MANCHESTER 1824
The University of Manchester

Event-Enhanced Proactive Congestion Management: A Data-Driven Approach to Reducing Congestion and CO_2 Emissions

A report submitted to The University of Manchester for the degree of
Bachelor of Science in Computer Science
in the Faculty of Science and Engineering

Year of submission
2025

Author
Lourenço Figueiredo Silva
Supervised by: Dr. Sandra Sampaio

Department of Computer Science

Contents

Contents . . . 2

List of figures . . . 5

List of tables . . . 6

List of abbreviations . . . 7

Abstract . . . 8

Declaration of originality . . . 9

Intellectual property statement . . . 10

Acknowledgements . . . 11

1 Introduction . . . 12
- 1.1 Project Context and Motivation . . . 12
- 1.2 Aims and Objectives . . . 13
- 1.3 Evaluation Strategy . . . 13
- 1.4 Report Structure . . . 13

2 Background and Related Work . . . 14
- 2.1 Intelligent Transportation Systems . . . 14
- 2.2 Traffic Forecasting . . . 15
  - 2.2.1 Time-Series Forecasting (ARIMA) . . . 16
  - 2.2.2 Machine Learning and Neural Network Techniques . . . 16
  - 2.2.3 Recent Advancements . . . 17
- 2.3 Congestion Management Approaches . . . 18
  - 2.3.1 Reactive Methods . . . 18
  - 2.3.2 Proactive Methods . . . 19
- 2.4 Traffic Simulation . . . 19
  - 2.4.1 Simulation of Urban MObility (SUMO) . . . 20
  - 2.4.2 Traffic Control Interface (TraCi) . . . 20
- 2.5 Research Gaps . . . 21
  - 2.5.1 Predictive Model Integration . . . 21
  - 2.5.2 Importance of External Contextual Data . . . 21
  - 2.5.3 Project Positioning and Justification . . . 21

3 Design and Implementation . . . 22
- 3.1 Data Storage . . . 22
- 3.2 Data Collection Pipeline . . . 22
  - 3.2.1 Traffic Data . . . 23
  - 3.2.2 Weather Data . . . 24
  - 3.2.3 Event Data . . . 25
- 3.3 Data Analysis . . . 26

3.3.1 Sensor and Venue Locations . . . 26
3.3.2 General Analysis and Preprocessing . . . 27
3.3.3 Correlation with Events . . . 29
3.3.4 Traffic Speed vs. Count Analysis . . . 29
3.4 Feature Engineering . . . 29
3.4.1 Traffic Features . . . 30
3.4.2 Weather Features . . . 30
3.4.3 Event Features . . . 30
3.4.4 LLM-based Attendance Estimation . . . 32
3.5 Predictive Modeling . . . 33
3.5.1 Design Decisions . . . 33
3.5.2 Traditional Statistical Model (ARIMA) . . . 35
3.5.3 Neural Network Model (LSTM) . . . 35
3.5.4 Neural Network Model (MLP) . . . 37
3.5.5 Hyperparameter Tuning . . . 38
3.5.6 Evaluation Strategy . . . 39
3.6 Traffic Simulation . . . 39
3.6.1 Design Decisions . . . 40
3.6.2 Map Generation . . . 42
3.6.3 Prediction Generation . . . 42
3.6.4 Demand Simulation . . . 43
3.6.5 Congestion Management Strategies . . . 45
3.6.6 Challenges Encountered . . . 46

4 Results and Discussion . . . 47
4.1 Model Performance Evaluation . . . 47
4.1.1 Hyperparameter Tuning . . . 51
4.1.2 Feature Engineering Comparison and Impact . . . 51
4.2 Simulation Outcomes . . . 54
4.2.1 Real-World Demand Generation Results . . . 55
4.2.2 Reactive Hyperparameter Search . . . 56
4.2.3 Baseline vs. Reactive vs. Proactive Strategies . . . 57
4.3 Key Observations . . . 60
4.4 Limitations and Sources of Error . . . 61

5 Conclusions and Future Work . . . 62
5.1 Summary of Contributions . . . 62
5.1.1 Achievements in Predictive Accuracy . . . 62
5.1.2 Advancements in Proactive Congestion Management . . . 62
5.1.3 Novel LLM-based Attendance Estimation . . . 62
5.2 Future Research Directions . . . 63
5.2.1 Expanding Data Sources . . . 63
5.2.2 Advanced Control Strategies for Congestion Management . . . 63
5.2.3 Field Testing and Large-Scale Deployment . . . 63
5.3 Key Takeaways . . . 64

References **64**

Appendices **69**

A Source Code **69**

B LLM-Based Attendance Estimation Prompt **69**

C Reproducibility **69**

Word count: 14959

List of figures

1 The flow of information in SCOOT based UTC system - an example of an ITS 15
2 A simplified MLP architecture diagram 16
3 A simplified LSTM architecture diagram 17
4 An example SUMO simulation 20
5 Architecture diagram of Manchester-I data collection 24
6 Architecture diagram of event data collection 26
7 Zoomed-out plot of traffic platforms and venues collected 27
8 Zoomed-in plot of traffic platforms and venues collected 27
9 Plot of traffic count over a year 28
10 Plot of traffic speed over a day 28
11 Plot of traffic data around an event at Etihad 29
12 Plot of correlation between traffic speed and traffic count for one direction 30
13 Architecture diagram of LLM-based Attendance Estimation 33
14 Architecture diagram of ARIMA predictive modelling 35
15 Architecture diagram of LSTM predictive modelling 36
16 Architecture diagram of MLP predictive modelling 38
17 Area boundary for simulation with labels 41
18 Architecture diagram of automated simulation sensor placement 43
19 Architecture diagram of automated simulation demand generation 44
20 Rule-based reactive implementation rules 45
21 Plot of ARIMA model forecast results compared with test data 48
22 Plot of LSTM model one-step forecast results compared with test data 48
23 Plot of LSTM model walk-forward forecast results compared with test data 49
24 Plot of MLP model forecast results compared with test data at three zoom levels . . . 50
25 Plot of MLP model forecast results compared with test data using basic time encoding 52
26 Plot of MLP model forecast results compared with test data using cyclic time encoding 53
27 Plot of MLP model forecast results compared with test data using binary event encoding 54
28 Plot of MLP model forecast results compared with test data using custom event encoding 54
29 Plot of MLP model forecast results compared with test data using LLM estimation . . 55
30 Plot of simulation-observed traffic speeds compared with the real world for a typical day 56
31 Plot of simulation-observed traffic speeds compared with the real world for an event day 56
32 Plot of results of hyperparameter optimisation for reactive rule strategy 57
33 Plot of congestion management techniques results on 2024 simulations 59
34 Plot of congestion management techniques results on 2024 simulations - outlier removed 60
35 Plot of congestion management techniques results on event-day simulations 60
36 Plot of congestion management techniques results on event-day simulations - outlier removed 61

List of tables

1 Example traffic data record stored in InfluxDB. 23
2 Example event data record stored in InfluxDB. 26
3 Hyperparameter grid for search on MLP architecture 39
4 MLP Model Performance Metrics 49
5 MLP Tuned Architecture 51
6 MLP Feature and Performance Comparison 52
7 Real-World Demand Generation Error Metrics 55
8 Reactive Hyperparameter Search Results and Metrics 57
9 Results of congestion management techniques on 2024 simulations 58
10 Results of congestion management techniques on event-day simulations 58

List of abbreviations

ITS	Intelligent Transportation System
MAE	Mean Absolute Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAPE	Mean Absolute Percentage Error
ARIMA	Auto-Regressive Integrated Moving Average
MLP	Multi-Layer Perceptron
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GNN	Graph Neural Network
CNN	Convolutional Neural Network
VSL	Variable Speed Limit
SUMO	Simulation of Urban MObility
TRaCi	Traffic Control Interface
TfGM	Transport for Greater Manchester
LLM	Large Language Model

Abstract

Traffic congestion significantly impacts urban mobility, increasing travel times, fuel consumption, and CO_2 emissions. Traditional congestion management systems typically rely on reactive strategies, responding only after congestion arises, making them ineffective during sudden surges caused by events like concerts or sports matches. This project addresses these challenges by developing a proactive congestion management framework that utilises advanced machine learning models and integrates external event data.

We built a comprehensive data pipeline that incorporates traffic sensor data, weather conditions, and large-scale event schedules. Event features were encoded using a custom method, and a novel LLM-based attendance estimator was introduced to measure the impact of events. These inputs fed into a Multi-Layer Perceptron model for long-term traffic speed prediction. The resulting forecasts were integrated into a variable speed limit strategy to proactively manage congestion before it occurred.

The event-enhanced predictive models outperformed current long-term forecasting models, achieving a MAPE of 12.44%. Evaluation using SUMO simulations across both regular and high-traffic event days showed that the proactive method consistently reduced CO_2 emissions by up to 3%, as well as reducing or maintaining time loss in traffic, compared to baselines. Our proactive method outperforms reactive methods on event days by 2.5%, demonstrating both technical effectiveness and real-world potential for improving urban traffic flow and sustainability during disruptive events.

Declaration of originality

I hereby confirm that this report is my own original work unless referenced clearly to the contrary, and that no portion of the work referred to in the report has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual property statement

i The author of this report (including any appendices and/or schedules to this report) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii Copies of this report, either in full or in extracts and whether in hard or electronic copy, may be made *only* in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the report, for example graphs and tables ("Reproductions"), which may be described in this report, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv Further information on the conditions under which disclosure, publication and commercialisation of this report, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Project restriction declarations deposited in the University Library, and The University Library's regulations (see `https://www.library.manchester.ac.uk/about/regulations/`).

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Sandra Sampaio, for her invaluable guidance, insightful feedback, and continuous support throughout this research. Her expertise and encouragement have been instrumental in shaping this project.

I also acknowledge Manchester-i for providing the traffic data that made this study possible, a data solution that collects, hosts, and exposes city open-data to a broad set of researchers and end-users operating/interested in urban-related disciplines.

A heartfelt thank you to my friends during this final year for their support, motivation, and well-timed distractions, which helped me stay balanced throughout this journey. Lastly, I am deeply grateful to my family, especially my parents and grandparents, for their continued encouragement, support, and belief in me.

1 Introduction

1.1 Project Context and Motivation

Traffic congestion remains a critical challenge in urban mobility, wasting time and fuel, creating excess CO_2 emissions, and incurring significant economic costs. In 2019 alone, congestion in the United States was estimated to have caused around $190 billion in lost time and excess fuel consumption while also contributing to 2% of CO_2 emissions [1], [2]. These figures highlight the need for better traffic management systems.

In response, researchers have focused on the Intelligent Transportation System (ITS) area, which aims to improve travel reliability, safety, and efficiency. Many traditional ITS solutions work through real-time monitoring and reactive control. These systems respond upon detecting congestion, for example, by adjusting traffic signal timings or speed limits. However, reactive measures often struggle in dynamic settings, especially during sudden traffic surges from large events.

Advances in neural networks and external datasets now enable a shift to predictive control. This shift enables systems to forecast congestion before it happens and take preemptive measures, for instance, adjusting traffic signals, routing flows differently, or alerting drivers in advance. Das *et al.* (2025) has demonstrated that using machine learning for traffic prediction and management can substantially reduce time lost by travellers, offering a “pragmatic, cost-effective solution” that can be integrated with existing infrastructure [3].

Despite the proven potential, predictive approaches are not yet widely used in ITS deployments, especially those that factor in external impacts like major events. Events such as concerts, sports matches, or festivals can rapidly overload urban road networks in ways that typical congestion detectors miss, leading to outsized disruption. Moreover, much of the existing research is limited, often lacking the integration of external data or the ability to provide simulations for evaluating their approaches.

This project aims to address these gaps. As part of this work, we have investigated the availability and quality of multiple data sources, including road traffic observations (such as traffic sensors) and traffic-impacting events (such as football matches and concerts). We have designed, implemented, and tested data collection and processing pipelines, performed detailed profiling of roads and event impacts, and created advanced data engineering pipelines, applying techniques for data cleaning and integration. We proposed and evaluated novel optimisation mechanisms for custom event feature encoding and attendance estimation based on a Large Language Model (LLM). The predictive traffic model was implemented and tested against state-of-the-art techniques using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). Traffic simulations were conducted using SUMO, comparing the effectiveness of proactive strategies against traditional reactive ones, evaluating based on criteria including travel time, CO_2 emissions, and time lost in traffic. By training models on real-world and historical data sources and validating them rigorously through simulations, the project demonstrates how anticipating problems and acting early can lead to improvements in traffic conditions, focusing on reductions in CO_2 emissions and time lost by travellers.

1.2 Aims and Objectives

This project aims to demonstrate that a proactive traffic management prototype, capable of accurately predicting and managing congestion before it occurs, can be effective. To achieve this, we will perform the following objectives:

- **Predictive Traffic Model Development**: Implement a robust machine learning model to forecast future traffic conditions accurately, evaluating performance using standard ML metrics such as MAE, RMSE, and MAPE.
- **Integration of External Data Sources**: Incorporate external data, notably major event schedules, to enhance prediction accuracy.
- **Proactive Congestion Management Algorithm**: Develop an algorithm capable of proactively mitigating congestion by expanding typical reactive congestion management methods.
- **Development of a Realistic Simulation Environment**: Create a realistic simulation environment using real-world traffic data to rigorously test and validate our proactive management algorithm.
- **Benchmarking and Analysis**: Evaluate the performance of our algorithm against existing congestion management techniques within our simulated environment, measuring impact using metrics such as travel time reductions and estimated CO_2 emissions.

1.3 Evaluation Strategy

To evaluate our proposed framework, we focus on two distinct areas: the prediction accuracy achieved by our models and the congestion reduction achieved in simulation.

First, prediction metrics such as MAE, RMSE, and MAPE will be extracted during model development, comparing different models and feature engineering techniques. These results are compared with state-of-the-art research to validate the importance of events in traffic prediction.

For simulations, we will test diverse traffic scenarios, including regular days and days with large-scale events, using the SUMO simulator. We compare key performance metrics from the simulations, including travel time and estimated CO_2 emissions, between baseline, traditional reactive, and our proactive congestion management techniques. Demonstrating measurable improvements in these metrics validates the real-world potential of our prototype.

1.4 Report Structure

Background and Related Work will cover a background literature review on relevant theory for the project (such as forecasting methods, congestion management approaches, and simulation techniques), as well as an overview of related work.

Design and Implementation presents the design choices made for the system and explains the architecture behind them and how they were implemented. This flows through data collection and storage, data analysis, predictive modelling, and traffic simulation. We also discuss the advantages and disadvantages of certain decisions and challenges encountered.

Results and Discussion covers the approach taken towards the evaluation and testing of the system. We present the results of both model performance and simulation outcomes, followed by a critical discussion of the findings.

Conclusions and Future Work reflects on the contributions and achievements of the proactive congestion management prototype and how it could be adapted for use in the industry. We also explore possible future work directions derived from the project's development.

2 Background and Related Work

This section provides an overview of ITSs with an emphasis on congestion management, reviews techniques for traffic forecasting (from classical time-series models to modern machine learning approaches), discusses congestion management strategies (distinguishing reactive and proactive methods such as adaptive signalling, dynamic rerouting, and speed control), outlines the role of traffic simulation tools (e.g. SUMO and TraCi) in evaluating traffic control strategies, and finally identifies research gaps that motivate the development of a proactive, data-driven congestion management system.

2.1 Intelligent Transportation Systems

ITSs are typical in today's smart infrastructure. These refer to the use of advanced information and communication technologies in transport infrastructures to reduce traffic congestion, improve road safety, and increase the overall efficiency of traffic networks. ITSs have quickly evolved over the past few decades from isolated, low-capability systems to integrated networks of adaptive systems with much more significant impact. These use a variety of sensors, communication forms, and efficient computing capabilities for traffic management.

Deployment. These modern ITSs are typically deployed with an array of road sensors (such as inductive loops, cameras, and radar) and communication mechanisms. Centralised traffic management centres then enable the continuous monitoring and control of the networks [1]. For instance, an example of an ITS is detectors on roads that feed real-time traffic data to the control centre, where operators can check conditions (often using CCTV feeds) and then respond accordingly by adjusting control devices such as traffic light timing [4]. A diagram of a typical ITS is shown in Fig. 1.

Modern ITS systems can automatically implement congestion management strategies. These include adaptive signal control systems that adjust traffic light timing in response to traffic conditions,

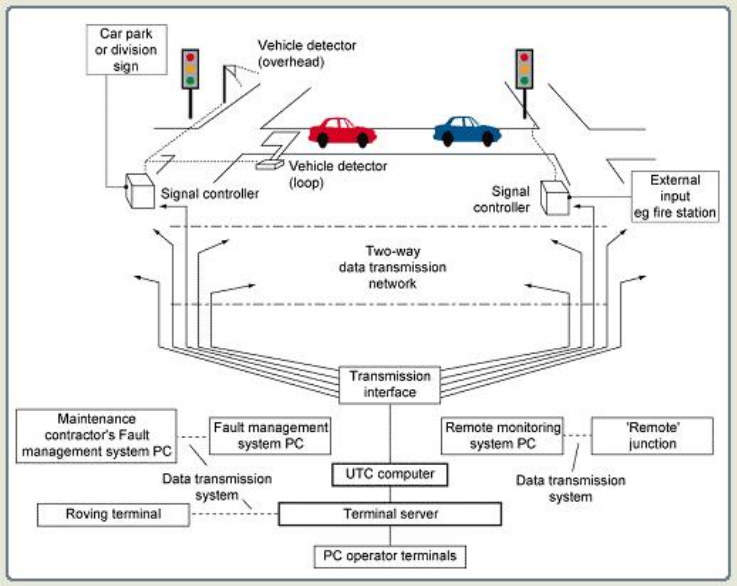

Fig. 1. The flow of information in SCOOT based UTC system - an example of an ITS from [5]

ramp metering systems that regulate the entry of vehicles to highways, or traveller information systems that inform drivers about road conditions and alternative routes [6].

Performance. These technologies have been shown to reduce delays and improve network performance; for example, adaptive signal control has been documented to smooth traffic flow, reducing delay by optimising signal timing in real-time [7]. In many congested metropolitan areas, multiple ITS technologies are deployed in conjunction – a traffic management centre may simultaneously utilise detection systems, coordinated signals, dynamic speed limit signs, and route guidance, each addressing different aspects of congestion. The overall trend in ITSs is toward increasingly integrated and intelligent systems, allowing a focus on more proactive traffic management approaches.

Despite these advancements, most current ITS implementations still function reactively, responding to congestion or incidents as they occur. These typically detect congestion (via connected sensors) and then trigger control measures to mitigate it. While these approaches are efficient, reacting after queues have formed may not always be fast or compelling enough to prevent traffic buildup and congestion. This limitation in the most used current ITSs motivates the incorporation of predictive analytics and proactive control, as discussed later in this section.

2.2 Traffic Forecasting

Accurate traffic forecasting is an essential mechanism to allow for advanced traffic management. While not a standalone solution, traffic forecasting enables proactive traffic control by informing decisions. Many real-world applications currently use predictions in some form. For example, navigation services (Google Maps, Waze) use traffic forecasting to predict travel times on routes to inform drivers of route choices [8], and traffic authorities use predicted traffic volumes to reac-

tively enable measures such as ramp metering or variable message signs. However, fully taking advantage of the power of traffic forecasting within ITS is still a work in progress.

2.2.1 Time-Series Forecasting (ARIMA)

Traditional approaches to traffic prediction rely on time-series modelling techniques. Among these, the Auto-Regressive Integrated Moving Average (ARIMA) model and its variants have been widely adopted for traffic flow and travel time prediction, albeit only for short-term predictions. These older models are based on more traditional statistical methods and have proven to capture recurring patterns in historical traffic data and are often used as a baseline for traffic forecasting [9].

In practice, however, these types of models (purely statistical) have severe limitations: they are essentially only useful for short-term forecasts; they assume linear relationships in the time series and, therefore, struggle to model the complex, nonlinear structure of trends inherent in the transport networks.

2.2.2 Machine Learning and Neural Network Techniques

In recent years, as with most domains, machine learning approaches have gained dominance in traffic forecasting due to their ability to model much deeper trends and accept many data sources. The feed-forwards Multi-Layer Perceptron (MLP) is a common approach that learns direct mappings from past data. A simplified architecture diagram of the MLP is shown in Fig. 2.

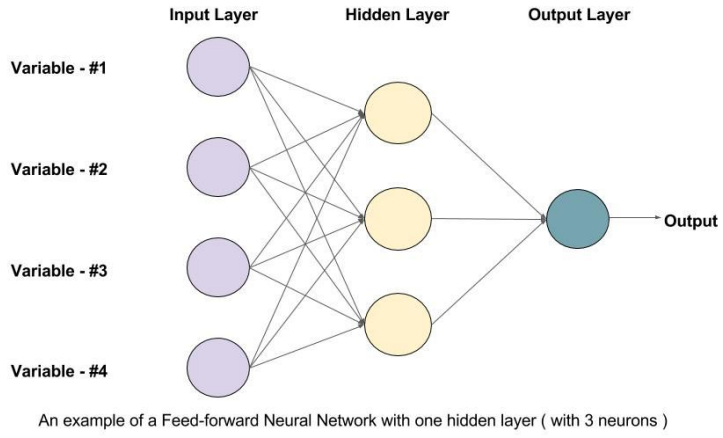

Fig. 2. A simplified MLP architecture diagram from [10]

Multi-Layer Perceptron (MLP). MLPs and other artificial neural networks have been successfully applied to predict traffic conditions, showing improved accuracy over the traditional models. For example, a team of researchers compared several predictive models (including linear and

polynomial regression, radial-basis function network, and a feed-forward neural network) for congestion prediction based on several inputs such as traffic waiting times, time-of-week indicators and holidays [11]. They achieved the highest accuracy with the neural network at about 97.6%, outperforming all the other techniques. Their research demonstrates the effectiveness of even simpler neural networks (such as MLPs) when used with rich input features.

