# INTERACTIVE, MULTI-PURPOSE TRAFFIC PREDICTION PLATFORM USING CONNECTED VEHICLES DATA

by

Maged Shoman

B.Sc. Civil Engineering, American University of Sharjah, 2015

M.Sc. Environmental Engineering, Technical University of Munich, 2019

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

**Doctor of Philosophy**

in

CIVIL AND ENVIRONMENTAL ENGINEERING

University of Missouri

(Columbia)

December 2022

The following individuals certify that they have read, and recommend to the Faculty of Graduate School at the University of Missouri (Columbia) for acceptance, the dissertation entitled:

## INTERACTIVE, MULTI-PURPOSE TRAFFIC PREDICTION PLATFORM USING CONNECTED VEHICLES DATA

submitted by **Maged Shoman** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Civil and Environmental Engineering**.

**Examining Committee:**

Yaw Adu-Gyamfi, Civil and Environmental Engineering
*Supervisor*

Praveen Edara, Civil and Environmental Engineering
*Supervisory Committee Member*

Carlos Sun, Civil and Environmental Engineering
*Supervisory Committee Member*

Timothy Matisziw, Geography, Civil and Environmental Engineering
*Supervisory Committee Member*

# Dedication

*To my beloved parents Ahmed Sabri and Wallaa,*

*and dear siblings Haitham, Hossam and Dana.*

# Acknowledgments

The work presented in this dissertation not only sums up many hours of work in study and experimentation, but it also exemplifies a success story that is celebrated by many bright scholars and researchers. Firstly, I would like to thank my advisor Dr. Yaw Adu-Gyamfi for supervising my dissertation and supporting me throughout my PhD. I would also like to thank all members of my committee; Prof. Praveen, Prof. Carlos and Prof. Timothy for their support, time, guidance and feedback which helped in shaping this PhD dissertation.

I express my gratitude to Dr. Derek Anderson, Dr. Ye Duan and my colleague Alex Morehead from the Electrical Engineering and Computer Science Department for solidifying several fundamental concepts in building deep learning models. I am also grateful to my fellow lab members: Vishal Mandal, Khaled Aati, Dr. Mark Amo-Boaeteng, Aboah Armstrong, Linlin Zhang and Peng Jin; whose dedication, hard work and collaboration has inspired me (both, academically and personally) and positively impacted my PhD journey.

In addition to the Mizzou faculty and students, I would like to thank my instructors, classmates and friends from my high-school, bachelors and masters journey in the United Arab Emirates, Egypt and Germany. Above all, a special feeling of gratitude to my family for their endless support and unconditional love.

# Table of Contents

# List of Tables

# List of Figures

# Glossary

**AI** Artificial Intelligence iv

**AITVS** Advanced Interactive Traffic Visualization System 96, 97

**ANN** Artificial Neural Network 64, 66

**API** Application Programming Interface 100

**ARIMA** Autoregressive Integrated Moving Average 39

**ASR** Automatic Speech Recognition 95

**ATSPM** Automated Traffic Signal Performance Measures 13, 14

**BSM** Basic Safety Messages 13, 14

**BTS** Bureau of Transportation Statistics 8, 94

**CAD** Computer-Aided Dispatch 25

**CCTV** Closed-Circuit Television 1, 25

**CNN** Convolutional Neural Networks 39, 44, 66, 69

**ConvLSTM** Convolutional Long Short-Term Memory 69, 80, 83, 85, 90

**CPU** Central Processing Unit x, 4, 5, 9, 89, 116

**CUDA** Compute Unified Device Architecture 5, 72, 89

**CV** Connected Vehicles iii, iv, 2–4, 9–12, 18, 21, 36, 62, 63, 68, 69, 74, 89, 94

**DCRNN** Diffusion Convolution Recurrent Neural Network 40

**DL** Deep Learning 10, 63

**DOT** Department of Transportation 22

**ETL** Extract, Transform, Load iii, 76

**FHWA** Federal Highway Administration 1

**GC** Graph Convolution 46, 56, 58

**GC-GRU** Graph Convolution – Gated Recurrent Unit 40, 53, 55, 56, 58

**GCN** Graph Convolution Network 40, 42, 66

**GDP** Gross Domestic Product 1, 60

**GeoJSON** Geographic JavaScript Object Notation 113, 117

**GNN** Graph Neural Network 44

**GPS** Global Positioning System 11, 15, 23, 63

**GPU** Graphics Processing Unit iii, x, 4, 5, 9, 10, 63, 68, 72, 77, 89

**GRU** Gated Recurrent Unit 39, 58

**HA** Historical Average 53, 56, 58, 81, 83, 85, 90

**ICT** Information and Communication Technology 8, 93

**ITS** Intelligent Transportation Systems 3, 6–8, 12, 38, 61–63, 67, 93

**JSON** JavaScript Object Notation 111

**KNN** K-Nearest Neighbor 39, 64

**LiDAR** Light Detection and Ranging 13, 14

**LOS** Level of Service 96

**LSTM** Long-Short-Term Memory 39, 47, 53, 56–58, 65

**MAC** Media Access Control 25

**MAE** Mean Absolute Error 53, 55, 57

**MAPE** Mean Absolute Percentage Error 53, 55, 58

**MDA** Multiscale Data Analysis 30, 69

**ML** Machine Learning 63, 102

**MP** Market Penetration 12

**MT** Machine Translation 95

**NE** Northeast 29, 70

**NLP** Natural Language Processing 50, 94

**NN** Neural Network 41

**NW** Northwest 29, 70

**OEM** Original Equipment Manufacturer 2, 12, 15, 17, 18

**RMSE** Root Mean Squared Error 55, 83, 85, 86

**RNN** Recurrent Neural Networks 39, 46, 65, 83

**SBU-LSTM** Stacked Bidirectional and Unidirectional LSTM Network 42

**SE** Southeast 29, 70

**SIMD** Single Instruction Multiple Data 5

**SODA** Search Over Data 96

**SQL** Structured Query Language 94, 96, 101

**SVM** Support Vector Machine 64, 66

**SVR** Support Vector Regression 39

**SW** Southwest 29, 70

**TMC** Traffic Management Center 1, 11, 23, 26

**V2V** Vehicle-To-Vehicle 12

**V2X** Vehicle-To-Everything 2, 11

**VMT** Vehicle-Mile-Travelled 14

# Lay Summary

The viability of using innovate data solutions to understand traffic and combining it with incidents and weather data is explored to enhance traffic forecasting. The overall objective is to offer a digital platform that can analyze historical and future data offering accurate and fast insights on traffic performance. Car-related information is collected from sensors mounted on a subset of cars within the traffic network and delivered to a cloud-based service that can store the data. The stored data is then used in different ways. First, the spatiotemporal relationships between data points are learned through structuring the data in multi-dimensional arrays and feeding them to an Artificial Intelligence (AI) model that can learn the spatiotemporal relationships. Then historical data and prediction results inferred from the trained AI model are displayed on a customized website that can quickly query and display map visualizations of the traffic performance. The website is built with support of voice commands, meaning that you can speak-to to navigate through different pages and request specific traffic information (ex: a certain road/day/time) to display.

# Abstract

Traffic congestion is a perennial issue because of the increasing traffic demand yet limited budget for maintaining current transportation infrastructure; let alone expanding them. Many congestion management techniques require timely and accurate traffic estimation and prediction. Examples of such techniques include incident management, real-time routing, and providing accurate trip information based on historical data. In this dissertation, a speech-powered traffic prediction platform is proposed, which deploys a new deep learning algorithm for traffic prediction using Connected Vehicles (CV) data. To speed-up traffic forecasting, a Graph Convolution – Gated Recurrent Unit (GC-GRU) architecture is proposed and analysis of its performance on tabular data is compared to state-of-the-art models. GC-GRU's Mean Absolute Percentage Error (MAPE) was very close to Transformer (3.16 vs 3.12) while achieving the fastest inference time and a six-fold faster training time than Transformer, although Long-Short-Term Memory (LSTM) was the fastest in training. Such improved performance in traffic prediction with a shorter inference time and competitive training time allows the proposed architecture to better cater to real-time applications. This is the first study to demonstrate the advantage of using multiscale approach by combining CV data with conventional sources such as Waze and probe data. CV data was better at detecting short duration, Jam and stand-still incidents and detected them earlier as compared to probe. CV data excelled at detecting minor incidents with a 90% detection rate versus 20% for probes and detecting them 3 minutes faster. To

process the big CV data faster, a new algorithm is proposed to extract the spatial and temporal features from the CSV files into a Multiscale Data Analysis (MDA). The algorithm also leverages Graphics Processing Unit (GPU) using the Nvidia Rapids framework and Dask parallel cluster in Python. The results show a seventy-fold speedup in the data Extract, Transform, Load (ETL) of the CV data for the State of Missouri of an entire day for all the unique CV journeys (reducing the processing time from about 48 hours to 25 minutes). The processed data is then fed into a customized UNet model that learns high-level traffic features from network-level images to predict large-scale, multi-route, speed and volume of CVs. The accuracy and robustness of the proposed model are evaluated by taking different road types, times of day and image snippets of the developed model and comparable benchmarks. To visually analyze the historical traffic data and the results of the prediction model, an interactive web application powered by speech queries is built to offer accurate and fast insights of traffic performance, and thus, allow for better positioning of traffic control strategies. The product of this dissertation can be seamlessly deployed by transportation authorities to understand and manage congestions in a timely manner.

# Chapter 1

# Introduction and overview

## 1.1 Background

Traffic congestion costs cities billions of dollars every year when factors such as accidents, pollution and delays are factored in. According to a recent report published by the Texas Transportation Institute [1], all 494 metropolitan areas in the United States experienced 8.7 billion vehicle-hours of delay in 2019, resulting in 3.5 billion gallons of wasted fuel and 190 billion in lost productivity, or about 0.15 percent of the nation's Gross Domestic Product (GDP). When traffic demand approaches or exceeds the traffic system's available capacity, traffic congestion occurs. Traffic Management Center (TMC)s utilize real-time traffic information to help relieve traffic congestion and improve safety. This requires operators to constantly monitor road conditions through data streaming from a variety of sources including traffic sensors, GPS-enabled devices (probes), closed-circuit cameras, dynamic message signs, etc.

The Federal Highway Administration (FHWA) has long maintained nationwide programs to track traffic trends and vehicle distributions in order to meet data requirements set forth in federal highway legislation. To gauge traffic flow, the following methods are commonly used: video analytics systems such as Closed-Circuit Television (CCTV) cam-

eras which can provide a 24/7/365 coverage of speed and location of vehicles. The locations of vehicles provided, however, can be inaccurate due to the camera being fixed or heavy weather conditions. Another common data source is in-roadway sensors which come in a variety of shapes and sizes, and they can be installed on or within the pavement. Such a data source provides full coverage as well, but doesn't collect any information about the vehicles so the traffic coverage is done at a macroscopic scale. Additionally, transponder toll devices in cars are used to transmit data that contains a unique identifier (the toll card number) and position. Data from mobile phones with location tracking turned on can be utilized as a traffic probe in aggregate. Floating cellular data is another name for this type of data. This method is considered a low-cost or no-cost solution since phones are owned by the user (commuter) and can travel everywhere. Identifying the transportation mode used, however, can be a challenging task and may require complex algorithms. Recently, Connected Vehicles (CV) have become a very useful and direct source of traffic data as more vehicles become connected via built-in telematics and onboard gadgets. CV data is low cost since no significant installation or maintenance required, and rich with very frequent location and speed updates for every vehicle so it provides a complete view of geographic areas.

This dissertation is structured as follows: we briefly introduce each chapter in the current section along with the overall objectives. In the numbered chapters that follow, we dive into each chapter by discussing related work, methodology of our approach, followed by results, analysis and discussion. While each chapter has its own summary, a conclusion section is added at the end to summarize the dissertation work.

## 1.2 Connected Vehicles Data

CV technology can be defined as an application that utilizes Vehicle-To-Everything (V2X) communications to address mobility and safety concerns on roadways. CV data avail-

ability has been exploding in recent years. This is as a result of the advent of Original Equipment Manufacturer (OEM)s, Telematics platforms, and other in-vehicle technologies, that can continuously stream high-resolution, reliable and accurate vehicle data. A probe vehicle feature, which is part of connected vehicle technology, collects data about the state of the vehicle. Information from the collected data is used to estimate critical performance indicators such as travel time. Data generally used for traffic analysis and prediction has two main issues: availability, size of data, and the overreliance on probe data. When qualified traffic data is unavailable, the trained prediction model's performance degrades since performance correlates with the quality of input data. While we can collect more probe data due to transportation infrastructure modernization, the data doesn't capture the microscopic changes in traffic behavior. The main difference between probe data and CV data is scale and velocity. Probe data provides speed per road segment (line data) at a frequency of around five minutes, however, CV data provides speed at the vehicle's location (point data) at a frequency of three seconds.

To our knowledge, most prior studies [2–4] used probe traffic data that was less than a year old and, in some cases, as recent as one or two months [5]. Probe data cannot capture the microscopic travel speed or volume, and using it for traffic forecasting is likely to yield unreliable estimates, in our study, since our goal is to predict speed and volume simultaneously. Therefore, there is a need to use a more reliable data source that can provide microscopic live travel information to improve the reliability of traffic predictions along road segments such as CV data.

The future of Intelligent Transportation Systems (ITS) is shifting towards big real-time data from CV as automobile makers rush to incorporate CV technology in novel and current vehicles for numerous apparent advantages, which include vehicle autonomy and navigation, vehicle sensor and driver monitoring, live over-the-air updates, advanced road warnings, improved battery and fuel efficiency. Government and state institutions

that create, maintain and manage road infrastructure may take advantage of the CV data available to know what is happening on the road and make informed decisions on traffic flow and road pavement infrastructure. Thus, it is critical to effectively process all CV data on a state level for statewide transportation infrastructure management.

The main goal of this chapter is to study the resolution, coverage, and diversity of real-time traffic data streams that enable operators to detect problem areas and respond to them in reasonable time. There is a growing interest among state agencies in leveraging CV data to improve operations, incident management and predictive analytics. The size, coverage, resolution and penetration rates of this new dataset offers new challenges and opportunities that need to be explored prior to full scale integration into day-to-day traffic operations.

## 1.3 Big Data Processing

As automakers scramble to integrate CV technology in new and existing vehicles for several salient benefits including vehicle autonomy and navigation, vehicle sensor and driver monitoring, live over-the-air updates, advanced road warnings, and improved battery and fuel efficiency, the future of transportation infrastructure management is shifting towards real-time big data from CV. In order to know what is happening on the road and make wise judgments about traffic flow and road pavement infrastructure, government and state organizations that design, maintain, and administer road infrastructure may benefit from the CV data readily available. How we efficiently process all CV data at the state level for statewide transportation infrastructure management remains to be seen.

The development of the modern urban economy in cities has been both an indicator of and a driver of the transportation infrastructure, which includes information about the road pavement surface, road networks, signals, and intersections, as well as parking. Processing and analyzing data from transportation infrastructure had not been difficult until

recently, and this is in accordance with Moore's Law, which states that computing power will double approximately every 18 months [6]. The development of Central Processing Unit (CPU)s has put them at the forefront of data processing. Moore's Law is thought to have reached its physical limits, and the emergence and growth of big data across several industry verticals, including finance, social networks, transportation, retail, telecommunications, and biology, has necessitated the adoption of fresh, cutting-edge methods for processing and analyzing this data in order to derive useful insights. Finding fresh and different approaches of digesting large amounts of data to produce useful insights has been the focus of recent research [7–9].

These strategies include quantum computing, Graphics Processing Unit (GPU) data processing, parallel computing, and edge computing (still in the infancy of its development). GPUs are massively parallel processing devices that were initially created to speed up graphics operations on computers. On GPUs originally designed for gaming and visual graphics processing, the advent of Nvidia Compute Unified Device Architecture (CUDA) ushered in the era of extremely parallel scientific computations [10, 11]. Although GPUs have limited computational power, the massively parallel architecture, also known as Sin-

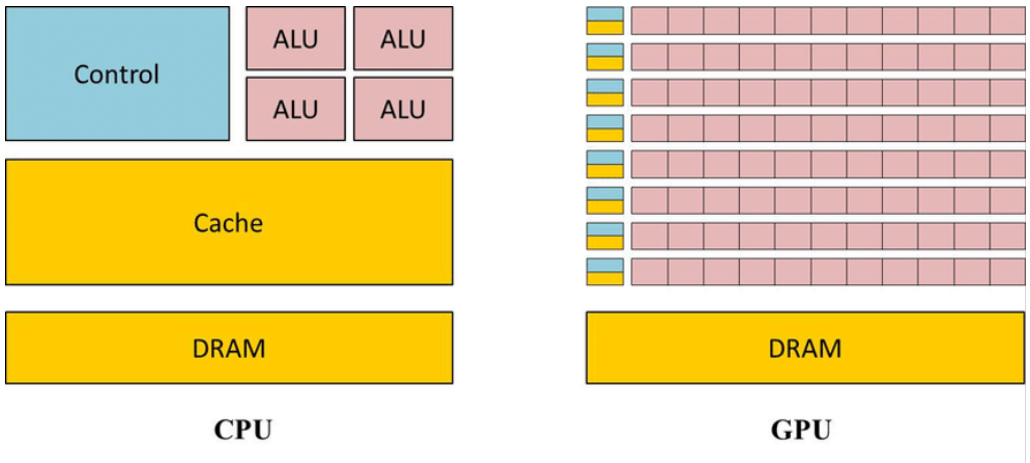



**Figure 1.1:** CPU vs GPU architecture

gle Instruction Multiple Data (SIMD), significantly accelerates simple data processing jobs using independent execution paths (see Figure 1.1).

Launching millions of threads on thousands of processing cores on a single GPU is typically required to make use of the tremendous parallelism of GPUs [12]. Constraints when employing GPUs for data processing typically include GPU memory limits and slow data transmission rates between the CPU and GPU. However, there are ways to get around or cover up certain problems that GPUs have, such as simultaneous data transfer and batch processing. Despite these difficulties, when used properly, GPUs have been shown to accelerate scientific computations up to 200,000$\times$ over CPUs [10, 11, 13]. As a result, the core of all supercomputing infrastructure today consists of GPUs and similar accelerator hardware technologies.

## 1.4 Traffic Forecasting

Traffic forecasting is a critical component of advanced traffic management systems that can help transportation planners in planning for volatile events ahead, by taking early actions and arrangements, which contributes to better traffic management and service quality. It may not only serve as a valuable reference for increasing the efficiency of limited traffic management resources, but it can also assist passengers in making arrangements ahead of time to minimize traffic congestion. Long-term projections are more likely than short-term forecasts to reduce travelers' average trip time [14]. Common forecasted traffic parameters include: traffic flow [15], traffic speed [16], and traffic time [17]. The increasing availability of large-scale traffic data, which can be looked at from a temporal and spatial lens, has paved the way to develop prediction models that are robust to capture the underlying driving mechanism of traffic volatilities, especially the random (unforeseen) components.

Many studies [18–20] have shown that traffic datasets can be used to predict traffic

congestion, allowing drivers to avoid congested areas (e.g., through traffic flow forecasting navigation systems), policymakers to decide on changes to traffic regulations (e.g., replacing a normal lane with a toll lane), urban planners to design better pathways (e.g., adding or removing a road lane), and transportation engineers to better plan for the timing of construction activities.

Temporally, majority of prior studies have focused on single-step traffic flow prediction for a single road or road section. Single-step predictions can be defined as a single value prediction at the next time-step while multi-step would predict a sequence of values into the future (multiple time steps). For some applications in ITS, such as traffic planning, single-step can be insufficient because it doesn't provide enough valuable insights for transportation planners when planning future strategies for traffic management As a result, multi-step traffic flow prediction is gaining popularity. Multi-step traffic flow prediction uses the same methodology as single-step traffic flow prediction in predicting the first time-step; however, for the future multi-steps there are different strategies that can be used such as:

- Direct Multi-step Forecast; where a separate model for each time step is developed.
- Recursive Multi-step Forecast; where the prediction output from the previous time-step is used as an input to predict the next step.
- Direct-Recursive hybrid multi-step Forecast; which combines the previous two strategies.
- Multiple Output Forecast; where the entire multi-step future is predicted in one-shot.

In addition, many studies only focused on predicting traffic on a single-route or a specific connection or crossing. The development of an ITS demands the need to explore multi-route predictions on a larger scale by considering the complex spatial dynamics of

a network [21]. While prior knowledge of the distance or travel time between regions can aid in capturing spatial correlation, there are still some hidden time-varying traffic patterns that data-driven methods must uncover. The challenge is resolving the intricate spatio-temporal dependencies, which refer to traffic information (e.g., speed or volume) at a certain location in space and moment in time. With the emergence of deep learning models, this research aims to solve the question of how to construct appropriate deep learning models to cope with large-scale complex network-wide traffic data.

Large-scale network traffic prediction demands an intelligent and efficient prediction methodology to forecast traffic on longer horizons and reflect the flow propagation. Numerous variables affect a region's future traffic state, including historical observations of traffic, road geometry and network dependencies, weather, incidents, and other external factors (holidays and special events). The technique used to fuse multi-purpose variables is a challenge for the current generation of prediction models when incorporating information from multiple senses together. The interrelationships between regions are intricate and complex, along with the use of big amount of data, adds to the challenges in developing a robust prediction model.

The main goal of this chapter is to create an accurate and reliable network-wide (by exploring multi-routes), multi-purpose (such as speed and volume), multi-step (longer prediction horizon) prediction model.

## 1.5 Web-based Visual Analytics

The increasing complexity of urban transportation networks makes it difficult to manage transportation operations in cities. ITS and Information and Communication Technology (ICT)s are frequently used to handle traffic monitoring, estimate, and control problems. In order to apply suitable control techniques, ITSs combine modern technology with real-time information about traffic conditions. Transportation networks are closely monitored,

resulting in massive traffic and incident databases. The problem of traffic congestion on the roads is serious and widespread, and the integration of various technologies and systems can greatly aid in its resolution. The requirement for massive traffic databases to be efficiently used by traffic operators and managers necessitates the development of innovative apps and state-of-the-art visualization tools as the amount of traffic data transmitted via ITSs grows fast. Interactive visualization allows extracting data of interest by displaying it in various visual forms and interacting with it through various filters.

Most transportation agencies use ArcGIS, Tableau, and D3 as their primary visual analytic platforms. Tableau, an analytical visualization tool, is used by the NHTSA (National Highway Road Safety Administration) to offer insights regarding speed-related traffic fatalities across the United States. Other agencies, like the Virginia Department of Transportation (VDOT2015), the Bureau of Transportation Statistics (BTS) (2019), and the Iowa Department of Transportation [22], employ comparable platforms to dig into work zone, traffic, and freight data. The data being visualized on these platforms might be anywhere from a few megabytes to a few gigabytes in size. When the size of the data being viewed surpasses 250 megabytes, significant latency might be detected in terms of updates. For all the heavy-lifting calculations (on large data sizes) such as data ingestion, aggregation, integration, and reduction, recent advances aimed at managing huge transportation data employ high-performance computing clusters in the backend [23]. The data is then provided to the front end for visual exploration after being filtered, aggregated, and lightweight. Although this method is useful for managing the challenges of massive data, it restricts the effectiveness of visual analytics because tiny details are lost in the aggregate and filtering processes [24].

The main purpose of this chapter is to develop an interactive visual analytics application that allows the big CV dataset (historical and predicted) to be visualized, interacted with, and analyzed in the browser (front end). The framework will make use of CPUs

to allow heavy-lifting computations like data reduction, aggregation, and filtering to be performed with user input from the front end.

## 1.6 Dissertation Objectives

The main purpose of this dissertation is to analyze and understand congestions better. To accomplish this purpose, we decompose the congestion understanding pipleine into predictions, data, processing and analysis. This break up into constituent parts presents the following research questions:

- Could there be a faster way to perform traffic predictions?
- What better sources of data exist and how do they compare to traditionally used sources?
- How can data be processed faster to scale prediction models along longer-horizons?
- How can decisions be made from the forecasted model results?

To accomplish our main purpose we deliver an interactive web application that can perform faster predictions with higher accuracy data. The following objectives, to be discussed in detail through each of the following chapters, answers each of the research questions:

- Developing a GC-GRU Deep Learning architecture that can perform faster predictions on tabular data
- Comparative Analysis of CV data to traditionally used traffic data such as probe. This study answers the question of how a different scale like the microscopic CV point data compares to macroscopic probe line data. Using a multiscale data mining approach, trends are compared between both datasets.

- Desigining a pipeline for performing simultaneous, pixel-level, dense prediction of traffic flow variables (speed and volume) while considering the network traffic temporal evolutions and spatial dependencies.
- Building an interactive web application that analyzes and predicts traffic performance using CV data, for a faster understanding of bottlenecks and situations triggering a degraded traffic performance.

The web app is designed to assist traffic operators in estimating traffic congestion. Using a map layer, it depicts various levels of congestion that are estimated and labeled with green, yellow, red, corresponding to normal, medium, and severe congestion, and updated automatically with the time frequency selected by the user. The second tool aims to make databases more understandable for traffic operators and analysts by providing tabular and graphical depiction, as well as several filtering options. The next sections of the dissertation are organized as follows: the second chapter introduces CV data and compares it to detector and probe data to understand its penetration rate and how it compares when detecting congestion and incident events. In the third chapter, we discuss state-of-the-art machine learning models for timeseries forecasting and propose a faster architecture for similar prediction results. In the fourth chapter, we introduce the different datasets used and how our data processing pipeline was developed to efficiently process large-scale data by leveraging distributed GPU clusters through Nvidia Rapids and Dask Framework. In the same chapter we also present our developed Deep Learning (DL) Framework for simultaneous, pixel-level prediction of traffic flow variables (speed and volume). In the fifth and final chapter we talk about how we developed a web-based platform to visually analyze the historical and predicted traffic data, powered by speech queries.

# Chapter 2

# GC-GRU Deep Learning Architecture for Tabular Data Predictions

## 2.1 Introduction

The path toward an ITS has recently become more feasible due to the influence of two major factors: 1) exponential growth of data collected by embedded traffic sensors, and 2) advancement of effective deep learning techniques [25]. An intelligent transportation system consists of several components, one of which is traffic forecasting. Traffic forecasting is an essential component of an ITS because it predicts future traffic flows on road networks by analyzing both historical traffic data and the configuration of road networks. Forecasted traffic flows are required for several traffic management applications, including traffic control [26, 27], traffic classification [28, 29], and vehicle scheduling [30]. Despite its many benefits, traffic forecasting remains a daunting task.

Traffic forecasting is a challenging task because traffic variables such as speed, volume, and traffic patterns are influenced by dynamic and static factors known as spatiotemporal correlations and external events. The above-mentioned factors can profoundly

influence the performance of a traffic forecasting system directly and indirectly. First, studies have shown that spatial information, precisely the locations of embedded communication sensors, significantly influences a traffic forecasting system [21]. This is because roads in a Euclidean space are bound to have different traffic conditions at any given time [21]. For example, on a two-lane highway network, there is usually a significant difference in the amount of traffic traveling in each lane at any given time. Also, the traffic speed on a given roadway is influenced and directed by the traffic condition further downstream. Second, traffic dynamics and their temporal dependencies can be different from one another by combining recurring patterns and unpredictability of occurrences. For example, traffic follows a cyclical pattern on a daily and weekly basis, but there may be dynamic shifts in temporal patterns due to crashes, and this results in difficulties during traffic forecasting. Third, extraneous factors such as one-time events and weather conditions can significantly impact traffic flow, making long-term traffic forecasting more difficult. According to the FHWA report on weather impacts on mobility, it was found that the average speed of traffic can be reduced by 2 to 13 percent in light rain, 3 to 16 percent in heavy rain and 5 to 40 percent in heavy snow.

Several methodologies for short- and long-term traffic forecasting have been implemented considering the numerous challenges identified in forecasting traffic. There are two types of methodologies usually employed. First, statistical methods such as K-Nearest Neighbor (KNN) [31, 32], Gaussian process, hidden Markov model [33], Support Vector Regression (SVR) [34], and Autoregressive Integrated Moving Average (ARIMA) [35, 36] were used in the past. Typically, these techniques are limited to less complex traffic conditions and situations with small data. Second, deep-learning-based methods, primarily Recurrent Neural Networks (RNN)s and Convolutional Neural Networks (CNN)s. RNNs, such as Long-Short-Term Memory (LSTM) [37] and Gated Recurrent Unit (GRU) [38], are commonly used for sequential and temporal learning, whereas CNNs, such as

ResNet [39], are commonly used for learning spatial structures. ST-ResNet [40] is a time-series model that uses a residual network and LSTM. Although the approaches have produced cutting-edge results, they do not consider the connectivity of road networks. This is significant because traffic conditions on one road will be influenced by another. As such, methodologies need to consider both traffic variables and the road network's configuration.

Most recently, researchers have begun modeling traffic data collected by road sensors using graph-theoretic approaches. The spatial correlations between traffic sensors are represented by a directed graph with nodes representing the sensors and edge weights representing the proximity of sensor pairs as determined by road network distance. Recent advancements in graph neural networks [41], especially convolutional graph neural networks [42], have fueled the development of several graph-based traffic prediction models [41, 43, 44]because sensor networks are naturally organized as graphs, as in the case of Diffusion Convolution Recurrent Neural Network (DCRNN). DCRNN [21] represents the road network as a directed weighted graph and proposes a diffusion convolutional RNN for traffic prediction. Even though GCN has achieved great success in spatial analysis over the years, few studies have investigated using GCN for spatiotemporal analysis. [45] combined recurrent neural networks and Graph Convolution Network (GCN) to perform spatiotemporal analysis. Although a significant result was obtained, the architecture is constrained by the limitations of the RNN, which cannot be used for long-term forecasting due to vanishing and exploding gradients. Other studies, such as [46], investigated combining LSTM and GCN to capitalize on LSTM's ability to learn from long-term dependencies. This was a significant accomplishment, as the results of these studies were state-of-the-art. Even though LSTM has achieved respectable results in recent years, training LSTM takes longer [47].

The primary goal of this chapter is to perform a comparative analysis among various

state-of-the-art models used for traffic forecasting. We also propose a Graph Convolution – Gated Recurrent Unit (GC-GRU) model to perform network–wide traffic forecasting. The dataset used to test the proposed architecture against state-of-the-art models is the benchmark dataset of [48] inductive loop detectors installed on freeways throughout the Greater Seattle area. This dataset contains freeway traffic performance score (TPS), speed, and volume information. The freeways include I-5, I-405, I-90, and SR-520. The traffic states of loop detectors on main lanes traveling in the same direction are aggregated every two miles in this dataset. In our proposed model, the GRU cells was used to model the temporal aspect of the problem, while the GC cells were used to model the spatial aspect of the problem based on the road network configuration using the adjacency matrix. The ability of a neural network to achieve high accuracy is determined by the training data and the hyperparameters used to train the model. Choosing hypermeters to train a neural network can be time-consuming and frustrating because it requires a lot of trial and error. In this study, the we also provide the best practice in choosing hyperparameters to train a GC-GRU to achieve great results. The rest of the chapter is structured as follows. Section two examines previous traffic forecasting methodologies. The third section explains the data used to test our newly developed architecture. Section four goes over the methodology used in developing the new GC-GRU model and the other models used as benchmarks . Section five presents and discusses the results. Sections six and seven present the chapter's conclusion and recommendations.

## 2.2 Related Work

This section reviews methodologies employed extensively by previous studies. Numerous traffic network modeling methodologies have been identified as useful for estimating and predicting traffic patterns. In the past, parametric statistical models were utilized frequently to model traffic data flow. Several studies favored the Autoregressive Integrated

Moving Average Model (ARIMA) because of the model's capacity to model sequential input [49–51]. In addition, various approaches based on machine learning have been suggested as possible ways to model traffic data. In previous studies, researchers made use of techniques such as Support Vector Regression (SVR) [52–54], k-nearest neighbors (k-NN) [55], Bayesian Networks (BN) [56], and feed-forward Neural Network (NN) [57–60]. A new frontier has been opened for traffic modeling based on deep learning due to traffic data's growing pervasiveness, availability, and size. Deep neural network–based methods have recently been shown to achieve high accuracies for traffic estimation and prediction tasks due to the availability of large amounts of data pertaining to traffic. This was made possible by the availability of big data. RNNs, or recurrent neural networks, are a subcategory of deep neural networks developed specifically to model sequential data.

Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs), which are both subclasses of RNNs, were used extensively in recent research to model the spatiotemporal behavior of traffic [47, 61]. For instance, Cui et al. [62] proposed a Stacked Bidirectional and Unidirectional LSTM Network (SBU-LSTM) to model traffic's chronological and reverse chronological temporal dependencies. Many authors used convolutional neural networks to improve the spatial modeling capability of deep learning models. This was done by combining multiple layers of neural connections (CNNs). CNNs can model both local and global relationships between neighboring pixels as they were initially developed for use in computer vision applications. CNNs were utilized in a great number of research projects to model the connection between neighboring stretches of roadway [63, 64]. For instance, Ke et al. [65] modeled lane-wise traffic speed and volume data by employing a CNN with multiple channels to analyze the data. The proposed method was developed to capture the spatial relationship between the traffic lanes that are immediately adjacent to one another.

