

# Utilizing Computer Vision and Data Mining for Predicting Road Traffic Congestion

Mohsen Amoei

A Thesis  
in  
The Concordia Institute  
for  
Information Systems Engineering

Presented in Partial Fulfillment of the Requirements  
For the Degree of  
Master of Applied Science (Quality Systems Engineering) at  
Concordia University  
Montréal, Québec, Canada

March 2020

© Mohsen Amoei, 2020

CONCORDIA UNIVERSITY  
School of Graduate Studies

This is to certify that the thesis prepared

By: **Mohsen Amoei**

Entitled: **Utilizing Computer Vision and Data Mining for Predicting Road  
Traffic Congestion**

and submitted in partial fulfillment of the requirements for the degree of

**Master of Applied Science (Quality Systems Engineering)**

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

\_\_\_\_\_ Chair  
*Dr. Chun Wang*

\_\_\_\_\_ External  
*Dr. Ciprian Alecsandru*

\_\_\_\_\_ Examiner  
*Dr. Farnoosh Naderkhani*

\_\_\_\_\_ Thesis Supervisor  
*Dr. Anjali Awasthi*

Approved by \_\_\_\_\_  
Dr. Mohammad Mannan, Graduate Program Director

March 18, 2020 \_\_\_\_\_  
Dr. Amir Asif, Dean  
Gina Cody School of Engineering and Computer Science

# Abstract

## Utilizing Computer Vision and Data Mining for Predicting Road Traffic Congestion

Mohsen Amoei

Traffic Congestion wastes time and energy, which are the two most valuable commodities of the current century. It happens when too many vehicles try to use a transportation infrastructure without having enough capacity. However, researches indicate that adding extra lane without studying the future consequences does not improve the situation. Our goal is to add another layer of information to the traffic data, find which type of vehicles are contributing to road traffic congestion, and predict future road traffic congestion and demands based on the historical data.

We collected more than 400,000 images from traffic cameras installed in Autoroute 40, in the city of Montreal. The images were collected for five consecutive weeks from different locations from April 14, 2019, up until May 18, 2019. To process these images and extract useful information out of them, we created an object detection and classification model using the Faster RCNN algorithm. Our goal was to be able to detect different types of vehicles and see if we have traffic congestion in an image. In order to improve the accuracy and reduce the error rate, we provided multiple examples with different conditions to the model. By introducing blurry, rainy, and low light images to the model, we managed to build a robust model that could do the detection and classification task with excellent accuracy.

Furthermore, by extracting the information from the collected images, we created a dataset of the number of vehicles in each location. After analyzing and visualizing the data, we find out the most congested areas, the behavior of the traffic flow during the day, peak hours, the contribution of each type of vehicle to the traffic, seasonality of the data, and where we can see each type of vehicle the most.

Finally, we managed to predict the total number of congestion incidents for seven days based on historical data. Besides, we were able to predict the total number of different types of vehicles on the road as well. In order to do this task, we developed multiple Regression,

Deep Learning, and Time Series Forecasting models and trained them with our vehicle count dataset. Based on the experimental results, we were able to get the best predictions with the Deep Learning models and succeeded in predicting future road traffic congestion with excellent accuracy.

**Keywords: Computer Vision, Data Mining, Time Series Forecasting, Road Traffic Congestion, Deep Learning**

# Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Anjali Awasthi, for her constant support throughout my journey at Concordia University.

I would also like to thank my fellow labmates, for their support. Taiwo, Hassan, Negar, Suganya, Ujjwal, Akhil, and Safwan. It was fun being your labmate.

I would like to thank my colleagues at SAP company for helping me and sharing their knowledge with me. Benjamin, Marc-Andre, Andre, Urmil, Sabrina, Tarandeep, Bianca, Nitheesh and Ishrar, I'm lucky to have the chance to work alongside you all.

I want to thank my friends for always being there and helping me coping with hard times. Ramtin, Parsa, and Adam, thanks for all the good times.

I must thank my dear Behshid for her unconditional and unparalleled love and support. It would be impossible to do it without you.

I must express my profound gratitude to my parents and my dear sister, Mehrnaz, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

# Contents

<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Research Objective . . . . .	2
1.2 Contributions . . . . .	3
1.3 Thesis Structure . . . . .	4
<b>2 Background</b>	<b>5</b>
2.1 Introduction . . . . .	5
2.2 Road Traffic Congestion . . . . .	5
2.3 Deep Learning Platforms and Libraries . . . . .	6
2.4 Performance Metrics . . . . .	10
2.4.1 Estimation and Prediction Performance Metrics . . . . .	10
2.5 Object Detection and Classification Algorithms . . . . .	12
2.5.1 Convolutional Neural Networks (CNN) . . . . .	12
2.5.2 Region-Based Convolutional Neural Networks (R-CNN) . . . . .	13
2.6 Time Series Forecasting Algorithms and Models . . . . .	16
2.6.1 Regression Analysis . . . . .	17
2.6.2 Deep Learning Methods . . . . .	18

2.6.3	Time series Forecasting Models . . . . .	22
2.7	Conclusion . . . . .	25
<b>3</b>	<b>Literature Review</b>	<b>26</b>
3.1	Introduction . . . . .	26
3.2	Object Detection and Classification . . . . .	27
3.2.1	Application Focused Researches . . . . .	27
3.2.2	Model Improvement Oriented Researches . . . . .	28
3.3	Time Series Forecasting . . . . .	29
3.3.1	Road Traffic Prediction Related Researches . . . . .	29
3.3.2	Non Road Traffic Prediction Related Researches . . . . .	30
3.4	Conclusion . . . . .	31
<b>4</b>	<b>Methodology</b>	<b>32</b>
4.1	Introduction . . . . .	32
4.2	High-level Model Description . . . . .	32
4.3	Data Acquisition . . . . .	34
4.3.1	Data Collection . . . . .	34
4.4	Object Detection and Classification . . . . .	37
4.4.1	Data Preparation . . . . .	37
4.4.2	Supervised pre-training . . . . .	40
4.4.3	Faster R-CNN Model . . . . .	41
4.5	Time Series Traffic Forecasting . . . . .	44
4.5.1	Data Preparation . . . . .	44
4.5.2	Regression Analysis . . . . .	45
4.5.3	Deep Learning Methods . . . . .	47
4.5.4	Time Series Forecasting Models . . . . .	51

4.6	Conclusion . . . . .	53
<b>5</b>	<b>Experimental Results</b>	<b>54</b>
5.1	Introduction . . . . .	54
5.2	Object Detection and Classification Models . . . . .	54
5.2.1	Data Cleaning Image Classification Model . . . . .	55
5.2.2	Vehicle Detection and Classification Model . . . . .	57
5.3	Data Visualization . . . . .	71
5.4	Time Series Forecasting . . . . .	80
5.4.1	Regression Analysis Models . . . . .	80
5.4.2	Deep Learning Models . . . . .	96
5.4.3	Time Series Forecasting Models . . . . .	107
5.5	Conclusion . . . . .	118
<b>6</b>	<b>Conclusion and Future Works</b>	<b>120</b>
6.1	Conclusion . . . . .	120
6.2	Future Works . . . . .	122
	<b>Bibliography</b>	<b>124</b>



# List of Figures

1	Traffic status of the city of Montreal on 18:36 February 25th, 2020. Courtesy of (TomTom, 2019) . . . . .	7
2	LabelImg Platform Environment . . . . .	9
3	Object detection system overview for R-CNN, Courtesy of (Girshick et al., 2014) . . . . .	13
4	Fast R-CNN architecture, Courtesy of (Girshick, 2015) . . . . .	15
5	Region Proposal Network (RPN) architecture, Courtesy of (Ren et al., 2015)	16
6	Model of an Artificial Neuron. Courtesy of (Agatonovic-Kustrin and Beresford, 2000) . . . . .	20
7	Model of an Artificial Neural Network. Courtesy of (Anderson and McNeill, 1992) . . . . .	20
8	On the left side we can see a typical RNN and on the right side is the unfolding version. Courtesy of (Tai and Liu, 2016). . . . .	21
9	LSTM Memory Cell with Gates. Courtesy of (Soutner and Müller, 2013). . . . .	22
10	Analyst in the loop model. Courtesy of (Taylor and Letham, 2018) . . . . .	24
11	The Methodology Steps. From left to Right: (1) Getting Data from Traffic Cameras, (2) Using Computer Vision to Extract Information, (3) Creating the Historical Dataset of the Vehicle and Congestion Count, (4) Develop Several Time Series Forecasting Models, (5) Visualize the Predictions, Compare the Results and Find the Best Model . . . . .	33

12	Traffic cameras provided by Quebec511 in city of Montreal. Courtesy of (Google-Maps, 2019)(Quebec511, 2019) . . . . .	35
13	Auto route 40. Courtesy of (Google-Maps, 2019) . . . . .	36
14	Images Collected from Traffic Cameras. Courtesy of (Quebec511, 2019) . .	36
15	Traffic Camera Footage is Unavailable(No-Footage). Courtesy of (Quebec511, 2019) . . . . .	38
16	An Example of an Image with All the Object Marked in it Using LabelImg	40
17	CNN Time Series Forecasting Model Architecture Flowchart . . . . .	49
18	LSTM Time Series Forecasting Model Architecture Flowchart . . . . .	51
19	Cross Entropy Loss of the Inception v3 Data Cleaning Image Classification Model . . . . .	55
20	Accuracy of the Inception v3 Data Cleaning Image Classification Model . .	56
21	Classification(Objectness) Loss Of the Faster RCNN Model . . . . .	59
22	Localization(Detection) Loss Of the Faster RCNN Model . . . . .	60
23	The Sum of the Classification and Localization Loss(Total Loss) . . . . .	60
24	An Example of the Object Detection Model Output on Images with Normal Day Light Setting . . . . .	61
25	An Example of the Object Detection Model Output on Images with Normal Night Light Setting . . . . .	62
26	An Example of the Object Detection Model Output on Images with Low Light Setting . . . . .	63
27	An Example of the Object Detection Model Output on Images with Rain Condition . . . . .	64
28	An Example of the Object Detection Model Output on Images with Low Light and Rain Condition . . . . .	64

29	An Example of the Object Detection Model Output on Images with Complex Road Architecture . . . . .	65
30	An Example of the Object Detection Model Output on Images with Complex Road Architecture and Low Light Setting . . . . .	66
31	An Example of the Object Detection Model Output on Images with Personal Cars In Them . . . . .	67
32	An Example of the Object Detection Model Output on Images with Trucks In Them . . . . .	67
33	An Example of the Object Detection Model Output on Images with Buses In Them . . . . .	68
34	An Example of the Object Detection Model Output on Images with Road Traffic Congestion In Them . . . . .	69
35	An Example of the Object Detection Model Output on Images with Lots of Vehicles in Them but No Traffic Congestion . . . . .	70
36	The Vehicle Count Dataset . . . . .	70
37	Total Number of Personal Car in each Hour of the Day . . . . .	71
38	Total Number of Trucks in each Hour of the Day . . . . .	72
39	Total Number of Buses in each Hour of the Day . . . . .	73
40	Total Number of Recorded congestion in each Hour of the Day . . . . .	74
41	Total Number of Personal Cars in each Location During the Month . . . . .	75
42	Total Number of Trucks in each Location During the Month . . . . .	76
43	Total Number of Buses in each Location During the Month . . . . .	77
44	Total Number of Congestion Recorded in each Location During the Month . . . . .	78
45	Heatmap of Correlation Between the Features . . . . .	79
46	Prediction Results for Number of Personal Cars with Linear Regression Model . . . . .	81

47	Prediction Results for Number of Trucks with Linear Regression Model . .	82
48	Prediction Results for Number of Buses with Linear Regression Model . . .	83
49	Prediction Results for Number of Recorded congestion with Linear Regres- sion Model . . . . .	85
50	Prediction Results for Total Number of Personal Car with Polynomial Re- gression Model with Degree Three . . . . .	87
51	Prediction Results for Total Number of Trucks with Polynomial Regression Model with Degree Two . . . . .	88
52	Prediction Results for Total Number of Bus with Polynomial Regression Model with Degree Two . . . . .	89
53	Prediction Results for Total Number of Personal Car with Polynomial Re- gression Model with Degree Two . . . . .	90
54	Prediction Results for Number of Personal Cars with Random Forest Re- gression Model with 18 Estimator . . . . .	92
55	Prediction Results for Number of Trucks with Random Forest Regression Model with One Estimator . . . . .	93
56	Prediction Results for Number of Buses with Random Forest Regression Model with Four Estimator . . . . .	94
57	Prediction Results for Traffic Congestion with Random Forest Regression Model with Five Estimator . . . . .	95
58	Prediction Results for Total Number of Personal Car with CNN Model 2 . .	98
59	Prediction Results for Total Number of Trucks with CNN Model 3 . . . . .	99
60	Prediction Results for Total Number of Buses with CNN Model 3 . . . . .	101
61	Prediction Results for Total Number of Road Traffic Congestion with CNN Model 1 . . . . .	102
62	Prediction Results for Total Number of Personal Cars with LSTM Model 1	104

63	Prediction Results for Total Number of Trucks with LSTM Model 2 . . . . .	105
64	Prediction Results for Total Number of Buses with LSTM Model 4 . . . . .	106
65	Prediction Results for Total Number of Road Traffic Congestion with LSTM Model 3 . . . . .	108
66	Prediction Results for Total Number of Personal Cars with ARIMA Model .	109
67	Prediction Results for Total Number of Trucks with ARIMA Model . . . . .	111
68	Prediction Results for Total Number of Buses with ARIMA Model . . . . .	112
69	Prediction Results for Total Number of Road Traffic Congestion with ARIMA Model . . . . .	113
70	Prediction Results for Total Number of Personal Cars with Prophet Model .	114
71	Prediction Results for Total Number of Trucks with Prophet Model . . . . .	115
72	Prediction Results for Total Number of Buses with Prophet Model . . . . .	117
73	Prediction Results for Total Number of Road Traffic Congestion with Prophet Model . . . . .	118

# List of Tables

1	Collected Information from Quebec 511 website . . . . .	37
2	Table of all the footages we expect from traffic cameras . . . . .	39
3	Weather Situation in City of Montreal During the Year (NOAA, 2019). . .	41
4	Convolutional Neural Network Model Architectures for Time Series Fore- casting Models . . . . .	49
5	Long Short Term Memory Time Series Forecasting Models . . . . .	50
6	Table of the Results of the Data Cleaning Image Classification Model on Different Images . . . . .	57
7	Faster RCNN Model Hyper-Parameters . . . . .	58
8	Momentum Optimizer Hyper-Parameters . . . . .	58
9	Faster RCNN Model Loss Function Hyper-Parameters . . . . .	58
10	Prediction Results and the Performance Metrics of the Linear Regression Model on Total Number of the Personal Car . . . . .	81
11	Prediction Results and the Performance Metrics of the Linear Regression Model on Total Number of the Trucks . . . . .	83
12	Prediction Results and the Performance Metrics of the Linear Regression Model on Total Number of the Buses . . . . .	84
13	Prediction Results and the Performance Metrics of the Linear Regression Model on Total Number of Road Traffic Congestion . . . . .	85

14	Prediction Results and the Performance Metrics of the Polynomial Regression Model on Total Number of Personal Car . . . . .	86
15	Prediction Results and the Performance Metrics of the Polynomial Regression Model on Total Number of Trucks . . . . .	88
16	Prediction Results and the Performance Metrics of the Polynomial Regression Model on Total Number of the Buses . . . . .	89
17	Prediction Results and the Performance Metrics of the Polynomial Regression Model on Total Number of Road Congestion . . . . .	90
18	Prediction Results and the Performance Metrics of the Random Forest Regression Model on Total Number Personal Cars . . . . .	91
19	Prediction Results and the Performance Metrics of the Random Forest Regression Model on Total Number of Trucks . . . . .	93
20	Prediction Results and the Performance Metrics of the Random Forest Regression Model on Total Number of Buses . . . . .	95
21	Prediction Results and the Performance Metrics of the Random Forest Regression Model on Total Number of Road Traffic Congestion . . . . .	96
22	Convolutional Neural Network Time Series Forecasting Models . . . . .	97
23	Convolutional Neural Network Time Series Forecasting Models Results with Performance Metrics for Personal Cars . . . . .	98
24	Convolutional Neural Network Time Series Forecasting Models Results with Performance Metrics for Trucks . . . . .	99
25	Convolutional Neural Network Time Series Forecasting Models Results with Performance Metrics for Buses . . . . .	100
26	Convolutional Neural Network Time Series Forecasting Models Results with Performance Metrics for Road Traffic Congestion . . . . .	101
27	Long Short Term Memory Time Series Forecasting Models . . . . .	103

28	Long Short Term Memory Time Series Forecasting Models Results and Performance Metrics for Total Number of Personal Cars . . . . .	103
29	Long Short Term Memory Time Series Forecasting Models Results and Performance Metrics for Total Number of Trucks . . . . .	105
30	Long Short Term Memory Time Series Forecasting Models Results and Performance Metrics for Total Number of Buses . . . . .	106
31	Long Short Term Memory Time Series Forecasting Models Results and Performance Metrics for Total Number of Road Traffic Congestion . . . . .	107
32	ARIMA Model Results and Performance Metrics for Total Number of Personal Cars . . . . .	109
33	ARIMA Model Results and Performance Metrics for Total Number of Trucks	110
34	ARIMA Model Results and Performance Metrics for Total Number of Buses	111
35	ARIMA Model Results and Performance Metrics for Total Number of Road Traffic Congestion . . . . .	113
36	Prophet Model Results and Performance Metrics for Total Number of Personal Cars . . . . .	114
37	Prophet Model Results and Performance Metrics for Total Number of Trucks	115
38	Prophet Model Results and Performance Metrics for Total Number of Buses	116
39	Prophet Model Results and Performance Metrics for Total Number of Road Traffic Congestion . . . . .	117



# Chapter 1

## Introduction

"The two most valuable commodities of the 21st century are time and energy; traffic congestion wastes both" (Pan et al., 2012). Traffic is one of the most critical problems of major cities like Montreal. Our goal is to add another layer of information to the traffic data, find which type of vehicles are contributing to road traffic congestion, and predict future road traffic congestion and demands based on the historical data.

We collected more than 400,000 images from traffic cameras installed in Autoroute 40, in the city of Montreal. The images were collected for five consecutive weeks from different locations from April 14, 2019, up until May 18, 2019. To process these images and extract useful information out of them, we created an object detection and classification model using the Faster RCNN algorithm. Our goal was to be able to detect different types of vehicles and see if we have traffic congestion in an image. In order to improve the accuracy and reduce the error rate, we provided multiple examples with different conditions to the model. By introducing blurry, rainy, and low light images to the model, we managed to build a robust model that could do the detection and classification task with excellent accuracy.

Furthermore, by extracting the information from the collected images, we created a dataset of the number of vehicles in each location. After analyzing and visualizing the

data, we find out the most congested areas, the behavior of the traffic flow during the day, peak hours, the contribution of each type of vehicle to the traffic, seasonality of the data, and where we can see each type of vehicle the most.

Finally, we managed to predict the total number of congestion incidents for seven days based on historical data. Besides, we were able to predict the total number of different types of vehicles on the road as well. In order to do this task, we developed multiple Regression, Deep Learning, and Time Series Forecasting models and trained them with our vehicle count dataset. Based on the experimental results, we were able to get the best predictions with the Deep Learning models and succeeded in predicting future road traffic congestion with excellent accuracy.

## **1.1 Research Objective**

We are all familiar with traffic congestion and experience it in our daily commutes. Traffic congestion happens when too many vehicles try to use a transportation infrastructure without having enough capacity (Papageorgiou et al., 2003). However, researches indicate that adding extra lane without studying the future consequences does not improve the situation (Duranton and Turner, 2011).

In order to study the data in a more granular fashion, we decided to process the image data instead of sensor or GPS collected data. By using the image dataset, we will have all the data we need, but we have to find a way to extract information and use them in our favor. In this research, without installing new equipment, we managed to create a valuable image dataset and used state-of-the-art algorithms to convert the data into information.

By adding another layer of information to the traffic data, we will be able to understand the contribution of each type of vehicle to the traffic, analyze the demand, and the behavior of traffic flow in more detail and predict future events with more accuracy. This way, we can prioritize the locations which need more infrastructure and find more insightful information

about the general distribution and the balance of the resources and the demand in the city of Montreal.

## 1.2 Contributions

The contributions of this research are divided into three main sections:

- **Data Collection, Cleaning, and Preprocessing** As we mentioned in previous paragraphs, our goal in this thesis is to predict traffic congestion by using image datasets instead of GPS or sensor-generated data. In the first step, We built an image dataset from the Autoroute 40 in the city of Montreal traffic cameras. We collected more than 400,000 images for five consecutive weeks. After analyzing the data, we developed an image classifier to clean the data and reduce the amount of noise in the image dataset. The main contribution of this step is a clean image dataset from the traffic cameras.
- **Implementing Object detection and Classification Models** The next step after building the image dataset is extracting information out of them. We trained an object detection and classification model using the Faster RCNN algorithm and managed to detect each type of vehicle and traffic congestion in an image. We applied this model to the entire image dataset and created a vehicle count dataset. After analyzing the data, we managed to find the contribution of each type of vehicle (personal car, bus, truck) to the traffic, traffic flow behavior, peak hours, and trends for each type of vehicle. The main contribution of this step alongside the mentioned results is a clean vehicle and congestion count dataset with a time series relationship.
- **Developing Multiple Time Series Forecasting Models** After creating the vehicle count dataset, we can use this dataset to predict future demand and traffic congestion.

In order to do this task, we developed multiple time series forecasting models with different Regression, Deep Learning, and time series forecasting algorithms. We researched to find the best model and how we can get the best prediction results with different architectures. The main contribution of this step is predicting future traffic congestion, the total number of each type of vehicle, and the best model to do this task.

### **1.3 Thesis Structure**

In this thesis, first, we start with definitions about the algorithms and tools we are going to use for this research and get familiar with them. Moreover, we review the researches that are related to the work we are doing in this thesis. Later, we discuss our methodology and how we implement our models. After the methodology, we go over the results that we obtained from each approach and compare them. In the end, we discuss the overall results and the next steps for this research.

# **Chapter 2**

## **Background**

### **2.1 Introduction**

In this section, we discuss the materials we need to develop this thesis. In our research in each step of the works, we use different platforms and different libraries, which we go over them and get familiarized with them. Furthermore, we explore the ways that we can utilize them in our favor to get the best results. First, we explore the deep learning platforms and libraries and discuss the advantages and disadvantages of each of them. Later, we explore the performance metrics and see how we can evaluate our models. After discussing performance metrics, we explore the object detection and classification algorithms and get to know more about their architecture and capabilities. In the end, we discuss the time series forecasting algorithms and explore some of the existing libraries for these kinds of tasks as well.

### **2.2 Road Traffic Congestion**

We are all familiar with traffic congestion and experience it in our daily commutes. It wastes energy and time and has lots of negative effects on the environment. (Bharadwaj

et al., 2017) conducted a field study in order to find the relationship between fuel consumption and the greenhouse gas emission. The results of this study shows that the more travel time under road traffic congestion is significantly responsible for more  $CO_2$  emissions from the fuel consumption. On the other hand it has negative impacts on the public health as well. (Levy et al., 2010) evaluate the public health impacts of being exposed to fine particulate matter concentrations associated with traffic congestion. The analyses shows that the public health impact is significant enough and should be considered when evaluating new policies for alleviating the traffic situation.

Traffic congestion happens when too many vehicles try to use a transportation infrastructure without having enough capacity (Papageorgiou et al., 2003). However, researches indicate that adding extra lane does not necessarily improve the situation. (Duranton and Turner, 2011) investigated the effect of lane kilometers of roads on vehicle-kilometers travel for different types of road in the US cities and provided direct evidence that the extension of most major roads resulted in a proportional increase in traffic.

Based on the TomTom Traffic indexing (TomTom, 2019) congestion level of the city of the Montreal is ranked third in the Canada, tenth in North America and 138th Globally. TomTom collects data from GPS devices based on the actual driven trips. In figure 1 we can see an screenshot of the TomTom website, informing the current traffic situation of the city of Montreal.

## **2.3 Deep Learning Platforms and Libraries**

To create different models from simple Regression models to more complex deep neural network models, we need to use some external libraries and platforms. Our goal in this section is to explore our options and get familiar with the advantages and disadvantages of each one. Later, we see how we can utilize them in our research to get the best results.

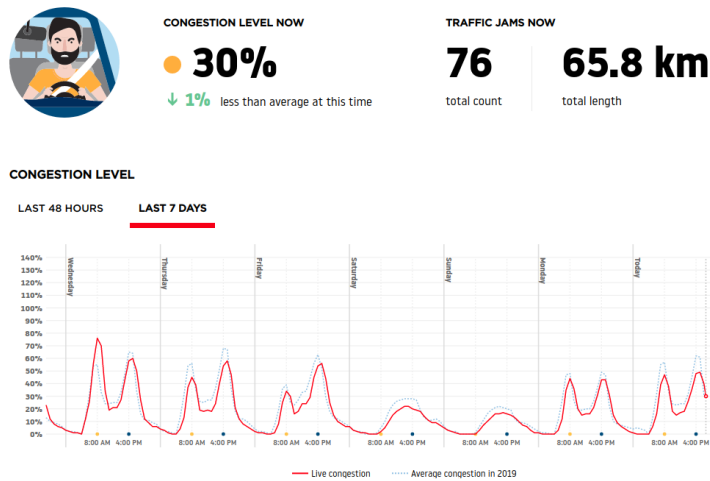


Figure 1: Traffic status of the city of Montreal on 18:36 February 25th, 2020. Courtesy of (TomTom, 2019)

**Tensorflow** Or Tf in short, is an open-source platform for machine learning. It was developed by Google and has a comprehensive set of tools, which makes it one of the best libraries for machine learning studies. Tensorflow computations are exposed as data flow graphs and have a unique tool called Tensorboard, which plots all the necessary graphs during and after the training phase. One of the disadvantages of Tensorflow is the complexity and difficulty of creating a simple model and deploying it. (Tensorflow, 2019).

**Keras** In the previous paragraph, we explained what is Tensorflow and for what purposes we can use it. We mentioned that one of the disadvantages of Tensorflow is the complexity of developing simple neural networks and create a model from them. Keras, which is a high-level neural network API, is the solution to this problem. It was written in Python programming language, and it is capable of running on top of the platforms like Tensorflow.(Keras, 2019).

Keras has different libraries with predefined neural network models and algorithms. We can create a simple neural network by just calling the library and put them into the desired order and architecture. We use Keras mainly for training our deep learning models for time

series traffic forecasting.

**Scikit-Learn** Scikit-Learn or Sklearn is a free machine learning library developed for Python programming language and features many regression, classification, and clustering algorithms. We mainly use the Sklearn for our regression analysis task and evaluating our predictions with the performance metrics available in the library (Scikit-learn, 2019).

**Numpy** When we are dealing with data, most of the time, they are in the form of big matrices or arrays. On the other hand, most of the complicated calculations happen when we are dealing with a large amount of data, so we need a library not only to handle these calculations correctly but also do it in the shortest time possible. Numpy is a library written mostly in the C language and developed for different languages to carry out large-scale and multidimensional mathematical calculations. Most of the libraries dealing with large scale data or massive calculations use Numpy in the back-end. We use Numpy for a variety of tasks from getting mean or median to transform an image data to a three-dimensional array of values to process it (Numpy, 2019).

**Pandas** Python has a long history of being one of the best programming languages for data manipulation and preparation, but not for data analysis and modeling. Python Data Analysis Library or Pandas is an open-source library that provides high-performance data structures and data analysis tools for the Python programming language. As mentions in the previous paragraph, Pandas is one of the libraries that use Numpy for handling all the large-scale calculations and data manipulation. We mainly use Pandas for creating different data frames, data manipulations and performing data transformations (Pandas, 2019).

**Matplotlib** Data visualization can help us get better insights into our data. It is easier to understand the differences and catch the essential information in one glance. Matplotlib is one of the best libraries developed for python language for data visualization purposes.



It offers a variety of 2D and 3D plots with different shapes and lots of configurations. Creating plots and fugues alongside, providing different text information and changing the color map of the plots is not a hard task in Matplotlib. We use this library mainly for visualizing our datasets, creating images with bounding-boxes on them and plotting the predictions next to the target values for time series forecasting models (Matplotlib, 2019).

**Seaborn** Seaborn is a Python library which is built on top of Matplotlib to create more colorful plots. (Seaborn, 2019).