Recurrent Neural Networks (RNNs). More sophisticated machine learning models may improve predictive performance for specific applications. In particular, Recurrent Neural Network (RNN)s and their variants, like Long Short-Term Memory (LSTM) networks, have shown great performance in traffic forecasting. A simplified architecture diagram of a LSTM network is shown in Fig. 3. LSTMs are designed to retain long-term dependencies and handle time-varying patterns such as seasonally well. However, these networks suffer from vanishing gradients and difficulties retaining distant dependencies, making them inefficient for long-term forecasting, just like the ARIMA models. Numerous studies have found LSTM-based predictors to outperform traditional models such as ARIMA on various traffic metrics. For instance, Katambire *et al.* (2023) deployed an LSTM and an ARIMA model to forecast traffic volumes at a busy urban junction. The results indicated that the LSTM was the best-fitting model for predicting monthly traffic flow, exceeding the accuracy of the ARIMA approach [9].

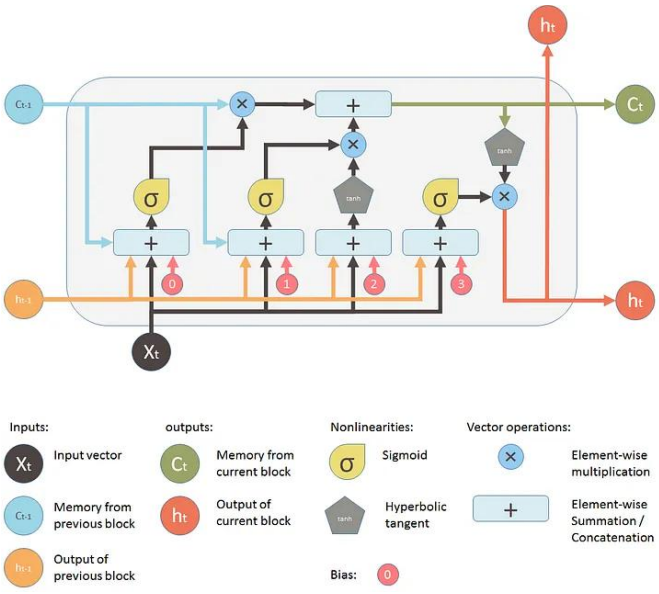

Fig. 3. A simplified LSTM architecture diagram from [12]

2.2.3 Recent Advancements

Hybrid Aproaches. Beyond individual models, researchers have recently explored hybrid and deep learning architectures to push predictive accuracy. One strategy is combining two models to their strengths, such as coupling an ARIMA model (that excels at capturing linear trends) with an MLP (great at nonlinear patterns). This type of combination can improve prediction accuracy compared to the standalone models [13].

Deep Learning Approaches. For long-term forecasting, recent advanced approaches incorporate spatial information, treating the traffic network as a graph of sensors and road segments. When doing so with Graph Neural Network (GNN)s or Convolutional Neural Network (CNN)s on spatio-temporal data, they show excellent adaptive performance to model interconnected congestion patterns by learning how congestion at one location influences another [14], [15]. Broadly, machine learning techniques (from basic MLPs to complex LSTM and graph networks) have significantly improved predictive accuracy and are increasingly being integrated into traffic management systems [16].

2.3 Congestion Management Approaches

Traffic congestion management strategies can typically be classified into two categories: reactive and proactive approaches. In the next subsections, we will review examples of each, highlight the traditional ITS approach of reacting to congestion, and discuss how emerging methods aim to predict and avert congestion before it occurs.

2.3.1 Reactive Methods

Reactive congestion management is the most traditional strategy, responding to traffic conditions or incidents as they occur to reduce congestion or contain it after it has already developed. Most conventional ITS implementations fall into this category. These systems continuously monitor traffic (for example, through loop detectors and cameras) and trigger control actions when congestion is detected, or traffic density thresholds are exceeded. A typical example is adaptive signal control, where traffic signal timings (green/red phases) are adjusted based on volumes at intersections. If an intersection becomes congested, the controller can extend the green time on the most congested approach to clear it. This type of system has been consistently deployed worldwide and has been shown to reduce delays by smoothing the flow through intersections [1]. Other examples of these methods include [1]:

- **Variable Speed Limit (VSL):** Reduced speed limits prevent shockwave formation when congestion occurs.
- **Dynamic Rereouting:** Drivers are alerted to congestion through dynamic signs or GPS systems and rerouted to avoid the congestion.
- **Ramp Metering:** Controls the flow of vehicles entering a motorway to improve traffic flow.

These reactive methods are significant components in current ITSs and have proven effective to an extent: by responding in real-time, they can reduce the duration and severity of congestion. However, the major limitation of these approaches is that they wait for congestion to occur before taking action. In scenarios of fast-rising demand (for example, the sudden influx of cars after a stadium event ends), purely reactive control might not be sufficient or act too late, as the congestion would have already built up by the time the system intervenes.

2.3.2 Proactive Methods

Proactive congestion management aids in covering the inefficiencies of reactive methods by predicting or anticipating traffic conditions and taking control actions in advance. This approach uses predictive analytics (like the earlier forecasting models) that are tightly integrated with traffic control algorithms. The systems continuously generate forecasts of traffic state for various, often critical, intersections and roads, looking slightly ahead. If congestion is predicted to build up soon, the system proactively implements control measures before the congestion fully occurs. These usually include the same strategies as in reactive control (signals, routing, adaptive speed limits), but the timing is guided by predictions rather than current measurements.

Using the same example of predictive signal control, the system would preemptively adjust green phases to accommodate surges just before they occur, reroute drivers in advance of a stadium match ending, or reducing speed limits just before a high influx of traffic.

Feasibility. The feasibility of proactive traffic management has recently greatly improved due to advances in both computing power and predictive models. Some research prototypes have successfully demonstrated that integrating traffic prediction control is possible and beneficial. For example, a recent study funded by the U.S. Federal Highway Administration developed a prototype system with traffic prediction and a reinforcement learning engine to recommend traffic control interventions [17]. This system could analyse real-time data and learn optimal interventions. At a city scale, the city of Groningen in the Netherlands piloted a similar system that combined traffic prediction with an adaptive control platform [18]. These examples demonstrate the potential of proactive methods to improve congestion management in continuously denser urban environments.

Reactive & Proactive Interplay. We note, however, that proactive strategies do not replace reactive ones but should enhance them. Even in proactive systems, reactive adjustments will still be needed for unforeseen traffic events (such as crashes). The proactive congestion management area is still emerging, and deployments are not yet typical, mainly due to the challenges in obtaining reliable predictions and the complexity of the decision-making algorithms required. Nonetheless, as we will discuss later, the growing computing capabilities, abundant available data, and modern machine learning algorithms provide a strong direction to enable proactive congestion management algorithms.

2.4 Traffic Simulation

Before new traffic control strategies can be implemented on real roads, they must be thoroughly tested. Traffic simulation tools play a crucial role in this process, allowing researchers and traffic engineers to model real-world traffic dynamics in a virtual environment. This allows new approaches to be tested without risk or costly field experiments. These tools replicate the behaviour

of individual vehicles and their interactions on road networks, providing detailed insights into how congestion develops and to what extent control measures are effective.

A variety of traffic simulation software packages have been developed, ranging from commercial products like VISSIM and Aimsun to open-source platforms like SUMO and VEINS (which couples SUMO with a communication network simulator OMNeT++).

2.4.1 Simulation of Urban MObility (SUMO)

Simulation of Urban MObility (SUMO), in particular, has become popular in academic and research settings as a flexible, microscopic traffic simulator capable of handling large networks [19]. It allows the modelling of each vehicle's movement through road networks with user-defined routes and supports the simulation of various transportation modes (such as cars, buses, and pedestrians). An example SUMO simulation is shown in Fig. 4.

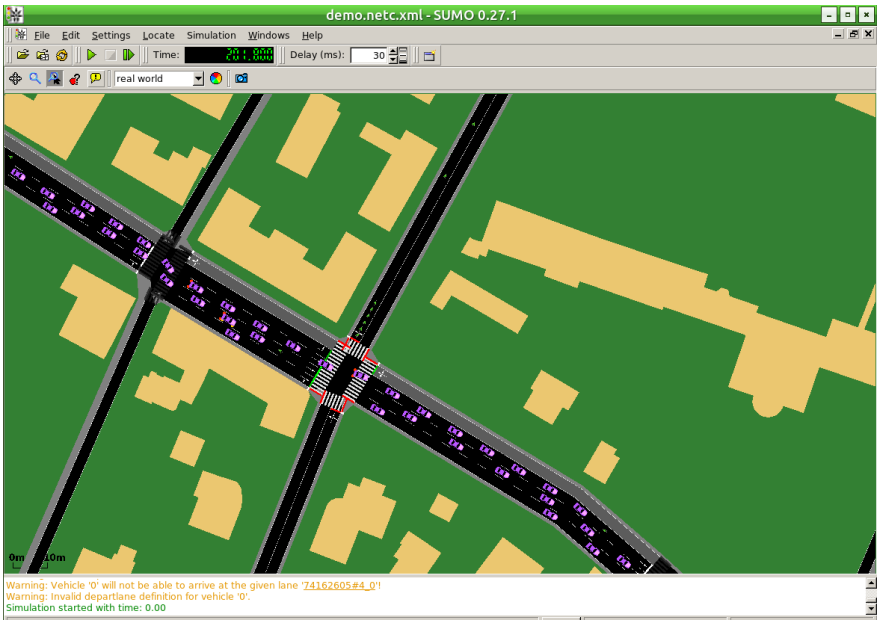

Fig. 4. An example SUMO simulation from [20]

One of the key advantages of SUMO is that it is open-source and highly customisable. It is, for example, frequently used to evaluate adaptive signal control algorithms by simulating a city's intersection and testing how the control algorithms impact queues and travel times [19]. Crucially, SUMO provides not just average performance metrics (such as total network delay) but also fine-grained metrics of traffic, which help tune parameters and investigate issues.

2.4.2 Traffic Control Interface (TraCi)

An important feature of SUMO (and several other modern simulators) is the ability to interact with the simulation in real time through an API. In SUMO's case, this is facilitated by Traffic Control Interface (TRaCi), which is a network-based interface that allows an external program (such as using Python) to query and control the state of the simulation at each time step [19]. Using this API,

we can retrieve fine-grained information from the simulation and send actions back in response as the simulation runs.

This capability makes SUMO powerful for testing closed-loop traffic management algorithms like the ones in this project's scope. As a concrete example, the proactive traffic management prototype mentioned earlier (by the U.S. Federal Highway Administration) was evaluated using a similar setup of emulated real traffic scenarios, allowing researchers to verify that their algorithms led to improvements compared to a baseline [17].

2.5 Research Gaps

The literature reviewed above shows strong progress in traffic sensing, modelling, and control, but there remain key gaps, particularly in how these domains are combined in real-world systems.

2.5.1 Predictive Model Integration

ITS deployments increasingly include complex monitoring and reactive control. However, most systems still rely only on real-time data, with limited use of predictive models for proactive congestion management. For instance, motorway ramp meters typically respond to current occupancy, and adaptive signal systems like SCOOT or SCATS adjust to measured queues but do not forecast future congestion [21]. Despite research showing the benefits of predictive control, deployments are rare due to integration challenges and a reluctance to act on uncertain forecasts. Closing this gap requires not just accurate prediction but also practical evidence that such predictions can be trusted in live environments.

2.5.2 Importance of External Contextual Data

A second gap lies in incorporating external contextual data, particularly large-scale event information, into traffic prediction and control. Congestion is not driven by traffic dynamics alone. Events such as football matches and concerts can cause significant, location-specific spikes in demand. While some research has explored this, such as Kwoczek *et al.* (2014) who showed a 35% improvement when incorporating event data in Cologne [22], these efforts often rely on ad hoc data collection and are rarely operationalised in ITS. Most real-world systems still handle such scenarios manually, if at all.

2.5.3 Project Positioning and Justification

This project directly addresses both gaps. We develop a predictive model that integrates traffic sensor data, temporal features, and large-scale event information to forecast congestion ahead of time. This is combined with a proactive control method, specifically, a VSL-based congestion management strategy—that uses these predictions to intervene before congestion forms. This

pairing is motivated by prior work showing the benefits of contextual data, and by our evaluation, which demonstrates that integrating prediction and control produces measurable improvements, confirmed by simulation. Further design decisions, including model structure and control logic, are detailed in **Section 3**.

3 Design and Implementation

In this section, we will present design choices, explain the architecture behind them and how they were implemented. The architecture flows through data storage and collection, data analysis, predictive modelling, and traffic simulation. We also discuss the advantages and disadvantages of certain decisions and challenges encountered.

Details of the hardware and software used for this project to ensure reproducibility are available in Appendix C.

A note to the reader: Due to the research-oriented nature of this project, design and implementation were closely intertwined. Therefore, they are presented sequentially rather than in separate sections.

3.1 Data Storage

A dedicated database solution is essential for the project due to the large amount of data that must be collected and organised in a single, easy-to-access location. The data collected ranges from historical traffic records to event schedules, which must be stored consistently. As we are working with time-series data, a database system specialised in this type of data will be the most efficient, which, also being a NoSQL database, allows for our varied data structure. The most important requirement is a solution that efficiently handles temporal queries and supports quick querying over time windows.

Choice and Implementation InfluxDB [23] was selected for suitability with timestamped data and easy integration with a data processing pipeline. InfluxDB was deployed on an external server to support our requirements, allowing us to access it remotely and separate the data workload from our main machine, leading to higher performance.

3.2 Data Collection Pipeline

Here, we describe our data collection pipeline. For the project, we require three types of data: traffic, weather, and event schedules. While focused on creating a prototype, we collected data from one city: Manchester.

3.2.1 Traffic Data

We require historical traffic data from Manchester. Most common sources of traffic data (Google, TomTom [24]) are either closed-source or only provide real-time data. However, A prior university project was completed as part of an EU-funded project to place sensors around Manchester and make these openly available, Manchester-I [25]. They provide data across traffic, weather, air quality and hydrology. For our purposes, we will only collect traffic data, for which sensors are placed across the city that provide both speed and count data split by lane direction and vehicle type, providing observations (data points) every 5 minutes, up to about 4 years of historical data.

Data Collected Manchester-I has several traffic sensor deployment projects. Out of these, the Drakewell project has deployed many sensors across Manchester in partnership with Transport for Greater Manchester (TfGM) [26]. This project provides the most significant number of sensors and allows us to work with a single, more interpretable data structure. We therefore decided to collect all the data from the sensors on this project, filtering to car traffic. We present an example of the data structure in Table 1.

Table 1. Example traffic data record stored in InfluxDB.

Field	**Value**
_measurement	Traffic
_time	2023-12-07T12:40:00.000Z
value	72.4
Tags	
platform_id	drakewell__1328
platform_label	Drakewell 1328
platform_description	Automatic Traffic Counter
location	[-1.9739, 53.582]
sensor_id	avgspeed_sw
sensor_type	vehicle-speed
unit	kilometre-per-hour

Collection Implementation We provide an architecture diagram of the collection implementation in Fig. 5. Manchester-I provides an API from which we can fetch the data. We use this API to fetch the data using a Python script, which converts it into an appropriate form for InfluxDB and writes it to the database. The structure of data in Manchester-I (Platforms -> Sensors -> Observations) requires an iterative and hierarchical approach to enable data fetching. Platforms are locations equipped with one or more sensors (for different directions or vehicle types) that capture time-series observations of speed or counts.

During development, we ran into a challenge for which two optimisations were then developed. The challenge was the time taken to perform the data collection. Since each list of observations was quite large, each of these took a long time to process. To improve this, we implemented concurrency to speed it up. The second improvement was to implement checking for existing data in our database. This allowed us to perform incremental runs of the script to finish the data collection instead of requiring it to run in one go.

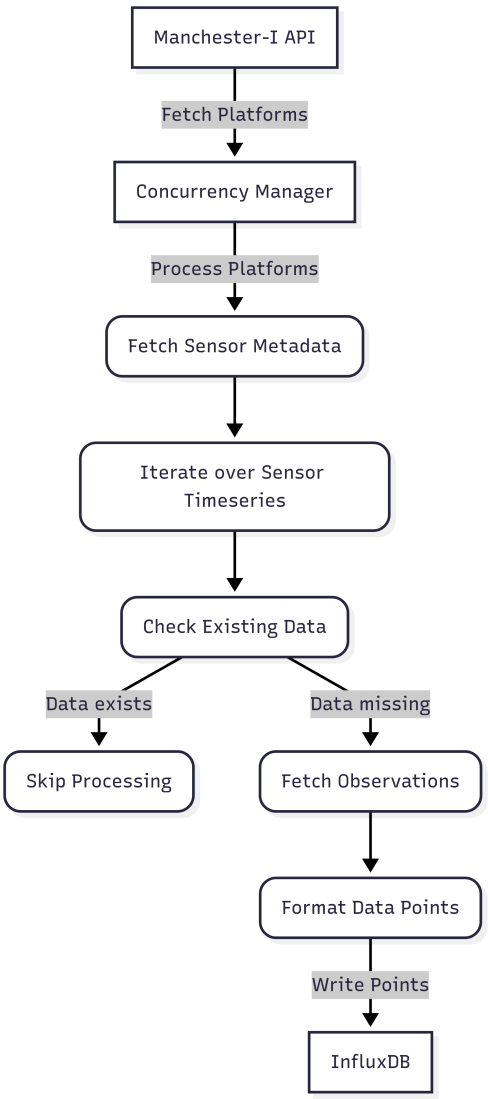

Fig. 5. Architecture diagram of Manchester-I data collection

3.2.2 Weather Data

The literature showed the possibility of a higher prediction accuracy when including weather data. Therefore, we also collected weather data. Specifically, literature on the impact of weather on traffic specifies precipitation, wind and visibility as the most important factors [27] in traffic. For this, we decided on the Open-Meteo API, which provides free access to historical data on these fields by location (coordinates) [28].

Collection Implementation As this data is quite simple and uses an easy-to-access API, storing it in our database is unnecessary. Therefore, we call the API to collect the data from our DataLoader during runtime.

3.2.3 Event Data

Event data serves a large purpose for this project. As events are proven to have a significant impact on traffic, this data is important for our prediction models and to further test the impact of congestion management techniques. As we focus on Manchester data, we collected data from the city's most prominent venues, which will have the most significant impact, and we can correlate venue locations with sensors. We chose two football match venues and two concert venues: Etihad Stadium, Old Trafford, Co-Op Live, and AO Arena.

Challenges Designing to collect this data is difficult for several reasons:

- While some venues provide schedule data on their website, this is usually in the future and not historical data.
- Some APIs exist to gather historical data of events, such as API-Football [29]. However, these are all paid and were not a convenient option.
- There was no centralised source for all these venues from which we could gather data.
- Some venues do not provide the start time for events on the website.

Data Gathering Design With this in mind, the design for gathering the event data required is achieved using the following two elements. We used the Wayback Machine [30] to access historical versions of venue websites. Regarding the actual data gathering itself, as there is no API or centralised source we can use, we use a scraping solution to extract the public data required from the HTML data of websites pertinent to each data source. For venues where start time data was unavailable, we used an approximate start time for that event category in the UK.

Collection Implementation We provide an architecture diagram of the collection implementation in Fig. 6. We scraped the required data from four sources: Concert Archives [31], ESPN [32], New York Times [33], and Manchester United [34]. We collected data starting from 2020 (approximately the same period we have available traffic data) for every event at the four venues. For this task, we constructed scraping scripts for each source, using the BeautifulSoup Python library to scrape the data and then format and write it to InfluxDB. We present an example of the data structure in Table 2. The "estimated_attendance" field that can be seen in this data is just a placeholder at this point, and its use is explained in **LLM-Based Attendance Estimation**.

Some sources provided the required data in a readily accessible JSON structure, while others had to be extracted from the HTML, requiring slightly different approaches. One challenge encountered during scraping was robot detection. Manually downloading pages needed to be performed where scraping failed, before being processed by our scripts.

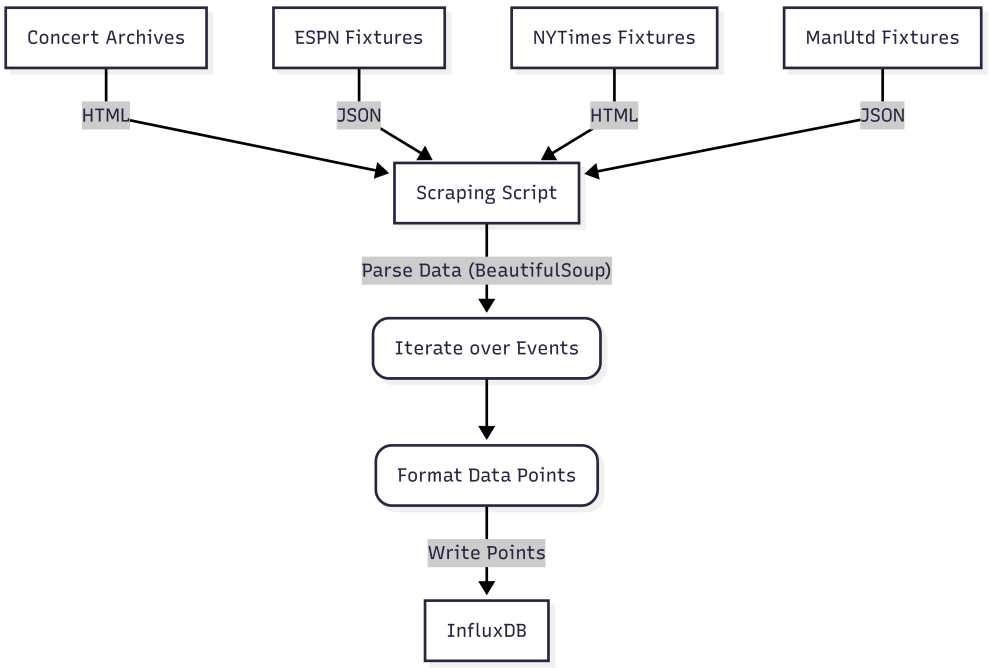

Fig. 6. Architecture diagram of event data collection

Table 2. Example event data record stored in InfluxDB.

