Final Year Project

Can Artificial Intelligence Reduce Traffic Congestion in Cities?

Alex O'Donnell

Student ID: 20456522

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Tahar Kechadi

UCD School of Computer Science

University College Dublin

April 26, 2024

Table of Contents

Abstract

The capabilities of humans executing complex tasks are rapidly being overtaken by artificial intelligence and machine learning. Conventional traffic light systems that are currently in place are inefficient and, in many cases, generate more traffic congestion than they resolve. In my final year project, I will aim to address these challenges by building a traffic congestion prediction model by implementing various artificial intelligence and machine learning techniques, focusing on increasing throughput, safety, and efficiency. This is especially important because the number of autonomous vehicles on our roads is increasing year after year. By researching and constructing different AI approaches to this issue, I can answer the question "Can Artificial Intelligence Reduce Traffic Congestion in Cities", reinforced by my own research, statistics, implementations, and conclusions.

Chapter 1: **Project Specification**

This research investigates the effectiveness of artificial intelligence (AI) in predicting and thereby reducing traffic congestion in Dublin City. This research paper is an attempt at addressing the critical implications of heavy traffic congestion for economic development, environmental impact, safety, and quality of life for Dublin's citizens. Through an extensive review of existing literature, I aim to highlight my project's approach by summarising key findings and methodologies I found most applicable to my project. This project encompasses extensive data preparation and analysis, as well as the selection of an appropriate machine learning model for traffic prediction. The results will be evaluated, and the integration of these models into a cohesive traffic prediction system will be discussed. The research concludes by highlighting key contributions and their potential impact on future urban planning and traffic management. The outline of this project is as follows:

- Comprehensively research the question 'Can Artificial Intelligence Reduce Traffic Congestion in Dublin City?'.
- Conduct an extensive review of existing research papers and studies on the implementation of AI to alleviate traffic congestion and summarise key findings.
- Provide clear documentation and explanations on the collection, analysis, and cleaning of the data used in this research project.
- Provide visualisations and statistical summaries to highlight patterns, trends, and potential outliers.
- Select an appropriate machine learning model for traffic prediction and provide justification.
- Specify how exactly the data will be prepared and reformatted to fit the model.
- Describe the process of training and testing the model.
- Specify hyperparameters, validation techniques, and any other relevant details regarding the training and testing of the prediction model.
- Present the final results of the prediction model.
- Discuss the implications of the results for future urban planning and traffic management, and how the models can be integrated into a cohesive traffic prediction system.
- Address potential challenges and limitations associated with the project, such as data quality issues, model complexity, or external factors affecting traffic.
- Provide an appropriate conclusion to the research done throughout this project.

Chapter 2: **Introduction**

Artificial Intelligence has revolutionised numerous fields and continues to enhance various aspects of our lives. Healthcare continues to advance with iterative development, which makes today's diagnosis more accurate, leading to the best possible results and the development of personalised treatment plans. In the finance sector, AI algorithms have made fraud detection and trading strategies significantly more sophisticated. AI automation has gone on to streamline manufacturing and logistics, making the processes more efficient and less prone to error. Customer service has been bolstered by Natural Language Processing (NLP) in the form of chat-bots and virtual assistants. As technology evolves, AI can adapt more to the learning surge, promising insistent enhancements leading to further innovations and providing diverse solutions in any given industry. The growth of artificial intelligence is so rapid that it outruns even the possibilities for human beings to accomplish complicated tasks.

However, the challenge of traffic congestion remains highly persistent in urban areas. As cities and populations grow, so does the strain on transportation systems and commuters. The ever-growing urbanisation and rise in vehicle population have put tremendous pressure on the available transportation infrastructure. This results in significant impacts on environmental pollution, economic losses, time waste, and quality of life. This is because prolonged periods of running motors and idling of vehicles in traffic jams lead to an increase in emissions of greenhouse gases and air pollutants. These emissions not only degrade air quality but also contribute to health issues like respiratory problems that make conditions like asthma and other pulmonary ailments worse. In addition to this, prolonged traffic congestion also increases fuel consumption, increasing our carbon footprint and, therefore, contributing to the ongoing global warming and climate change crisis. Traffic congestion further causes heavy economic losses at the individual and business level, due to productivity losses caused by commuters and workers wasting time travelling to their destinations. This eventually translates to increased costs for individuals as well as business owners. Vehicle wear and tear also contributes to these problems by raising operational costs for vehicle owners, companies, and governments.

Conventional traffic light systems that are currently in place are often inefficient and, in many cases, generate more traffic congestion than they are capable of resolving. In addressing this critical issue, the integration of Artificial Intelligence (AI) emerges as a promising solution for major cities. This report delves into the strategic implementation of AI technologies aimed at reducing traffic congestion in Dublin City. By harnessing the power of various AI technologies and machine learning models, I believe Dublin City can access a range of innovative solutions aimed at optimising traffic flow, enhancing public transportation, and boosting efficient urban mobility. The integration of machine learning is a tangible, actionable strategy to confront the intricate issues of today's urban traffic, with the potential to drastically transform the cities of tomorrow.

The project revolves around leveraging comprehensive data sets that I have obtained from Dublin City Council to implement a sophisticated machine learning model to predict traffic flow and congestion in specified areas. The gathered data sets contain time series traffic flow and congestion data from the last 2 years in Dublin, as well as an array of contextual variables contributing to this congestion. By employing modern machine learning models, these data sets will be meticulously analysed to decipher insights, correlations, and relationships within the data. The objective is to identify traffic trends, hot spots of congestion, and potential factors contributing to the growing traffic issues in the city. Through predictive analysis, the machine learning models will aid in forecasting traffic congestion scenarios based on various parameters derived from the data sets, such as location, date, and time. These invaluable insights can then be used to conceptualise

innovative and effective solutions to alleviate traffic congestion, such as optimising traffic signal timings, rerouting strategies, or suggesting infrastructure improvements. By leveraging machine learning models and predictive analysis, this report aims to navigate through intricate traffic data sets, aiming not only to forecast congestion scenarios but also to unearth proactive and impactful solutions.

In the following section of this report, I will delve into my background research on related works that I spent a significant amount of time analysing and dissecting for inspiration and insight regarding how to approach the issue myself. This comprehensive research not only provided me with a thorough understanding of the subject matter but also served as a catalyst for innovative ideas and informed perspectives. Following this, I will delve into my 'Implementation Outline' section, where I will discuss the means by which I plan on tackling this problem. This will include a thorough data collection and analysis phase, followed by comprehensive data processing in order to manipulate the traffic data to be prepared in an optimal manner for my machine learning models of choice. Subsequently, I will delve into the implementation process of my machine learning models, discussing and comparing various configurations of the model as well as outlining the coding process. This will be followed by an in-depth evaluation process, where I will choose the most suitable evaluation techniques and metrics to determine the performance of the prediction models. Afterward, in the 'Results' chapter, I will showcase the results of the prediction model, and also discuss the meaning and implications of these results. Finally, I will present my overall conclusions in the 'Conclusions' chapter, followed by discussing the possibilities of extending this project as well as any unrealised ideas in the 'Future Work' chapter.

Chapter 3: Background Research & Related Works

Several studies have highlighted the growing challenges of traffic congestion in urban areas. Factors contributing to this problem include population growth, increased urbanisation, and, what I will be delving into in this report, the reliance on traditional traffic management systems that struggle to adapt to dynamic traffic patterns. Before embarking on my mission to investigate this, I analysed a wealth of studies that have also examined similar ideologies, in order to improve my understanding of the task and further educate myself on the specifications of traffic congestion analysis. In this literature review, I will outline how other researchers have approached this problem, their reasons for addressing the issue, their methodologies, predictions, conclusions, and any patterns that have occurred throughout the different instances of research that I have detected.

3.1 The Need For Change

According to the research paper 'Intelligent Traffic monitoring, Prioritizing and Controlling Model based on GPS' [1], the need for improving traditional traffic congestion control systems can be broken down into five key aspects:

- Efficiency
- Environmental Impact
- Economic Development
- Emergency Response
- Safety
- Quality Of Life

Through researching various other papers and reports on this specific issue, I have found that there are a plethora of sources to further reinforce these core beliefs. These critical reasons to address inadequate congestion control resonated with me. Allow me to break down the importance of these core reasons, and how they align with various other research papers and reports on implementing artificial intelligence to reduce traffic congestion.

Implementing a well-structured traffic management system stands as a pivotal tool in diminishing the likelihood of accidents. By actively regulating traffic flow and ensuring vehicles maintain safe speeds, this system acts as a shield against potential risks, significantly contributing to road safety. This can be achieved by implementing artificial intelligence into our traffic congestion management systems. In the conference paper 'Artificial Intelligence Traffic Analysis Framework for Smart Cities' [2], the researcher states "However, the safety of driver is the main concern and remote monitoring of driver and vehicles is the only way to have smart city without accidents". I believe that it is a government's responsibility to ensure the safety of the roads for its citizens,

which can be significantly improved by the continuous implementation of new modernised traffic control systems. This is especially important for commuters during rush hours in the mornings or evenings. By ensuring the safety of our drivers, we are also vastly improving their quality of life.

The efficiency of a traffic congestion control system is closely tied to the environmental impact that it has. A robust traffic management system orchestrates the synchronised operation of traffic signals, orchestrating a seamless flow and minimising congestion. The resultant expedited travel times not only curb fuel consumption and emissions but also foster an overall enhancement in efficiency on roadways. At the core of a well-orchestrated traffic management system lies a commitment to reducing the environmental footprint (or carbon footprint) of vehicular transit. Traffic congestion aggravates time-wasting, especially during peak hours. It further stresses the issues of global warming and pollution. By curbing unnecessary idling and optimising fuel consumption, this system plays a crucial role in mitigating the environmental impact of vehicular activity [3]. Further enhancement of these pressing matters being a critical issue that needs to be addressed by updating current traffic control systems can be found in the paper 'Intelligent Traffic monitoring, Prioritising and Controlling Model based on GPS' [1], where the research delves into the various urgent concerns that traditional traffic congestion management frameworks generate. The author dives into the harmful environmental impacts that failing to address major traffic congestion is having on our climate: "A proper traffic management system can help reduce the environmental impact of traffic by minimising idling and reducing fuel consumption". In order to introduce a more robust traffic management orchestration, we need to be more proactive about implementing artificial intelligence into our systems. The conference paper Traffic Prediction for Intelligent Transportation Systems Using Machine Learning [4] further drives this point home with the concept of ITS (Intelligent Transportation System): " The main advantage of ITS is to provide a smooth and safe movement of road transportation. It's also helpful from the perspective of environmental friendliness to reduce carbon emissions. It provides many opportunities for the automotive or automobile industries to enhance the safety and security of their travellers".