**LabelImg** Our image dataset consists of images with lots of information in them. It would be hard for the model to understand each type of vehicle on its own from these images. To help the model understand what exactly it should look for, we annotate each type of vehicle in the image and create another file with coordinates of the mentioned annotations in it. The new file is in the XML format and has the same name as the image. To do this task, we used a graphical annotation tool called LabelImg. LabelImg is written in Python programming language and uses the QT library for its graphical interface, which makes it easy for us to annotate all the images that we need for the training purposes. In Figure 2 you can see the environment of the LabelImg (Tzutalin, 2015).



Figure 2: LabelImg Platform Environment

## 2.4 Performance Metrics

Regardless of the model and its purpose, we are always curious to see how the model performs on the test dataset. Our goal in this section is to explore different performance metrics that we can use to evaluate our models. Since there are different performance metrics for different tasks, we separate them into two different subsections to explore them based on their architecture and their results. First, we explore how we can measure the performance of a classification task, and later we discuss different prediction performance metrics and how we utilize each of them.

### 2.4.1 Estimation and Prediction Performance Metrics

In the previous section, we discussed the performance metrics that can help us get a better understanding of the performance of our classification models. In this section, we explore the metrics that we can use to know the performance of our regression and time series forecasting models. We what are the advantages and disadvantages of each metrics and see how we can utilize them to understand our models better.

**Mean Squared Error** Mean Squared Error, which also called Mean Squared Deviation, is an estimator measure the average of the squares of the errors. The errors in our case are the distance between the target values and predicted values. Mean Squared error can be calculated as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (1)$$

One thing that we can notice from (1) is that the distance between the target and the predicted value is squared so that the total value can accumulate to a large value.

**Mean Absolute Error** Mean Absolute Error, like MSE, measures the distance between the target and predictions.

$$MAE = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i| \quad (2)$$

**R2 Score** We explored two different ways to measure the distance between the target values and prediction values in previous paragraphs. One of the problems with the previous approaches is that since we are concerned with the distance of two values, we are not considering their scale into the equation. This means that for larger values, we have bigger errors, which does not necessarily represent the performance of the model. To solve this issue and get a better understanding of the model's performance, we can use R squared, which is defined as follows:

$$SS_{tot} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3)$$

(3) represents the total sum of the squares which is proportional to the variance of the data.

$$SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2 \quad (4)$$

(4) is the sum of the squares of the residuals.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \quad (5)$$

(5) is one of the most general definitions of the R2 score or the coefficient of determination. We can see the R2 score can be ranging from minus infinity to one. The bigger the value of R2 score is, the better the model can explain the behavior of the data, or in other words, The closer the R2 score value is to one, the more accurate the model is.

## 2.5 Object Detection and Classification Algorithms

In this section, we discuss object detection and classification methods that we can use to train our vehicle detection model. First, we explore the Convolutional Neural Network (CNN) and its advantages in detail. Later we discuss Faster Region-based Convolutional Neural Network (Faster R-CNN) and explore why we are building our model with this algorithm.

### 2.5.1 Convolutional Neural Networks (CNN)

CNN was inspired based on the research of Hubel and Wiesel on the visual cortex of a cat (Hubel and Wiesel, 1962). They proposed two different complexity for the cell structure of neurons in the visual cortex of the cat. One activates when the cat sees a simple shape. The other one activates when the cat sees a shifted or rotated version of the original object. In 1998, Le Cun, Bottou, Bengio, and Haffner in the handbook of brain theory and neural networks proposed CNNs (LeCun et al., 1995).

Although the idea of CNN is not new, it became popular relatively recently. The main reasons contributing to the recent usage of CNNs are(Koirala et al., 2018):

- The exponential growth in processing power
- Data collection devices like cameras and sensors

In order to be able to utilize a CNN model and get good results, you need to have a huge amount of data and enough computational power to process it. The mentioned factors helped cover these problems.

Moreover, the publication of Krizhevsky's article (Krizhevsky et al., 2012) showed a massive improvement for object detection and image recognition using ImageNet Deng et al. (2009) dataset, which drew a lot of attention from the researchers to the potential of CNN in the image recognition area (Yosinski et al., 2014).

In this thesis, we will discuss on the applications of CNN for object detection and classification task and time series forecasting task as well. Later in this chapter, we describe a specific type of the CNN family, R-CNN, and explain how this type, in particular, helps us develop a vehicle classifier.

## 2.5.2 Region-Based Convolutional Neural Networks (R-CNN)

We discussed how CNN became famous for its capabilities in image processing. In this section, we will discuss an approach that made a huge improvement in this area. Girshick (Girshick et al., 2014) named the approach R-CNN: Regions with CNN features and the reason behind it was the fact that they combined region proposals with CNNs. In figure 3, we can see the architecture of the RCNN in four steps. This approach combined two key insights:

1. Enables applying high-capacity CNN to bottom-up region proposals to localize and segment objects
2. In case of scarce labeled training data, supervised pre-training for auxiliary task followed by domain-specific fine-tuning yields a significant performance boost.

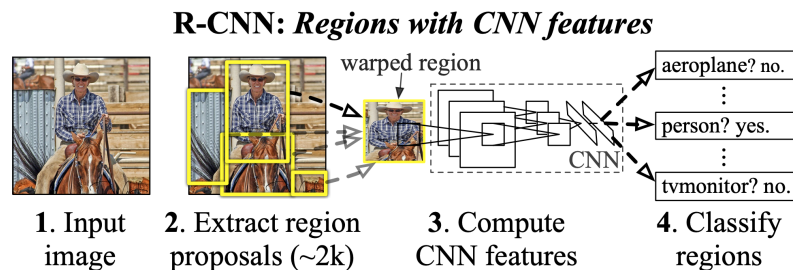


Figure 3: Object detection system overview for R-CNN, Courtesy of (Girshick et al., 2014)

The proposed object detection model consists of three modules. The objective of the first module is to generate category independent region proposals. These proposals are

regions that are fed to the next module as the areas that are most likely to find an object. The next module is a CNN that looks into each of the proposed regions to extract a fixed-length feature vector. The third module is a set of class-specific linear Support Vector Machines (SVMs). R-CNN also managed to achieve a mean average precision(mAP) of 53.7% on PASCAL VOS 2010. (Girshick et al., 2014) Although the accuracy of R-CNN is good, it has some major drawbacks, which makes it challenging to use. Some of these drawbacks are (Girshick, 2015):

1. Multi-stage pipeline for training the model
2. Training the model requires lots of space and time
3. Applying the model on images for object detection is slow

In the next paragraph we will explore a way to improve speed and accuracy of the R-CNN model.

**Fast R-CNN** In the previous paragraph, we discussed how R-CNN works and what are the advantages and disadvantages of this algorithm. One of the significant drawbacks of R-CNN is the speed for the training of using the model for object detection. The main reason that the R-CNN is slow is due to the fact that it operates a ConvNet forward pass for each of the object proposals without sharing the computations(Girshick, 2015). The goal of Fast R-CNN is to fix the mentioned problems with the R-CNN. In figure 5, we can see the architecture of the new approach. An input image and the multiple regions of interest are input into a fully convolutional neural network. There is a fixed-sized feature map for each region of interest (ROI). Each ROI is pooled into the mentioned feature map and then mapped into a feature vector by fully connected layers. For each ROI, there are two output vectors in the network:

- Softmax probabilities

- Per-class bounding box regression offsets.

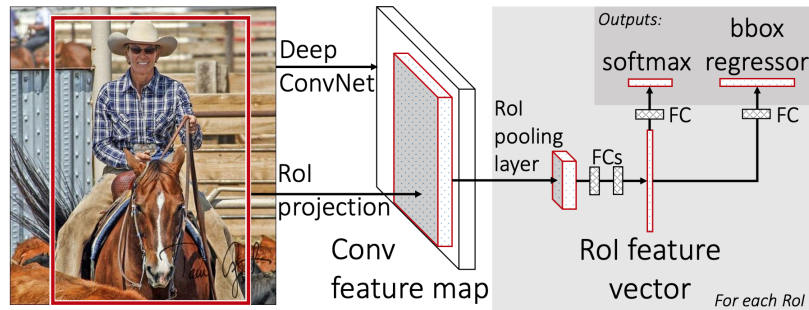


Figure 4: Fast R-CNN architecture, Courtesy of (Girshick, 2015)

The advantages of Fast RCNN are (Girshick, 2015):

1. The Higher detection quality(mAP) than previous methods
2. Training is no longer a multi-stage and using a multi-task loss
3. Training can update all network layers
4. No disk storage is required for feature caching

**Faster R-CNN** In the previous paragraphs, we explored the state-of-art approaches for object detection and classification. First, we discussed the advantages and disadvantages of R-CNN and how Fast R-CNN answered those issues. In this paragraph, we explore another approach that tried to improve these models from both speed and accuracy perspective. The Faster R-CNN approach tries to reduce the training and testing time of the model by improving the region proposal computations. The new region proposal network (RPN) shares full-image convolutional features with the detection network resulting in cost-free region proposals(Ren et al., 2015). The architecture of the new RPN can be seen in the figure 5.

The results in terms of both speed and accuracy for the Faster R-CNN object detection system was 73.2% mAP on PASCAL VOC 2007 and 70.4% mAP on PASCAL VOC

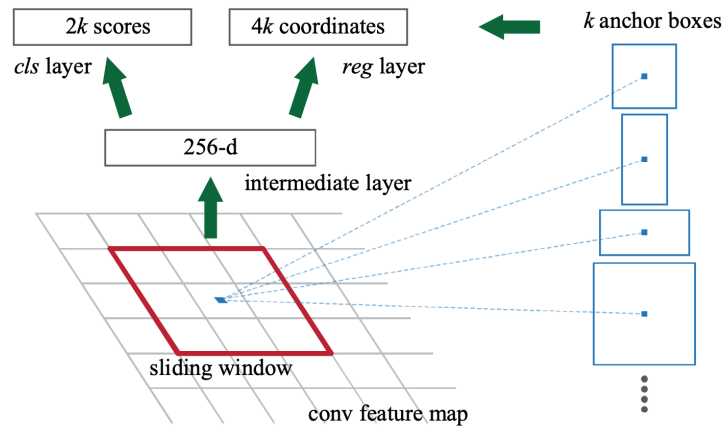


Figure 5: Region Proposal Network (RPN) architecture, Courtesy of (Ren et al., 2015)

2012(Ren et al., 2015). Since we have a limited amount of computational resources, we need a model to do the calculations fast while not sacrificing the accuracy, which with Faster R-CNN, we can achieve these goals.

## 2.6 Time Series Forecasting Algorithms and Models

In this section, we discuss different time series forecasting algorithms and explore the mathematical idea behind each of them. First, we talk about different regression analysis algorithms, explore their differences and similarities, and discuss the advantages and disadvantages of each of them based on their capabilities. Later, we explore different neural network models and try to understand how each architecture of these networks work and what are the ways to find the fine-tuned models. In the end, we explore two different models that are mainly developed for different time series forecasting tasks and talk about how they work and what could be achieved using them.



### 2.6.1 Regression Analysis

There are many different types of regression available to study, but in this thesis, we mainly focused on three different regression analysis algorithms. First, we start with the simplest form of regression, which is Linear Regression, and then we explore the Polynomial Regression. At last, we see what Random Forest Regression is and how we can improve our predictions with it.

**Linear Regression** Linear regression is the simplest form of regression analysis, which studies the linear relationship between two variables. The linear regression equation (6) is a straight line that defines the effect of independent variable  $X$  on the dependent variable  $Y$ .

$$Y = aX + b \tag{6}$$

In (6),  $a$  is the slope of the line, and  $b$  is the  $y$ -intersect. The regression line lets one to predict the value of dependent variable  $Y$  from the independent variable  $X$  (Schneider et al., 2010). To mention a few examples, we would be able to predict a person's salary from his or her years of experience. In our case, we can use linear regression to predict future traffic situation to predict different types of vehicle contributing to the traffic based on the traffic situation.

**Polynomial Regression** In the previous paragraph, we discussed the linear regression and how the dependent and independent variables could be related. However, the relationship between these variables is not always linear. Polynomial regression studies the relationship of variables which are related to each other with higher degrees. The polynomial function of order  $k$  consists of using the explanatory variable  $X$  in different powers, including the power of  $k$  in the regression equation.

$$Y = a_0X + a_1X^2 + \dots + a_kX^k + b \quad (7)$$

Adding an order to the equation adds a segment with different slop signs to the curve representing the fitted values (Polynomial-Regression, 2019)(LEGENDRE, 1998).

**Random Forest Regression** The idea of the Random Forest was proposed by Breiman (Breiman, 2001), which added another layer of randomness to bagging. Unlike standard trees that each node splits using the best split between all variables, in the Random Forest, each tree splits based on the best subset of predictors, which are randomly selected at the given node. In comparison to other approaches like SVMs and even neural networks, this idea proved to be performing better and being robust against over fitting (Breiman, 2001)(Liaw et al., 2002).

The random forest regression algorithm can be summarized in three steps (Liaw et al., 2002):

1. Draw  $n_{\text{tree}}$  bootstrap samples from the original dataset
2. For each of the bootstrap samples, at each node randomly sample  $m_{\text{try}}$  of the predictors and chose the best split from those variables.
3. For regression task, the predictions are calculated by getting the average of the predictions of the  $n_{\text{tree}}$  trees

Later we explain that how we used Sklearn to create the random forest regression model and find the best number of predictors with a simple search.

## 2.6.2 Deep Learning Methods

In the previous section, we explored different types of regression models and explored the mathematical ideas behind them. In this section, we discuss more complex approaches and

dive into the idea of neural networks and deep learning methodology. Just like regression analysis, there are lots of different types of neural networks that are suitable for the time series forecasting tasks. First, we explore the simplest form of neural networks, which is Artificial Neural Networks (ANN). Next, we reuse the CNN with a different design for a times series forecasting task. Later we explore another type of neural network, which is the Recurrent Neural Networks (RNN), and see how they learn better by perceiving some of the information in each neuron. In the end, we discuss another type of RNNs, which is Long Short Term Memory (LSTM), and see how it can fix the issues of not recalling the far past information that we had in RNN.

**Artificial Neural Networks (ANN)** Artificial neural networks are known as computing systems inspired by biological neural networks that constitute animal brains. The neural network is a framework for different machine learning algorithms to process complex data inputs. These data inputs can be images, signals, voice recordings. Just like a human brain, you can train such a system to perform tasks not by coding explicitly but rather through considering similar examples.

The most basic structural and functional unit of a neural network is a neuron, which is illustrated in figure 6. These neurons are generally stacked together in the layered structure connected via weights, as shown in figure 7. These systems are self-learning and trained through backward or forward propagation. Neural networks are the fundamental blocks on which deep architectures are built.

Each cell can have multiple inputs and multiple outputs. To produce the outputs of the neuron, input signals are multiplied by their weight and then combined together and passed to the transfer function. The most commonly used transfer function is sigmoid function (Agatonovic-Kustrin and Beresford, 2000), which produces a value between 0 and 1.

ANNs without cycles are called feedforward neural networks (FNNs). The most widely

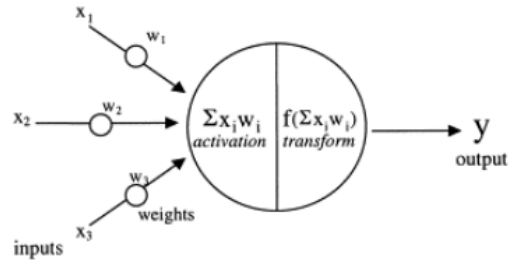


Figure 6: Model of an Artificial Neuron. Courtesy of (Agatonovic-Kustrin and Beresford, 2000)

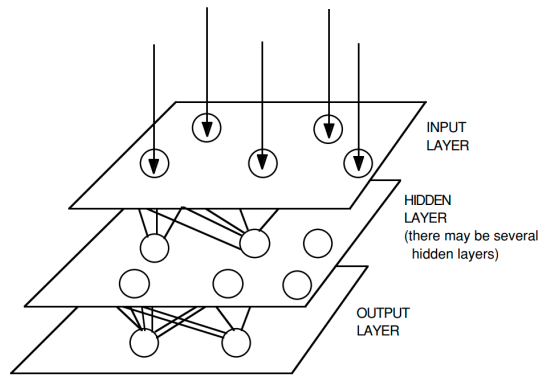


Figure 7: Model of an Artificial Neural Network. Courtesy of (Anderson and McNeill, 1992)

used form of FNN is the multi-layer perception (MLP) (Rumelhart et al., 1985)(Bishop et al., 1995).

**Recurrent Neural Network (RNN)** In the previous paragraph, we discussed ANNs in which their connections did not form cycles. By adding cycles to the neural network, we can create an RNN. Just like ANNs, there are many types of RNNs proposed by many researchers as well (Kawakami, 2008).

As discussed in the previous paragraph, an MLP only maps from input to output vectors, but RNNs can map from the entire history of previous input vectors to each of the output vectors. This helps the neural network to persist some of the information from previous inputs in the network’s internal state and affects the output. Figure 8 shows a Recurrent Neural Network alongside the unfolding version where the information from previous steps is conveyed to the next steps (Tai and Liu, 2016).

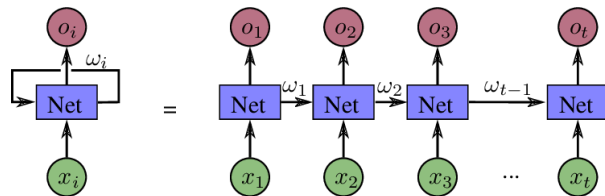


Figure 8: On the left side we can see a typical RNN and on the right side is the unfolding version. Courtesy of (Tai and Liu, 2016).

In this thesis, we are interested in one of the variations of RNNs, Long Short Term Memory(LSTM), which solves a fundamental issue with this type of neural network, and that is not being able to remember key information from far past.

**Long Short Term Memory(LSTM)** As mentioned before, Recurrent neural networks can use their feedback connections to store representations of recent input events in the form of activation. Although the idea is fascinating, there is a problem when it comes to

practical use. The problem, as (Hochreiter and Schmidhuber, 1997) explained, is the error signals flowing backward in time tend to either first, blow up, or second vanish. Basically, what it means is that the RNN has a problem with remembering the information from the very past. (Hochreiter and Schmidhuber, 1997) represents the idea of LSTM as a novel recurrent network architecture. In Figure 9 we can see the architecture of an LSTM cell.

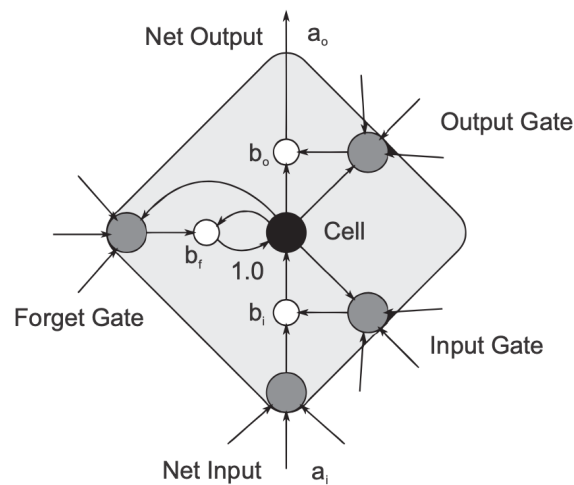


Figure 9: LSTM Memory Cell with Gates. Courtesy of (Soutner and Müller, 2013).

### 2.6.3 Time series Forecasting Models

In the previous sections, we explored different algorithms from regression analysis and deep learning models. In this section, we discuss two well-known time series forecasting platforms. The first platform is ARIMA, which has been around for a few decades, and there are lots of research utilizing ARIMA to implement a time series forecasting model. The second platform, Prophet, which is recently developed by the Facebook company, has the same goal of automating time series forecasting tasks.

**ARIMA** Almost everyone who is dealing with time series forecasting had heard the name of ARIMA. ARIMA is one of the widely used time series forecasting models and stands for

Auto-regressive integrated moving average and created by Box and Jenkins in 1970(Box et al., 2015). ARIMA models can represent several different types of time series tasks like Auto-regressive, Moving average, and the combination of these two. One of the significant downsides of this library is the pre-assumed linear form of the model, which means that ARIMA is only able to capture the pattern of the data if there is a linear correlation structure that exists between the time series values(Zhang, 2003).

ARIMA model has three components, which are the pure autoregressive model (p), the moving average model (q), and the number of differences needed to make the time series model stationary (d). This model is usually organized as the following formation(Chen et al., 2008):

$$\nabla^d = (1 - B)^d \quad (8)$$

$$\Phi(B) = 1 - \phi_1 B - \dots - \phi_p B^p \quad (9)$$

$$\Theta(B) = 1 - \theta_1 B - \dots - \theta_q B^q \quad (10)$$

where p, d and q are all integers.

$$\Phi(B)\nabla^d x_t = \Theta(B)\varepsilon_t \quad (11)$$

$$E(\varepsilon_t) = 0, Var(\varepsilon_t) = \sigma_\varepsilon^2, E(\varepsilon_t \varepsilon_s) = 0, s \neq t \quad (12)$$

$$Ex_s \varepsilon_t = 0, \forall s < t \quad (13)$$

The  $\varepsilon_t$  is the estimated residual at each point and  $\sigma_\varepsilon^2$  is the variance of residuals.

The formulation of the ARIMA model can be summarized in the following four steps(Tseng

and Tzeng, 2002)(Chen et al., 2008):

1. Identifying the components of the ARIMA model which are the  $p$ ,  $q$ , and  $d$
2. Coefficient estimation
3. Fitting test on the estimated residuals
4. create future forecasted values based on the historical data

**Prophet** Like ARIMA, Facebook Prophet is a time series forecasting model that uses the analyst-in-the-loop approach. This approach helps analysts to use their knowledge to alter the model and add some coefficient to it without having any understanding of the underlying statistics(Taylor and Letham, 2018). Figure 10 illustrate the model approach.

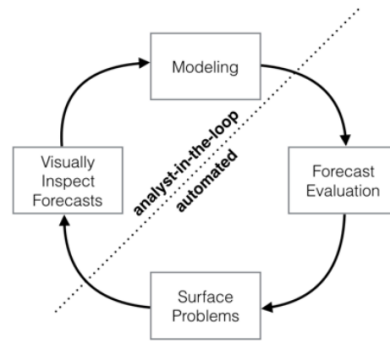


Figure 10: Analyst in the loop model. Courtesy of (Taylor and Letham, 2018)

To alter the model for achieving better results, there are for coefficients that can be identified to the model:

- Capacities: To specify the total market size
- Changepoints: Known dates of change points
- Holidays and seasonality: If a holiday has an impact on the data and the predictions it can be specified beforehand



- Smoothing parameters: The analyst can directly change the global or local effect of holiday magnitude and tell the model how much variation is expected.

The Facebook Prophet model can be decomposed into three components: trend, seasonality, and holidays and they are combined in the following equation(Taylor and Letham, 2018):

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \quad (14)$$

In (14), three different things are affecting the final results, which are the periodic  $s(t)$  and non-periodic  $g(t)$  trends plus the holidays  $h(t)$  which occur on irregular schedule over one or more days. The error term  $\varepsilon_t$  represents the changes which model could not capture.

## 2.7 Conclusion

Before we started surveying other scientific research and implementing our methodology, we needed to familiarize ourselves with some basic definitions, platforms, algorithms, and libraries. In this section, we started the discussion with the introduction of several deep learning libraries and explained how each one of them could help us to achieve our goals. Later we reviewed different performance metrics and explored their differences. Then we discussed object detection and classification algorithms and explored the very idea of computer vision and how we can utilize it for our research. At last, we explored time series forecasting algorithms and models, which can help us predict future trends of traffic based on the historical data.

# Chapter 3

## Literature Review

### 3.1 Introduction

In the previous section, we reviewed some definitions, platforms, algorithms, and libraries. Our goal in this section is to explore other researchers' approaches and ideas to see how they are utilizing these tools to conduct research and see their results. First, we explore researches related to object detection and classification and divide the papers into two different classes. In the first class, we explore application-focused researches in which the researcher is trying to apply the same methodologies on different problems and manages to achieve better results. In the second class, we explore researches in which the researcher managed to develop a new algorithm or improved the existing algorithms. Later we explore the time series forecasting algorithms and approaches as well and divide them into two different classes. In the first class, we explore researches which are directly applied to the traffic data, and the goal is to predict future traffic situation. In the second class, we explore researches that used time series forecasting methods to predict events, stock market, price of a good or commodity, and future demands.

## 3.2 Object Detection and Classification

### 3.2.1 Application Focused Researches

The goal of (Cai et al., 2018) is to build a family monitoring alarm system based on deep learning methods to distinguish the types of invaders by training a model and focusing on the different characteristics of multiple species. They managed to build an intelligent monitoring system based on the YOLO network with high detection precision and speed. Furthermore, the author claims that the false detection rate is zero in the background of home remote monitoring.

(Jiang and Learned-Miller, 2017) demonstrated state-of-art face detection results using Faster R-CNN on three different datasets. The authors suggest that the face detector model performs well even when trained on one dataset and tested on another. The experimental results of this research show that the effectiveness of Faster R-CNN comes from the region Proposal Network Module(RPN).

Pedestrian safety is one of the research areas which is receiving increasing attention in academia. (Zhao et al., 2016) presents a Faster R-CNN based pedestrian detection system to detect a specific class of objects. In this research, a VGGNet (Simonyan and Zisserman, 2014) model was tuned and trained. The author suggests that the proposed system can process images of arbitrary size and outputs their bounding box coordinates alongside the pedestrian scores.

One of the problems that computer vision is seeking to find a solution is anomaly detection. (Mahadevan et al., 2010) proposed a novel framework for anomaly detection in crowded scenes. The authors identified three properties as important factors to design a localized video representation suitable for anomaly detection in such scenes. The authors proposed a different approach for each of the properties and suggested that the results are showing the proposed model is outperforming the various state of the art anomaly detection

techniques.

### **3.2.2 Model Improvement Oriented Researches**

(Fan et al., 2016) aims to conduct multiple experiments using Faster R-CNN on the KITTI (Geiger et al., 2013) dataset to study the object proposing, localization vs. recognition and iterative training. The goal of this research is having a deeper understanding of how to tune and modify Faster R-CNN for specific applications and datasets and improve the default performance of the Faster R-CNN for vehicle detection on the KITTI dataset.

To improve the accuracy of vehicle detection (Gong et al.) proposed a multitask Cascade Convolution Neural Network(MC-NN) based on Full Frame Histogram Equalization(FFHE) algorithm. The authors proposed an object detection algorithm based on Faster R-CNN, which removes the shadow of complex background and outputs every object location in the image. Moreover, they proposed an object classification algorithm based on CNN, which outputs the characteristics of the vehicle. The authors compared the proposed algorithm's accuracy with the Faster R-CNN without FFHE and suggested that the accuracy of the proposed algorithm has been greatly improved.

(Hsu et al., 2018) proposed a simplified fast region-based convolutional neural network (R-CNN) for vehicle detection. The proposed detection system consists of several convolutions and max-pooling layer, followed by a region-of-interest pooling layer and connected to a sequence of fully-connected layers. The fully-connected layers branches into two and produce the probability of the existence of a vehicle and the and four real values numbers for the bounding box enclosing the vehicle. The authors show that the proposed method can detect and localize vehicles in various views effectively.

(Dong et al., 2015) proposed a semi-supervised convolutional neural network that was applied to the vehicle frontal-view images. The authors introduced a sparse Laplacian filter learning to obtain the filters of the network with large amounts of unlabeled data. A

Softmax classifier is trained by multi-task learning with a small amount of labeled data. The model can provide the probability of each type of vehicle and the class it belongs to. The results on the dataset which the authors created show the effectiveness of the proposed method.

(Byeon and Kwak, 2017) presents a performance comparison between Faster RCNN and ACF to do the pedestrian detection task. The results of this research indicated that the precision of the Faster RCNN model is 56.73% higher than the precision of ACF on the manual work. Moreover, the authors applied an additional method, HOG-SVM, but the faster RCNN showed that its precision is much higher than other approaches.

## **3.3 Time Series Forecasting**

### **3.3.1 Road Traffic Prediction Related Researches**

(Min and Wynter, 2011) developed a method for traffic prediction at a fine granularity and over multiple periods. The method takes into account the spatial characteristics of a road network, which reflects the distance and the average speeds of the links. The authors suggest that the accuracy they managed to achieve with the developed model exceeds that of other published works on 15-min data and can achieve very good accuracy on the more volatile 5-min data.