Field	**Value**
_measurement	Event
_time	2024-05-13T04:00:00.000Z
_field	estimated_attendance
value	0
Tags	
venue	match
event_type	OldTrafford
event_name	Arsenal @ Man Utd
location	[53.463056, -2.291389]

3.3 Data Analysis

We will now analyse the collected data. This aims to form our research base, extract knowledge from the data, and crucially provide generalised guidance on formulating our ML models. We will focus on one venue (Etihad) and one related sensor (drakewell-1163). As our data set is extensive, we will investigate patterns from this combination, allowing us to generalise these findings to our whole data set. The Jupyter notebook with the complete analysis is available in `traffic-analysis.ipynb`, located in the `data_analysis` directory of the repository (Appendix A).

3.3.1 Sensor and Venue Locations

We collected data from 48 platforms, each of which provides counts and speeds for each lane direction. A plot of all these locations and the locations of our chosen venues is available in Fig. 7, with a zoomed-in version containing venue labels in Fig. 8.

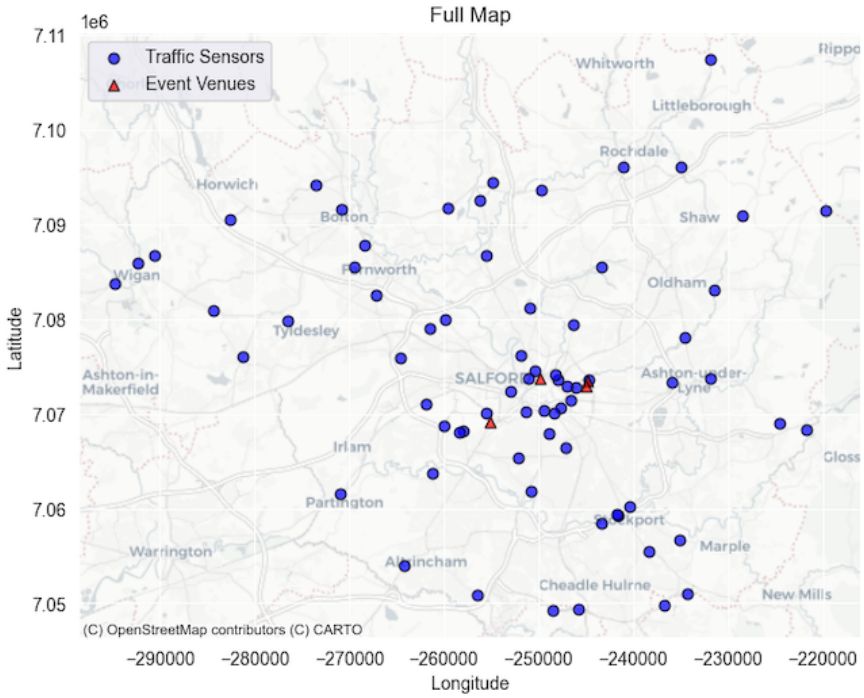

Fig. 7. Zoomed-out plot of traffic platforms and venues collected

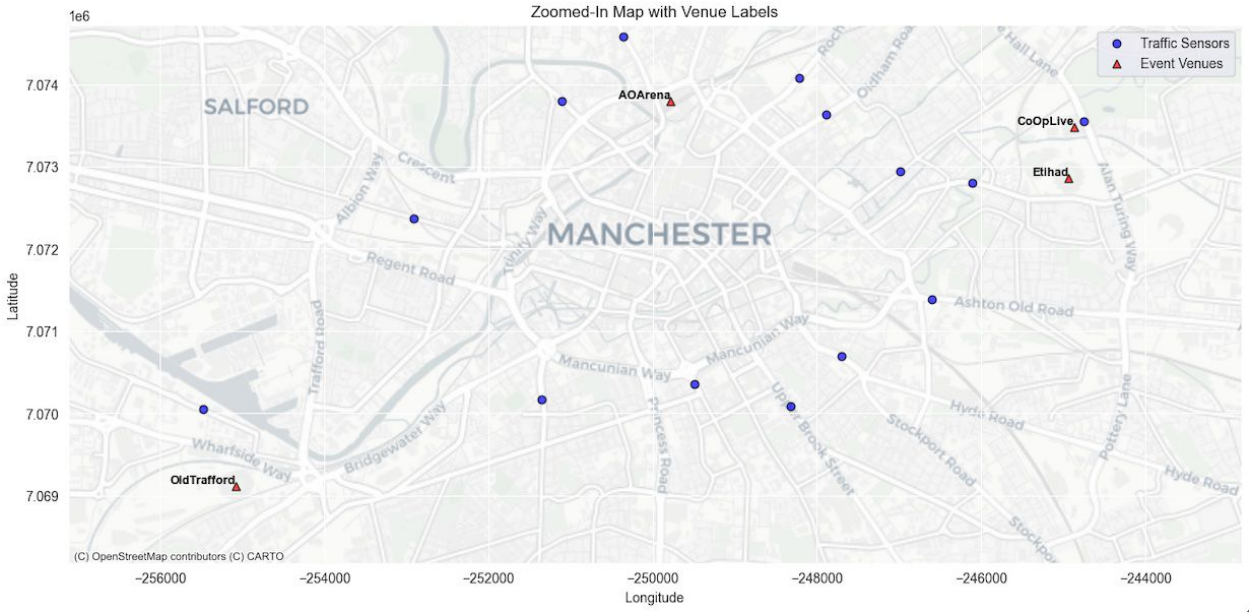

Fig. 8. Zoomed-in plot of traffic platforms and venues collected

3.3.2 General Analysis and Preprocessing

Upon exploring the traffic data, several observations can be made. The data appears consistent with expected patterns. However, notable gaps have been identified, primarily involving large recent missing data segments, despite earlier periods showing continuous data. An example is shown in Fig. 9. Traffic speed data is precise, consisting only of average speeds recorded separately for each direction, while traffic counts are subdivided by vehicle type. Given this project's scope, the analysis will be restricted to car counts and average speeds.

During preprocessing, two primary issues were observed: zero values in speed data, likely indicat-

ing sensor errors, and prolonged intervals of missing data, potentially caused by more significant sensor malfunctions. Therefore, large missing segments will be excluded entirely from the analysis, as interpolation could result in inaccuracies, and zero-valued speed entries will be removed. Count data will not require this treatment, as zero values represent valid observations and do not show sudden drops. An example is shown in Fig. 10.

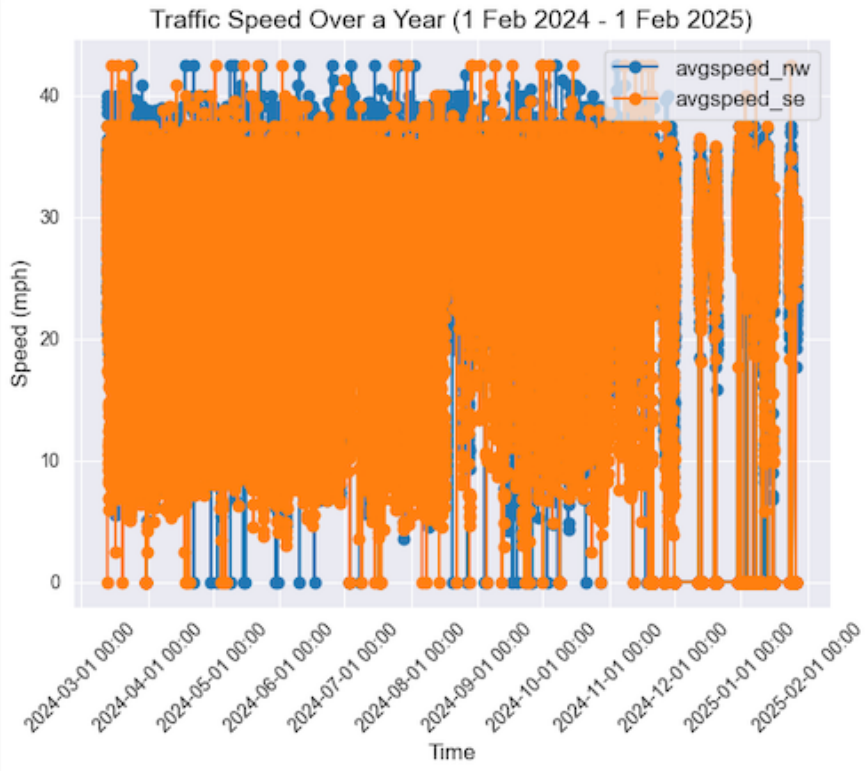

Fig. 9. Plot of traffic count over a year

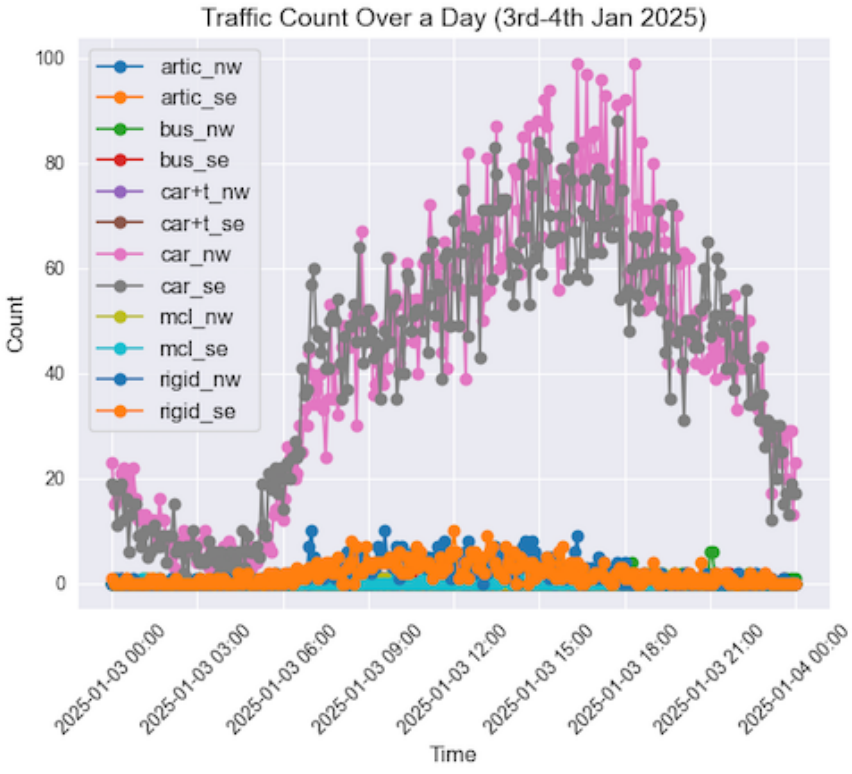

Fig. 10. Plot of traffic speed over a day

3.3.3 Correlation with Events

Next, we investigate the correlation of events with traffic data. For this, we compiled plots of the traffic data around each event in the collected data. An example of this Plot is available in Fig. 11, where the red dashed line represents the event's start time. Here, we can identify that speeds tend to decrease gradually before the event time and increase afterwards before decreasing again after a few hours (presumably when the event ends). This pattern confirms that congestion occurs around events, supporting the model's relevance. The count metric shows similar but inverse trends, but less descriptive and pronounced, which are less useful for our purpose.

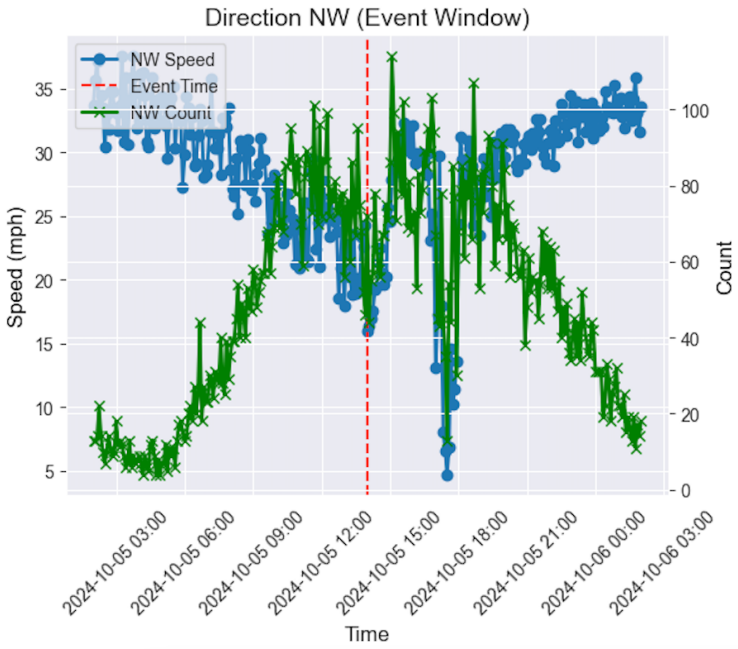

Fig. 11. Plot of traffic data around an event at Etihad

3.3.4 Traffic Speed vs. Count Analysis

Given the analysis in prior sections, we complete a final analysis on speed vs. count for traffic. This results from a correlation analysis that resulted in Fig. 12 and an average correlation coefficient of -0.43. This shows that while a negative correlation is found, it is not very strong. This is expected and can be caused by external factors such as weather conditions, roadworks, or accidents, which can affect traffic speed without significantly impacting traffic count. Due to the nature of this project, reasons given earlier when performing this analysis directly in comparison with events, and given these observations, we will use traffic speed as the primary metric in our models, as this is more descriptive and has a clearer relationship with our project goal.

3.4 Feature Engineering

In this section, we will explore the feature engineering work performed for the project. Feature engineering transforms raw data into formats suitable for machine learning models [35].

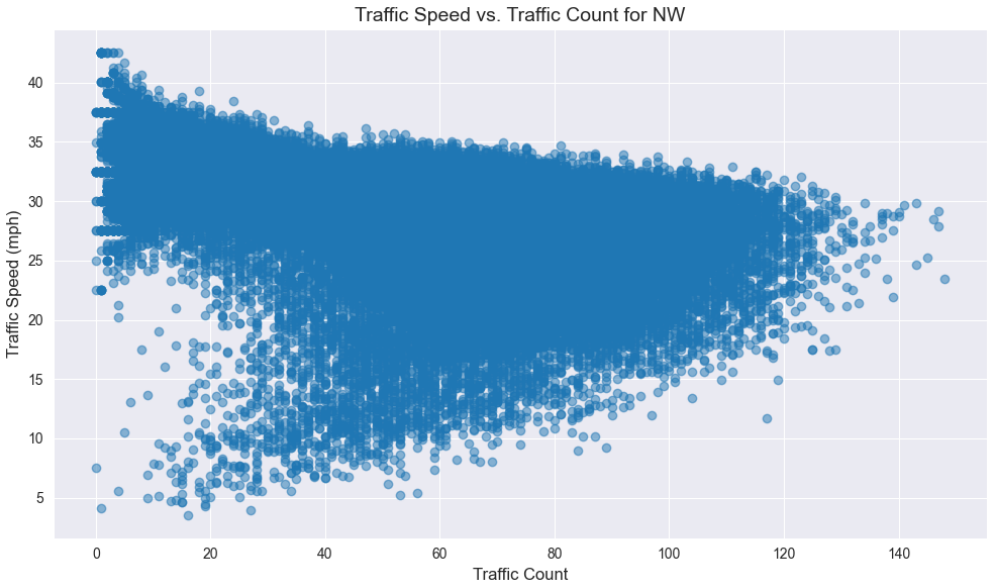

Fig. 12. Plot of correlation between traffic speed and traffic count for one direction

3.4.1 Traffic Features

Traffic features are relatively simple. We can use only the data's time as its feature. To achieve higher performance, we designed two different encodings.

Basic Time Encoding In the typical time encoding, we split the timestamp into several valuable features that the model can use to identify trends such as seasonality, differences in time of day, and weekdays vs. weekends. The complete list of features derived from the timestamp we use is as follows: year, month, day, weekday, hour, and minute.

Cyclic Time Encoding In cyclic time encoding, we use the same features but encoded in a cyclic manner (using sin and cos values) to model the cyclic trends in this data based on existing research [36]. We do this for the month, weekday, hour, and minute features, which allows the model to understand continuance in, for example, minutes 59 to 0. The year and day features will remain non-cyclic, as years are continuous and days are irregular depending on the month.

3.4.2 Weather Features

As the weather features are simple and direct, we pass the values straight through: precipitation, wind speed, and relative humidity (which identify conditions for fog).

3.4.3 Event Features

Event Features are the most complex due to our focus on event impact for this project. We explore two different encodings of event schedules as inputs to our models. One is a simple binary event

encoding, while in the second section, we will design a custom encoding.

Binary Event Encoding In this encoding type, we utilise a single feature to represent an event affecting a time point. We use two variables to control this:

- The time difference of the event to a time point: x
- The location distance between the event venue and a traffic sensor: y

If an event occurs within x time of a particular timepoint and the distance to it is smaller than y, then we encode this feature as True; otherwise, it is False.

Custom Event Encoding Here, we design a more complex encoding to incorporate more information into the models that could lead to higher prediction accuracies. The general idea is to provide the actual information of the time difference and location distance, the type of event, and the same binary encoding as above. This leads to four features:

- The location distance to the event
- The time difference to the event
- The event type (for our data, “match” or “concert”)
- The binary representation, as in the earlier section

In this encoding type, a particular design decision occurs regarding which event to select when multiple events occur on the same day. As we are dealing with traffic data, we assume that the event happening closest to the venue will be the one with the most impact, and therefore, we compute the remaining features for this event. With this type of encoding, we keep the notion of a maximum time difference and distance, over which we ignore that event.

Implementation The implementation of the feature encodings for events is as follows. For every point of traffic data being used, we iterate over every event in that same timeframe, followed by:

1. Computing the time difference between the event and the traffic data point
2. Confirming if the time difference is within our maximum. If not, discard.
3. Computing the distance difference between the event and traffic data point using the GeoPy package.
4. Confirming if the distance difference is within our maximum. If not, discard.

The hyperparameters of time and distance, x and y, were determined through experimentation. The maximum distance was set to 3 km, as we found no difference in accuracy from this point onwards. The maximum time was set to 12 hours, as we found events to have an effect up to then, after testing 4, 6, and 10 hours.

Once this is computed for all events, we find the event with the minimum distance and allocate the features accordingly. There are two essential aspects to note in the implementation. Firstly, the use of the time difference is not absolute; therefore, we provide the model with both the magnitude and direction of the time difference, allowing it to understand the differences between traffic before and after the event (such as traffic peaks at the event's end). Secondly, as features cannot be set to null when an event does not impact a traffic data point, we set these to a high default number to allow the model to understand that there is no impact. This is used for both the distance and time features.

Optimisations Since the feature engineering of events is highly computationally intensive ($O(n^2)$ runtime), we implement two significant optimisations to ensure this process runs in an acceptable time.

1. We use NumPy vectors to represent the data and intermediate arrays, precomputing the initial arrays whenever possible.
2. We implement concurrency in this step to allow the feature engineering to run on multiple cores in parallel.

3.4.4 LLM-based Attendance Estimation

This section proposes a novel Large Language Model (LLM)-based attendance estimation pipeline to enhance the event features. A problem faced when using event-based features is not having any information on the actual attendance of an event. While each venue has a specific capacity, events may not continuously operate at them. Given that the traffic impact is generally directly correlated with the number of attendees at an event, knowing this number could improve predictive accuracy.

Pipeline Design An architecture diagram of the proposed system is available in Fig. 13. The pipeline gathers the event data collected and uses this to generate a prompt that will query an LLM, along with context on the task, to generate estimates of each event's attendance. The LLM is given the event name, type, venue, and date. The prompt used for this is given in **Appendix B**. The LLM returns the estimates in JSON format, which are extracted and then used to update the event data points in InfluxDB to contain this estimate.

Implementation This pipeline is implemented through a Python script that iterates through every event in the database and uses the above process to update them with the estimate. To interact

with LLMs, we used OpenRouter, a unified API that allows access to many models with no or low cost [37]. For the model, we used Gemini 2.0 Flash Lite Preview [38], which was available at no cost, runs in an acceptable time (compared to a reasoning model), and returns estimates that, when manually reviewed, seem very reasonable for the venues and events in question.

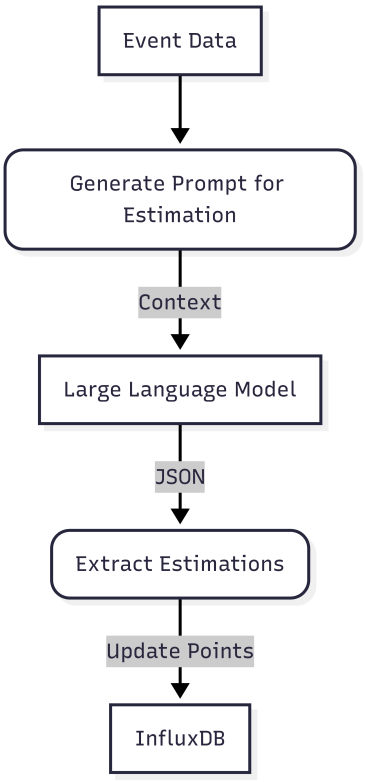

Fig. 13. Architecture diagram of LLM-based Attendance Estimation

3.5 Predictive Modeling

We now explain the modular approach towards the design of the predictive modelling system. This system is designed using Python. Each component has a clear responsibility: `batch_training.py` handles training across multiple sensors, while `train_model_nn.py` focuses on neural network training for a single sensor. `predict.py` generates predictions from trained models, and `model_training.py` defines the model architectures and training routines. Data ingestion is managed by `data_loader.py`, and `feature_engineering.py` encapsulates all transformations for traffic, event, and weather features. Additionally, `create-estimations.py` estimates event attendance using LLMs (as described in the earlier sections), and `config.py` forms part of a tool that centralises configuration and key management. This modular structure enables the clean separation of concerns, easier experimentation, and flexible scaling across different modelling scenarios.

3.5.1 Design Decisions

Flow Identification First, we discuss the identification of flow. As the data collected provides us with the traffic speed for a specific sensor in both lane directions, we must choose between

combining the flow and predicting the flow irrespective of direction or treating these as two separate flows. Given the data analysis performed, where we can see different trends depending on the direction (for example, traffic incoming to a venue vs outgoing), we will treat these separately to provide greater accuracy.