The effects of traffic congestion on economic development are often overlooked. A proficiently designed traffic management system plays a pivotal role in bolstering business landscapes. By optimising delivery times and enhancing accessibility to commercial areas, it not only aids in current business operations but also acts as a beacon for attracting fresh investment and fostering the growth of new businesses [1]. These beliefs are reinforced by the research done in the paper 'Traffic Monitoring and Management for Congestion Control using IoT (Internet of Things) and AI' [5], where the researcher states their core beliefs regarding the need for vast improvement in traffic management in cities: "traffic jams have a direct and indirect bearing on the economy and health of the inhabitants of a nation". Traffic congestion hinders business development, leading to reduced revenues and growth [6]. The socio-economic impact of congestion is complex, affecting various sectors such as population, GDP, and the environment [7]. Empirical studies in the US have shown that congestion slows job growth and productivity, particularly when travel delays exceed certain thresholds [8]. I firmly believe that by implementing more sophisticated traffic congestion management systems by implementing artificial intelligence and machine learning, it would help facilitate business expansion and aid local businesses in flourishing.

Finally, a well-oiled traffic management operation is highly crucial with regards to the smooth operation of our emergency services in times of emergency. By providing real-time traffic information and enabling emergency vehicles to navigate through congested areas, it ensures swift access to accident scenes and other incidents, expediting emergency response and rescue operations, which ultimately will help save the lives of those in urgent need [1].

3.2 Possible Approaches

In the pursuit of researching how to improve traffic control models in cities by implementing more sophisticated models powered by artificial intelligence and machine learning, I have found an array of methodologies leveraging these technologies emerging as promising avenues to delve further into. These approaches encompass a multifaceted range, from predictive modelling and machine learning algorithms to real-time data analysis. The intricate exploration of these methodologies reveals their potential to revolutionise traditional traffic management systems, offering dynamic and adaptive solutions to tackle the ever-growing dilemma of congestion in urban environments. This introductory dive discusses the realm of AI-based traffic management methodologies and offers a compelling pathway towards understanding and implementing innovative measures to alleviate traffic gridlock and enhance the urban commuting experience. Of course, before applying your methodology to machine learning and other AI models, one must go through the processes of data collection, pre-processing, and analysis. I will delve deeper into what approaches other researchers chose and how I will navigate my own approach to this in the 'Data Collection' section 4.1.

As reported by the author of the research article 'A Review of Traffic Congestion Prediction Using Artificial Intelligence' [9], "Traffic flow is a complex amalgamation of heterogeneous traffic fleets. Thus, traffic pattern prediction modelling could be an easy and efficient congestion prediction approach". This research paper specifies an approach that is subdivided into three primary artificial intelligence implementation approaches:

- Probabilistic Reasoning
- Shallow Machine Learning
- Deep Machine Learning

Probabilistic Reasoning

Probabilistic reasoning stands as a cornerstone within the realm of Artificial Intelligence (AI), specifically addressing the management of uncertain reasoning and knowledge. Probabilistic Reasoning in AI refers to a method of reasoning that deals with uncertainty by using the principles of probability theory [10]. This facet plays a pivotal role in tackling the complexities inherent in predicting traffic congestion. An array of algorithms rooted in probabilistic reasoning serves as an anchor in various traffic congestion prediction studies. The various models that this researcher chose were:

1) Fuzzy Logic, a mathematical framework that deals with imprecise information, allowing for reasoning based on degrees of truth by utilising degrees of membership in sets rather than strict binary (true or false) values [11].

2) Hidden Markov Model, a statistical model used to analyse sequences, assuming an underlying structure that cannot be directly observed. They involve a probabilistic approach with hidden states influencing observable outcomes, often used in areas such as speech recognition and language processing [12].

3) Gaussian Mixture Models, a probabilistic model that assume data is generated from a mixture of several Gaussian distributions. They aim to estimate the underlying structure by fitting multiple Gaussian distributions to the data [13].

4) Bayesian Networks, a graphical model representing probabilistic relationships among variables using directed acyclic graphs. They utilise conditional probabilities to model dependencies between variables, allowing for inference and reasoning under uncertainty [14].

Shallow Machine Learning (SML)

SML encompasses fundamental and uncomplicated machine learning algorithms, typically characterised by a limited depth, often comprising just one hidden layer. These algorithms primarily lack the ability to autonomously extract features from the input data, necessitating predetermined feature definitions. Notably, training the model can solely commence post explicit feature extraction, limiting its autonomous learning capacity. The SML models that this researcher chose to use were:

1) Artificial Neural Network (ANN), a computational model inspired by the human brain's structure, used for pattern recognition and machine learning tasks. Comprising interconnected nodes (neurons) organised in layers, ANNs process data by transmitting signals, learning from input-output pairs, and adjusting connections to make predictions or classifications [15].

2) Regression Model, a statistical method used to analyze the relationship between dependent and independent variables, predicting continuous outcomes. It aims to establish and understand the pattern of the dependent variable based on one or more predictors, identifying correlations and making predictions based on the learned relationships [16].

3) Decision Tree, a common tree-like flowchart structure used in machine learning to make decisions based on multiple conditions, splitting data into smaller groups to classify or predict outcomes [17].

4) Support Vector Machine (SVM), a powerful supervised learning model that classify data by finding the best hyperplane, maximising the margin between different classes. SVMs can be formatted into multiple different kernels depending on the complexity of the dataset, i.e linear, poly, RBF and Sigmoid [18].

Deep Machine Learning (DML)

DML algorithms, characterised by multiple hidden layers, are designed to handle nonlinear complexities. Their key advantage lies in autonomously extracting features from input data without prior specifications. In contrast to Shallow Machine Learning (SML), DML integrates feature extraction with model training. The DML models chosen by this researcher are as follows:

1) Convolutional Neural Network (CNN), a model that enhances basic neural networks by incorporating specialised layers for feature extraction from image data. CNNs use convolutional layers and pooling to efficiently process spatial information, making them ideal for image recognition and computer vision tasks [19].

2) Long Short-Term Memory (LSTM), a type of recurrent neural network with specialised memory cells, capable of retaining and utilising long-range dependencies in sequential data. It excels in modeling sequential patterns and time series data due to its ability to capture and remember long-term information [20] [21].

3) Extreme Learning Machines (ELMs), a type of unconventional neural networks that randomly set input weights and analytically determine output weights, resulting in rapid learning with minimal tuning. ELMs are efficient for various tasks and large-scale datasets, offering quick training and straightforward implementation [22].

Before delving into dissecting my own dataset, my plan involves a comprehensive consideration of the various AI models highlighted in this article [9]. I aim to evaluate the suitability of each model in addressing specific aspects of my dataset analysis.

To further fortify these methods as a valid and promising methodology from the research I mentioned in [9], I found many overlapping features in the approach described by the author of the paper 'Traffic Prediction for Intelligent Transportation Systems using Machine Learning' [4]. For example, the use of Decision Tree's as a machine learning model in order to predict values accu-

rately and reliably based on data sets fed to it: "We have applied and tested different machine algorithms for achieving higher efficiency and accurate results. To identify classification and regression we have used a Decision Tree Algorithm (DT). The goal of this method is to predict the value of the target variables.". The predicted values of a set 'target class' are ultimately used to predict the level of congestion based on an array of input parameters. In addition to the use of Decision Trees, this research paper also described the use of Support Vector Machines (SVMs) in order to detect outliers from within the dataset, which is critical step in the process of accurately predicting levels of traffic.

In addition to these overlapping features in these approaches, the author of [4] also mentions the use of a Random Forest Algorithm, which is a robust machine learning algorithm defined as bootstrap aggregation, or, in simpler terms, it means it randomly selects a subset of the data with replacement. This means that some data points may be selected multiple times, while others may not be selected at all: "The bootstrap algorithm is used to generate multiple models from a single training data set. A bootstrap algorithm has also used a sample to estimate statistical quantities."

The author highlights the simplicity of the algorithms implemented with a simple breakdown in pseudocode. Despite the simplicity of the methodology, the results are promising and inspiring. The algorithms are as follows:

Algorithm 1 For identifying the congested situation

1. Collect the traffic data in every 5 min with features:
   A. Location (Measured with GPS)
   B. Direction
   C. Speed
   D. Start-End Junction
2. Group every 5 min interval with their corresponding data.
3. Calculate the distance between each vehicle with all another vehicles within specified junction.

if the distance is less than the specific threshold between two vehicles **then**
  those vehicles are considered to be the neighbourhood vehicles
else
  Not considered as neighbour vehicles.
end if

Figure 3.1: Identifying Congested Situation

Algorithm 2 For classifying the congested situation

1. This will eventually give us the matrix A.
2. Now assign 1 to $A[i, j]$

if $A[i, j] < threshold$ **then**
  $A[i, j] = 1$
else
  $A[i, j] = 0$
end if

3. Count $A[i, j]$=1 and label i, j as neighbourhood vehicles
4. Repeat above steps in every 5 min for 45 min
5. Plot the graph between neighbourhood vehicles and time interval.

if the neighbourhood vehicles shows an increasing graph **then**
  the traffic congestion is identified
else
  No traffic
end if

Figure 3.2: Classifying Congested Situation

The researcher then highlights the evaluation of the model by first displaying the accuracy, precision, recall, and time on a table, clearly indicating the efficiency of the model implemented. Following this, the researcher then plots the false positive rate versus the true positive rate on a Receiver Operating Characteristic (ROC) graph and signifies the reliability by evaluating the area under the curve. These two results figures showcase the impressive results simply for the reader:

Algorithm	Accuracy	Precision	Recall	Time
Decision Tree	88%	88.56%	82%	108.4sec
SVM	88%	87.88%	80%	94.1sec
Random Forest	91%	88.88%	82%	110.1sec

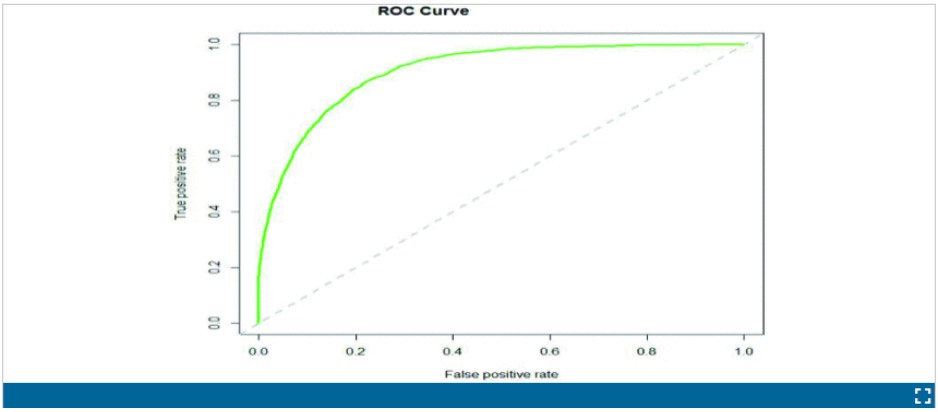

Figure 3.3: Results

As you can see, the area under the curve is substantial. A significant AUC-ROC suggests that the model is effective in differentiating between the positive and negative classes. The combination of overlapping features in approaches implemented and encouraging results in these two papers [9] and [4] indicate quite promising methodologies to consider when implementing my own. It is also worth noting that the researcher in [4] used the evaluation metrics accuracy, precision, and recall, which are common metrics in machine learning due to their reliability. However, in the paper 'Traffic Monitoring and Management System for Congestion Control using IoT and AI' [5], the researcher states the Root-Mean-Squared Error (RMSE) and F1-Score as their evaluation metric of choice. Both RMSE and F1-Score are viable and accurate evaluation metrics that I will consider using when implementing my own machine learning model.