In (Yasdi, 1999) The authors analyzed the potential of neural networks for the prediction of road traffic and developed a neural network to predict the future values of the traffic time series using its past values. The results of the proposed method show that the MSE is less than 0.003, and the results are better than the compared methods. The model improved the forecasting by 20% for the road traffic flow.

Previous traffic measure parameters required to predict both long-range and short-range dependencies. (Shu et al., 1999) provides a procedure to model and predict traffic using

FARIMA models. The results illustrated that the FARIMA model is able to capture the property of the actual traffic. The authors provided guidelines to simplify the FARIMA model fitting procedure and reduce the time of the traffic modeling.

In (Xu et al., 2017), a real-time road traffic state prediction based on the auto-regressive integrated moving average (ARIMA) and the Kalman Filter is proposed. The ARIMA model is a time series built on the basis of the historical road traffic data and combined with the Kalman Filter to construct the road traffic state prediction algorithm. The experimental results of this research show that the proposed model can predict the real-time road traffic state prediction with high accuracy.

### **3.3.2 Non Road Traffic Prediction Related Researches**

In (Siame-Namini et al., 2018) compares the accuracy of two models, ARIMA and LSTM, as the primary techniques when forecasting time series data. The authors implemented and applied a model with each approach and compared the results on a set of financial datasets. The results of this research indicate that the LSTM achieved better results than ARIMA. Moreover, the LSTM algorithm improved the results by 85% on average in comparison to the ARIMA model.

One of the most interesting applications of the time series forecasting is predicting the future of the stock price. (Pai and Lin, 2005) proposed a hybrid methodology that exploits the unique strength of the ARIMA model and the SVMs model in forecasting the stock price problem. In this research, real data of the stock process were used to examine the forecasting accuracy of the proposed model. The authors suggest that the obtained results with the proposed model are very promising.

The electric power demand forecast is really important to the electric industry. To forecast the monthly electric power demand per hour in Spain (Garcia-Ascanio and Maté,

2010) presents a comparison between two different models. The first model is vector autoregressive(VAR) forecasting applied to interval time series(ITS), and the second model is the multi-layer perceptron model adapted to interval data. The authors suggest that the ITS forecasting method can be used as a tool to reduce the risk when making power system planning and operational decisions.

In (Palomares-Salas et al., 2009), an ARIMA model and a backpropagation type neural network are developed in order to forecast the wind speed. The data for the experiment were acquired from a unit located in Southern Andalusia, with a soft orography(10 minutes between measurements). The results of this research indicate that the ARIMA model is better than NNT for short-time-interval to forecast.

### **3.4 Conclusion**

In this chapter, we reviewed other researches in both areas of object detection/classification and time series forecasting. In the first section, we saw that most of the researchers got good results when applying the Faster R-CNN model on their dataset to detect or classify different objects in a set of images. Moreover, we reviewed road traffic-related researches and got familiar with the most powerful tools and algorithms in order to get better results when dealing with this kind of task.

# Chapter 4

## Methodology

### 4.1 Introduction

In the previous section, we discussed the related works that have been done in the area of object detection classification and time series forecasting. First, We are familiarized with the top-notch algorithms and best practices to detect and classify objects in images and found out that the Faster R-CNN object detection model is doing a better job than other algorithms. Moreover, we reviewed different time series forecasting models and how other researchers used them on their data to get better prediction results. Our goal in this chapter is to train an object detection model with our collected data and create a numerical data set for traffic time series forecasting. Later, we apply different algorithms and develop different models to achieve the best traffic prediction results.

### 4.2 High-level Model Description

In previous sections, we explained the general idea of how we will conduct the research. In this section, we explore how we get to the predictions by collecting the image dataset with more details. In figure 11, you can see each step and how they are connected. The results



and output of one step will be used as the input for the next steps.



Figure 11: The Methodology Steps. From left to Right: (1) Getting Data from Traffic Cameras, (2) Using Computer Vision to Extract Information, (3) Creating the Historical Dataset of the Vehicle and Congestion Count, (4) Develop Several Time Series Forecasting Models, (5) Visualize the Predictions, Compare the Results and Find the Best Model

### 1. Data Collection, Cleaning, and Preprocessing

As we mentioned in previous paragraphs, our goal in this thesis is to predict traffic congestion by using image datasets instead of GPS or sensor-generated data. In the first step, We built an image dataset from the Autoroute 40 in the city of Montreal traffic cameras. We collected more than 400,000 images for five consecutive weeks. After analyzing the data, we developed an image classifier to clean the data and reduce the amount of noise in the image dataset.

### 2. Implementing Object detection and Classification Models

The next step after building the image dataset is extracting information out of them. We trained an object detection and classification model using the Faster RCNN algorithm and managed to detect each type of vehicle and traffic congestion in an image.

### 3. Creating and Analyzing Vehicle Count Dataset

We applied The object detection and classification model to the entire image dataset and created a vehicle count dataset. After analyzing the data, we managed to find the contribution of each type of vehicle to the traffic, traffic flow behavior, peak hours, and trends for each type of vehicle.

### 4. Developing Multiple Time Series Forecasting Models

After creating the vehicle count dataset, we can use this dataset to predict future

demand and traffic congestion. In order to do this task, we developed multiple time series forecasting models with different Regression, Deep Learning, and time series forecasting algorithms.

## 5. Predictions and the Best Models

Finally, We research to find the best model and how we can get the best prediction results with different architectures. We use the best model for each metric to do the predictions.

## 4.3 Data Acquisition

For making any predictions, we need to have historical data. The data that we are going to use should be without any noise and distortion to have the best prediction results. In this section, we explain how we collected the data from different sources and what was our approaches to ensure the data is flawless and without any issues.

### 4.3.1 Data Collection

Montreal is the largest city in Canada's province of Quebec. It is set on an island in the Saint Lawrence River and named after Mt. Royal, the triple-peaked hill at its heart. We are able to collect images of the road from the different locations during different hours of the day from traffic cameras installed all over the city and province by the city of Montreal, The Société des traversiers du Québec. Quebec511 is an information service that makes it possible for road users to find the information needed about the Quebec road network (Quebec511, 2019). In Figure 12 we can see the map of the city of Montreal and the traffic cameras that are installed in the major auto routes.

In this study, we focus on Autoroute 40, which Also known as Autoroute Felix-Leclerc outside of Montreal and Metropolitan Autoroute within the city of Montreal. Autoroute 40

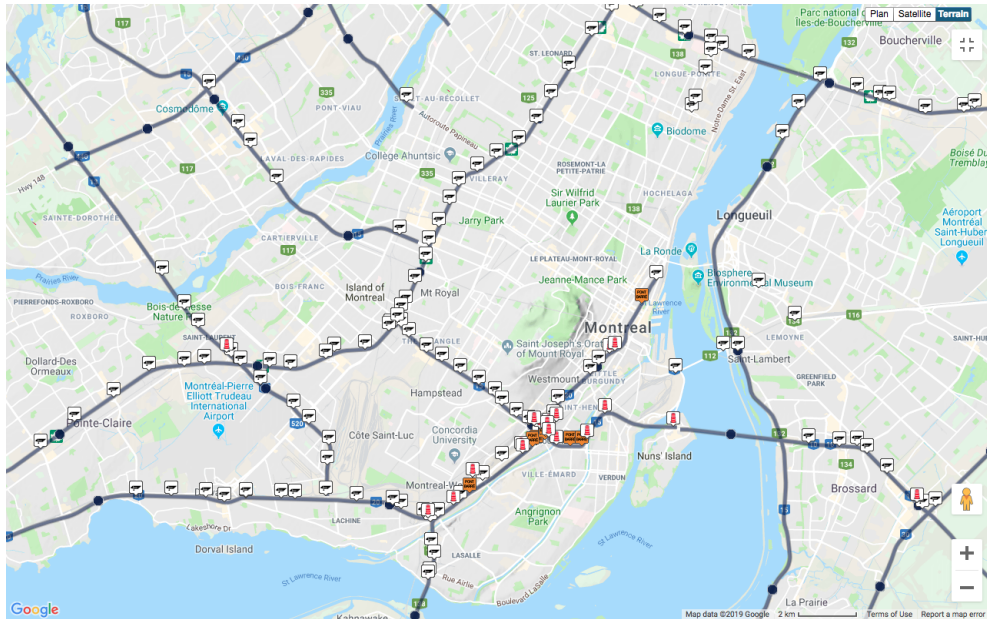


Figure 12: Traffic cameras provided by Quebec511 in city of Montreal. Courtesy of (Google-Maps, 2019)(Quebec511, 2019)

is a freeway on the north shore of the St. Lawrence River in the province of Quebec, and currently, it is 347 km long (Google-Maps, 2019). In Figure 13 Autoroute 40 is marked with red color.

For collecting data from the Quebec511 website, we developed a web crawler to download images and information related to each road from the website. The entire program is in Python programming language, and it saves the images, date and time, road name, and direction in which the camera is looking alongside the image file.

In figure 14, you can see some of the images which are collected from traffic cameras from different locations. There are around 42 traffic cameras in Autoroute 40, and the majority of them are functional 24/7. There is periodic maintenance, which makes the camera unavailable, and a message is shown instead of the footage. We collected images from these cameras every five minutes for five consecutive weeks and aggregated more than 400,000 images during this period. Later we divided the images into separate folders based on the date of the collected image to have a better-organized dataset.



Figure 13: Auto route 40. Courtesy of (Google-Maps, 2019)



Figure 14: Images Collected from Traffic Cameras. Courtesy of (Quebec511, 2019)

Table 1 is the list of all the information we collected from the Quebec 511 website alongside all the images.

Data	Description
Image	The footage from the traffic camera
Time	The exact time of the current footage
Date	The date of the current footage
Address	The location of the camera
Direction	The direction which the camera is looking at

Table 1: Collected Information from Quebec 511 website

## 4.4 Object Detection and Classification

In the previous section, we presented how we collected the images and mentioned that we collected more than 400,000 images for five weeks. It is a really time consuming and frustrating task for humans to count all the different vehicles in an image and repeat this task for the entire data set. In this section, First, we try to make sure that the data that we are dealing with is flawless and without any problem. Later, we explore the ways that we can automate the feature extraction task by applying different practices and algorithms to create a dataset containing the information extracted from the images.

### 4.4.1 Data Preparation

We collected images every five minutes, and more than 400,000 images were collected for five weeks. Since the environment that we are dealing with is a non-deterministic environment, we should expect different kinds of footage while we are collecting the data. However, most of the time, the images are without any noise or distortion. If the camera is under maintenance, no footage is shown, and instead, a message appears. Figure 15 indicates that there is a problem with the camera, and no footage is available.



Cette image n'est pas  
disponible en ce moment.

Nous nous excusons  
de cet inconvénient.

Figure 15: Traffic Camera Footage is Unavailable(No-Footage). Courtesy of (Quebec511, 2019)

The table 2 is the list of expected footage from the traffic cameras.

**Data Cleaning** As mentioned in the previous paragraph during the data collection phase, we may end up introducing some noise to the dataset due to unexpected events. Our goal in this section is to find the reason behind it and the occurring pattern and a way to clean the dataset from the noise.

The first step to clean the dataset is to find a way to clean the image dataset from the no-footage images, which can be seen in figure 15. Following approaches can help us remove this type of noise from the image dataset:

- Find which camera has a scheduled maintenance time and remove the downloaded images from that specific window of time.
- With a simple image subtraction method, we can subtract the downloaded image from figure 15 and delete the image if they were matched. We can use the OpenCV Python library to do this task.

Although for the no-footage images, we can have systematic approaches, for the distorted images, we can not find a simple approach like what we mentioned above. This type of noise happens without any scheduled time and has a random pattern. For these kinds






Image	Condition
 <p>13/04/2019 14:04</p>	Normal day camera footage
 <p>13/04/2019 22:08</p>	Normal night camera footage
 <p>19/04/2019 11:55</p>	Rainy day camera footage
 <p>19/04/2019 20:25</p>	Rainy night camera footage
 <p>13/04/2019 14:02</p>	Distorted image

Table 2: Table of all the footages we expect from traffic cameras

of images, we need to come up with a more sophisticated way to remove them from the dataset. The best way to do this task is to create an image classifier and train it with distorted and not distorted images. We selected a pre-trained model and trained it on our own dataset to detect the classify the distorted and not distorted images.

## 4.4.2 Supervised pre-training

**Image Labeling** When we train any machine learning model, we never explicitly code or tell our model what exactly to do in each situation. We want the model to learn on itself with the examples we provide. To teach our model to find different types of vehicles, we need to provide different examples of each vehicle in different situations. In each image, we may have lots of objects, and it makes it hard to tell the model to look for a specific object, so we need to find a way to annotate the object within the image and give them a specific label. As explained in the definition chapter, LabelImg is a tool that we can use to mark each object in an image and create an XML file with all the coordinates of the marked objects. Later we use the XML files alongside the images to feed them to the model for training and testing. Figure 16 is an example of an image in which all the different vehicles are marked in it.

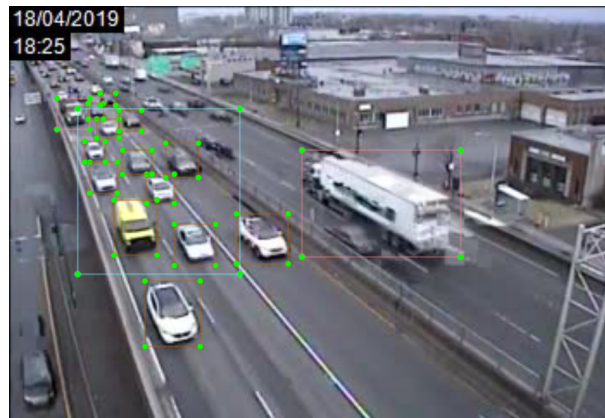


Figure 16: An Example of an Image with All the Object Marked in it Using LabelImg

**Train and Test sets** As mentioned before, we need to create a dataset for training and a dataset for testing our model. We used around 1000 images with different weather situations and from different locations on the island of Montreal. Table 3 is the weather situation according to NOAA(National Centers For Environmental Information) (NOAA, 2019).



Month	High/Low(°C)	Rainy/Snowy Days
January	-4° / -12°	11 days
February	-3° / -11°	10 days
March	2° / -6°	11 days
April	9° / 1°	10 days
May	17° / 8°	10 days
June	24° / 15°	10 days
July	26° / 18°	10 days
August	26° / 17°	9 days
September	21° / 13°	9 days
October	13° / 6°	10 days
November	6° / 0°	10 days
December	-1° / -8°	13 days

Table 3: Weather Situation in City of Montreal During the Year (NOAA, 2019).

As the table suggests, We have almost 10 days of rain each month, which roughly one-third of each month. This fact proves that one-third of our image dataset is collected in the rain, and if we do not address this issue, we lose a large amount of data. On the other hand, again, almost one-third of the data which is 8 out of 24 hours of data were collected throughout the night and if our model does not work in the low light setting we will lose another huge chunk of our data again. For addressing this issue, our model should be able to detect and classify objects in the night, in the rain, and both. To make our model robust, we collected data during the night while it was raining during the day and during the night to cover all the situations and labeled our data. We passed the labeled dataset alongside the data that we collected from sunny days to train the model and created a train set and test set for this purpose.

#### 4.4.3 Faster R-CNN Model

We previously discussed the Faster R-CNN algorithm and how it works. Furthermore, we reviewed some of the works that compared the Faster R-CNN algorithm with the other state

of the art algorithms and proved that, as of now, the Faster R-CNN method is one of the most accurate algorithms for object detection and classification. Tensorflow provided an object detection API with different models like MobileNet (Howard et al., 2017) versions one and two, different types of ResNet (He et al., 2016), and Faster R-CNN. We use a pre-trained Faster R-CNN model and use our annotated data to train it with our dataset.

**Training** In the previous section, we created the training and testing dataset of images and the coordinate of the location of objects as XML files. Since we are using the pre-trained Tensorflow object detection API, we need to create a TFrecord to pass alongside our data to the model to train it. TFrecord, as explained in the Tensorflow documentation, is a file that contains a sequence of records that can only be read sequentially. Tensorflow provides a method to apply a function to each element of the dataset, and the function must operate in Tensorflow graph mode(Tensorflow, 2019). To prepare all the requirements, we need to do the following steps:

1. **Create the CSV file from the coordinate XML files** We labeled all the images and created the XML file of coordinates for each of them. We need to create a CSV file of all the coordinates to create the TFrecord. We utilize the OS library to open each XML file, and Pandas library functions to create a data frame of coordinates and save it as a CSV file. We need to create one CSV file for the train set and one CSV file for the test set.
2. **Create the label map class** We are trying to detect multiple types of objects in images. Since we labeled them with names, we need to create a label map to map the categorical labels to integers to identify each object by a number rather than a name. The objects that we are trying to identify are personal cars, trucks, buses, and traffic congestion.

3. **Generate the TFRecord** After completing the previous steps, we are now able to generate the TFRecord. Tensorflow provides the necessary codes and functions to generate the TFRecord, and we need to provide the right inputs. The inputs are the CSV file and output address to create the TFRecord for both train and test sets.

After creating the TFRecord we need to create a configuration file to pass addresses and pre-defined values to our model. This file should contain the address to train and test TFRecord files and training hyperparameters. Hyperparameters are non-learnable values that can affect the results of our model. Some of these metrics are:

- Batch size
- Epochs
- Loss Function
- Activation Function
- Optimizer
- Initial learning rate
- Decay steps
- Decay rate

In the configuration file, we can also define the fine-tune checkpoints as well. These checkpoints contain everything that model learned by far. The model can continue the training process from each of these checkpoints as well.

## 4.5 Time Series Traffic Forecasting

### 4.5.1 Data Preparation

We collected data for 5 weeks, and we use the first 4 weeks to train the model and last week to see how close the predictions are to the target values. The reason that we are not randomizing the train and test datasets is that we have to make sure the model never sees the data from the last week to make sure the predictions are made without any knowledge of the target. This task can be easily done by using Numpy array functionalities for chunking an array to different sizes. The final result would be four datasets that we use for different purposes.

**Data Cleaning and pre-processing** In this section we are interested to investigate the dataset and see if we have any abnormalities, missing values or scaling issues. The goal is to clean the dataset from the mentioned problems in case we have it. In order to do it we can use the following approaches to clean and pre-process the data:

1. Removing the Null values
2. Removing the outliers
3. One-Hot encode the categorical data
4. Scale the data if the measurement unit of the features are different

**Feature-Based Approach** We have four numerical features in our dataset, which are the number of personal cars, trucks, buses, and congestion. In this approach, our goal is to predict traffic congestion based on the number of vehicles on the road. We use the first 4 weeks of personal cars, trucks, and busses as the matrix of features for training purposes. The first four weeks of the congestion column are used as the vector of the dependent

variable. Finally, the last week of the congestion column is used as the Y-test dataset, and other columns are the matrix of features for X-test dataset. One of the disadvantages of this approach is that we are using the present data from one metric to predict another metric behavior. This means that we do not consider a data point for the future, and we are not able to predict the future. The next approach, Time Step, solves the aforementioned problem.

**Sliding or Rolling Window approach** This approach, as mentioned previously, instead of focusing on the feature, tries to capture the relation of the values regarding the time of the occurrence. We define a window of time, like seven, and then we assign the first day as independent values, and what happens on day eighth would be the result of the previous seven days and the dependent variable. This way, every seven consecutive days of the first four weeks in our data set are part of the X-train dataset, and each eighth day of the mentioned seven days would be in the Y-train dataset. The same methodology applied in other to create the test datasets. This way, we can predict the data points in the future.

## 4.5.2 Regression Analysis

**Linear Regression** Our goal in this section is to train a Linear Regression model with our data and fit the model to our test dataset to evaluate the model and the predictions. We created the Linear Regression model using the Scikit-Learn library which contains all the regression models that we are going to use. As explained before, we have two ways of feeding the data to train the model. First, we explore the feature-based approach.

To create the feature-based datasets for training and testing purposes, we use the "personal\_car", "truck" and "bus" for creating the matrix of independent variables or X and the "congestion" as the vector of dependent variables or y. We keep the last week away from the reach of the model to make sure that model predictions are not affected by the target

values and be sure that the predictions are valid. The Linear Regression function does not need any extra argument as input for the training, and we can call it by using the Sklearn library.

The next approach is based on the time step datasets. Like the previous approach, we use the first four weeks for training purposes and last week for test and evaluation. In this approach, the matrix of independent variables is constructed with a window of each seven consecutive days, and the vector of variables is constructed with the immediate 8Th day of the window. We use the same Scikit-learn library to train the Linear Regression model, and to check the predictions, we plot them next to the target values, which is the last week of our data or the `y_test` dataset. In this approach, we can predict future data points and values.

In the end, to see how the model is performing, we evaluate the predicted result by using our performance metrics and plotting the prediction result alongside the target values.

**Polynomial Regression** For training a Polynomial Regression model, we follow the same path just as we did for Linear Regression but with one difference, the degree of Polynomial. We know that the degree of a Linear Regression is always one, but for Polynomial Regression, we need to find the best degree to fit the model on our data. One of the simplest ways to do it finding the best degree is by plotting the accuracy of the model for each degree and chose the degree with the best accuracy. Like Linear Regression, we have two approaches to train the model based on the different datasets that we created.

Just Like Linear Regression, to see how our Polynomial Regression model is doing, we use our performance metrics to evaluate the model, and also we plot the prediction results alongside the target values to how close the model and predict.

**Random Forest Regression** Unlike Linear Regression and Polynomial Regression, Random Forest is an ensemble learning method that has two sub-classes, Random Forest Classifier and Random Forest Regressor. Since we are dealing with numerical values and our goal is not classification but rather prediction, we use the Random Forest Regressor model. As explained before[ref], Random Forest consists of many decision trees, and the number of trees affects the prediction results. One of the ways to find the best number of trees is to do a simple search.

The Scikit-learn library has a built-in Random Forest Regressor model, which accepts many arguments as input. The most important input for us is the `n_estimators`, which defines the number of decision trees. To be able to create the same results each time with the same data, another input can be given as the random state to make sure the results are not different in each run.

Our approach for training the model and feeding the data to it is exactly like Linear Regression and Polynomial regression. We create two different datasets, feature-based, and time step for different purposes and train the model with them. Later, to see how the model is performing, we use our performance metrics to evaluate the model and the prediction results. Later we plot the predicted results alongside the target values to see how close the model can predict.

### **4.5.3 Deep Learning Methods**

In the previous section, we explored the ways we that we can utilize Regression Analysis to predict either another features behavior or future values of a feature. In this section, we are interested to see how Neural Network models would perform with the same datasets. We explore different algorithms with different structures and try to fine-tune them to get the best results. We mainly use the Keras platform with the Tensorflow back end to create and

train our models. Furthermore, we use other libraries like Numpy for mathematical calculations and Scikit-Learn for performance metrics. At last, we use the Matplotlib library to plot our predictions and our figures.

**Convolutional Neural Network** Previously we used CNN to detect and classify objects in our image dataset. In this section, we explore the CNN again to see how well it can perform on our time series dataset and compare the results and the performance of the model with other models.

Unlike regression models, deep learning models can have different architectures, which ultimately yield different results. Furthermore, we need to find the best hyper-parameters as well. We explained before that hyper-parameters are those parameters that are not learnable but can affect the accuracy of the model. Some of these hyperparameters are learning rate, number of epochs, decay rate and etc. Additionally, we need to set an activation function and a loss function for the model as well. We built our model with Keras platform with a Tensorflow backend. It has CNN in layer for different dimensions, and each layer can have different activation and loss functions as well. With the Help of Keras, we can specify how many layers of CNN and in layer how many neurons we need independently. This functionality helps us to implement and to have a better understanding of the architecture of our CNN. After we designed the model with different layers, we can compile the model and train it with our datasets.

Our goal is to build several convolutional neural networks and compare them with each other and see which model is performing better to predict the traffic situation for the next seven days. To build several models, we have to change the hyperparameters or change the architecture completely. Table 4 is representing four of the mentioned models that we created to predict our metrics with convolutional neural networks. Later in the following chapter, we discuss the results and use the best model to make the predictions.

Aside from the hyper parameters the architecture of the model is really important as



Model	Epochs	Batch size	Optimizer	Layers	Loss Function
CNN 1	100	20	Adam	1 CNN + Max Pooling + 1 Dense	MSE
CNN 2	128	10	Adam	2 CNN + Max Pooling + 2 Dense	MSE
CNN 3	50	7	Nadam	3 CNN + Max Pooling + Flatten + 1 Dense	MAE
CNN 4	75	7	Nadam	2 CNN + Max Pooling + Flatten + 2 Dense	MAE

Table 4: Convolutional Neural Network Model Architectures for Time Series Forecasting Models

well. If the model is too simple it will not learn and if it is too complex the error will propagate through the network and reduce the accuracy of the model. Figure 17 is one the architecture of one of the CNN models we created for time series forecasting purpose.

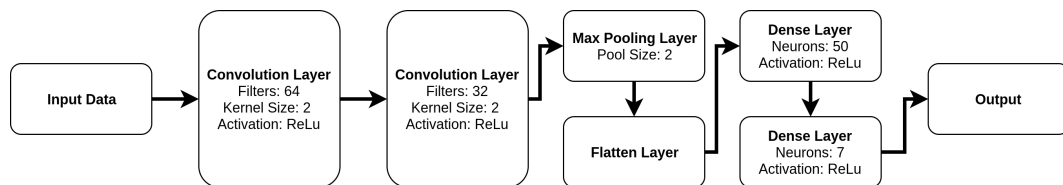


Figure 17: CNN Time Series Forecasting Model Architecture Flowchart

In this model, the data is reshaped and fed to the first convolution layer with 64 filters and a kernel size of two with the ReLU activation function. Then the output is fed to the second convolution layer with 32 filters and the same kernel size and activation function. The output of convolution layers is the Fourier transformed data in two dimensions. The output of previous steps is fed to the max-pooling layer with a pool size of two to reduce the number of parameters. Max pooling, as we mentioned before, can help reduce the noise as well. In the next layer, we flatten the data, which means that we reduce the dimensions from two to one and feed it to the first dense layer. The first dense layer has 50 neurons with the ReLU activation function, and the second dense layer has seven neurons, which represents

Model	Epochs	Batch size	Optimizer	Layers	Loss Function
LSTM 1	150	7	Adam	1 LSTM + 1 Dense	MSE
LSTM 2	128	7	Adam	2 LSTM + 1 Dense	MSE
LSTM 3	50	7	Nadam	2 LSTM + 1 Dropout Layer + 2 Dense	MSE
LSTM 4	75	7	Nadam	3 LSTM + 1 Dense	MAE
LSTM 5	70	7	Nadam	4 LSTM + 3 Dropout + 1 Dense	MAE

Table 5: Long Short Term Memory Time Series Forecasting Models

the next seven days that we are trying to predict. Dense layers are fully connected layers that we mostly use at the end of the network to produce the outputs.

**Long Short Term Memory** As mentioned in the previous paragraph, for deep learning models, we can have several architectures, and it is also true for our LSTM models. LSTM model is more complicated and requires more time for training and testing. On the other hand, we can achieve much better results with it.

To build the LSTM models, we use the same library that we used for CNN, which is Keras, and we use the Tensorflow backend as well. LSTM layers are available in Keras as a built-in module. We simply create a sequential model and add the LSTM and Dropout Layers to it, compile it, and fit the model to our datasets. We try different models with different hyperparameters and different architectures to build a model that can predict future values of our metrics with more accuracy. In table 5, we can see some of the models that we tried to achieve this goal.

Like CNN, we can have different architectures for an LSTM model as well. In figure 18 we can see one of the models. The mentioned model has three LSTM layers, which sequentially make the prediction. These kinds of models are also known as stacked LSTM models. The input data for LSTM has to be three dimensional, and creating it is our first step. Next, we feed the data to the LSTM layers, which have 100 neurons and use the

ReLU activation function. The last layer of the network is a fully connected or dense layer to produce the outputs for the next seven days with seven neurons and the ReLU activation function.