Model Design Additionally, there are two options regarding model design. Firstly, a model that performs predictions across all the sensors as one big predictive model that collects all the information and then predicts based on the features in conjunction with the location of the requested sensor. Secondly, training separate models for each sensor and direction would allow for many smaller models to predict traffic at that particular sensor and direction.

While having one large model would allow for generalisation based on location, allowing nearby sensors to share traffic patterns and recognise the effects of each other, we decided to perform the training of separate models due to the following reasons:

1. The data we have available for traffic is limited in locations. Most of the sensors are spread around the city, meaning the advantage stated above would not hold.

2. Using smaller models for each sensor allows focusing specifically on patterns at each location while maintaining a much smaller, more efficient model.

Model Choices In our exploration of models for traffic speed prediction, we evaluated a wide range of neural network architectures based on our background research. These included MLPs, RNNs, LSTMs, CNNs, GNNs, and transformer-based models. These models were considered to be suitable for capturing temporal or spatial dependencies, the nature of our data, and computational requirements. We noted that while RNNs and their variants, like LSTMs, are excellent at handling sequential data, they struggle with long-term dependencies. This results in them only being able to handle short-term forecasting. CNNs and GNNs, on the other hand, require lots of data or interconnected locations to work correctly. Due to the nature of our data, these were ruled out. Transformer-based models, although state-of-the-art and very powerful, were ruled out due to their high computational cost, which may be impractical for an ITS with limited processing resources.

Given these considerations, we selected three models to implement. The first is ARIMA, the classical time-series forecasting model typically used as a benchmark. For neural networks, we settled on LSTM and MLP architectures. LSTM was chosen for its ability to capture trends in data better; however, it can only forecast short-term. MLP was selected for its efficiency in processing features (a significant focus of this project) and long-term forecasting. This selection allows us to balance computational efficiency with predictive accuracy, given data availability. Regarding short-term vs long-term forecasting, we choose to implement these two models with different types of forecasting in order to test them before deciding which to use for our congestion management design.

3.5.2 Traditional Statistical Model (ARIMA)

An architecture diagram for this model is provided in Fig. 14. This architecture is quite simple, as is the model itself—our dataloader queries for all the traffic data available for that sensor and direction. Once done, we preprocess according to the **data preprocessing**. This data is then split using a train-test split of 80-20%, and the model is fitted on this data using an ARIMA(1,1,1) architecture. Once complete, we forecast the testing data and compare it with the actual values.

As this model works only on temporal data, we provide only the raw traffic speed values and no other features. The ARIMA(1,1,1) model was chosen as it works well with little knowledge of the underlying data trends or structure and covers all the base ARIMA features. This model was quickly abandoned, as discussed in the **results**, and therefore, no optimisations past this baseline were performed.

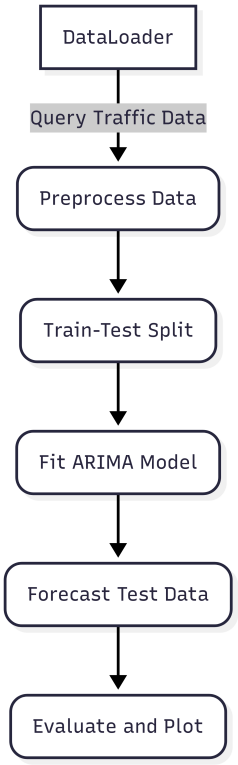

Fig. 14. Architecture diagram of ARIMA predictive modelling

3.5.3 Neural Network Model (LSTM)

An architecture diagram for this model is provided in Fig. 15. The design of this model is far more complex, as it is a neural network similar to that of the MLP in the next section. In this model, our DataLoader queries traffic and event data, which is passed to the FeatureEngineering module to add all the features and drop unused columns. We then perform some preprocessing by splitting the data into the same test-train split of 80-20% and scaling the data. A peculiarity of this model is the need to create time-series windowed data from the generated data before using our Trainer class to train the model. Once this is done, we perform two forecasts before evaluating and plotting the results. We delve into more detail in the sections below. This model

was abandoned, as discussed in the **results**, and therefore, no optimisations past the architecture below were performed.

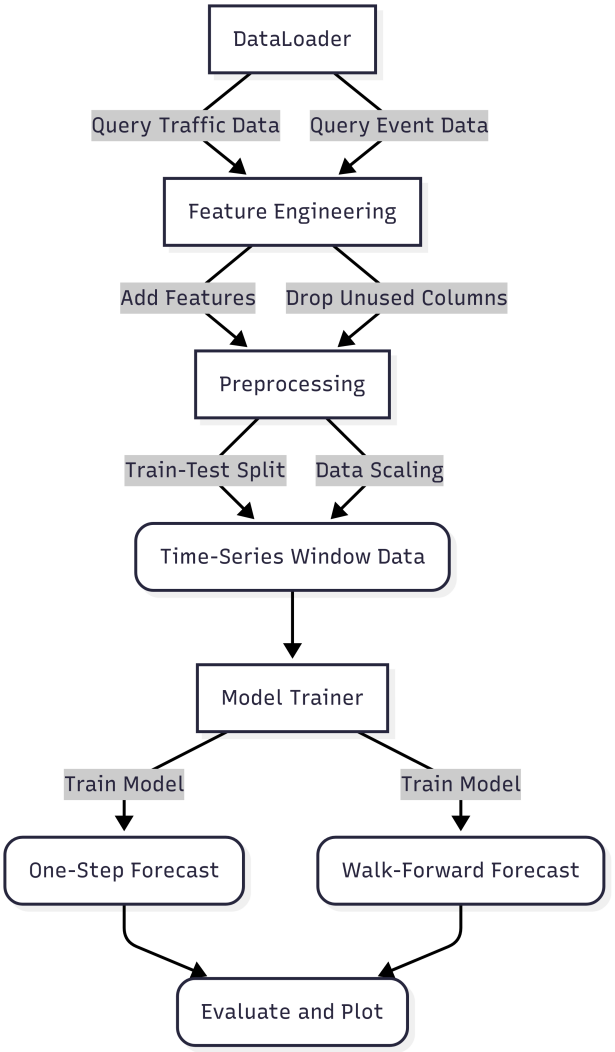

Fig. 15. Architecture diagram of LSTM predictive modelling

Data Scaling We applied MinMax scaling for this model to ensure all input features fall within the same range. This is crucial because LSTMs are sensitive to the scale of data. Unscaled features can lead to unstable training and slower convergence; therefore, scaling helps the model learn temporal patterns more effectively, leading to higher precision [39].

Time-Series Window Data Creating time-series windowed data is necessary for LSTM models as they are designed to learn from sequential patterns over time. The model can capture temporal dependencies and trends spanning multiple time steps by structuring the data into fixed-length input sequences (windows). This approach enables LSTMs to use historical context when forecasting future traffic speeds, which is essential for this task. We implemented this function using a variable window size that can be tweaked.

LSTM Architecture and Training For the architecture, we use a typical simple LSTM network with two layers of 100 units, each with a dropout of 10%. This model is optimised using an ADAM

optimiser [40], EarlyStopping to ensure no overfitting or wasted computational power over 50 epochs, and a validation split of 20% of the previous 80% training data. This ensures we train on a part of the data, optimise our model learning over another part, and then perform the final testing on another, completely separating them to ensure valid results.

Forecasts For this model, we implement two types of forecasts: a One-Step Forecast and a Walk-Forward Forecast. Since the LSTM model is most suitable for short-term forecasting, it should excel at a one-step forecast, which uses the model to predict a single step ahead. A walk-forward forecast, however, attempts to consecutively predict many steps in advance to construct a more long-term forecast, which we predict will be an issue for the LSTM.

3.5.4 Neural Network Model (MLP)

We provide an architecture diagram for this model in Fig. 16. In this model, our DataLoader queries traffic and event data, which is passed to the FeatureEngineering module to add all the features and drop unused columns. We then perform some preprocessing by splitting the data into the same test-train split of 80-20%

Data Scaling We applied robust scaling to this model to ensure all input features fall within the same range. This is for the same reasons as discussed in **data scaling**. However, here, we use robust scaling instead of MinMax since it is tolerant of outliers, ensuring our models perform better even with outliers that may be present in the traffic data by using the inter-quartile range instead of the minimum and maximum to scale [41].

Modular MLP Architecture and Training For the MLP architecture, we make use of a modular architecture. Instead of specifying a certain number of layers, neurons per layer, dropout rate, among others, we specify a modular architecture with several options for each parameter. This allows us to then tune the parameters in **hyperparameter tuning**, achieving the best architecture within our search for the required task. The base architecture consists of a dense input layer, followed by several hidden layers and an output layer. As with the LSTM architecture, we use a validation split of 20% of the previous 80% split training data, ensuring the separation of the three stages. We discuss the hyperparameters and training of this architecture in more detail in **hyperparameter tuning**.

Outputs To enable the use of the trained models in the remainder of the project, the flow of training the MLP model saves the trained model weights and the scalers used to scale the data. This will allow us to load the model weights, use the same scalers on new data, and generate predictions in the next stage.

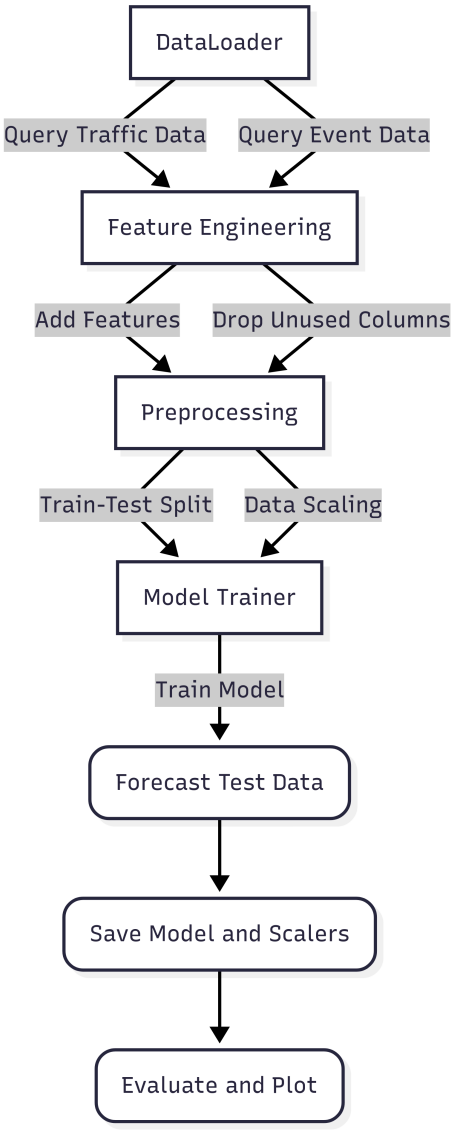

Fig. 16. Architecture diagram of MLP predictive modelling

3.5.5 Hyperparameter Tuning

This section covers hyperparameter tuning and model training in more detail, specifically for the MLP model as our choice of model for predictions. To perform hyperparameter tuning, we set up a modular architecture, as described above, and then ran a random search over 100 trials to determine the optimal configuration of hyperparameters for the network. During this search, we focus on minimising the validation loss and achieving the highest validation accuracy while also implementing Early Stopping to prevent overfitting. We present the hyperparameters chosen for search in Table 3.

The choices or ranges shown above were chosen to balance a range of testing with computational efficiency. In summary, we chose three values for each hyperparameter, except for the categorical ones, where we chose only two. For all the choices, we used the standard range typically used in ML, specifically MLP structures. As it would be unfeasible to run this hyperparameter search for each model of sensor or one single sensor over the whole training period, we will run the search on one specific sensor and direction as in the **data analysis**. We will then extrapolate the findings

Table 3. Hyperparameter grid for search on MLP architecture

Hyperparameter	Type	Choices / Range	Description
`units_input`	Integer	{64, 128, 256}	Number of neurons in first dense (input) layer.
`activation`	Categorical	{relu, tanh}	Activation function used in each dense layer.
`dropout_input`	Float	[0.1, 0.3], step=0.1	Dropout rate applied after the input layer.
`num_layers`	Integer	[2, 4]	Total number of hidden layers.
`units_i` (per layer)	Categorical	{64, 128, 256}	Number of neurons in each hidden layer.
`dropout_i` (per layer)	Float	[0.1, 0.3], step=0.1	Dropout rate applied after each hidden layer.
`optimizer`	Categorical	{adam, rmsprop}	Optimization algorithm used for training.

for all other sensors and the whole training cycle.

3.5.6 Evaluation Strategy

We now present a more detailed description of the model's evaluation strategy, including the metrics we focus on and the plots and forecast visualisations.

Loss Function Metric Several metrics can be used to evaluate the model's performance and use it in the loss function during training. As traffic speed is a continuous variable, this problem is a regression task, for which the most common metrics include MAE and MSE. We decided to use MSE as it performs better with the highs and lows that are common in traffic speed prediction.

Forecast Visualisations Forecast prediction visualisations are an excellent tool for understanding how the model functions, comparing changes made, and assessing the model's accuracy. These visualisations are also the ones used in the **results section**. The designed plots overlay the actual traffic speeds with the model's forecasted ones on the test dataset. Since this spans over a few months, each Plot generated was designed to show the whole timeline, the middle 10%, and the middle 3%. These zoomed-in versions allow the researcher or reader to understand the model over months, weeks, or days and gather a finer understanding of trends on different scales.

Metrics Gathering Finally, we discuss the gathering of metrics from the models. For this task, our training class evaluates the trained model against the test data, using MAE, MSE, RMSE, and MAPE accuracy. From the options for loss metrics, we add RMSE as the squared value of MSE to standardise units and MAPE, as it allows us to compare our results with other datasets and studies. These four metrics are the most common in most works in the area. These metrics are then saved to a CSV file, along with the metadata of that model (dates trained on, platform ID, sensor ID).

3.6 Traffic Simulation

We developed a semi-automated pipeline to streamline our simulation process using SUMO, enabling efficient evaluation and comparison of congestion management strategies. This pipeline

integrates several key components, including automated map generation, predictive model training, demand simulation, and congestion strategy implementations. With this pipeline, we can simulate various scenarios reflecting real-world conditions and rigorously assess the impact of different congestion management techniques on overall traffic efficiency and metrics.

3.6.1 Design Decisions

Congestion Management Strategy Firstly, we consider the choice of a congestion management strategy to simulate. As covered in the **background chapter**, many congestion management strategies can be applied. The most common ones are VSL, Dynamic Rerouting, or Ramp Metering. As ramp metering is used in motorways, and we focus on venues that do not have motorways as their main access point, we discard this option. The choice between VSL and Dynamic Rerouting is more fine-grained. However, we chose VSL as our strategy for the following reasons:

- Most venues explored have one single point of entry. Using dynamic rerouting as a congestion management approach would not reveal results, as any alternative route would increase travel time by more than the traffic through the initial route.
- SUMO simulations already perform dynamic rerouting internally. While disabling this is possible, we prefer to focus on a congestion management strategy that proves results compared to a typical baseline.
- Implementing a VSL strategy using SUMO and TraCi is powerful and allows for fine-grained control and comparison.

Simulation Area We now discuss choosing an area of the city to run the simulation. There were several considerations towards this choice, namely:

- An area large enough to measure the whole network effect, not only just local road segments, but small enough that it would be computationally feasible to simulate.
- An area that contains event venues to measure the impact of events on the congestion.
- An area for which we have enough traffic sensor data to generate demand.

The best area that fit all these requirements was the area containing both the Etihad Stadium and Co-Op Live venues, for which we had two separate sensor platforms covering an area of about 1.27 miles. The area's boundary is detailed in Fig. 17, with labels for both the venues and the sensor locations.

Simulation Timeline Another design decision is related to the timeline of the simulation that we will run. It is computationally infeasible to run simulations for all the days of data that we have collected. Our main requirements are to run simulations over general and event days. We will run simulations on the following days to ensure a statistically significant sample.

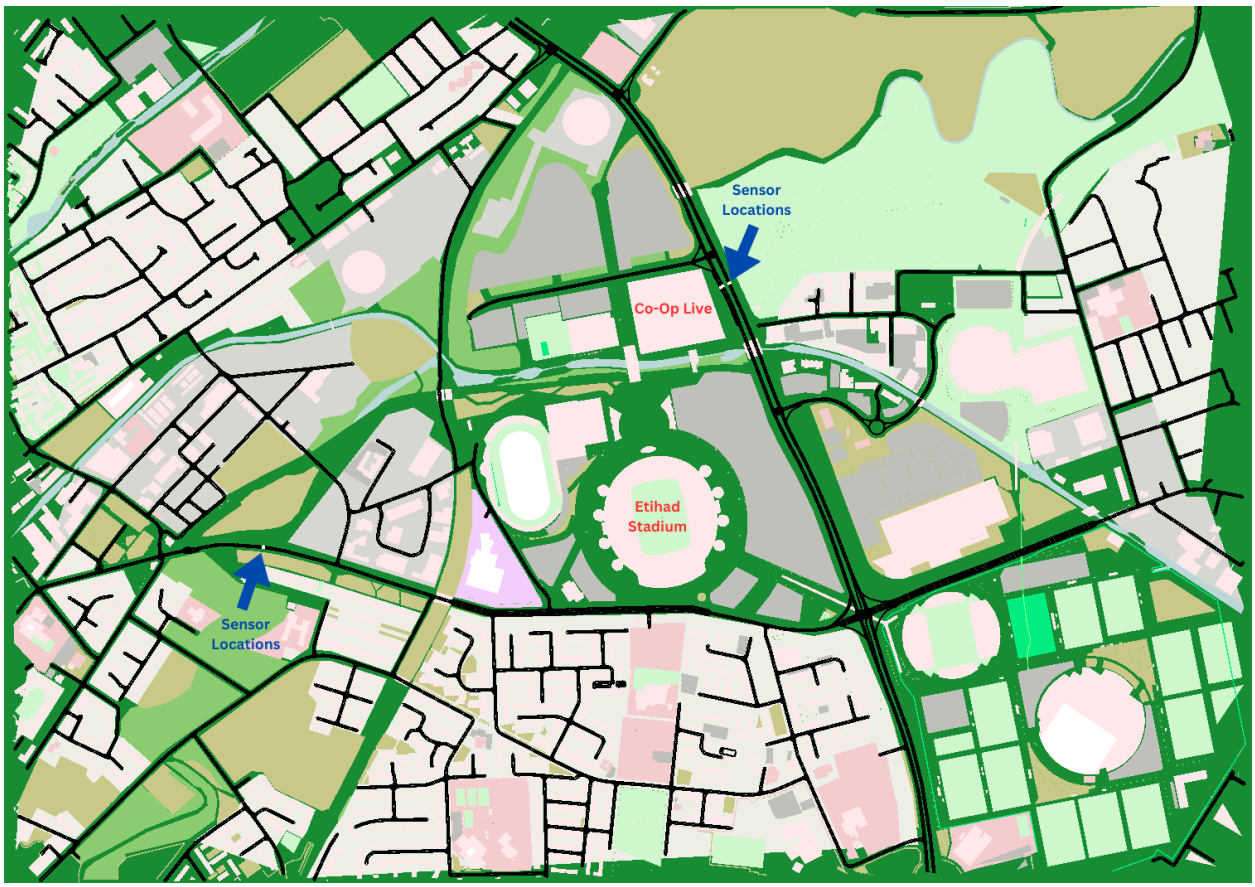

Fig. 17. Area boundary for simulation with labels

- 2024 Simulations: This batch of simulations covers 16 distributed days over 2024, specifically, the 1st and 15th day of each month from March to October.
- Event-Day Simulations: This batch of simulations covers 9 distributed days over 2024, manually chosen to be uniform around the year and to contain only days where an event occurs in either venue within our boundary.

Performance Metrics Finally, we discuss the possible performance metrics used to evaluate the strategies during simulation, the ones chosen and the reasons behind them. Many possible evaluation metrics exist in traffic management, and SUMO makes many of these easily accessible. Options include travel time, route length, waiting time, time lost, time spent on a specific edge (road segment), such as the edge closest to a venue, the number of vehicles, CO2 emissions, and others. To fully evaluate our strategies, we will focus on overall average network metrics instead of local ones. This means focusing on metrics that measure congestion or effect over the whole network map instead of focusing on just edges close to venues that may represent a significant improvement locally but then lead to congestion elsewhere in the network, allowing for real-world representative results.

After analysing all these metrics, we decided to focus on three that are the most descriptive: CO_2 emissions, trip duration, and time loss. Trip duration and time loss metrics help ensure that any congestion management techniques applied either reduce or maintain how long an average trip takes or the time lost in traffic during these. CO_2 emissions are one of the focuses of this project,

where a reduction in CO_2 emissions signifies a successful congestion management technique, reducing congestion and stop-and-go traffic [42], but also shows the impact the technique can have on the environment.

3.6.2 Map Generation

This section details the map generation of the simulation. As we have selected an area to simulate, we now require converting this area into a simulation in SUMO. This includes the general map area and all the roads, junctions, and traffic lights.

osmWebWizard osmWebWizard is a tool supplied with SUMO that assists in this task. It generates a SUMO simulation based on a selected area and generates random traffic for the simulation. Even though this latter feature will not be helpful to use as we will be generating our own traffic, the tool can convert a general area into a realistic simulation of that environment.

Assumptions When generating a simulation map using this tool, there are a few assumptions taken that we detail below:

- As we are only interested in car traffic, we add only lanes usable by car traffic.
- All traffic lights in the simulation run with a fixed length cycle set by osmWebWizard in "adaptive" mode, which adapts the cycles according to traffic. While the adaptive mode should be representative, the fixed cycle time used in the real world is unknown.

3.6.3 Prediction Generation

Our simulations will require the use of predictive speed values for the predictive congestion management techniques. We discussed the training of models in detail in the **MLP design**. In this section, we will explore the training of these models in batches and the generation of predictions from these that will feed into our simulations.