Nevertheless, despite the validity and reliability of these various methodologies, a range of studies have consistently demonstrated the efficiency of Long Short-Term Memory (LSTM) Neural Networks across multiple applications, particularly in managing chaotic time-series data, such as traffic data [23]. An LSTM functions similarly to a conventional artificial neural network, which is inspired by the architecture of neural networks in human brain cells. Neurons in a biological neural network are represented by nodes in their artificial counterparts. A conventional artificial neural network is made up of three primary abstract layers, the input layer, the output layer, and the hidden layer. Each layer can be made up of a variable number of nodes, at the discretion of the architect. Datasets with more complex relationships between their features may require multiple hidden layers, with a high number of nodes in each. In the hidden layers, each node is connected to all of the nodes from the previous layer, and each connection has a corresponding 'weight' that is initialised to small random numbers close to zero. Data points are inputted through the input layer and then passed through the subsequent hidden layers. A weighted sum of the inputs is calculated at each node, and the activation function of choice is then applied to this sum. The activation function is a mathematical formula that determines whether or not a node should be "fired" or "activated", meaning how much signal should be passed from the current node to the next layer. Sigmoid and ReLU are common activation functions used in neural networks. The final output is then passed to the output layer, where the actual classification is made. This process is known as "forward propagation". A process known as "backpropagation" is then applied afterwards, which

is how the network "learns" by traversing back through the network and adjusting weights based on the level of error. This is done by employing a gradient descent algorithm, which can calculate the gradients of error with respect to each weight in the network [24].

An LSTM is a type of recurrent network. Although similar to a conventional neural network, it has architectural differences that allow it to be more effective when dealing with time-series data such as traffic congestion data. An LSTM differs from other networks by having parameterised nodes with internal loops that allow data to remain for long periods of time, giving it its memory capabilities. The flow of information is controlled by a series of gates, which decide what data is thrown away and what data is kept in memory to enhance its evaluation. A more in depth tutorial of how LSTMs function is covered in the paper [21].

It was found that an LSTM model outperformed other models in traffic flow prediction in [23], where the author said "We propose a LSTM model to predict the short-term traffic flow. LSTM is able to exploit the long-term dependency in the traffic flow data sequence. As one of the deep learning approach, LSTM is able to discover the latent feature representations hidden in the traffic flow." It is highlighted in their results that the LSTM produced a more accurate prediction than other models, such as the Support Vector Machine when evaluating the results using reliable metrics such as Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE). For their model, they used an input length of size 6, and a hidden layer made up of 40 neurons. The promising results outputted from this model can be seen in the table and figure below, providing a clear indication of why a LSTM might be the ideal model for my project.

Metrics	RW	SVR	WNN	SAE	LSTM
MAPE(%)	8.11	10.00	9.37	6.57	5.40
RMSE	69.27	46.82	53.86	47.32	40.34

Table 3.1: Comparative performance metrics.

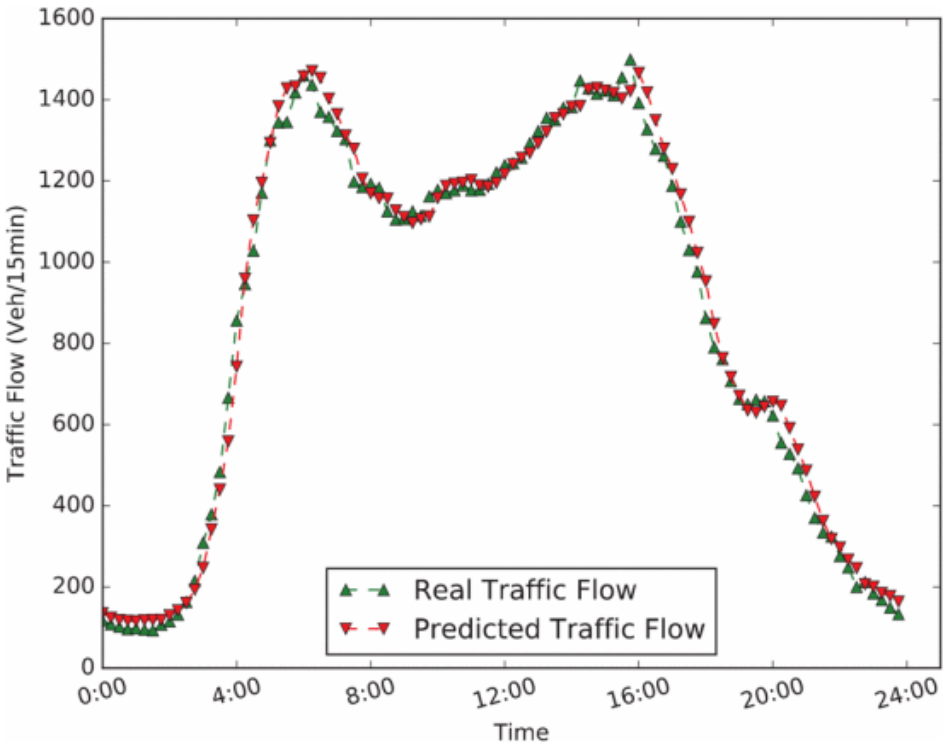

Figure 3.4: Results for LSTM Model.

Chapter 4: Implementation Outline

4.1 Data Collection

The intriguing results concluded from the research in [9] originate from the initial declaration that there are five vital parameters to evaluate regarding traffic congestion, and they are as follows:

- Travel Time (while monitoring and predicting traffic congestion)
- Traffic Congestion Index
- Traffic Volume
- Traffic Density
- Occupancy

In this case, traffic volume is a measure of the number of vehicles that pass a certain point, whereas traffic density refers to the number of vehicles occupying a given length of road at a particular instant. Occupancy is the proportion of time that a point on the roadway or detection zone is occupied by vehicles, and the traffic congestion index is a numerical scale that quantifies the level of congestion. It is worth noting that the data collection process in [9] can be subdivided into two primary classes: stationary and probe data. Stationary data can be categorised into two main types: sensor data and fixed cameras. In contrast, the probe data employed in the studies consisted of GPS data affixed to vehicles. The researcher in [1] employs a strategy much like the probate data collection described here, by employing an IoT (Internet of Things), in this case an interconnected network of devices reporting data back to their implemented cloud platform for data analysis, as well as GPS data: "The system collects real-time traffic data from GPS and Internet of Things (IoT) devices installed on vehicles and traffic signals. The data processed on a cloud-based platform is being used by AI and Deep Learning algorithms to predict traffic patterns and optimise traffic flow". A very similar approach can be seen in the paper "Artificial Intelligence Traffic Analysis framework for Smart Cities" [2]: "Sensors are embedded in the car to collect variety of information, this information collected in an internal gateway then forwarded to the cloud storage" A prototype of [1] data collection system can be seen here:

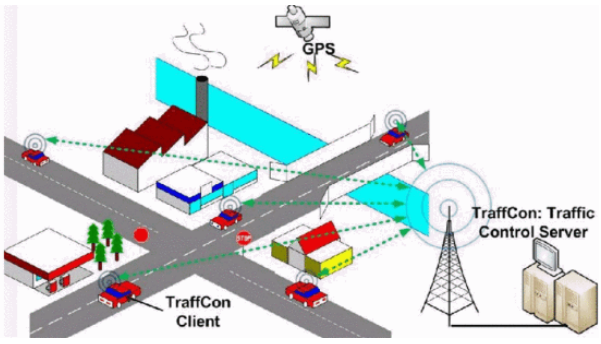

Figure 4.1: Data Collection Prototype

It is worth considering that gathering real-time traffic data involves monitoring and collecting information from various sources, including potentially sensitive data about individuals, which raises concerns regarding data privacy and ethical considerations. Despite the use of probe data [1][9] being a viable approach to take when carrying out your data collection phase, I have decided to use stationary data, i.e., traffic sensors, to monitor traffic flow and congestion. The reasons for making this decision include, but are not limited to:

- Upfront Cost: While initial setup costs may vary, deploying sensors for traffic monitoring can be cost-effective in the long run, especially when considering their durability and minimal maintenance requirements
- Precision in Location Data: Since the sensors will be static and positioned permanently, extracted data for location will be precise and give consistent results.
- Reliability: Sensor data is typically more reliable than probe data, as it is not dependent on the movement of individual vehicles. The sensor provides a continuous and consistent data supply with very low possibilities for data parts to be missed or to have wrong information in them.
- Standardised Quality: Sensor data usually come with quality standardisation at different points of monitoring, which helps in comparability and the ability to analyse congestion across various sensors' areas of interest

The traffic flow and congestion data that I decided to use was provided by South Dublin City Council (SDCC), which was collected using the SCOOT software provided by TRL (SCOOT) already installed for monitoring traffic flow congestion in the areas of South Dublin. SCOOT represents a self-learning system for the control of traffic signals at intersections in real-time. Major installations aimed at easing traffic congestion and increasing the overall capacity and efficiency in large towns comprise the SCOOT system. SCOOT collects flow data in real time from integrated traffic sensors and dynamically monitors its traffic data. The model of this kind of sophisticated data collection, like the one above, shows great promise as a tool for me to work with since it is a renowned and reliable source for data collection. Although SCOOT has packages to analyse the data so you can generate reports on it, I will be carrying out the data analysis myself in order to create a new, fresh perspective on the issue.

4.2 Data Analysis

The primary set of datasets that I have selected to work with is a representation of 'Traffic Flow Data' in the form of structured CSV data curated by SCOOT from 2022 and 2023. Both datasets cover the traffic flow in the period spanning from January to June. I have chosen these datasets because they contain the highest number of entries (rows) and the highest number of features (columns) compared to other datasets made available to the Irish public regarding traffic monitoring, which left a lot to be desired. Both datasets contain just over 1 million entries and 12 feature spaces per six-month time frame. These hugely populated datasets make the data easier to process, manipulate, and predict future data via machine learning. The following list contains a breakdown of the feature sets contained in my 'Traffic Flow Data' datasets, according to the publishers.