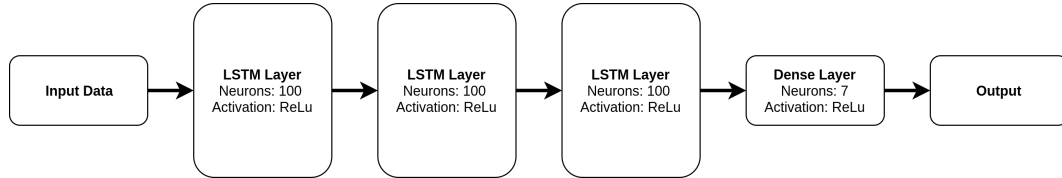


Figure 18: LSTM Time Series Forecasting Model Architecture Flowchart

#### 4.5.4 Time Series Forecasting Models

In the previous sections, we tried to develop models with regression analysis or deep learning methods to do the time series forecasting task. In this section, we use two out-of-the-box libraries that were developed for the same purpose. We got familiar with them in the Definition section, and here we see the functionalities and how we can create a model to fit on our datasets. First, we start with ARIMA and see the approaches we can use to build a model. Then we explore the functionalities of Facebook Prophet and build several models and compare them with each other. Later in the last chapter, we compare the results and select the best model to do the prediction task.

**ARIMA** We discussed the ARIMA and how it can predict future values based on historical data. ARIMA was originally a package developed for R programming language, and later it was developed for the Python programming language with the same functionalities as well. In order to build the ARIMA model, we need to use the "statsmodel" library, which has the ARIMA model already built-in. ARIMA requires three parameters as input, which the standard way to show it is ARIMA(p,d,q). Each of these values is positive integers and defined as follows:

- p: The lag order, which is the number of lags of observations(represents AR).

- d: The degree of difference, which is the number of times that the raw observations are different (represents I).
- q: The moving average, which the size of the moving average windows(represents MA)

We can build several models with different values for p,d, and q. However, we need to find the optimal values which make the ARIMA model make the predictions with more accuracy based on the data. One of the ways to find the best combination for these values is grid search. ARIMA model has a built-in function called AIC, which stands for Akaike Information Criteria, and has the ability to quantifies the fitness and the simplicity of the model into a single statistic. With the help of this function, we search through all of the models and find the most accurate one and fit it on our dataset to predict future values for seven days.

**Prophet** Facebook Prophet is a Time Series Forecasting Model like ARIMA. The main goal of the Prophet is to involve the analysts more into the process of making the predictions and that is the main reasons this model is much more tweak-able than ARIMA. Facebook prophet, by default, only requires one value which is the number of the days in future we need to predict, but we can pass the following parameters as well:

- Growth
- Holiday
- Change-points
- Seasonalities

In the definition chapter, we explained each of these parameters, but we build several models with each of these parameters and compare them with each other to see which

model is performing better on our datasets. Later we use the best model to do the prediction task with it.

## **4.6 Conclusion**

In this chapter, we started with approaches and ways that we can build a regression model to predict future values with time series forecasting method. We explored the ways we can build different models with different parameters. Moreover, we explored different algorithms of the deep learning method and discussed different models with different hyperparameters and architectures. At the end of this chapter, we discussed two of the most prominent time series forecasting libraries and explored how we can build the most optimal models with each of them. In the following chapter, we see the results of each model and compare them with each other.

# Chapter 5

## Experimental Results

### 5.1 Introduction

In the previous section, we presented how we developed different object detection and classification models and time series forecasting models. In this section, first, we explore the data itself and visualize it with different charts from a different perspective to have a better understanding and insight regarding our data. Moreover, we will plot the results from different models and compare them to see which model is doing a better job in object detection and classification or predicting the future traffic situation.

### 5.2 Object Detection and Classification Models

The first step for us is the results for the object detection and classification models. In this section, first, we go over the results of our image classifier and see if the results are reliable. Later, we see the results of the training and testing process of our object detection models and visualize the results to see how good our models are doing. Later we create the traffic dataset with the results we achieved in this section.

## 5.2.1 Data Cleaning Image Classification Model

As discussed in the data acquisition section while we were collecting data, lots of noise introduced into the dataset. In order to reduce the noise and improve the accuracy of the object detection and classification model we need to clean the data and remove the noise from it. To do this task we used a pre-trained Inception V3 model which was trained on ImageNet (Deng et al., 2009) images initially and used our own dataset to retrain the model. We collected around 200 distorted images and around 1600 not distorted images. For each class we tried to find as many examples as we can to make sure that model is robust and able to classify the images in any condition. We used 90% of the images for training and 10% of them for the testing process. The following figures are generated using Tensorboard and its functionalities.

**The Loss and the Accuracy of the Model** In this paragraph we are interested to see the accuracy and loss of the model. The classification model that we created here is much simpler and lighter than the models we will create in following sections for the vehicle detection and classification. Figure 19 shows the cross entropy loss of the Inception v3 (Szegedy et al., 2016) image classification model.

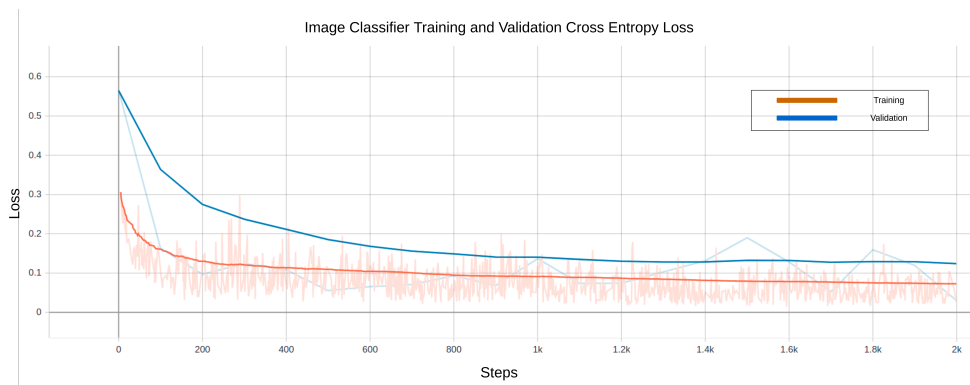


Figure 19: Cross Entropy Loss of the Inception v3 Data Cleaning Image Classification Model

We can see that at the beginning of the training process the training loss is 0.3545 and

the validation loss is 0.5424 and after 2000 steps the loss is reduced to 0.0757 and 0.1307 correspondingly.

Figure 20 shows the accuracy of the model for both training and validation.

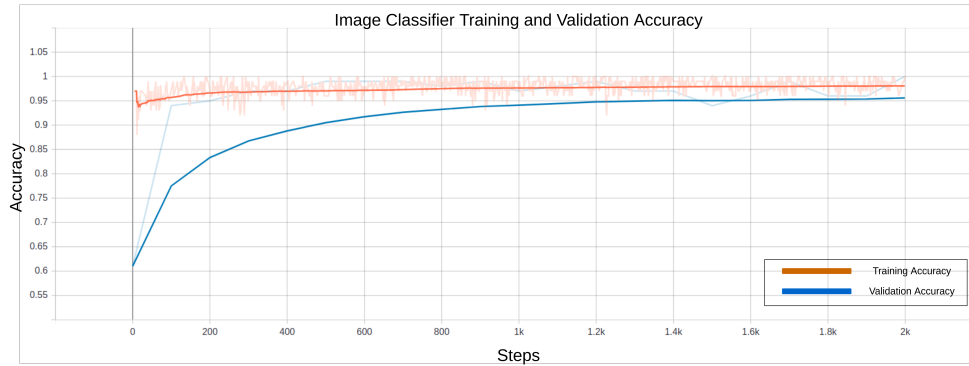


Figure 20: Accuracy of the Inception v3 Data Cleaning Image Classification Model

As shown in the figure 20 at the beginning of the training process the accuracy of the model for the validation is 0.66 and for the training is 0.91. After 2000 steps the accuracy of the model improved to 0.953 and 0.9788 accordingly.

**Results** In previous paragraph we discussed about the training and validation accuracy of the model. In this paragraph we are interested to check the model accuracy by applying it on our images dataset and see how the model is doing. In order to this task we input an image at a time to the model and record the output of the model for each image. Our goal is to see if the model can perform well on each image with different type of distortion. Later we will apply the model on the entire dataset and remove the images with high score of distortion.

In table 6 we can see the output of the classification model for some of the example images. Note that these images were not part of the training nor the test set. As we can see for both day and night the model has good accuracy of classifying the images into distorted and not distorted classes.

The final accuracy of the model on the test set is 95.9% which means the the model is







Image	Classification Model Output
	not distorted (score = 0.98590) distorted (score = 0.01410)
	not distorted (score = 0.99954) distorted (score = 0.00046)
	distorted (score = 0.97715) not distorted (score = 0.02285)
	distorted (score = 0.98736) not distorted (score = 0.01264)

Table 6: Table of the Results of the Data Cleaning Image Classification Model on Different Images

doing the classification task with a good accuracy based on comparing the model results and the grand truth.

### 5.2.2 Vehicle Detection and Classification Model

**Training and Hyper-Parameters** As we mentioned in the previous chapters, we built a pre-trained Faster R-CNN model with a Tensorflow object detection API (Huang et al., 2019). We used around 1000 images to train the model and around 10% of it for testing and evaluating the model. We labeled the images with an image label tool and created an

Model	Classes	Batch Size	Steps	Gradient Clipping	Optimizer
Faster RCNN NAS	4	2	200000	10	Momentum

Table 7: Faster RCNN Model Hyper-Parameters

Optimizer	Learning rate	steps	momentum optimizer value
Momentum Optimizer	0.0003	900000	10

Table 8: Momentum Optimizer Hyper-Parameters

XML file containing coordinates of the bounding boxes of objects for each image. Later a dictionary of the corresponding bounding boxes to each class was created and used to create the frozen graph, which ultimately is used to do the object detection and classification task. The four classes are personal cars, trucks, buses, and congestion.

Furthermore, we need to pass a configuration file containing all the hyper-parameters as well, which we can fine-tune the model and get better results. In the configuration file, there are around 50 hyper-parameters, but we do not need to change all of them. In the following tables, we show some of these values and explain their function.

Table 7 shows some of the hyper-parameters used to train the Faster R-CNN model. Batch size is the number of the images we are feeding to the model at the same time, and the main reason we used the value two is due to limitation in calculation power.

For the optimizer the Momentum optimizer is used and table 8 shows some of the hyper-parameters of the optimizer.

Furthermore, in table 9 we can more hyper-parameters for loss function. The main goal of the loss function is to give feedback to the model to show how close the results and target values in each iteration are.

Score Threshold	IOU	Localization loss weight	Objectness loss weight	Score converter
0.5	0.7	2.0	1.0	SOFTMAX

Table 9: Faster RCNN Model Loss Function Hyper-Parameters

**Losses** After the training process, we are interested to see how the model learned and how it managed to reduce the error rates. The model has to deal with two different tasks at the same time, detection and classification. After each iteration, the model gets feedback from the results and learn from the previous errors. Tensorflow comes initially with a tool to plot all the results and losses during or after the training process. We discussed more the functionalities of Tensorbord in the definition chapter, and we are going to use it for plotting our losses during the training. As we mentioned before, the model must tackle two different tasks at the same time, and we have a loss function for each of these tasks.

First, we start the detection task. We have to find the location of the object that we are looking for in the image, and then we can classify it. Figure 21 shows the localization loss of the model during the 15660 steps. The initial error rate was around 5.34 and reduced after each iteration to reach the minimum value of 0.36 at the end. In table 7, we mentioned that the first number of steps is 200000, but the model stopped after 15660 steps. The reason behind it is that the model reaches a plateau in error rate after around 15660 steps, and since no improvements are made, the model stops the training.

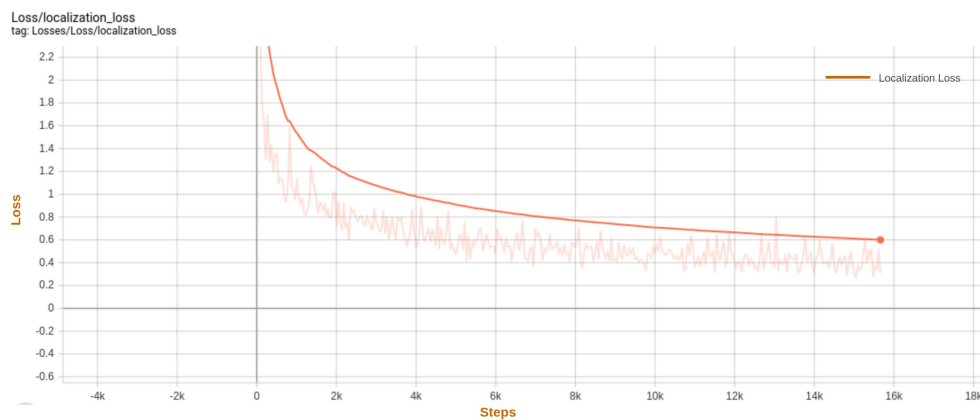


Figure 21: Classification(Objectness) Loss Of the Faster RCNN Model

After localizing the objects, the model does the classification task. We have four different classes for the objects that we are looking for in the images, which are personal cars, trucks, buses, and traffic congestion. In Figure 22, we can see the curve of the classification

loss after around 15660 steps. The initial error rate was 21.6, and in the end, the minimum amount of error is 1.7.

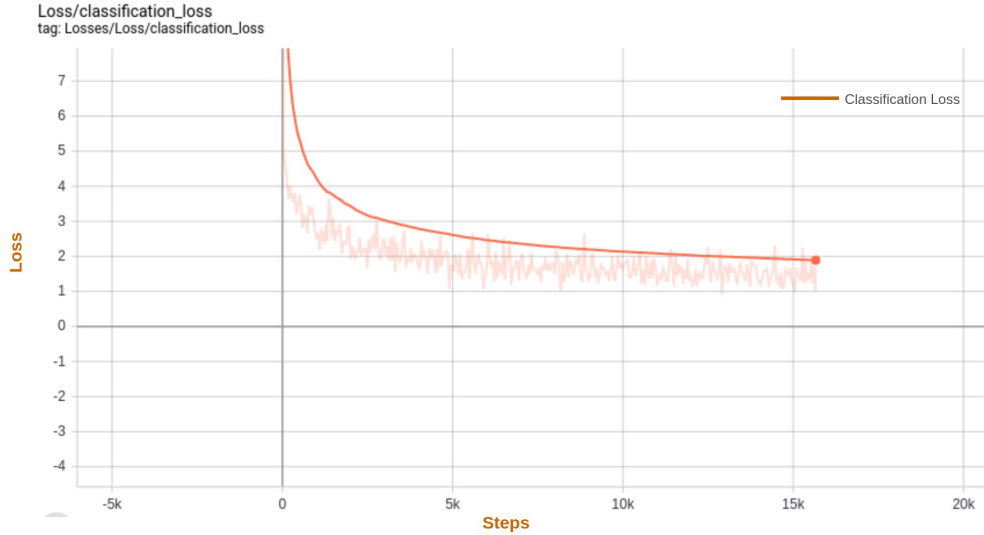


Figure 22: Localization(Detection) Loss Of the Faster RCNN Model

The final step is the total error, which is the sum of the localization and classification error. Figure 23 shows that the initial total error is 26.85, and after 15660 steps, the total amount of error is 2.41.

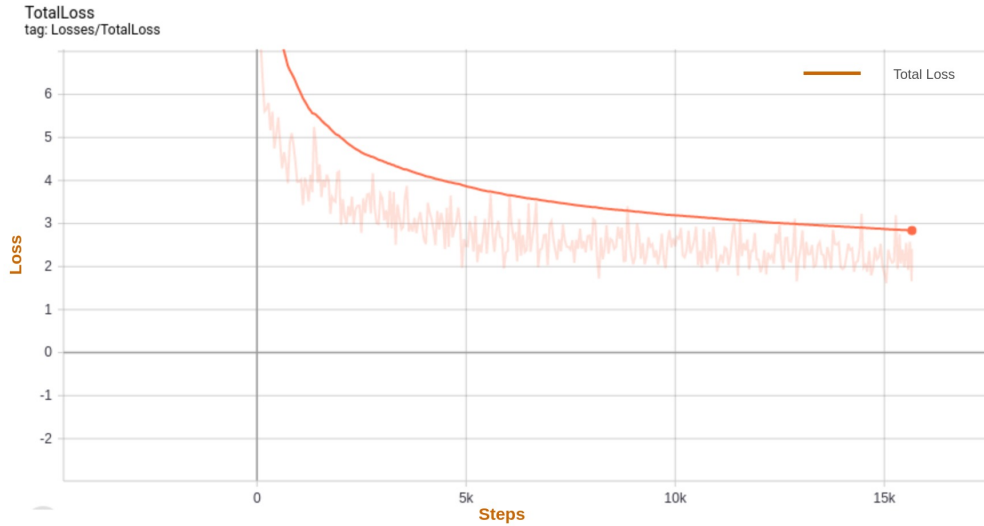


Figure 23: The Sum of the Classification and Localization Loss(Total Loss)

**Result Images** In the previous paragraphs, we explored model hyper-parameters and losses and saw how the model is reducing the error in each step. In this paragraph, we present the image results of the model and explain how the model managed to detect and label the objects in each image. In the data preparation section, we explained that we are expecting different weather and lighting situations, and we need to build a robust model to make sure it can detect and classify different types of vehicles in every situation. The following images are the output of our object detection and classification model in different situations.

- Day Images

Day images are normal images during the day without any condition, which makes the images blurry. Since the data was collected during April and May, most of the images fall into this category, and to make sure the model can do the detection and classification task with more accuracy, we select the most of the training images from this category. Figure 24 is an example of these images, and we can see the model can detect and classify the objects with relatively high scores.

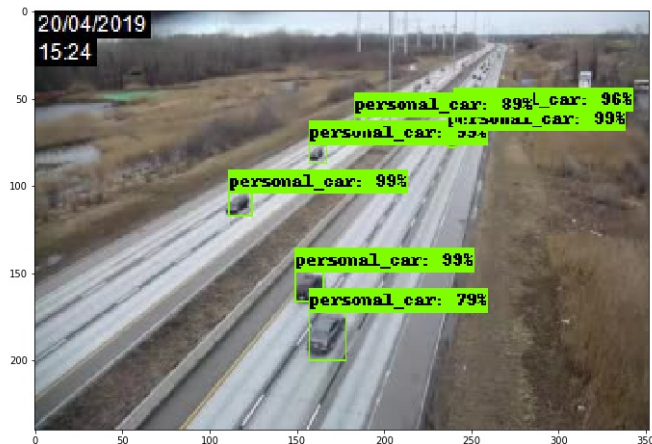


Figure 24: An Example of the Object Detection Model Output on Images with Normal Day Light Setting

- Night Images

Night images are images collected during the night, and the light setting can make the detection and classification tasks harder for the model. In these images, we do not have any condition that makes the image blurry. After normal day images, most of the images fall into this category. Figure 25 is an example of these images. Since the light setting is good in most parts of auto-route 40, the model can detect and classify the vehicles with good scores.

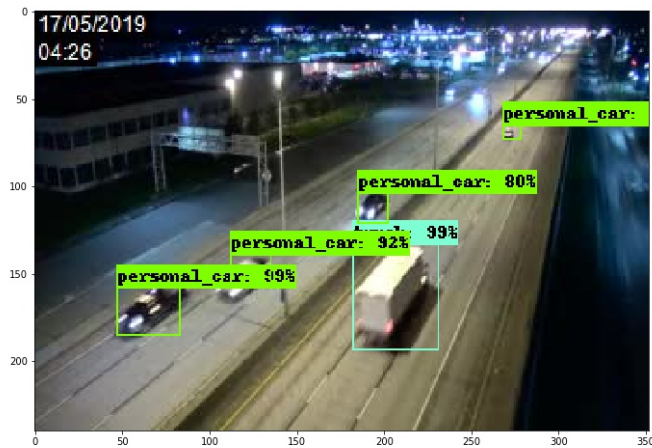


Figure 25: An Example of the Object Detection Model Output on Images with Normal Night Light Setting

On the other hand, in some locations, lack of proper lighting infrastructure reduce the quality of the image and ultimately reduce the accuracy of object detection and classification task. We provided enough examples to the model to make sure that even in these situations, it can still detect and classify the objects and keep the confidence level of the model relatively high. Figure 26 is another example of images with low Light Setting.

- Rainy Day Images

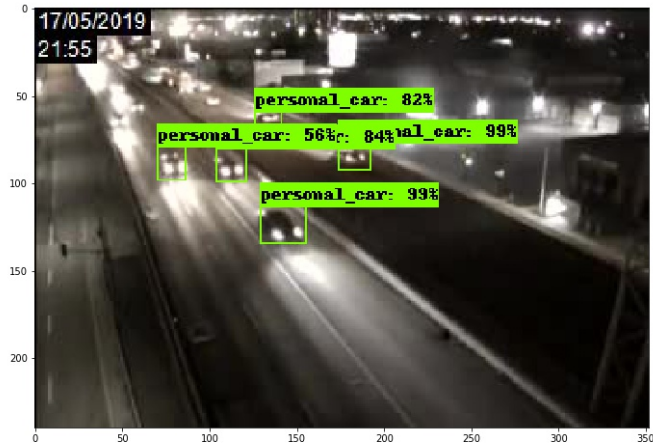


Figure 26: An Example of the Object Detection Model Output on Images with Low Light Setting

As mentioned before, based on the weather forecast, about 30% of our images are on rainy days. To get better results, we included images with rain conditions to our training set, and here we are interested to see the goodness of the results. Figure 27 shows the results of object detection in the rainy condition in daylight. Although lots of noise introduced to the image, we can see the model is doing a good job detecting and classifying the vehicles.

- Rainy Night Images

One of the hardest tasks for the model is detecting and classifying the vehicles on a rainy night. Like rainy days, we tried to feed enough data with a similar condition to the model to make sure it can do the task as accurately as possible. Figure 30 shows the result of the detection and classification task on a rainy night.

- Complex Day Images

In some of the images, we can see vehicles in different directions, and the architecture of the road is complex. In figure 29 We can see an overpass above a highway, and the

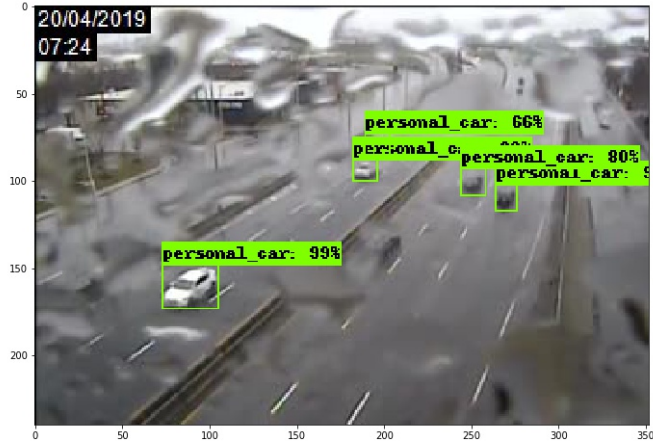


Figure 27: An Example of the Object Detection Model Output on Images with Rain Condition

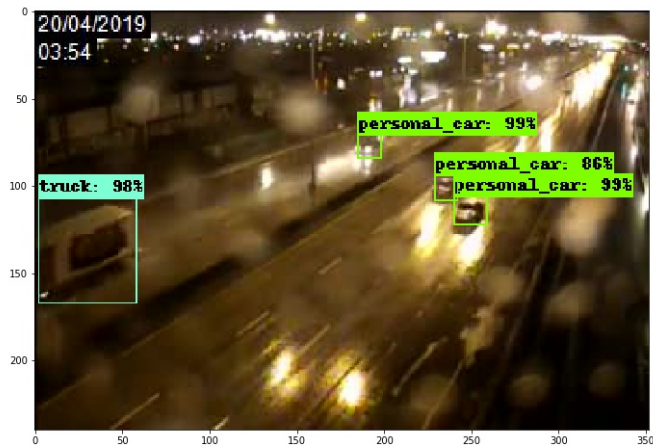


Figure 28: An Example of the Object Detection Model Output on Images with Low Light and Rain Condition



camera angle is a wide shot, which makes the detection and classification task harder for the model.

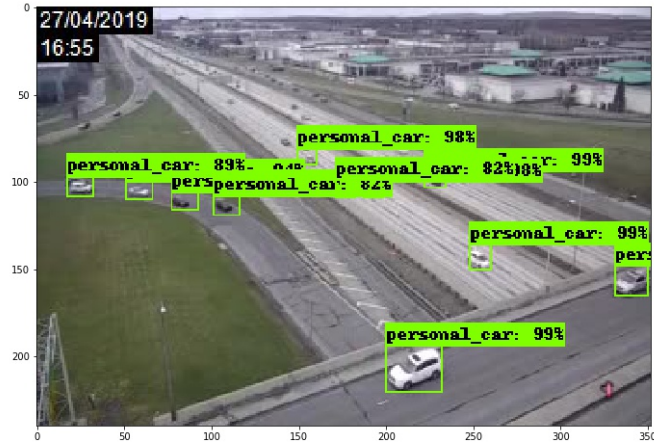


Figure 29: An Example of the Object Detection Model Output on Images with Complex Road Architecture

- Complex Night Images

We explained the difficulty of detection and classification tasks in images with complex road architecture. On top of this complexity, if we have a poor light setting as well, the model struggles if we do not provide enough examples of similar conditions. In figure 30, we can see the model is doing a good job detecting and classifying the vehicles in the image.

In the following images, we are interested to see the classification capabilities of the model. We want to see if the model can detect different types of vehicles and able to classify them correctly. We provided enough examples of each type of vehicle to make sure the model does this task as precisely as possible.

- Images with Personal Cars

The most common type of vehicles on the roads are personal cars. Most of the images



Figure 30: An Example of the Object Detection Model Output on Images with Complex Road Architecture and Low Light Setting

that we provided to the model contain a personal car from different perspectives or angles. Figure 31 shows one of the images which model was able to detect two personal cars in it. The model avoided detection of the light reflect as a personal car and able to detect personal cars with a high score.

- Images with Trucks

The other type of vehicle that we are interested in collecting data about is trucks. We would like to know their contribution to the traffic, and in which parts of the city, they are most active. We introduced many examples of trucks from different sizes and colors to different angles to the model to make sure that it can detect this type of vehicle with good accuracy as well. Figure 32 is the result of our model showing one detected truck on a rainy night.

- Images with Buses

In the process of collecting data, we could not collect a lot of data about buses, which is due to the situation in autoroute 40. Although there are not many buses in

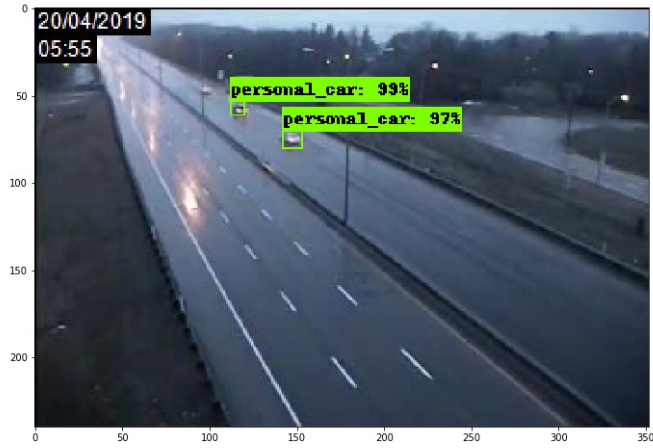


Figure 31: An Example of the Object Detection Model Output on Images with Personal Cars In Them



Figure 32: An Example of the Object Detection Model Output on Images with Trucks In Them

our dataset, we tried to feed as many data with bus examples as possible to make sure the model can detect and classify a bus in a photo with a good score. In figure 33 alongside other types of vehicles, we were able to detect a bus in a relatively far distance.

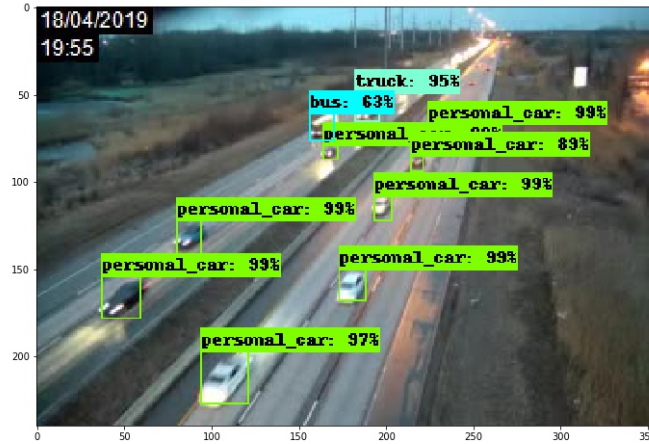


Figure 33: An Example of the Object Detection Model Output on Images with Buses In Them

- Images with Traffic congestion

Earlier in the research, we understand that the total number of vehicles is not necessarily a good representation of traffic congestion. The width of the road and the angle of the camera can affect this decision. To find the traffic metrics, we introduced another layer to the training dataset, congestion. We manually labeled traffic congestion in an image and provided different examples from different locations and different situations to the model. Here we are interested to see the results of detection and classification of the traffic congestion in images. In figure 34 we can see alongside the personal cars and a truck the model detected the traffic congestion and its location as well.