GCN were also used to model spatial traffic dependencies [66, 67]. These dependen-

cies were derived from the topological structure of the road network. In their study [45], Zhao et al. combined GCN for spatial modeling with GRU for temporal modeling, and the result was a Temporal Graph Convolutional Network (T-GCN). T-GCN was trained and tested for its ability to predict traffic speed using two distinct datasets: data based on a probe-vehicle type and data based on a loop detector. An Attention-based Spatio-Temporal Graph convolutional network (ASTGCN) was proposed in a paper by Guo et al. [68]. Inspired by these works and aiming towards developing an accurate timeseries traffic prediction model, we utilized two cells (temporal and spatial) to propose a GC-GRU architecture.

## 2.3 Data

### 2.3.1 Training Data

In order to understand how the proposed architecture performs against state-of-the-art models, we use the Seattle inductive loop benchmark dataset. Traffic Performance Score in the dataset is calculated based on the data collected from almost 8000 inductive lop detectors deployed on the freeway network in the Northwest region in Washington State. The freeways covered in the study include I-5, I-405, I-90, and SR-520 as shown in Figure 2.1. The dataset consists of traffic performance score, spatio-temporal speed and volume information of the freeway system. Each blue icon represents the loop detectors at a milepost, the speed data at a milepost is averaged from the several loop detectors on the main lanes in the same direction at a particular milepost. The training dataset used in the study is sampled from January 1st, 2020, to May 31st, 2020, at 15-minute sampling intervals. The horizontal header represents the average speed (AVG Spd), average volume (AVG Vol) and traffic index (TrafficIndex) for the general purpose (GP) and High Occupancy Vehicle (HOV) lanes. The vertical header represents the timestamp.

### 2.3.2 Testing Data

In order to test the performance of the model, there are 15 testing data points which will represent the ground truth in the study with 36 previous time steps. The test data covered weekday, weekends, morning and afternoon peak hours. The developed model is required to predict the next 12 steps ahead of 'TrafficIndex GP'.

## 2.4 Methodology

### 2.4.1 GC-GRU Architecture

In this section, we discuss our end-to-end traffic prediction model, designed to tackle the traffic forecasting challenge using the loop detectors data, which consists of spatial-temporal GC-GRU cells for the encoder and decoder components. The overall architecture of the proposed model is illustrated in Figure 2.2. Specifically, the proposed architecture consists of two main parts: encoder and decoder. The encoder module consists of GC and GRU cells to encode the traffic features (input) and the decoder module consists of GC and GRU cells to forecast traffic parameters (output) from the encoded state. GC is employed to learn the complex spatial relationships between traffic detectors and GRU is

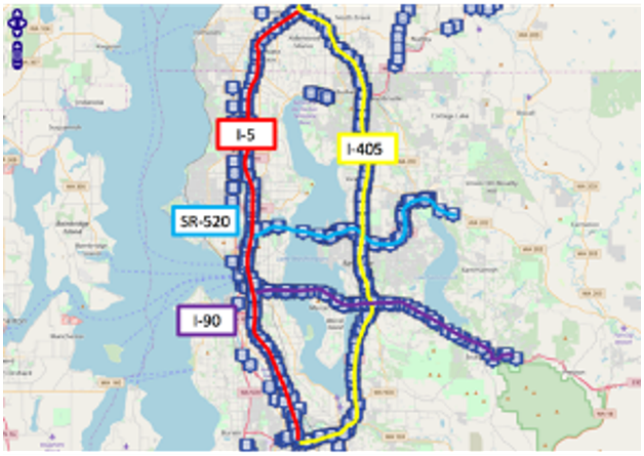



**Figure 2.1:** Freeways within the study area

utilized to capture the dynamic temporal dependencies of traffic data reported by traffic detectors at different times.

### 2.4.2 GC for Spatial Relationships

In the past decade, neural networks have experienced tremendous success. Although many data sets in the real world contain underlying graph structures that are not Euclidean, early neural network variations could only be built using regular or Euclidean data. New developments in Graph Neural Network (GNN) have been made possible by the non-regularity of data structures. GCN are one of the many variations of GNNs that have been developed over the last several years and are regarded as one of the fundamental Graph Neural Networks subtypes. The actions carried out by GCNs are similar to CNN, however the model learns the features by looking at the nearby nodes. The primary distinction between CNNs and GNNs is that the former was developed specifically

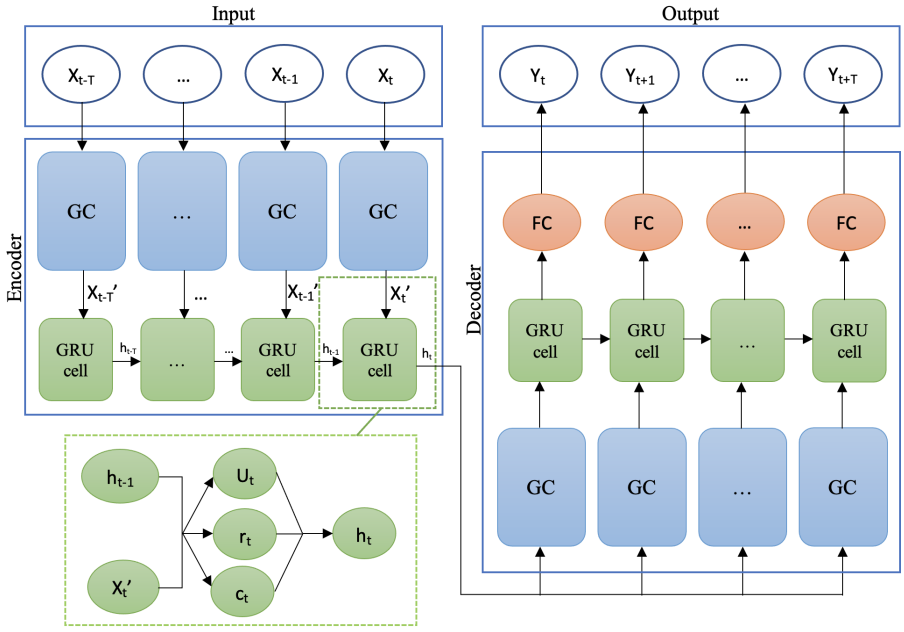



**Figure 2.2:** GC-GRU model architecture

to function on regular (Euclidean) organized data, whilst the latter are generalized CNNs with varying numbers of connections and unordered nodes. In our proposed model, the input to the model is a network of traffic loop detectors represented by nodes ($N$) and edges ($E$). Nodes represent the location or order of the detectors while edges represent the existence of a link in between the nodes. This can be represented by an undirected graph where $G = (N, E)$ with $N$ defined as the set of detectors. One detector $ni$ is connected by vertices $i$ and $j$ which represent the network edges ($ni, nj$). To consider the impact of traffic events, bidirectionally on upstream and downstream roads, we let $G$ be an undirected graph even if certain roads are directed. Subsequently, the Adjacency square matrix ($A$) element for each $E$ is represented as $Ai, j = Aj, j$ with values of 0,1, in which $Ai, j = 1$ for an existing connection between nodes $ni$ and $nj$ and $Ai, j$ = 0 otherwise. Node self-features are also considered by performing a self-loop, adding the adjacency and identity matrix using the following formula for the -hop neighborhood of a node (clipping nodes that might exceed one):

$$\tilde{A}^k_{i,j} = min((A + 1)^k_{i,j}, 1) \tag{2.1}$$

To capture the spatial properties in the coded graph ($G$), the Graph Convolution (GC) layer builds a filter in the Fourier domain that acts on the nodes and their first-order neighbors using the input adjacency matrix ($A$). Adjacency matrix weights for all edges are then input into a diagonal vector ($D$) that now contains information about the degree (edges count) of each vertex. The stronger a node's affiliation with a particular group or cluster, the lower its degree. To ensure numerical stability during training, symmetric normalization is then performed using the following formula:

$$\hat{A}^k_{i,j} = \tilde{D}^k_{i,j}{}^{-\frac{1}{2}} \tilde{A}^k_{i,j} \hat{D}^k_{i,j}{}^{-\frac{1}{2}} \tag{2.2}$$

Output for each GC layer ($z$) is then calculated using the dot product of the normalized adjacency matrix ($\hat{A}^k_{i,j}$) , input feature ($X^k_{i,j}$) and the weight matrix ($W^{k\ i-1}_{i,j}$) of the previous GC layer. Rectified Linear Activation Unit ($ReLu$) is then used as the activation function. The calculation can be expressed using:

$$L^z = ReLu(\hat{A}^k_{i,j} X^k_{i,j} W^{k\ (i-1)}_{i,j}) \tag{2.3}$$

The output from our 2-layer GC model can finally be expressed using:

$$gc(S_t, A) = \hat{A}^k_{i,j} L_i W^{k\ (i)}_{i,j} \tag{2.4}$$

### 2.4.3 GRU for Temporal Relationships

Another significant issue with traffic prediction is acquiring temporal dependence. Given the exceptional time series predictions, the RNN has shown good results over the past decade. However, because of flaws like gradient disappearance and explosion, classical RNN has limitations for long-term projections. To solve these issues, the LSTM cell and the GRU cell are explored as an addition to our proposed architecture. Nearly all the fundamental ideas of both are the same. Both models can handle longer task sequences, and all include gated systems. In contrast to LSTM, GRU reduces the data flow by combining the forget gate and the input gate into one update gate. As a result, it has a simpler structure, fewer parameters, and a faster convergence speed. Each cell in the GRU computes its internal state ($c_t$) and hidden state ($h_t$) based on two gating units that regulate the flow of information: update gate ($u_t$) and reset gate ($r_t$). The update gate decides the amount of information the unit updates its content from the input features while the reset gate decides the amount of previous information to forget. W and b represent the weight and bias matrices in the training process. $\sigma$(.) and tanh(.) denote the sigmoid and

hyperbolic tangent activation functions, respectively. The computations are repeated for each element in the modeled sequence ($t$). The specific calculations can be expressed as:

$$u_t = \sigma(W_u[gc(S_t, A), h_{t-1}] + b_u) \tag{2.5}$$

$$r_t = \sigma(W_r[gc(S_t, A), h_{t-1}] + b_r) \tag{2.6}$$

$$c_t = tanh(W_c[gc(S_t, A), (r_t * h_{t-1})] + b_c) \tag{2.7}$$

$$h_t = (1 - u_t) * c_t + u_t * h_{t-1} \tag{2.8}$$

## 2.5 Data Preprocessing

The traffic loop detectors data and its corresponding adjacency matrix were used to train and evaluate the proposed GC-GRU architecture to predict the future Traffic Index on the General Purpose (GP) lane. The training set is from January 1st 2020 to May 31th 2020 with 15-minute time interval. In other words, the number of rows in the provided data is 14,551 reporting speeds at different segments (87 detectors) and time steps. The respective adjacency matrix provided for the data had a few additional nodes, so it was pruned by detector (node) number to match the information provided in the traffic data. A few 'nones' also existed in the data, indicating non-available information so we changed them to '0', since the model only accepts integers.

## 2.6 Models Evaluation

The test for the model for the models is performed on a separate (unseen) data for 15 days; from June 1st 2020 to June 16th 2020 also with 15-minute time interval. Each day provides 36 time-steps at which the prediction is to be made for the future 12 time-steps. The ground truth is not provided so the predicted values are then submitted as a json file

to the leaderboard for evaluation using the MAPE calculated as:

$$MAPE = \frac{1}{n}\Sigma_{i=1}^{n}(\frac{yi - \hat{yi}}{yi}) \tag{2.9}$$

where $y_i$ and $\hat{yi}$ represent the ground truth and predicted value respectively. The training/testing split on the data was experimented with a split of 80/20 and 90/10. Five datasets are then input into the model; training data, training labels, test data, test labels and adjacency matrix.

## 2.7 Models Setup

### 2.7.1 GC-GRU Setup

Once the training data was cleaned, pre-processed and split for the model, it is fed into the GC cell encoder as a normalized tensor of shape $(N, T_{in}, D)$ where N is the number of rows used for training, Tin is the length of time-steps and D is the number of detectors. The outputs from the GC network $gc(S_t, A)$ for each time is then fed into the encoder GRU cell to encode the temporal features between each time index. The decoder is then used to shape the output in the desired tensor of shape $(N, T_{out}, D)$ where Tout is the length of time-steps for the forecasted horizon. The decoder GC cell uses the encoded intermediate representation vector to decode the spatial relationships and then passes it to the decoder GRU cell to decode the temporal relationships between output time-steps. The flattened output is then passed into a fully connected network layer which uses multi-output sigmoid activated layers to generate the predicted tensor which was then denormalized and reshaped to match the submission format. The loss function used for the proposed architecture is $L2$, defined as the sum of all the squared differences between the ground truth and predicted values then adding a 'regularization term' to avoid overfitting with a lambda loss ($\lambda$) of 0.0015 applied on the weights of the respective pre-

diction ($i$) . The loss function can be expressed as:

$$L(Y_{true}, Y_{pred}) = \frac{1}{NxT_{out}xD}\Sigma_n^N\Sigma_t^{T_{out}}\Sigma_d^D \mid (T_{true})_{n,t,d} - (T_{pred})_{n,t,d} \mid^2 + \lambda\Sigma_{i=1}^M \mid w_i \mid \tag{2.10}$$

### 2.7.2 Transformer Model for Traffic Forecasting

The transformer architecture was originally designed for Natural Language Processing (NLP) to provide contextual meaning for word tokens, which was a missing concept in feedforward networks like RNN, LSTM, and GRU. Prior to the introduction of transformers in NLP, word tokens were typically passed sequentially through NLP architectures.

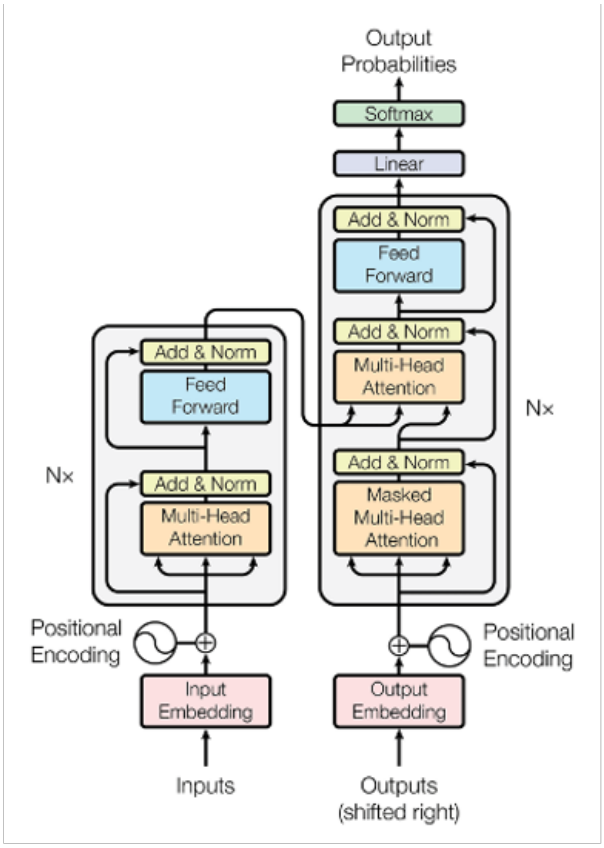



**Figure 2.3:** Transformer Model Architecture

This method results in local understanding of word tokens rather than global understanding. Transformers, on the other hand, introduced a self-attention mechanism that allows each word token to attend to all other word tokens at the same time, giving it a global contextual meaning. Currently, transformers remain the state-of-art model for NLPs and other research areas such as computer vision.

**Table 2.1:** Transformer trained model Parameters

| **Model Parameter** | **Value** |
|---|---|
| Batch size | 40 |
| Sequence length | 36 |
| Prediction length | 12 |
| Learning rate | 0.00001 |
| Epochs | 100 |
| Minimum delta | 0.0005 |
| Patience | 10 |
| Model dimension | 512 |
| Model number of heads | 8 |
| Model number of layers | 6 |
| Model dropout | 0.3 |

The transformer network shown in Figure 2.3 employs an encoder-decoder architecture similar to that of recurrent neural networks. The difference is that the input sequence can be passed in parallel. The encoder block accepts both input embeddings and positional embeddings. The encoder network is made up of a multi-head attention and a feedforward neural network. The attention layer computes an attention vector for each word token. The attention vectors are fed into the feedforward network, one vector at a time. Each attention network is self-contained, allowing for parallelization. The feedforward network is a simple neural network that is applied to each of the attention vectors. In practice, feedforward networks are used to convert attention vectors into a format that can be processed by the next encoder or decoder block. The feedforward network's final output is passed to the decoder block. The decoder block is made up of three components, two of which are similar to the encoder block. The decoder output is passed

through a linear layer before being passed through a SoftMax to calculate probabilities. The model was adopted from [69] and a summary of the model hyperparameters is presented in Table 2.1.

### 2.7.3 LSTM Model for sequence-to-sequence Traffic Forecasting

LSTM, or long short-term memory, is a particular class of RNN that can learn long-term sequences. Long-term reliance issues are specifically avoided in its design. Its method of operation involves recalling lengthy sequences over an extended period of time. The fact that each LSTM cell has a mechanism involved contributes to the popularity of LSTM. In a typical RNN cell, the activation layer transforms the input at the time stamp and the hidden state from the previous time step into a new state. In contrast, the LSTM process is a little more complicated because it requires input from three different states at once: the current input state, the short-term memory from the previous cell, and finally the long-term memory.

Figure 2.4 depicts the usual layout of an LSTM memory block with a single cell. An outside input is received by the input gate, which processes the fresh data. The forget gate chooses the ideal time lag for the input sequence by determining when to forget the

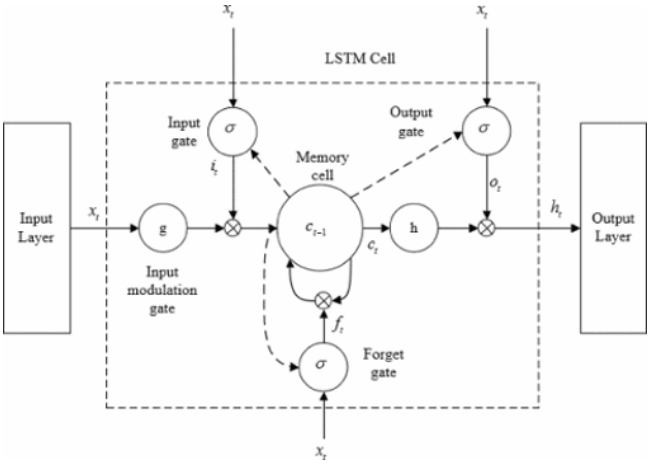



**Figure 2.4:** LSTM Model Architecture

**Table 2.2:** LSTM trained model Parameters

| Model Parameter | Value |
|---|---|
| Batch size | 40 |
| Sequence length | 36 |
| Prediction length | 12 |
| Learning rate | 0.00001 |
| Epochs | 100 |
| Minimum delta | 0.0005 |
| Patience | 10 |
| Input dimension | 87 |
| Hidden dimension | 87 |
| Output dimension | 87 |

prior state. The output gate produces output for the LSTM cell using all the calculated results. In language models, a soft-max layer is typically introduced to control the NN's final output. On the output layer of the LSTM cell in our traffic flow prediction model, a linear regression layer is used. The model was adopted from [69] and a summary of the model hyperparameters is presented in Table 2.2

## 2.8 Results

Table 2.3 summarizes the training time, inference time and leaderboard scores for our proposed model along with the benchmark models. A Historical Average (HA) model was also added as an additional benchmark to evaluate the performance of a simple statistical estimation. Our model performance on the provided test data ranked second in terms of Mean Absolute Percentage Error (MAPE) which is very close to Transformer's performance. We also evaluated additional parameters such as training and inference time and found that LSTM had the fastest training time which was expected given the simplicity of the model architecture when compared to the other models. It's worth noting that our model not only had the fastest inference time, which is one of the crucial

factors when implementing the model in real-time applications, but also had a training time that is six times faster than transformer. Figure 2.5 reflects a violin plot of the distribution of errors Mean Absolute Error (MAE) along different models with a scaled distribution amplitude. HA has the highest distribution of errors followed by LSTM. Comparing GC-GRU and Transformers at bigger errors, we can observe that GC-GRU has a smaller distribution.

**Table 2.3:** Summary of model results

| | LSTM | Transformer | GC-GRU |
|---|---|---|---|
| Leaderboard MAPE | 4.50 | 3.12 | 3.16 |
| Training time (sec) | 76.23 | 1,397.88 | 217.79 |
| Inference time (sec) | 4.58 | 8.84 | 2.04 |

### 2.8.1 GC-GRU for Traffic Forecasting

The proposed model was implemented with the aid of Tensorflow deep learning library [70] and the GridSearchCV was used from scikit-learn [71] to find the optimal hyperparameters that can optimize the model performance. Table 2.4 summarizes the hyperpa-

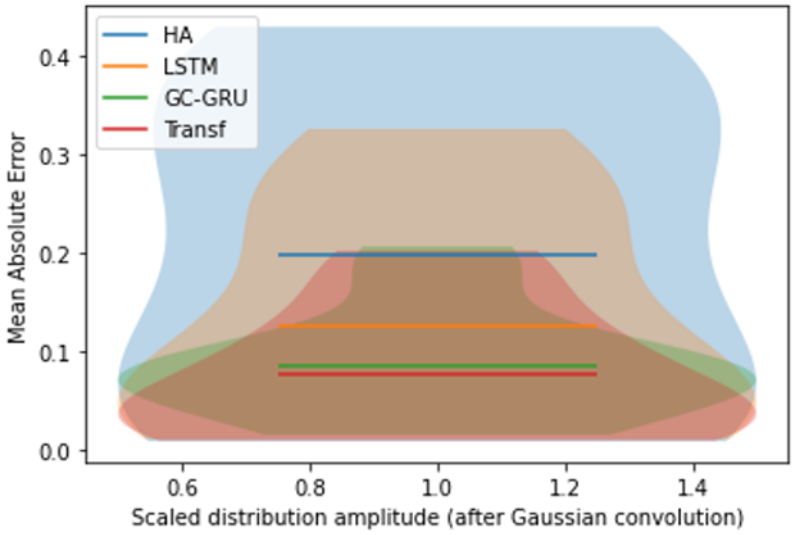



**Figure 2.5:** MAE visualized distribution over segments

rameters values used and Figure 2.6 illustrates the results in a 3D bubble plot along different model parameters, colored by MAPE. The selected bubble highlights the parameters for the best performing model.

**Table 2.4:** GC-GRU Model training parameters

| Model Parameter | Value |
|---|---|
| Training ratio | 0.8, 0.9 |
| Sequence length | 36 |
| Prediction length | 12 |
| Learning rate | 0.0001, 0.001, 0.01 |
| Batch Size | 8, 16, 32, 64 |
| Epochs | 20, 50 |
| Dropout | 0.1, 0.2, 0.5 |
| GC layers | 2, 3, 4 |
| GC layer size | 4, 8, 12, 16 |
| GRU layers | 2, 3, 4 |
| GRU layer size | 16, 32, 64 |
| Optimizer | AdamOptimizer |

Table 2.5 presents the results for each of the trained models. The models are evaluated along the forecasted horizon of 12 time-steps or 3-hours (12 time-steps divided

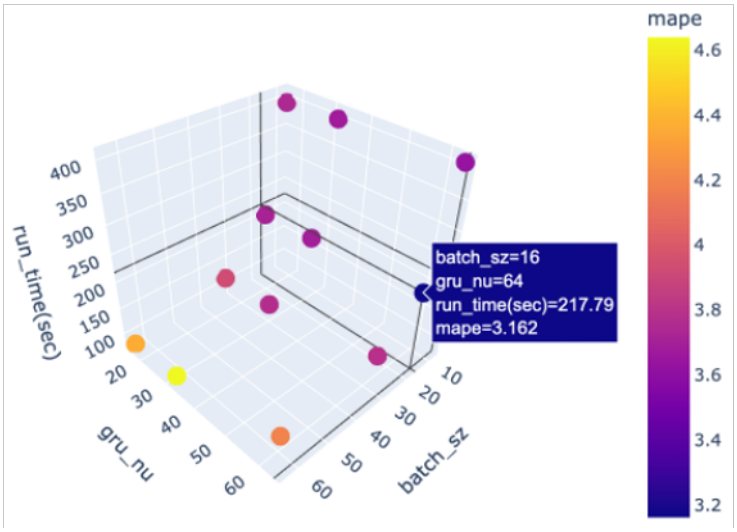



**Figure 2.6:** 3D bubble plot of performed experiments

**Table 2.5:** Prediction results of the proposed model and baseline models

| Horizon | Metric | HA | LSTM | Transformer | GC-GRU |
|---|---|---|---|---|---|
| | | $1-hour forecast$ | | | |
| 15 min | MAE | 0.131 | 0.07 | 0.048 | **0.043** |
| | RMSE | 0.221 | 0.108 | 0.083 | **0.066** |
| | MAPE | 0.248 | 0.109 | 0.076 | **0.069** |
| 30 min | MAE | 0.132 | 0.079 | 0.053 | **0.052** |
| | RMSE | 0.231 | 0.118 | 0.087 | **0.08** |
| | MAPE | 0.272 | 0.132 | **0.079** | 0.086 |
| 45 min | MAE | 0.141 | 0.08 | 0.058 | **0.054** |
| | RMSE | 0.239 | 0.122 | 0.092 | **0.082** |
| | MAPE | 0.294 | 0.144 | **0.089** | 0.092 |
| 60 min | MAE | 0.154 | 0.088 | 0.059 | **0.058** |
| | RMSE | 0.252 | 0.136 | 0.094 | **0.089** |
| | MAPE | 0.314 | 0.157 | **0.09** | 0.095 |
| | | $2-hour forecast$ | | | |
| 75 min | MAE | 0.171 | 0.102 | **0.068** | 0.075 |
| | RMSE | 0.276 | 0.16 | 0.11 | **0.105** |
| | MAPE | 0.381 | 0.207 | **0.115** | 0.131 |
| 90 min | MAE | 0.162 | 0.1 | 0.07 | **0.067** |
| | RMSE | 0.27 | 0.158 | 0.108 | **0.097** |
| | MAPE | 0.372 | 0.208 | **0.114** | 0.118 |
| 105 min | MAE | 0.172 | 0.104 | 0.072 | **0.071** |
| | RMSE | 0.277 | 0.168 | 0.105 | **0.1** |
| | MAPE | 0.404 | 0.236 | **0.127** | 0.135 |
| 120 min | MAE | 0.181 | 0.111 | **0.079** | 0.086 |
| | RMSE | 0.281 | 0.18 | **0.113** | 0.114 |
| | MAPE | 0.413 | 0.253 | **0.141** | 0.159 |
| | | $3-hour forecast$ | | | |
| 135 min | MAE | 0.18 | 0.112 | **0.076** | 0.084 |
| | RMSE | 0.275 | **0.182** | 0.111 | 0.115 |
| | MAPE | 0.393 | 0.251 | **0.139** | 0.155 |
| 150 min | MAE | 0.176 | 0.12 | **0.08** | 0.096 |
| | RMSE | 0.272 | **0.195** | 0.116 | 0.131 |
| | MAPE | 0.393 | 0.273 | **0.155** | 0.183 |
| 165 min | MAE | 0.176 | 0.126 | **0.079** | 0.100 |
| | RMSE | 0.274 | 0.205 | **0.121** | 0.145 |
| | MAPE | 0.39 | 0.285 | **0.157** | 0.197 |
| 180 min | MAE | 0.158 | 0.121 | **0.071** | 0.095 |
| | RMSE | 0.257 | 0.196 | **0.114** | 0.146 |
| | MAPE | 0.347 | 0.265 | **0.139** | 0.19 |

by four 15-minute intervals per hour). The evaluation is performed on the test data split from the provided traffic features data and all three models use the same exact data for testing. We split the table along each forecasted hour to better understand how each model performs on different horizons. The metrics used for evaluation are MAE, Root Mean Squared Error (RMSE) and MAPE. MAE is defined as the average magnitude of the

differences between the predicted and observed true values while, RMSE is the standard deviation or a measure of how spread the differences between predictions and observed true values are, and MAPE is defined as how far the predicted values are away from the corresponding observed truth on average.

In general, the proposed GC-GRU model achieved the best results along 1-hour forecasted horizon while transformers were dominant along the 3-hour forecasted horizon. Along the 2-hours horizon both our model and Transformer seemed to share equal success when looking at the performance metrics. The HA model performed the worst in predictions demonstrating the need for a deep-learning model. The stand-alone LSTM model seemed to struggle in performing predictions with errors almost double in magnitude when compared to the other two models. This signifies the fact that learning from temporal dependency only is not enough and the synergetic effect of combining spatial and temporal dependencies is critical to maximize the network's learning capabilities for more accurate predictions.

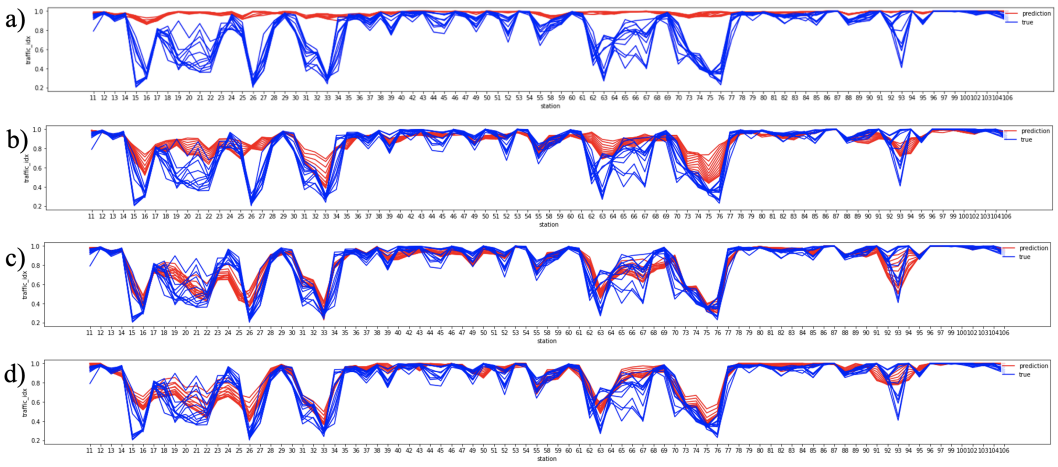



**Figure 2.7:** Visualization results of all prediction horizons along traffic detectors for: (a) HA, (b) LSTM, (c) Transformer, (d) GC-GRU

### 2.8.2 Spatial Analysis of Trained Models

Figure 2.7 presents the predicted values of each model and the observed truth along each detector. The plot is for all predicted time-steps, so we have twelve lines for predictions and twelve for the true values. Once again, the HA had the worst performance with predictions very far off the ground truth. Generally, we can observe that Transformer and GC-GRU predicted peak values much better than LSTM. Transformer predictions seem to be much closer to true values during peaks than GC-GRU and we suspect that may be due to the smooth filter applied by the GC that captures spatial features by constantly moving the filter. This leads to much smoother peaks. The results also show that there still seems to be a certain error even at non-peaks which can be explained by the fact that there are times when we don't have traffic data, or the values are very small.

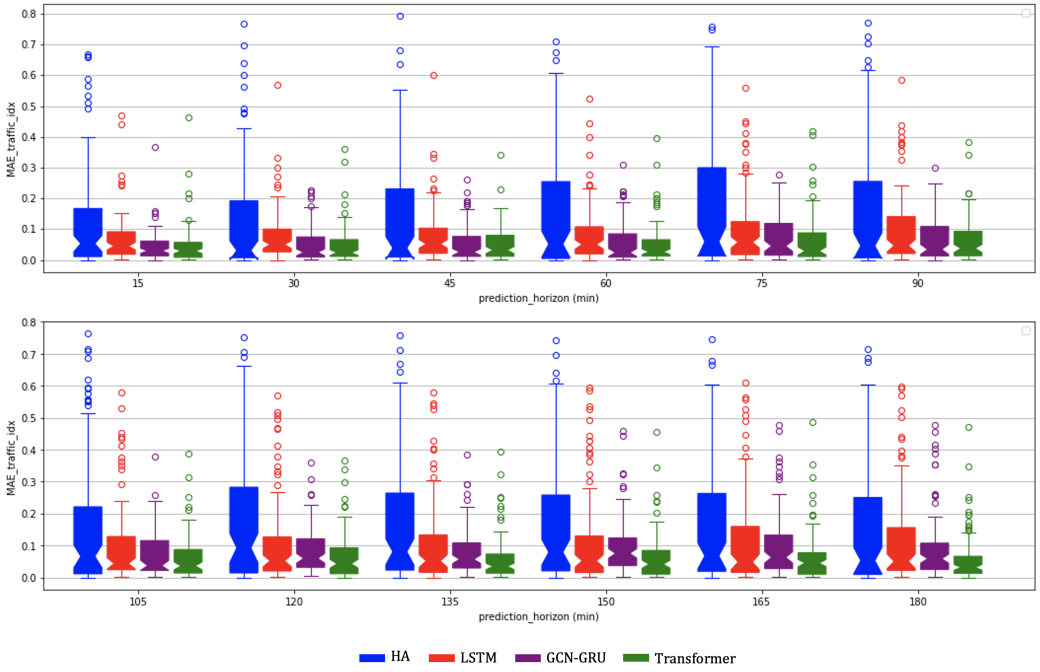



**Figure 2.8:** MAE box-plots of all traffic stations results along future prediction horizons

### 2.8.3 Temporal Analysis of Trained Models

Figure 2.8 presents the MAE box-plot for each model along the forecasted horizon. Box plots provide a standardized way of interpreting the distribution of errors based on the minimum, maximum, median, 25th and 75th percentiles and the outliers. In a box-plot, density of values is inversely proportional to the size of the box which means that smaller boxes have a higher number of values packed in their respective range. HA model had the largest amount to errors as expected. MAE box-plots for the other models increase over the length of the forecasted horizon indicating a positive association between the forecasting errors and the distance to future predictions. LSTM seemed to have the highest range and distribution of errors when compared to Transformer and GC-GRU. Comparing our proposed model to Transformer, we can notice that our model had smaller errors (better performance) on the closer predictions up to 60 minutes ahead, after which the Transformer model has a better performance on the longer horizon.