- Site: Locations region serial number.
- Day: The day on which the data was recorded.
- Date: The actual data collection date.
- Start Time: Time period start time.
- End Time: Time period stop time (15 minute intervals).
- Flow: A representation of demand (flow) for each link built up over several minutes by the SCOOT model - A smoothed average of values over a longer period, SCOOT will choose to use the appropriate profile depending on a number of factors.
- FlowPC: Flow percentage calculated based on a specific formula by PC SCOOT.
- Cong: Congestion is directly measured from the detector. Congestion is calculated from:No of congested intervals x 4 x 100 cycle time in seconds.
- CongPC: Congestion percentage calculated based on a specific formula by PC SCOOT.
- DSat: The ratio of the demand flow to the maximum possible discharge flow, i.e. it is the ratio of the demand to the discharge rate (Saturation Occupancy) multiplied by the duration of the effective green time. The Split optimiser will try to minimise the maximum degree of saturation on links approaching the node.
- DsatPC: DSat percentage calculated based on a specific formula by PC SCOOT.
- ObjectID: Identifier.

The variables in this dataset are described in more detail on data.gov.ie

All registered locations in this dataset are represented by the site's region serial number, which are stored in a separate dataset, also supplied by SDCC and operated by the SCOOT system. In order to represent results from the prediction model at a higher level for readability, I will use a mapping method to represent each serial number as the corresponding location when displaying the results later on in the project. There are 37 unique locations contained in the dataset.

Although the features 'FlowPC', 'CongPC' and 'DSatPC' provide a broader understanding of the data to the reader of the dataset, they are only percentages derived from the original features calculated by a specific formal by PC SCOOT. Consequently, they do not provide any additional information during the training of a machine learning prediction model. In fact, their inclusion may cause more harm than good due to over-fitting of the model, thereby diminishing it's effectiveness. For this this reason, I decided to drop these features and move on with the data processing phase.

4.3 Data Preprocessing

Before developing a prediction model, this data must be prepared in an optimal format for the given model to be developed—in this case, the LSTM neural network. This, in turn, will require rigorous data pre-processing. The first stage includes a coherent data quality check prior to any data manipulation or fitting into the neural network. This is very much required to find and fix any inconsistencies in the dataset. Key steps in this process include seeking duplicate entries within the dataset, verifying that the data types are all of the preferred type, confirming that there is not any missing or erroneous data within the dataset, and putting into place an effective strategy for the outliers in the dataset.

For the datasets provided by SDCC, no imputation was required for missing values or duplication of entries, as the data quality was mostly of a high standard. However, through the Z-score approach, an significant number of outliers were detected. A Z score (or standard score) represents how far a value is from the mean of a group of values in terms of standard deviations. The z score gives a measure of how atypical or typical a value is within the distribution. This can be given through the following formula with regard to the z-score (z) of the data point (x):

$$z = \frac{x - \mu}{\sigma} \tag{4.1}$$

Using a threshold of ± 3 standard deviations from the mean is the most traditional approach when defining an outlier based on its Z score value. However, using a threshold of ± 2 allows a broader range of data points to be identified as potential outliers. This can be beneficial when dealing with time series datasets, such as the one I am using for this project. A potential disadvantage of this approach is that it may lead to identifying too many outliers, but because my dataset is so large, we can afford this. The equation for using a Z-score with these parameters is as follows:

$$\text{Outlier if } z < -2 \text{ or } z > 2 \tag{4.2}$$

The following table highlights the number of outliers detected in each of the numerical features of the dataset after employing this Z-score method for detecting outliers. There were no duplicate outliers detected, meaning that each row identified as an outlier is unique and doesn't appear as an outlier in more than one column. This is a good sign, as it suggests that the outliers identified in different columns are distinct and not overlapping. As a result of this, the detected outliers could then be safely removed from the dataset before moving forward with the data preprocessing.

Year	Flow	Cong	DSat
2022	68112	18148	35937
2023	69822	22356	33285

Table 4.1: Outliers Data for 2022 and 2023.

As indicated in the table above, a total of 122197 entries were removed from the 2022 dataset, and 125463 entries were removed from the 2023 dataset. Considering that both datasets have just over a million entries, removing these outliers is justified as it reduces the noise in the dataset which will increase the prediction models efficiency.

It is common knowledge in machine learning that the next logical step in the data processing phase is to normalise the data, which is a critical step when training a LSTM neural network [25]. The reasons for this include, but are not limited to:

- Convergence Speed: The gradient-based optimisation algorithms used by neural networks are sensitive to the scale of the input data. These algorithms can be assisted in converging faster by normalising the data, which provides a more consistent scale across all input features. For LSTMs, this is especially crucial as they are often used on larger datasets such as ones I am using (1 million + entries) and can be much more computationally intensive [26].
- Learning Efficiency: Neural networks are prone to misinterpreting the importance of weights associated with features having larger scales when all of the features have vastly different scales. Normalising the data can help prevent this [27].
- Weight initialisation: Initially, neural networks initialise weights to random small values close to zero. If the data is not normalised, this could potentially cause the network to update some weights significantly more than others due to the inconsistencies in data scales in the dataset [27].
- Prevent gradient vanishing: During back-propagation, LSTMs are susceptible to issues with gradients during the back-propagation phase of the model training. The learning process can be destabilised or in the worst cases completely halted when gradients become either too small or too large. Dealing with a narrower and more consistent range of data inputs can aid in mitigating this potential issue by preventing extreme gradient values [28].

I decided to use a linear normalisation, also known as a "Min-Max Transformation" as my normalisation technique of choice. Min-Max normalisation confines the values in a collection of data to a fixed range, typically [0, 1]. When input features are of similar scale, neural networks tend to perform to a higher standard, which can be reinforced by the studies documented in the research article [29]. In comparison to other normalisation techniques, the adjusted min-max normalisation approach produced better results when working with a neural network in terms of the amount of variation explained in various sample sizes, according to the study's findings. Adjusted Min-Max is when a slight adjustment is made to the range in which the values are being confined, for example, using the range [0.1, 0.9] instead of [0, 1]. Transformed features can sometimes result in extreme values, which can be problematic for certain models. This adjustment is made in an attempt to prevent this. However, for simplicity, I will use a standard min-max normalisation technique for this project. The following equation describes how a linear Min-Max is performed:

$$z_i = \frac{x_i - \min(X)}{\max(X) - \min(X)}$$

Where:

- z_i is the normalised value of x_i.
- $\min(X)$ is the minimum value in the dataset.
- $\max(X)$ is the maximum value in the dataset.

This equation is applied to each value in the numerical columns that we want to normalise, in this case, 'flow', 'cong', and 'dsat'.

After removing the overly saturated amount of low values from the features 'flow', 'cong', and 'dsat', which could skew the training of the prediction model when preparing it for unseen values, the graphing of the distribution of the data after the normalisation function was applied reveals a uniform scaling across these data features: [0, 1]. This uniformity ensures that they are optimally prepared as input values for our LSTM neural network, as shown in the figures below, which were generated using the Python library 'Matplotlib' [30].

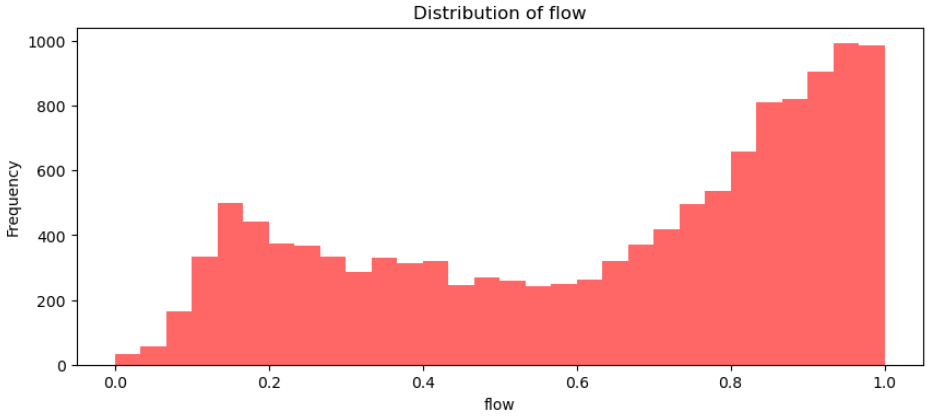

Figure 4.2: Flow Normalised

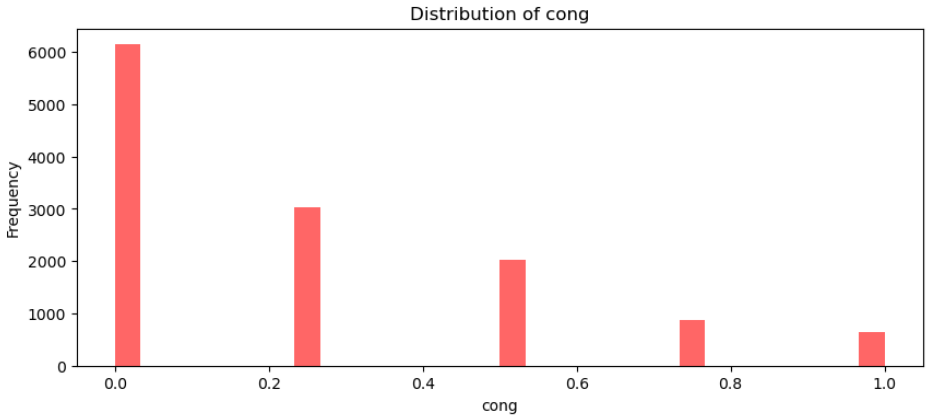

Figure 4.3: Congestion Normalised

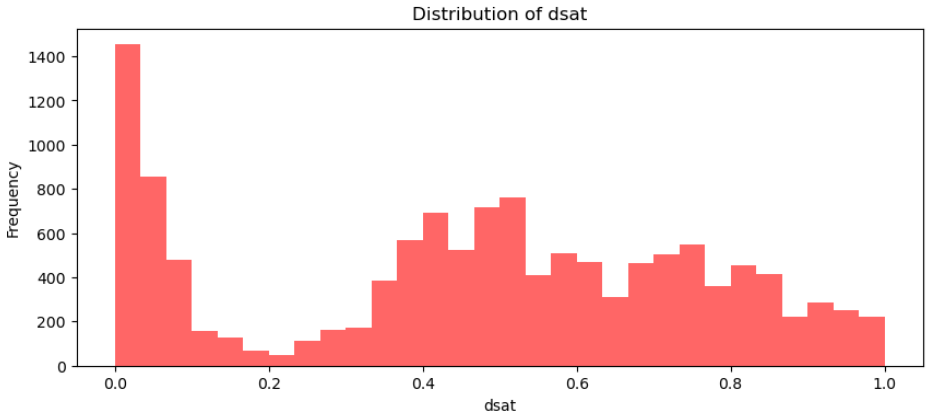

Figure 4.4: DSat Normalised

It is also worth considering that the underlying patterns in traffic related data are subject to a high degree of variability. A substantial pattern to consider is that traffic flow and congestion will vary considerably depending on whether or not the data is from during the working week (Monday-Friday), or during the weekend period (Saturday-Sunday). The figures below indicate the severity of these patterns, and for this reason, I decided to treat the original dataset as two separate datasets divided into weekdays and weekends.