- Example of Crowded Images without Traffic Congestion

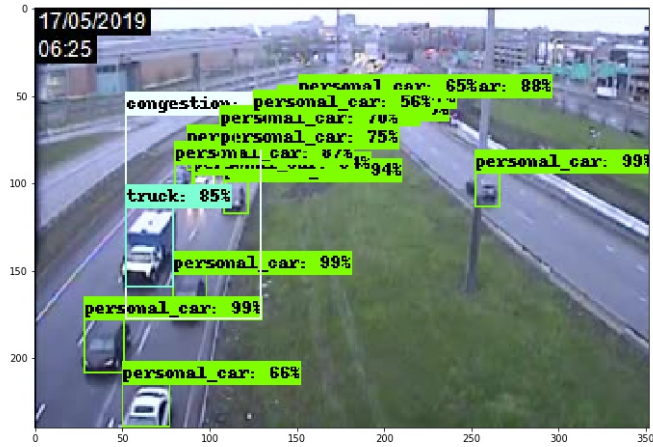


Figure 34: An Example of the Object Detection Model Output on Images with Road Traffic Congestion In Them

As we explained in the previous paragraph, the main reason that we introduced another layer to our dataset is that the number of vehicles alone can not represent the traffic situation. In figure 35, we can see although like figure 34 many personal cars and a truck are detected, but there is no traffic congestion in the road.

**Creating The Traffic Dataset** In the previous section, we presented how we are generating images with bounding boxes in them, defining the detected objects. To make traffic predictions, we need to export this information from images to a single CSV file and create out traffic dataset. The final dataset is a giant spreadsheet with features as columns and information of each image as rows. In figure 36, we can see a sample of the entire dataset. We have seven columns that represent the features and 328376 rows, which is the information of each image.

The columns are date, time, personal\_car, truck, bus, congestion, and address. The first two columns, date and time, are collected with a Python script and assigned to each image. The address is collected alongside the image from the Quebec511.info (Quebec511, 2019) website. The main four features, personal\_car, truck, bus, and congestion, are collected

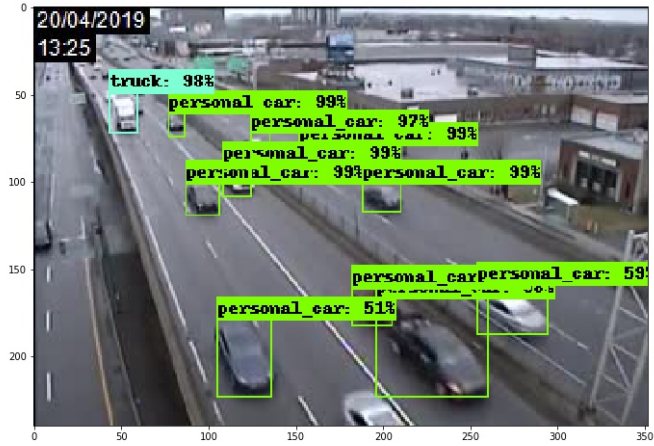


Figure 35: An Example of the Object Detection Model Output on Images with Lots of Vehicles in Them but No Traffic Congestion

Index	date	time	personal_car	truck	bus	congestion	address
314	2019-04-13	14:49:52	6	0	0	0	Aut. 40 at boul. de La Côte-Vertu
315	2019-04-13	14:49:52	15	0	0	0	Aut. 40 at boul. Saint-Michel
316	2019-04-13	14:49:52	1	0	0	0	Autoroute 40 (westbound), at the centre of the Décarie interchange
317	2019-04-13	14:49:52	7	0	0	0	Aut. 40 at rue Halpern
318	2019-04-13	14:49:52	18	0	0	0	Aut. 40 at Côte-de-Liesse interchange
319	2019-04-13	14:49:52	4	0	0	0	Aut. 40 at aut. 15
320	2019-04-13	14:49:53	10	0	0	0	Aut. 40, east of d'Anjou interchange
321	2019-04-13	14:49:53	1	0	0	0	Aut. 40 at the CN overpass
322	2019-04-13	14:49:53	6	0	0	0	Aut. 40 at Golf de l'Île de Montréal
323	2019-04-13	14:49:53	6	0	0	0	Aut. 40 at boul. des Galeries-d'Anjou
324	2019-04-13	14:49:53	19	0	0	0	Aut. 40 at av. Marlen
325	2019-04-13	14:49:53	8	0	0	0	Aut. 40 at boul. Bourget
326	2019-04-13	14:49:53	3	0	0	0	Aut. 40 at boul. Henri-Bourassa
327	2019-04-13	14:49:53	1	0	0	0	Aut. 40 at pont Charles-De-Gaulle (Montréal side)
328	2019-04-13	14:49:53	2	0	0	0	Aut. 40 at the boul. du Tricentenaire exit
329	2019-04-13	14:54:51	5	0	0	0	Aut. 40 at pont de l'Île-aux-Tourtes (sainte-Anne-de-Bellevue side)
330	2019-04-13	14:54:52	4	0	0	0	Aut. 40 at aut. 13
331	2019-04-13	14:54:52	6	0	0	0	Aut. 40 at rue Halpern
332	2019-04-13	14:54:52	19	0	0	1	Aut. 40 at rue du Marché-Central
333	2019-04-13	14:54:52	2	0	0	0	Aut. 40 at boul. Cavendish (southward)
334	2019-04-13	14:54:52	6	0	0	0	Aut. 40 at boul. Saint-Jean
335	2019-04-13	14:54:52	16	0	0	0	Aut. 40 at av. Christophe-Colomb
336	2019-04-13	14:54:52	7	0	0	0	Aut. 40 at boul. Cavendish (northward)

Figure 36: The Vehicle Count Dataset

by passing each image to the trained model exporting the numerical output of the count of each vehicle.

## 5.3 Data Visualization

One of the best ways to explore the data is by visualizing it. Now that we created our numerical traffic dataset, we want to explore it more and see how it looks like from different angles. Our goal in this section is to visualize the numerical data and try to explore it from different perspectives and see what we can learn from them. We used Matplotlib and Seaborn to plot our datasets.

**Line plots of distribution of features in each hour during the month** We are interested to see which hour during the day are the most crowded hours and how each type of vehicle is participating in it.

- **Personal Cars**

In Figure 37 you can see the distribution of personal cars during the day in each hour.

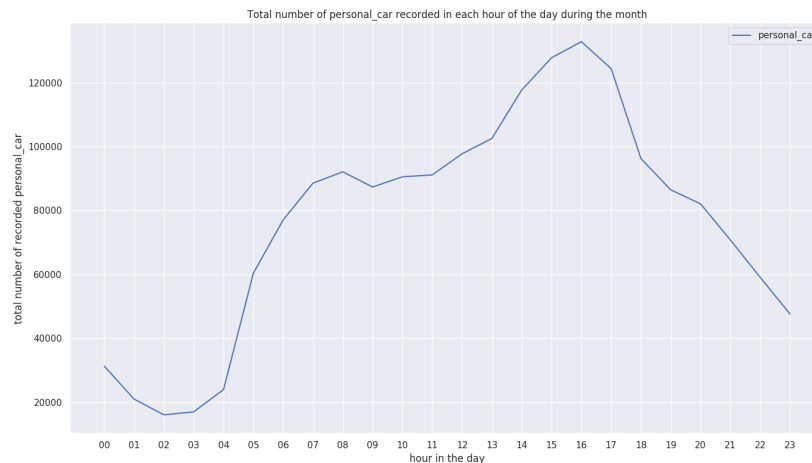


Figure 37: Total Number of Personal Car in each Hour of the Day

One of the things that we can easily notice from Figure 37 is that we have a sudden

increase in the number of personal cars around 4 am, which reaches a local maximum around 8 am and again we have the global maximum around 5 pm. We are interested to see if other metrics are following the same order.

- **Trucks** We used the same approach to plot the total number of trucks recorded in each hour of the day. Figure 38 is the result.

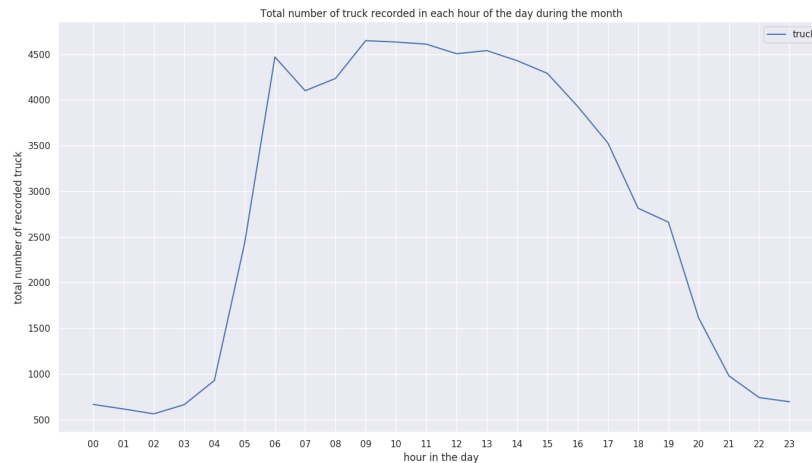


Figure 38: Total Number of Trucks in each Hour of the Day

Just like personal cars, the number of trucks recorded by the cameras started to increase around 4 am and reached a local maximum of around 6 am. The total number of truck stays almost the same until 9 am and starts to decrease afterward. We can also see that the total number of trucks between 9 pm to 4 am is below 1000.

- **Busses**

As we mentioned before, in our data set, there are not many buses recorded, and data may not be accurate. We mentioned that the reason is we are just collecting data from autoroute 40, and not many buses use the autoroute. We used the same approach and plotted the total number of busses we recorded during the day. Figure 39 show how the total number of busses is distributed in each hour.

Since the number of buses is small, we can easily see the distribution range in each



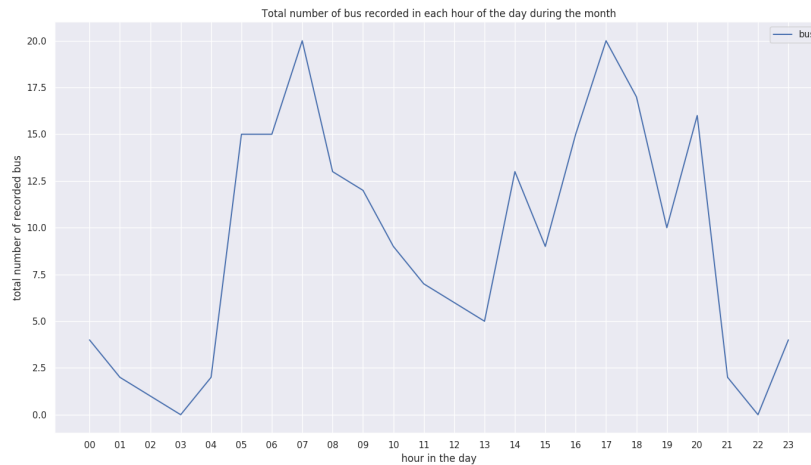


Figure 39: Total Number of Buses in each Hour of the Day

hour as well. Like personal cars and truck, the total number of busses start to increase around 4 am as well and reach a local maximum of 20 buses around 7 am. Furthermore, in the afternoon around 5 pm, we can see another peak hour with around 20 buses recorded at that time as well. The total number of buses drop below 5 around 9 pm and is almost the same till 4 am.

- **Congestion**

To find congestion, we introduced a new bounding box in our object detection model to not estimate the traffic situation based on the total number of cars visible in front of the camera. Here we are interested to see the results and to compare them with the data that we collected from the total number of personal cars, trucks, and buses. Figure 40 shows the distribution of traffic congestion during the day, which is collected for the entire month.

As we suspected, there is an obvious correlation between the number of personal cars, trucks, and buses, and the congestion level. We can see that from 5 am the congestion level starts to increase till the local maximum around 7 am, and 8 am. On the other hand, in the afternoon, we have a huge spike around 4 pm, which marks the

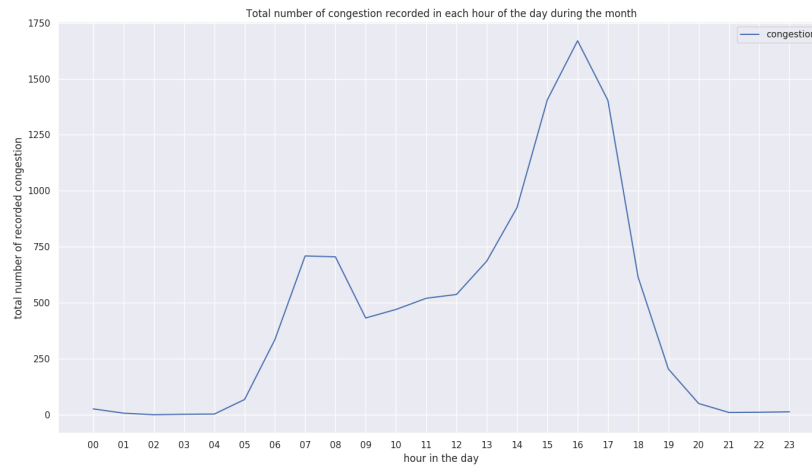


Figure 40: Total Number of Recorded congestion in each Hour of the Day

peak hour. The results from Figure 40 prove that our initial assumption about having a separate label for congestion is correct, and the data is more reliable.

**Bar plots of distribution of features in each location during the month** In this section, we are interested to see which part of the city is the most congested place. We would like to know how each type of vehicle is most common in different parts of the autoroute 40. This information can help us to prioritize the most congested areas to build new infrastructure.

- **Personal Cars** First explore the most congested areas with personal cars. Having this information can help us come up with better solutions to alleviate the traffic situation designed for personal cars in these areas. In Figure 41 we plotted total number of cars recorded in each location.

Based on the information we gathered and plotted, following places has highest record in total number of cars:

1. Boulevard Sainte-Croix
2. Rue Stinson
3. Avenue Deslauries

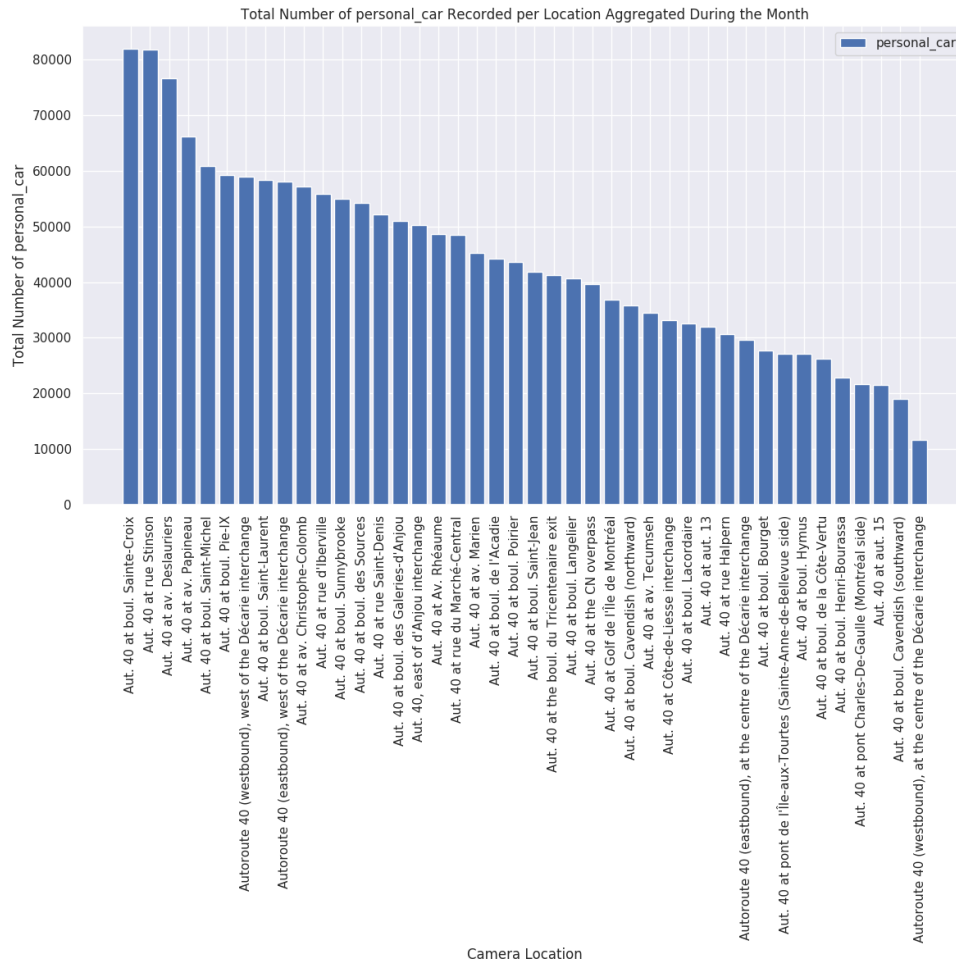


Figure 41: Total Number of Personal Cars in each Location During the Month

- 4. Avenue Papineau
- 5. Boulevard Saint-Michel

• **Trucks** The other feature, which is important to us, is the total number of trucks. We are interested to see the distribution of trucks in autoroute 40. Figure 42 shows most congested areas with truck.

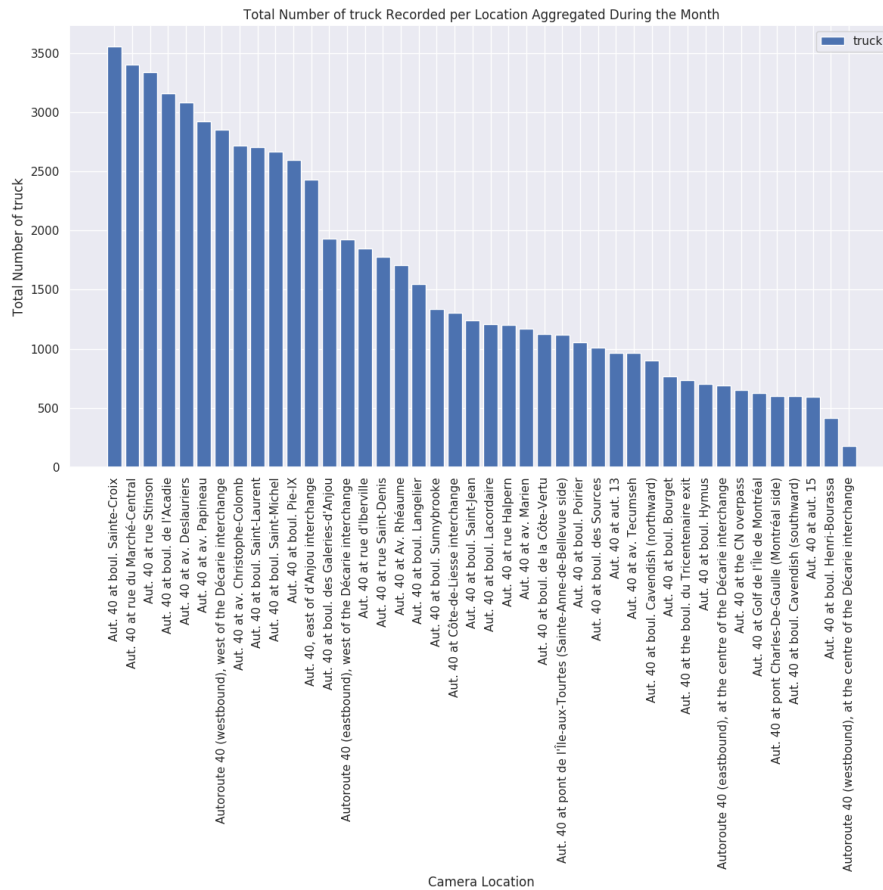


Figure 42: Total Number of Trucks in each Location During the Month

Based on the provided information we can see that truck were more populated in the following areas:

- 1. Boulevard Sainte-Croix
- 2. Rue du Marche-Central

3. Rue Stinson
4. Avenue Deslauries
5. Boulevard de l'Acadie

- **Buses** Although the total number of buses was no as much as personal cars or trucks, we are still interested to see how the buses participated in the traffic and which part of the autoroute 40 they have been the most. Figure 43 shows most congested areas with buses.

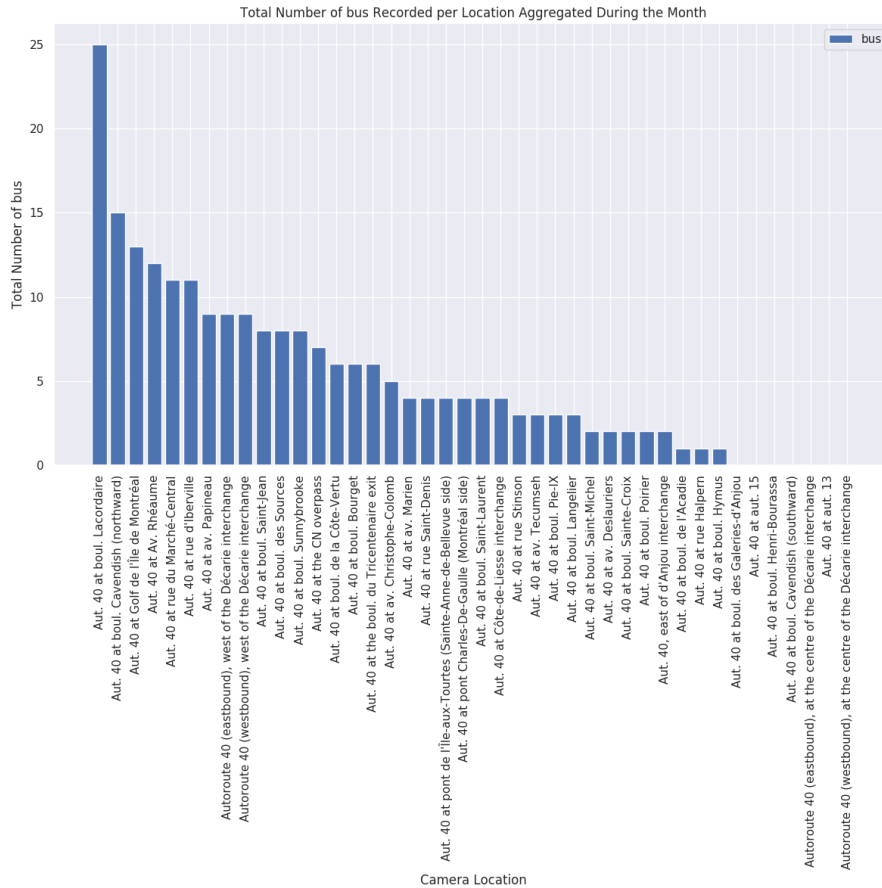


Figure 43: Total Number of Buses in each Location During the Month

Based on the collected information in following areas we witness most of the buses.

1. Boulevard Lacordaire

2. Boulevard Cavandish
3. Golf de l'île de Montreal
4. Avenue Rheaume
5. Rue du Marche-Central

- **Congestion** In the previous parts, we saw how different types of vehicles are distributed, and we are interested to see the traffic situation distribution as well. Figure 44 show the distribution of traffic in autoroute 40.

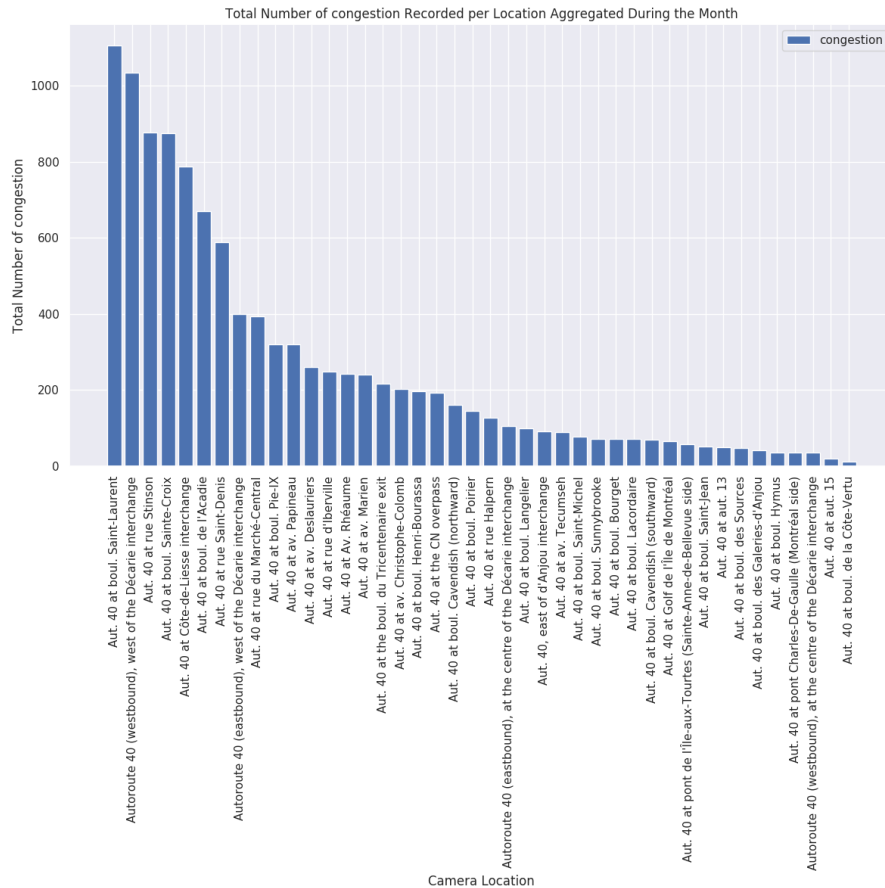


Figure 44: Total Number of Congestion Recorded in each Location During the Month

Based on the collected information following areas are the most congested areas.

1. Boulevard Saint-Laurant

2. West of the Decarie Interchange(westbound)
3. Rue Stinson
4. Boulevard Sainte-Croix
5. Cote-de-Liesse Interchange

**Heatmap of Correlation Matrices** In the previous section, we plotted all the metrics and explored more about them regarding their distribution during the day and location. Now we are trying to find out which metric is contributing more to the traffic congestion. We created a heatmap based on the correlation matrix of the features. Figure 45 show the correlation of the metrics to each other and traffic congestion. The darker each tile has, the more correlation exists between the features.

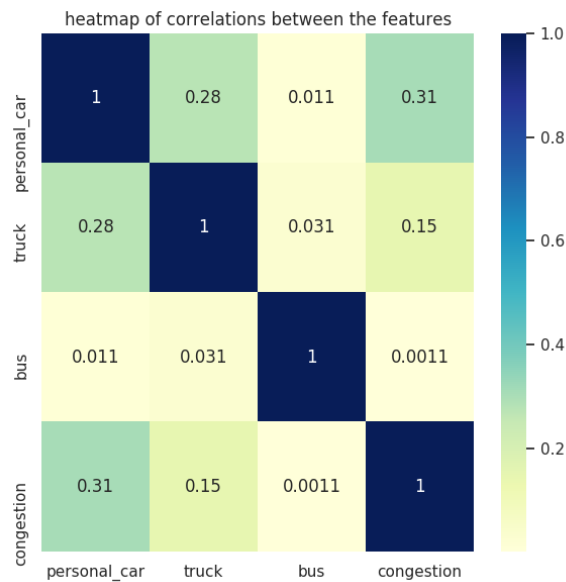


Figure 45: Heatmap of Correlation Between the Features

From the heatmap, we can understand that the number of personal cars has the highest correlation to traffic congestion, followed by the number of trucks and buses.

## 5.4 Time Series Forecasting

In previous sections, we constructed a dataset based on the information we collected from the images. Then we visualized the data and got better insights about it, now we are interested to see the results of our models for the time series traffic tasks.

### 5.4.1 Regression Analysis Models

As mentioned in previous chapters, we got familiarized with three regression models and explored the ways to develop them. We developed three regression models for each task, and we are curious to see how each model is performing. In this section, we explore the regression model's performance.

**Linear Regression** We used each of our metrics independently and developed a univariate Linear Regression model based on the time step approach. The following are the results for each of our metrics.

As mentioned before, we used the first four weeks of the data for training purposes, and last week for evaluating the model. It means that the model never saw the data from last week, and the goal is to predict these values.

- **Personal Cars**

First, we explore the prediction for the number of personal cars with the Linear Regression. In Figure 46, we can see the data we collected and trained the model based on in green and grand truth or the target values in orange. The blue line is the prediction results of our Linear Regression model for the last week of the data.