## 2.9 Summary

In this chapter, we performed a comparative analysis of various traffic forecasting methods as well as introduced a GC-GRU based neural network-based traffic forecasting method. To model the aggregated loop detector data , we utilized a graph convolution, in which the nodes on the graph represent the roads, the edges indicate the connections between the roads, and the attribute of the nodes on the graph is the traffic information on the roads. In order to acquire the spatial dependence, the GC cell is used to capture the topological structure of the graph. In order to obtain the temporal dependence, the GRU cell is used to capture the dynamic change in node attribute. We then performed a comparative analysis on the proposed model along with benchmark models such as: HA, LSTM and Transformer. In summary, our GC-GRU model performance on the provided test data ranked second with a MAPE of 3.16 which is very close to Transformer's performance of

3.12. In terms of training and inference time, we found that LSTM had the fastest training time which was expected given the simplicity of the model architecture when compared to the other models. It's worth noting that our model not only had the fastest inference time, but also had a training time that is six times faster than Transformer. Comparative analysis of the results on the test data demonstrates that the proposed GC-GRU is a strong competitor to state-of-the-art traffic forecasting approaches.

# Chapter 3

# Comparative Analysis of Connected Vehicles and Probe Data.

## 3.1 Introduction

TMCs utilize real-time traffic information to help relieve traffic congestion and improve safety. This requires operators to constantly monitor road conditions through data streaming from a variety of sources including traffic sensors, Global Positioning System (GPS)-enabled devices (probes), closed-circuit cameras, dynamic message signs, etc. The accuracy, resolution, coverage, and diversity of real-time traffic data streams enable operators to detect problem areas and respond to them in reasonable time. There is a growing interest among State agencies in leveraging connected vehicle data to improve operations, incident management and predictive analytics. The size, coverage, resolution and penetration rates of this new dataset offers new challenges and opportunities that need to be explored prior to full scale integration into day-to-day traffic operations. The current paper evaluates this new dataset and compares it to existing traffic data sources for congestion and incident detection.

CV technology can be defined as an application that utilizes V2X communications to address mobility and safety concerns on roadways. CV data availability has been exploding in recent years. This is as a result of the advent of OEMs, Telematics platforms, and other in-vehicle technologies, that are able to continuously stream high-resolution, reliable and accurate vehicle data. A probe vehicle feature, which is part of connected vehicle technology, collects data about the state of the vehicle. Information from the collected data is used to estimate some critical performance indicators such as travel time, a critical parameter in traffic management.

The use of CV data has demonstrated improved traffic performance. Paikari and Far [72] investigated the impact of Vehicle-To-Vehicle (V2V) communication and ITS applications on traffic safety and mobility The study demonstrated that CVs have the potential to significantly improve traffic safety and mobility. Olia et al. [73] modeled the exchange of information between connected vehicles using Paramics simulations. The study's findings indicate that if CVs were used, travel time could be reduced by 37 percent. Olia et al. [74] also assessed the impact of connected and automated vehicle technology on highway system capacity through the development of an analytical framework that demonstrated that CV has potential to increase highway capacity by 300 percent. Vander Werf et al. [74] conducted a study to determine the effect of vehicle communication on highway capacity. The study concluded that by maintaining a 0.5-second gap between CVs, the capacity of the highway could be doubled under certain conditions. When vehicle communication was implemented in a four-lane highway merger scenario, van Arem et al. [15] found that it had a statistically significant effect on traffic flow. The analysis discovered a slight improvement in traffic flow efficiency when vehicles were not equipped. Additionally, a microsimulation of vehicle-to-vehicle communication on a freeway with an on-ramp was used to assess the effect on traffic performance [75]. The study demonstrates that CV significantly impacted traffic flow as measured by the Market Penetration

(MP). In comparison, Arnaout and Bowling [76] used a microscopic traffic simulator to investigate the feasibility of improving traffic flow for high occupancy vehicles (HOVs) on a four-lane freeway at lower MP levels (CVs). Congestion could be significantly reduced if vehicles utilized HOV lanes at a rate of up to 40percent of the MP.

Mekker et al. [77] integrated CV data and Light Detection and Ranging (LiDAR) data to evaluate the impact of work zone geometry on traffic operations. The authors of the study considered two case studies where geometric anomalies were identified. The study discovered that the work zone features in both case studies did not conform to project specifications but were difficult to assess safely by an inspector on the field due to the high volume of traffic. The authors suggested utilizing connected vehicle data to identify recurring congestion and LiDAR to evaluate work zone geometry. Li et al. [78] conducted a study using CV data to reassess dilemma zone performance of heavy vehicles. The study had three objectives; (1) to assess whether matching Basic Safety Messages (BSM)s to virtual waypoints provides sufficient performance for dilemma zone mitigating tactics; (2) to develop a dilemma zone mitigating tactic for CV; and (3) evaluate the performance of the tactic using Automated Traffic Signal Performance Measures (ATSPM) data. The study used BSM data to map-match virtual waypoints. Also, the ATSPM projection indicated that dilemma zone incursions would break even for the northbound approach, with a net reduction of 34 percent for the southbound approach. The study concluded by recommending a more robust control support for dilemma zones and other emerging CV applications.

The goal of this chapter is to compare connected vehicle data and traditional probe data based on traffic flow estimates and their ability to detect congestion and traffic incidents. A data conflation methodology is developed to integrate CV data with traditional traffic data sources such as probe and Waze data feeds. The study analyzed speed bias trends using a multiscale data mining approach. To the best of the authors' knowledge,

this is the first study that integrates real world connected vehicle data with Probe and Waze data for comparative analysis. The remainder of the chapter is structured as follows. Section two summarizes previous studies on connected vehicles based on the type of data used in their research. The chapter's data is presented in section three. Section four discusses the methodology used in this study. Section five presents the findings and analysis of this study. Finally, section six discusses the conclusion and recommendations.

## 3.2 Related Work

This section reviews the previous work in understanding the reliability of CV data and its applications. It was discovered that the State of Maryland's Vehicle-Mile-Travelled (VMT) can be inferred from CV data with as little as 1.5–2 percent penetration [79]. Additionally, GPS-based ATSPM with increased coverage and scalability have been developed using CV data. With as little as 0.04 percent [80], a rapid diagnosis of the current signal performance problems can be made. Using CV data, it was possible to identify arterial congestion based on the percentage of slow-moving vehicles and queue propagation around freeway bottlenecks [81]. Additionally, a microsimulation of vehicle-to-vehicle communication on a freeway with an on-ramp was used to assess the effect on traffic performance [82]. The study demonstrates that CV significantly impacted traffic flow as measured by the market penetration (MP). In comparison, Arnaout and Bowling [83] used a microscopic traffic simulator to investigate the feasibility of improving traffic flow for high occupancy vehicles (HOVs) on a four-lane freeway at lower MP levels (CVs). Congestion could be significantly reduced if vehicles utilized HOV lanes at a rate of up to 40 percent of the MP.

Mekker et al. [84] integrated CV data and LiDAR data to evaluate the impact of work zone geometry on traffic operations. The authors of the study considered two case studies where geometric anomalies were identified. The study discovered that the work zone

features in both case studies did not conform to project specifications but were difficult to assess safely by an inspector on the field due to the high volume of traffic. The authors suggested utilizing connected vehicle data to identify recurring congestion and LiDAR to evaluate work zone geometry. Li et al. [85] conducted a study using CV data to reassess dilemma zone performance of heavy vehicles. The study had three objectives: 1) to assess whether matching BSMs to virtual waypoints provides sufficient performance for dilemma zone mitigating tactics; 2) to develop a dilemma zone mitigating tactic for CV; and 3) evaluate the performance of the tactic using automated traffic signal performance measures (ATSPM) data. The study used BSM data to map-match virtual waypoints. Also, the ATSPM projection indicated that dilemma zone incursions would break even for the northbound approach, with a net reduction of 34 percent for the southbound approach. The study concluded by recommending a more robust control support for dilemma zones and other emerging CV applications.

Additional applications of Wejo trajectory data were used by Saldivar-Carranza et al. [86] to evaluate the traffic impact on nearby arterial roads, particularly for the unauthorized detour that was made possible by Google Maps Navigation. Over the course of the 11 weeks, volumes during the weekly afternoon peak hour, split failures, travel time, downstream bottleneck, and arrivals on green were monitored. Another related application of near ubiquity is [87] using Wejo data for areas where sensing is scarce or nonexistent sensing infrastructure. In addition, the same data was utilized by Khadka et al. [88] to directly measure queue length and its spread on motorway bottlenecks. In Arlington, Texas, an interstate portion was discretized into 0.5-mile segments and a local empirical speed threshold of 45 mph was applied. Although just a portion of the total traffic stream was included in the sample trajectory, a strong link between trip time and slow trajectories were found. To summarize the main advantages of using CV data in literature are:

- Greater spatial precision - OEM telematics frequently incorporate GPS and cellular antennae into the car for more dependable telemetry, producing data of greater quality.
- Additional data attributes: In addition to location, CV data frequently include details about how the car operates, like if the wipers are on or off, whether seatbelts are being used, and wether it makes hard stops.
- Data is reported at a constant higher frequency every few seconds.
- Underpins new mobility propositions.

## 3.3 Methodology

This section discusses the use of the three collected datasets (CV, Probe and Waze) described in the previous section to fuse them together and integrated using spatiotemporal conflation which maps the points data from connected vehicles and Waze dataset to road line segments from probe dataset into a unified data layer. Construction of road segments from road shapes from the conflated datasets is then performed. With datasets conflated in time and space, a framework for comparative analysis is developed based on speed differentials, incident and congestion trend analysis. The methodology adopted for evaluating connected vehicle data consists of several key components as illustrated by Figure 3.1.

Studying the resolution, coverage, and diversity of real-time traffic data streams is critical to enable operators to detect problem areas and respond to them in reasonable time. There is a growing interest among state agencies in leveraging CV data to improve operations, incident management and predictive analytics. The size, coverage, resolution and penetration rates of this new dataset offers new challenges and opportunities that need to be explored prior to full scale integration into day-to-day traffic operations. The

study area of our analysis in the current chapter is the city of Saint Louis and the Figure 3.2 presents a visual instance (colored by speed) of each of the used datasets; CV, Waze and Probe.

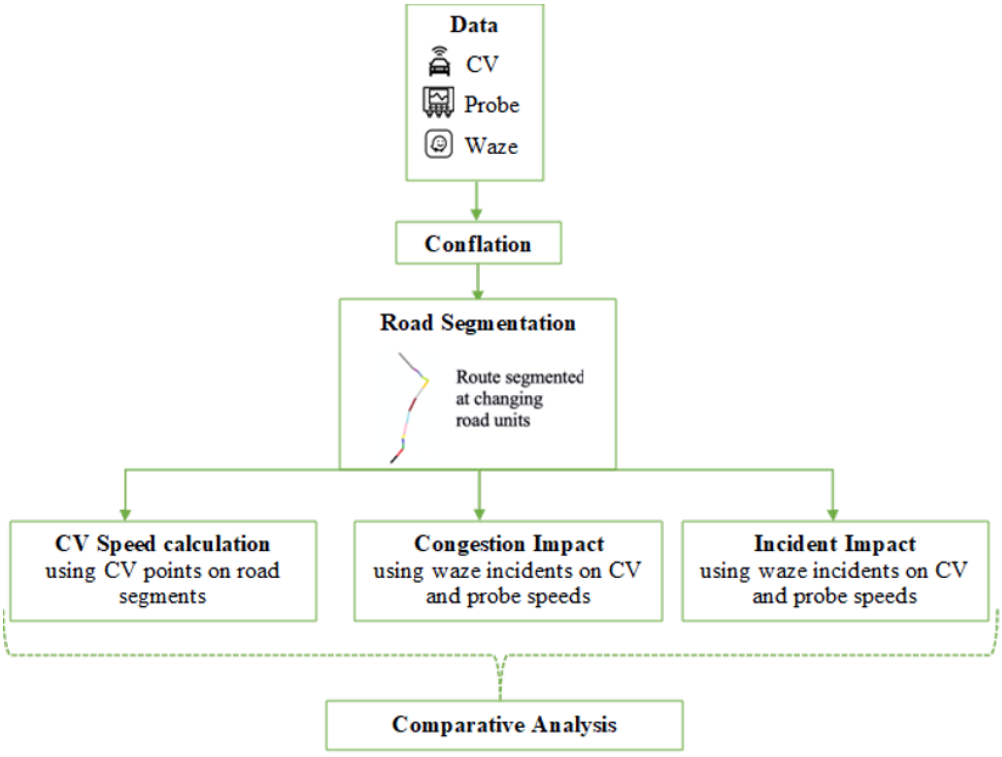



**Figure 3.1:** Overview of Probe vs CV data Comparative Analysis


**Figure 3.2:** Data Sources and Analysis Region: a) Connected vehicle trajectories. b). Waze incidents. c). INRIX probe data.

### 3.3.1 Connected Vehicles Data

The CV data landscape has changed from a few cars to OEM data providers (like INRIX, Wejo and Otonomo) who have compiled millions of equipped vehicle data points from commercial fleet operations to passenger cars. Despite differences between OEMs, these kinds of data are gathered through the OEM's telematics system using the built-in wireless communication capability in the most recent vehicles. One of the main benefits of such commercialized CV data over ad hoc CV data or CV data combined with handheld devices is the granularity. This offers enormous possibilities for incident management, operation, and maintenance of transportation systems. These newly developed data sets offer vehicle telemetry data in addition to high-resolution waypoint data (e.g., Movement –vehicle trajectory, acceleration (including lateral), geo-position, speed, heading, trips and Event Information –hard-braking, seat-belt status and other discrete events.). According to the National Renewable Energy Lab report, INRIX dataset covers 10 percent of all automobile travel in the US [89]. However, the 10 percent penetration rate is not evenly distributed in the geographical and temporal dimensions, thus coverage may be limited in some isolated places or at night. In comparison to on-board devices, data obtained from smartphones with low power consumption profiles is typically sparser. According to internal estimates by Wejo, a top CV data provider, it receives data from 1 in 20 automobiles in the United States and 1 in 50 vehicles in Europe. According to other reports, Otonomos' platform includes more than 4 billion data points from more than 40 million registered automobiles [90]. This study uses CV data obtained from Wejo Group Ltd. It offers vehicle waypoints (latitude and longitude), a time stamp, instantaneous speed, the direction the vehicle is traveling, and other metrics. For a vehicle, it is discovered that the typical ping interval between two consecutive waypoints is 3 seconds. The data's spatial resolution has a 6-digit decimal point, or a resolution of roughly 3 meters (lane-level resolution). Given its granularity, sample size, and coverage, the volume of the data is a bar-

rier in terms of data storage, processing, visualization, and analytics. Figure 3.3 presents a snapshot of point CV data in the State of Missouri, colored by speed of vehicles.

The Data collection process by Wejo starts with the OEM who receives data from automobiles, packages it, cleans it from errors, and then transfers it to Wejo, who consolidates and transmits it in real time. The procedure has a maximum latency of 60 seconds, or 30 to 60 seconds from the time it leaves the vehicle until it is prepared for applications. The Wejo CV data is collected in CSV format presenting the following attributes:

- Journey ID – the identification number for the trip performed by a specific vehicle from start to finish
- Captured Timestamp – a 19-digit time format used by Wejo to define the year:month:day:hour:minute:second (e.g., 2021-02-09 10:05:34 for September 2nd, 2021, at the 10th hour, 5th minute and 34th second) for each record.
- Latitude – geographic coordinate describing the north-south location of a point on

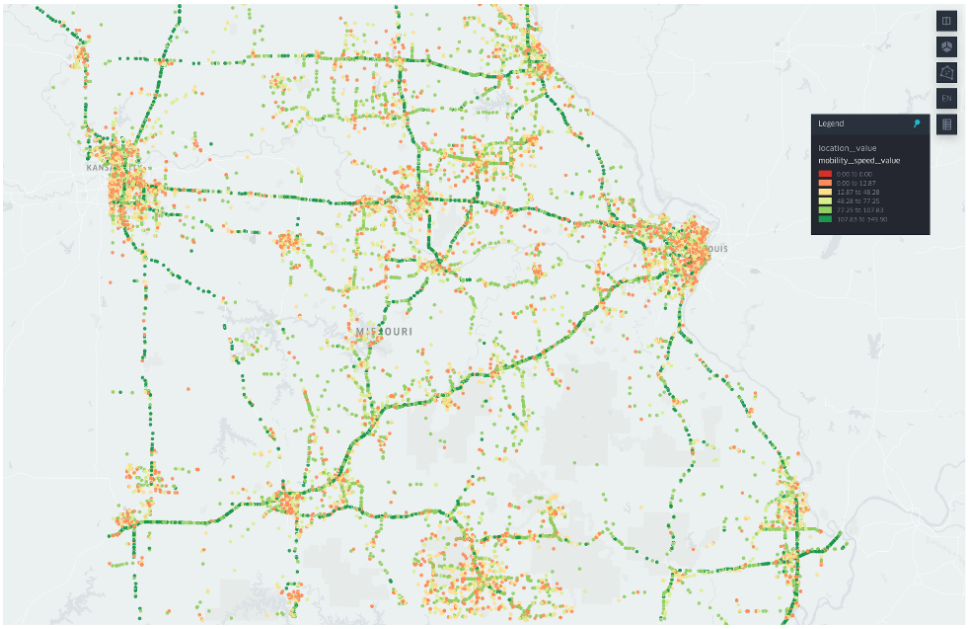



**Figure 3.3:** Snapshot of point CV data in the state of Missouri, colored by speed

the earth's surface. The angle ranges from -90 degrees at the south pole to 90 degrees at the north pole with 0 degrees at the equator.

- Longitude - geographic coordinate describing the east-west location of a point on the earth's surface. The angle ranges from -180 degrees at the west pole to 180 degrees at the east pole with 0 degrees at the prime meridian.
- Speed – representing the current speed of the vehicle at the captured timestamp or record in kilometers per hour.
- Heading – representing the angle of direction of the vehicle at the captured timestamp or record in degrees.
- Ignition Status – the operation status of the vehicle
- Event Type – the type of journey the vehicle is making
- Acceleration Type – type of acceleration which is calculated based on the change in speed of the vehicle
- Journey Event Type – status of the vehicle event calculated on its location along its journey's route.
- Postal Code – series of digits representing the location of the geographical area

Table 3.1 presents a row sample of the collected CV data. As observed, there is a variation in the data types across different attributes/columns such as: characters; integers; floats and strings. We can also notice that at certain attributes such as '`location_road_name`', the field value includes a comma, which is character used to separate fields or values in a CSV file.

To understand the amount of CV points as a newer source of data, we compare it to conventionally used loop detector data and estimate its penetration rate for a specific

**Table 3.1:** CSV sample (one row) for the collected CV Data.

| Attribute | Value |
|---|---|
| vehicle identification otonomo id | 1ddb0685638666fb70a4eb6fb4bb9851 |
| metadata time epoch | 1639245925350 |
| location country code | US |
| location latitude value | 38.8084493 |
| location longitude value | -90.8640632 |
| mobility heading angle | 137.03 |
| mobility speed value | 91.73238 |
| mobility acceleration value | 0.13052416 |
| mobility acceleration lateral | 0.00815776 |
| metadata provider name | 9e00681ca48f27ea4c05b3485245a9 |
| location polygon geohash | 9yzku2fg9g7c |
| location country name | United States |
| location state name | Missouri |
| location county name | Saint Charles County |
| location town name | Wentzville |
| location road name | I 70, US 40 |
| location road id | I 70, US 40 |
| location zone postal code | 63385 |
| location road speed limit | 104 |
| location road type | 1 |

road along the hours of day. Figure 3.4 presents the loop detectors (red points) and the different CVs colored by their unique ID. The data is presented for I-70 on 19th February 2021.

To calculate penetration rate, a buffer of size 20m is used around each detector and the number of unique CVs passing through the detector during a specific time period (one hour) is recorded. The filters used on both datasets are then location, date and hour. The volume from CV data is then divided by the reported volume from detectors to calculate the penetration rate. It's worth nothing that penetration rate varies between different times of day and locations and so we averaged the rates spatially and temporally, achieving a rate of 8 percent. The variations of volume between detectors and CV data are presented in Figure 3.5.

### 3.3.2 Probe Data

INRIX Probe data also provides high-resolution, segment by segment traffic speed and travel time information from millions of GPS-enabled vehicles, mobile devices, and other sources. The data collected is processed near real-time, creating aggregated traffic infor-


**Figure 3.4:** Map visualization of CV data (small colorful points) and Detector data (large red points)

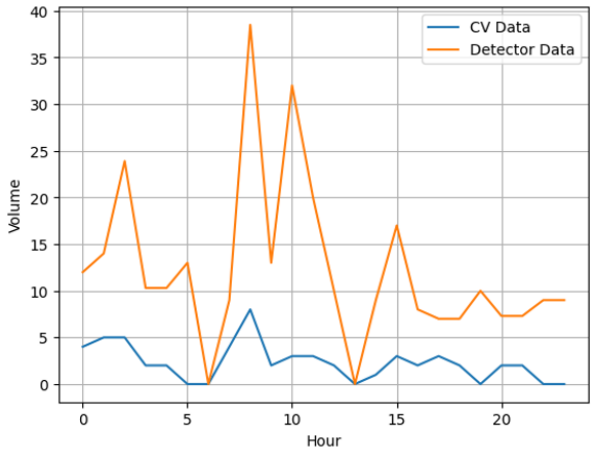



**Figure 3.5:** Variations of CV and Detector volume counts over time.

mation for major freeways, highways and arterials. The quantity of probes on the road network has a significant impact on the quality of the data collected. The network coverage improves with the number of probes. In situations where real-time data is not accessible, INRIX also offers historical data. The quality of the data improves with increasing device penetration (i.e., more probes). Figure 3.6 presents a snapshot of probe line data in the city of Saint Louis, colored by traffic speed along roads.

This study uses INRIX line data along each mile-long travel segment at a frequency of one minute with geographical location information, timestamp and traffic attributes such as: speed and volume. INRIX uses the following methods to produce historical flow data [91]:

- Traffic sensors - local Department of Transportation (DOT) or private sector businesses install sensors in the road from which traffic speed is either recorded or inferred. The sensors make use of one of several technological platforms: Toll tag readers, Radar sensors, and Embedded induction loop sensors.

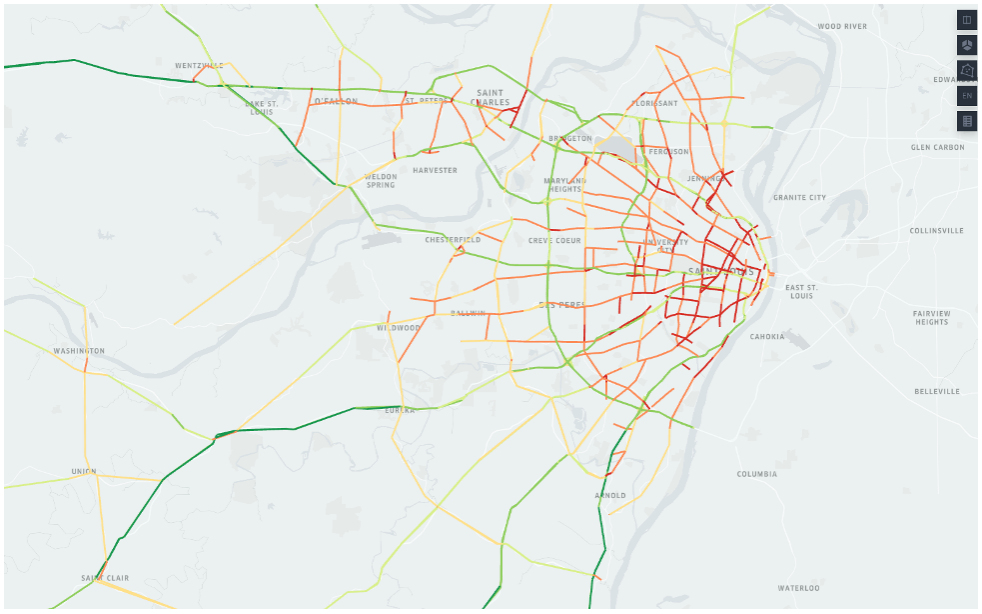



**Figure 3.6:** Overview of probe data in the city of Saint Louis, colored by speed.

- Probe vehicles - Hundreds of thousands of probe vehicles, including trucks, taxis, buses, and passenger cars, are part of the INRIX network and may communicate speed and position data back to a central site. To get the speed and location information discreetly, INRIX has agreements with numerous fleets.
- INRIX Smart Dust Network - This network combines real-time GPS probe data from more than 650,000 commercial vehicles across the United States that travel on a particular section of road during a specific time window, physical sensor information, and other real-time traffic flow information with hundreds of market-specific factors that affect traffic, such as construction and road closures, real-time incidents, sporting and entertainment events, weather forecasts, and schedules. The speed that occurs on that road segment is calculated with a measured level of precision by this component after it collects all input points and weights them properly based on the quality and latency of the input.

A typical INRIX dataset contains the following important information:

- TMC code – spatial unit that INRIX uses to present traffic flow data with each segment defined by a 9-digit TMC code.
- Measurement timestamp – a 19-digit time format used by INRIX to define the year:month:day:hour:minute:second (e.g., 2021-02-09 10:05:34 for September 2nd, 2021, at the 10th hour, 5th minute and 34th second) for each record.
- Speed – representing the segment's historical mean speed for the respective segment in miles per hour
- Travel Time – representing the accumulation of information provided by GPS probes.
- Road order – the index of the segment along its respective road.
- Bearing – the direction or position of the segment relative to a fixed point

- Miles – representing length of the road segment
- Start latitude – the starting point of the road segment defined by a geographic coordinate describing the north-south location of a point on the earth's surface. The angle ranges from –90 degrees at the south pole to 90 degrees at the north pole with 0 degrees at the equator.
- Start longitude - the starting point of the road segment defined by a geographic coordinate describing the east-west location of a point on the earth's surface. The angle ranges from –180 degrees at the west pole to 180 degrees at the east pole with 0 degrees at the prime meridian.
- End latitude - the ending point of the road segment defined by a geographic coordinate describing the north-south location of a point on the earth's surface. The angle ranges from –90 degrees at the south pole to 90 degrees at the north pole with 0 degrees at the equator.
- End longitude - the ending point of the road segment defined by a geographic coordinate describing the east-west location of a point on the earth's surface. The angle ranges from –180 degrees at the west pole to 180 degrees at the east pole with 0 degrees at the prime meridian.
- C value – The confidence value is a scale from 0 to 100 that agencies can use to assess if the INRIX value satisfies their requirements for real-time data.

Table 3.2 presents a row sample of the collected Probe data. As observed, there is a variation in the data types across different attributes/columns such as: characters; integers; floats and strings.

**Table 3.2:** CSV sample (one row) for the collected Probe Data.

| Attribute | Value |
|---|---|
| tmc | 119P14457 |
| road | TOWER GROVE AVE/CENTER CROSS DR |
| direction | NORTHBOUND |
| county | ST. LOUIS (CITY) |
| zip | 63116 |
| start latitude | 38.60442 |
| start longitude | -90.25886 |
| end latitude | 38.60463 |
| end longitude | -90.25882 |
| miles | 0.014646 |
| road order | 2 |
| type | P1.11 |
| bearing | 8.46 |
| dxn | NE |

### 3.3.3 Events Data

Transportation authorities can learn about traffic events through Emergency services Computer-Aided Dispatch (CAD), media reports, and staff monitoring of CCTV feeds of the routes. Organizations may also use crowdsourced information, which can be categorized as passive or active. Mobile devices serving as probes, such as cars and trucks, can collect passive data without the user having to manually enter any information. This information may be gathered either voluntarily, as in the case of location tracking used by most mobile phone navigation applications, or inadvertently, as in the case of Bluetooth sensors placed along a road that identify the unique Media Access Control (MAC) address of discoverable Bluetooth-equipped devices. The following information can be gathered passively: speed, journey times, pavement texture, and weather [92]. In contrast, active data necessitates that individuals willingly and manually report on the state of the roads. This can be as conventional as people calling the TMC directly, but it frequently consists of social media posts about traffic that are geotagged to the incident's location and posted as soon as possible. Popular sources of crowdsourced traffic data on social media include Twitter and Facebook. An event (congestion or incident) occurrence time,

reliability and other attributes such as confidence, location, and streets are captured in a Waze dataset. Waze congestion and accident reported data were assessed by [93] and found reasonable spatial and temporal accuracy. [94] used Waze accident report data and found acceptable reliability of the reported events. [95] used a t-test to prove that travel times from Waze data and the ground truth are almost equal. Figure 3.7 presents a snapshot of events point data in the city of Saint Louis, colored by traffic speed along roads.

This study uses Events data obtained from Waze which is a very popular mobile navigation app used by over 100 million people every month. Users of the Waze app can report traffic jams, weather conditions, queues, collisions, disabled vehicles, and other road-related information using the app. Other Waze users are prompted by the app to confirm an ongoing incident as they pass it while traveling. Users are given increasing experience levels as they submit more reports, which enhances the perceived reliability of their reports. And so, Waze data includes incidents such as crashes, traffic jams, construction, road closures, stalled vehicles, weather events and other road hazards. Waze

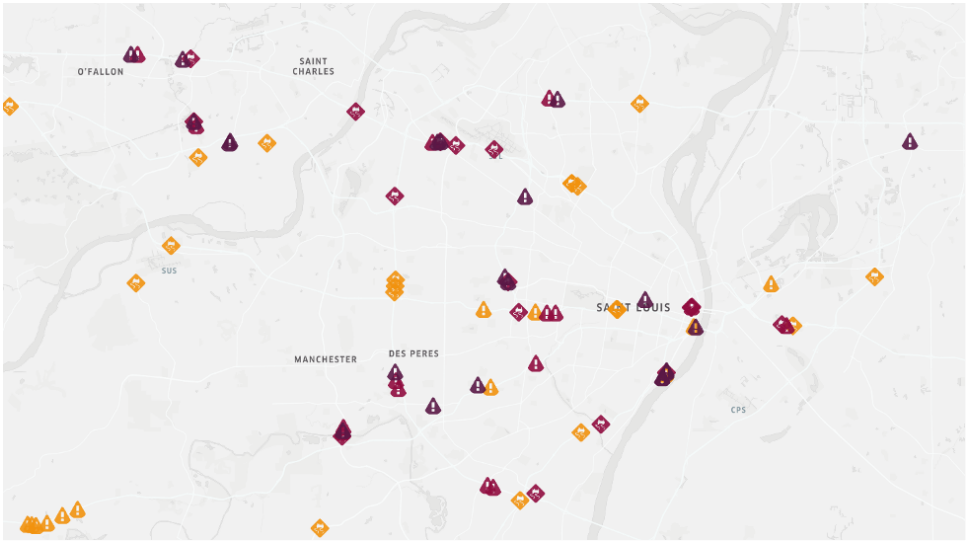



**Figure 3.7:** Overview of Waze data in the city of Saint Louis, colored by type.

offers point data of the event type with geographical location information and timestamp. It contains the following important information:

- Sub Type – the severity of the reported event type with each type having multiple subcategories to represent the magnitude of the event.
- Type – the type of the reported event as reported by the app user
- UUID – unique identification number for the reported event
- Longitude - geographic coordinate describing the east-west location of the reported event on the earth's surface. The angle ranges from –180 degrees at the west pole to 180 degrees at the east pole with 0 degrees at the prime meridian.
- Latitude - geographic coordinate describing the north-south location of the reported event on the earth's surface. The angle ranges from –90 degrees at the south pole to 90 degrees at the north pole with 0 degrees at the equator.
- Time Stamp - a 19-digit time format used by WAZE to define the year:month:day:hour:minute:second (e.g., 2021-02-09 10:05:34 for September 2nd, 2021, at the 10th hour, 5th minute and 34th second) for each record.

Table 3.3 presents a row sample of the collected Waze data. As observed, there is a variation in the data types across different attributes/columns such as: characters; integers; timestamp; floats and strings.

## 3.4 Data Conflation

Conflation enables us to fuse the different datasets into one table, allowing for multi-dimensional analysis of the datasets. The conflation process is illustrated in Figure 3.8.