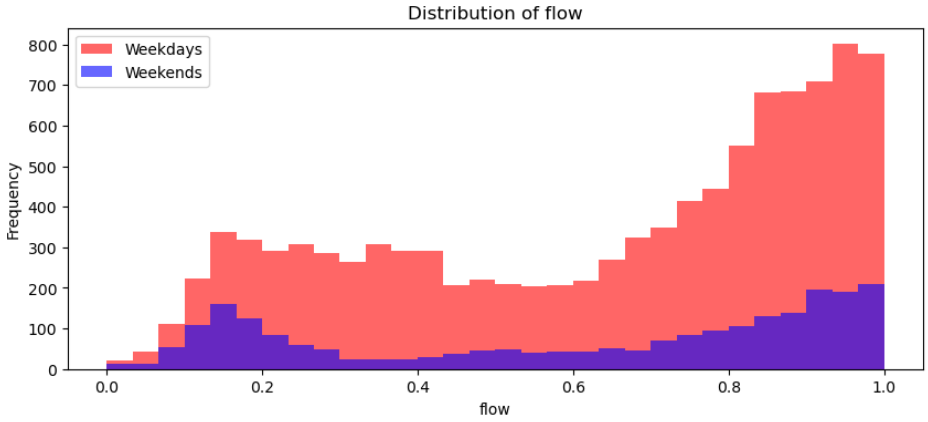

Figure 4.5: Flow Normalised

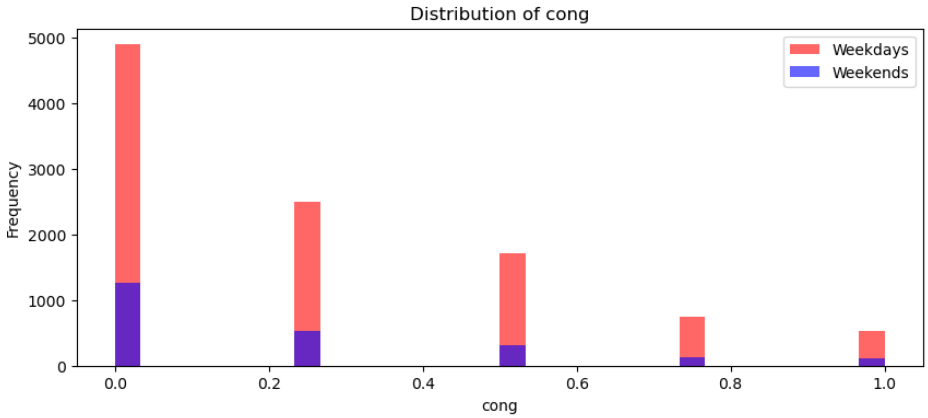

Figure 4.6: Congestion Normalised

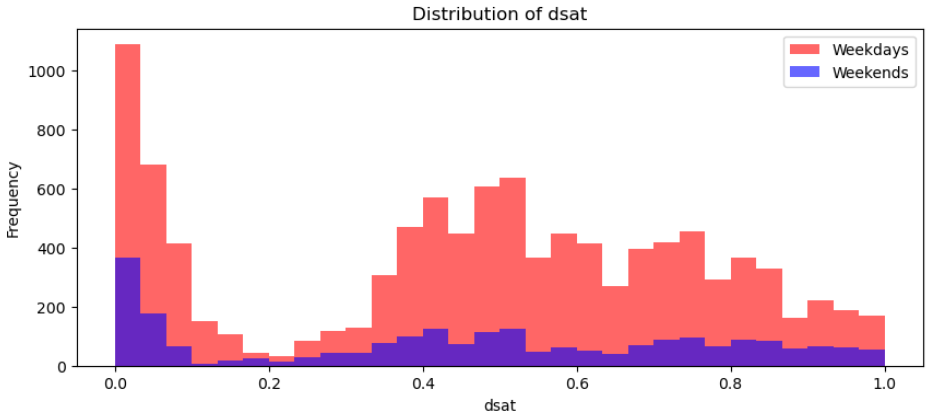

Figure 4.7: DSat Normalised

Now that the primary data processing steps are complete, there are a handful of final reformatting tasks specific to my dataset to take so that the data frame would optimally fit the LSTM network. This reformatting process was carried out using the Pandas Python library [31]. These tasks can be broken down into the following steps:

- Handle "24:00" in 'end_time' by replacing with "00:00".
- Convert 'date', 'start_time' and 'end_time features to datetime format.
- Increment the date by one day when 'end_time' is "00:00".
- Extract year, month and day as new features so they can be treated separately for more effective prediction.
- Calculate 'duration' as new feature instead of representing as time periods for each interval.
- Extract hour and minute from 'start_time' and 'end_time'.
- Convert 'site' and 'day' features to String types as opposed to Python Objects.
- One-hot encode 'site' and 'day' features for network efficiency.
- Drop original 'date', 'start_time' and 'end_time' features as they have been replaced by the reformatted features for the network.

4.4 Implementation Process

In this section, I will discuss in detail the implementation process for the LSTM neural network prediction model for traffic flow. To recap, here is a thorough summary of the methodology that I have been following for this project. We have now reached the stage of splitting the prepared data into training and testing data for the network.

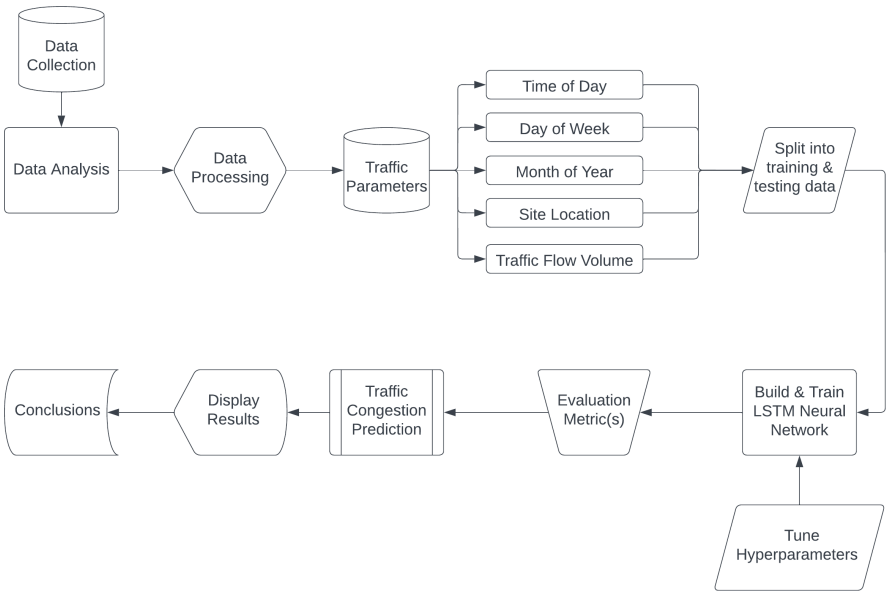

Figure 4.8: Methodology Flow Chart

The training of any machine learning model involves splitting our dataset into two distinct parts: the training set and the test set. Although other ratios are often used, the conventional split involves assigning 70% of the data to the training set, and the other 30% to the test set [32][33], and it is common practice to do this using a random seed of some kind to control the shuffling applied to the data before applying the split. Within this split, data has to be separated into two classes known as features or inputs, which are represented by 'X', and targets or outputs, represented by 'Y'. Y refers to the target variable(s) that we are trying to train the model to predict and what is eventually used to compare with the model's output to evaluate its performance and accuracy. These are known as the dependent variables. Conversely, X represents the input feature matrix that is used in conjunction with Y to train the model on the underlying patterns in the data and ultimately produce a prediction. We refer to these variables as independent variables. This type of training of a machine learning model is referred to as 'Supervised Learning', meaning that model is provided with labelled input data that is fully correct, and the model then learns to map the input to the output [34].

While experimenting with the data splitting process, it became clear to me that the model would be more accurate at predicting a single target variable as opposed to three (flow, cong and dsat). This is because predicting three variables as opposed to one is much more computationally complex. The features 'cong', 'flow', and 'dsat' are closely related and correlated. In the figure below, a heat-map indicates that 'dsat' and 'flow' are 66 percent correlated. This is mostly likely because, as mentioned before in my Data Analysis section 4.2, the calculation of 'dsat' is directly linked to 'flow'. In contrast to this, although 'cong' and 'flow' do not seem to be directly correlated on the heat map, in reality, traffic congestion and flow of traffic are tightly related. I believe the reasons for this discrepancy may be that the data quality for 'cong' is significantly lower than that for 'flow' and 'dsat' in this dataset. Both 'flow' and 'dsat' record a broad range of values, whereas 'cong' is confined to only 5 values, significantly decreasing its value as a feature when training the

prediction model. Despite this, I believe there is a safe solution to this potential data quality issue. A high flow of traffic in a confined area typically indicates a higher level of congestion due to the higher number of vehicles occupying the roads in the area. If we can accurately predict the exact level of flow in an area for a given location, day and time, then we can comfortably estimate the level of congestion using these results.