In Table 17, we can see the performance of the model calculated with different performance metrics, but we are mainly interested in the  $r^2$  score of the model, which is almost 0.45.



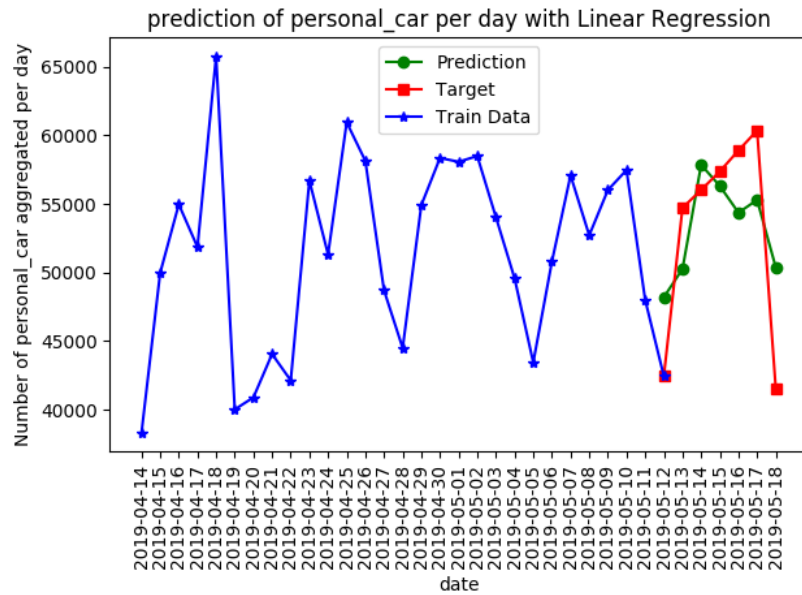


Figure 46: Prediction Results for Number of Personal Cars with Linear Regression Model

Target Values	Linear Regression Model Predictions
42439	48196.2897
54708	50253.6545
56015	57843.0119
57367	56304.7188
58890	54358.9869
60349	55286.9023
41518	50373.9975
R2 Score	0.4498
MAE	3575.2183
MSE	16996951.5623

Table 10: Prediction Results and the Performance Metrics of the Linear Regression Model on Total Number of the Personal Car

- **Trucks**

We used the same methodology to develop a Linear Regression model to predict the number of trucks for the seven days of last week. In Figure 47, we can see the predictions for the last week in the blue line, target values in orange, and training data in green.

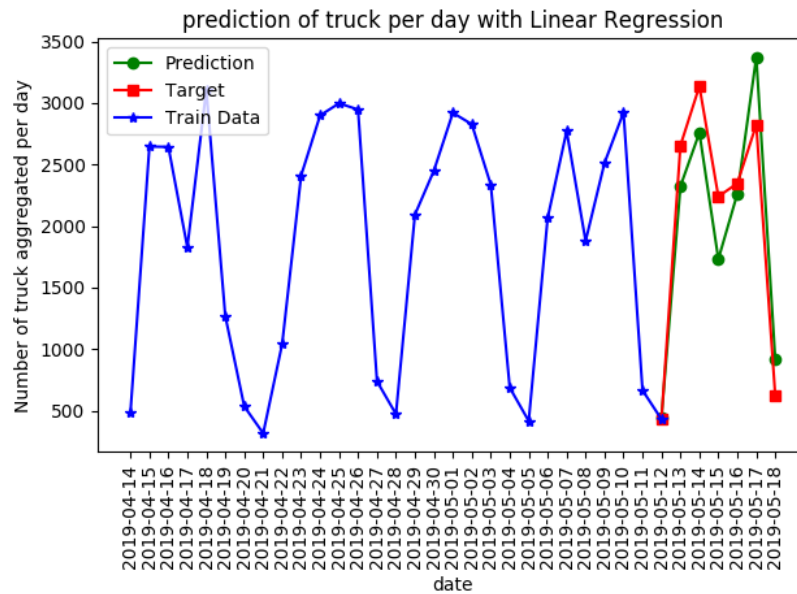


Figure 47: Prediction Results for Number of Trucks with Linear Regression Model

We can see the results for predicting the number of trucks in Table 11 and the r2 score that we achieved with the model, which is 0.75. For a simple model like Linear Regression, this prediction results are really good.

- **Buses**

We developed a Linear Regression model for buses as well, like the total number of personal cars and trucks. In Figure 48, you can see the prediction results for number buses for the last week.

One of the main reasons of the negative value of r2 score in Table 12 is that we did not have enough data, and it is hard for the linear model to predict based on that.

Target Values	Linear Regression Model Predictions
430	444.4231
2654	2323.0764
3136	2758.2367
2240	1731.0966
2347	2261.9273
2824	3370.8444
621	915.0013
R2 Score	0.7488
MAE	397.4485
MSE	236759.8716

Table 11: Prediction Results and the Performance Metrics of the Linear Regression Model on Total Number of the Trucks

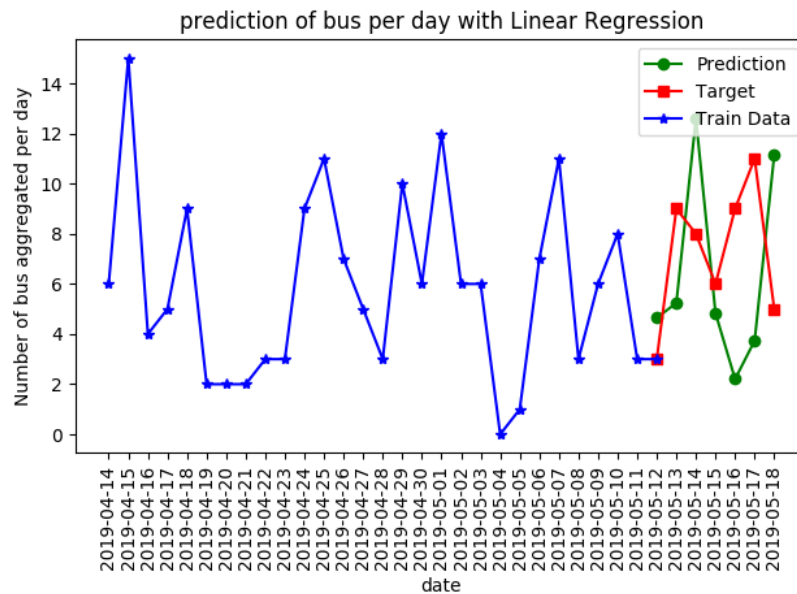


Figure 48: Prediction Results for Number of Buses with Linear Regression Model

Target Values	Linear Regression Model Predictions
3	4.6433
9	5.2413
8	12.5940
6	4.8377
9	2.2049
11	3.7347
5	11.1372
R2 Score	-2.0544
MAE	3.9056
MSE	21.9825

Table 12: Prediction Results and the Performance Metrics of the Linear Regression Model on Total Number of the Buses

Furthermore, a negative  $r^2$  score means that the model is doing a worse job than just getting the mean of the data. On the other hand, the reason that mean absolute error and mean squared error are better in predicting number of busses than personal cars or truck, is because the total number is much smaller than the mentions metrics and these two performance metrics rely on the magnitude of the data.

- **Congestion**

Just like previous metrics, we developed a univariate Linear Regression model for the total number of recorded congestion as well. You can see the predictions results in Figure 53. Like previous plots, the prediction values are in blue, and the target values are in orange, and the green line represents the historical data that we used to train the model.

Based on Table 13, the  $r^2$  score for our Linear Regression model to predict the next seven days of traffic situation is 0.4942. We can improve the  $r^2$  score by applying more sophisticated approaches in the next sections.

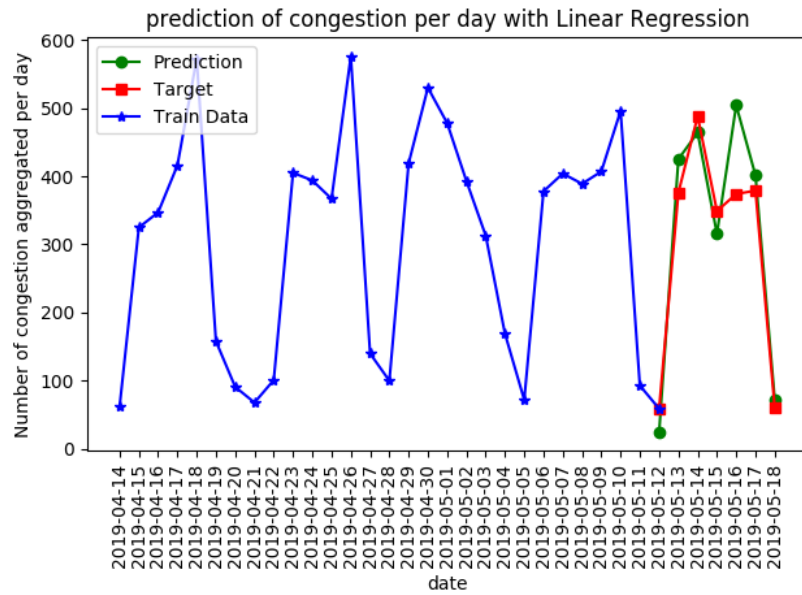


Figure 49: Prediction Results for Number of Recorded congestion with Linear Regression Model

Target Values	Linear Regression Model Predictions
59	24.3638
376	425.1287
488	464.1596
349	315.6578
374	504.1421
379	401.4689
61	72.3213
R2 Score	0.4942
MAE	90.0127
MSE	12980.5399

Table 13: Prediction Results and the Performance Metrics of the Linear Regression Model on Total Number of Road Traffic Congestion

Target Values	Polynomial Regression Model Predictions
42439	43847.9530
54708	58282.3072
56015	60753.8089
57367	54976.1864
58890	57715.1357
60349	55738.9814
41518	46190.5077
R2 Score	0.3253
MAE	3858.5803
MSE	20393889.2532

Table 14: Prediction Results and the Performance Metrics of the Polynomial Regression Model on Total Number of Personal Car

**Polynomial Regression** After Linear Regression we are interested to see how a polynomial regression model would perform on our dataset. The main difference between linear regression and polynomial regression is the degree that we need to define for the model. In order to find the best regression model we searched through the degrees and selected the best degree with the best r2 score results. First, we start with the total number of personal cars and find the best polynomial degree and build a model with it. we will do the same thing for other metrics and plot the results for each of them. We used the first four weeks of data in order to train the model and last week for the test purpose.

- **Personal Cars** After searching for the best polynomial degree on the total number of personal cars we got the best results with the polynomial degree of three. We can see the polynomial regression results with degree three in table 18.

The r2 score is 0.3253 which is lower than the linear model. It means the the linear model can make better predictions than polynomial model with degree of three. It may be due to the behaviour of our data.

- **Trucks** We applied the same method on the total number of the trucks and found the best polynomial degree for this metric as well. We got the best results with

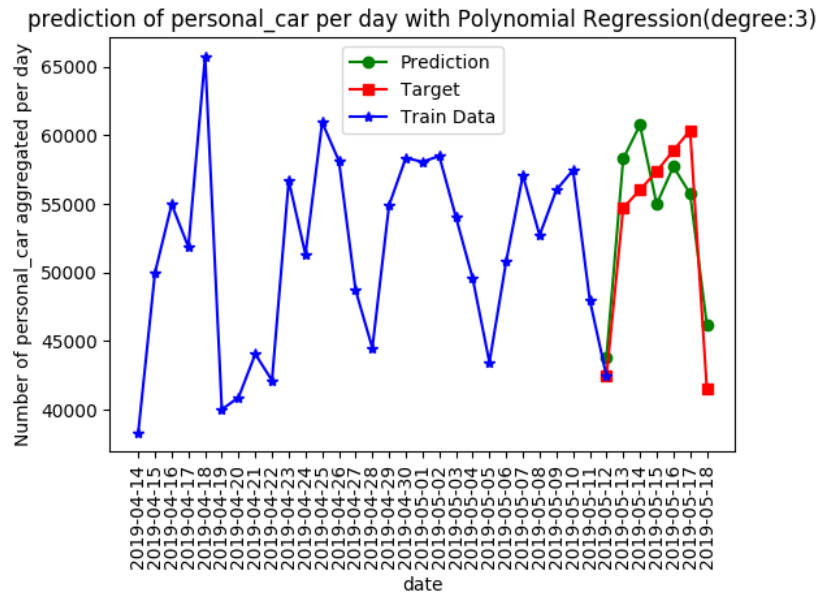


Figure 50: Prediction Results for Total Number of Personal Car with Polynomial Regression Model with Degree Three

polynomial degree of two and fit the model on our data to get the results for the prediction. In table 15 we can see the results for this model.

- **Buses** We applied the same approach for the polynomial regression on the bus metric as well. First we searched through the degrees to find the best results. We got the best results with polynomial degree of two and fit the model on the data to get the results. In table 16 we can see the predictions alongside the performance of the model for this metric.
- **Congestion** The last metric is congestion. After searching for the best polynomial degree we got the best results with degree two.

**Random Forest Regression** In the previous section, we explored the results for two different Regression models. In this section, we explore how the Random Forest Regression model is performing the prediction task on our data. Like previous sections, we start with

Target Values	Polynomial Regression Model Predictions
430	349.2531
2654	1146.8583
3136	1661.2473
2240	2388.7453
2347	2376.8041
2824	4034.4875
621	1418.3530
R2 Score	0.3960
MAE	615.1382
MSE	584332.9466

Table 15: Prediction Results and the Performance Metrics of the Polynomial Regression Model on Total Number of Trucks

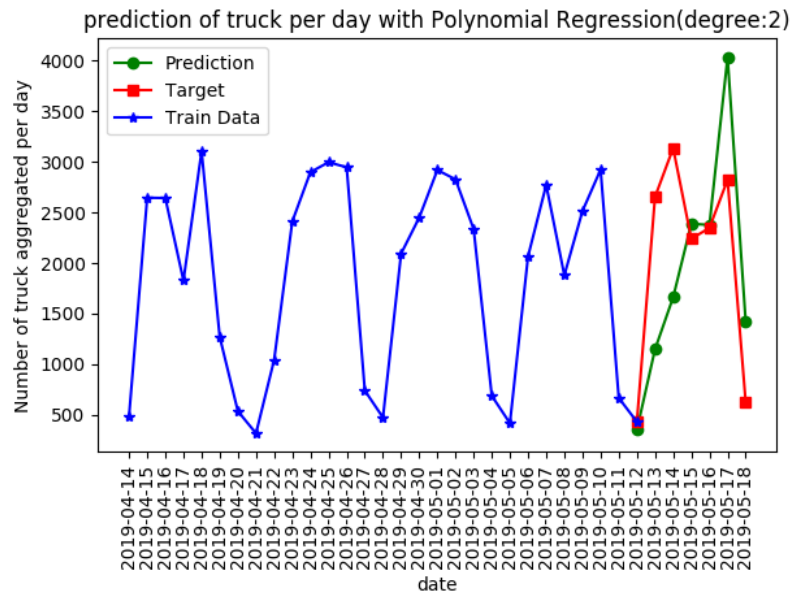


Figure 51: Prediction Results for Total Number of Trucks with Polynomial Regression Model with Degree Two



Target Values	Polynomial Regression Model Predictions
3	4.349
9	3.1114
8	14.8195
6	3.1194
9	7.9338
11	-0.4654
5	15.6249
R2 Score	-4.9567
MAE	5.2607
MSE	43.79475

Table 16: Prediction Results and the Performance Metrics of the Polynomial Regression Model on Total Number of the Buses

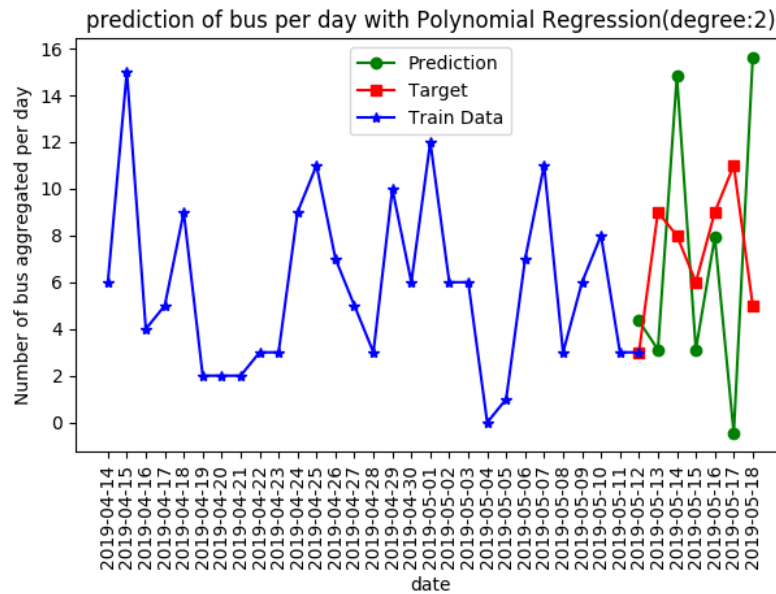


Figure 52: Prediction Results for Total Number of Bus with Polynomial Regression Model with Degree Two

Target Values	Polynomial Regression Model Predictions
59	-187.7558
376	260.9183
488	454.8129
349	449.6281
374	418.1361
379	643.1869
61	-56.9476
R2 Score	-1.3102
MAE	177.1153
MSE	56203.9212

Table 17: Prediction Results and the Performance Metrics of the Polynomial Regression Model on Total Number of Road Congestion

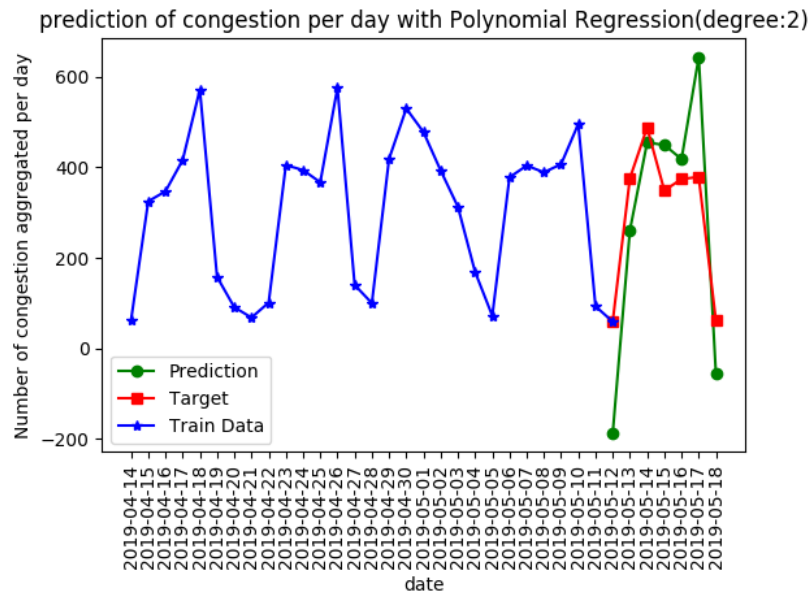


Figure 53: Prediction Results for Total Number of Personal Car with Polynomial Regression Model with Degree Two

Target Values	Random Forest Regression Model Predictions
42439	46015.2777
54708	53147.1666
56015	57272.77
57367	56447.555
58890	57146
60349	54417.2222
41518	48028.7222
R2 Score	0.56378
MAE	3227.9546
MSE	14168119.8282

Table 18: Prediction Results and the Performance Metrics of the Random Forest Regression Model on Total Number Personal Cars

the total number of personal cars and finish with the results of the univariate random forest regression model for the congestion metric.

To train the random forest regression model, we need to specify the number of estimators. Estimator, as we mentioned before, is the number of our decision trees. First, we need to find the optimal number of decision trees, and then we can train a random regression model with the optimal number of estimators. We explained in the methodology chapter that since we have limited resources and our dataset is relatively simple, we do not need to choose extreme values for the number of estimators, and a simple search gives us the desired results.

- **Personal Cars**

As mentioned before, first, we searched for the optimal number of the estimators, and then we train the random forest regression model. For the total number of personal cars, we found out that the optimal number of estimators is 18, and we trained the model 18 decision trees. we can see the prediction results in Figure 54. The prediction values are specified with the blue line, orange represents the target values, and green stands for the historical data and the data that we used to train or model.

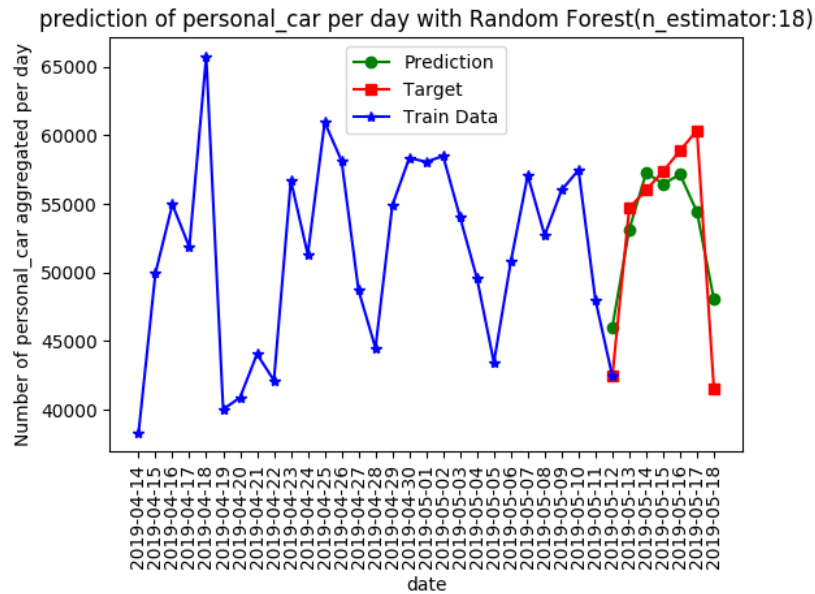


Figure 54: Prediction Results for Number of Personal Cars with Random Forest Regression Model with 18 Estimator

- **Trucks**

We developed a random regression model with the same idea for predicting the total number of trucks for the 5th week of the data. First, we explored what the optimal number of the estimator for our random forest model is, based on the data we collected for the total number of trucks. After searching, we got the best results for prediction with 1 estimator.

Figure 55 show the predictions in the blue line, target values in orange, and the historical data which we used for training in green.

Table 19 shows how the model is performing based on different performance metrics. Our r2 score value is 0.6694, which is in an acceptable range for this model.

- **Buses**

Like previous metrics, we applied the same idea for developing the random forest regression for predicting the total number of buses as well. In our search to find the

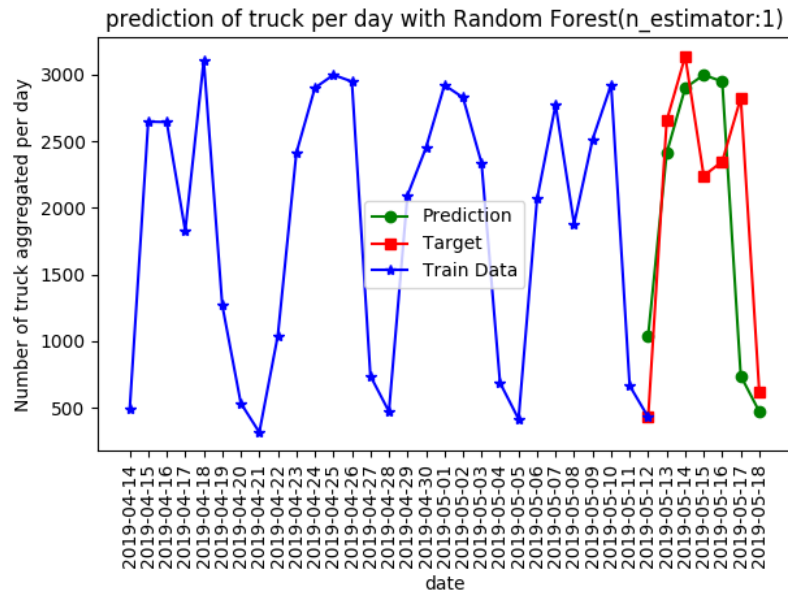


Figure 55: Prediction Results for Number of Trucks with Random Forest Regression Model with One Estimator

Target Values	Random Forest Regression Model Predictions
430	1042
2654	2411
3136	2899
2240	2997
2347	2947
2824	739
621	472
R2 Score	0.6694
MAE	403.8367
MSE	313307.5510

Table 19: Prediction Results and the Performance Metrics of the Random Forest Regression Model on Total Number of Trucks

optimal number of estimators, we found out that we are getting the best prediction results with four estimators. The figure shows the prediction results in blue, the target values in orange, and historical data in green.

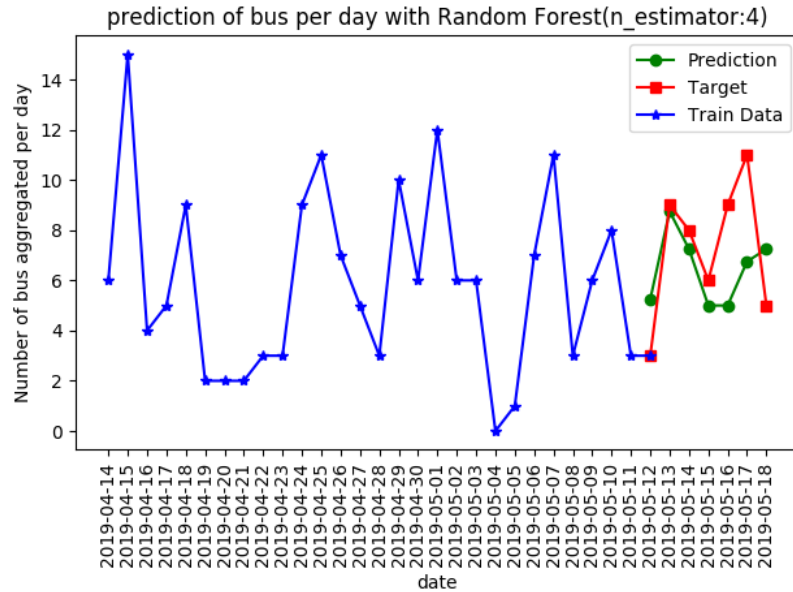


Figure 56: Prediction Results for Number of Buses with Random Forest Regression Model with Four Estimator

As we mentioned in previous sections, the lack of data for this metric makes it hard for our models to predict the last week values with good accuracy. The r2 score for the predictions of this metric, as marked in Table 20, is 0.0706.

- **Congestion**

We are interested to see the future of traffic congestion and to know how accurate our random forest regression model can predict it. We developed a univariate random forest regression model to this task, and Figure 57 shows the results.

To do the random forest, we searched for the optimal number of the estimator to predict traffic congestion and got the best results with five estimators.

In Table 21 we can see the best results that we can get with the mentioned number of

Target Values	Random Forest Regression Model Predictions
3	5.25
9	8.75
8	7.25
6	5.
9	5.
11	6.75
5	7.25
R2 Score	0.0706
MAE	1.9540
MSE	6.64158

Table 20: Prediction Results and the Performance Metrics of the Random Forest Regression Model on Total Number of Buses

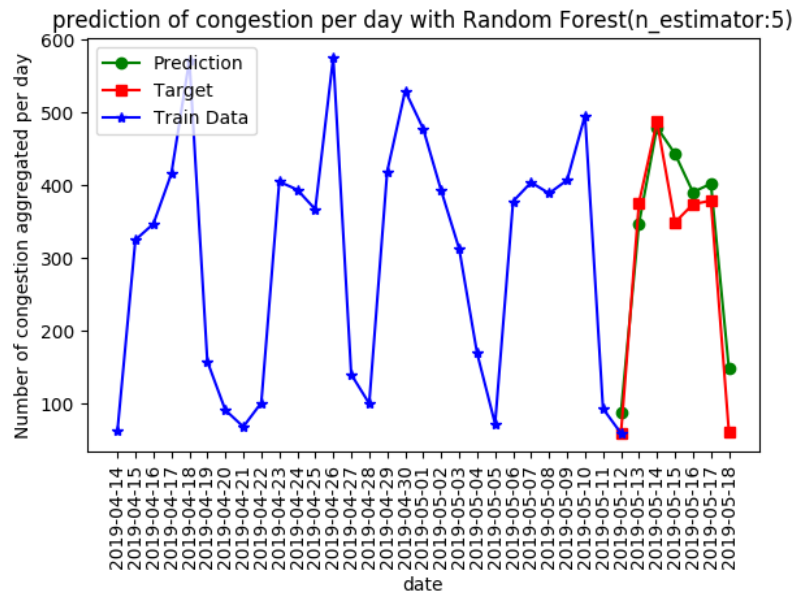


Figure 57: Prediction Results for Traffic Congestion with Random Forest Regression Model with Five Estimator

Target Values	Random Forest Regression Model Predictions
59	88.
376	347.2
488	479.8
349	443.4
374	390.6
379	402.2
61	148.
R2 Score	0.6370
MAE	76.4897
MSE	9280.6840

Table 21: Prediction Results and the Performance Metrics of the Random Forest Regression Model on Total Number of Road Traffic Congestion

estimators.