Both Waze and CV datasets were conflated to roads for which probe data was available for at least every 1 minute. First, a multi-line string geometry is generated using the

**Table 3.3:** CSV sample (one row) for the collected Waze Data.

| Attribute | Value |
|---|---|
| country | US |
| n thumbs up | 1 |
| city | Saint Louis |
| report rating | 0 |
| confidence | 0 |
| reliability | 6 |
| type | ACCIDENT |
| uuid | 0c85168e-ce9b-493d-be25-0f2fba901c82 |
| road type | 7 |
| magvar | 91 |
| subtype | ACCIDENT MAJOR |
| street | MO-38 |
| report description | 911-reported accident |
| longitude | -92.979457 |
| latitude | 37.355442 |
| pub millis | 2022-04-27 16:49:26.000000 |
| request millis | 2022-04-27 16:51:11.553082 |
| county | WEBSTER |
| event class | ACCIDENT MAJOR |
| req date | 2022-04-27 00:00:00.000000 |
| start time | 2022-04-27 16:51:11.553082 |
| end time | 2022-04-27 18:07:43.106747 |
| duration | 76.5258944166667 |

start and end coordinates for each probe segment. Point geometries are also generated for connected vehicles and Waze data using their respective coordinates. A 12 feet buffer is generated around each line string to create a polygon layer for spatial joining with the point geometry layers. Each point is then mapped to the corresponding (closest) line segment if contained within its buffer. The 12 feet buffer distance was chosen based on

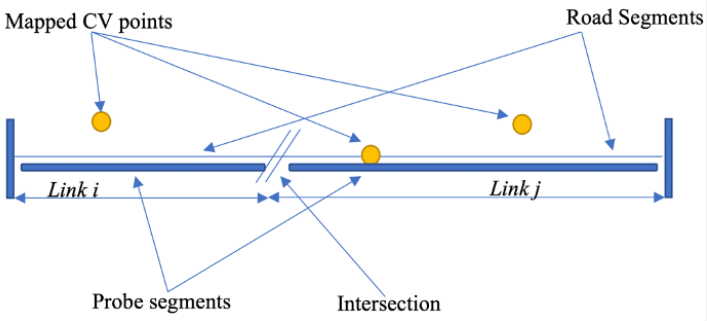



**Figure 3.8:** Road Segments with mapped CV points and Probe segments

the standard US lane width. To account for direction of travel, a direction column is created for each dataset based on heading and bearing information. This resulted in four 5 categories of travel directions: Northeast (NE), Southeast (SE), Southwest (SW) and Northwest (NW). The direction column is subsequently used to refine the initial proximity mappings. The process is repeated for all other point geometry datasets. The final step in the conflation process is to match the timestamps of the 8 different datasets. In this study, a 1 – minute aggregation window is used for temporal mapping. Each row contains information from all three datasets where available. A manual verification of a subset of the integrated data showed a conflation accuracy of about 97 percent. Figure 3.9 summarizes the key elements of a conflated dataset. To ensure that the fused datasets are compared along the same routes, we adopted the road segmentation scheme provided by the probe data vendor. About 80 percent of all road segments used in this study were between 0.1 to 0.5 miles. Figure 3.10 presents the distribution of road segments for a subset of Probe data. The ratio of the connected vehicle speed divided by the length of the conflated probe segment length is used to estimate the travel time. Where there are multiple CVs on a probe segment, the average speed of all vehicles is used.

## 3.5 Multiscale Data Analysis

Multiscale Data Analysis (MDA) is used to extract and compare trends at different frequency. The Wavelet transform is used to implement MDA in the current study. Wavelets use mathematical functions to segment data into distinct frequency components and then investigate each component with a resolution proportional to its scale. They outperform traditional Fourier methods in analyzing physical situations characterized by discontinuities and sharp spikes. Detailed discussion of wavelet decomposition can be found in [78]. Wavelets are a class of time and frequency localized basis functions that can be

expressed as:

$$\Psi_{su}(t) = \frac{1}{\sqrt{s}}\Psi(\frac{t-u}{s}) \tag{3.1}$$

| Probe Data | | Connected Vehicles Data | | Waze Data | |
|---|---|---|---|---|---|
| tmc_code | 119+19153 | Journey_id | 750ad6c439516a5b9c.. | Sub_type | Accident_Major |
| Measurement_tstamp | 2021-02-09 10:00:00 | Captured_timestamp | 2021-02-09 10:05:34 | Type | Accident |
| speed | 9.0 | latitude | 38.626501 | | |
| travel_time_minutes | 0.99 | | | uuid | 5ba72319… |
| Road_order | 4.0 | longitude | -90.198566 | | |
| Road | N 18th ST | | | | |
| County | St. Louis (city) | speed | 7.0 | Longitude | -94.4989 |
| Zip | 63102 | | | | |
| Start_latitude | 38.63141 | heading | 17.0 | | |
| Start_longitude | -90.20572 | | | | |
| End_latitude | 38.63347 | Ignition_status | MID_JOURNEY | Latitude | 39.1314 |
| End_longitude | -90.20492 | Event_type | JOURNEY | | |
| Miles | 0.15 | Acceleration_type | Hard Braking | | |
| Road_order | 4.0 | Journey_event_type | END | Time_stamp | 2021-02-07 00:02:37 |
| Bearing | 25 | Postal_code | 63102 | | |

**Figure 3.9:** Sample table of conflated datasets

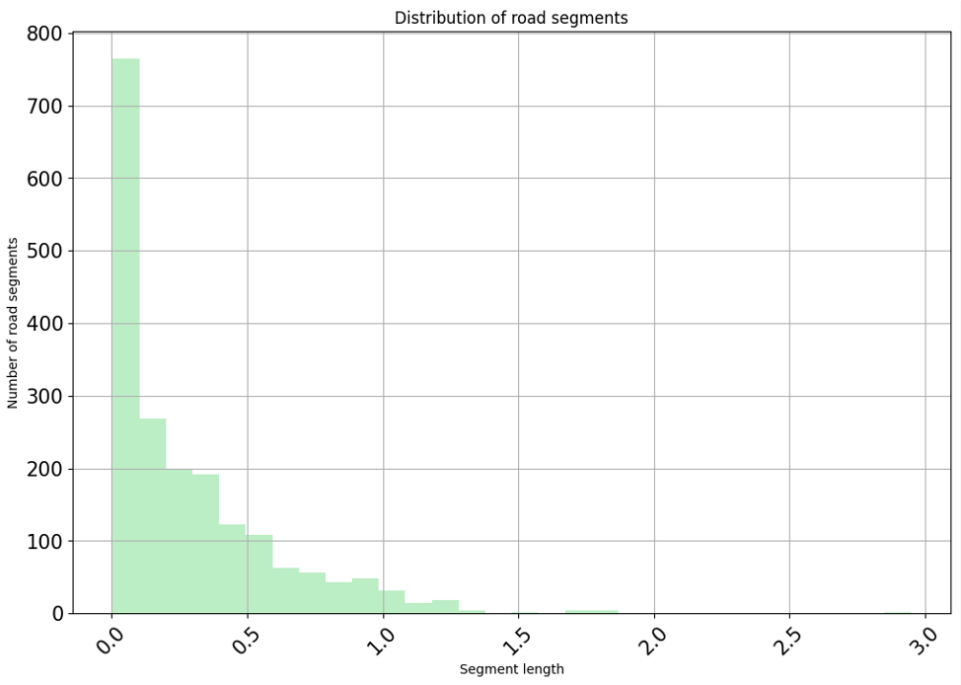



**Figure 3.10:** Histogram plot of count for road segment lengths

where $s$ and $u$ denote the parameters for dilation and translation, respectively. The central wavelet ($t$) is time and frequency localized, occupying an equal area above and below the time-axis. The wavelet dilation and translation parameters are usually discretized dyadically for most practical applications to measure data as: $s = 2^m, u = 2^m k$.

Where $m$ and $k$ are integers indicating the dilation and translation parameters, respectively. The resulting wavelet family is denoted by:

$$\Psi_{mk}(t) = 2^{-m/2}\Psi(2^{-m})(t-k) \tag{3.2}$$

The translation parameter specifies the wavelet's location in the time domain, whereas the dilation parameter specifies its location in the frequency domain, taking into account the extent of the time–frequency localization. The wavelet equation represented in Equation 3.2 may be designed to be orthonormal to one another and to have varying degrees of smoothness.

### 3.5.1 Short-Term, Medium Term and Long-Term Speed Variation

Figure 3.11 shows results of wavelet decomposition of both CV and probe data. Three main trends are apparent: Short-term, Medium-term and Long-term trends. In the current study, short-term trends capture five - 15-minute variations in the original dataset. In Figure 10, mode decomposition 1 (mode 1) illustrates the short-term trends for CV and probe data collected over a 15- to 30-minute period on Freeways and arterials. Medium-term trends capture hourly variations (1–3 hours) and the average daily trend. They contain critical information about the hourly peak and off-peak periods. These are illustrated by the mode 6 decomposition. The final wavelet batch of variations represents observed daily trends in the datasets. As illustrated in Figure 10, specifically the general trend, there is a strong correlation between the two datasets.

### 3.5.2 Connected Vehicles vs Probe Data – Speed Bias, Congestions and Incidents

The key performance measures used to compare the different datasets are Latency and Speed Bias. Latency –used to calculate the accuracy of CV and probe data for congestion and incident detection. In the current study, the term "Latency" refers to the difference between the recorded (actual) start and clearance time of an event and the corresponding times captured by the CV or probe data. Speed Bias – Is the absolute difference in speed between the different datasets at different times of the data. The average speed bias is reported as the mean absolute difference between CV and probe speed for all road types. Three main experiments were conducted to evaluate the opportunities and challenges with using connected vehicle data as compared to traditional probe datasets. The experiments which are discussed as follows include: a comparison of the bias in speed estimates from both data sources, as well as the congestion detection and incident detection accuracy.

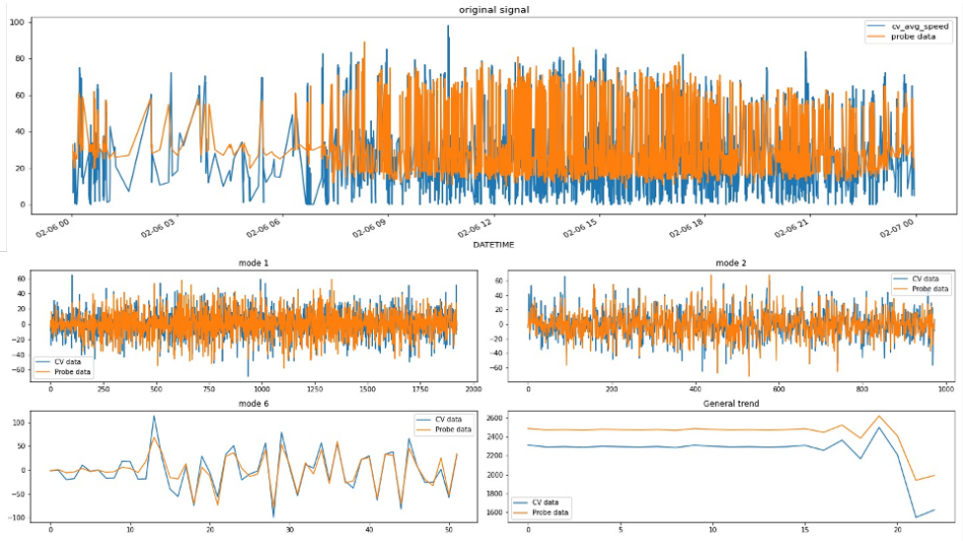



**Figure 3.11:** Short, Medium, and Long-Term trends extracted from CV and Probe data using Wavelet decomposition.

### 3.5.3 Speed Bias Comparison

There is almost always a bias in speed estimates from both datasets as shown in the first row of Figure 3.12. About 95 percent of the time the speed bias stays within a range of 0 to 20 mph. The bias, however, varies by location, road type, and time of day. As shown in second row of Figure 3.12, the differences in speed are much smaller on freeways as compared to arterials. It is also worth noting that CV data generally shows high speed variability on both freeways and arterials as compared to probe data. The low penetration rate of probe data on local routes and arterials could be the reason for this trend.

Lastly, in Table 3.4 and the heatmap in Figure 3.13, we compare the differences in speed during peak and off-peak hours. Although the differences in speed bias are not significantly high, there is an observable reduction in speed bias (between 2 and 5 mph) during peak vs non-peak hours. The heatmap confirms this trend and shows the variability in speed bias across different locations.

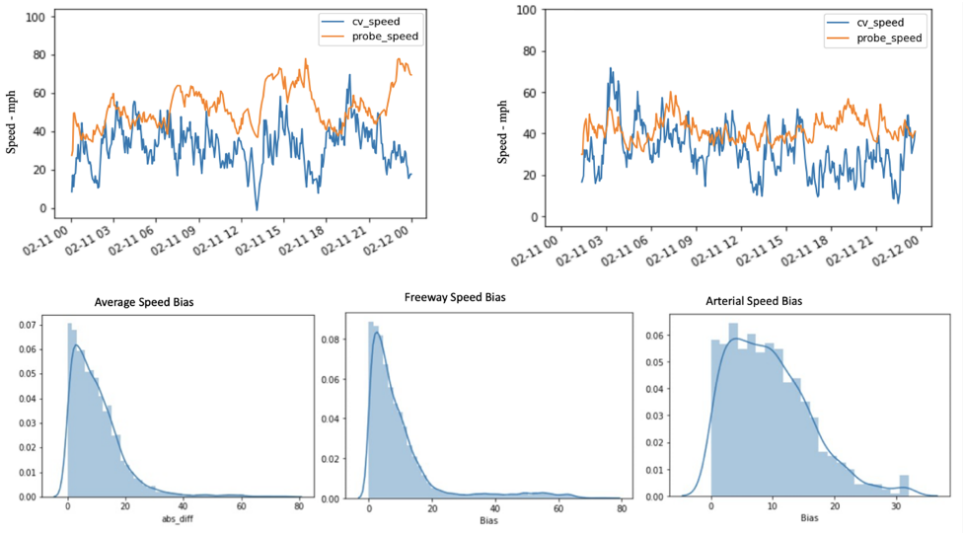



**Figure 3.12:** Speed variations of CV data and Probe data across datetime: first row – superimposed plot of CV and probe speeds for road segment over time. second row - PDF plot of mean absolute difference between CV speed and probe speed for all road types, freeways

**Table 3.4:** Absolute Mean Difference between CV and probe speeds on Freeways and Arterials

| | Freeway (mph) | Arterial (mph) |
|---|---|---|
| AM peak hours | 10.51 | 9.30 |
| PM peak hours | 9.68 | 9.00 |
| off-Peak hours | 9.70 | 9.61 |

While Waze data is our source of incident and congestion events, there exists some limitations using such data where it is unable to provide information on traffic incidents that do not fall under one of its predefined event categories [93]. Additionally, Waze was made specifically for use in private vehicles; so, it does not offer information about public transportation. Waze's data are also restricted to registered users, making it challenging for the public sector (such as traffic management organizations) to access them. Such limited access limits the exposure of non-users to the app and hence limits the amounts or accuracy of reported events.

### 3.5.4 Congestion Detection

This experiment evaluates the ability to detect congestion events with probe and connected vehicle data for a total of 28 congestion events. Two main levels of congestion are


**Figure 3.13:** Heatmap of speed bias by road and hour of day

investigated based on classifications from Waze incident data feed: 1) Jam, Stand-Still Traffic, 2) Jam, Heavy Traffic and 3) Jam, Moderate Traffic. The congestion detection and clearance time latency is used as a measure of performance for comparing the accuracy of the different datasets for congestion detection. As shown in Figure 3.14, both datasets can detect all three types of incidents identified in this study. CV data were slightly better at detecting short duration, jam, stand-still incidents. It is also observed that for the jam-stand-still-traffic condition, the CV data detected the freeway congestion about 3 - minutes on average prior to the probe data. Similar trends were observed on both freeways and arterials.

### 3.5.5 Incident Detection

The final experiment evaluates the ability to detect different types of incidents using probe and connected vehicle data for a total of 10 incidents. Three main types of incidents (based on Waze feed classifications) are used in this study: 1) Major Accident, 2) Minor Accident, 3) Road Construction. Other incidents such as stalled vehicles, weather events and road closures were not included due to low impact on traffic flow during the analysis period. Like the congestion detection case, the incident detection and clearance time latency is used as a measure of performance for incident detection. As shown in Figure 3.15, both datasets can detect most major accidents and road construction events

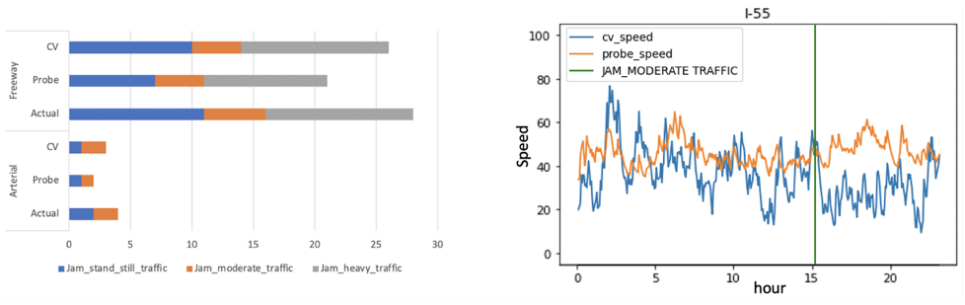



**Figure 3.14:** Probe and CV data congestion detection rate comparison on freeways and arterials (left), probe and CV speed changes during a congestion event (right)

accurately. For minor accidents, it is observed that while CV data had 100 percent detection rate, probe data could only detect about 20 percent. The incident detection and clearance time latencies were also significantly lower for CVs: 5 – 8 minutes faster than probe data.

## 3.6 Summary

The first purpose of this chapter was to evaluate the opportunities and challenges for using CV data to estimate travel times, detect congestion and incidents. Data was integrated with traditional traffic data sources for comparative analysis purposes. This is the first study that integrates actual CV data (not simulated) with conventional data streams including WAZE and probe data. The study made use of wavelet decomposition to observe the short, medium and long-term trends observed between the two datasets. The study performed three main analyses to compare the bias in speed estimates from the CV and probe data, as well as their congestion and incident detection accuracies. For most of the time, CV data outperforms probe data in terms of incident and congestion detection on both freeways and arterials due to its microscopic nature. Although the probe data appears to perform better than the CV in some instances, the penetration rate of the CV was low in those instances. Overall, CV dataset performs much better than the probe

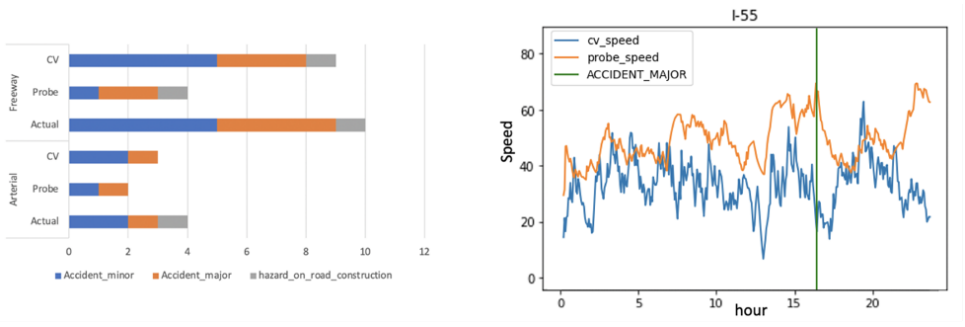



**Figure 3.15:** Probe and CV data incident detection rate comparison on Freeways and Arterials (left), probe and CV speed changes during an incident event (right)

data in congestion and incident detection and thus, offers the possibility to be used not only in analyzing historical traffic patterns but also in performing predictions. To examine its reliability in future predictions, we leveraged Deep learning models to perform a multi-step traffic forecasting model on CV data and relevant influencing datasets such as weather and events. The next chapter introduces the methodology, results and analysis of the developed models.

# Chapter 4

# Multi-Purpose, Multi-Step Deep Learning Framework for Network-Level Traffic Flow Prediction

## 4.1 Introduction

Traffic congestion costs cities billions of dollars every year when factors such as accidents, pollution and delays are factored in. According to a recent report published by the Texas Transportation Institute all 494 metropolitan areas in the United States experienced 8.7 billion vehicle-hours of delay in 2019; resulting in 3.5 billion gallons of wasted fuel and 190 billion dollars in lost productivity, or about 0.15 percent of the nation's GDP. These costs drive the need for a data-driven strategy to solve these issues. When traffic demand approaches or exceeds the traffic system's available capacity, traffic congestion occurs. Many studies [18–20] have shown that traffic datasets can be used to predict traffic congestion, allowing drivers to avoid congested areas (e.g., through traffic flow forecasting navigation systems), policymakers to decide on changes to traffic regulations (e.g., re-

placing a normal lane with a toll lane), urban planners to design better pathways (e.g., adding or removing a road lane), and transportation engineers to better plan for the timing of construction activities. Traffic forecasting is a critical component of advanced traffic management systems that can help transportation planners in planning for volatile events ahead, by taking early actions and arrangements, which contributes to better traffic management and service quality. It may not only serve as a valuable reference for increasing the efficiency of limited traffic management resources, but it can also assist passengers in deciding ahead of time to minimize traffic congestion. Longterm projections are more likely than short-term forecasts to reduce travelers' average trip time [14]. Common forecasted traffic parameters include traffic flow [15], traffic speed [16], and traffic time [17]. The increasing availability of large-scale traffic data, which can be looked at from a temporal and spatial lens, has paved the way to develop prediction models that are robust to capture the underlying driving mechanism of traffic volatilities, especially the random (unforeseen) components. Temporally, majority of prior studies have focused on single-step traffic flow forecast for a single road section with a time interval of less than 30 minutes. For some applications in ITS, such as traffic planning, it can be insufficient.

Another issue is the increased frequency of collected (input) data which allowed the value of long-time horizon predictions to supersede shorter term. As a result, multi-step traffic flow prediction is gaining popularity. Multi-step traffic flow prediction uses the same methodologies as single-step traffic flow prediction; however, the prediction performance rapidly degrades as the number of steps grows. Developing a practical multi-step prediction model is, thus, more important than a single-step prediction task because it provides valuable insights over longer time horizons which allows for better positioning of traffic management strategies. In addition, many studies only focused on the spatial component by predicting traffic on a single-route or a specific connection or crossing. The development of an ITS demands the need to explore multi-route predictions on a larger

scale by considering the complex spatial dynamics of a network [21]. While prior knowledge of the distance or travel time between regions can aid in capturing spatial correlation, there are still some hidden time-varying traffic patterns that data-driven methods must uncover. The challenge is resolving the intricate spatiotemporal dependencies, which refer to traffic information (e.g., speed or volume) at a certain location in space and moment in time. With the emergence of deep learning models, this research aims to solve the question of how to construct appropriate deep learning models to cope with large-scale complex network-wide traffic data. Large-scale network traffic prediction demands an intelligent and efficient prediction methodology to forecast traffic on longer horizons and reflect the flow propagation. Numerous variables affect a region's future traffic state, including historical observations of traffic, correlation with other regions, and external factors (holidays and special events). The technique used to fuse muti-purpose variables such as traffic speed and volume is a challenge for the current generation of prediction models. The interrelationships between regions are intricate and complex which adds to the challenges in developing a prediction model. As a result, more research into how to create an accurate and reliable network-wide (by exploring multi-routes), multi-purpose (such as speed and volume), multi-step (longer prediction horizon) prediction model is required.

Reliability of the estimates obtained from the developed models is another issue since it greatly depends on the data source. Data used for traffic forecasting has two main issues: availability, size of data, and the overreliance on probe data. When qualified traffic data is unavailable, the trained model's performance degrades since performance correlates with the quality of input data. While we can collect more traffic data due to transportation infrastructure modernization, the data is frequently of poor quality, with noise and critical features missing. Currently, the amount of qualified traffic data available for analysis is insufficient. To our knowledge, most prior studies [72, 73] used probe traffic

data that was less than a year old and, in some cases, as recent as one or two months [74]. Probe data cannot capture the live travel time or volume on road segments and using it for traffic forecasting is likely to yield unreliable estimates. Therefore, there is a need to use a more reliable data source that can provide microscopic live travel information to improve the reliability of traffic predictions along road segments. The projected growth of CV will provide an alternative way of collecting real-time data for traffic forecasting. The future of ITS is shifting towards big real-time data from CV as automobile makers rush to incorporate CV technology in novel and current vehicles for numerous apparent advantages, which include vehicle autonomy and navigation, vehicle sensor and driver monitoring, live over-the-air updates, advanced road warnings, and improved battery and fuel efficiency. Government and state institutions that create, maintain and manage road infrastructure may take advantage of the CV data available to know what is happening on the road and make informed decisions on traffic flow and road pavement infrastructure. Thus, it is critical to effectively process all CV data on a state level for statewide transportation infrastructure management. This study's CV data is from wego technologies. The data was collected and transmitted every 3 seconds. The study estimated travel times on arterials and freeways by analyzing data from connected vehicles, including the vehicle's speed, acceleration, GPS location, and "brake press". Additionally, the current chapter advances the state-of-the-art by developing a traffic forecasting model using UNet architecture. Figure 4.1 presents the framework for the network-wide predictions using the image outputs from each phase. The significant contributions of this chapter are summarized below:

- Propose a pipeline for processing and learning from large-scale spatiotemporal data by leveraging distributed GPU clusters.
- Propose a data fusion technique that enables state-of-the-art Machine Learning (ML) models to learn from multisource data, by leveraging GPU computing through

Nvidia Rapids and Dask framework.

- Design a DL framework for simultaneous, pixel-level, dense prediction of traffic flow variables (speed and volume) while considering the network traffic temporal evolutions and spatial dependencies using a UNet model that learns traffic data through 3-dimensional matrices.

## 4.2 Related Work

Developing an ITS is a promising solution to provide more accurate travel information based on future predictions to transportation users and developers. The techniques used to predict traffic across the literature are summarized into the following categories: statistical, light machine learning and deep learning. Statistical mainly uses time series analysis models such as historical and moving averages which can be helpful with short-term predictions on static data. Standard machine learning models include Artificial Neural Network (ANN), Support Vector Machine (SVM) and KNN, which are generally better performing than statistical models because of their architecture's capability in capturing more features. However, extracting the complex and dynamic patterns in the spatio-temporal dynamic traffic data adds to the model limitations. The rise in faster Graphics Processing Units (GPU) paved the way for the increased use of deep learning models to perform predictions. Due to their superior ability to capture complex traffic patterns, var-

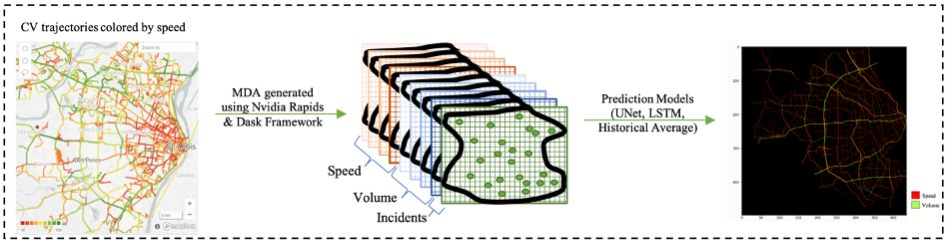



**Figure 4.1:** Framework for network wide traffic predictions

ious deep learning-based methods for traffic prediction have recently been successfully applied to traffic forecasting. Table 4.3 presents a sample of the recent use of deep learning models for traffic predictions. The table presents the authors, used prediction model, predicting variables, road-type, prediction horizon and results.

Short-term traffic forecasts restrict many existing approaches, and there are few successful methods existing for predicting long-term traffic status. Short-term is also referred to as single step. We define short term predictions as any predictions that fall in the range of 5 to 15 minutes into the future. Medium-term predictions are 15 minutes to one hour, and long-term predictions are beyond one hour. Long-term or multi-step forecasting is more difficult than short-term prediction because of the sensitivity of error propagation [96]. Real-time traffic control is where short-term forecasting is most useful. Long-term forecasting that is accurate and timely may assist managers in making early judgments, actions, and overall arrangements, which can help improve traffic management and service quality.

It can not only serve as a valuable reference for increasing the efficiency of limited traffic management resources, but it can also assist passengers in planning ahead of time to avoid traffic congestion [16]. Sequence-to-sequence (Seq2Seq) is a frequently used technique in multi-step forecasting [66, 97, 98]. It is also common to capture temporal dependency using RNNs and temporal convolutional networks (TCNs) [76–78]. LSTM is a widespread technique that Chen et al. (2021) and Cui et al. (2018) used in their studies to predict the short-term changes in traffic flow and speed, respectively. Several authors used graph neural networks by focusing attention on different space and time features. Yu (2021) performed short-, medium- and long-term predictions of traffic speed on urban roads while other authors (Li et al. (2021), Yin et al. (2021), Zhao et al. (2020)) used freeway segments. It is worth noting that Zhao et al. (2020) attention of temporal changes in their model performed much better than Li et al. (2021) on the same road type

**Table 4.1:** Comparison of recent use of deep learning models for traffic predictions

| Author | Model | Predicting | Road Type | Horizon | Results |
|---|---|---|---|---|---|
| CHEN ET AL. (2021) | LSTM + Ensemble Empirical Model Decomposition (EEMD) | Traffic flow | Freeway | Short (5-15min) | RMSE:0.79 |
| WU ET AL. (2018) | CNN + Recurrent Neural Network (RNN) | Traffic flow<br>Traffic flow | Freeway<br>Freeway | Short (5-15min)<br>medium (15min - 1hr) | RMSE: 15min = 32.16, 30min = 34.29, 45min = 36.08 |
| YAO ET AL. (2019) | CNN + LSTM | Traffic flow and volume | Network-wide | Medium (15min - 1hr) | RMSE: 24.10 |
| MA ET AL. (2015) | LSTM Neural Network (LSTM NN) | Traffic Speed | Freeway | Short (5-15min) | MAPE: 5min = 3.78<br>10min = 3.78<br>15min = 3.78 |
| LI ET AL. (2021) | Graph Convolution Network (GCN) | Traffic flow | Freeway | Short (5-15min)<br>medium (15min - 1hr) | RMSE: 15min = 32.17<br>30min = 32.96,<br>45min = 33.68,<br>60min = 34.53 |
| YU (2021) | Generative Adversarial Graph Attention Network | Traffic Speed | Urban | Short (5-15min)<br>medium (15min - 1hr)<br>and long (1 - 4hr) | MAPE: Short: 6.1,<br>Medium: 8.3,<br>Long: 12.6 |
| YIN ET AL. (2021) | Multi-stage Attention Spatial-Temporal Graph Network (MASTGN) | Traffic flow and Speed | Freeway | Short (5-15min) | RMSE: 17.73 |
| ZHAO ET AL. (2020) | Temporal Graph Convolutional Network (TGCN) | Traffic Speed | Freeway | Short (5-15min)<br>medium (15min - 1hr) | RMSE: 15min = 4.53,<br>30min = 5.01,<br>45min = 5.35,<br>60min = 5.64, |
| CUI ET AL. (2018) | Deep stacked bidirectional and unidirectional LSTM | Traffic Speed | Network-wide | Short (5-15min) | MAPE: 5.6 |
| CHOI (2020) | UNet | Traffic Speed and Volume | Network-wide | Short (5-15min)<br>medium (15min - 1hr) | MSE: 0.0016 |

and prediction horizon. Traffic prediction algorithms can be divided into two categories: single and multi-purpose techniques.

The single-purpose technique is focused on modeling traffic condition using one variable (such as speed, flow, or occupancy), whereas the multi-purpose approach is based on constructing a model that considers more than one variable. These models, unlike single-purpose models, are capable of capturing travel characteristics from multiple dimensions of a transportation network over time. Multilinear regression models were used by [99] to forecast bus arrival time using multi-purpose attributes such as: distance, number of passengers at stops, stop numbers, and weather conditions. The performance of regression models will degrade as the dimension of the data rises, because the attributes in transportation services are frequently not independent but connected with one another. Complex interactions and noisy data demand the use of machine learning

algorithms. Other authors [100] proposed a data clustering and genetic programming technique for predicting highway trip time. Two of the most extensively used machine learning models in multi-purpose bus travel time prediction are ANN, and SVM, [101]. Kalman Filtering models, which use both historical and real-time data, have been widely used to estimate bus arrival times [22, 102, 103]. Previous research in this field has mostly focused on constructing models for anticipating delay as a self-contained single-purpose prediction process.

Numerous GNNs were used in literature, to extract spatial dependency from traffic networks [21, 104–106]. Chen et al. (2021) filtered freeway segments from the traffic network and achieved good results. The entire network was used by Cui et al. (2018). However, most models lacked the use of a network-wide dataset and longer-term predictions. Image segmentation and classification has been widely successful using UNet [107]. U-Net is a CNN based on a fully convolutional neural network where its architecture is altered and expanded to work with fewer training images to obtain significantly precise segmentation results. Choi (2020) achieved strong results using a UNet model for predicting traffic speed and volume for multiple routes in the short and medium term. Although GCN-based techniques may learn more hidden aspects of traffic networks than CNNs, they are ineffective at capturing dynamic spatial traffic dependency. The term "multi-routes" presents a challenge since the relationship between two static places can change over time. For example, during morning and evening peak hours, the spatial links between residential and commercial districts are more important than at other times. Since most previously published works fall short of accurately maintaining spatial information while simultaneously making good predictions [108, 109], this study seeks to adopt an approach that sought to maintain spatial information. Numerous publications [106**?** ] have described the development of an adaptive matrix for data-driven spatial correlation discovery using spatial correlation data.

