." [Online]. Available: <https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/>. [Accessed: 06-May-2020].
- [22]. "A Comprehensive Guide to Convolutional Neural Networks ..." [Online]. Available: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. [Accessed: 06-May-2020].
- [23]. "Convolutional Neural Networks (CNN) from Scratch," *courses.analyticsvidhya.com*. [Online]. Available: 3. https://courses.analyticsvidhya.com/courses/convolutional-neural-networks-cnn-from-scratch?utm_source=blog&utm_medium=architecture-of-convolutional-neural-networks-simplified-demystified/. [Accessed: 06-May-2020].
- [24]. Sensys Networks SensTraffic. [Online]. Available: <https://sensysnetworks.com/products/senstraffic>. [Accessed: 10-May-2020].

APPENDIX A

#Code to calculate the average value of speed of the vehicles

if initialx == x or initialy == y:

start = time.time()

print("Start: ",start , x, y)

if finalx == x or finaly == y:

end =time.time()

elif initialx == x and finalx != x:

ttime = 0

print("End:" ,end , x, y)

ttime = end - start

print("total time: ",ttime)

elif initialy == y and finaly != y:

ttime = 0

print("End:" ,end , x, y)

ttime = end - start

print("total time: ",ttime)

if ttime > 0:

speedx = (finalx - initialx)/ttime

speedy = (finaly - initialy)/ttime

avgttime = (avgttime + ttime)/2

```
avgspeedx = (speedx + avgspeedx)/2  
  
    avgspeedy = (speedy + avgspeedy)/2  
  
elif ttime < 0:  
  
to=to+1  
  
    #print("There is traffic congestion")
```

APPENDIX B

```
#Code for data processing
```

```
import pandas as pd
```

```
import datetime
```

```
import re
```

```
import numpy as np
```

```
#Read Data
```

```
data=pd.read_csv('dataset.csv',header=None)
```

```
station_id=data.loc[:,1]
```

```
#Processing timezone as described in paper
```

```
timezone=data.iloc[:,0]#get the entire timezone column from the data
```

```
#Splitting timezone string type into mm-dd-yyyy and hh-mm format
```

```
time_zone_split=[]
```

```
for i in range(len(timezone)):
```

```
    time_zone_split.append(re.split(r'[:,\s]\s*',timezone[i]))
```

```
dayOfMonth=[]
```

```
month=[]
```

```
year=[]
```

```
hours=[]
```

```
week_days=[]
```

```
minutes=[]
```

```

for time_date in time_zone_split:

    date=time_date[0]

    time=time_date[1]

    #Getting weekday info from date

    week_days.append(datetime.datetime.strptime(date, '%m/%d/%Y').weekday()+1)#getting
week days as described(sunday=1 and saturday=7)

#Getting the month, day and year info from date

    date_info=re.split('/',date)

    dayOfMonth.append(int(date_info[1]))

    month.append(int(date_info[0]))

    year.append(int(date_info[2]))

#Getting the hours info from time

    hour=re.split(':',time)

    hours.append(int(hour[0]))

    minutes.append(int(hour[1]))

#generate dataframe for the first date-time features

minutes_copy=minutes#Used later

station_id_copy=station_id#Used Later

station_id=pd.DataFrame(station_id)

dayOfMonth=pd.DataFrame(dayOfMonth)

month=pd.DataFrame(month)

```

```

year=pd.DataFrame(year)

hours=pd.DataFrame(hours)

week_days=pd.DataFrame(week_days)

minutes=pd.DataFrame(minutes)

date_time_final_features=pd.concat((station_id,dayOfMonth,month,year,hours,week_days,minutes),axis=1)

date_time_final_features.columns=['stationid','day_of_month','month','year','hour','day_of_week','minutes']

#converting data from long to wide and generating Flow_00 to Flow_55

unique_min=sorted(list(set(minutes_copy)))

flows_per_5_min=np.zeros((len(date_time_final_features),len(unique_min)))

for i in range(len(flows_per_5_min)):

    k=int(date_time_final_features.iloc[i]['minutes']/5)

    flows_per_5_min[i][k]=data.iat[i,9]

flows_per_5_min=pd.DataFrame(flows_per_5_min)

columns=[]

for i in range(flows_per_5_min.shape[1]):

    columns.append('flow_'+str(i*5))

flows_per_5_min.columns=columns

#concat the datetime dataframe and flow dataframe on stationid as generated above namely,
date_time_final_features and flows_per_5_min

data_parsing=pd.concat((date_time_final_features,flows_per_5_min),axis=1)

```

```

#Remove the minutes column as we no longer require this in final parsed data

data_parsing=data_parsing.drop(['minutes'],axis=1)

#Group data and aggregate(sum) flows by station_id, hour, day_of_month, year, month and
day_of_week to include logic for zero

grouped_flows=pd.DataFrame()

for i in range(len(unique_min)):

    flow_5_diff=pd.DataFrame(data_parsing.groupby(['stationid','day_of_month','month','year','hour'
    ,'day_of_week'])['flow_'+str(unique_min[i])].sum())

    grouped_flows=pd.concat((grouped_flows,flow_5_diff),axis=1)

#Transform data and remove index column

grouped_flows=grouped_flows.reset_index()

#Final output after Data processing and parsing

final_data_parsed=grouped_flows

#Saving the output to a file

final_data_parsed.to_csv('Data_parsing_output.csv',index=False)

```