### 5.4.2 Deep Learning Models

In the previous section, we saw the results of our regression analysis models. In this section, we explore the results of the models that we created with each deep learning algorithm and see how we can fine-tune them to get better results. First, we explore the CNN results and see how they are doing on a non-image dataset. Later we discuss the results of the LSTM models and see which model is doing a better prediction job. At last, we compare the results of both algorithms and see which model and architecture did a better job predicting the future of traffic congestion.

**Convolutional Neural Network** After developing different regression analysis models with different algorithms, we developed deep learning models, starting with convolutional neural networks. We built different CNN models to do the time series forecasting prediction task, and among them, we got the best results with models in table 22. We apply each model on every metric and plot the best results.

- Personal Car



Model	Epochs	Batch size	Optimizer	Layers	Loss Function
CNN 1	100	7	Adam	2 Conv. Layer [200,100] + Flatten Layer + 1 Dense Layer[7]	MAE
CNN 2	100	7	Nadam	4 Conv. Layer[128,64,32,16] + Flatten Layer + 2 Dense Layer[50,7]	MAE
CNN 3	100	7	Nadam	Conv. Layer[64] + Dense Layer[32] + Conv. Layer[32] + Flatten Layer + Dense[32,7]	MAE

Table 22: Convolutional Neural Network Time Series Forecasting Models

We trained each model in table 22 with the first four weeks of data and compared the prediction results of the last week with the real values. Table 23 shows the predicted values for the last seven days alongside the real values and how each model performed in different performance metrics.

Based on the table 23, we can see that the CNN model 2 predictions are closer to the real values and has the highest r2 score, 0.8467, and the lowest mean absolute error and mean squared error. In figure 58, we plotted the prediction values alongside the target values to see the differences and how well the CNN model 2 performed.

- Truck

Just like personal cars, we applied the models in table 22 on the truck metric data and achieved different results. The results of each model are in table 24.

Based on the table 24, we can see that the model 3 prediction results are better than the other two models. We got the r2 score of 0.83586 and lower mean absolute and meant squared error. We plotted the prediction values alongside the target values in figure 24 to see the differences.

Target Values	CNN 1	CNN 2	CNN 3
42439	48398.324	43917.934	43221.11
54708	53836.305	48674.152	52041.65
56015	57303.23	56180.6	52528.203
57367	57829.918	56115	55370.086
58890	60760.07	56788	59579.76
60349	55988.56	58357.29	61334.508
41518	46726.92	44179.484	47506.156
R2 Score	0.75824	0.8467	0.83154
MAE	2860.2287	2240.7968	2370.7996
MSE	12541529.7113	7951186.5039	8738793.7287

Table 23: Convolutional Neural Network Time Series Forecasting Models Results with Performance Metrics for Personal Cars

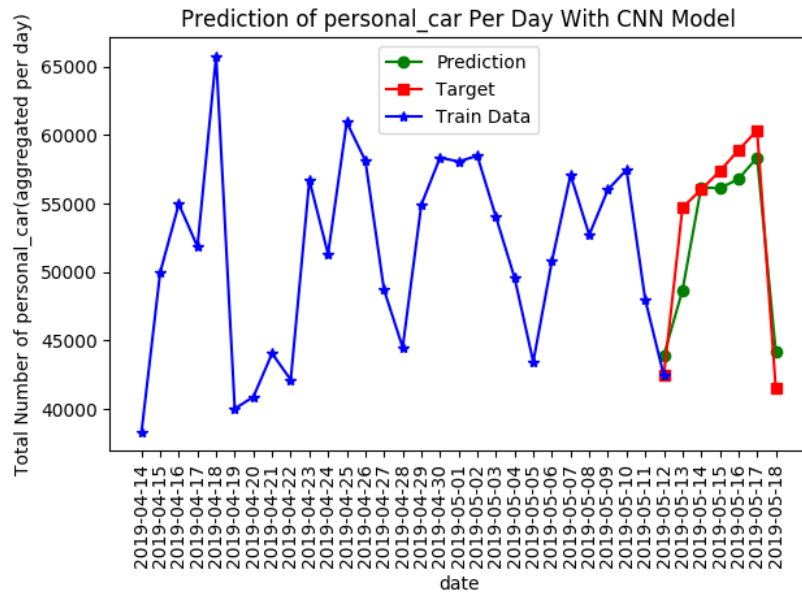


Figure 58: Prediction Results for Total Number of Personal Car with CNN Model 2

Target Values	CNN 1	CNN 2	CNN 3
430	1565.5918	892.48553	799.7568
2654	2855.138	2484.7798	3016.493
3136	2774.5	2556.2808	3255.0613
2240	2412.0923	2136.1108	2486.5923
2347	1929.9503	1991.7736	2094.0676
2824	2482.4749	2304.5706	2187.9421
621	1613.8171	1066.2959	1192.1664
R2 Score	0.6009	0.8292	0.83586
MAE	517.3877	376.4665	365.4371
MSE	395225.5626	169096.7016	162554.279

Table 24: Convolutional Neural Network Time Series Forecasting Models Results with Performance Metrics for Trucks

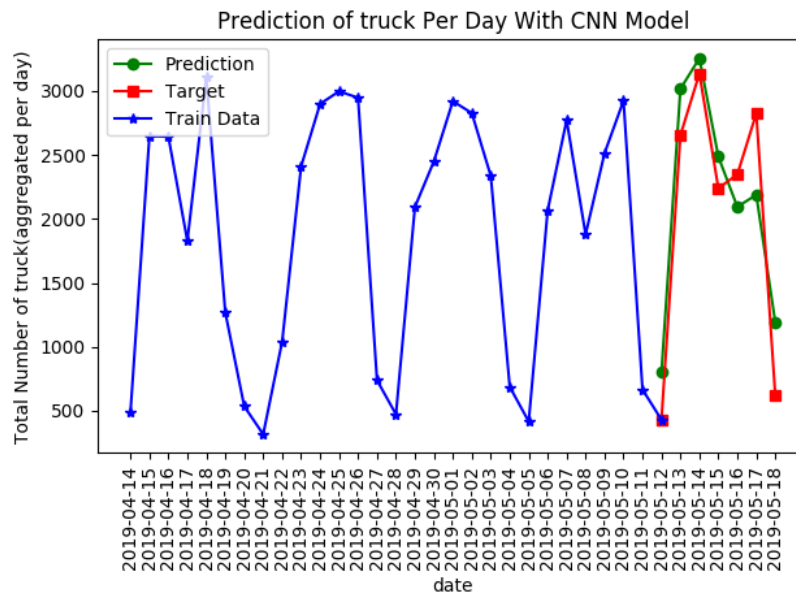


Figure 59: Prediction Results for Total Number of Trucks with CNN Model 3

Target Values	CNN 1	CNN 2	CNN 3
3	6.0576844	2.3426623	4.0255723
9	8.356019	2.2464995	6.8501797
8	7.265272	5.351576	7.0361266
6	7.900469	3.3484209	7.0700936
9	9.282643	8.645057	11.856122
11	8.394331	7.023836	10.845278
5	6.6095595	4.240264	6.0598717
R2 Score	0.4854	-0.6861	0.6246
MAE	1.5478	2.54309	1.3257
MSE	3.3394	10.9428	2.4360

Table 25: Convolutional Neural Network Time Series Forecasting Models Results with Performance Metrics for Buses

- Bus

After the total number of personal cars and trucks, we applied the same method on the number of buses as well. The results for each model can be in table 25.

We got the best results for the total number of buses with CNN model 3. The r2 score is 0.6246, which is better than other models but much lower than other metrics. We explained in the previous section that since we do not have enough data for the total number of buses, the prediction task error for this metric is bigger. We plotted the prediction results alongside the target values in figure 59.

- Congestion

The last metric is congestion. We applied all the three models in table 22 on this metric as well to see which model is performing better. The results can be seen in table 26.

Based on table 26, we can see that CNN model 1 is performing better than other models. The r2 score for this model is 0.9183, which is good. We plotted the prediction values alongside the target values in figure 61 to see how close these values are.

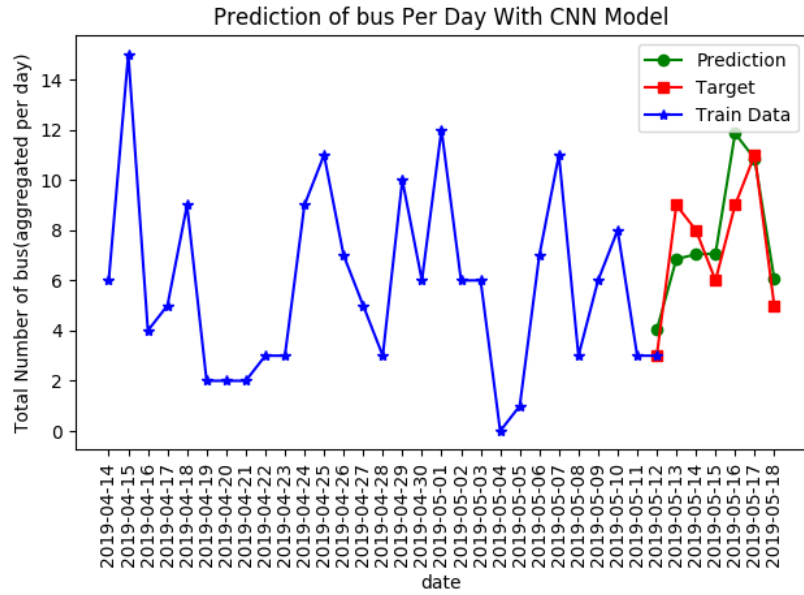


Figure 60: Prediction Results for Total Number of Buses with CNN Model 3

Target Values	CNN 1	CNN 2	CNN 3
59	121.04735	74.44145	156.02776
376	346.10715	312.85147	290.31763
488	467.24347	413.51785	536.63275
349	335.15387	398.70264	366.04407
374	305.8225	342.00693	376.00873
379	316.65878	325.32727	392.27982
61	56.429523	42.458046	87.69699
R2 Score	0.91830	0.9032	0.8808
MAE	37.3760	43.8546	41.4817
MSE	1988.7845	2356.0343	2900.6595

Table 26: Convolutional Neural Network Time Series Forecasting Models Results with Performance Metrics for Road Traffic Congestion

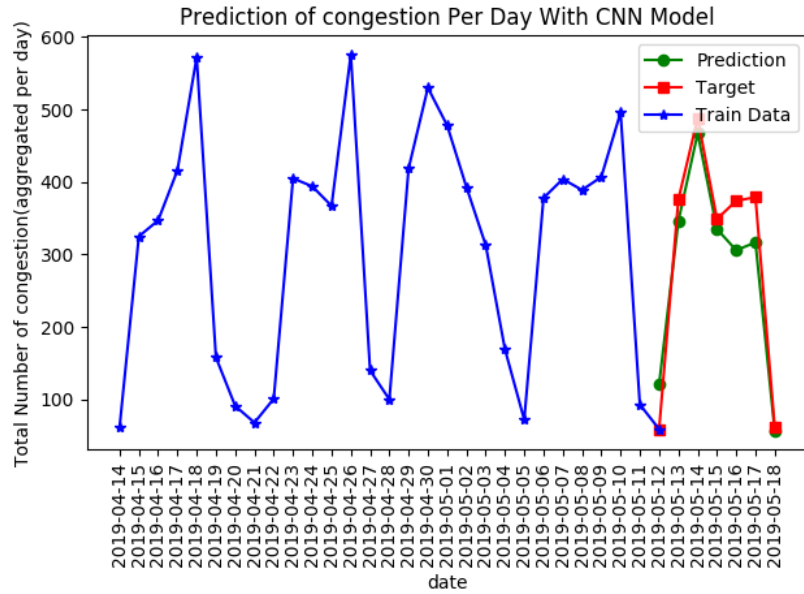


Figure 61: Prediction Results for Total Number of Road Traffic Congestion with CNN Model 1

**Long Short Term Memory** In this section, we explore the results for our LSTM models on different features. Like CNN, we start with personal cars and explore different model architectures see how each of these models is performing. We then continue with trucks, buses, and congestion, plot the predictions, and select the best model to do the prediction task.

As we explained in the Methodology chapter, we explored and tested different architectures to find the best approach to predict each feature. After exploring and fine-tuning many models, we built four models. The optimizer for all of these models is Nadam (Dozat, 2016), the loss function is mean absolute error, the batch size is seven, the epochs are 75, and the activation function for each layer is ReLU (Nair and Hinton, 2010). Table 27 Shows the univariate time series forecasting models that we are going to use to predict the total number of personal cars, trucks, buses, and congestion.

- Personal Car

We trained each model in table 27 with the first four weeks of data and compared

Model	Epochs	Batch size	Optimizer	Layers	Loss Function
LSTM 1	75	7	Nadam	3 LSTM [100,100,100] + 1 Dense[7]	MAE
LSTM 2	75	7	Nadam	3 LSTM[128,64,32] + 1 Dense[7]	MAE
LSTM 3	75	7	Nadam	3 LSTM[128,64,32] + 2 Dropout Layer[0.02,0.02] + 1 Dense[7]	MAE
LSTM 4	75	7	Nadam	3 LSTM + 1 Dropout Layer[0.02] + 1 Dense[7]	MAE

Table 27: Long Short Term Memory Time Series Forecasting Models

Target Values	LSTM 1	LSTM 2	LSTM 3	LSTM 4
42439	44807.617	48204.1	49675.98	46394.08
54708	52216.7	50689.684	50914.344	52712.543
56015	56359.668	58275.98	57033.465	57755.96
57367	56488.125	59633.996	58351.098	58989.53
58890	57204.95	60618.074	58451.	59452.062
60349	55756.273	57522.793	55494.63	56484.707
41518	46347.8	51821.566	51922.785	52097.254
R2 Score	0.8348	0.5132	0.44707	0.5801
MAE	2455.86	4167.03	4104.47	3474.23
MSE	8566807.60	25253104.87	28684089.5352	21779618.85

Table 28: Long Short Term Memory Time Series Forecasting Models Results and Performance Metrics for Total Number of Personal Cars

the predicted results of the last week with the real values. Table 28 shows the predicted values for the last seven days alongside the real values and how each model performed calculated with R2 score, MAE, and MSE.

Based on the table 28 we got the best results with LSTM model 1. The r2 score is 0.8348, which is better in comparison to other models. We plotted the prediction values alongside the target values in figure 62 to see how close these values are.

- Truck

After the personal cars, we applied all the LSTM models in table 27 on the total

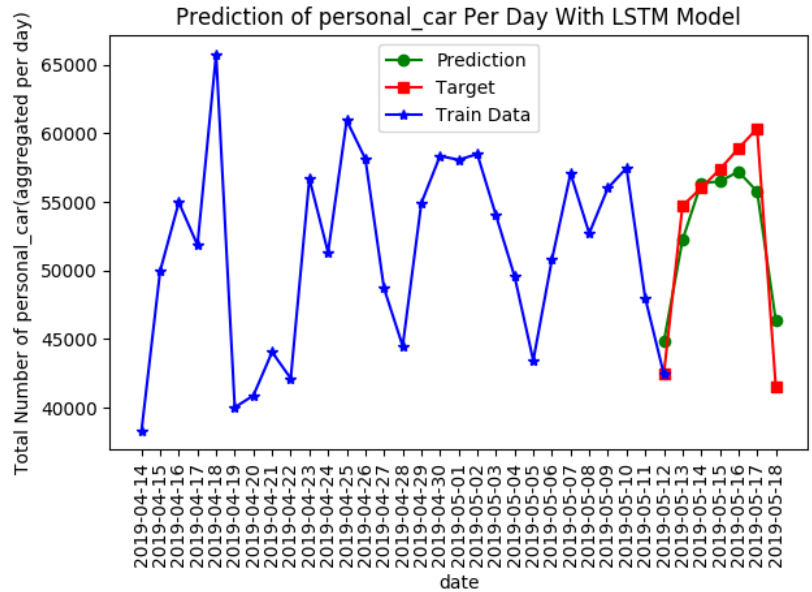


Figure 62: Prediction Results for Total Number of Personal Cars with LSTM Model 1

number of trucks and predicted future values. Table 29 shows the predicted values for the last seven days alongside the real values and how each model performed.

We got the best results with LSTM model 3. The r2 score is 0.9039, which is close to LSTM model 2 and better than other models. Figure 63 shows how close the predicted values are to the target values.

- Bus

After the total number of trucks, we fit the LSTM models in table 27 on our bus metric. Since there is a lack of data for this metric, we are interested to see how LSTM models are performing. Table 30 shows the predicted values for the last seven days alongside the real values and how each model performed.

As presented in table 30, we got the best results with LSTM model 4. The other models showed poor performance, and interestingly, LSTM model 4 has much simpler architecture. We managed to get better results with CNN for this metric. Figure 64 shows how close the predicted values are to the target values.



Target Values	LSTM 1	LSTM 2	LSTM 3	LSTM 4
430	718.4146	729.79236	306.83942	837.06067
2654	2608.4116	2537.8638	2228.8328	1662.4645
3136	2944.561	2955.7976	2642.8774	2435.2598
2240	2995.934	2734.8108	2578.45	1996.3715
2347	3119.7402	2670.6091	2645.5342	1683.9783
2824	3013.2131	2389.7944	2809.3508	1885.3081
621	774.1186	703.9281	772.5591	676.5154
R2 Score	0.8053	0.9017	0.90393	0.5639
MAE	342.34	275.95	263.5203	571.4562
MSE	192817.26	97258.30	95137.3350	431865.0092

Table 29: Long Short Term Memory Time Series Forecasting Models Results and Performance Metrics for Total Number of Trucks

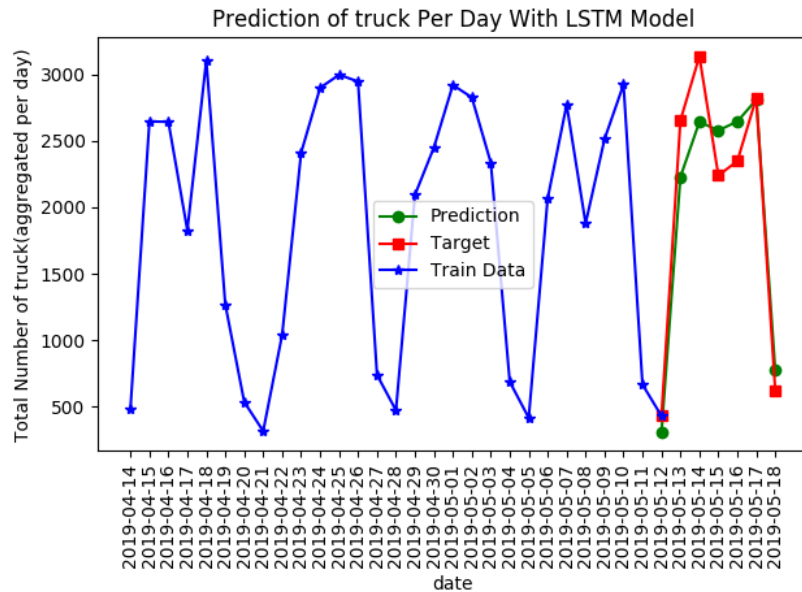


Figure 63: Prediction Results for Total Number of Trucks with LSTM Model 2

Target Values	LSTM 1	LSTM 2	LSTM 3	LSTM 4
3	2.74	5.351	3.803	5.083
9	0.42	3.821	5.113	6.468
8	5.11	6.614	5.064	6.565
6	9.3	5.366	4.207	6.922
9	7.98	5.714	9.883	8.143
11	6.11	6.014	3.240	7.808
5	6.37	8.428	3.467	5.519
R2 Score	-1.6428	-0.8066	-1.0012	0.4530
MAE	3.1929	3.0355	2.7990	1.6485
MSE	17.15	11.7245	12.9880	3.5497

Table 30: Long Short Term Memory Time Series Forecasting Models Results and Performance Metrics for Total Number of Buses

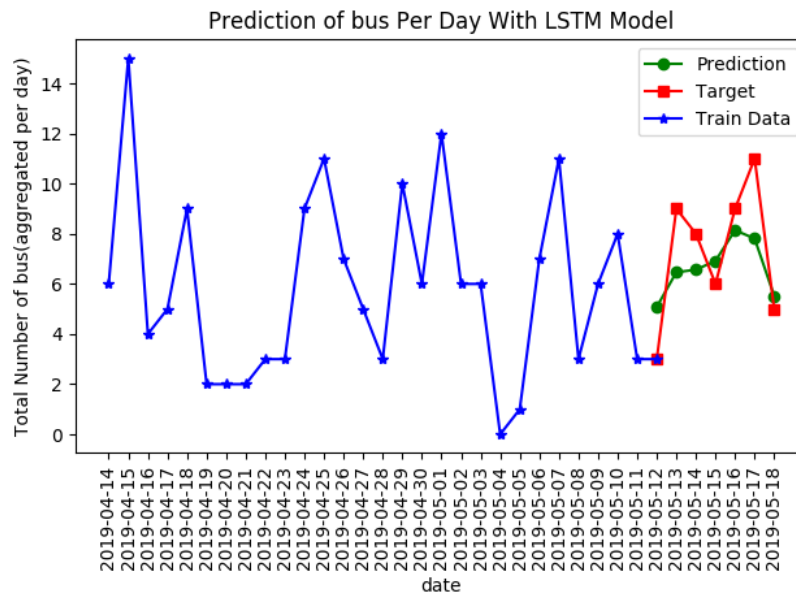


Figure 64: Prediction Results for Total Number of Buses with LSTM Model 4

Target Values	LSTM 1	LSTM 2	LSTM 3	LSTM 4
59	120.44719	133.25793	43.595146	196.46846
376	293.75287	334.33902	367.57687	344.3339
488	412.9298	485.76373	482.43088	420.79974
349	377.2233	420.43713	387.16214	343.81744
374	388.82367	361.11984	364.10248	391.03378
379	331.50214	392.96768	325.36612	302.96353
61	110.91818	95.95277	37.219933	176.62398
R2 Score	0.8712	0.9181	0.9686	0.7424
MAE	51.31	35.9132	22.1243	64.3159
MSE	3134.29	1991.54	762.2383	6269.0931

Table 31: Long Short Term Memory Time Series Forecasting Models Results and Performance Metrics for Total Number of Road Traffic Congestion

- Congestion

The last metric is traffic congestion, and like other metrics, we applied the LSTM models in table 27 and compared the results. In table 31, we can see all the real values and predicted values alongside the performance of each model.

We got the best results with LSTM model 3. The r2 score is 0.9686, which is better than the other models and much better than the results we got from CNN models.

Figure 65 shows how close the predicted values are to the target values.

### 5.4.3 Time Series Forecasting Models

In previous sections, we developed different regression and deep learning algorithms and tried different architectures to achieve better results. In this section, we are interested to see the results of the most popular time series forecasting libraries, ARIMA and Prophet, and compare them with each other and previous results from different algorithms. Each library has different parameters and coefficients, and they are tweakable in their ways. We start with ARIMA and explore different variations of input to get better results with our dataset. Later we try to build different prophet models and see how each parameter can affect the

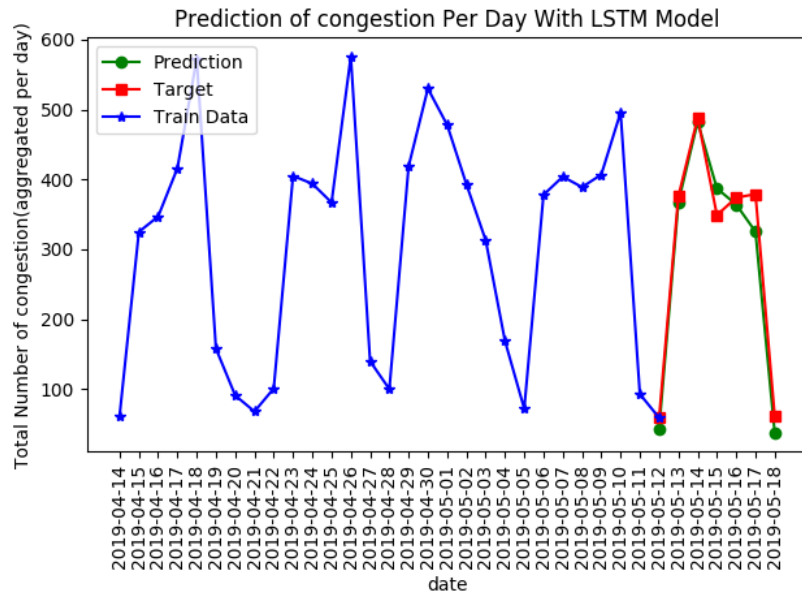


Figure 65: Prediction Results for Total Number of Road Traffic Congestion with LSTM Model 3

results.

**ARIMA** In previous sections, we discussed time series forecasting libraries and explored ARIMA and its functionalities. In this section, we develop a seasonal ARIMA model and see how good the model is performing on our dataset.

To build the model, we used the pmdarima (ARIMA, 2019) library, which finds the best combination of the values automatically for your model. One of the most important values that we manually set for the model is the period for seasonal differencing, which based on the visualizations from the data and based on the results of different ARIMA models, this value should be set to weakly seasonal.

We Trained the model with first four weeks of the data and did the prediction for next seven days for each metric. Later we compared the results with the target values and see how well the model is performing on each metric.

- Personal Car Like other approaches we start with the total number of personal cars

Target Values	ARIMA
42439	47655.956243
54708	56661.524226
56015	61062.380681
57367	58935.143889
58890	63031.255085
60349	61815.507265
41518	55821.432691
R2 Score	0.22107
MAE	4813.885
MSE	40408120.046

Table 32: ARIMA Model Results and Performance Metrics for Total Number of Personal Cars

and fit the model on this metric. In table 32 we can see the results of the ARIMA model compared to the target values.

Based on table 32 we can see that r2 score of the model for this metric is around 0.22 which is really low in comparison to other models. In figure 66 we can see that the predicted values are not really close to the target values.

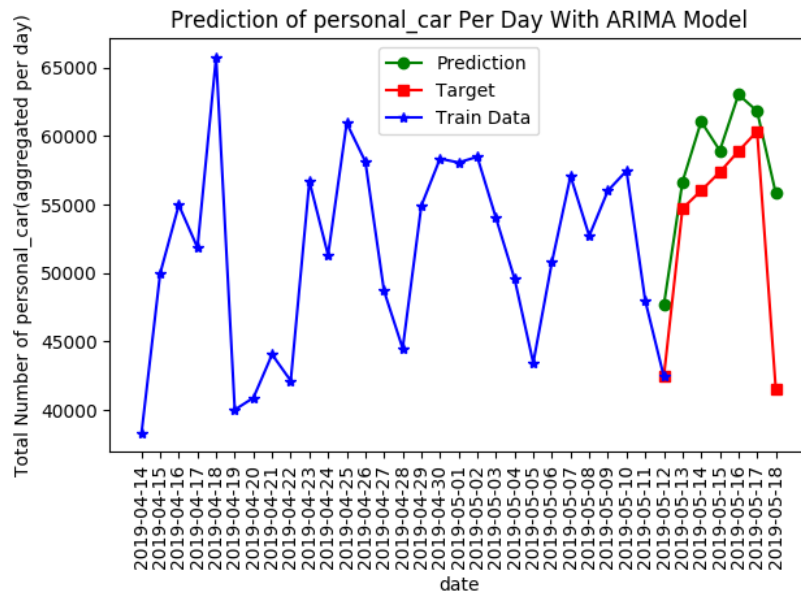


Figure 66: Prediction Results for Total Number of Personal Cars with ARIMA Model

Target Values	ARIMA
430	566.762161
2654	2010.346487
3136	2605.862798
2240	2452.439853
2347	2871.606039
2824	2367.612787
621	729.875989
R2 Score	0.81904
MAE	373.2659
MSE	179217.80612

Table 33: ARIMA Model Results and Performance Metrics for Total Number of Trucks

We applied the same model on the other metrics to see if the results are better.

- Truck We applied the same model on the total number of trucks and predicted next seven days and compare them with the target values. The results can be seen in table 33.