When the data is insufficient or noisy, the efficiency of data-driven methods is limited, and accurate prior knowledge may help the model perform better in these situations. Most studies focus exclusively on predefined correlations or data-driven correlations for prediction. Also, most prediction models suffer from information dilution, observed in other multi-step prediction models [66, 110]. The original data from each input step has been diluted several times by both the encoder and decoder cells before reaching a specific output step in the sequence. When there is sufficient data, the dilution effect can be mitigated; however, insufficient data can exaggerate the effect, resulting in decreased prediction performance. To eliminate the issues mentioned earlier, we use connected vehicle data in this study for traffic forecasting. Each experimental traffic feature (for example, traffic speed and flow) has both spatial and temporal attributes (i.e., its observation location and time).

Generally, studies [111, 112] extracted spatiotemporal patterns solely from traffic features without fully exploiting those traffic features' spatiotemporal attributes. By providing additional information, these attributes, on the other hand, can directly aid the model in identifying spatiotemporal correlations between traffic states. Apart from that, they can augment existing spatiotemporal information when sufficient feature data is unavailable. Furthermore, versatile and extendable transportation data integration frameworks are critical for modern transportation analysis and management. Data Fusion is the challenge of merging data from several sources and giving consumers a consistent representation of that data [113]. Data integration system design is a critical step in a wide range of real-world applications, particularly in ITS. Other common challenges in traffic prediction, such as planning issues and traffic estimation, are similarly involved with multi-source fusion [114]. In both research and practice, transportation data integration frameworks and tools have been devised and deployed for a variety of applications. To address the challenges mentioned earlier and limitations, we employed a large-scale GPU cluster-

based data processing framework to fuse large-scale datasets and then leveraged the UNet architecture for multi-step forecasting by combining the volatile traffic features on a network-wide level to augment the spatiotemporal information contained in the model input.

## 4.3 Problem Formulation and Overview

Many existing studies ignore the use of large-scale data to develop traffic prediction models and thus disregard the complex topological structure of road networks and temporal patterns by using a single-route, single-step and single-purpose predictions. Such approaches are motivated by faster computations and reporting with high accuracies. However, in practice, the applications of such techniques are very limited since it only captures the instantaneous and steady-state interactions among traffic variables, therefore, a multi-step, multi-purpose traffic prediction framework for multiple routes should be developed.

Specifically, let $x_i = (T * W * H * C)$ represent the input data tensor for a specific day from the training data ($i$) with $C$ number of channels or purpose, ($W * H$) is the 2D array width and height and $T$ is the time bins per day with each time step aggregated by five-minute intervals. The goal is to predict $x_p$ on test data ($p$) using only one hour from the test dataset ($T + 1, T + 2, ..., T + 11$) to predict the remaining hours in the same day ($T + 12, T + 13, ..., T + 287$). Compared to existing approaches where $C$ is usually one (i.e, 3D tensor instead of 4D) and predictions are usually short or medium-termed

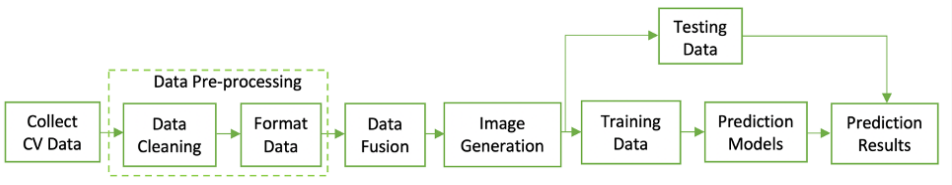



**Figure 4.2:** Framework of the proposed methodology

(i.e, $T+12, T+13, ..., T+23$), the current framework proposed in this study addresses the multi-purpose, multi-step large-scale traffic forecasting challenge at a network level. Figure 4.2 presents the framework of the proposed methodology. Firstly, CV data was collected for one month in Saint Louis County. The data provides attributes such as the vehicle's location, heading, speed, volume and flow, etc. This work uses historical traffic speed, volume, and incidents to forecast future speed and volume. The data then goes through a pre-processing stage to make it feasible in our prediction models. Data cleaning is then performed to clean the data from missing values and anomalies. Cleaned data is then formatted by grouping different headings and time bins together. Data fusion is then performed on the different datasets along with the same spatial and temporal bins. MDAs or images are then generated to efficiently leverage smaller size compacted data layers as an input to the proposed prediction model. In the prediction model, UNet is used as our CNN as it is designed to learn from the MDA matrices and make predictions. The accuracy and robustness of the UNet model are compared to the conventional Convolutional Long Short-Term Memory (ConvLSTM) model and a statistical historical average model. The following section discusses data fusion and MDA generation in detail.

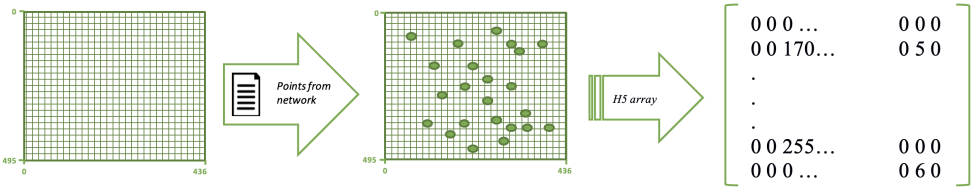



**Figure 4.3:** Spatial bins created for a consistent scaling of H5 arrays

## 4.4 Input Data Structuring

### 4.4.1 Multi-Dimensional Arrays (MDA)

Thirty-one unique dates (days) of CV data are used as input data. Twelve separations are exported from hourly CSV files since 12 (five-minute) time bins per hour (60 minutes per hour/5-minute bins). Each separate file then goes through the splitting channels process where separations are made for the unique columns/channels: incidents, speed and volume. The direction column is estimated and added based on the bearing information provided by the speed and volume channels. Four main heading quadrants are used in the estimation: NE, SE, SW, NW. However, the incidents channel did not provide a bearing column, so we could not split its directions. To create an MDA for each CSV exported in the previous step, we first created an empty raster with latitude and longitude coordinates for the study area, scaled at a height and width of 495 and 436, accordingly. The coordinates used in spatial bins are fixed throughout all MDAs to ensure that they are all developed at the same scale, as presented in Figure 4.3. We use the mean of values within a temporal (frame) and spatial (bin) for the speed channel to get the average speed. We use the sum of values within a temporal (frame) and spatial (bin) for each volume and incident channel to get the total count. Each array created at this step has the shape [495*436].

### 4.4.2 Incidents and Weather Events

We also investigated additional data that may help us make more accurate forecasts such as using an additional input channel (incidents and weather events) for forecasting the main channels (speed and volume). Additional channels or traffic variables could provide useful information about a particular place and aid in the development of a more realistic model. To embed such channels in our input data, we explored three different experiments to make sure that the features from event data can be learned by the model:

- Experiment 1: Events data is binned in the same way speed and volume data were binned.
- Experiment 2: The binned data points from experiment 1 are extended along the respective event duration.
- Experiment 3: The binned points from experiment 3 are scaled by a factor to magnify the impact of the event taking place.

Figure 4.4 presents a snapshot of the three experiments explored over time with the variation of binned data points.

MDA for speed along with four directions, volume along with four directions, incident and weather events are then stacked together to form a stacked array of the shape [495*436*10]. The channels are stacked along the third axis/dimension. Time bins along each hour are then stacked together along all channels to form another stacked array of the shape [12*495*436*10]. Time bins are stacked along the fourth axis/dimension. Each

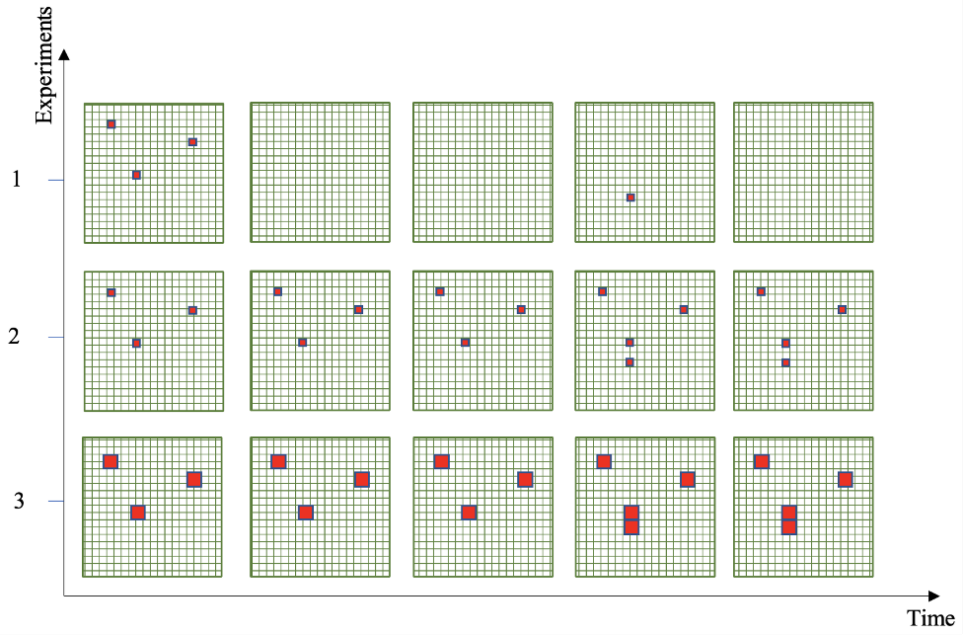



**Figure 4.4:** Experiments with static events channel over time

day, hourly bins are stacked together along stacked time bins and channels to form a further stacked array of the shape [288*495*436*10]. Hourly bins are stacked along the fourth axis/dimension. This process is performed for each unique date, so at the end, we can have an array of shapes [288*495*436*10] for each day. Figure 4.5 presents a visualized shape of the temporal aggregation of channels and bins of an MDA.

### 4.4.3 Processing Pipeline

To accelerate the processing of big CV data in our study, we use Nvidia Rapids and Dask framework. Nvidia Rapids is an open-source suite of software libraries for end-to-end data science and analytics pipelines on GPUs. Rapids is built on top of Nvidia CUDA for accelerated computing and Apache Arrow for GPU in-memory computing, see Figure 4.6(a) and includes several libraries across the data science toolchain. Rapids is the GPU implementation of conventional data science libraries and natively scales from workstations to clusters to cloud systems with the help of Dask libraries. A comparison of Rapids library with popular data science libraries is shown in Figure 4.6(b).

Dask natively scales Python data frames (CPU and GPU) across several nodes and partitions. Dask also offers advanced parallelism and data processing pipelines that enable large-scale analytics by using a directed acyclic graph (DAG) lazy execution framework, which ensures that computational work is scheduled, rebalanced, and optimized before the data is needed. This allows for fast prototyping and experimenting even on

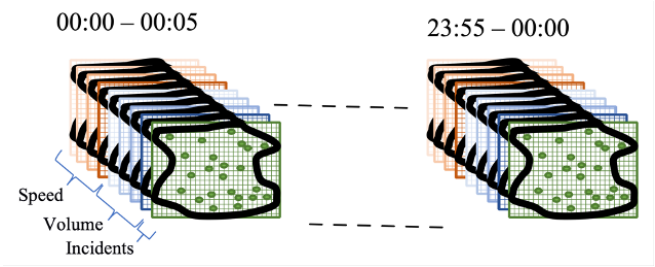
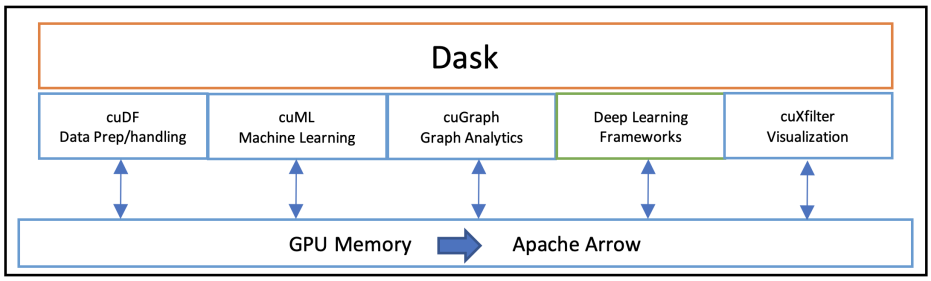



**Figure 4.5:** Temporal aggregation of MDA per day

massive cluster systems. Dask integration with Rapids allows for large-scale GPU cluster-based data processing.

The experimental setup for the project was on AWS GPU virtual machines with Intel Xeon Platinum 8259CL 48 core vCPUs @ 2.50GHz, 192 GB of RAM and 4xT4 GPUs with 16 GB vRAM each. The virtual machines were running AWS optimized Ubuntu 18.04 LTS operating system software. The software stack installed included CUDA 10.2 with driver version 440.33.01. Additional software includes Docker CE v18.03.1-ce and Nvidia Docker2 software for GPU containerized setup. The DLI RAPIDS Course – Base Environment container image v1.0.0 available at the Nvidia Container Catalogue (NGC) was used to launch a Python Jupyter Lab environment for this experiment on the AWS virtual machine. The NGC DLI RAPIDS container image already comes with preinstalled software including Rapids, Conda, Graphiz, cuDF, cuPy, etc., simplifying the experimental setup. In addition, the pull and launch of the container image expose internal ports to the container

(a)




(b)

| | CPU | GPU/RAPIDS |
|---|---|---|
| *Data handling* | Pandas | cuDF |
| *Machine Learning* | Scikit-learn | cuML |
| *Graph analytics* | NetowrkX | cuGraph |

| | CPU | GPU/RAPIDS |
|---|---|---|
| *Viz* | Bokeh/ Datashader | cuXfilter |
| *Geospatial* | GeoPandas/ SciPy.spatial | cuSpatial |
| *Signals* | NetowrkX | cuSignal |
| *Cyber* | cyberpanda | CLX |

**Figure 4.6:** (a) Nvidia Rapids Framework, (b) Comparison of Rapids to popular libraries

and allow for global internet access to the Jupyter Lab environment outside the localhost environment. The algorithm and overview of the data structuring approach for processing the CV and sensor data fusion are presented in Figure 4.7, with each step numbered in curly brackets.

The main reason behind structuring the data in such a format is because MDAs can store and organize large amounts of data better than CSV, which allows for more efficient processing of files. One CSV file size 16GB can be structured into a 20MB MDA. Our approach was to query the data from several CV data files across several folders and drives into a giant temporary in-memory database and then transform it into a Spatio-temporal 3D lattice with unique attributes that can be further used for attribute-based hyper-dimensional analysis. To achieve this, we used the Dask framework for massively large distributed data processing and filtering with the GPU backend on Nvidia Rapids. After setting up a local cluster, the Dask framework was used to read all the CV and sensor data files 1 and filtered on interest columns into a giant in-memory data lake 2. A new unique index was computed for the data, and the data was repartitioned to reduce the number of Dask workers and optimize performance while at 2. In order to translate the data into a 3D Spatio-temporal matrix, unique indices of each data row were computed using the procedure in 3a. This began with the computation of the unique spatial discretized bins for longitude and latitude, and each data row longitude and latitude were used to compute the spatial positional index and placed in the appropriate bin.

The same procedure created a discretized temporal bin based on the day, hour and minute. Using the unique spatial, temporal, and directional indices, unique unrolled positional global indices were computed for each data point which was then used to translate the in-memory database into the 3D spatial-time lattice 3b. Each spatial-time lattice cube 4 contained all data entries with the same index as well as other attributes such as speed and direction, which could then be used in hyper-dimensional data operations based on

the data attributes 5. After filtering and stacking based on attributes, other analytics based on speed, data counts and direction were performed and used in this study.

To benchmark the experimental setup, we used the in-built Python timeit() function

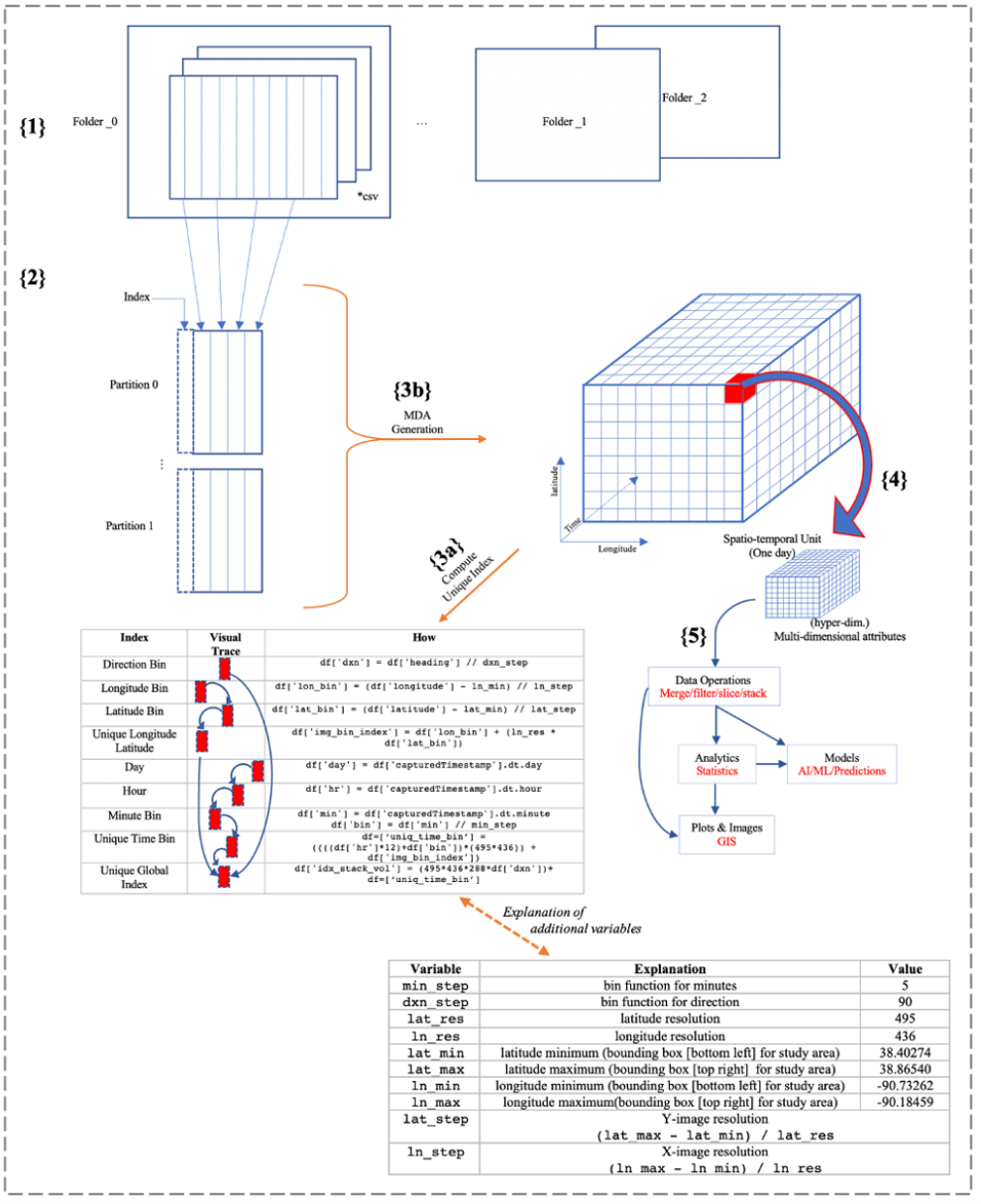



**Figure 4.7:** Overview of data structuring approach

**Table 4.2:** Overview of Weather stations in the city of Saint Louis, colored by weather condition

| Number | CV Data | Name | Platform |
|---|---|---|---|
| 1 | | Data Binning | |
| 2 | | Indexing - Latitude | |
| 3 | Speed | Indexing - Longitude | |
| 4 | | Normalization | |
| 5 | | Data Export | |
| 6 | | Reduction - Count | CPU/GPU |
| 7 | | Reduction - Sum | |
| 8 | | Indexing - Latitude | |
| 9 | Volume | Indexing - Longitude | |
| 10 | | Filter | |
| 11 | | Normalization | |
| 12 | | Data Export | |

with repeat() method to run each algorithm a couple of times. The standard deviations of the average, best, and worst running times were noted and examined. Table 4.2 provides the setting for these studies considering different algorithm modifications for speeding up Extract, Transform, Load (ETL) workflows for huge CV data.

### 4.4.4 Comparison of CPU versus RAPIDs GPU Source Code

Algorithm 1 below displays the original code for the preparation of the large CV data. Here, the code initializes the standard libraries before implementing the traffic volume and speed data preparation logic. The algorithm for binning traffic speed data to a 2D picture array is shown in Algorithm 1 , and the indexing of the speed data along the latitude bins is presented. Algorithm 2 illustrates how to use the RAPIDs framework to achieve the same outcomes. Standard libraries are initialized, and a GPU cluster is set up as presented.

### 4.4.5 Performance Evaluation of the Running Times

The performance evaluation of the ETL pipeline for huge CV data under various algorithm optimizations stated in the part before is presented in this section. The section displays

**Algorithm 1** Sample code from big CV data on CPUs

**procedure** BINING SPATIAL POINTS
    $d \leftarrow$ ('2021-02-06', '2021-02-07', '2021-02-08',...)
    $h \leftarrow$ ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ...)
    $t \leftarrow$ range(1,13)
    *loop*: $(t)$*in range(1,13):*
    $df \leftarrow' time\_bins\_csv/' + str(n) + '/' + str(d) + '/' + str(h) + / + str(t) + '.csv'.$
    $df \leftarrow df[['latitude', ' longitude', ' speed']].$
    $df['latitude'] \leftarrow df['latitude'] * -1.$
    $x_{cut} \leftarrow pd.cut(df.latitude, np.linspace(-38.71, -38.53, 248), right = False).$
    $y_{cut} \leftarrow pd.cut(df.longitude, np.linspace(-90.32, -90.18, 219), right = False).$
    $df \leftarrow df.groupby([x_{cut}, y_{cut}]).mean().$

**Algorithm 2** Sample code from big CV data on GPU using RAPIDS and Dask CUDA

**procedure** BINING SPATIAL POINTS
    $df['bin'] \leftarrow df['min']//min_{step}.$
    $df['dxn'] \leftarrow df['heading']//dxn_{step}.$
    $df['lat\_bin'] \leftarrow (df['latitude'] - lat_min)//lat_{step}.$
    $df['lon\_bin'] \leftarrow (df['longitude'] - ln_min)//ln_{step}.$

the experiment run times under various conditions, as stated in Table 4.3. The summary of 25 runs for each experiment on CPUs and GPUs are shown here. When the experiment is run on the GPU, the speed improves noticeably, going from 25.6 times to 72.2 times faster. Overall, the GPU experiment lasted only 25 minutes, while the typical experiment on the CPU lasted over 42 hours.

### 4.4.6 UNet Model

Image segmentation and classification has been widely successful using UNet. U-Net is a CNN based on a fully convolutional neural network where its architecture is altered and expanded to work with fewer training images to obtain significantly precise segmentation results. While training on an NVIDIA GTX 1080 Ti GPU, the segmentation of a 495 * 436 image took less than a second. As shown in Figure 4.8, UNet's architecture consists of a contracting path to absorb context and an expansive symmetric path to facilitate precise

**Table 4.3:** Running times of the ETL algorithms by number of CV data in seconds

| | Metric | Metric | CPU (seconds) | GPU (seconds) | Speedup (X) |
|---|---|---|---|---|---|
| Data Binning (Bucketing) | Speed | Avg | 31207.30 | 442.92 | |
| | | Min | 28250.79 | 399.21 | |
| | | Max | 34100.80 | 482.94 | 70.45 |
| | | Std. Dev. | 1610.11 | 28.54 | |
| | Volume | Avg | 270.36 | 4.52 | |
| | | Min | 244.85 | 3.75 | |
| | | Max | 292.50 | 4.99 | 59.74 |
| | | Std. Dev. | 14.57 | 0.30 | |
| Data Indexing | Latitude (Speed) | Avg | 17302.22 | 248.48 | |
| | | Min | 15621.62 | 222.11 | |
| | | Max | 19099.29 | 268.87 | 69.63 |
| | | Std. Dev. | 978.03 | 14.20 | |
| | Latitude (Volume) | Avg | 20386.04 | 295.99 | |
| | | Min | 18430.08 | 269.42 | |
| | | Max | 22027.04 | 314.43 | 68.87 |
| | | Std. Dev. | 1103.70 | 13.92 | |
| | Longitude (Speed) | Avg | 16096.90 | 222.91 | |
| | | Min | 14394.11 | 202.23 | |
| | | Max | 17544.64 | 246.68 | 72.21 |
| | | Std. Dev. | 971.42 | 14.31 | |
| | Longitude (Volume) | Avg | 18564.84 | 260.43 | |
| | | Min | 17018.96 | 236.06 | |
| | | Max | 20318.35 | 285.40 | 71.28 |
| | | Std. Dev. | 987.98 | 15.64 | |
| Normalize | Speed | Avg | 1148.60 | 16.63 | |
| | | Min | 1048.05 | 15.35 | |
| | | Max | 1245.09 | 17.95 | 69.03 |
| | | Std. Dev. | 63.48 | 0.86 | |
| | Volume | Avg | 704.06 | 10.10 | |
| | | Min | 640.62 | 9.02 | |
| | | Max | 765.62 | 10.99 | 69.64 |
| | | Std. Dev. | 38.22 | 0.58 | |
| Reduction | Count Unique (Volume) | Avg | 37.46 | 1.46 | |
| | | Min | 34.21 | 0.53 | |
| | | Max | 41.60 | 1.97 | 25.57 |
| | | Std. Dev. | 2.09 | 0.36 | |
| | Sum (Volume) | | 42178.21 | 599.04 | |
| | | Min | 37627.96 | 535.47 | |
| | | Max | 46179.47 | 647.91 | 70.40 |
| | | Std. Dev. | 2641.91 | 31.79 | |
| | Filtering (Volume) | Avg | 826.43 | 11.83 | |
| | | Min | 752.76 | 11.03 | |
| | | Max | 886.79 | 12.98 | 69.81 |
| | | Std. Dev. | 38.37 | 0.54 | |
| Data Export | Speed | Avg | 250.79 | 4.40 | |
| | | Min | 232.20 | 3.53 | |
| | | Max | 276.38 | 4.99 | 56.94 |
| | | Std. Dev. | 13.55 | 0.34 | |
| | Volume | Avg | 142.45 | 2.55 | |
| | | Min | 124.80 | 1.95 | |
| | | Max | 152.32 | 2.97 | 55.77 |
| | | Std. Dev. | 7.46 | 0.26 | |
| | Overall | Avg | 149115.73 | 2121.34 | 70.29 |

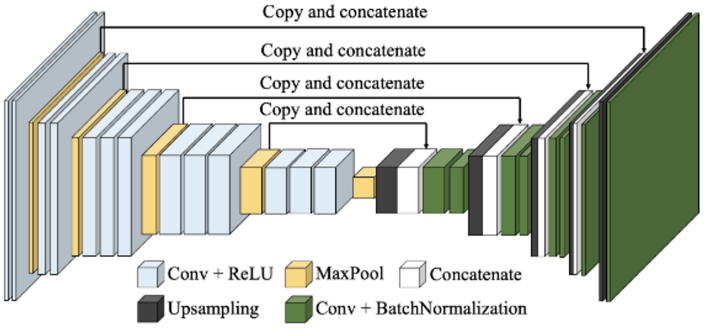
localization. The contracting path follows the typical convolutional network with multiple convolutions accompanied by Rectified Linear Unit ($ReLU$) and max-pooling operation.

Similarly, the contracting part reduces spatial information and increment in features information. However, the expansive path integrates spatial and feature information using upconvolutions with feature information from the contracting path. In our model, the convolution layer was heavily connected to the average pooling layer and then decoded using one deconvolution layer trailed by one convolution layer. We decided to use average pooling because of its ability to retain features and give smooth arrays. The learning rate is 3e-4 and was configured/lowered to improve the model performance. Adam optimizer was used as the optimization algorithm, and mean squared error was used to measure how well each model performed.

Table 4.4 presents the UNet input parameters used in our model and explains how each value was extracted/calculated.

The input to the training model is the MDAs generated from the study area with spatial and temporal characteristics, which can be defined as:

$$X_j^i = [v_i, v_{i+1}, ..., v_{i+o-1}], i \in [1, L - I - F + 1] \tag{4.1}$$




| | Output Shape |
|---|---|
| DenseBlock-1 | (495, 436, 64) |
| AvgPool | (248, 218, 64) |
| DenseBlock-2 | (248, 218, 96) |
| AvgPool | (124, 109, 96) |
| DenseBlock-3 | (124, 109, 128) |
| ... | |
| DenseBlock-7 | (8, 7, 128) |
| AvgPool | (4, 4, 128) |
| DenseBlock-8 | (4, 4, 128) |
| Conv Layer | (4, 4, 128) |
| DeconvBlock-1 | (8, 7, 128) |
| DeconvBlock-2 | (16, 14, 128) |
| ... | |
| DeconvBlock-7 | (495, 436, 128) |
| ConvLayer | (495, 436, 96) |

**Figure 4.8:** Designed UNet architecture with output shape per block

**Table 4.4:** UNet model input parameters

| Input Parameter | Value | Explanation |
|---|---|---|
| No. of training files | 24 | First 24 days |
| No. of validation files | 3 | Three random days |
| No. of testing files | 3 | Last three days |
| No. of frames/day | 288 | (60mins per hour / 5mins time bin) * 24 hours/day |
| No. of frames before | 12 | Previous hour time frames |
| No. of frame sequence | 24 | Used time frames (12) + Frames to predict (12) |
| No. of frames output | 12 | Subset to predict |
| Height | 495 | Image height |
| Width | 436 | Image width |
| No. of channels | 9 | Speed (4 directions) + Volume (4 directions) + Incidents |
| No. of channels output | 8 | Speed (4 directions) + Volume (4 directions) |
| Visual input channels | 108 | [channels (9) * Used time frames (12)] |
| Visual output channels | 96 | No. of channels output (8) * No. of frames output (12) |
| Batch size | 2 | No. of samples processed |
| Learning rate | 3e-4 | The amount that the weights are updated during training |
| Number of epochs | 20 | No. of complete passes through the training dataset |

Where,

- $i$ is the image index;
- $j$ is the channel index;
- $v_i$ is a column vector representing the traffic variable (speed/volume);
- $O$ is the span of output intervals;
- $I$ am the span of input intervals and
- $L$ is the period intervals.

The input image goes through convolution and pooling to extract the significant image features, which is the principal phase of the UNet model where the output size gets smaller in dimension. The output from this phase can be defined as:

$$O_m^k = P(\sigma(W_m^k x_m^k) + b_m^k), k \in [1, c_1] \tag{4.2}$$

Where,

- $P$ is the pooling procedure;
- $\sigma$is the activation function;
- ($W_m^k$, $b_m^k$) is the parameters of the mth layer and
- $k$ is the convolutional filter channel index.

The output from the preceding convolutional layer is max-pooled in the succeeding block, and then the identical architecture is applied again. Max pooling is applied to down-sample the size of the image (pixels), reducing the number of used parameters. The joining of layers together is done in the concatenation phase.

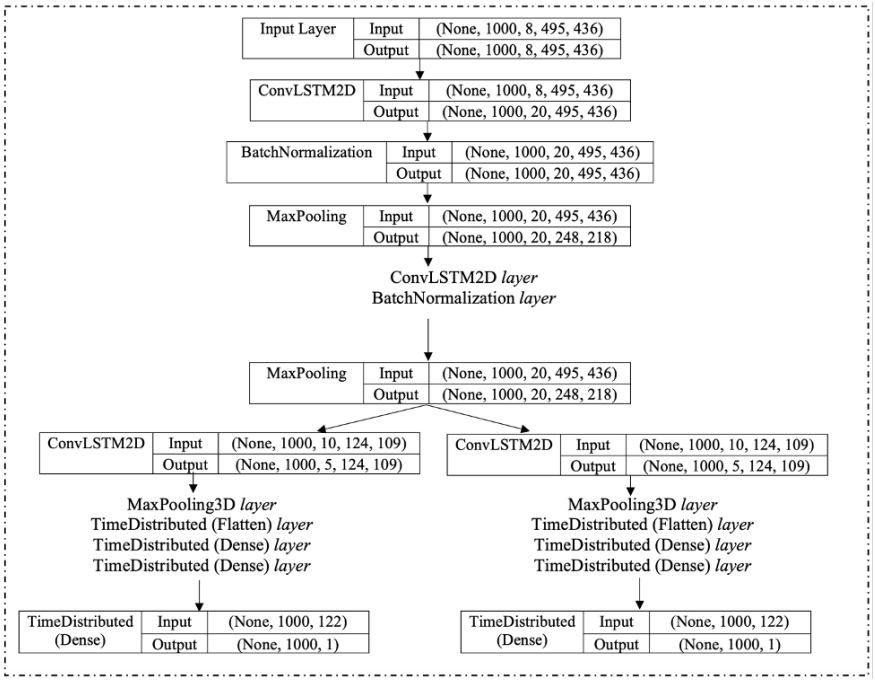



**Figure 4.9:** ConvLSTM architecture

### 4.4.7 ConvLSTM Model

The first benchmark model used to validate the accuracy and robustness of our proposed UNet model is ConvLSTM. LSTM is a Recurrent Neural Network (RNN) that focuses on learning long-term dependencies. A series of memory blocks make up the LSTM architecture. Each block has one or more self-contained memory cells, as well as three gates: input, forget, and output. The input gate receives new data from the outside and processes it. The forget gate determines when to forget the initial state and, as a result, the input sequence's ideal time lag. The output gate is responsible for generating output for the LSTM cell by combining all the computed results. ConvLSTM is a recurrent layer, except convolution operations are used instead of internal matrix multiplications.