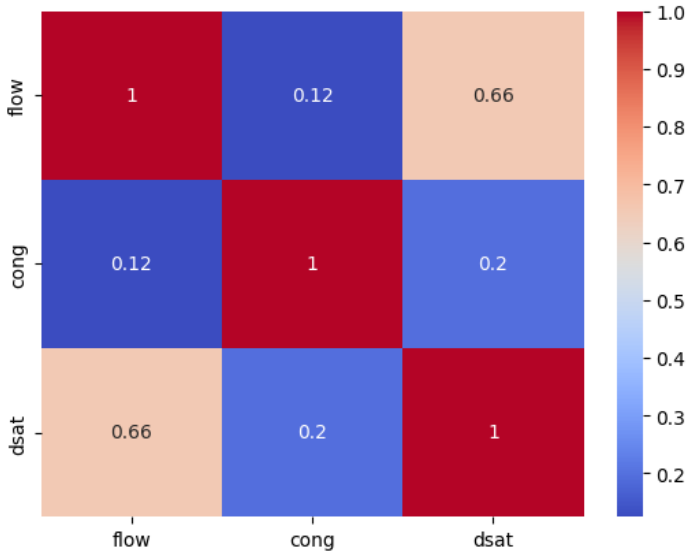

Figure 4.9: Dependent Variables Heatmap

Now that the optimum target variable has been selected, the LSTM neural network can be built and trained. I will utilise two commonly used machine learning tools, Keras [35] and TensorFlow [36]. Keras is a high-level application programmable interface, or API. It provides users with a simple way to define and train deep learning machine learning models, such as an LSTM network. The cutting-edge performance levels of Keras make it especially suitable for time-series data forecasting. TensorFlow, on the other hand, is a suite of tools and libraries that make up the backend engine to power Keras operations. Keras abstracts the complex details needed to work with TensorFlow, simplifying the process of building and training advanced machine learning models for users. Full documentation for these technologies can be found on their websites: Keras & TensorFlow

The next step in building the network is constructing an array of hyperparameters for the network and identifying which set of hyperparameters is best suited to my data. Hyperparameters in machine learning are a collection of low-level settings that make up the overall configuration and architecture of the model. Fine-tuning hyperparameters is a crucial step when building a model because they determine the behaviour of the model and have a significant impact on the overall performance and efficiency of the model. Based on my knowledge of neural networks, and the research done in the research papers [37] and [38], the most important hyperparameters to tune when training an LSTM neural network include, but are not limited to:

- Hidden Layers: Number of hidden layers in the network.
- Neurons: Number of neurons in each hidden layer.
- Activation: The activation function used in each in each neuron.
- Learning rate: Rate of correction based on estimated error.
- Optimiser algorithm: Optimising algorithm used to minimise the cost function (error).
- Epochs: Number of complete cycles through the dataset.

However, relying on a single configuration of a neural network is not enough to produce the most suitable network for your dataset. The researcher in [37] defines a broad search space of hyperparameters when testing various configurations for the network, including optimiser algorithms, hidden layers, neuron counts, epoch counts, and evaluation metrics. In addition to this, the researcher in [38] discusses the importance of comparing various learning rates with each configuration to determine the most suitable architecture for their network. Both researchers vouch for the reliability and efficiency of the ADAM (Adaptive Moment Estimation) optimiser algorithm. Building off of this, I have constructed the following hyperparameter search space, which results in 81 total LSTM networks to compare and evaluate.

Hyperparameters	**Search Space**
Hidden Layers	1, 2, 3
Neurons	10, 50, 100
Learning Rate	0.001, 0.01, 0.1
Activation Function	ReLU, Tanh, Sigmoid
Optimiser	ADAM
Epochs	100

4.5 Evaluation Metrics

There are a number of standard evaluation metrics commonly used when determining the overall performance of a neural network. Among the most popular are Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), Root Mean Square Error (RMSE), and R-Squared (R^2). Based on the connection between my project and the research done in [23] and the evaluating techniques applied in [37], I have decided to evaluate my model using all of the above-mentioned metrics, considering the results of all while focusing on MAE and R^2 as the primary evaluation methods. I will use the standard Python Math library to employ these metrics [39]. MAE and R^2 are given by the following mathematical formulas:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where:

- n is the number of observations in the dataset.
- y_i represents the actual values of the data.
- $\hat{y}_i$ represents the predicted values generated by the model.
- The absolute value of the difference between each actual value y_i and predicted value $\hat{y}_i$ is summed up and then divided by n to calculate the average.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where:

- n is the number of observations in the dataset.
- y_i represents the actual values of the data.
- $\hat{y}_i$ represents the predicted values generated by the model.
- $\bar{y}$ represents the mean of the actual values y.
- The numerator, $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, is the sum of the squared differences between each actual value y_i and predicted value $\hat{y}_i$, known as the sum of squares of residuals.
- The denominator, $\sum_{i=1}^{n}(y_i - \bar{y})^2$, is the sum of the squared differences between each actual value y_i and the mean $\bar{y}$, known as the total sum of squares.
- R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

4.6 Code Implementation

Below is a comprehensively documented code run-though of how the LSTM neural network described in this paper was implemented in Python using Keras and TensorFlow, following the hyperparameter search space, network architecture, and evaluation metrics described in 4.4. This code could only be implemented correctly after following all of the data processing steps highlighted in 4.3. The entire source code for my project can be viewed on github.com/aoddev232.

```
# Normalise features to fit model
columns_to_normalize = df_weekday.columns.difference(['flow'])
X_to_normalize = df_weekday[columns_to_normalize]
scaler = MinMaxScaler()
X_normalized = scaler.fit_transform(X_to_normalize)
X_normalized_df = pd.DataFrame(X_normalized, columns=columns_to_normalize
    , index=df_weekday.index)

# Add 'start_hour' back to the features without normalization
X_normalized_df['start_hour'] = df_weekday['start_hour']

# Remove cong and dsat target variables
X_normalized_df = X_normalized_df.drop(columns=['cong', 'dsat'])

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X_normalized_df, df_weekday[['flow']], test_size=0.3, random_state=23)

# Reshape input to be [samples, time steps, features]
X_train_reshaped = X_train.values.reshape((X_train.shape[0], 1, X_train.
    shape[1]))
X_test_reshaped = X_test.values.reshape((X_test.shape[0], 1, X_test.shape
    [1]))

# Function to create and evaluate the model
def train_and_evaluate_model(neurons, lr, activation, num_layers):
    model = Sequential()
    for i in range(num_layers):
        if i == 0:
            # First layer needs input shape specified
            model.add(LSTM(neurons, activation=activation, return_sequences
                =(i != num_layers - 1), input_shape=(1, X_train.shape[1])))
        else:
            # Following Layer(s)
            model.add(LSTM(neurons, activation=activation, return_sequences
                =(i != num_layers - 1)))
    model.add(Dense(1))
    model.compile(optimizer=Adam(learning_rate=lr), loss='
        mean_squared_error')

    # Fit model
    model.fit(X_train_reshaped, y_train, epochs=100, validation_data=(
        X_test_reshaped, y_test), batch_size=32, verbose=0)

    # Predict on test set
    predictions = model.predict(X_test_reshaped)
```

```
40
41      # Calculate evaluation metrics
42      mae = mean_absolute_error(y_test, predictions)
43      mse = mean_squared_error(y_test, predictions)
44      rmse = sqrt(mse)
45      r2 = r2_score(y_test, predictions)
46      mape = np.mean(np.abs((y_test - predictions) / y_test)) * 100
47
48      print(f"Neurons:{neuron} || Learning Rate:{lr} || Activation Function
            :{activation} || Number of Hidden Layers:{num_layers}")
49      print(f"MAE:{mae} || RMSE:{rmse} || MSE:{mse} || R2:{r2} || MAPE:{mape
            } || Model: {model}")
50      print("================================================================")
51
52      return mae, mse, rmse, r2, mape, model
53
54  # Hyperparameter configurations to loop through
55  neurons = [10, 50, 100]
56  learning_rates = [0.001, 0.01, 0.1]
57  activation_functions = ['relu', 'tanh', 'sigmoid']
58  num_layers_options = [1, 2, 3]
59
60  # Store all results from all hyperparameter configurations
61  results = []
62
63  # Iterate over hyperparameters
64  for neuron in neurons:
65      for lr in learning_rates:
66          for activation in activation_functions:
67              for num_layers in num_layers_options:
68                  mae, mse, rmse, r2, mape, trained_model =
                        train_and_evaluate_model(neuron, lr, activation,
                        num_layers)
69                  results.append({
70                      'neurons': neuron,
71                      'learning_rate': lr,
72                      'activation': activation,
73                      'num_layers': num_layers,
74                      'MAE': mae,
75                      'MSE': mse,
76                      'RMSE': rmse,
77                      'R2': r2,
78                      'MAPE': mape,
79                      'model': trained_model
80                  })
81
82  # Find the best model configuration
83  best_result = sorted(results, key=lambda x: x['MAE'])[0]
84  best_model = best_result['model']
```

Listing 4.1: LSTM Prediction Model Implementation

Chapter 5: **Results**

In this section, I will discuss the results from the performance evaluation of the LSTM prediction model when predicting the level of traffic flow based on location, time of day, day of the week, and month of the year. Of the 81 neural networks that were compared and contrasted, the table below displays the most effective hyperparameter configurations for the network when being evaluated using the metrics MAE and R^2.

Table 5.1: Results for Neural Network Performance

Period	Neurons	Learning Rate	Hidden Layers	Activation	MAE	R^2
2022 Weekdays	50	0.001	3	relu	0.071	0.887
2022 Weekends	50	0.01	3	sigmoid	0.070	0.909
2023 Weekdays	50	0.001	2	relu	0.071	0.883
2023 Weekends	50	0.01	3	sigmoid	0.061	0.932

Firstly, we can easily identify some patterns in the optimum hyperparameters to our network. We can see that the ReLU activation function coupled with a learning rate of 0.001 was the most efficient hyperparameter combination for predicting weekday traffic, whereas the Sigmoid activation function paired with a learning rate of 0.01 performed better on weekend traffic. ReLU is most effective when there is a linear relationship between input and target variables, as it is linear by nature. In contrast, sigmoid is non-linear by nature and is better suited to relationships in data that are non linear. Weekday traffic is inherently more consistent and stable than weekend traffic due to regular commuting and delivery patterns, whereas weekend traffic is subject to being less foreseeable due to events and other recreational activities that usually take place at the weekend. This could be a possible reason for the two activation functions performing better during different time periods. Another pattern that is instantly identifiable is that 50 neurons per hidden layer is the most appropriate neuron count, with a minimum of two hidden layers. This is significantly more complex than the network used by the researcher in [23], indicating that the underling relationships in my data are considerably more complex than the datasets used in Shao's project.

As highlighted in the performance metrics section of the table, the prediction model produced promising results. We can see the model performed slightly better at predicting traffic flow at the weekend than during the working week, producing an MAE rounded range of [0.061, 0.070], meaning that on average, the model predicted a traffic flow between 6.1% and 7% of the actual level of traffic flow. This low level of error is quite a promising result when dealing with time series data pertaining to traffic. The range of error produced by the model for the weekday traffic was only slightly worse than this, outputting a rounded MAE of 0.071 (7.1%) for both 2022 and 2023. This range is much narrower, meaning that although it produced a slightly higher error rate, the model was much more consistent at predicting weekday traffic. This correlates to the trends in the activation functions mentioned previously. In addition to this, the R^2 results are also promising, and follow the same pattern as the MAE results. The overall results are better for weekends, which yielded a rounded range of [0.909, 0.932], meaning that the model could explain approximately 91% to 93% of the variance in the dependent variable (flow), whereas for the weekdays it outputted a rounded range of [0.883, 0.887], approximating roughly 88.3% to 88.7% of the dependent variable variance. Once again, the range for the weekdays is overall slightly lower, but much narrower, indicating that it is more consistent at predicting the level of flow during the week than during the weekend hours. These are quite high values, which suggests that the model fits the traffic data well and is efficient at explaining a large proportion of the variance. Concluding from this, I can

confirm these configurations for the LSTM neural network result in promising performance levels when predicting the level of traffic flow in a given region during a specified time period.