Batch Training Taking advantage of our modular approach to the predictive model implementation, we developed a small tool, `batch_training.py` in the `model` directory in the repository (Appendix A), that trains models for all the sensors available. As discussed previously, some of these sensors had significant malfunctions that made them unfeasible for our purposes. To prevent this, the tool trains only models that, during the specified period of data, contained over 80% of data points. Running this tool will train all the possible models, exporting data required for analysis of predictive accuracy, but also to be used in the next step, prediction generation.

Prediction Generation To generate the predictions required for simulation, we developed another tool, `predict.py` in the `model` directory in the repository (Appendix A), that generates predictions in batch for specified sensors and date ranges. In our case, this tool will be used to generate the predictions for the sensors within the simulation boundaries and for the dates outlined in the **design decisions**. This tool saves the predictions to CSV files that are ingested during the simulation and creates simple plots to visualise any errors in predictions easily.

3.6.4 Demand Simulation

The simulation pipeline's next step is generating demand according to real-world data. This will allow us to test our techniques in similar environments to the real world, which is especially important with predictive algorithms. The conditions in the simulation need to be as similar as possible for the predictive algorithms to function correctly. To achieve this, we cover two main aspects: the automated placement of the real-world traffic sensors into the simulation and the actual demand generation based on those sensor readings.

Virtual Sensor Placement An architecture diagram for this task is provided in Fig. 18. There are many types of sensors in SUMO. However, induction loop sensors are the best conversion for our data, providing both speed and count aspects, and therefore, we will place them as such. Since the traffic data we have is for all lanes across an edge, but each lane in SUMO requires a separate sensor, any data converted to or from simulation is divided or summed across the number of lanes, accordingly.

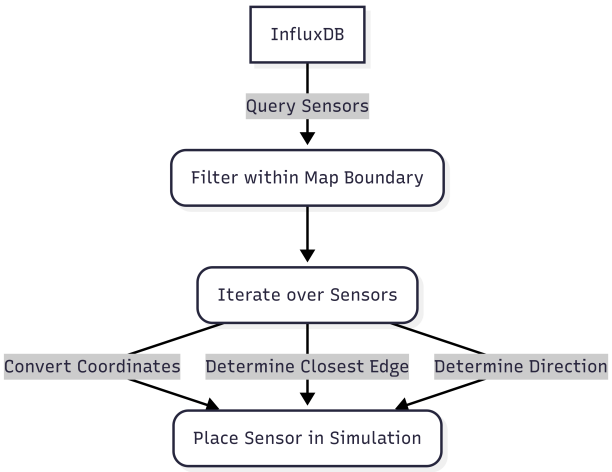

Fig. 18. Architecture diagram of automated simulation sensor placement

We have developed a tool for sensor placement that works by querying InfluxDB for all sensors and reducing them down to sensors within the boundary of the specific simulation. We then place these by converting their real-world coordinates to coordinates within the simulation map, and then we use a point-to-polyline algorithm to determine the road edge closest to these. Once this is found, we place the sensor according to its direction metadata by comparing the lane directions of that edge. The tool completes by creating an additional file for the simulation that adds this information so that SUMO can add these sensors during runtime.

Demand Generation An architecture diagram for this task is provided in Fig. 19. The next step in our simulation pipeline is to generate the required demand in the traffic network that illustrates the real-world conditions during each day of the simulations. To do so, we once again built a small tool that used our data sources and SUMO-provided tools to generate demand files for use in the SUMO simulation that relayed the real-world demand during that day. One issue with this method is that demand can only be generated using traffic counts, not speeds. While the tool includes the speed data in the intermediate edge data files, the `routeSampler` tool only uses the counts.

This tool works by querying speed and count data for the specified simulation day, and also generates random trips for the simulated cars. Once this is done, we load the sensor placements and SUMO network that we will base our calculations on, and apply scaling to prevent total network congestion. With this data, we generate edge data files. These intermediate files contain the required speeds and counts of the edges for which we have sensor data. The challenges related to the use of counts, random trips, and scaling are detailed in the **challenges section**.

With this data, the SUMO routeSampler tool [43] generates the demand files for the network, creating as much demand as needed to achieve the required vehicle counts for the edges where we have count data. This tool uses heuristic sampling to generate demand that achieves the required constraints of the counts we generated earlier. Since our data sources are limited, this sampling introduces randomness on edges that are further away from our sensor locations, as we have no available data from which to sample.

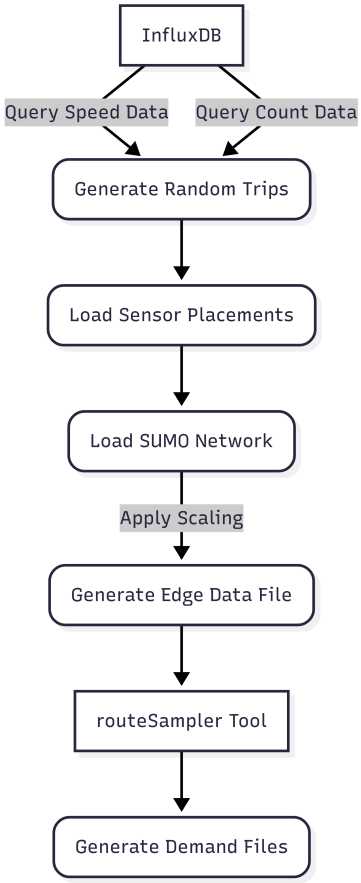

Fig. 19. Architecture diagram of automated simulation demand generation

3.6.5 Congestion Management Strategies

As discussed previously, we will compare different congestion management strategies in simulation using a VSL approach. The strategies will each use different algorithms to alter the maximum speed limit on the road edges for which we have traffic data. We will use the data from the simulation to compare these. We will compare baseline, reactive, and proactive strategies, which are detailed below.

Baseline Implementation We will simulate our baseline using no congestion management technique. This means the speed limit for the edges will remain at its maximum during the whole simulation. As another control technique, we implement a second baseline (Lower Limit) that uses a lower speed limit for all these edges. This is to ensure that any results retrieved are indeed from a valid congestion management technique and not from a lower fixed speed limit alone. This lower speed limit is set to the same value as the reactive technique's lowest used, described in the next section.

Reactive Implementation A few reactive strategies can be used for the reactive implementation. These include using a moving average or a rule-based version. The moving average attempts to take advantage of a rolling average of traffic speed and apply this value to maintain constant speeds. The rule-based version uses a fixed set of rules defined in advance and usually tuned to the particular edge it is used on to derive the speed limits to set. This rule-based technique is the most used in current deployments, and initial testing with the moving average technique yielded inconsistent results with our simulation setup. Therefore, our implementation makes use of a rule-based technique described below.

Our rule-based implementation works using the current speed and density of vehicles on an edge and some hyperparameters that can be tuned. The rules are shown in Fig. 20. To summarise, we reduce speed heavily if the traffic density is over a high-traffic threshold. If not, but the current speed is under a less dramatic speed limit, we set it to that. We use the default maximum speed limit if neither of the other conditions applies. This type of rule-based system is typical for use in reactive VSL systems, for which we have adapted its use here [44], [45].

```
cur_meas = traci.edge.getLastStepMeanSpeed(edge)
traffic_density = traci.edge.getLastStepVehicleNumber(edge)

if traffic_density > HIGH_TRAFFIC_THRESHOLD:
    vsl = SPEED_REDUCTION_2  # Heavy congestion
elif cur_meas < SPEED_REDUCTION_1:
    vsl = SPEED_REDUCTION_1  # Moderate congestion
else:
    vsl = DEFAULT_SPEED_LIMIT  # Normal conditions
```

Fig. 20. Rule-based reactive implementation rules

To set the hyperparameters in this approach, we will run half of the "2024 Simulations" we described to set them according to the best results. To optimise computational efficiency, we will

also add a timeout for the simulations, where if a simulation step takes more than 2 seconds, we discard that combination. This can be safely done, as a step taking this long means the whole network is congested and is, therefore, not a good combination.

Proactive Implementation For the proactive implementation, we use the predictions from our models in a few different ways to test the effectiveness of the proposed algorithm. The first and pure implementation is to set the speed limit to the output of the predictive model at that time step (in our case, 5 minutes, as this is the data frequency). This algorithm has a few inherent issues and optimisations that can be applied, especially concerning applicability to real-world use:

- Since we are using a predictive model, it may make sense to use the predictions some time ahead instead of what the predictions at the current time step would be, attempting to anticipate traffic congestion further in advance.
- In a real-world scenario, it would be impractical to set the speed limit to a continuous number, such as 27.95 MPH, which is the pure output of our models. Instead, rounding would be necessary for drivers to be able to follow the limits. We need to investigate whether this rounding may impact the effectiveness of a prediction technique.

To enable the comparison of the strategies, our implementation consists of a pure predictive approach, as well as alterations to use prediction 30 minutes in advance and rounding to the nearest 2 or 5 MPH.

3.6.6 Challenges Encountered

The design and implementation of the simulations were, by far, the most challenging part of this project. Many challenges were faced, many of which we could overcome, while some still affected our results and may need further work.

Count-Generated Demand The first and largest challenge was the limited demand generated through traffic count. The focus on speed was justified earlier due to its larger descriptive power. Since we can only generate simulation demand through count, we lose this descriptiveness, which may lead to inferior simulation outcomes. We will discuss the effects of this in the **results section**.

Simulation Inefficiencies The second challenge concerns simulation inefficiencies and their separation from real-world traffic networks. Several inefficiencies are inherent in this type of simulation model, leading to inefficiencies in the network that cause overscaled congestion when modelling the same counts as real-world traffic. This can be from traffic light programming differences, sensor errors, or even map errors where junctions are improperly configured. Due to this, simulating with real traffic counts leads to completely congested networks and, therefore, unreliable results. To address this, we will scale down the traffic counts by 30% before generating demand to ensure networks with realistic demand curves.

Random Trip Information During demand generation, we require a list of "trips", which are routes that the cars generated will follow. This can be made randomly or according to an Origin-Destination matrix, typically provided by a traffic authority, that contains the origins and destinations of trips in a traffic network. Unfortunately, TfGM does not provide these matrices, so we had to use random trips. This becomes another inefficiency, which can reduce the accuracy of the results.

4 Results and Discussion

This section will present the project's results, evaluate them, and discuss the findings. We will first focus on evaluating the predictive models' performance before reviewing the outcomes of the traffic simulations and congestion management strategies. Finally, we will finish with a detailed discussion of the findings and limitations.

4.1 Model Performance Evaluation

This first results section evaluates the predictive models created and their variations. We present the results of the three models implemented, including metrics, visualisations and a discussion.

ARIMA Model This model was abandoned from our final prototype after disappointing results due to its insufficient complexity for our purposes. The results of the ARIMA forecast on the test sensor are shown in Fig. 21. The plot shows the actual traffic speed split by train and test data, with the forecast in the red dashed line. As can be seen, the fluctuations present in the traffic data are too complex for the ARIMA model, resulting in the model simply constantly fitting the mean of the data as its output. This is a disappointing result; however, it is expected as the ARIMA model uses only the traffic data and pure timestamps to fit the model. This is just too complex for our purposes and was, therefore, abandoned.

LSTM Model The LSTM model showed interesting results that were expected, as previously discussed. We hypothesised that the LSTM would work exceptionally well for short-term but not long-term forecasting. This is precisely what our results show. Fig. 22 shows a plot of the actual vs predicted traffic speed from the LSTM model in a one-step forecast, meaning the model predicts one timestep ahead from the current real one, and this is performed for all data points. Fig. 23, on the other hand, shows the same Plot but with a walk-forward forecast, where the model attempts to predict the speed 100 timesteps into the future based on the previous prediction. Please note that these plots use scaled speed instead of real speed. The results are clear: the LSTM model performs exceptionally well for short-term forecasting, showing a precise following of the real traffic curve in our results. On the other hand, this model becomes unusable for long-term forecasting, as shown in the results of the walk-forward forecast. The forecast initiates at the same speed level. However, it quickly diverges from the real data as it loses any understanding of the data,

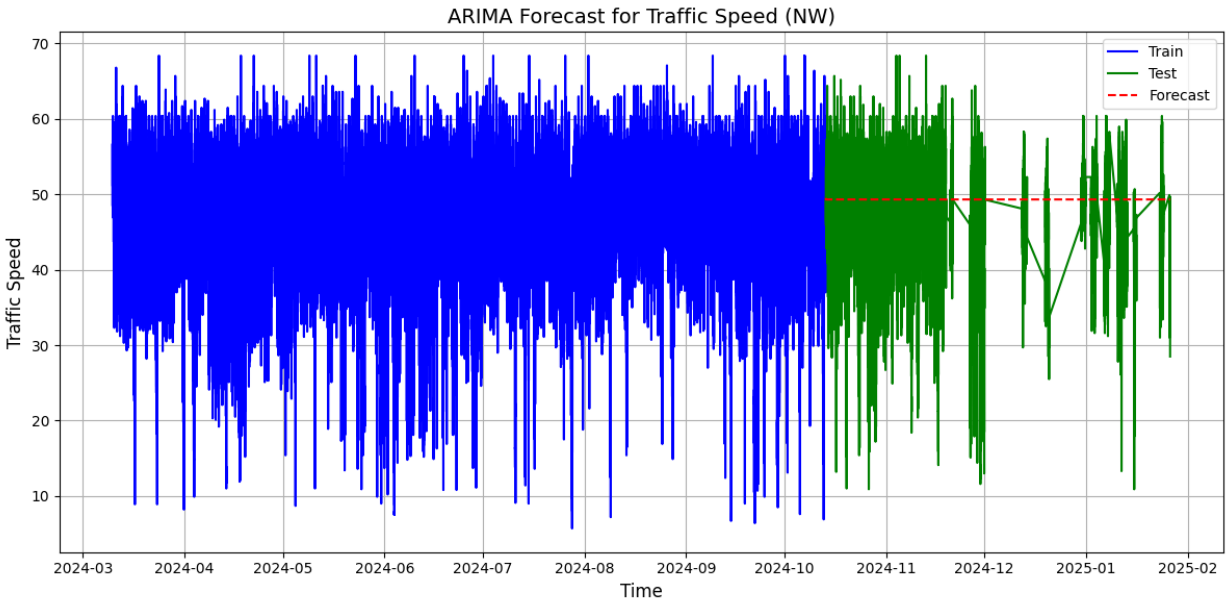

Fig. 21. Plot of ARIMA model forecast results compared with test data

multiplying the errors of each time step. This model was also abandoned as we focus on long-term forecasting as explained in the **model decision** section.

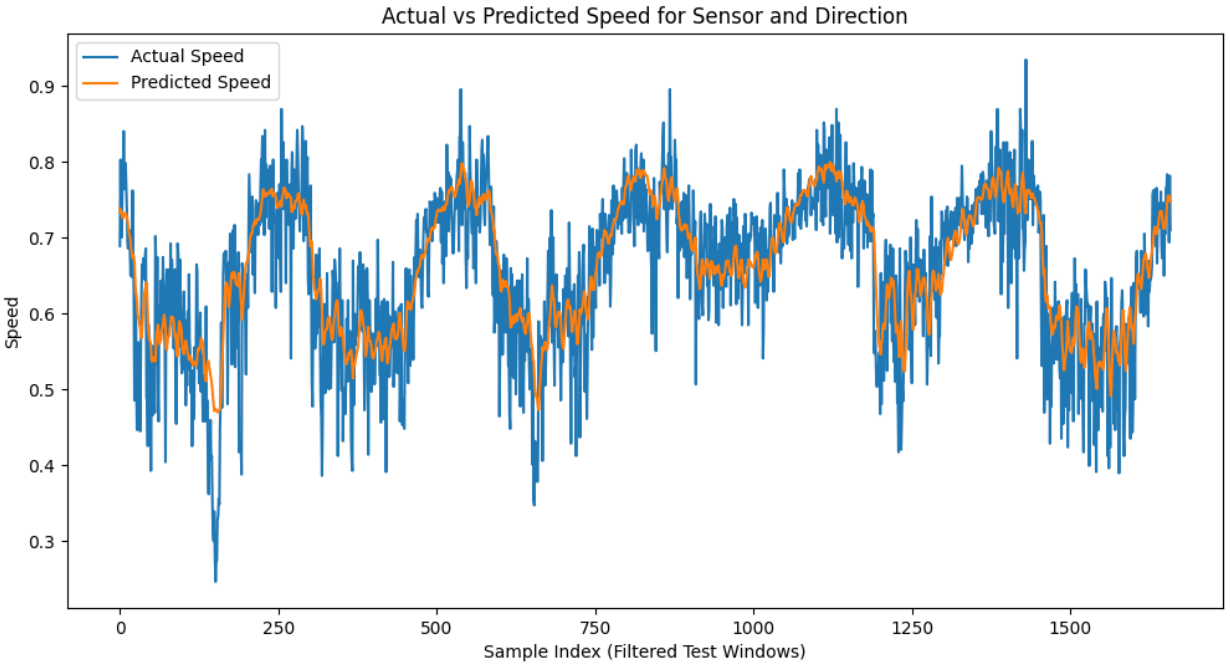

Fig. 22. Plot of LSTM model one-step forecast results compared with test data

MLP Model Our final MLP Model performed exceptionally well, especially after fine-tuning. It was able to model the complex time features and the impact of events, furthering the model's prediction accuracy. In the next section, we will go into more detail about the features used in this final model, while we present the final model performance results here. Fig. 24 shows plots of the final MLP model on the test sensor. These show the performance of the model's predictions compared with the actual data for the whole test data period. The plot is split into three, showing the whole timeline and the middle ten and three per cent to zoom into specific trends. Looking

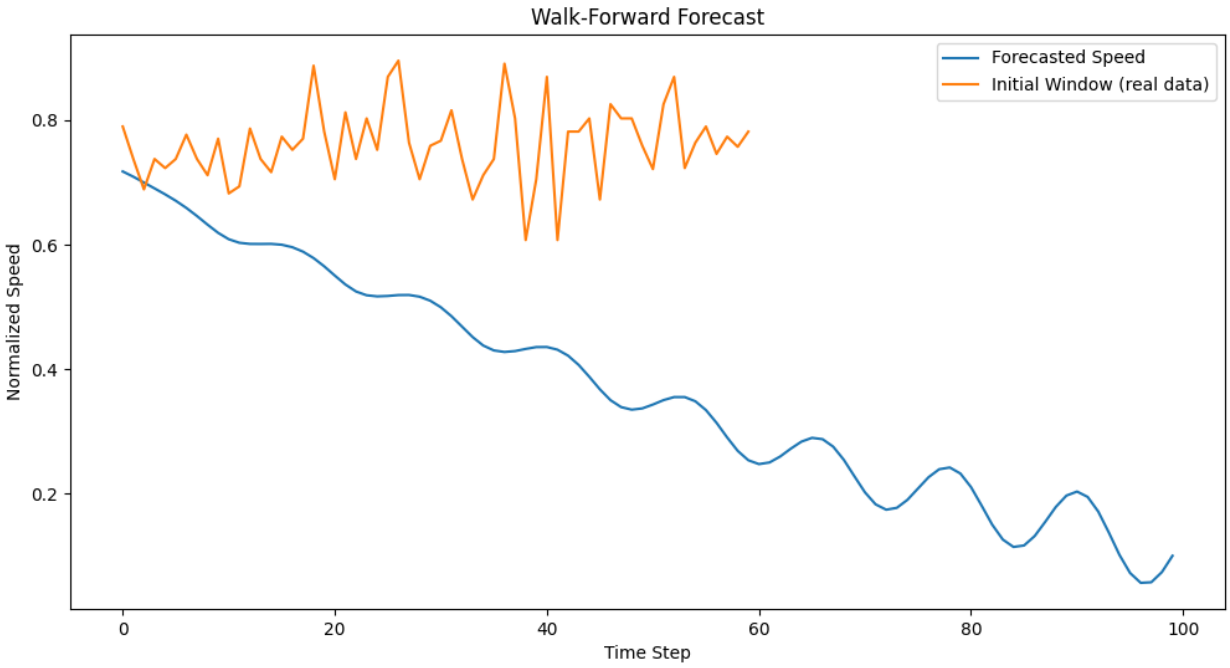

Fig. 23. Plot of LSTM model walk-forward forecast results compared with test data

at the middle 3 per cent, we can see the model performs very accurate predictions, following the curve of actual results along the time axis, further confirmed by the middle 10 per cent plot. Looking at the whole timeline is where we can understand the impact of events. Many sharp declines can be seen throughout the timeline, where events were happening around the area and where the model predicted sharper traffic speed declines due to this. We can see these follow the actual data, with the sharp declines coinciding with the ones in the actual data, although at a grander scale.

Overall, the predicted traffic closely tracks the ground truth, with minimal error during peak and off-peak periods, presenting a robust model for predictions. The results of this model are provided in Table 4. This table provides the errors of the same test data set, as shown in the Plot and the average errors of training on all available, consistent sensors. For this, we trained all sensors as we described in the design, resulting in 63 sensor-direction combinations of models trained to predict the speed data.

Table 4. MLP Model Performance Metrics

Model	**Metrics**			
	MAE (MPH)	**MSE (MPH)^2**	**RMSE (MPH)**	**MAPE (%)**
Test Sensor (drakewell-1163-nw)	2.83	14.49	3.81	11.47
Average of All Sensors	2.32	15.85	3.98	12.44

The results in metrics are also highly encouraging. For sensors where speed hovers around the speed limit (30 MPH), an average MAE of 2.32 MPH and RMSE of 3.98 indicate a low prediction error, corresponding to less than 12.5% deviation on average. This suggests that the model can accurately capture typical traffic speed patterns in free-flow conditions. These error values are sufficiently low to support real-time traffic monitoring and early detection of slowdowns, as even minor deviations from the free-flow speed can reflect emerging congestion. We will further compare these results in the **conclusions** against other works using MAPE.