As a consequence, instead of being a 1D vector containing features, the data that travels through the ConvLSTM cells retains the input dimension (3D in our case). ConvLSTM has been proven in recent literature that it is capable of handling the spatial temporal dependence in traffic data, however, due to its complex structure it has a longer training time. MDA (Images) is used as the model input. Figure 4.9 presents the model architecture, and Table 4.5 presents the input parameters. The shape of data is presented in the following format: (samples, frames, channels, rows, cols). The final input format is when the frames are limited to 1000 per sample, and the image is an eight-channel 495x436 pixel picture (samples, 288, 8, 495, 436). The number of available trailers for training is referred to as samples. 'returnsequences' is set to True, which means the output should be (samples, frames, categories), but because the model has eight separate outputs, the result should be (categories, samples, frames, 1), implying (8, samples, 1000, 1). Return sequences have the effect of classifying each frame into several categories.

The model architecture begins with two ConvLSTM layers, each with a 'BatchNormalization' and a 'MaxPooling' layer in between. It breaks into branches in order, one for each category. All branches start with one ConvLSTM layer and then a MaxPooling layer. The

output is then linked to a Dense network that is completely connected. Finally, the final layer is a Dense single-cell.

**Table 4.5:** ConvLSTM model input parameters

| Input Parameter | Value | Explanation |
|---|---|---|
| No. of frame sequence | 24 | Used time frames (12) + Frames to predict (12) |
| No. of frames output | 12 | Subset to predict |
| Height | 495 | Image height |
| Width | 436 | Image width |
| No. of channels | 8 | Speed (4 directions) + Volume (4 directions) |
| No. of channels output | 8 | Speed (4 directions) + Volume (4 directions) |
| Batch size | 2 | No. of samples processed |
| Number of epochs | 20 | No. of complete passes through the training dataset |
| Activation | Relu | Linear Function |
| Padding | same | The output will have the same size as the input |

### 4.4.8 Historical Average (HA) Model

The second benchmark model used to validate the accuracy and robustness of our proposed UNet model is a simple historical average model. HA simply uses the average of historical variables as predictions. We calculated the average at a spatial and temporal level for each variable/channel, meaning, data was filtered along each pixel and time bin for each day and then the average is calculated along all days. The formula used can be defined as:

$$x_{T,j,k,z} = \Sigma_{i=1}^{d} \frac{[T, W_j, H_k, C_z]}{d} \tag{4.3}$$

Where,

- $x_(T, j, k, z)$ represent the predicted pixel along a specific time-step ($T$);
- $W_j and H_k$ are the pixel index along the tensor width and height, respectively;
- $C_z$ is the channel index and

- $d$ is the number of days used in the training model.

## 4.5 Model Training

The prediction model uses the previous hour (12 frames) to predict the future hour (12 frames). The output file is a tensor of the shape (12, 495, 436, 8). The first dimension of six represents the future 12 time-bins: 5min, 10min, 15min, 20min, 25min, 30min, 35min, 40min, 45min, 50min, 55min and 60min. The width of an image is 495, and the height is 436. The main task is to forecast traffic conditions so the first eight channels (speed and volume, for each of the four headings) are forecasted. The ratio of data used for training, validation and testing is (0.8:0.1:0.1).

## 4.6 Model Testing

### 4.6.1 Recursive multi-step forecast

For the testing dataset, we select the last three days of available data to test the reliability of the forecasting model. We perform forecasting throughout all hours of the day, using an hour of actual data to predict the future hour and then using every new predicted hour for a newer prediction, as presented in Figure 4.10. The main prediction task is to test the UNet algorithm in predicting network-wide traffic speed and volume. Eventually, we forecast the traffic flow propagation throughout the day by performing a multi-step prediction. The previous hour (12 steps) of observed data is fed into the trained model to predict the next hour (12 steps), and then every new predicted hour is an updated input bin to predict the next hour.

### 4.6.2 Losses and metrics of trained model results

This section evaluates the performance of the trained UNet model against a test dataset which consisted of the last three days of data from the data collected for one month. In

order to test the performance of the proposed algorithm, statistical and deep learning-based algorithms are chosen for comparison. HA and ConvLSTM neural network is used, an extension of RNN, which is more popular due to its capability to deal with longer-term memories and evade fading gradient problems that conventional RNNs suffer from [115]. First, we will present the general results for the UNet model performance compared to benchmark models: HA and ConvLSTM, followed by a visual comparison of a few images exported from the results of each model and a deeper dive into the UNet model results. While forecasting CV speed and volume, errors from the models are calculated from the observed CV speed and volume and shall be used to justify forecasting results.

RMSE is the performance metric we use in evaluating our model because of its very

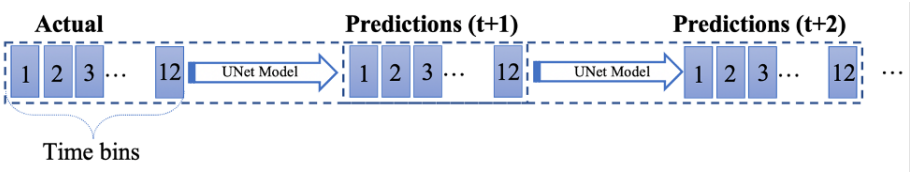



**Figure 4.10:** Testing data hourly predictions

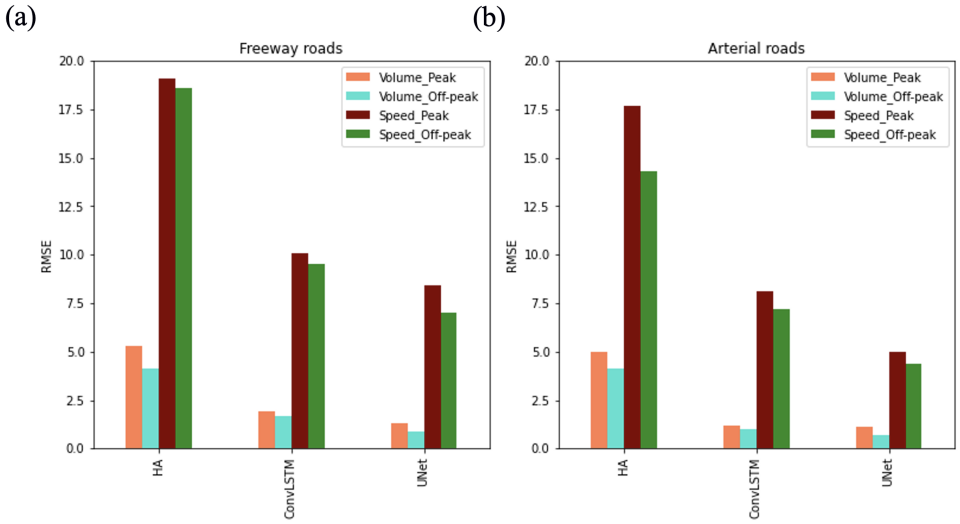



**Figure 4.11:** RMSE results across various models: (a) freeway roads and (b) arterial roads

intuitive statistic interpretation in terms of having the same measurement unit as the variable predicted, with smaller RMSE values indicating higher model accuracy. The formula can be defined as:

$$RMSE = \sqrt{\Sigma_{i=1}^{N} \frac{(\hat{y_i} - y_i)^2}{n}} \tag{4.4}$$

Where,

- $\hat{y_i} - y_i$ represents the difference between actual and predicted values and
- $n$ represents the number of samples

Structural Similarity Index Measure (SSIM) is also used when comparing images exported from each model since it is a more indicative metric that can reflect perceived structural similarity by taking image texture into account. Structural similarity refers to the assumption that pixels have many interdependencies, especially when close together. SSIM values closer to (1) indicate higher similarity, while (-1) indicate lower similarity.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4.5}$$

Where,

- $\mu_x$ is the mean of $x$;
- $\mu_y$ is the mean of $y$;
- $\sigma_x^2$ is the variance of $x$;
- $\sigma_y^2$ is the variance of y;
- $\sigma_x y$ is the covariance of $x$ and $y$;
- $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ are two variables that stabilize the division;
- $L$ is the dynamic range of pixel-values and

- $k_1$ and $k_2$ are 0.01 and 0.03, respectively, by default.

In terms of RMSE, the performance of models across all subsets can be seen in Figure 4.11 and ranked: UNet, ConvLSTM and HA, where UNet saw an average improvement of 65 percent over HA model and 15 percent over ConvLSTM. UNet significantly outperforms the other models because it applies a considerable amount of kernels to each image to perform the dense predictions at a pixel level. Ultimately, this leads to a lower RMSE across volume and speed channels, too, though the significance of error varies enormously (Speed - UNet peak NZ: 7 kph, Volume - UNet peak NZ: 1 vehicle). The reason for this is relatively simple: speed channels are normalized from 1 to 255 while volume is normalized from 0 to 255. As a result, incorrect speed forecasts are more likely to be penalized (for example, volume, which is usually close to zero for most pixels). Generally, both channels' forecasts along arterials were better than freeways, reasoned by the higher density of data points (pixels) on arterials than freeways. RMSE peaks occur during peak hours (bin 72: hour 6:00) and (bin 192: hour 16:00), reflecting the model's challenging task with higher volume around peak hours.

### 4.6.3 Extracted images comparison

Figure 4.12 presents a few images exported from the forecasting results of the models. The forecasting snippet is for hours: 5:00, 6:40, 17:00 and 20:00. SSIM and RMSE are presented above each image exported from the model and calculated concerning the observed image. The count of non-zero pixels for each image is presented below each image to analyze performance concerning spatial granularity. In terms of results, the forecasting models need to decide the non-zero positions through a map with 215,820 spaces, which is a challenging assignment because the model input state of traffic could be reduced or expanded spatially. The performance of the UNet model was dominant in predicting closer non-zero pixels, higher SSIM and lower RMSE, followed by ConvLSTM and

HA models. UNet exhibits an excellent learning ability in comprehending images because of its locally linked layers which means that output neurons are linked to local adjacent input layers, rather than all input neurons in fully-linked layers. The pooling mechanism

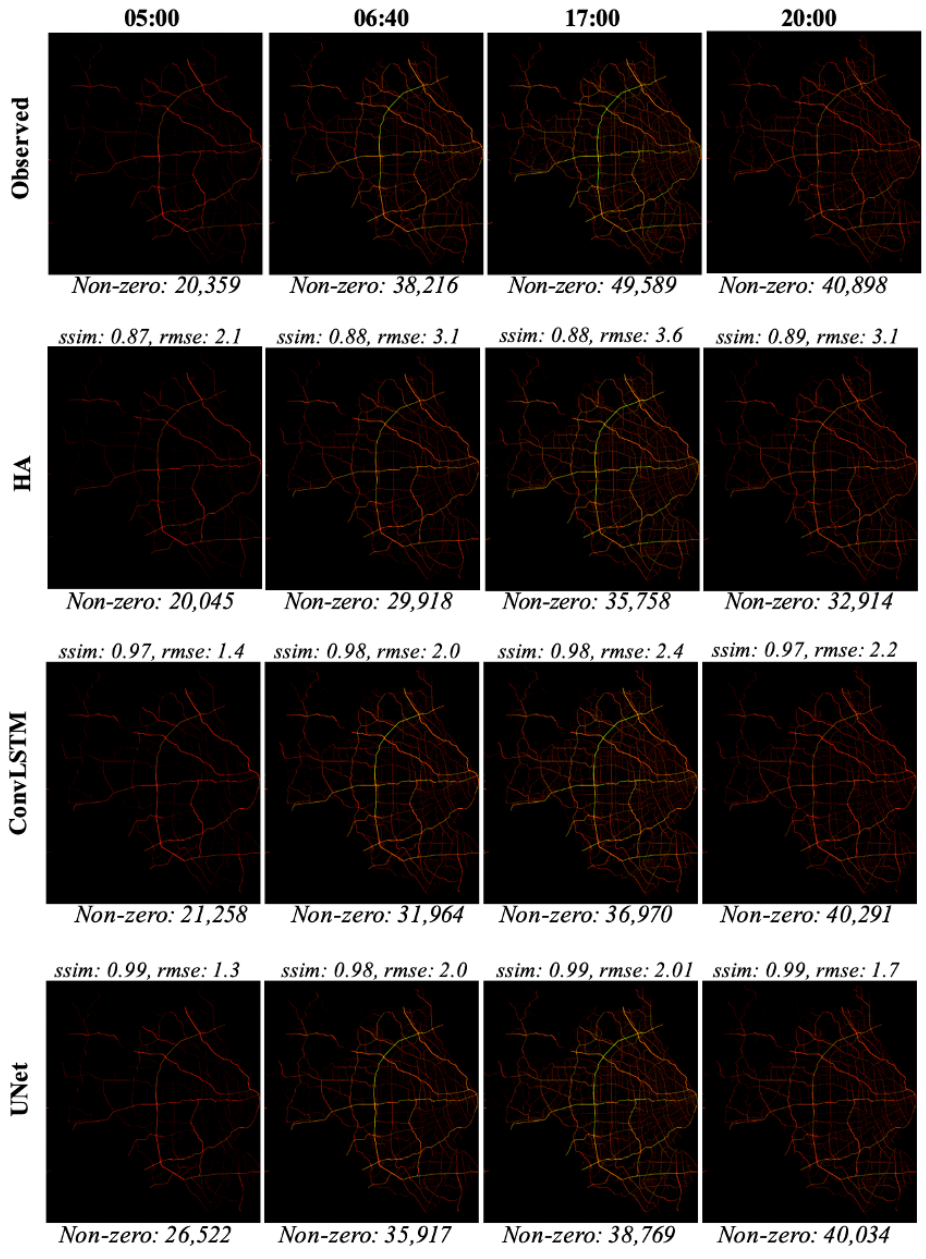



**Figure 4.12:** Forecasted snippets from prediction algorithms

in the UNet model also enhances the model to retain the essential image features while efficiently reducing the number of used parameters.

### 4.6.4 Influence of forecasting horizon

To understand the influence of the length of the forecasting period and road type on our proposed forecasting UNet model, we present Figure 4.13 as a box plot analysis of the change in RMSE along 12 future time steps averaged for the entire day forecast. Box plots provide a standardized way of interpreting the distribution of errors based on the minimum, maximum, median, 25th and 75th percentiles and the outliers. RMSEs for all plots increase over the length of prediction time steps, indicating a positive association between prediction errors and the span of prediction length. On shorter prediction horizons,

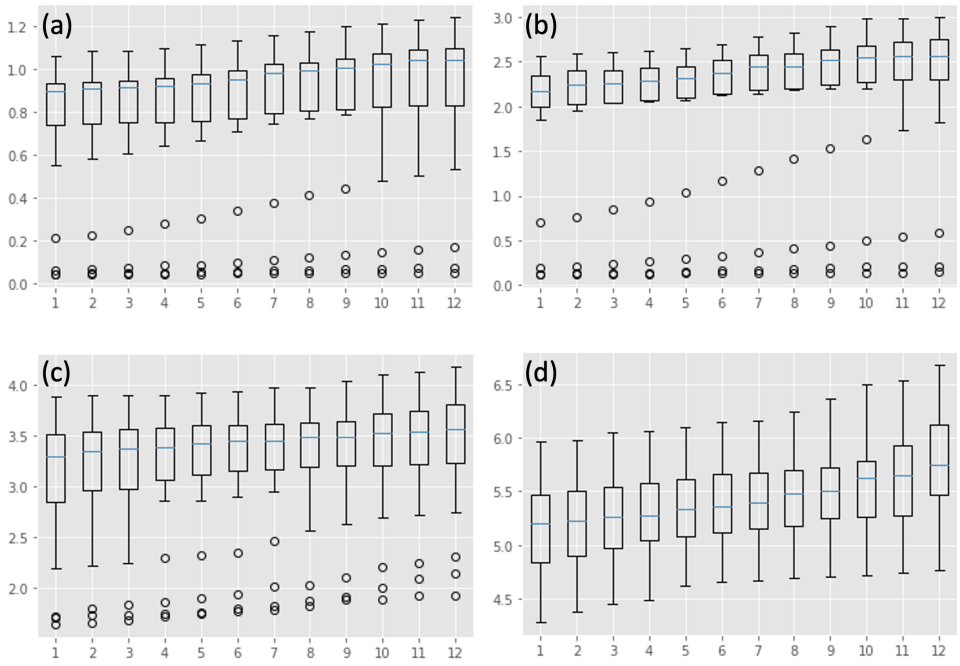



**Figure 4.13:** RMSE Box plots along 12 future time steps: (a) volume channel on arterials, (b) volume channel on freeways, (c) speed channel on arterials, (d) speed channel on freeways

the lower RMSE errors indicate that the model had better prediction estimates near-term because closer time-steps have much lower variations. The median of RMSE on Arterials is very close for volume and speed channels with minimal deviations. For Freeways, the number of time steps is larger than 6, RMSE deviations start increasing and are much larger than other cases. The number of predicted horizon time-steps tends to influence performance in such a case. It is worth noting that generally, the errors and range of errors throughout the forecasting period was stable with insignificant increases, which implies that the proposed model was robust in learning temporal features achieving the most accurate forecasts in all circumstances.

## 4.7 Additional experiments

We also experimented with different encoder and decoder structures. Instead of using average pooling in the encoder, we implemented a convolution pooling layer in two other models. We added a linear interpolation layer in parallel path to one of the additional models in addition to the deconvolution layer, which may be thought of as the inverse of the average pooling layer. The decoder block also includes tightly coupled convolution layers. We cannot assert that the additional trials are superior to the finally implemented model based on test set assessment scores alone. Performance varies per training iteration, but there is no noticeable difference in terms of performance between them in general.

## 4.8 Summary

Working with the massive amounts of data from linked automobiles continues to be difficult for everyone. In our experiment, it typically takes two days to process data from 1,500 different trips made by connected automobiles in a single day. The transportation authorities and security agencies receive no benefit from the 48-hour delay between data

sensing and interpretation. Considering this, numerous initiatives are underway to discover quicker and more effective methods of working with massive data from connected vehicles. We concentrated on GPUs for this work since they are more developed, accessible, and cost-effective. We saw up to 72 times faster performance in the GPU trials when compared to conventional CPU-based processing. Our findings support what other scientists have reported in many fields of study. For instance, one researcher claimed that employing GPUs to speed up his computing workloads resulted in speedups of up to 400X [97, 98]. The authors of a recent flood forecasting study claimed that using GPUs sped up their work by between 80X and 88X. Similar studies have demonstrated that employing GPUs as coprocessors can speed up image processing by between 10x and 20x. In this chapter, we investigated how the RAPIDS framework and Dask CUDA may be used to accelerate big CV data pipelines on GPU. According to our findings, the complete procedure was 70 times faster when the computation was reduced from 41 hours to 25 minutes. In addition, the RAPIDS and Dask architecture made the source codes considerably easier by condensing most computations to a single line of code, except for the initial library imports and cluster setup routines. The original CPU code, however, required numerous lines of code (about 20 for each task). In conclusion, the necessity for real-time sensor data processing and data fusion will continue to be a difficulty as our community's connected vehicles and sensors proliferate. Considering this, we think that utilizing contemporary tools like GPUs and other accelerators can offer a low-cost means of processing these data in real-time. This will enhance the administration and security of our transportation infrastructure while enabling faster real-time insights. The faster developed data processing pipeline paved the way for faster processing of input data to the traffic forecasting models presented. Prediction of traffic flow has seen a rich use of deep learning methods, which yielded satisfactory results. These approaches can perform dense predictions and portray more non-linear functions than other neural networks [115, 116].

However, most of these studies address a single step, channel or route prediction. A multipurpose, multi-step, spatiotemporal forecasting is necessary to improve the accuracy of predictions and provide a longer prediction length into the future. In the scope of this study, the UNet model has the following properties: (a) space and time features can be extracted automatically because of the implementation of convolutional and max-pooling layers; and (b) represents speed, volume and incident features on a pixel-level dense traffic network that are then used to create traffic speed and volume predictions on all routes. The testing model used one hour of actual data to forecast all future hours. To test the applicability of the proposed model and its performance, the comparison to HA statistical method and ConvLSTM saw an average improvement of 65 percent and 15 percent, respectively. The image snippets from each prediction model to the actual image showed that image textures were more similar in UNet than the benchmark models used. UNet's dominance in performing image predictions was also evident in multi-step forecasting, where the increase in errors was relatively minimal over longer prediction spans. Most existing traffic flow prediction research, to our knowledge, focuses on finding models with higher prediction accuracy; however, this work not only provides a long-term prediction model with trustworthy accuracy, but also examines the underlying process of structuring network-wide data. It provides a different way of thinking about structuring large-scale point data to forecast high-level traffic features. With the availability of more accurate traffic predictions and historical transportation data from multiple datasets, we are now able to develop a web-app that can query and visualize transportation user requested information in an efficient manner. In the next chapter, we present the architecture of the developed platform and how leveraging AI power can speed-up web user requests.

# Chapter 5

# Interactive Web Platform Powered by Speech Queries

## 5.1 Introduction

The increasing complexity of urban transportation networks makes it difficult to manage transportation operations in cities. ITSs and ICTs are frequently used to handle traffic monitoring, estimate, and control problems. In order to apply suitable control techniques, ITSs combine modern technology with real-time information about traffic conditions. Transportation networks are closely monitored, resulting in massive traffic and incident databases. The problem of traffic congestion on the roads is serious and widespread, and the integration of various technologies and systems can greatly aid in its resolution. The requirement for massive traffic databases to be efficiently used by traffic operators and managers necessitates the development of innovative apps and state-of-the-art visualization tools as the amount of traffic data transmitted via ITSs grows fast. Interactive visualization allows extracting data of interest by displaying it in various visual forms and interacting with it through various filters. The amount of data generated is rising quickly

in the digital age. Huge amounts of data are typically stored in a database and filtering is typically made possible using query languages such as Structured Query Language (SQL). However, queries from the database can be a daunting task since it demands knowledge from the user's side about the exact schema of the database, functions of different entities in the query and correct join paths of different tables within the database. The technical challenges of formal query languages typically overwhelm non-technical users of the database. To navigate this daunting task and allow users to easily make requests to the database, the use of speech or NLP can be helpful. If users would want to find all accidents on I-70, for instance, the input question through the microphone would be: "Show me all accidents on I-70" and the platform would translate the keywords into an SQL query that it can then use to query the requested information from the database. Most transportation agencies use ArcGIS, Tableau, and D3 as their primary visual analytic platforms. Tableau, an analytical visualization tool, is used by the NHTSA (National Highway Road Safety Administration) to offer insights regarding speed-related traffic fatalities across the United States. Other agencies, like the Virginia Department of Transportation (VDOT2015), the BTS (2019), and the Iowa Department of Transportation, employ comparable platforms to dig into work zone, traffic, and freight data. The data being visualized on these platforms might be anywhere from a few megabytes to a few gigabytes in size. When the size of the data being viewed surpasses 250 megabytes, significant latency might be detected in terms of updates. For all the heavy-lifting calculations (on large data sizes) such as data ingestion, aggregation, integration, and reduction, recent advances aimed at managing huge transportation data employ high-performance computing clusters in the backend [117]. The data is then provided to the front end for visual exploration after being filtered, aggregated, and lightweight. Although this method is useful for managing the challenges of massive data, it restricts the effectiveness of visual analytics because tiny details are lost in the aggregate and filtering processes [118]. The main purpose of this chapter is to

develop an interactive visual analytics application that allows the big CV dataset (historical and predicted) to be visualized, interacted with, and analyzed in the browser (front end). The framework will allow speech queries and heavy-lifting computations like data reduction, aggregation, and filtering to be easily performed with user input from the front end.

## 5.2 Related Work

The large volume of traffic data recorded in transportation databases makes it difficult for humans to understand and extract traffic patterns directly from the data. Given the variety of such complex and big datasets such as transportation data, data visualization is critical and required for their interpretation. It simplifies the process of discovering the structure, characteristics, anomalies, patterns, and interconnections in complex data, which can be time-consuming. For performing speech queries, Automatic Speech Recognition (ASR) and Machine Translation (MT) are commonly used in speech-to-text translation systems [119]. Hundreds of hours are then needed to perform the audio transcription and building a high-quality MT would demand millions of words of parallel text – resources of which are only available for a small fraction of the estimated 7,000 languages [120]. To process words from a listener (microphone), [121] suggested a system to tokenize words that uses the knowledge of the underlying database to automatically construct a lex file (spelling dictionary) which contains information about the underlying database, such as columns and table names. Speech is first turned into text in the initial phase, followed by a grammar checker to check whether the text is syntactically correct or not. In the following phase, a lexer, parser and syntax guided translation are used to map the text into an intermediate question. The intermediate query's (SELECT) and (WHERE) clauses are extracted in the fourth phase. The fifth phase is when all necessary tables are located to create the (FROM) clause and build the SQL query. A pre-

pared SQL query is then sent to the database and returned in the sixth phase. To test their developed system, they used it on single and numerous tables, and it produced accurate results only when the input query compiled with the syntactic rules in terms of syntax. [122] proposed a strategy that is most frequently used in addressing the speech understanding issue by using an unsupervised Bayesian network. They start by describing three techniques for vector representation of words, which are meant to aid the Bayesian network in developing effective concepts. The approach is then put to test using data from two different applications comparing the results of Bayesian network to those of Kohonen maps and K-means algorithm. [123] suggested the use of Search Over Data (SODA) that uses keyword searches of business users and automatically produces executable SQL and provides data with a search experience analogous to Google. The key concept is to employ a graph pattern matching algorithm that takes advantage of the data warehouse's metadata scheme. For displaying the data, visualization specifies a variety of visual forms and interactions. It can not only provide a qualitative overview of large data sets, but it can also help identify areas of interest and parameters for more detailed quantitative study. This prompted some academics to concentrate their efforts on developing visualization tools to aid humans in comprehending traffic patterns. Shekhar et al. [124] created CubeView, a web-based visualization software for monitoring sensor network measurements collected from the Minneapolis-St. Paul (Twin-Cities) metropolitan area's motorway system. The app allows users to identify patterns and rules from previous data to help make better decisions. Approximately 900 sensor stations make up the sensor network. Sensors have one to four loop detectors, depending on the number of lanes. Sensors measure the amount of traffic on the road and send the information to the Traffic Management Center. Raw data acquired by loop detectors is saved in binary format in CubeView, then transformed to text data and stored in database servers. Traffic managers, traffic engineers, travelers and commuters, as well as researchers and

planners, can use the transportation visualization tools. Piringer et al. [125] investigated tunnel surveillance videos. Different sorts of occurrences were automatically detected, ranked, and marked in place and time. Users could view the original videos for each event. The visualization techniques used by Zaiat et al. [126] are as follows: a map-based view of the performance state of local transportation systems; filtering dashboard information by transport domains, modes, and components; aggregations for any Level of Service (LOS) and geographic abstraction; and charting perspective of transport system behavior over time. The Advanced Interactive Traffic Visualization System (AITVS) was proposed by Lu et al. [127], which provides data cube visualization features for real-time and historical pattern analysis. It's a web-based visualization system that uses cutting-edge visualization components including spatial and temporal plots and a data cube to evaluate and monitor traffic conditions, volume, speed, and occupancy, and so overcomes the shortcomings of other systems. AITVS, like our proposed web applications, is geared for traffic analysts and managers rather than travelers. Pack [128] presented a web-based visual analytics solution for finding and observing major bottlenecks. It includes a dashboard with a map and a popup window for displaying the journey time index, various contour plots, an interactive animated map for displaying average speeds, travel times, reliability, and other metrics, interactive charts and graphs, and a performance summary table. The time spiral image, which depicts the time of occurrence and how long the bottleneck lasted until it was resolved, is an intriguing visualization tool included in the bottleneck ranking (Figure 5.1 adopted from [127]).

Web-based traffic visualization tools are generally simple to use and can be used to reduce complex and tedious statistical data, offering useful information to both traffic specialists and travelers. Several visualization tools have been built expressly for the analysis and understanding of congestion levels. CongestionGrid [85] is an example of congestion level estimation and representation. It's a platform that automatically gathers current

congestion data from a traffic data provider and displays previous patterns on a grid for customers to see. Users can use CongestionGrid to investigate temporal traffic trends by seeing congestion data from a certain week or an aggregation of data over a period of time. (Figure 5.2 adopted from [85]) uses visual depiction of traffic states for traffic estimation, which uses red, yellow, and green colored cells to indicate high, normal, and low traffic, respectively.

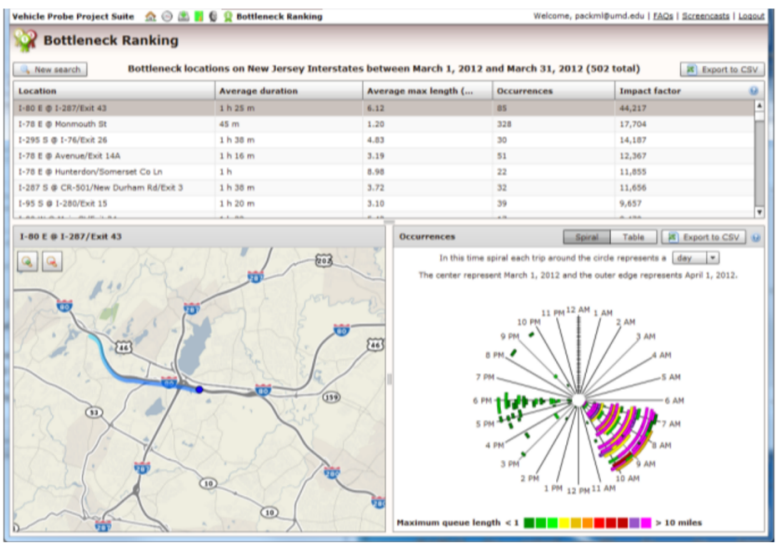



**Figure 5.1:** Visualization of ranked bottlenecks locations

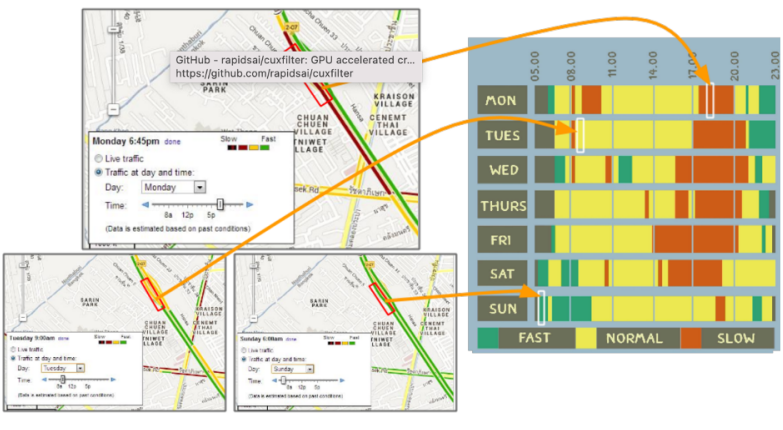



**Figure 5.2:** Congestion grid created from historical traffic data

Maps, graphs, and clusters were employed as visualization tools by Diker and Nasibov [129] and Yoon et al. [130]. Diker and Nasibov clustered road segments based on traffic congestion levels, while Yoon employed spatio-temporal traffic status plots of trace data in addition to threshold-based quadrant clustering. As a source for visualization, Wang et al. [131] used traffic trajectories (as a significant form of traffic data obtained from road sensors), as well as incidents, road speed, and traffic congestion. This Beijing-based approach also organizes the relationships between traffic jams. Visual study of traffic trajectories frequently necessitates aggregation, such as a density map [132]. The density map depicts the trajectory density and allows for the detection of "hot" regions. The authors used a variety of techniques, including propagation graph level estimates, traffic jam density display on a map (OpenStreetMap), topological filters, temporal and size filters, map matching, and so on. The authors used animation, flow maps, and graph layout approaches to show the propagation graph. The system provides five visualization views: 1) pixel-based road velocity view, which displays speeds and events; 2) graph list view, which displays propagation graphs; and 3) graph projection view, which renders the topological relationships of propagation graphs; and 4) spatial view, which represents traffic; 5) multidimensional filter view, which allows for filtering by time, space, size, and topology, as well as the propagation path of a single chosen graph; and sorting the propagation graphs by size and similarity yields a structured representation. Unlike this technique, Pack et al. [133] and Khotanzad [134] looked at transportation incident datasets rather than traffic trajectories. They created Incident Cluster Explorer, a web-based visual analytics tool. It is a program that displays the spatial, temporal, and multi-dimensional characteristics of incidents using an integrated view interface. Users' engagement is aided with choices for selection, filtering, and clustering incidents, as well as an emphasis on a smaller dataset. Multiple visualization tools, such as histograms, interactive maps, two-dimensional and parallel coordinate plots, can interact with each other at the same time.