We can further our understanding of these results by visualising the predictions. Contained in the figures below are the model's predictions of traffic flow compared against the actual level of traffic flow in specified locations at specified times.

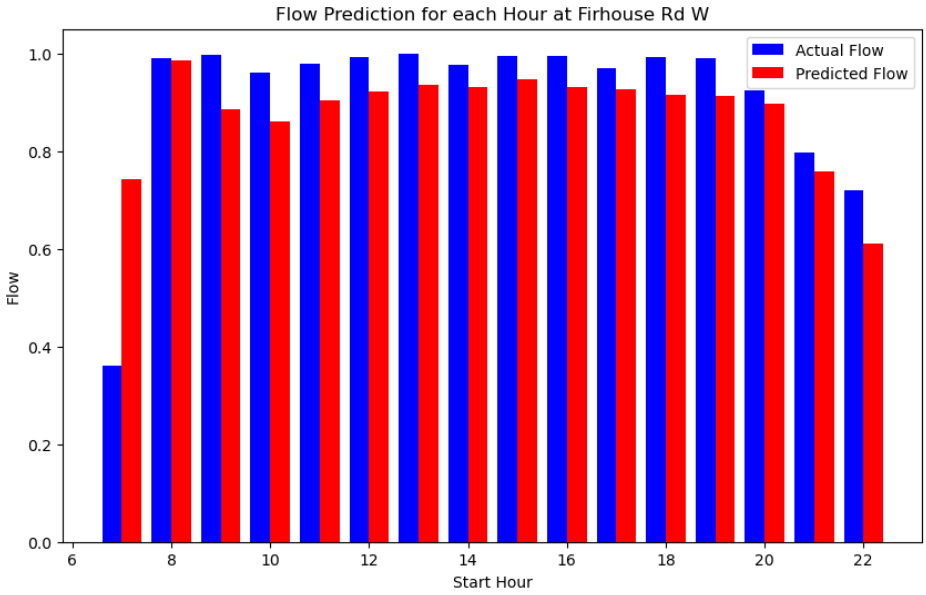

Figure 5.1: Flow prediction for Firhouse - Example Monday 2022

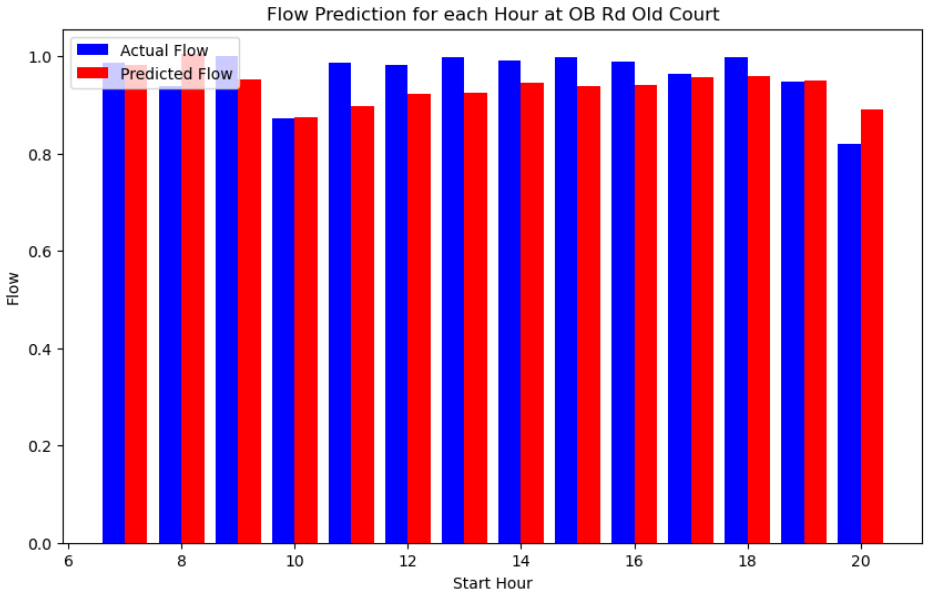

Figure 5.2: Flow prediction for Old Court Road - Example Tuesday 2022

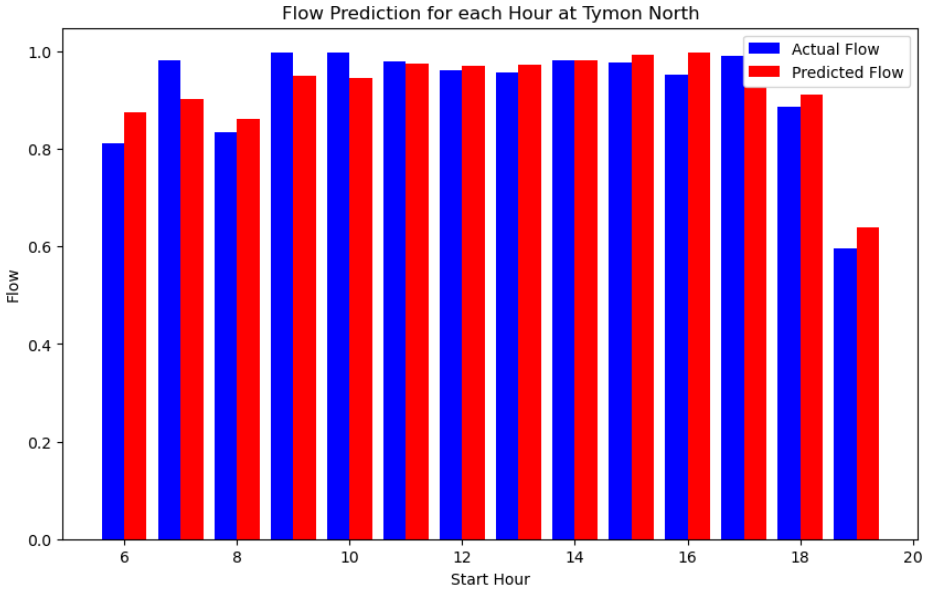

Figure 5.3: Flow prediction for Tymon North Road - Example Monday 2023

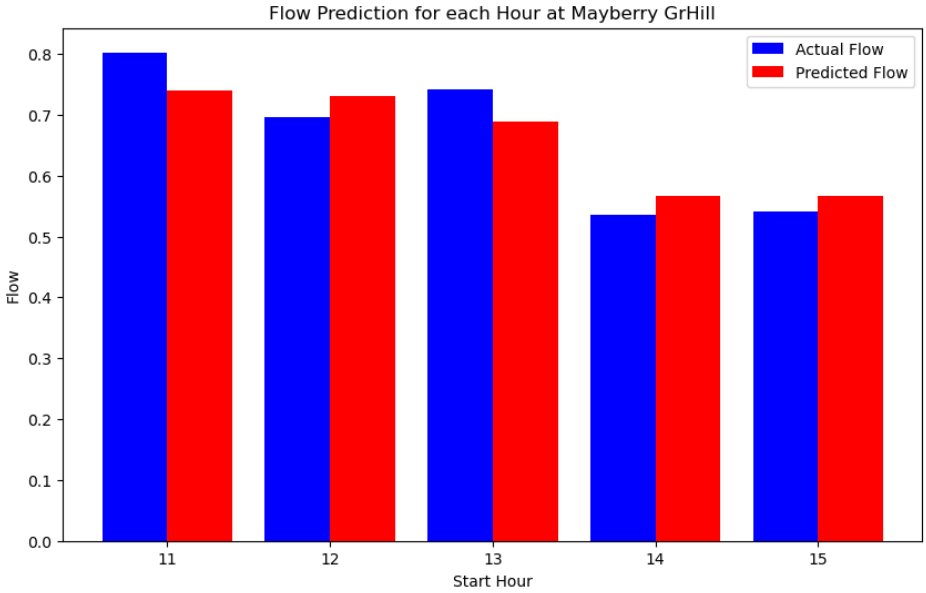

Figure 5.4: Flow prediction for Mayberry Road - Example Saturday 2022

It is worth noting that some results in a small number of locations were erroneous and could not be graphed. I suspect this is due to some data quality issues that I missed during the data preprocessing phase 4.3, most likely a lack of input in these locations compared to others, causing the model's predictions to be way off. Generally speaking, the model performed very well in predicting the level of traffic flow, as illustrated in the figures above.

Chapter 6: **Conclusions**

Throughout this project, I obtained densely populated traffic-related datasets from SDCC, which I analysed in detail and processed thoroughly to efficiently prepare them for the 81 different LSTM neural networks I configured with varying hyperparameters and architectural differences. The results of each were compared and contrasted to select the best-performing models for both weekdays and weekends in 2022 and 2023. The results were clearly documented and displayed in a simple and intuitive manner. However, my project implementation and results are not perfect and can definitely be improved and extended upon in different ways that I will discuss in my 'Future Work' section 7, such as collecting a higher concentration of high-quality congestion data as well as processing the data for various other machine learning models like support vector machines and convolutional neural networks.

Initially, I was posed with the question, "Can artificial intelligence reduce traffic congestion in major cities?". My aim was to research existing approaches to this urban issue and use this knowledge to collect, analyse, and process my own data, which I then subsequently used to build and train my own machine learning model to predict traffic congestion in Dublin City. Based on my background research and my own results and evaluations, I am happy with my decision to use a Long Short Term Memory Neural Network as my machine learning model, as the best performing hyperparameter configurations produced promising results in terms of traffic flow prediction, achieving estimations of traffic flow within 7% of the exact flow levels as well as explaining up to 93% of the variance of traffic flow levels in South Dublin. These traffic flow level predictions can inherently be used to accurately approximate traffic congestion in the respective area.

I believe that these positive results can be used going forward by governments and city planners to conceptualise an actionable strategy to confront the complex issue of today's traffic congestion issues in major cities. By integrating a reliable AI-powered prediction model like the one I have showcased in this research project, in conjunction with computing paradigms like edge computing [40], I think that we can significantly lower the levels of congestion in our city. The positive implications of this include reducing our national carbon footprint through lowering our emission rates and reducing fuel consumption, lowering our economic losses by optimising our travel patterns, and increasing the overall quality of life for our commuters. Factors to consider when hypothetically installing a more sophisticated traffic management system like this are higher upfront costs and potentially a need for a slow integration process so that we can safely make the transition from a traditional traffic management paradigm to an advanced AI-powered system.