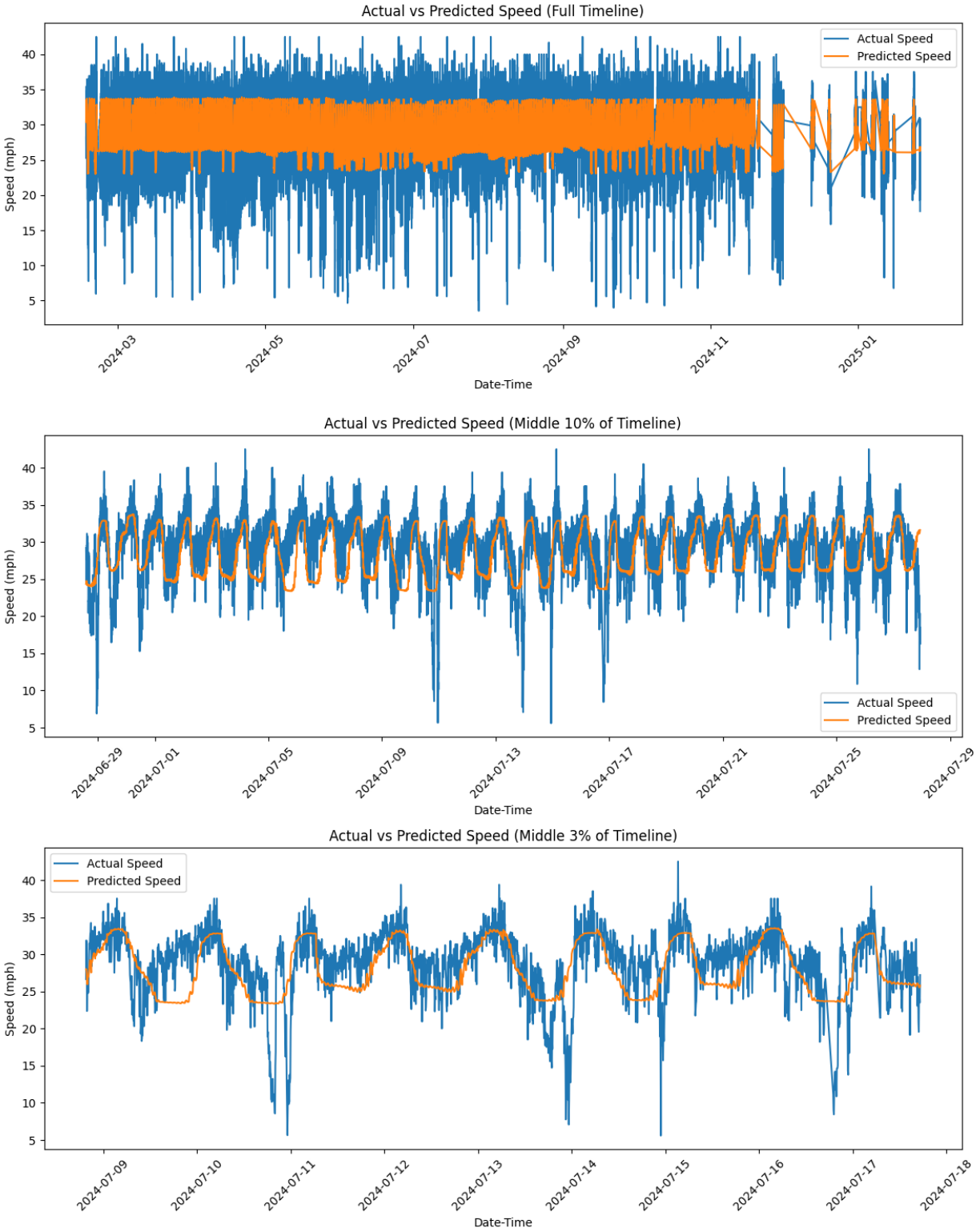

Fig. 24. Plot of MLP model forecast results compared with test data at three zoom levels

Model Decision and Discussion of Results The ARIMA model was abandoned as it was simply unsuitable for our purposes. The LSTM model was also abandoned, as while the short-term forecasting capabilities were excellent, these were just too short-term for our purposes, and the long-term forecasting was disappointing. We decided on the MLP model that showed excellent results, a good integration of extra features, and the capability of long-term forecasting that is superior to short-term forecasting, as it provides a broader window for proactive intervention as

well as strategic planning:

1. For our purposes, a long-term forecast allows for proactive interventions to occur much earlier if needed.
2. Long-term forecasting learns trends and the impact of features compared to heavily relying on the last measurement. These can provide sustained improvements over a longer period.
3. These forecasts allow for advanced warnings and observability, even for human operators. When considering using a predictive model outside of automatic congestion management, long-term observability and warning can be incredibly useful for human operators to plan and allocate resources effectively.
4. Long-term forecasting can be made more computationally efficient, as it can be run in the background in the cloud or fog layer a long time in advance, compared to short-term forecasting that might need to run locally and needs to be run at the moment.

4.1.1 Hyperparameter Tuning

We briefly describe the results of the hyperparameter tuning run for the MLP Model. The resulting optimal architecture is shown in Table 5. The architecture derived from the optimisation is quite deep, with four layers, showing a preference for a deeper network to derive more complex patterns from our features. Another point to note is that the dropout rates vary to balance regularisation but are higher towards the initial layers, lowering as the network becomes deeper. One final note is the consistent use of ReLU activation, which is consistent with typical uses in deep networks, as it prevents the vanishing gradient problem.

Table 5. MLP Tuned Architecture

Layer	Units	Activation	Dropout
Input Layer	96	ReLU	0.3
Hidden Layer 1	256	ReLU	0.2
Hidden Layer 2	256	ReLU	0.1
Hidden Layer 3	96	ReLU	0.3
Hidden Layer 4	256	ReLU	0.0
Optimiser	-	RMSprop	-

4.1.2 Feature Engineering Comparison and Impact

In this section, we delve deeper into the MLP model results. More specifically, we focus on comparing the impact of different features and the features used for the final model. Table 6 shows the results of training the test sensor model with different combinations of features and their performance. Our baseline model is the MLP with basic time encoding (Model ID 0), achieving a MAE of 2.90, RMSE of 4.23 and MAPE of 12.70%. With feature engineering effort, and most importantly, through our encoding of events in the model, we managed to bring this down to a MAE of 2.83,

RMSE of 3.81, and MAPE of 11.47%, resulting in our final model (Model ID 3), using only cyclic time encoding and our custom event encoding. We delve into feature importance and impact in more detail below.

Table 6. MLP Feature and Performance Comparison

Model ID	Basic Time	Cyclic Time	Binary Event	Custom Event	LLM Estimations	Weather	MAE (MPH)	MSE $(MPH)^2$	RMSE (MPH)	MAPE (%)
0	✓						2.90	17.89	4.23	12.70
1		✓					2.85	17.27	4.16	12.47
2		✓	✓				2.88	16.57	4.07	12.15
3		✓		✓			2.83	14.49	3.81	11.47
4		✓		✓	✓		2.84	16.78	4.10	12.33
5		✓		✓		✓	2.88	16.36	4.05	12.27

Time Encoding As discussed previously, we tested two designs of time feature encoding: basic and cyclic. As shown in the results table, this switch can improve performance slightly (~3.5% in RMSE and ~0.2% in MAPE). We verify our hypothesis that the cyclic encoding can better model these features as continuous and, therefore, help the model understand the trends of time features. We can visualise this difference in the figures below. Fig. 25 shows the model predictions using basic time encoding (Model ID 0), while Fig. 26 shows the model predictions using cyclic time encoding (Model ID 1). Using the basic time encoding, we can see the build-up and break of the time features continuously over the whole timeline, while the cyclic encoding flows continuously.

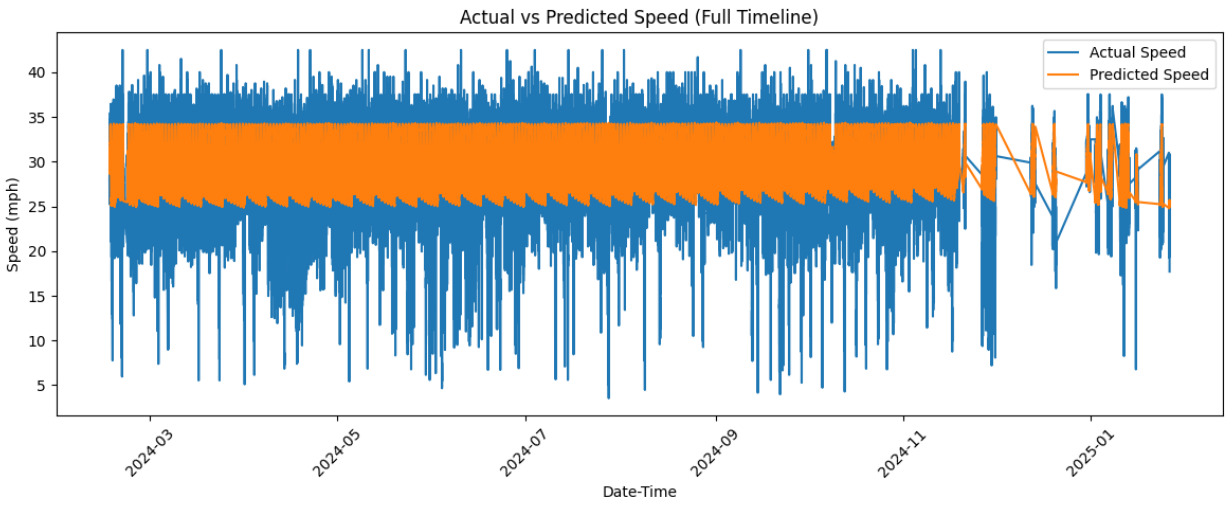

Fig. 25. Plot of MLP model forecast results compared with test data using basic time encoding

Weather Features As discussed previously, we tested adding weather features to our model, as weather can seriously impact traffic congestion. The results of adding these features are in Model ID 5, with a MAPE of 12.27% and a RMSE of 4.05. This is considerably worse performance compared to the same model without weather features (Model ID 3), which had a MAPE of 11.47% and RMSE of 3.81. Our hypothesis of why the performance decreased with the added weather features is twofold:

- Our weather data source may not be accurate enough to present useful features. Open-Meteo provides weather data correct to the nearest kilometre and hour. Since weather-

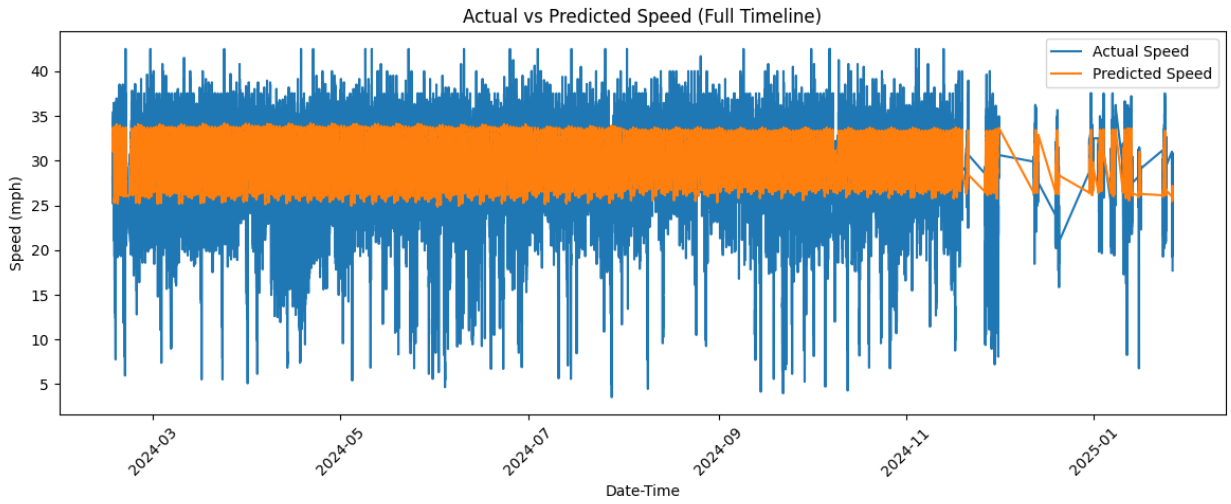

Fig. 26. Plot of MLP model forecast results compared with test data using cyclic time encoding

impacting traffic can sometimes be particular to a location and time, this may not be of enough resolution to impact our model.

- Our MLP architecture may not be deep enough to enable a deeper understanding of the weather impact. A deeper network or different architecture may be required to use weather features.

Event Features Event features were by far the most important features for our model predictions, further confirming the validity and usefulness of this project. We tested three different architectures for this: binary encoding, custom encoding, and custom encoding with the added LLM Estimations. Fig. 27 shows the model predictions using binary event encoding (Model ID 2), Fig. 28 shows the model predictions using custom event encoding (Model ID 3), and Fig. 29 shows the model predictions using custom event encoding combined with LLM Attendance Estimation (Model ID 4). Introducing binary event encoding improved accuracy by ~0.3% in MAPE and ~2% in RMSE. This improvement is significant, especially in RMSE, which is coherent with our expectations. MSE and RMSE give higher importance to the peaks and dips of predictions, where the impact of event features lies. Fig. 27 clearly shows the impact of events on the predictions, leading to the more significant dips seen during events. Using our custom event encoding, we see an even more considerable improvement of ~1% in MAPE and of ~8.8% in RMSE, compared to no event features. Fig. 28 shows the visualisation of this effect, where the encoding allows the model to differentiate the magnitude of dips according to the event type and time. These results prove the impact of our custom event encoding in significantly improving the accuracy of predictive models in traffic engineering.

Finally, we tested our LLM-based Attendance Estimation feature. However, the results were disappointing as they decreased the accuracy compared to the custom event encoding alone. Future work would be required to explain the exact cause; however, our hypothesis is the following:

- The Event Type features already present in our custom event encoding are sufficient for the model to understand attendance by differentiating between matches and concerts.

- The venues we focus on in this project are large venues where events usually fill them. Therefore, the attendance estimation is of less importance as the venue itself is characteristic of the attendance.
- Once again, our model architecture might not be deep enough for this type of feature to be useful. The visualisation (Fig. 29) shows that adding this feature breaks the model as the event peaks are no longer present, and some predictions are lost.

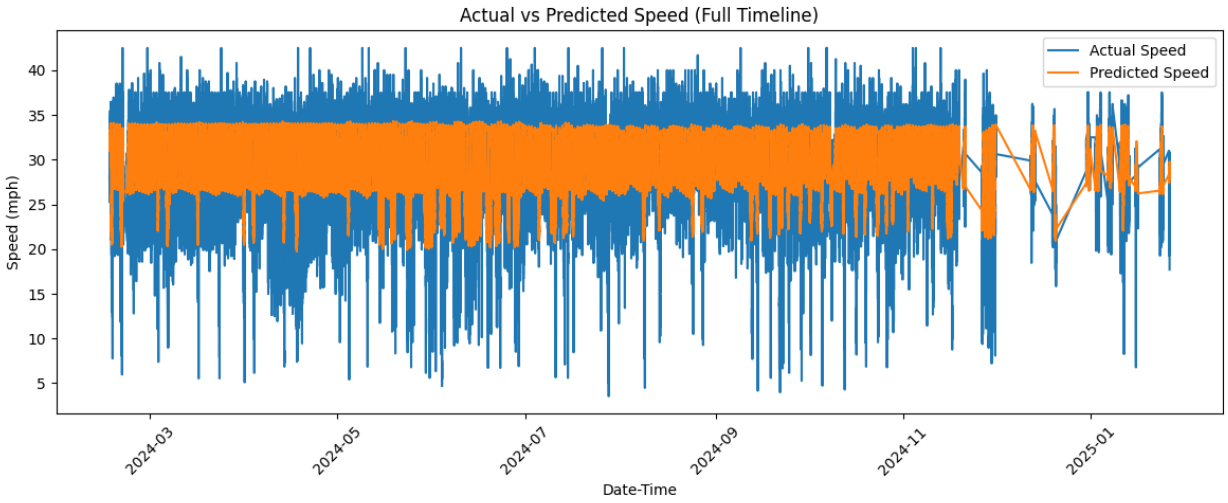

Fig. 27. Plot of MLP model forecast results compared with test data using binary event encoding

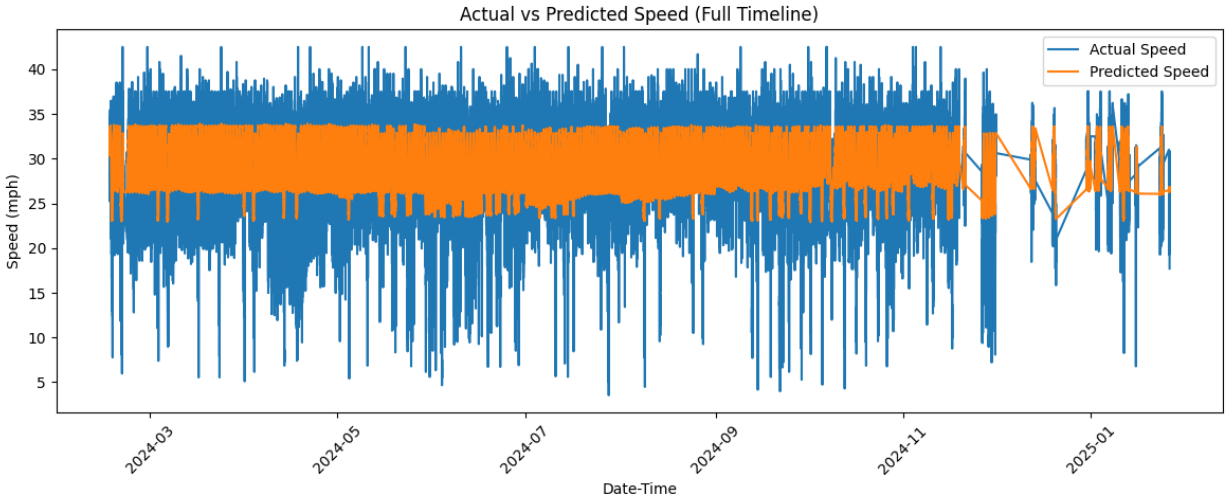

Fig. 28. Plot of MLP model forecast results compared with test data using custom event encoding

4.2 Simulation Outcomes

In this section, we will delve into the results and discussion of the simulations. We start with real-world demand generation results to understand the accuracy level of the simulated scenarios. We then move on to some quick results of the reactive scenarios' hyperparameter choices, which allowed us to set the reactive rule hyperparameters used for the simulation run. After this, we showcase the simulation results, comparing the different strategies and demonstrating the effectiveness of our proactive techniques using predictions.

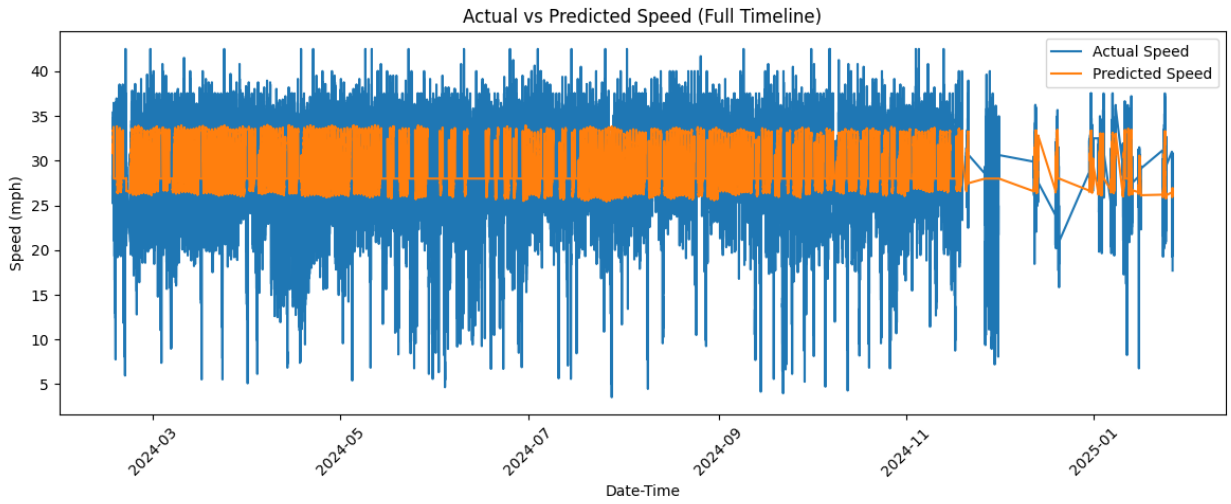

Fig. 29. Plot of MLP model forecast results compared with test data using LLM estimation

4.2.1 Real-World Demand Generation Results

Firstly, we cover the results of our demand generation technique for simulation. This allows the understanding of the validity of the results through the accuracy of the simulated scenarios they are run in. Table 7 shows the results of the errors in the simulated scenarios, comparing the gold standard (the actual speed data collected) with the results of the induction loop sensor data fed back from the simulation. We present the results of a typical and event day for a single data point, which we plot for visualisation, as well as the average over all sensors and simulation days.

Table 7. Real-World Demand Generation Error Metrics

Simulation	Metrics			
	MAE (MPH)	MSE $(MPH)^2$	RMSE (MPH)	MAPE (%)
Typical Day (2024-04-15)	2.39	11.67	3.42	8.13
Event Day (2024-05-04)	3.87	23.67	4.87	18.48
Typical Days Average	3.71	31.66	5.63	12.61
Event Days Average	3.94	30.60	5.53	19.13

Given the challenge of simulating demand through traffic count and comparing the observed speed, these results are minimally encouraging. While, on average, the errors are pretty high (12.61% for typical days and 19.13% for event days), the trend is the most important aspect of simulated demand for our purposes. Fig. 30 shows a plot of the actual real-world speed compared with the observed in simulation for our test sensor on a typical day, and Fig. 31 for an event day. These show that our simulated demand follows the curve closely at a lower overall speed level. This results from scaling down the demand to prevent total network congestion. As such, while the metric errors are relatively high, we are still confident that the demand generated is sufficiently close to the real world to provide results. For the event day demand, this also confirms our concern with count-based demand generation, where it struggles to model the sudden peaks and drops of speed during events, further evidenced by the higher MAPE. We will discuss this in more detail in the following section.