The authors utilized either scatter plot mode or grid mode to depict relationships between a pair of variables. There are also two mapping types to choose from: icon mode and heat mode. This program is far more complete, sophisticated, and user-friendly than some websites (FARS), which present a considerable amount of row data but do not offer any visualization options, leaving this challenging task to the user who can only download them. Another advantage of this tool over commercial data visualization programs like Spotfire and Tableau is the ability to plot data on a map using heat maps to minimize occlusion and overcrowding when dealing with massive datasets like transportation incident data. Anwar et al. [135] proposed Traffic Origins, a simple way for visualizing the impact of road incidents on congestion and vehicle flow in their immediate neighborhood, as well as the cascading effect of many incidents on a road network. The incident site is marked with an expanding circle just before a traffic incident to disclose the basic traffic flow map, and it recedes once the incident is over. They designed appealing visualizations to assist traffic management controllers in simply comprehending and accessing traffic and congestion data. One of the most efficient and visually pleasing map tools, to date, is Kepler.gl. This tool has evolved from a single page app to a robust geo-analytics and visualization platform since its inception as an internal product in 2020, It creates an all-in-one geographic data exploration and visualization environment, and it's been widely utilized by Uber engineers, analysts, and data scientists to fuel advanced geospatial analytics. Data scientists, architects, visualization specialists, and engineers from Mapbox, Limebike, Airbnb, Sidewalk Labs, HERE technologies, Atkins Global, Cityswifter, UBILabs, and 300000kms have found kepler.gl's simplicity, capability, and speed to be extremely valuable. Academics, such as architecture student Diego Crescêncio from Estácio de Sá in Rio de Janeiro, have also used the software. The author used open crime data at kepler.gl for his studies to better understand the built environment for urban design research. To understand how urban architecture might improve safety within favelas, he's been em-

ploying 2D and 3D representations of data relevant to city-wide crime statistics. A data scientist at CitySwifter, explored the origin and destination trips (home-to-work) in New York city using Kepler's brushing interaction (Figure 4). The brushing interaction allows the user of the map dashboard to hover the mouse (click) over different regions (origins) to display the arcs of the destination of such trips. Processing maps with many different variable or fused datasets is a challenging and time-consuming task. The fundamental reason for this is because maps contain a lot of overlapping data, to deliver as much insight as possible. As a result, reducing user effort in map processing and developing effective interactive tools for visualizing traffic data can transportation planners better comprehend map representations and integrate them into a variety of applications and eventually make better data-driven decisions.

## 5.3 Methodology

This section introduces the developed analysis platform. Figure 5.3 presents the components of our developed web app. At the top of the diagram, the device is composed of the monitor (hardware) that will display the platform powered by React app (frontend) and SQLite database (backend), which are connected with Express (Application Programming Interface (API)).

### 5.3.1 Speech to SQL Queries

To speed up queries performed using speech on our platform, the development of a system that can quickly listen and render results on the platform is critical. Our simple speech to SQL system is designed without the use of any trained models, in comparison to the conventional approach used by other authors in literature. Figure 5.4 presents an example of our designed system, where a user would perform a speech request through the microphone and the listener returns a series of words. We use React's speech Recognition [136], to extract the words in a text format. The returned words are then compared

to a list of words we have created from the fields within our database to assure that it matches the user's request. Once that phase is successfully passed, an SQL statement is generated from the keywords and performed on the summary table named (Data). The returned information is then used to perform queries on the actual tables to finally display the results on screen. The main concept is that our main data table (Data) is structured (merged) on common attributes before feeding it into the database so that SQL queries don't have to perform any (JOIN) methods. This avoids the systems confusion with the received text and helps in skipping the use of a ML model to fully understand the state-

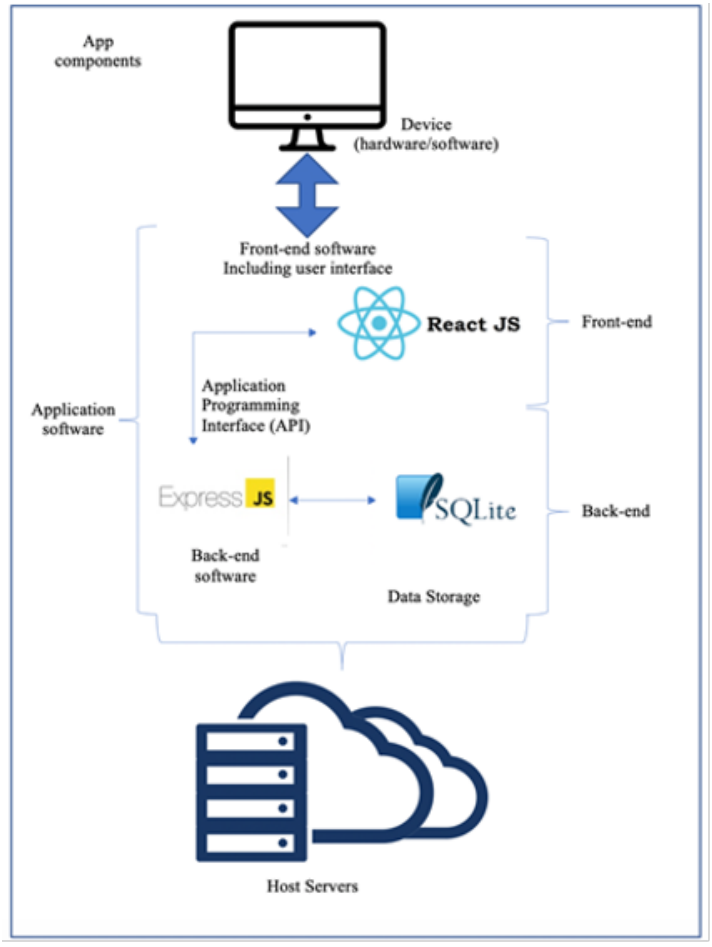



**Figure 5.3:** WebApp components

ment made by the user.

The logic of the developed framework is like a decision tree heuristic in a way; depending on the availability of data returned from the query requested by user, further queries are activated from other data tables. In addition to querying data, the voice commands can also perform mouse functions and help navigate through the different pages in the platform by activating windows methods to control the screen size, reroute to a new page, scroll through a page, save a map layer, export filtered table, etc.

### 5.3.2 User Perspective

From a user perspective, we aim towards developing a delightful experience for users navigating through the website. Thus, it's important to consider an organized structure for the web-app architecture and design layout. The structure of our developed application is presented in Figure 5.5. The home page presents the main page through which website users can navigate to parent pages. Each parent page then links to children pages. The hierarchical website structure is used in our design in order to form understandable, discoverable and predictable patterns.

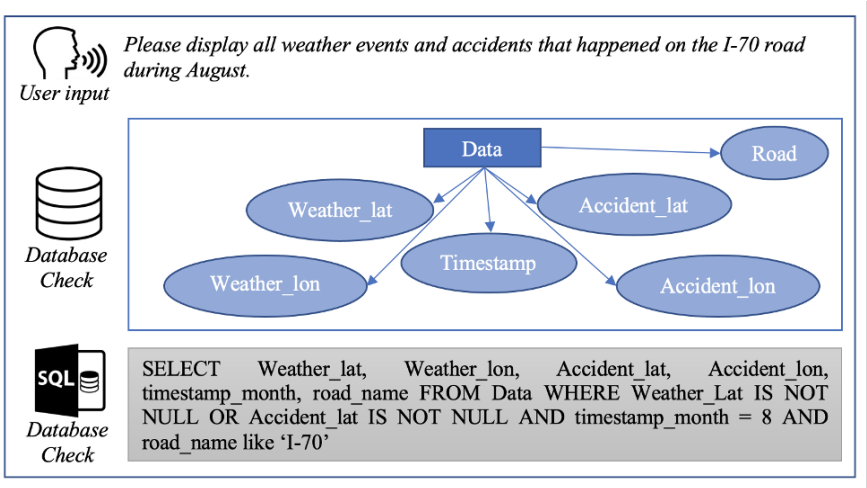



**Figure 5.4:** Design of Speech to SQL system

The home page at the top of the structure acts as a hub for the application visitors through which they have cards displaying the pages that a user can navigate to such as: Historical Analysis, Predictions and Whatif Scenarios. Categorization of the data we are displaying in this format allows for faster and easier decisions by users to reduce the amount of time spent considering a decision. Subcategories within each category allow for a structured methodology when browsing and categorizing information, especially with complex data. Individual pages or child pages at the bottom of the hierarchy contain the basic elements of the website so that the user's time browsing the website or consuming content can be minimized.

### 5.3.3 Development and Design Perspective

For the development of the Web-App and User Interface(UI) coding, we used the following languages: "$HTML, CSS, JavaScript, Python, SQL$" where:

- HTML: Hyper Text Markup Language that builds the main structure of the web page.

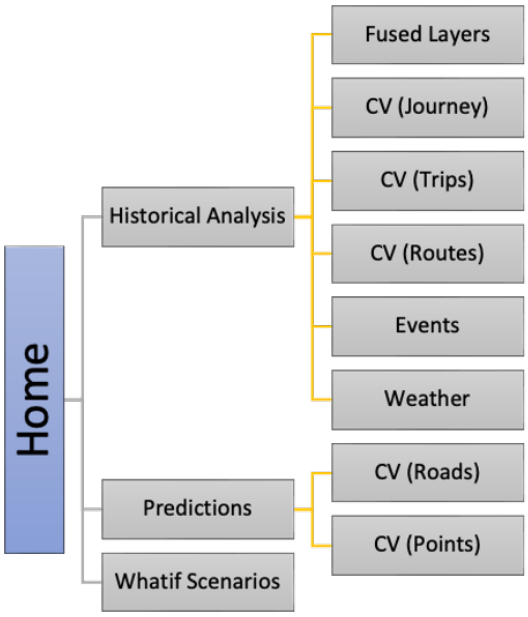



**Figure 5.5:** Design of Web Application pages

- CSS: Cascading Style sheets that are used to style the web page.
- JavaScript: Allows for dynamic behavior and interactions on the web page.
- Python: Structuring ?? files before inserting them into the database – in line with the goal in mind.
- SQL: Manipulating the tables in the backend database (SQLite) with functions such as addition, deletion...etc.

Since such a project demands multiple pages with heavy interactions, it's critical to have a structured folder approach. The structure presented in Figure 5.6 worked best for our project after multiple trial and error experiments of design, testing, integration and delivery of content. A few aspects that were kept in mind during the entire design process are: Easiness of locating files, consistency throughout the application with the design structure and naming of files to easily locate components.

### 5.3.4 Frontend Development

First is the main folder named "Web-App Structure" which contains two main folders "Front End" and "Back End". The user-side of a web application is also known as the frontend. Inside the browser, the frontend is the interface that the user can access and interact with. The client-primary side's goal is to collect data from users in an engaging manner. JavaScript code is used to script the frontend components built in React.js. React.js is an open-source JavaScript package that is used to create single-page apps' user interfaces. For web and mobile apps, it's utilized to manage the view layer. We can also make reusable UI components with React. There are several open-source systems, such as Angular, that make developing front-end web applications easier, however, React has competitive benefits over other frameworks such as:

- React is relatively easy to use thanks to its component-based design, well-defined

lifecycle, and use of only plain JavaScript.

- React is simple to comprehend for anyone with a basic understanding of programming, whereas Angular and Ember are described as 'Domain-specific Languages,' meaning that they are harder to learn. You only need a basic understanding of CSS

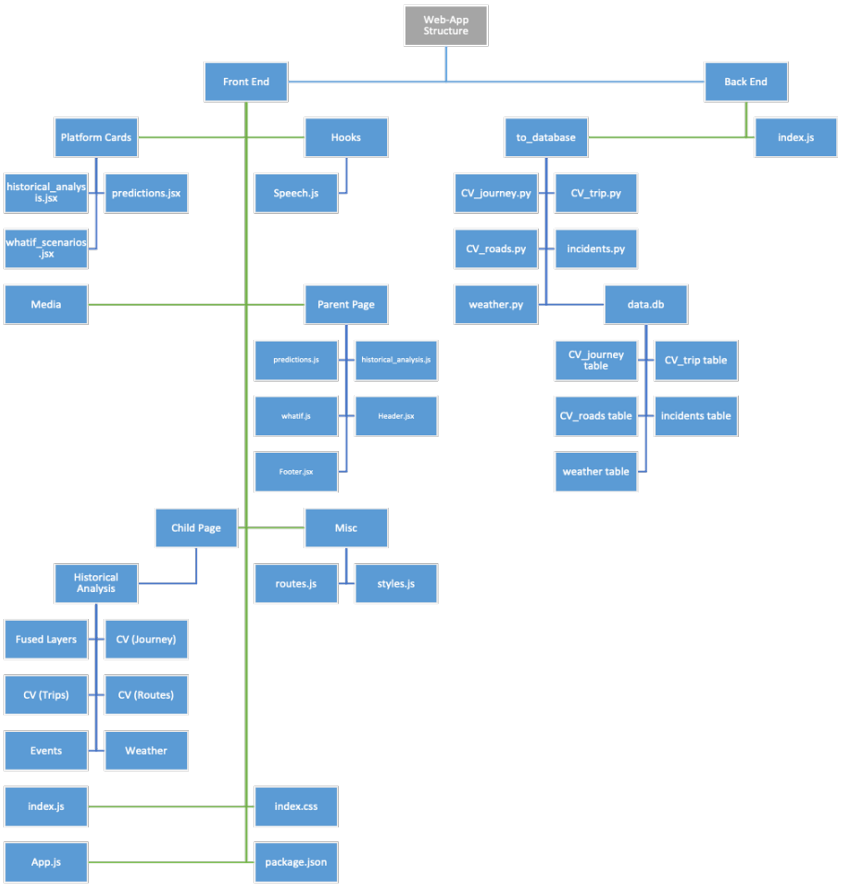



**Figure 5.6:** Design of Web-Application structure

and HTML to "React". HTML is a markup language that may be used to generate both static and dynamic web pages and apps. CSS is a style sheet language that controls how documents produced in a markup language are displayed.

- The Virtual Document Object Module (DOM), which represents the document structure, style and content, is used by React to keep track of the values of each component's state. When the state of a component changes, React compares the current DOM state to the new DOM state. After that, it determines the most cost-effective method of updating the DOM.

- When data is updated, React's simple programming approach allows it to alter state automatically. This takes place in the memory; thus, it is quick.

- React's library is likewise quite small. It's only about 6 kilobytes in size. This is a fraction of the size of its competitors.

Inside the "Front End" folder, we have multiple folders and files that make up the frontend of the project such as:

- Platform Cards:

  $Historical\_analysis.jsx$: contains an exported function that defines a media card along with its dimension properties, colors, font details and button actions for the historical page.

  $Predictions.jsx$: contains an exported function that defines a media card along with its dimension properties, colors, font details and button actions for the predictions page.

  $Whatif.jsx$: contains an exported function that defines a media card along with its dimension properties, colors, font details and button actions for the Whatif page.

- Misc:

  $Routes.js$: contains a list of items for the navigation bar with each item described with its respective id, label, path, icon, active icon and component. The id describes the items order on the navigation bar. Label gives a name for the item on the navigation bar. Path defines the item's route. Icon presents the item's image on the navigation bar while inactive (unclicked), while the active icon presents the item's image on the navigation bar while active (clicked). Component links the actual child page to the item.

  $Styles.js$: includes styling of the different pages with alignment, width, height, margins, padding, colors, position, etc. For example, setting 'flex' as a display property allows for automatic adjustment of elements within a webpage upon stretching or shrinking a webpage.

  $Navigation.js$: contains a drawer element that displays menu items and link them to their respective routes upon clicking. The navigation panel can also be minimized and maximized easily with a click, to allow users a wider display of page content.

  $MenuItem.js$: returns a list of menu items mapped to their respective icons with an icon appearing based on the user's choice or click.

- Parent Page:

  $Header.jsx$: contains a static bar at the very top of the page that is visible during all website interactions and navigations. A home button is added so that users can easily navigate to the home page at any time.

  $Footer.jsx$: contains a static bar at the very bottom of the page that is visible only on the home page providing information about the website developers.

  $Historical\_analysis.js$: Contains a summary description for each of the child

pages within historical analysis (parent page) along with a button that routes to the clicked menu item page once clicked. The page also contains a navigation bar that can easily switch the user from one child page to another within the history page.

$Predictions.js$: Contains a summary description for each of the child pages within predictions (parent page) along with a button that routes to the clicked menu item page once clicked. The page also contains a navigation bar that can easily switch the user from one child page to another within the predictions page.

$Whatif.js$: Contains a summary description for each of the child pages within Whatif (parent page) along with a button that routes to the clicked menu item page once clicked. The page also contains a navigation bar that can easily switch the user from one child page to another within the Whatif page.

- $Fused\_config.json$: json file describing the properties of the rendered map with information such as: attributes to display, filters to use, styling of points, data type, map state and map style.

$Fused\_dash.js$: returns the navigation bar along with the map from $Fused_map.js$

$Fused\_map.js$: returns a map function that fetches data from the backend database and dispatches it to the map layer. Customized reducers are then used to control the map.

$CV(Journey)$... similar structure to Fused Layers folder with naming relevant to the current folder.

$CV(Trips)$ ... similar structure to Fused Layers folder with naming relevant to the current folder.

$CV(Roads)$ ... similar structure to Fused Layers folder with naming relevant to the current folder.

$Weather$ ... similar structure to Fused Layers folder with naming relevant to the current folder.

$Events$ ... similar structure to Fused Layers folder with naming relevant to the current folder.

- Hooks:

  $Speech.js$: contains a list of commands that can be heard by the web-app user. The commands are translated to window methods and queries to the backend database. For example: user can say "open history page" and that will activate an open method within the window to route (/history)

  $Index.js$: Contains the ReactDom that renders the React elements to the web page

  $Index.css$: Stylng of the main home page with margins, padding, positions and colors.

  $App.js$: Contains two main components: speech and routes. They are both linked on this page so that speech is always activated throughout different pages. Routes provide the path for all parent and child pages.

  $Package.json$: Collection of all the libraries and packages imported in this project along with their version number.

### 5.3.5 Frontend Development

The server-side of the application is also known as the backend. SQLite is used for data processing, handled by the backend components. The data is manipulated and validated in the backend stores in response to the requests that users send are handled. The majority of requests "fetch" the data that the user has requested. SQLite is a relational database management system that runs without a server. It is a zero-configuration, in-memory

open-source library that does not require installation. It is also convenient because it's only 500kb in size, far smaller than other database management systems. The following are some of the benefits of utilizing SQLite as an application file format:

- For tables, SQLite employs dynamic types. It indicates that any value, regardless of data type, can be stored in any column.
- SQLite allows several database files to be accessed at the same time using a single database connection. This adds several useful functionalities, such as connecting tables across databases or copying data between databases with a single query.
- SQLite can create in-memory databases that are extremely quick to work with.

Inside the "Back End" folder, we have multiple folders and files that make up the back-end of the project such as:

- to database:

  $CV\_journey.py$: After connecting to the created SQLite database, a CV_journey table is created with the table fields mirroring the CSV file columns. Datatype for each field is also defined in this step. The CSV file is then read line by line and each column value is inserted into the created table.

  $CV\_trip.py$ ... similar structure to $CV\_journey.py$ file with naming relevant to the current folder.

  $CV\_roads.py$ ... similar structure to $CV\_journey.py$ file with naming relevant to the current folder.

  $Incidents.py$ ... similar structure to $CV\_journey.py$ file with naming relevant to the current folder.

  $Weather.py$ ... similar structure to $CV\_journey.py$ file with naming relevant to the current folder.

- Data.db

  $CV\_journey\_table$

  $CV\_trip\_table$

  $CV\_roads\_table$

  $Incidents\_table$

  $Weather\_table$

  $Index.js$: Here we connect the created SQLite database and then use Express API to send the data from each of the created tables to a unique route. Each layer in the created map on the front end demands a specific structuring of data to be sent.

### 5.3.6 Application Programming Interface

Finally, to connect the frontend and backend components, Express server is used to create controllers, routes and server. Controllers are server-side routines that process all requests sent to specific API endpoints (i.e, when you use the retrieve endpoint in the React app to get all the traffic flow on a specific data from the map layer, one of these functions will generate the response). Routes will be the next quick and straightforward step after controllers. The Express library will then be used to develop the router for all API endpoints, such as retrieving traffic flow on all days, routes with highest congestion index, incidents causing highest delays, etc.

## 5.4 Performance Evaluation

### 5.4.1 Query Speeds and User-Friendliness

The queries performed on the platform use six main datasets as presented in Table 5.1 with information about the number of rows and query response times. The output of each query is a JavaScript Object Notation (JSON) formatted data that is loaded from a comma-delimited (CSV) file containing all the information requested by the user. The displayed times are the maximum query times for all the rows within the dataset.

**Table 5.1:** Query response time for data tables

| Dataset | Rows Count | Query Time (sec) |
|---|---|---|
| Incidents | 191,075 | 2 |
| Weather | 227,140 | 2 |
| Connected Vehicles (point) | 3,163,946 | 5 |
| Connected Vehicles (arc) | 919,413 | 3 |
| Connected Vehicles (line) | 9,854,555 | 8 |
| Connected Vehicles (journey) | 9,854,555 | 8 |

In addition, the platform enables interactive user-friendly operations from users. Users can select or use voice to filter locations of interest from the chart by zooming in, zooming out, or using a circular filter. Different time periods can also be selected using the accompanying charts as explained in the following sections. The following characteristics contribute towards making our platform user-friendly:

- Clear logic and navigation for broad topics only, where users arriving at the platform for the first time can easily navigate around without the need for any guesswork.
- Responsive and compatible design, where the application can function properly on different screen sizes (desktop, tablets, phones) with a range of browsers.

- Easily digestible content with most important information offered first.
- Clickable links have a consistent clickable look so that users can easily be directed to home or exit pages.
- Accessible using voice commands than can respond back to the user with helpful commands that can help direct the user around the platform.

## 5.5 Web-App Pages and Features

Starting at the first page, web-app visitors are first welcomed with the home page as presented in the Figure 5.7. The Navigation bar is then presented along all parent and children pages or routes. A visual of the navigation bar is displayed on the right (non-active and active icons).

To manage the state and data flow of the CV dataset, we use Kepler.gl. Kepler, a React component, is a web-based application for visual exploration of large-scale geolocation data sets that is data agnostic and high-performance. This web app uses Mapbox GL and deck.gl to render millions of points representing thousands of trips and conduct spatial aggregations on the fly. Layers are used as building blocks in kepler.gl to generate interactive maps, with customizable layer generation and data (e.g., fares, ETA, and timestamps) encoded to visual channels (e.g., circle size,

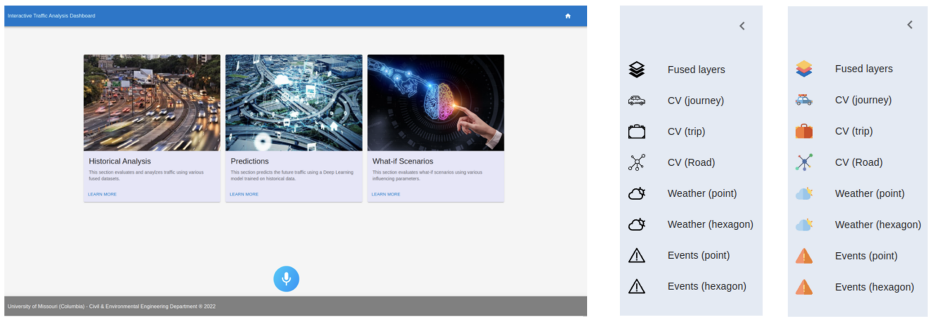



**Figure 5.7:** Screenshot of Web-App: Home Page (left) and Navigation Bar (Right)

arc color, and circle color) with scaling functions (e.g., linear, quantile, and quantize). A point layer, for example, can be used to plot event and place locations; an arc layer can be used to visualize origin-destination correlations; a hexbin or grid layer can be used to aggregate a collection of points to show its distribution; and a polygon layer can be used to visualize a choropleth map showing aggregate statistics of geographic regions. Beyond the usual 2D x and y cartographic plane, we use the Geographic JavaScript Object Notation (GeoJSON) format discussed earlier to add Line-string geometry along with a timestamp for each point; a third dimension to encode data in an isometric perspective view that displays the movement of vehicle from one point to another. A user can more rapidly spot abnormalities in an aggregate map when actual movement of vehicle is displayed, as demonstrated in Figure 5.8.

In addition to standard metric-based filtering, we add Brushing properties (Figure 5.9) to allow users to highlight arcs and points that originate within a particular radius of where the mouse is now located (hovered) over the map. This feature is particularly useful for visualizing origin-destination correlations in order to better

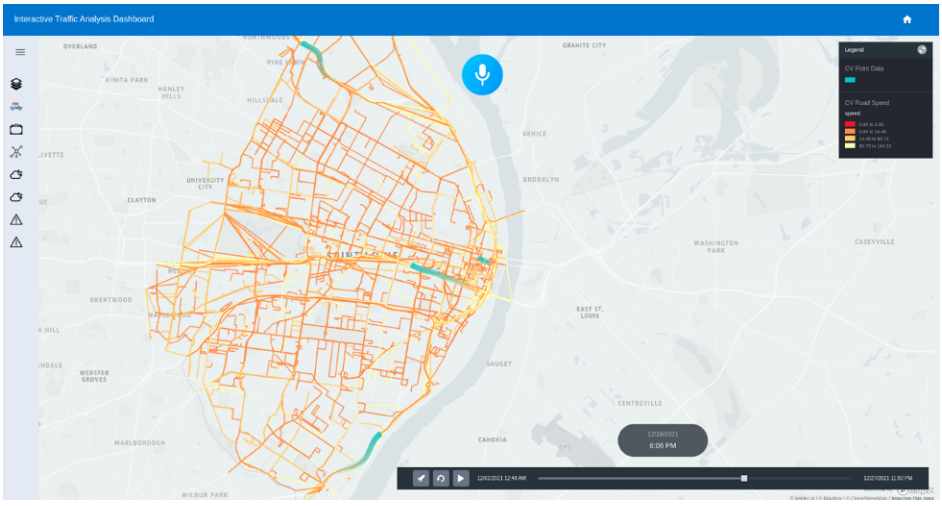



**Figure 5.8:** CV journey's from point to point along their routes

understand how different locations are connected.

We developed the current web-app from a single page app to a robust geo-analytics and visualization platform. It creates an all-in-one geographic data exploration and visualization environment, and can be used to fuel advanced geospatial analytics. The speed, deep insights and geo-analytical capabilities of the toolbox allows us to achieve powerful and quick data analysis for CV historical data and predictions and perform road predictions as displayed in Figure 5.10.

## 5.6 Strategy Canvas

To understand how our developed platform compares to common relational database systems, we use the blue ocean strategy. [116] introduced the phrases “red ocean” and “blue ocean” to represent the market universe in their influential book “Blue Ocean Strategy”. All the existing industries make up the known space or “red oceans”. Industry oceans and the game’s rules of competition are well known. Traditional

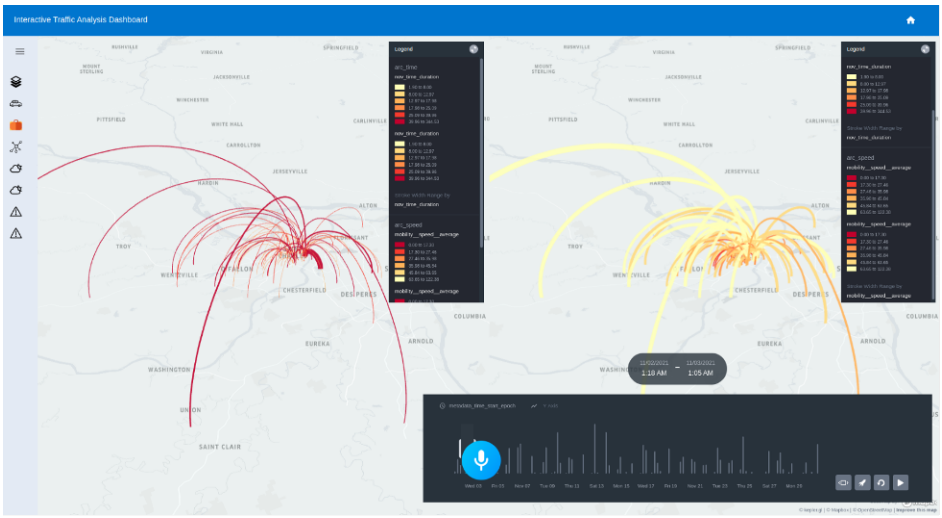



**Figure 5.9:** Layers can be rendered using subtractive blending (left) and additive blending (right)

players strive to outperform their rivals in order to snag a bigger piece of the market. Profits and growth are decreased as the market becomes more crowded which fuels fierce competition; thus, the name "red ocean". In contrast, "blue oceans" represent all sectors that do not yet exist and have an untapped market space. Demand is created rather than contested. Competition is unimportant since the rules of the game are still being established. An analogy used to depict the greater and deeper potential to be discovered in untapped market space is a "blue ocean". In terms of lucrative growth, a blue ocean is large and deep. We use the same concept to higlight the value of our application as presented in Figure 5.11, with the strategy canvas labeled in terms of costs and capabilities for our developed platform (blue) and the traditionally used platform such as Oracle (Red). A score out of 10 was given for each characteristic and so a higher score reflects a higher value. Our platform utilizes open-source language and software tools, so the cost of development is much more affordable. However, integration of such open-source demands effort and technical knowledge from the developer's side. The usage of CPUs for querying

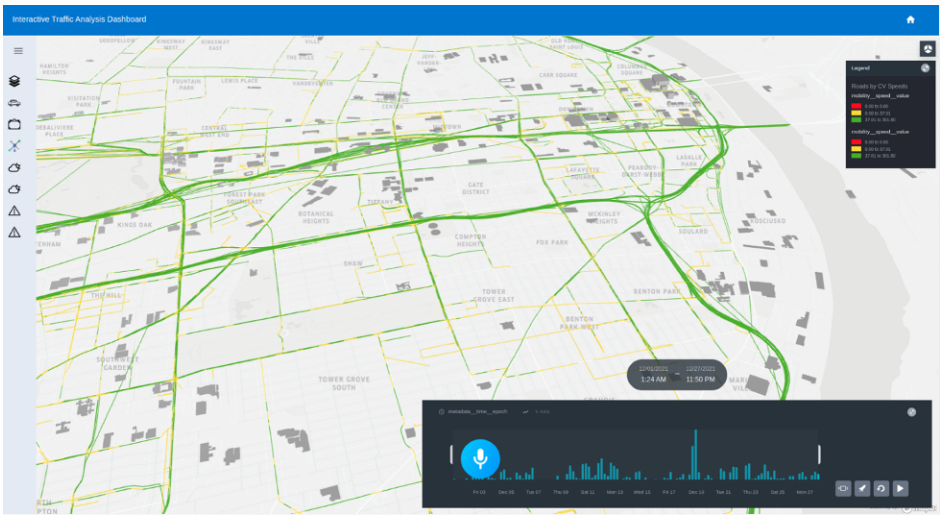



**Figure 5.10:** Roads colored by speed from CV speeds

data for both platforms is reflected with a tied score for cloud costs. The location of the platform's deployment largely affects the administration's cost. A relational database service like Oracle costs substantially more if the server is built internally. In terms of automation, many procedures on our platform are automated, meaning that with new unseen data, the app can quickly update to the changes. For Oracle, however, an administrator needs to monitor the changes, rewrite queries and in turn increase the workload. In terms of capabilities, our platform offers much more than just interactive visualizations of 2GB Data (Oracle) where we can perform that on large amounts of data seamlessly with no delays on the front-end. The platform developed also allows for geospatial analysis and working with GeoJSON files, essentially offering predictive analytics. Lastly, we offer speech navigation and queries for easier browsing.