In conclusion, I believe it is clear from the results that, with the right amount of research, investment, and patience, artificial intelligence can contribute to reducing traffic congestion in major cities.

Chapter 7: **Future Work**

Although I am happy with the progress I made throughout this project, there are some aspects of it that feel partially unfinished or could be extended if I could allocate more time to the project in the future. The first thing I would do to improve the standard of the project is collect more data in conjunction with South Dublin City Council, specifically congestion data rather than traffic flow data. As mentioned previously, the congestion data provided by SDCC was of lower quality compared to traffic flow, which is what merited the decision to instead predict traffic flow to approximate traffic congestion. This would be highly beneficial, as it would improve the sophistication of the model by enabling it to predict the level of flow and congestion simultaneously for a given location and time period. In addition to improving the sophistication of the model, collecting more data would also improve the model's performance by providing a larger training set for the training phase of the model. I also believe collecting more data from new locations would be beneficial. Expanding upon the current collection of 37 sites would increase the surface space for prediction for the model. A model capable of predicting traffic flow and congestion in 100 unique locations, for example, would be a highly beneficial tool for the SDCC, but it would also require a more complex LSTM as the input matrix would be much larger than the one used in this project.

In the event that this project's time frame is increased, I would also like to explore building and testing other models that I investigated in my background research chapter 3.2. Other types of neural networks, such as convolutional neural networks (CNN) [19], typically used for image processing but could definitely be adapted for time-series data forecasting. I could also implement completely different machine learning models, such as Support Vector Machines (SVMs) [18] which could also be tuned for traffic flow and congestion prediction. I think implementing and comparing a wide range of prediction models would be beneficial for the project, as it would be a more comprehensive approach to traffic prediction than I was capable of carrying out in the limited time frame.

With increased time and resources, I would have liked to have been able to use the most efficient prediction models to assemble a conceptual implementation plan to possibly integrate the congestion and flow prediction models into a small sample of traffic lights across south Dublin, most likely the most congested roads from our dataset. In theory, it would be quite simple to do this without requiring much physical change to the current infrastructure in place, rather simply by implementing the concept of 'edge computing'[40]. Edge computing is a paradigm that involves situating computational and data storage servers within close proximity of where they are needed to reduce latency, save bandwidth, and improve security and scalability. If we hosted the best prediction models on edge servers located in South Dublin and then connected the selected traffic lights to this model, they could alter the timing of their light rotations based on the level of traffic flow and congestion predicted to build up within the next hour. This could provide some relief to areas that become congested very quickly due to stubborn light rotations. To reinforce this concept, I would compare and contrast simulations of the traditional system with this conceptual artificial intelligence-integrated system. This could be achieved using various traffic simulation software that is publicly available or by implementing my own, powered by Unreal Engine [41]. These simulations would demonstrate the significant impact that artificial intelligence-integrated traffic systems could have on traffic congestion alleviation.

Chapter 8: **Acknowledgements**

I would like to express my deepest gratitude to Dr. Tahar Kechadi for his invaluable guidance and expertise throughout the development of this research project. His insights and suggestions significantly enhanced the quality of this work. I would also like to thank my friends and family for supporting me throughout my years spent studying at University College Dublin and completing this research project.

Bibliography

1. C, M. *et al. Intelligent Traffic Monitoring, Prioritizing and Controlling Model based on GPS* in *2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA)* (2023), 297–299.
2. Tarawneh, M., Alzyoud, F. & Sharrab, Y. *Artificial Intelligence Traffic Analysis framework for Smart Cities* in (Apr. 2023).
3. Marve, S. R., Bhorkar, M. & Baitule, P. A Survey on Environmental Impacts Due to Traffic Congestion in Peak Hours. *International Journal For Science Technology And Engineering* **2,** 151–154. https://api.semanticscholar.org/CorpusID:212498046 (2016).
4. Meena, G., Sharma, D. & Mahrishi, M. *Traffic Prediction for Intelligent Transportation System using Machine Learning* in *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)* (2020), 145–148.
5. Larhgotra, A., Kumar, R. & Gupta, M. *Traffic Monitoring and Management System for Congestion Comtrol using IoT and AI* in *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)* (2022), 641–646.
6. Santos, L. P. A. *Influence of Traffic Congestion in Business Development: A Literature Review* in (2020). https://api.semanticscholar.org/CorpusID:231836287.
7. Jie, L. An Analysis of the Socio-economic Impact of Traffic Congestion. *Journal of Beijing Jiaotong University.* https://api.semanticscholar.org/CorpusID:156726507 (2012).
8. Sweet, M. N. Traffic Congestion's Economic Impacts: Evidence from US Metropolitan Regions. *Urban Studies* **51,** 2088–2110. https://api.semanticscholar.org/CorpusID:152942961 (2014).
9. Akhtar, M. & Moridpour, S. A review of traffic congestion prediction using artificial intelligence. *Journal of Advanced Transportation* **2021,** 1–18 (2021).
10. Rényi, A. *Probability theory* (Courier Corporation, 2007).
11. Shawi, R. E., Tomasiello, S. & Liiva, H. *Research perspective: the role of automated machine learning in fuzzy logic* in *International Workshop on Fuzzy Logic and Applications* (2021). https://api.semanticscholar.org/CorpusID:248215986.
12. Poritz, A. B. Hidden Markov models: a guided tour. *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing,* 7–13 vol.1. https://api.semanticscholar.org/CorpusID:62479678 (1988).
13. Reynolds, D. A. *Gaussian Mixture Models* in *Encyclopedia of Biometrics* (2009). https://api.semanticscholar.org/CorpusID:1063711.
14. Niedermayer, D. *An Introduction to Bayesian Networks and Their Contemporary Applications* in *Innovations in Bayesian Networks* (2008). https://api.semanticscholar.org/CorpusID:16696554.
15. Assistant, A. P. *Artificial Neural Network Kshirsagar* in (2012). https://api.semanticscholar.org/CorpusID:212546924.
16. Kumar, S. & Bhatnagar, V. A Review of Regression Models in Machine Learning. *Journal of Intelligent Systems and Computing.* https://api.semanticscholar.org/CorpusID:244209278 (2021).

17. Navada, A., Ansari, A. N., Patil, S. & Sonkamble, B. A. Overview of use of decision tree algorithms in machine learning. *2011 IEEE Control and System Graduate Research Colloquium,* 37–42. https://api.semanticscholar.org/CorpusID:42879624 (2011).

18. Suthaharan, S. *Support Vector Machine* in (2016). https://api.semanticscholar.org/CorpusID:67438765.

19. Véstias, M. P. Convolutional Neural Network. *Programming with TensorFlow.* https://api.semanticscholar.org/CorpusID:224885070 (2021).

20. Mojica, F., Villaseñor, C., Alanis, A. Y. & Arana-Daniel, N. Long Short-Term Memory with Smooth Adaptation. *2019 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC),* 1–6. https://api.semanticscholar.org/CorpusID:215723449 (2019).

21. Staudemeyer, R. C. & Morris, E. R. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks* 2019. arXiv: 1909.09586 [cs.NE].

22. Chen, C., Li, K., Duan, M., Li, K. & Li, K. *Extreme Learning Machine and Its Applications in Big Data Processing* in (2017). https://api.semanticscholar.org/CorpusID:63032381.

23. Shao, H. & Soong, B.-H. *Traffic flow prediction with Long Short-Term Memory Networks (LSTMs)* in *2016 IEEE Region 10 Conference (TENCON)* (2016), 2986–2989.

24. Amari, S.-i. Backpropagation and stochastic gradient descent method. *Neurocomputing* **5,** 185–196 (1993).

25. Brownlee, J. *Better deep learning: train faster, reduce overfitting, and make better predictions* (Machine Learning Mastery, 2018).

26. Baron, R. J. & Girau, B. Parameterized normalization: application to wavelet networks. *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)* **2,** 1433–1437 vol.2. https://api.semanticscholar.org/CorpusID:62253084 (1998).

27. Ren, M., Liao, R., Urtasun, R., Sinz, F. H. & Zemel, R. S. Normalizing the Normalizers: Comparing and Extending Network Normalization Schemes. *ArXiv* **abs/1611.04520.** https://api.semanticscholar.org/CorpusID:5462200 (2016).

28. Dai, Z. & Heckel, R. Channel Normalization in Convolutional Neural Network avoids Vanishing Gradients. *ArXiv* **abs/1907.09539.** https://api.semanticscholar.org/CorpusID:198179616 (2019).

29. Aksu, G., Güzeller, C. O. & Eser, M. T. The Effect of the Normalization Method Used in Different Sample Sizes on the Success of Artificial Neural Network Model. *International Journal of Assessment Tools in Education* **6,** 170–192 (2019).

30. Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* **9,** 90–95 (2007).

31. Pandas development team, T. *pandas-dev/pandas: Pandas* version latest. Feb. 2020. https://doi.org/10.5281/zenodo.3509134.

32. Liu, H. & Haig, E. Semi-random partitioning of data into training and test sets in granular computing context. *Granular Computing* **2,** 357–386. https://api.semanticscholar.org/CorpusID:13672832 (2017).

33. Sugali, K., Sprunger, C. & Inukollu, V. N. AI Testing: Ensuring a Good Data Split Between Data Sets (Training and Test) using K-means Clustering and Decision Tree Analysis. *International Journal on Soft Computing.* https://api.semanticscholar.org/CorpusID:232220662 (2021).

34. Vermeulen, A. F. in *Industrial Machine Learning: Using Artificial Intelligence as a Transformational Disruptor* 63–136 (Apress, Berkeley, CA, 2020). ISBN: 978-1-4842-5316-8. https://doi.org/10.1007/978-1-4842-5316-8_4.

35. Chollet, F. *et al. Keras* https://github.com/fchollet/keras.

36. Martín Abadi *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015. https://www.tensorflow.org/.

37. Mao, Y. *et al.* Selection of Precise Long Short Term Memory (LSTM) Hyperparameters based on Particle Swarm Optimization. *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC),* 1114–1121. https://api.semanticscholar.org/CorpusID:249809197 (2022).

38. Li, W., Ng, W. W. Y., Wang, T., Pelillo, M. & Kwong, S. T. W. HELP: An LSTM-based approach to hyperparameter exploration in neural network learning. *Neurocomputing* **442,** 161–172. https://api.semanticscholar.org/CorpusID:233795807 (2021).

39. Van Rossum, G. *The Python Library Reference, release 3.8.2* (Python Software Foundation, 2020).

40. Cao, K., Liu, Y., Meng, G. & Sun, Q. An Overview on Edge Computing Research. *IEEE Access* **8,** 85714–85728 (2020).

41. Epic Games. *Unreal Engine* version 5.3.2. Apr. 19, 2024. https://www.unrealengine.com.