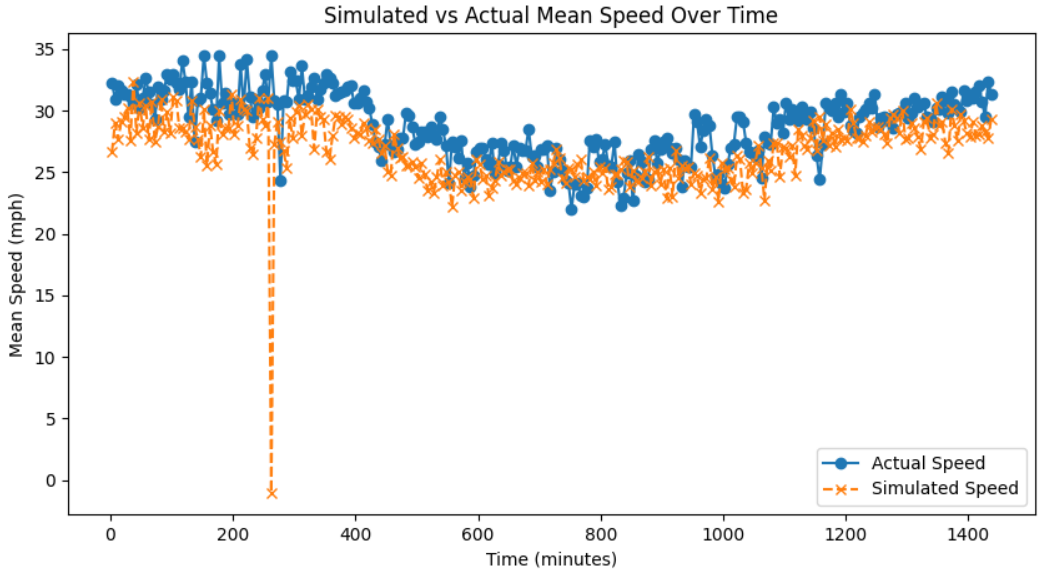

Fig. 30. Plot of simulation-observed traffic speeds compared with the real world for a typical day

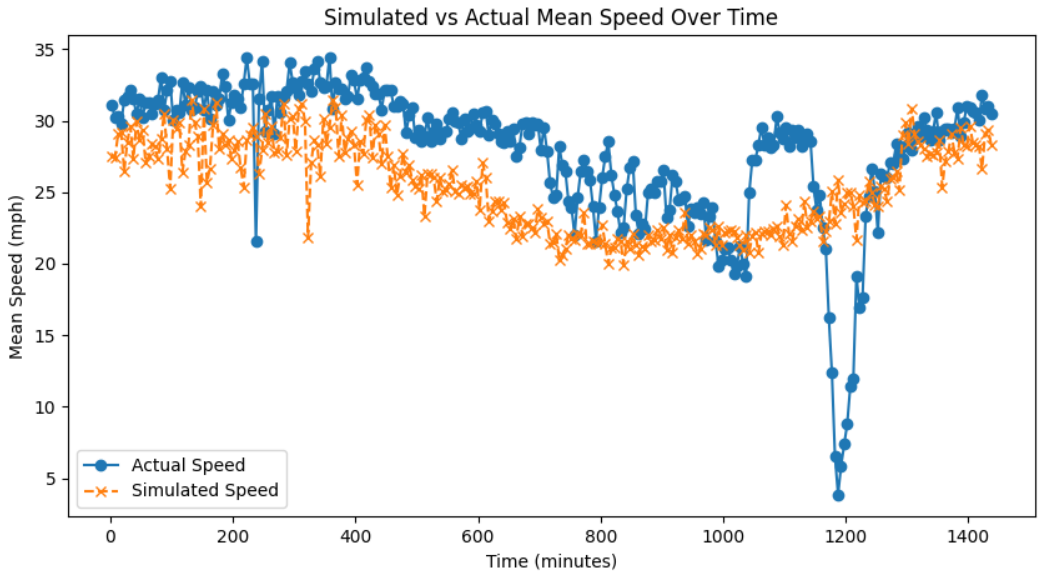

Fig. 31. Plot of simulation-observed traffic speeds compared with the real world for an event day

4.2.2 Reactive Hyperparameter Search

We cover the results of the hyperparameter search performed on the reactive strategy to choose the best option to simulate against the proactive and baseline strategies. The results are shown in Table 8 and Fig. 32. Since we implemented an optimisation to discard all combinations where the simulation timed out, which discarded all the worst combinations, we are left with all the combinations that perform equally. The results of this rule are further validated by the significant decrease in emissions and a slight decrease in time loss, with a minimal trip duration increase. Since all the options remaining performed equally, we chose the option which had the highest number of individual hyperparameters present among all the options. This resulted in the rule with a High Traffic Threshold (HTT) of 50, Speed Reduction 1 (SR1) of 25 MPH and Speed Reduction 2 (SR2)

of 20 MPH, which becomes our reactive rule.

Table 8. Reactive Hyperparameter Search Results and Metrics

Control Mode	HTT	SR1	SR2	Avg. Trip Duration (s)	Avg. Emissions (g)	Avg. TimeLoss (s)
Baseline	–	–	–	212.23	3604.93	54.41
reactive_rule_ril_0_rih_3_ti_0	30	25	10	213.63	3507.14	53.52
reactive_rule_ril_0_rih_3_ti_1	40	25	10	213.63	3507.14	53.52
reactive_rule_ril_0_rih_3_ti_2	50	25	10	213.63	3507.14	53.52
reactive_rule_ril_1_rih_3_ti_2	50	25	15	213.63	3507.14	53.52
reactive_rule_ril_2_rih_3_ti_0	30	25	20	213.63	3507.14	53.52
reactive_rule_ril_2_rih_3_ti_1	40	25	20	213.63	3507.14	53.52
reactive_rule_ril_2_rih_3_ti_2	50	25	20	213.63	3507.14	53.52

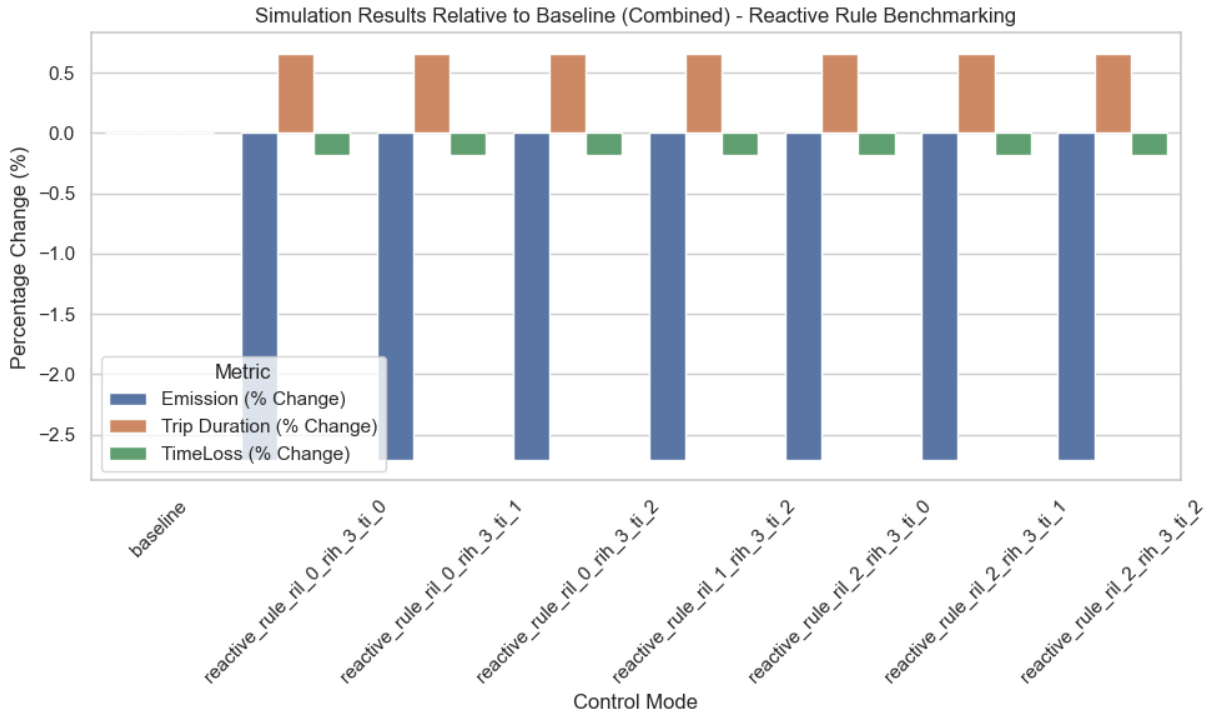

Fig. 32. Plot of results of hyperparameter optimisation for reactive rule strategy

4.2.3 Baseline vs. Reactive vs. Proactive Strategies

The simulations were run using all the defined strategies on both the 2024 simulations and event-day simulations, as previously discussed. The control modes are self-descriptive; however, to make clear, “predictive” uses the raw model output of speed for that particular timestep, while predictive_30 uses predictions 30 minutes in advance. The “predictive_round2” and “predictive_round5” use the current predictions rounded to the nearest 2 and 5 MPH, respectively. To reduce the number of combinations, the rounding was only tested on the “predictive” mode, as for both simulation rounds, “predictive” obtained better results than “predictive_30”.

Table 9 shows the average results over the whole traffic network tested for the 2024 simulations, while Table 10 shows the results for the event-day simulations. The tables present the mean and standard deviations for the three metrics we designed to focus on through all the control modes for that particular simulation batch run. We also provide visualisations through plots of the means

of the metrics, using the percentage differences from the baseline for a more specific analysis. Fig. 33 shows this plot for the 2024 simulations and Fig. 35 for the event-day simulations. For both of these runs, an outlier control mode resulted in significant congestion compared to all others. For easier interpretation, we have removed these outliers ("predictive_30" for the 2024 simulations and "lower_limit" for the event-day simulations) in Fig. 34 and Fig. 36, respectively.

Table 9. Results of congestion management techniques on 2024 simulations

Control Mode	Avg. Trip Duration (s)		Avg. Trip Emissions (g)		Avg. Trip TimeLoss (s)	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
baseline	212.29	4.17	3576.74	886.23	53.85	4.24
lower_limit	220.33	2.33	3909.72	573.59	53.99	2.40
predictive	214.18	5.55	3410.36	764.62	54.56	5.44
predictive_30	287.86	299.11	4983.88	6360.49	118.11	258.47
predictive_round2	212.90	2.54	3380.20	457.54	53.19	2.46
predictive_round5	214.02	3.01	3578.66	715.42	53.39	2.85
reactive_rule	212.97	2.75	3408.62	551.42	53.11	2.60

Table 10. Results of congestion management techniques on event-day simulations

Control Mode	Avg. Trip Duration (s)		Avg. Trip Emissions (g)		Avg. Trip TimeLoss (s)	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
baseline	209.35	2.06	3553.06	513.19	51.68	1.96
lower_limit	218.18	1.83	3960.41	427.48	52.54	1.74
predictive	211.01	1.98	3359.95	404.68	51.52	1.79
predictive_30	210.82	2.01	3488.96	424.20	51.32	1.82
predictive_round2	211.38	2.07	3424.87	489.96	51.66	1.86
predictive_round5	212.12	2.30	3543.96	404.90	51.79	2.09
reactive_rule	210.75	1.95	3526.20	415.62	51.63	1.70

Discussion of Results The first important point is that simply setting a lower speed limit as a congestion management technique is ineffective. Both trip duration and emissions largely spike upwards, confirming that the results obtained with the other strategies are not simply due to a lower speed limit.

Secondly, we note that the reactive strategy is effective. For both the 2024 simulation runs and event-day-specific ones, we see that the reactive strategy significantly reduces emissions while only slightly increasing trip duration. The effect of this strategy is much higher for the overall simulations compared to event days. For these, it is still efficient but at a much lower magnitude. This is expected, and we expect the proactive congestion techniques to fill the gap.

Discussing the proactive techniques, we present exciting results. For the 2024 simulations, the pure predictive approach performs just as well as the reactive versions, with a slight increase in trip duration and time loss. However, the predictive technique outperforms its reactive counterpart for event days. Here, the predictive approach achieves reduced emissions of more than 5% on average, compared to the reactive ~0.5%, as well as a more significant reduction of time loss while keeping similar trip durations. These results prove the potential of proactive congestion management techniques, especially during events.

For the variants with rounding, rounding to the nearest 5 MPH presents disappointing results. However, rounding to the nearest 2 MPH improves the results for the general 2024 simulations

and only slightly reduces the effectiveness in the event-day simulations, proving that the predictive approach can outperform typical reactive techniques even when rounding is applied.

However, the most interesting result of the simulations is the use of predictions further in advance. For the 2024 simulations, this resulted in total network congestion, while for event days, it performed relatively well but worse than the base predictive mode. This result is slightly different to our initial expected outcome, for which further work is needed, but we hypothesise the following:

1. A typical reactive or real-time predictive technique may work best for normal conditions. For event days with a more erratic speed pattern, a more advanced prediction may be useful.

2. Advanced predictions of 30 minutes may be too far in advance and cause congestion through a lower speed limit before it occurs, instead of preventing it early.

3. The predictions from our ML models are not completely accurate to the specific time. This means that the predictive model may predict a speed decrease some minutes in advance or after the actual speed decrease. This variance in the predictions may already be providing the algorithm with enough information in advance to achieve great results, while manually advancing these by 30 minutes may accentuate this error.

When comparing the variance of the results using their standard deviation, we can also see that the best-performing technique ("predictive_round2") has a lower or similar variance to both the baseline and the reactive version, showing that it is also a robust technique that presents consistent results across the simulations tested.

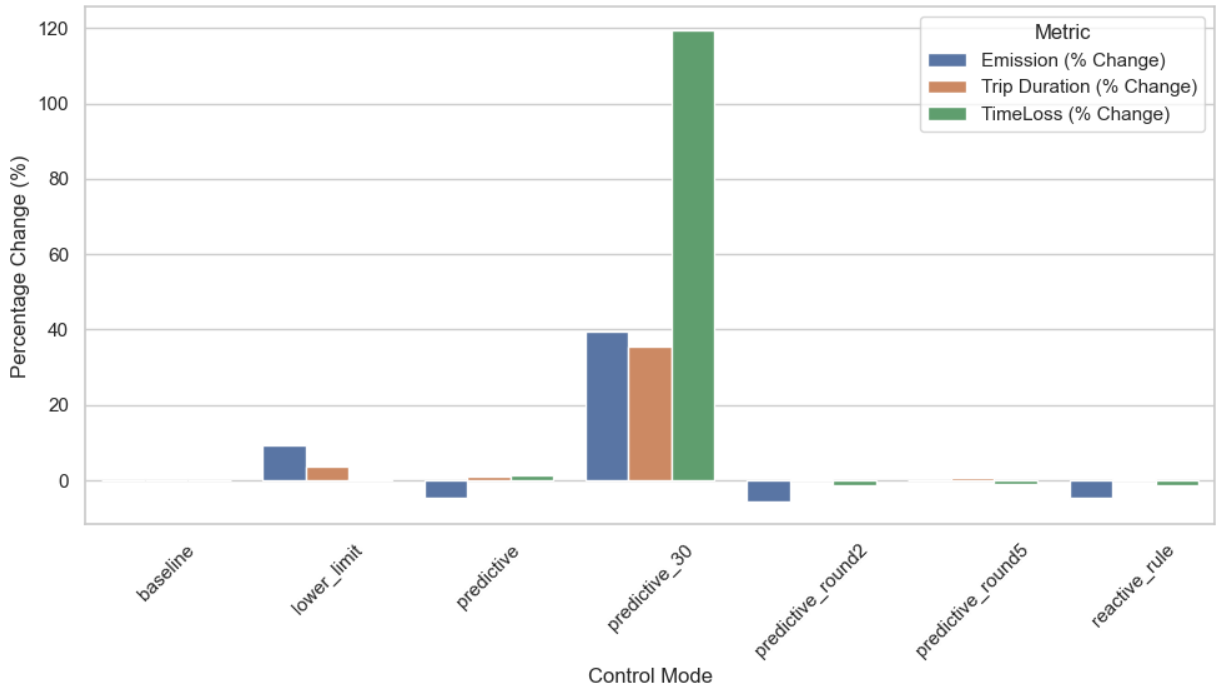

Fig. 33. Plot of congestion management techniques results on 2024 simulations

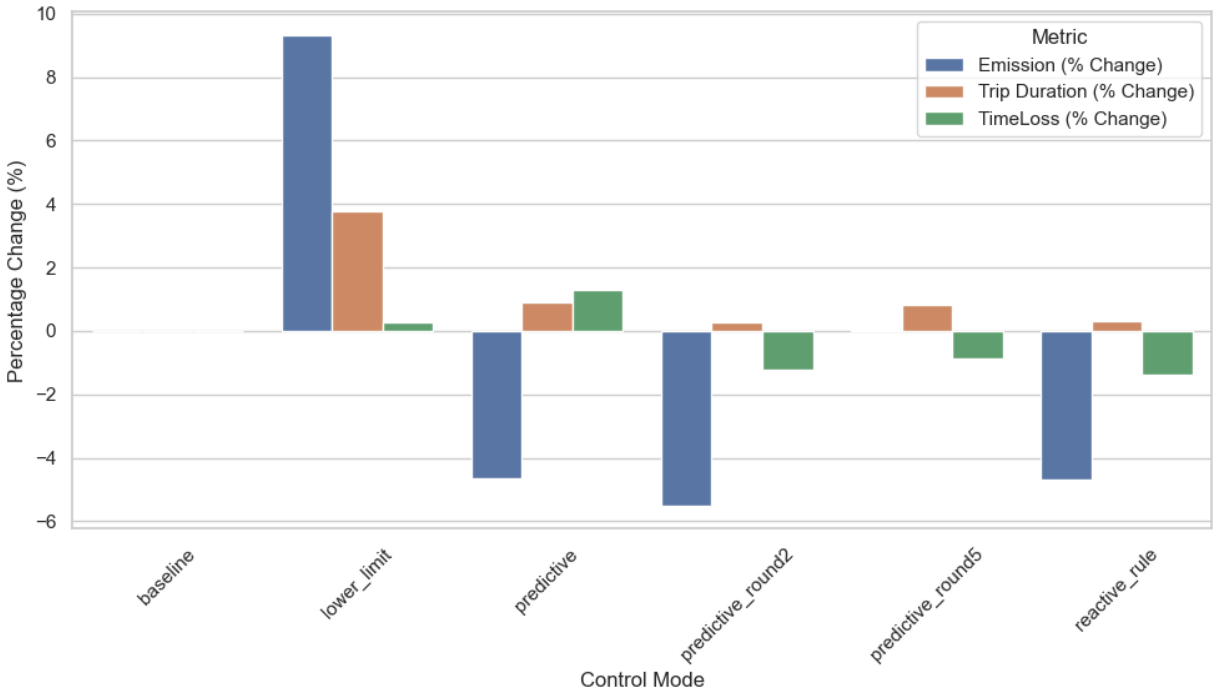

Fig. 34. Plot of congestion management techniques results on 2024 simulations - outlier removed

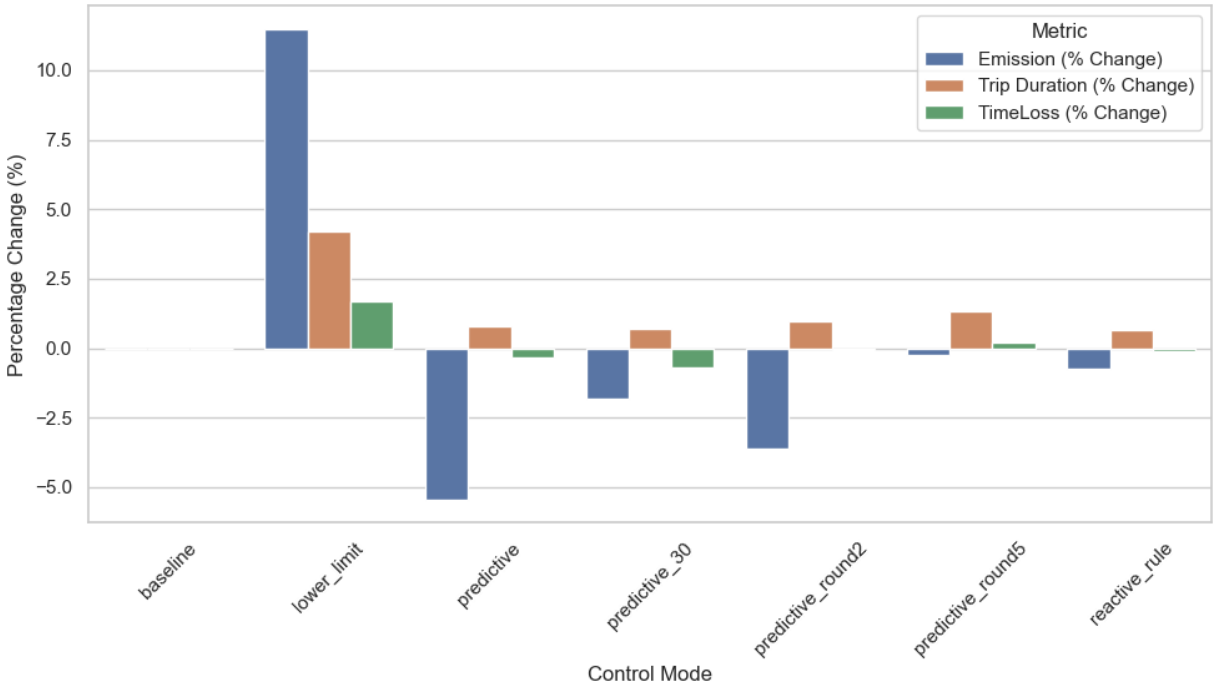

Fig. 35. Plot of congestion management techniques results on event-day simulations

4.3 Key Observations

In summary, a proactive congestion management technique powered by a predictive model can perform far superior to a typical reactive strategy, especially during large-scale events. This technique can significantly decrease congestion, as modelled by the CO_2 emissions reductions and time lost during trips while minimally impacting trip duration. In particular, using the predictions of the exact moment of traffic, coupled with rounding towards the nearest 2 MPH, shows these

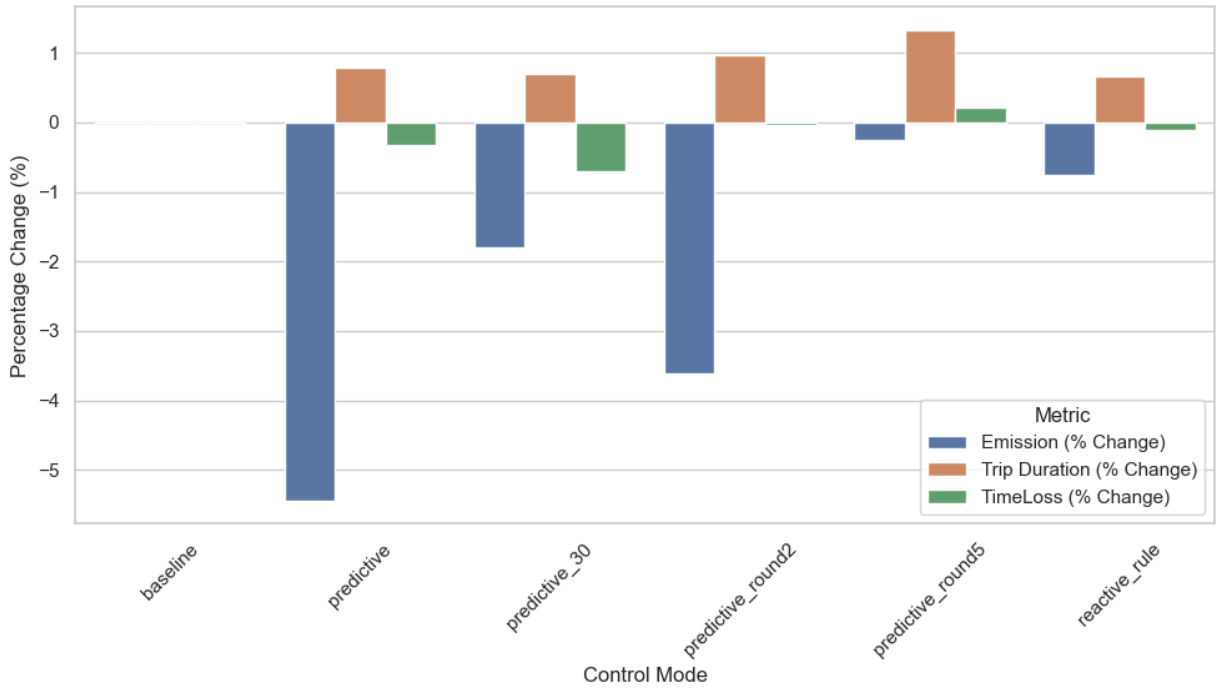

Fig. 36. Plot of congestion management techniques results on event-day simulations - outlier removed

results while allowing for real-world deployment.