The r2 score is around 0.82 which is much better than the results of this model on personal car. We plotted the results in figure 67 and we can see that the predicted value are much closer to the target values in comparison to the total number of personal cars.

- Bus

We applied the model on the bus metric and we compared the results with the target value. The results are in table 34 and we can see that for this metric the ARIMA model is relatively doing better than other regression and deep learning models.

In table 34 we can see that r2 score is 0.60. The prediction results can be seen in figure 68. Although the prediction are not close to the target values, we can see that ARIMA model did a relatively good job predicting the next seven days of traffic for total number of buses.

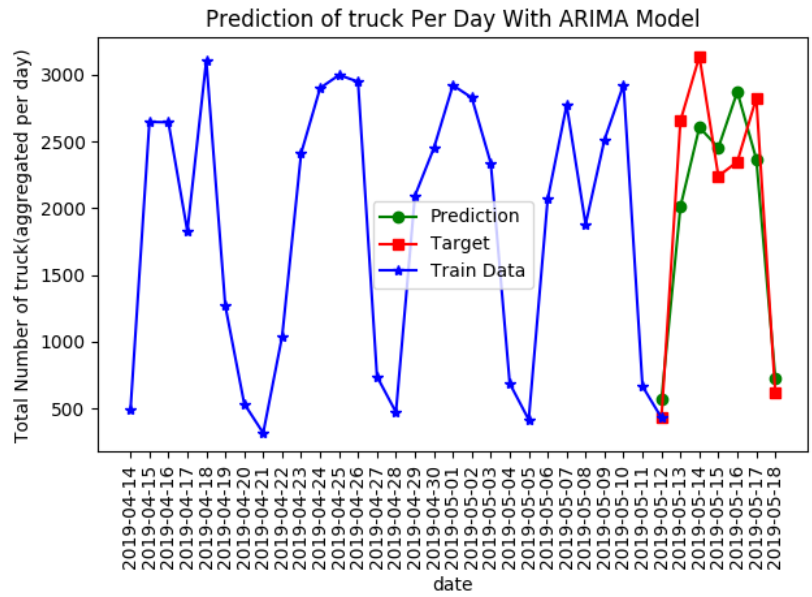


Figure 67: Prediction Results for Total Number of Trucks with ARIMA Model

Target Values	ARIMA
3	3.143433
9	9.146315
8	7.223554
6	8.167135
9	9.087948
11	7.530667
5	4.268439
R2 Score	0.60552
MAE	1.07459
MSE	2.56007

Table 34: ARIMA Model Results and Performance Metrics for Total Number of Buses

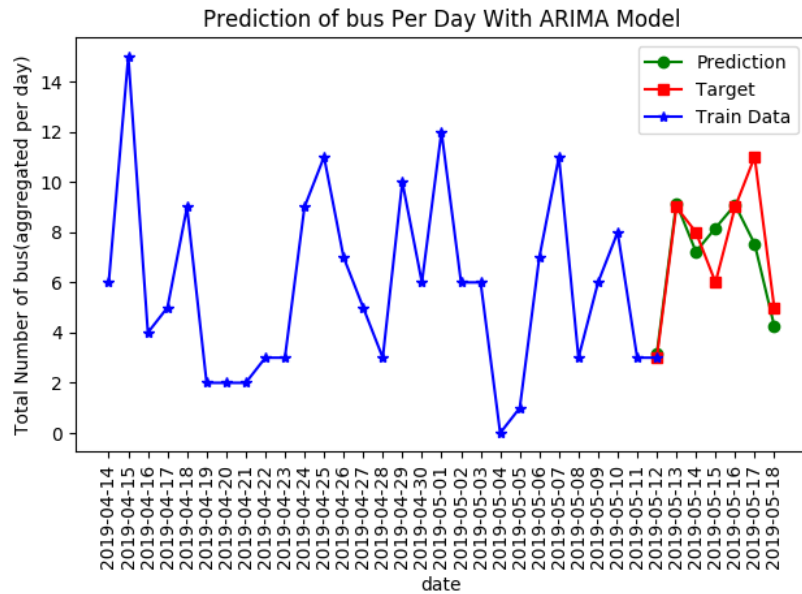


Figure 68: Prediction Results for Total Number of Buses with ARIMA Model

- Congestion The last metric is traffic congestion and applied the same ARIMA model on this metric as well. The results can be seen in table 35.

The r2 score for this metric with ARIMA model is around 0.90 which is better than the results of this model for other metrics but still not as accurate as deep learning methods. The predictions can be seen in figure 69 alongside the target values.

**Prophet** We explored the prophet model and discussed the parameters of this model. In this section we are interested to see how a Prophet model can perform on our dataset. Training a Prophet model is relatively easy and it can be done by calling the model in your code and set the parameters in a way that results in best accuracy. Like other models we train the Prophet model on the first four weeks of the dataset and predict the next seven days and compare the results with the target values. We will apply the same model on every metric and see how it is performing.

- Personal Car After building the prophet model we applied it on the total number of personal cars and managed to make the predictions for the fifth week of the data. In



Target Values	ARIMA
59	84.187179
376	371.954381
488	442.974812
349	417.394477
374	340.063760
379	288.217361
61	87.308903
R2 Score	0.8976
MAE	41.95432
MSE	2491.592490

Table 35: ARIMA Model Results and Performance Metrics for Total Number of Road Traffic Congestion

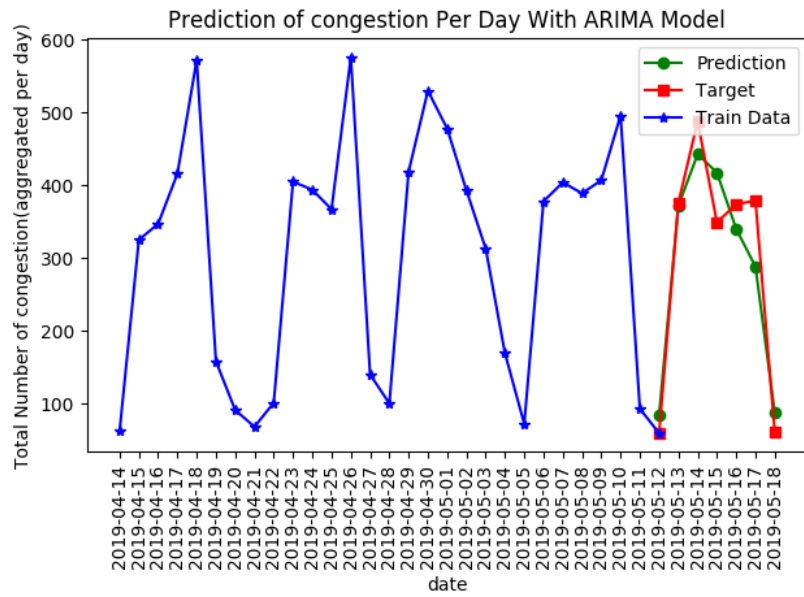


Figure 69: Prediction Results for Total Number of Road Traffic Congestion with ARIMA Model

Target Values	Prophet
42439	45242.8911
54708	53197.70756
56015	59317.97528
57367	56960.99031
58890	62711.86704
60349	56695.41250
41518	48433.4466
R2 Score	0.73289
MAE	3202.0099
MSE	13856578.6167

Table 36: Prophet Model Results and Performance Metrics for Total Number of Personal Cars

table 36 we can see the prediction values and performance of the model.

Unlike the ARIMA model, the Prophet model is predicting the next seven days of the data with better accuracy and the predicted values are much close to the target values. The r2 score of the Prophet model is 0.732 and we can see how model is performing in figure 70.

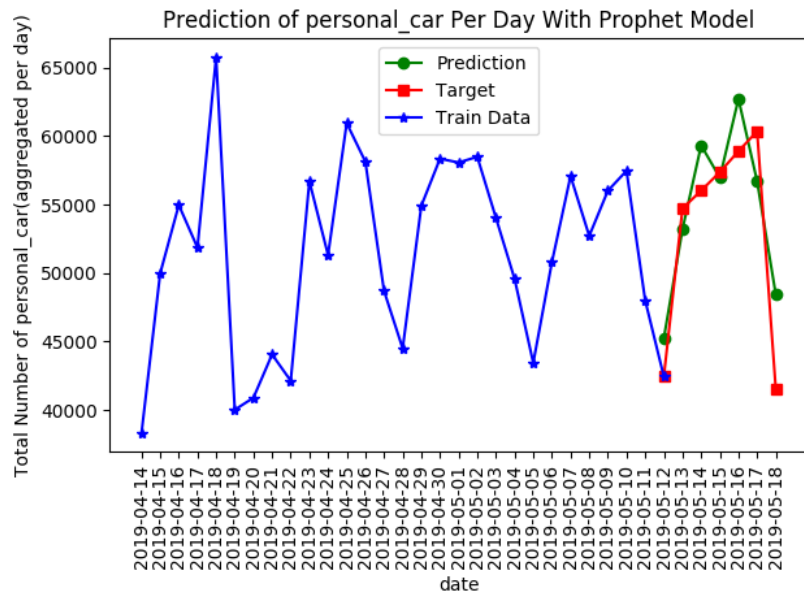


Figure 70: Prediction Results for Total Number of Personal Cars with Prophet Model

Target Values	Prophet
430	563.1236
2654	2239.3212
3136	2821.2456
2240	2491.92471
2347	2896.8353
2824	2598.50928
621	787.8648
R2 Score	0.89423
MAE	293.8103
MSE	104746.51691

Table 37: Prophet Model Results and Performance Metrics for Total Number of Trucks

- Truck

We applied the same Prophet model on the total number of trucks and predicted the values for the next seven days. We can see the results for the last week of the data alongside the predictions and performance metrics in table 37.

Based on the table 37 the r2 score is around 0.90 which it means that predictions are close to the target values and the it can be seen in the figure 71.

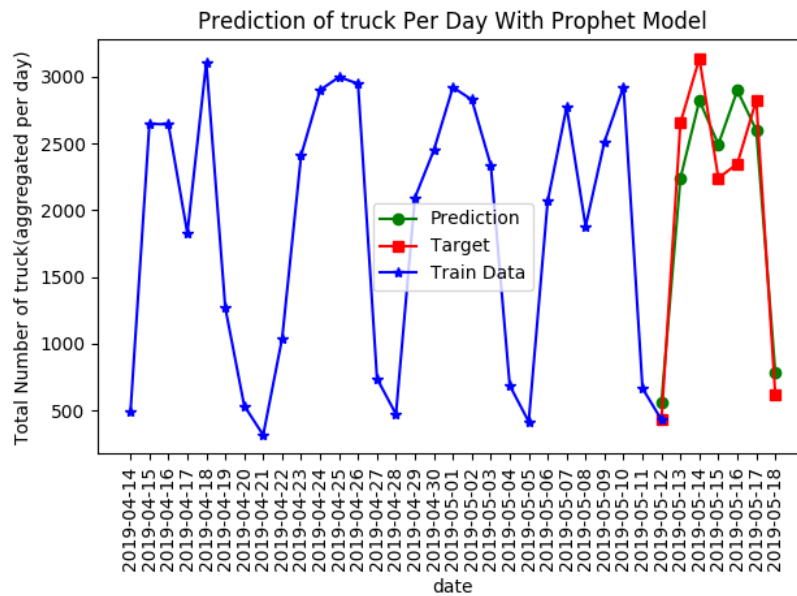


Figure 71: Prediction Results for Total Number of Trucks with Prophet Model

Target Values	Prophet
3	3.64011
9	9.44791
8	7.044762
6	7.645617
9	8.8472
11	7.44545
5	3.64043
R2 Score	0.587540
MAE	1.25081
MSE	2.676777

Table 38: Prophet Model Results and Performance Metrics for Total Number of Buses

- Bus We applied the same prophet model on the bus metric and predicted the next seven days. The predictions are in table 38 alongside the target values and performance metrics.

Based on the table 38 we can see that the r2 score of the Prophet model for this metric is 0.587. In comparison to other metrics the r2 score is low but in comparison to other models for this metric Prophet is giving us relatively good results. The predictions are in table 72 alongside the target values and performance metrics.

- Congestion The last metric is congestion and we applied the same model on this metric as well. The results of the time series prediction task for last week can be seen in table 39 alongside real values and performance metrics.

Based on the table 39 we can see that model has the r2 score of 0.8891 which is really close to the ARIMA model. The predictions are in table 38 alongside the target values and performance metrics.

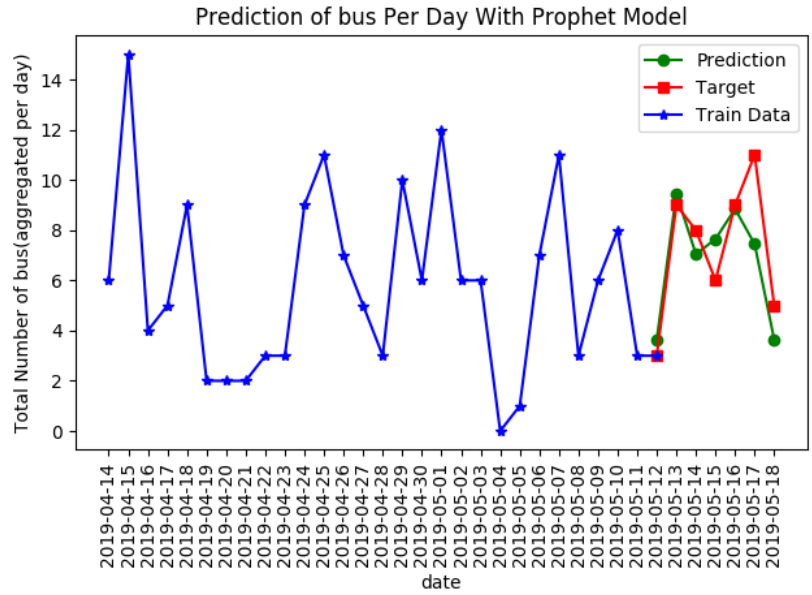


Figure 72: Prediction Results for Total Number of Buses with Prophet Model

Target Values	Prophet
59	90.52057
376	338.11567
488	453.11865
349	423.52934
374	440.93801
379	402.75042
61	129.1773
R2 Score	0.88913
MAE	48.24019
MSE	2699.0037

Table 39: Prophet Model Results and Performance Metrics for Total Number of Road Traffic Congestion

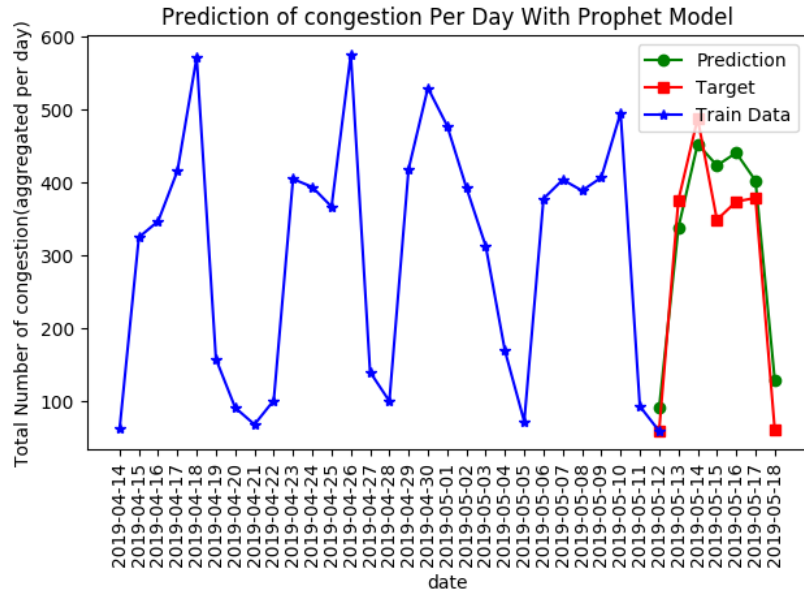


Figure 73: Prediction Results for Total Number of Road Traffic Congestion with Prophet Model

## 5.5 Conclusion

In the previous chapter, we described in detail each regression and deep learning model and how to implement them. In this chapter, we developed the models and generated the results and compared them with each other for each metric to see which models are doing a better job in predicting the last week of the data.

Based on the results in the regression analysis section, we discovered that the random forest regression is better than the other regression model since the predicted values were closer to the target values based on the  $r^2$  scores we achieved with this model.

In the deep learning section, we discovered that the LSTM model is doing a better job of predicting the congestion metric. On the other hand, a CNN model can predict the last week of the data for the bus metric with closer values to the target values based on the  $r^2$  score we get from these models.

In the last section, we build two models with time series forecasting libraries and predicted the values for the last seven days of the dataset fro each metrics and compared the

results with target values. We saw a significant difference in the total number of personal car predictions between two libraries, and the Prophet model managed to achieve better results. On the other hand, we got almost similar results in other metrics.

# Chapter 6

## Conclusion and Future Works

### 6.1 Conclusion

In this thesis, we explored the idea of predicting traffic conditions by collecting and analyzing image data. We collected more than 400,000 images from Quebec511(Quebec511, 2019) website for five weeks from multiple traffic cameras with different weather and light conditions.

After cleaning and preprocessing the data, we built an object detector and classifier to detect different types of vehicles in each image with a state-of-the-art algorithm, Faster RCNN. We managed to reduce the error rate for both detection and classification tasks by building a robust model to different conditions and tried to increase the accuracy by introducing blurry, rainy, and low light images to the model.

Furthermore, we created a dataset based on the information in each image. We cleaned and preprocessed the data to reduce the noise and make sure that the data is valid. After analyzing and visualizing the dataset, we find out the most congested areas, road traffic behavior in each day, peak hours, and the contribution of each type of vehicle to the traffic condition. We noticed a weekly seasonal behavior of the number of vehicles for each type of vehicle and the same trends for road traffic congestion as well. The dataset indicates



that Saturdays and Sundays are the least congested, and on the other hand, Tuesdays and Wednesdays are the most crowded days of the week. We discovered that personal cars are contributing the most to the traffic congestion. The morning peak hour for both of these metrics is between 7 am, and 8 am, and in the afternoon is around 4 pm. Top areas with most personal cars recorded are Boulevard Sainte-Croix, Rue Stinson, and Avenue Deslauriers. For the trucks, we detected that Boulevard Sainte-Croix, Rue de Marche-Central, and Rue Stinson are top areas with the most recorded number of this type of vehicle. We saw a different behavior of peak hours for trucks in comparison to personal cars and congestion having the total number of trucks start increasing around 4 am and a 6 am morning peak hour. Although we need to do more investigation for the buses, we also find out that Boulevard Lacordaire, Boulevard Cavendish, and Gold de l'Île de Montreal are the areas with the most amount of the number of recorded buses.

Moreover, we build multiple Regression and Deep Learning, time series forecasting models to predict future traffic situation and compare the performance of each model and see which model is doing a better prediction job for each metric. We discovered that the deep learning models could capture the behavior of the data better than other algorithms and make more accurate predictions for all of the metrics. For the total number of personal cars, we managed to predict the coming seven days with 84% accuracy using a CNN model. We got the best results for the total number of buses with another CNN model with 62% accuracy due to a lack of data. For the total number of trucks and congestion incidents, we got the best prediction results with LSTM models. For the trucks, the accuracy of the prediction for seven days in the future was 90%, and for the traffic congestion was 96%. Since the hyper-parameters can affect the results of the prediction, we will explore the idea of hyper-parameter optimization, and we will be able to improve these models by setting the optimal and fine-tuned values.

## 6.2 Future Works

We already built a pipeline to get the data automatically, preprocess the data, apply object detection and classification, create the dataset, and predict future demand based on the historical data. For the next steps, we would like to collect image data from the entire island of Montreal for at least one year to have a more rich dataset. By obtaining more data following objectives could be achieved as well:

- Data Collection
  - Get the video feeds of each camera
  - Improve the quality of the images by either getting the source data or using the state-of-the-art approaches
  - Use camera footage of other cities to implement transfer learning
- Object Detection and Classification Models
  - New class for the object detection model to able to detect electric cars
  - Improve the object detection and classification model to be able to track the objects and calculate the speed of each type of vehicle
  - Utilize more computational power to do the object detection and classification task real-time or near real-time
- Time Series Forecasting Models
  - Build more models by trying new algorithms and approaches
  - Implement a hyper-parameter optimization method to find the most optimized combination automatically
  - Apply attention mechanism on deep learning models to achieve better results

- Explore the idea of generating data by utilizing approaches like Generative Adversarial Networks
- Build an on-line pipeline to feed the real-time data back to each model and update the detection and prediction results

# Bibliography

- S Agatonovic-Kustrin and R Beresford. Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research. *Journal of pharmaceutical and biomedical analysis*, 22(5):717–727, 2000.
- Dave Anderson and George McNeill. Artificial neural networks technology. *Kaman Sciences Corporation*, 258(6):1–83, 1992.
- ARIMA. Auto ARIMA for python. <http://alkaline-ml.com/pmdarima/>, 2019. Accessed: 2019-11-13.
- Shashank Bharadwaj, Sudheer Ballare, Chandel MK Rohit, and MK Chandel. Impact of congestion on greenhouse gas emissions for road transport in mumbai metropolitan region. *Transportation Research Procedia*, 25:3538–3551, 2017.
- Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.
- George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- Yeong-Hyeon Byeon and Keun-Chang Kwak. A performance comparison of pedestrian detection using faster rcnn and acf. In *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 858–863. IEEE, 2017.
- Chengtao Cai, Boyu Wang, and Xin Liang. A new family monitoring alarm system based on improved yolo network. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 4269–4274. IEEE, 2018.
- Peng Chen, Hongyong Yuan, and Xueming Shu. Forecasting crime using the arima model. In *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, volume 5, pages 627–630. IEEE, 2008.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

- Zhen Dong, Yuwei Wu, Mingtao Pei, and Yunde Jia. Vehicle type classification using a semisupervised convolutional neural network. *IEEE transactions on intelligent transportation systems*, 16(4):2247–2256, 2015.
- Timothy Dozat. Incorporating nesterov momentum into adam. 2016.
- Gilles Duranton and Matthew A Turner. The fundamental law of road congestion: Evidence from us cities. *American Economic Review*, 101(6):2616–52, 2011.
- Quanfu Fan, Lisa Brown, and John Smith. A closer look at faster r-cnn for vehicle detection. In *2016 IEEE intelligent vehicles symposium (IV)*, pages 124–129. IEEE, 2016.
- Carolina Garcia-Ascanio and Carlos Maté. Electric power demand forecasting using interval time series: A comparison between var and implp. *Energy Policy*, 38(2):715–725, 2010.
- Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- Hua Gong, Yong Zhang, Ke Xu, and Fang Liu. A multitask cascaded convolutional neural network based on full frame histogram equalization for vehicle detection. In *2018 Chinese Automation Congress (CAC)*, pages 2848–2853. IEEE.
- Google-Maps. Google maps. <https://www.google.ca/maps/>, 2019. Accessed: 2019-11-13.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- Shih-Chung Hsu, Chung-Lin Huang, and Cheng-Hung Chuang. Vehicle detection using simplified fast r-cnn. In *2018 International Workshop on Advanced Image Technology (IWAIT)*, pages 1–3. IEEE, 2018.

- Jonathan Huang, Vivek Rathod, Derek Chow, Chen Sun, Menglong Zhu, Matthew Tang, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, Kovalevskiy Viacheslav Uijlings, Jasper and, and Kevin Murphy. Github repository of tensorflow object detection api. [https://github.com/tensorflow/models/tree/master/research/object\\_detection](https://github.com/tensorflow/models/tree/master/research/object_detection), 2019. Accessed: 2019-11-13.
- David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.
- Huaizu Jiang and Erik Learned-Miller. Face detection with the faster r-cnn. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 650–657. IEEE, 2017.
- Kazuya Kawakami. Supervised sequence labelling with recurrent neural networks. *Ph. D. thesis*, 2008.
- Keras. Keras: The python deep learning library. <https://keras.io/>, 2019. Accessed: 2019-11-13.
- Janaki Koirala et al. Food object recognition: An application of deep learning. 2018.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
- PIERRE LEGENDRE. Model ii regression user's guide, r edition. *R Vignette*, 14, 1998.
- Jonathan I Levy, Jonathan J Buonocore, and Katherine Von Stackelberg. Evaluation of the public health impacts of traffic congestion: a health risk assessment. *Environmental health*, 9(1):65, 2010.
- Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.
- Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1975–1981. IEEE, 2010.
- Matplotlib. Matplotlib website. <https://matplotlib.org/>, 2019. Accessed: 2019-11-13.
- Wanli Min and Laura Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606–616, 2011.

- Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- NOAA. National centers for environmental information. <https://www.ncdc.noaa.gov/>, 2019. Accessed: 2019-11-13.
- Numpy. Numpy website. <https://numpy.org/>, 2019. Accessed: 2019-11-13.
- Ping-Feng Pai and Chih-Sheng Lin. A hybrid arima and support vector machines model in stock price forecasting. *Omega*, 33(6):497–505, 2005.
- JC Palomares-Salas, JGG De La Rosa, JG Ramiro, J Melgar, A Aguera, and A Moreno. Arima vs. neural networks for wind speed forecasting. In *2009 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pages 129–133. IEEE, 2009.
- Bei Pan, Ugur Demiryurek, and Cyrus Shahabi. Utilizing real-world transportation data for accurate traffic prediction. In *2012 IEEE 12th International Conference on Data Mining*, pages 595–604. IEEE, 2012.
- Pandas. Pandas website. <https://pandas.pydata.org/>, 2019. Accessed: 2019-11-13.
- Markos Papageorgiou, Christina Diakaki, Vaya Dinopoulou, Apostolos Kotsialos, and Yibing Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12): 2043–2067, 2003.
- Polynomial-Regression. Polynomial regression, daniel borcard, département de sciences biologiques, université de montréal. [http://biol09.biol.umontreal.ca/PLcourses/Polynomial\\_regression.pdf](http://biol09.biol.umontreal.ca/PLcourses/Polynomial_regression.pdf), 2019. Accessed: 2019-11-26.
- Quebec511. Québec 511 website. <https://www.quebec511.info/>, 2019. Accessed: 2019-11-13.
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- Astrid Schneider, Gerhard Hommel, and Maria Blettner. Linear regression analysis: part 14 of a series on evaluation of scientific publications. *Deutsches Ärzteblatt International*, 107(44):776, 2010.

- Scikit-learn. scikit-learn: Machine learning in python. <https://scikit-learn.org/>, 2019. Accessed: 2019-11-13.
- Seaborn. Seaborn: statistical data visualization. <https://seaborn.pydata.org/>, 2019. Accessed: 2019-11-13.
- Yantai Shu, Zhigang Jin, Lianfang Zhang, Lei Wang, and Oliver WW Yang. Traffic prediction using farima models. In *1999 IEEE International Conference on Communications (Cat. No. 99CH36311)*, volume 2, pages 891–895. IEEE, 1999.
- Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401. IEEE, 2018.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- Daniel Soutner and Luděk Müller. Application of lstm neural networks in language modelling. In *International Conference on Text, Speech and Dialogue*, pages 105–112. Springer, 2013.
- Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
- Lei Tai and Ming Liu. Deep-learning in mobile robotics-from perception to control systems: A survey on why and why not. *arXiv preprint arXiv:1612.07139*, 2016.
- Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.
- Tensorflow. Tensorflow website. <https://www.tensorflow.org/>, 2019. Accessed: 2019-11-13.
- TomTom. Tomtom traffic index. [https://www.tomtom.com/en\\_gb/traffic-index/ranking/?country=CA, MX, US](https://www.tomtom.com/en_gb/traffic-index/ranking/?country=CA, MX, US), 2019. Accessed: 2020-02-26.
- Fang-Mei Tseng and Gwo-Hshiung Tzeng. A fuzzy seasonal arima model for forecasting. *Fuzzy Sets and Systems*, 126(3):367–376, 2002.
- Tzutalin. Labelimg. <https://github.com/tzutalin/labelImg>, 2015. Accessed: 2019-11-13.
- Dong-wei Xu, Yong-dong Wang, Li-min Jia, Yong Qin, and Hong-hui Dong. Real-time road traffic state prediction based on arima and kalman filter. *Frontiers of Information Technology & Electronic Engineering*, 18(2):287–302, 2017.



- Ramin Yasdi. Prediction of road traffic using a neural network approach. *Neural computing & applications*, 8(2):135–142, 1999.
- Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.
- G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.
- Xiaotong Zhao, Wei Li, Yifang Zhang, T Aaron Gulliver, Shuo Chang, and Zhiyong Feng. A faster rcnn-based pedestrian detection system. In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2016.