4.4 Limitations and Sources of Error

This section briefly discusses the limitations and sources of error for the results presented above. These should be taken into account when analysing the results.

Data Scarcity The data collected for events and traffic is limited. Therefore, our results are based on this available data, which may not be fully representative of real-world traffic networks. Missing event data can be seen in the earlier model plots, where our predictive model can predict the slower speeds for events it knows about, but many other drops we can not account for are seen. While adding more event data should only improve our algorithms' accuracy, including more traffic data may alter the simulation outcomes. In collaboration with traffic authorities, traffic data could be severely improved.

Data Variability Event data is regularly changing, cancelled, or rescheduled. This data needs to be continuously fed into the predictive model for an accurate model.

Simulation using Vehicle Count As previously discussed, demand generation using vehicle count does not fully represent the actual speeds, especially during events. This lack of representation in the simulations may affect the performance of the algorithms. In theory, if these could be modelled into the simulation demand, it should improve the performance of proactive methods as the actual demand will be closer to the predictive model's drops.

5 Conclusions and Future Work

5.1 Summary of Contributions

Our work comprehensively explores long-term traffic prediction and proactive congestion management using machine learning and real-world event integration. We present an integrated model architecture for accurate traffic speed forecasting, an applied congestion mitigation framework for large-scale event days, and an LLM-based attempt at estimating event attendance. Our contributions span predictive performance, operational deployment feasibility, and insight into data augmentation strategies using modern language models.

5.1.1 Achievements in Predictive Accuracy

We have developed an MLP-based model for long-term traffic speed prediction, achieving an average MAE of 2.32 and RMSE of 3.98 MPH, resulting in a MAPE of 12.44% across all trained sensors. This was achieved using a combination of cyclic encoding of time features and custom large-scale event encoding. This model architecture performs better than current approaches by at least 1% compared to other works performing long-term traffic speed prediction [46], [47]. Notably, this is due to the lack of integration of large-scale event data in existing long-term speed prediction models, which enables us to achieve higher accuracy. Most existing models also focus solely on prediction for up to a few days or weeks, while our proposed model can be used as far in advance as necessary, given that the relevant event data is available.

5.1.2 Advancements in Proactive Congestion Management

While we faced significant challenges in generating demand representative of the real world in our simulations, showing errors of 12% to 19%, we are still confident that the generated demand trend is sufficiently close to provide valid results. From this, we have developed a proactive congestion management technique, based on our prediction model, that achieves a significant reduction of more than 3% in CO^2 emissions from baseline, compared to a typical reactive technique's 0.5%, for days where large-scale events take place, while also reducing time loss in traffic and minimally affecting travel time. Our proposed technique is similar to the reactive technique during typical days, showing its usefulness in traffic congestion management overall, with more significant results during event days. This technique also involves rounding speed limits to demonstrate the possibility of real-world deployment while maintaining results that surpass those of reactive techniques.

5.1.3 Novel LLM-based Attendance Estimation

While the LLM-based Attendance Estimation feature designed and tested in this project concluded with disappointing results, the architecture and research behind it are still significant achievements.

We integrated an LLM-based architecture to augment the feature engineering of a more typical MLP Model. While the results in this particular use case were disappointing, this has the potential of much better performance when applied to a greater variety of events or cities, especially where venues are typically not at full capacity for every event, or as a stepping stone towards vaster integration of LLMs for feature engineering in ML models.

5.2 Future Research Directions

Several routes for future research have been identified based on the methods and findings of this project. These directions aim to improve prediction accuracy, enable generalisation across broader contexts, and bridge the gap between simulation and real-world deployment.

5.2.1 Expanding Data Sources

Firstly, the expansion of data sources, both in terms of events and traffic, could significantly improve the performance of the proposed models. Collaborations with traffic authorities, Google Maps or the usage of the TomTom API could augment this data. In particular, if vast traffic data were available, a graph-based neural network could perform extremely well.

5.2.2 Advanced Control Strategies for Congestion Management

While we tested a subset of possible learning models and congestion management strategies, many others, including more advanced control strategies, can also be tested and have a significant impact on the area. Firstly, while we could not test reinforcement learning algorithms due to the lack of available data, these algorithms have shown great potential and could significantly improve the proposed architecture. Additionally, a mix of predictive and reactive algorithms may provide the best congestion management technique, as mentioned earlier in the hybrid approach.

5.2.3 Field Testing and Large-Scale Deployment

The results show our predictive model reliably forecasts speeds across all sensors, and our VSL-based proactive technique outperforms reactive methods in simulation. While demand-generation limitations may affect accuracy, refining these will confirm real-world viability. The approach can be adapted to other networks, and fog or cloud computing can offset its higher computational demands. Partnering with traffic authorities on pilot tests can smooth the path to deployment, cutting congestion, emissions, and travel time.

5.3 Key Takeaways

To summarise, we have been able to design a prototype capable of event-enhanced traffic prediction that motivates the use of proactive congestion management techniques through simulation verification, achieving the following initial goals:

- **Predictive Traffic Model Development**: MLP with cyclic time encoding and custom event features achieved a MAPE of 12.44%, surpassing current techniques.
- **Integration of External Data Sources**: Ingested major event schedules with custom encoding, yielding a 1 % lift over no-event baselines.
- **Proactive Congestion Management Algorithm**: Our VSL-based proactive algorithm cuts CO_2 emissions by more than 3 % compared to 0.5 % with reactive on event days, with similar time loss and trip duration.
- **Realistic Simulation Environment**: Built SUMO scenarios for both regular and large-event days using real sensor and event data, although with a high error rate.
- **Benchmarking and Analysis**: Compared proactive vs. reactive methods on travel time, emissions, and time-loss.

References

[1] U. S. G. A. Office, *Intelligent Transportation Systems: Benefits Related to Traffic Congestion and Safety Can Be Limited by Various Factors | U.S. GAO*, en. [Online]. Available: https://www.gao.gov/products/gao-23-105740 (visited on 03/23/2025) (cited on pp. 12, 14, 18).

[2] *Emissions of Carbon Dioxide in the Transportation Sector | Congressional Budget Office*, en, Dec. 2022. [Online]. Available: https://www.cbo.gov/publication/58861 (visited on 03/29/2025) (cited on p. 12).

[3] T. Das, I. Chatterjee, S. Mondal, T. Das, I. C. Ms, and S. Mondal, "Traffic Congestion Prediction Using Machine Learning Algorithm," en, *Cureus Journals*, vol. 2, no. 1, Jan. 2025, Publisher: Cureus Journals. DOI: 10.7759/s44389-024-01981-y. [Online]. Available: https://cureusjournals.com/articles/1981-traffic-congestion-prediction-using-machine-learning-algorithm (visited on 03/23/2025) (cited on p. 12).

[4] H. A. Kurdi, "Review of Closed Circuit Television (CCTV) Techniques for Vehicles Traffic Management," en, *International Journal of Computer Science and Information Technology*, vol. 6, no. 2, pp. 199–206, Apr. 2014, ISSN: 09754660, 09753826. DOI: 10.5121/ijcsit.2014.6216. [Online]. Available: http://www.airccse.org/journal/jcsit/6214ijcsit16.pdf (visited on 03/24/2025) (cited on p. 14).

[5] *Urban traffic control systems taxonomy and description*. [Online]. Available: https://www.its.leeds.ac.uk/projects/konsult/private/level2/instruments/instrument014/l2_014a.htm (visited on 03/27/2025) (cited on p. 15).

[6] *What is ITS?* [Online]. Available: https://www.dot.ny.gov/divisions/operating/oom/transportation-systems/systems-optimization-section/ny-moves/what-is-its (visited on 03/23/2025) (cited on p. 15).

[7] Y. Zhao and Z. Tian, "An Overview of the Usage of Adaptive Signal Control System in the United States of America," en, *Applied Mechanics and Materials*, vol. 178-181, pp. 2591–2598, May 2012, ISSN: 1662-7482. DOI: 10.4028/www.scientific.net/AMM.178-181.2591. [Online]. Available: https://www.scientific.net/AMM.178-181.2591 (visited on 03/24/2025) (cited on p. 15).

[8] *Google Maps 101: How AI helps predict traffic and determine routes*, en-us, Sep. 2020. [Online]. Available: https://blog.google/products/maps/google-maps-101-how-ai-helps-predict-traffic-and-determine-routes/ (visited on 03/24/2025) (cited on p. 15).

[9] V. N. Katambire, R. Musabe, A. Uwitonze, and D. Mukanyiligira, "Forecasting the Traffic Flow by Using ARIMA and LSTM Models: Case of Muhima Junction," en, *Forecasting*, vol. 5, no. 4, pp. 616–628, Dec. 2023, Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2571-9394. DOI: 10.3390/forecast5040034. [Online]. Available: https://www.mdpi.com/2571-9394/5/4/34 (visited on 03/23/2025) (cited on pp. 16, 17).

[10] V. Gupta, *Understanding Feedforward Neural Networks | LearnOpenCV*, en-US, Oct. 2017. [Online]. Available: https://learnopencv.com/understanding-feedforward-neural-networks/ (visited on 03/27/2025) (cited on p. 16).

[11] *Traffic Congestion Prediction Based on Multivariate Modelling and Neural Networks Regressions*, en. [Online]. Available: https://colab.ws/articles/10.1016%2Fj.procs.2023.03.028 (visited on 03/23/2025) (cited on p. 17).

[12] S. Yan, *Understanding LSTM and its diagrams*, en, Nov. 2017. [Online]. Available: https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714 (visited on 03/27/2025) (cited on p. 17).

[13] R. V and G. S, "Hybrid Time-Series Forecasting Models for Traffic Flow Prediction," *Promet*, vol. 34, Jul. 2022. DOI: 10.7307/ptt.v34i4.3998 (cited on p. 17).

[14] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Expert Systems with Applications*, vol. 207, p. 117 921, Nov. 2022, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.117921. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422011654 (visited on 03/24/2025) (cited on p. 18).

[15] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, “Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction,” en, *Sensors*, vol. 17, no. 4, p. 818, Apr. 2017, Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1424-8220. DOI: 10.3390/s17040818. [Online]. Available: https://www.mdpi.com/1424-8220/17/4/818 (visited on 03/24/2025) (cited on p. 18).

[16] N. Kumar and M. Raubal, “Applications of deep learning in traffic congestion detection, prediction and alleviation: A survey,” *Transportation Research Part C: Emerging Technologies*, vol. 133, p. 103 432, Dec. 2021, arXiv:2102.09759 [cs], ISSN: 0968090X. DOI: 10.1016/j.trc.2021.103432. [Online]. Available: http://arxiv.org/abs/2102.09759 (visited on 03/23/2025) (cited on p. 18).

[17] Exploratory Advanced Research Program (U.S.), “Predictive Real-Time Traffic Management in Large-Scale Networks Using Model-Based Artificial Intelligence,” English, Tech. Rep. FHWA-HRT-23-107, Feb. 2024. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/74165 (visited on 03/27/2025) (cited on pp. 19, 21).

[18] *Predictive traffic management prevents traffic congestion*, en. [Online]. Available: https://www.technolution.com/move/cases/predictive-traffic-management-prevents-traffic-congestion/ (visited on 03/23/2025) (cited on p. 19).

[19] D. Krajzewicz, “Traffic Simulation with SUMO – Simulation of Urban Mobility,” en, in *Fundamentals of Traffic Simulation*, J. Barceló, Ed., New York, NY: Springer, 2010, pp. 269–293, ISBN: 978-1-4419-6142-6. DOI: 10.1007/978-1-4419-6142-6_7. [Online]. Available: https://doi.org/10.1007/978-1-4419-6142-6_7 (visited on 03/27/2025) (cited on p. 20).

[20] *Getting Started - SUMO Documentation*, en. [Online]. Available: https://sumo.dlr.de/docs/Contributed/SUMOPy/GUI/Getting_Started.html (visited on 03/27/2025) (cited on p. 20).

[21] A. Stevanovic, C. Kergaye, and P. Martin, “SCOOT and SCATS: A Closer Look into Their Operations,” Jan. 2009 (cited on p. 21).

[22] S. Kwoczek, S. Di Martino, and W. Nejdl, “Predicting and visualizing traffic congestion in the presence of planned special events,” *J. Vis. Lang. Comput.*, vol. 25, no. 6, pp. 973–980, Dec. 2014, ISSN: 1045-926X. DOI: 10.1016/j.jvlc.2014.10.028. [Online]. Available: https://doi.org/10.1016/j.jvlc.2014.10.028 (visited on 03/23/2025) (cited on p. 21).

[23] *InfluxDB | Real-time insights at any scale*, Jan. 2022. [Online]. Available: https://www.influxdata.com/home/ (visited on 04/01/2025) (cited on p. 22).

[24] *Traffic APIs*. [Online]. Available: https://www.tomtom.com/products/traffic-apis/ (visited on 03/30/2025) (cited on p. 23).

[25] *Manchester-i*. [Online]. Available: https://manchester-i.com/ (visited on 03/30/2025) (cited on p. 23).

[26] *Bee Network*, en-GB. [Online]. Available: https://tfgm.com (visited on 03/30/2025) (cited on p. 23).

[27] H. Bi, Z. Ye, and H. Zhu, "Data-driven analysis of weather impacts on urban traffic conditions at the city level," *Urban Climate*, vol. 41, p. 101 065, Jan. 2022, ISSN: 2212-0955. DOI: 10.1016/j.uclim.2021.101065. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212095521002959 (visited on 04/01/2025) (cited on p. 24).

[28] *Free Open-Source Weather API | Open-Meteo.com*. [Online]. Available: https://open-meteo.com/ (visited on 04/01/2025) (cited on p. 24).

[29] *API-Football - Restful API for Football data*, en. [Online]. Available: https://www.api-football.com/ (visited on 04/15/2025) (cited on p. 25).

[30] *Wayback Machine*. [Online]. Available: https://web.archive.org/ (visited on 04/01/2025) (cited on p. 25).

[31] *Concert Archives - Remember all the concerts you've seen.* [Online]. Available: https://www.concertarchives.org/ (visited on 04/01/2025) (cited on p. 25).

[32] *2025 Football Schedule - ESPN (UK)*. [Online]. Available: https://www.espn.co.uk/football/fixtures (visited on 04/01/2025) (cited on p. 25).

[33] *Football News - Results, Fixtures, Scores, Stats, and Rumors*, en. [Online]. Available: https://www.nytimes.com/athletic/football/ (visited on 04/01/2025) (cited on p. 25).

[34] *Fixtures & Results | Man Utd Men's First Team*, en. [Online]. Available: https://www.manutd.com/en/matches/fixtures-results (visited on 04/01/2025) (cited on p. 25).

[35] *What is a feature engineering? | IBM*, en, Jan. 2024. [Online]. Available: https://www.ibm.com/think/topics/feature-engineering (visited on 04/02/2025) (cited on p. 29).

[36] H. Pelletier, *Cyclical Encoding: An Alternative to One-Hot Encoding for Time Series Features*, en-US, May 2024. [Online]. Available: https://towardsdatascience.com/cyclical-encoding-an-alternative-to-one-hot-encoding-for-time-series-features-4db46248ebba/ (visited on 04/02/2025) (cited on p. 30).

[37] *OpenRouter*, en. [Online]. Available: https://openrouter.ai (visited on 04/02/2025) (cited on p. 33).

[38] *Gemini 2.0: Flash, Flash-Lite and Pro- Google Developers Blog*, en. [Online]. Available: https://developers.googleblog.com/en/gemini-2-family-expands/ (visited on 04/02/2025) (cited on p. 33).

[39] V. Sharma, “A Study on Data Scaling Methods for Machine Learning,” en, *International Journal for Global Academic & Scientific Research*, vol. 1, no. 1, pp. 31–42, Feb. 2022, Number: 1, ISSN: 2583-3081. DOI: 10.55938/ijgasr.v1i1.4. [Online]. Available: https://journals.icapsr.com/index.php/ijgasr/article/view/4 (visited on 04/03/2025) (cited on p. 36).

[40] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs], Jan. 2017. DOI: 10.48550/arXiv.1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980 (visited on 04/03/2025) (cited on p. 37).

[41] *RobustScaler*, en. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.preprocessing.RobustScaler.html (visited on 04/05/2025) (cited on p. 37).

[42] C. Hofer, G. Jäger, and M. Füllsack, “Large scale simulation of CO2 emissions caused by urban car traffic: An agent-based network approach,” *Journal of Cleaner Production*, vol. 183C, pp. 1–10, Feb. 2018. DOI: 10.1016/j.jclepro.2018.02.113 (cited on p. 42).

[43] *Turns - SUMO Documentation*, en. [Online]. Available: https://sumo.dlr.de/docs/Tools/Turns.html (visited on 04/09/2025) (cited on p. 44).

[44] E. F. Grumert, A. Tapani, and X. Ma, “Characteristics of variable speed limit systems,” *European Transport Research Review*, vol. 10, no. 2, p. 21, May 2018, ISSN: 1866-8887. DOI: 10.1186/s12544-018-0294-8. [Online]. Available: https://doi.org/10.1186/s12544-018-0294-8 (visited on 04/15/2025) (cited on p. 45).

[45] P. Allaby, B. Hellinga, and M. Bullock, “Variable Speed Limits: Safety and Operational Impacts of a Candidate Control Strategy for Freeway Applications,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 4, pp. 671–680, Dec. 2007, ISSN: 1558-0016. DOI: 10.1109/TITS.2007.908562. [Online]. Available: https://ieeexplore.ieee.org/document/4382934 (visited on 04/15/2025) (cited on p. 45).

[46] D. Tedjopurnomo, F. Choudhury, and A. Qin, *TrafFormer: A Transformer Model for Prediction Long-term Traffic*. Feb. 2023. DOI: 10.48550/arXiv.2302.12388 (cited on p. 62).

[47] M. M. Kara, H. I. Turkmen, and M. A. Guvensan, “Smart Organization of Imbalanced Traffic Datasets for Long-Term Traffic Forecasting,” *Sensors (Basel, Switzerland)*, vol. 25, no. 4, p. 1225, Feb. 2025, ISSN: 1424-8220. DOI: 10.3390/s25041225. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11860824/ (visited on 04/14/2025) (cited on p. 62).

Appendices

A Source Code

All the accompanying source code for this project is available at `https://github.com/lourencofsilva/dissertation`

B LLM-Based Attendance Estimation Prompt

Context: You are an expert event planner.
Please estimate the attendance for the following real events in Manchester.
Return your answer as a JSON object where each key is the event ID and the value is the estimated attendance.

C Reproducibility

Compute Infrastructure The workload was split between a main workstation and an off-site server. The off-site server hosted the InfluxDB database and ran the data scraping and conversion scripts in the background. All other experiments (model training, simulations, data analysis) ran on the main workstation.

Off-site Server

- Cloud instance: Oracle Cloud `VM.Standard.A1.Flex`
- Block storage: 200 GB
- OS: Canonical Ubuntu 24.04

Main Workstation

- Machine: Apple MacBook Pro
- CPU: Apple M1 Pro (10-core CPU, 16-core GPU)
- RAM: 16 GB
- Storage: 512 GB SSD
- OS: macOS Sequoia (15.4)

Software Environment

- Python: 3.12.6
- SUMO: 1.20.0
- InfluxDB: 2.7.11
- All Python dependencies are pinned in the `requirements.txt` file from the source code.