## 5.7 Summary

A fully functional, interactive Web-App has been designed for storing, retrieving, fusing and visualizing numerous massive transportation datasets. The development of the application was done utilizing the latest developments in the science of

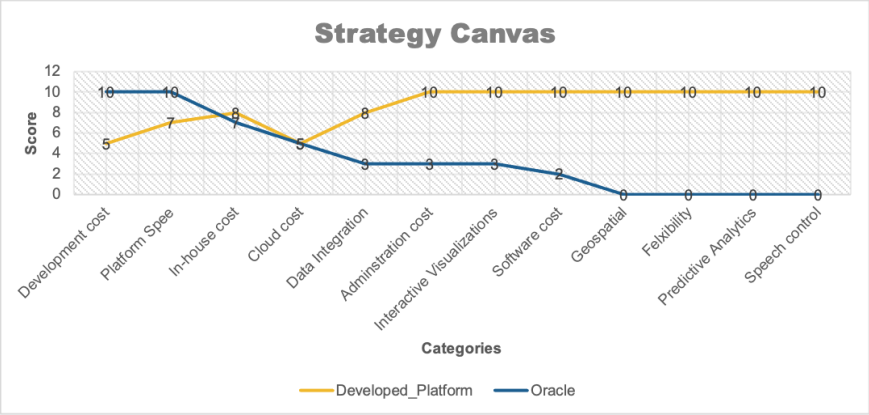



**Figure 5.11:** Strategy canvas (value curve) for the developed platform and Oracle

big data. The developed web-app allows for a lightning-fast analysis and visualization of the data. To create the application, a modular design structure was adopted for the front and back ends. A user navigating through the front end submits requests through various clicks or voice commands that are then sent to the back end to fetch the data. The main goal of designing a fast and interactive platform is to minimize the latency between front and back ends which eventually minimizes the amount of time spent by the user on the application and provides quicker insights.

# Chapter 6

# Conclusion

In this dissertation a solution to control traffic congestions was developed by delivering an interactive web application that performs faster prediction with higher accuracy data. That was made possible by accomplishing four designed objectives. First, we proposed a GC-GRU based neural network traffic forecasting model and compared various traffic forecasting techniques for a small dataset of few routes. After that, we conducted a comparison analysis between the suggested model and state-of-the-art models like HA, LSTM, and Transformers. A comparison of the model results show that the suggested GC-GRU is a challenging rival to cutting-edge traffic forecasting techniques. The developed architecture has the following advantages:

- Model performance on the provided test data ranked second with a MAPE of 3.16 which is very close to Transformer's performance of 3.12.
- It's worth noting that our model not only had the fastest inference time, but also had a training time that is six times faster than Transformer.

Connected vehicle data offers a wide array of opportunities for transportation sys-

tems and operations management. This technology makes use of vehicle-to-everything (V2X) communications to address roadway mobility and safety concerns, such as travel time and near-accident occurrence. Agencies interested in integrating this data with existing data sources should understand the added benefits and limitations of the technology. In this dissertation, we compared CV data to traditionally used probe data to extract the benefits it can offer in accurately detecting incident and congestion events. The key contributions that were found are as follows:

- Both CV and probe data indicated a strong correlation between speed on the freeway. The observed difference is that the CV data captures the peak hours better than the probe data.
- The bias on freeways was found to be significantly less than the bias on arterials. This can be explained by the fact that most probe data do not cover enough arterials, resulting in large biases.
- For the jam-stand-still-traffic condition, it is observed that the CV data detected the freeway congestion about 3-minutes on average prior to the probe data.
- Also, CV data detected more traffic incidents on the freeway than probe data.
- Similarly, the CV data detected more incidents on the arterial than probe data.
- It was observed that the influence of probe penetration rate is insignificant, if an incident leads to jammed or heavy traffic.
- The study's findings indicate that the higher the penetration rate, the lower the speed bias.

In addition, we developed larger scale models that can handle the volume of CV data and learn not only the temporal but also the spatial features that may exist. The key contributions and findings are summarized below:

- We investigated how the RAPIDS framework and Dask CUDA may be used to accelerate big CV data pipelines on GPU and managed the complete procedure at 70 times faster speed, reducing the computation time from 41 hours to 25 minutes.
- In addition, the RAPIDS and Dask architecture made the source codes considerably easier by condensing most computations to a single line of code, except for the initial library imports and cluster setup routines. The original CPU code, however, required numerous lines of code (about 20 for each task).
- To test the applicability of the proposed UNet model, the comparison to HA statistical method and ConvLSTM saw an average improvement of 65 percent and 15 percent, respectively. The image snippets from each prediction model to the actual image showed that image textures were more similar in UNet than the benchmark models used.
- UNet's dominance in performing image predictions was also evident in multi-step forecasting, where the increase in errors was relatively minimal over longer prediction spans.

To accomplish the last objective, we developed an interactive visual analytics web application that enables fast speech queries, visualization, interaction, and analysis of the big CV data (historical and anticipated) in the browser (front end). The web application leverages Central processing units (CPUs) to enable computationally intensive tasks like data reduction, aggregation, and filtering to be carried out with user input from the front end.

The presented work, however, comes with a few limitations and recommendations for future work. While the average penetration rate of CVs in the study is around 8%, it was observed that CVs on freeways contribute more to that number than CVs on arterials. One way to tackle this issue can be by performing simulation on the

roads with fewer CVs to ensure a consistent average and more reliable results. The traffic events data used to locate congestions and incidents is reported by drivers on the road and so their location might not be very accurate. To accomodate for that, we filtered the most severe events by type with the highest number of votes by drivers. A possible extension to improve its reliability can involve comparing it to police reports, to better understand the bias such data can generate.

The prediction models trained in this dissertation were focused on the study area of Saint Louis city in the state of Missouri, but can be scaled for other geogrpahical areas of interest by applying transfer learning. This allows for the use of a smaller dataset than the one we used for training the current model. Another possible extension for the platform features can be the use of hand gestures (vision) to navigate along with the speech queries. The additional feature can allow drivers to navigate the application in more than one method without having to touch the screen while driving.

Finally, when analyzing the predicted recurring congestion events through the developed platform its important to be cautious of the positive feedback loop when collecting data in the predicted locations since the higher number of detected events can be due to the efforts focused on the source rather than an actual increase in the number of events.

# Bibliography


[1] B. E. David Schrank and T. Lomax, "2019 urban mobility report," 2019. → page 1

[2] D. Ni, J. Li, S. Andrews, and H. Wang, "A methodology to estimate capacity impact due to connected vehicle technology," *International Journal of Vehicular Technology*, vol. 2012, 11 2012. → page 3

[3] A. Olia, B. Abdulhai, H. Abdelgawad, and S. Razavi, "Assessing the potential impacts of connected vehicles: Mobility, environmental and safety perspectives," *Journal of Intelligent Transportation Systems Technology Planning and Operations*, vol. 20, 06 2015.

[4] A. Olia, S. Razavi, B. Abdulhai, and H. Abdelgawad, "Traffic capacity implications of automated vehicles mixed with regular vehicles," *Journal of Intelligent Transportation Systems*, vol. 22, no. 3, pp. 244–262, 2018. [Online]. Available: https://doi.org/10.1080/15472450.2017.1404680 → page 3

[5] J. V. Werf, S. E. Shladover, M. A. Miller, and N. Kourjanskaia, "Effects of adaptive cruise control systems on highway traffic flow capacity," *Transportation Research Record*, vol. 1800, no. 1, pp. 78–84, 2002. [Online]. Available: https://doi.org/10.3141/1800-10 → page 3

[6] C. Schulz, "Efficient local search on the gpu - investigations on the vehicle routing problem," *J. Parallel Distributed Comput.*, vol. 73, pp. 14–31, 2013. → page 5

[7] Z. Horváth, R. A. P. Perdigão, J. Waser, D. Cornel, A. Konev, and G. Blöschl, "Kepler shuffle for real-world flood simulations on gpus," *The International Journal of High Performance Computing Applications*, vol. 30, pp. 379 – 395, 2016. → page 5

[8] A. R. Brodtkorb, T. R. Hagen, and M. L. Sætra, "Graphics processing unit (gpu) programming strategies and trends in gpu computing," *J. Parallel Distributed Comput.*, vol. 73, pp. 4–13, 2013.

[9] S. Lee and S. Park, "Performance analysis of big data etl process over cpu-gpu heterogeneous architectures," in *2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW)*, 2021, pp. 42–47. → page 5

[10] C.-T. Yang, C.-L. Huang, and C.-F. Lin, "Hybrid cuda, openmp, and mpi parallel programming on multicore gpu clusters," *Computer Physics Communications*, vol. 182, no. 1, pp. 266–269, 2011, computer Physics Communications Special Edition for Conference on Computational Physics Kaohsiung, Taiwan, Dec 15-19, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465510002262 → pages 5, 6

[11] M. Garland, S. M. L. Grand, J. R. Nickolls, J. Anderson, J. Hardwick, S. A. Morton, E. H. Phillips, Y. Zhang, and V. Volkov, "Parallel computing experiences with cuda," *IEEE Micro*, vol. 28, 2008. → pages 5, 6

[12] P. Du, R. Weber, P. Luszczek, S. Tomov, G. D. Peterson, and J. Dongarra, "From cuda to opencl: Towards a performance-portable solution for multi-platform gpu programming," *Parallel Computing*, vol. 38, no. 8, pp. 391–407, 2012-08 2012. → page 6

[13] Y. Zhang and Y. Jia, *Parallelization of Implicit CCHE2D Model using CUDA Programming Techniques*, pp. 1777–1792. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/9780784412947.175 → page 6

[14] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks," *Sensors*, vol. 17, no. 7, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/7/1501 → pages 6, 64

[15] B. Medina-Salgado, E. Sánchez-DelaCruz, P. Pozos-Parra, and J. E. Sierra, "Urban traffic flow prediction techniques: A review," *Sustainable Computing: Informatics and Systems*, vol. 35, p. 100739, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210537922000725 → pages 6, 36, 64

[16] T. Han, K. Tang, and T. Oguchi, "Short-term travel speed prediction for urban expressways using convolutional neural network and tensor decomposition," *Transportation Research Procedia*, vol. 48, pp. 962–974, 2020, recent Advances and Emerging Issues in Transport Research – An Editorial Note for the Selected Proceedings of WCTR 2019 Mumbai. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146520305421 → pages 6, 64, 68

[17] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? estimating travel time based on deep neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11877 → pages 6, 64

[18] B. Pan, U. Demiryurek, and C. Shahabi, "Utilizing real-world transportation data for accurate traffic prediction," in *2012 IEEE 12th International Conference on Data Mining*, 2012, pp. 595–604. → pages 6, 63

[19] B. Pan, U. Demiryurek, C. Shahabi, and C. Gupta, "Forecasting spatiotemporal impact of traffic incidents on road networks," in *2013 IEEE 13th International Conference on Data Mining*, 2013, pp. 587–596.

[20] A. PHUSITTRAKOOL, C. JEENANUNTA, and P. PRATHOMBUTR, "Evaluation of network performance under provision of short predictive traffic information," vol. 13, p. 433–450, Jun. 2015. [Online]. Available: https://wjst.wu.ac.th/index.php/wjst/article/view/1506 → pages 6, 63

[21] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SJiHXGWAZ → pages 8, 13, 14, 65, 70

[22] M. Lenzerini, "Data integration: A theoretical perspective," 01 2002, pp. 233–246. → pages 9, 70

[23] M. J. Islam, A. Sharma, and H. Rajan, "A cyberinfrastructure for bigdata transportation engineering," 04 2018. → page 9

[24] Y. Adu-Gyamfi, "Gpu-enabled visual analytics framework for big transportation datasets," pp. 147–159, 12 2019. → page 9

[25] M. R. Jabbarpour, H. Zarrabi, R. H. Khokhar, S. Shamshirband, and K.-K. R. Choo, "Applications of computational intelligence in vehicle traffic congestion problem: A survey," *Soft Computing*, vol. 22, pp. 2299–2320, 2018. → page 12

[26] J. F. Gilmore and N. Abe, "Neural network models for traffic control and congestion prediction," *J. Intell. Transp. Syst.*, vol. 2, pp. 231–252, 1995. → page 12

[27] H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang, and Z. Han, "Capsule network assisted iot traffic classification mechanism for smart cities," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7515–7525, 2019. → page 12

[28] Y. Li, D. Deng, U. Demiryurek, C. Shahabi, and S. Ravada, "Towards fast and accurate solutions to vehicle routing in a large-scale and dynamic environment," vol. 9239, 08 2015, pp. 119–136. → page 12

[29] M. Asghari, D. Deng, C. Shahabi, U. Demiryurek, and Y. Li, "Price-aware real-time ride-sharing at scale: an auction-based approach," 10 2016, pp. 1–10. → page 12

[30] D. Shi, J. Ding, S. M. Errapotu, H. Yue, W. Xu, X. Zhou, and M. Pan, "Deep ${Q}$ -network-based route scheduling for tnc vehicles with passengers' location differential privacy," *IEEE Internet of Things Journal*, vol. 6, pp. 7681–7692, 2019. → page 12

[31] H. Su, L. Zhang, and S. Yu, "Short-term traffic flow prediction based on incremental support vector regression," *Third International Conference on Natural Computation (ICNC 2007)*, vol. 1, pp. 640–645, 2007. → page 13

[32] Y. Xie, Y. Sun, and D. Chen, "Gaussian processes for short-term traffic volume forecasting," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2165, pp. 69–78, 12 2010. → page 13

[33] Y. Qi and S. Ishak, "A hidden markov model for short term prediction of traffic conditions on freeways," *Transportation Research Part C-emerging Technologies*, vol. 43, pp. 95–111, 2014. → page 13

[34] R. Wang, D. B. Work, and R. Sowers, "Multiple model particle filter for traffic estimation and incident detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3461–3470, 2016. → page 13

[35] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing, "Discovering spatio-temporal causal interactions in traffic data streams," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 1010–1018. [Online]. Available: https://doi.org/10.1145/2020408.2020571 → page 13

[36] P. Cai, Y. Wang, G. Lu, P. Chen, C. Ding, and J. Sun, "A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting," *Transportation Research Part C Emerging Technologies*, vol. 62, pp. 21–34, 01 2016. → page 13

[37] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014. [Online]. Available: https://arxiv.org/abs/1412.3555 → page 13

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735 → page 13

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. → page 14

[40] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10735 → page 14

[41] J. Zhang, Y. Zheng, J. Sun, and D. Qi, "Flow prediction in spatio-temporal networks based on multitask deep learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, pp. 468–478, 2020. → page 14

[42] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," 2016. [Online]. Available: https://arxiv.org/abs/1606.09375 → page 14

[43] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016. [Online]. Available: https://arxiv.org/abs/1609.02907 → page 14

[44] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 3634–3640. [Online]. Available: https://doi.org/10.24963/ijcai.2018/505 → page 14

[45] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2020. → pages 14, 17

[46] K. Sohn, "Forecasting road traffic speeds by considering area-wide spatiotemporal dependencies based on a graph convolutional neural network (gcn)," *Transportation Research Part C Emerging Technologies*, vol. 114, pp. 189–204, 02 2020. → page 14

[47] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 324–328, 2016. → pages 14, 16

[48] TRBAIAC, "traffic4cast data challenge." [Online]. Available: https://trbaiac.web.app/data → page 15

[49] H. K. Nihan, N.L., "se of the box and jenkins time series technique in traffic forecasting." *Transportation*, vol. 9, pp. 125–143, 1980. → page 16

[50] S. Kumar and L. Vanajakshi, "Short-term traffic flow prediction using seasonal arima model with limited input data," *European Transport Research Review*, vol. 7, 09 2015.

[51] S. R. Chandra and H. Al-Deek, "Predictions of freeway traffic speeds and volumes using vector autoregressive models," *Journal of Intelligent Transportation Systems*, vol. 13, no. 2, pp. 53–72, 2009. [Online]. Available: https://doi.org/10.1080/15472450902858368 → page 16

[52] M. Castro-Neto, Y.-S. Jeong, M. K. Jeong, and L. D. Han, "Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions," *Expert Syst. Appl.*, vol. 36, pp. 6164–6173, 2009. → page 16

[53] W.-C. Hong, Y. Dong, F. Zheng, and S.-Y. Wei, "Hybrid evolutionary algorithms in a svr traffic flow forecasting model," *Applied Mathematics and Computation*, vol. 217, pp. 6733–6747, 04 2011.

[54] M. T. Asif, J. Dauwels, C. Goh, A. Oran, E. Fathi, M. Xu, D. Mohan, N. Mitrovic, and P. Jaillet, "Spatiotemporal patterns in large-scale traffic speed prediction," *IEEE Transactions on ITS*, vol. 15, pp. 1–11, 01 2013. → page 16

[55] H. Chang, Y. Lee, B. Yoon, and S. Baek, "Dynamic near-term traffic flow prediction: System-oriented approach based on past experiences," *Intelligent Transport Systems, IET*, vol. 6, pp. 292–305, 09 2012. → page 16

[56] Z. Zhu, B. Peng, C. Xiong, and L. Zhang, "Short-term traffic flow prediction with linear conditional gaussian bayesian network," *Journal of Advanced Transportation*, vol. 50, no. 6, pp. 1111–1123, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/atr.1392 → page 16

[57] Q. Ye, S. Wong, and W. Szeto, "Short-term traffic speed forecasting based on data recorded at irregular intervals," in *13th International IEEE Conference on Intelligent Transportation Systems*, 2010, pp. 1541–1546. → page 16

[58] J. Lint, S. Hoogendoorn, and H. van Zuylen, "Accurate travel time prediction with state-space neural networks under missing data," *Transportation Research Part C: Emerging Technologies*, vol. 13, p. 347–369, 10 2005.

[59] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PLOS ONE*, vol. 10, no. 3, pp. 1–17, 03 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0119044

[60] J. Tang, F. Liu, Y. Zou, W. Zhang, and Y. Wang, "An improved fuzzy neural network for traffic speed prediction considering periodic characteristic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2340–2350, 2017. → page 16

[61] A. Abdelraouf, M. Abdel-Aty, and Y. Wu, "Using vision transformers for spatial-context-aware rain and road surface condition detection on freeways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 546–18 556, 2022. → page 16

[62] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," 2018. [Online]. Available: https://arxiv.org/abs/1801.02143 → page 16

[63] Y. Wu and H. Tan, "Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework," 2016. [Online]. Available: https://arxiv.org/abs/1612.01022 → page 16

[64] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," 2017. [Online]. Available: https://arxiv.org/abs/1707.01926 → page 16

[65] R. Ke, W. Li, Z. Cui, and Y. Wang, "Two-stream multi-channel convolutional neural network for multi-lane traffic speed prediction considering traffic volume impact," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2674, p. 036119812091105, 03 2020. → page 16

[66] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 1234–1241, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5477 → pages 16, 68, 71

[67] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," 2019. [Online]. Available: https://arxiv.org/abs/1906.00121 → page 16

[68] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01,

pp. 922–929, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/3881 → page 17

[69] Z. Cui, M. Zhu, S. Wang, P. Wang, Y. Zhou, Q. Cao, C. Kopca, and Y. Wang, "Traffic performance score for measuring the impact of covid-19 on urban mobility," 2020. [Online]. Available: https://arxiv.org/abs/2007.00648 → pages 26, 27

[70] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," 2016. [Online]. Available: https://arxiv.org/abs/1605.08695 → page 28

[71] Scikit-learn, "Tuning hyperparameters of an estimator." [Online]. Available: https://scikit-learn.org/stable/modules/grid_search.html → page 28

[72] L. Ge, H. Li, J. Liu, and A. Zhou, "Temporal graph convolutional networks for traffic speed prediction considering external factors," in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2019, pp. 234–242. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/MDM.2019.00-52 → pages 36, 65

[73] S. Fang, Q. Zhang, G. Meng, S. Xiang, and C. Pan, "Gstnet: Global spatial-temporal network for traffic flow prediction," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 2286–2293. [Online]. Available: https://doi.org/10.24963/ijcai.2019/317 → pages 36, 65

[74] J. J. Q. Yu and J. Gu, "Real-time traffic speed estimation with graph convolutional generative autoencoder," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3940–3951, 2019. → pages 36, 66

[75] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, and S. He, "Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 890–897, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/3877 → page 36

[76] C. Chen, K. Li, S. G. Teo, X. Zou, K. Wang, J. Wang, and Z. Zeng, "Gated residual recurrent graph neural networks for traffic prediction," *Proceedings*

*of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 485–492, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/3821 → pages 37, 68

[77] J. Li, Z. Han, H. Cheng, J. Su, P. Wang, J. Zhang, and L. Pan, "Predicting path failure in time-evolving graphs," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, Eds. ACM, 2019, pp. 1279–1289. [Online]. Available: https://doi.org/10.1145/3292500.3330847 → page 37

[78] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 1907–1913. [Online]. Available: https://doi.org/10.24963/ijcai.2019/264 → pages 37, 54, 68

[79] "Using big gps trajectory data analytics for vehicle miles traveled estimation," *Transportation Research Part C: Emerging Technologies*, vol. 103, pp. 298–307, 2019. → page 38

[80] J. M. Waddell, S. M. Remias, J. N. Kirsch, and T. Trepanier, "Utilizing low-ping frequency vehicle trajectory data to characterize delay at traffic signals," *Journal of Transportation Engineering, Part A: Systems*, vol. 146, no. 8, p. 04020069, 2020. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/JTEPBS.0000382 → page 38

[81] S. Khadka, P. T. Li, and Q. Wang, "Developing novel performance measures for traffic congestion management and operational planning based on connected vehicle data," *Journal of Urban Planning and Development*, vol. 148, no. 2, p. 04022016, 2022. → page 38

[82] J.-N. Meier, A. Kailas, O. Abuchaar, M. Abubakr, R. Adla, M. Ali, G. Bitar, R. Deering, U. Ibrahim, P. Kelkar, V. V. Kumar, E. Moradi-Pari, J. Parikh, S. Rajab, M. Sakakida, and M. Yamamoto, "On augmenting adaptive cruise control systems with vehicular communication for smoother automated following," *Transportation Research Record*, vol. 2672, no. 22, pp. 67–77, 2018. [Online]. Available: https://doi.org/10.1177/0361198118796375 → page 38

[83] M. M. Mekker, Y.-J. Lin, M. K. I. Elbahnasawy, T. S. A. Shamseldin, H. Li, A. F. Habib, and D. M. Bullock, "Application of lidar and connected vehicle data to evaluate the impact of work zone geometry on freeway traffic

operations," *Transportation Research Record*, vol. 2672, no. 16, pp. 1–13, 2018. [Online]. Available: https://doi.org/10.1177/0361198118758050 → page 38

[84] H. Li, T. Platte, J. Mathew, W. B. Smith, E. Saldivar-Carranza, and D. M. Bullock, "Using connected vehicle data to reassess dilemma zone performance of heavy vehicles," *Transportation Research Record*, vol. 2674, no. 5, pp. 305–314, 2020. [Online]. Available: https://doi.org/10.1177/0361198120914606 → page 38

[85] S. Pongnumkul, N. Kamsiriphiman, J. Poolsawas, and W. Amornwat, "Congestiongrid: A temporal visualization of road segment congestion level data," in *2013 13th International Symposium on Communications and Information Technologies (ISCIT)*, 2013, pp. 589–591. → pages 39, 101, 102

[86] H. M. L. H. M. J. Saldivar-Carranza, E.D. and D. Bullock, "Longitudinal performance assessment of traffic signal system impacted by long-term interstate construction diversion using connected vehicle data." *Journal of Transportation Technologies*, vol. 11, pp. 644–659, 2021. → page 39

[87] L. H. Saldivar-Carranza, E.D. and D. Bull-ock, "Diverging diamond interchange performance measures using connected vehicle data." *Journal of Transportation Technologies*, vol. 11, pp. 628–643, 2021. → page 39

[88] A. Sharma, V. Ahsani, and S. S. Rawat, "Evaluation of opportunities and challenges of using inrix data for real-time performance monitoring and historical trend assessment," 2017. → page 39

[89] N. R. E. Laboratory, "Nrel examines u.s. transportation patterns during covid-19 pandemic." → page 42

[90] Otonomo, "The promise of connected vehicle data." → page 42

[91] A. V. R. S. Sharma, Anuj, "Evaluation of opportunities and challenges of using inrix data for real-time performance monitoring and historical trend assessment," 2017. → page 47

[92] J. D. Adler, J. Horner, J. Dyer, A. Toppen, L. Burgess, and G. Hatcher, "Estimate benefits of crowdsourced data from social media." 2014. → page 50

[93] A. D. C. B. e. a. Vallejos, S., "Mining social networks to detect traffic incidents." vol. 23, 2021, p. 115–134. → pages 51, 59

[94] M. Amin-Naseri, P. Chakraborty, A. Sharma, S. B. Gilbert, and M. Hong, "Evaluating the reliability, coverage, and added value of crowdsourced

traffic incident reports from waze," *Transportation Research Record*, vol. 2672, no. 43, pp. 34–43, 2018. [Online]. Available: https://doi.org/10.1177/0361198118790619 → page 51

[95] S. Santos, C. Davis Jr, and R. Smarzaro, "Integration of data sources on traffic accidents," 11 2016, pp. 192–203. → page 51

[96] J. Patnaik, S. I.-J. Chien, and A. K. Bladikas, "Estimation of bus arrival times using apc data," *The Journal of Public Transportation*, vol. 7, pp. 1–20, 2004. → page 68

[97] Z. Zhang, M. Li, X. Lin, Y. Wang, and F. He, "Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies," *Transportation Research Part C: Emerging Technologies*, vol. 105, pp. 297–322, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X18315389 → pages 68, 95

[98] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 3656–3663, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4247 → pages 68, 95

[99] M. Elhenawy and H. Chen, "Dynamic travel time prediction using data clustering and genetic programming," *Transportation Research Part C: Emerging Technologies*, vol. 42, p. 82–98, 05 2014. → page 69

[100] B. Yu, W. H. Lam, and M. L. Tam, "Bus arrival time prediction at bus stop with multiple routes," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1157–1170, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X11000155 → page 70

[101] F. Sun, Y. Pan, J. White, and A. Dubey, "Real-time and predictive analytics for smart public transportation decision support system," 05 2016. → page 70

[102] C. Bai, Z.-R. Peng, Q.-C. Lu, and D. J. Sun, "Dynamic bus travel time prediction models on road with multiple bus routes," *Computational intelligence and neuroscience*, vol. 2015, p. 432389, 07 2015. → page 70

[103] S. Shekhar, S. Pradhan, F. Sun, A. Dubey, and A. Gokhale, "Empowering the next generation city-scale smart systems," in *Proceedings of the 2015 IEEE 22nd International Conference on High Performance Computing Workshops*

*(HiPCW)*, ser. HIPCW '15. USA: IEEE Computer Society, 2015, p. 64. → page 70

[104] J. Sun, J. Zhang, Q. Li, X. Yi, Y. Liang, and Y. Zheng, "Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 5, pp. 2348–2359, may 2022. → page 70

[105] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, and S. He, "Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting." AAAI Press, 2019. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.3301890

[106] B. Yu, H. Yin, and Z. Zhu, "St-unet: A spatio-temporal u-network for graph-structured time series modeling," 2019. [Online]. Available: https://arxiv.org/abs/1903.05631 → page 70

[107] S. Choi, "Utilizing unet for the future traffic map prediction task traffic4cast challenge 2020," 2020. [Online]. Available: https://arxiv.org/abs/2012.00125 → page 70

[108] Q. Zhang, C. Yin, Y. Chen, and F. Su, "Igcrrn: Improved graph convolution res-recurrent network for spatio-temporal dependence capturing and traffic flow prediction," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105179, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622002822 → page 70

[109] J. Hu, C. Guo, B. Yang, and C. S. Jensen, "Stochastic weight completion for road networks using graph convolutional networks," *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 1274–1285, 2019. → page 70

[110] F. Zhou, L. Li, K. Zhang, and G. Trajcevski, "Urban flow prediction with spatial–temporal neural odes," *Transportation Research Part C: Emerging Technologies*, vol. 124, p. 102912, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X2030810X → page 71

[111] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "Gaan: Gated attention networks for learning on large and spatiotemporal graphs," 2018. [Online]. Available: https://arxiv.org/abs/1803.07294 → page 71

[112] J. Ye, J. Zhao, and K. Ye, "Multi-stgcnet: A graph convolution based spatial-temporal framework for subway passenger flow forecasting," *2020*

*International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020. → page 71

[113] N.-E. E. Faouzi, H. Leung, and A. Kurian, "Data fusion in intelligent transportation systems: Progress and challenges – a survey," *Information Fusion*, vol. 12, no. 1, pp. 4–10, 2011, special Issue on Intelligent Transportation Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253510000643 → page 71

[114] Y. Wu, H. Tan, L. Qin, B. Ran, and Z. Jiang, "A hybrid deep learning based traffic flow prediction method and its understanding," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 166–180, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X18302651 → page 71

[115] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X15000935 → pages 89, 95

[116] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015. → page 95

[117] M. J. Islam, A. Sharma, and H. Rajan, "A cyberinfrastructure for bigdata transportation engineering," 04 2018. → page 98

[118] Y. Adu-Gyamfi, "Gpu-enabled visual analytics framework for big transportation datasets," pp. 147–159, 12 2019. → page 98

[119] A. Waibel and C. Fugen, "Spoken language translation," *IEEE Signal Processing Magazine*, vol. 25, no. 3, pp. 70–79, 2008. → page 99

[120] L. Besacier, E. Barnard, A. Karpov, and T. Schultz, "Automatic speech recognition for under-resourced languages: A survey," *Speech Communication*, vol. 56, p. 85–100, 01 2014. → page 99

[121] S. Kumar, A. Kumar, P. Mitra, and G. Sundaram, "System and methods for converting speech to sql," 08 2013. → page 99

[122] S. Jamoussi, K. Smaïli, and J.-P. Haton, "From speech to sql queries : a speech understanding system," 07 2005. → page 100

[123] L. Blunschi, C. Jossen, D. Kossmann, M. Mori, and K. Stockinger, "Soda: Generating sql for business users," *Proc. VLDB Endow.*, vol. 5, no. 10, p. 932–943, jun 2012. [Online]. Available: https://doi.org/10.14778/2336664.2336667 → page 100

[124] H. Piringer, M. Buchetics, and R. Benedik, "Alvis: Situation awareness in the surveillance of road tunnels," in *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2012, pp. 153–162. → page 100

[125] C.-T. Lu, A. Boedihardjo, and J. Zheng, "Aitvs: Advanced interactive traffic visualization system," 05 2006, pp. 167 – 167. → page 101

[126] S. Shekhar, C. Lu, R. Liu, and C. Zhou, "Cubeview: a system for traffic data visualization," in *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*, 2002, pp. 674–678. → page 101

[127] M. L. Pack, "Wide-area, web-based mobility analysis using probe data," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 2012, pp. 1682–1686. → page 101

[128] J. Yoon, B. Noble, and M. Liu, "Surface street traffic estimation," 06 2007, pp. 220–232. → page 101

[129] A. C. Diker and E. Nasibov, "Estimation of traffic congestion level via fn-dbscan algorithm by using gps data," in *2012 IV International Conference "Problems of Cybernetics and Informatics" (PCI)*, 2012, pp. 1–4. → page 103

[130] Y. Zhang, S. Wang, B. Chen, and J. Cao, "Gcgan: Generative adversarial nets with graph cnn for network-scale traffic prediction," 07 2019, pp. 1–8. → page 103

[131] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. van de Wetering, "Visual traffic jam analysis based on trajectory data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, pp. 2159–2168, 2013. → page 103

[132] N. Willems, H. v. d. Wetering, and J. J. v. Wijk, "Visualization of Vessel Movements," *Computer Graphics Forum*, 2009. → page 103

[133] M. L. Pack, K. Wongsuphasawat, M. VanDaniker, and D. Filippova, "Ice–visual analytics for transportation incident datasets," in *2009 IEEE International Conference on Information Reuse  Integration*, 2009, pp. 200–205. → page 103

[134] A. Khotanzad and E. Zink, "Color paper map segmentation using eigenvector line-fitting," in *Proceeding of Southwest Symposium on Image Analysis and Interpretation*, 1996, pp. 190–194. → page 103

[135] A. Anwar, T. Nagel, and C. Ratti, "Traffic origins: A simple visualization technique to support traffic incident analysis," in *2014 IEEE Pacific Visualization Symposium*, 2014, pp. 316–319. → page 104

[136] R. S. R. Package. [Online]. Available: https://www.npmjs.com/package/react-speech-recognition → page 105

# Appendix A

# Publications

1. **M. Shoman,** A. Aboah, Y. Adu-Gyamfi, "Deep Learning Framework for Predicting Bus Delays on Multiple Routes Using Heterogenous Datasets," *Journal of Big Data Analytics in Transportation*, vol. 2, pp. 275–290, 2020.
2. **M. Shoman**, A. Aboah, V. Mandal, S. Davami, Y. Adu-Gyamfi, A. Sharma, "A Vision-based System for Traffic Anomaly Detection using Deep Learning and Decision Trees," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4207-4212, 2021.
3. **M. Shoman**, M. Amo-Boateng, Y. Adu-Gyamfi, , "Multi-purpose, Multi-Step Deep Learning Framework for Network-Level Traffic Flow Prediction," *Advances in Data Science and Adaptive Analysis*, pp. 2250, 2022.
4. **M. Shoman**, A. Aboah, V. Mandal, S. Davami, Y. Adu-Gyamfi, A. Sharma, "A Region-Based Deep Learning Approach to Automated Retail Checkout," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3210-3215, 2022.
5. M. Amo-Boateng, **M. Shoman**, Y. Adu-Gyamfi, , "Accelerating Statewide Con-

nected Vehicles Big (Sensor Fusion) Data ETL Pipelines on GPUs," *Transportation Research Board* , 2023.

6. **M. Shoman,** A. Aboah, A. Daud, Y. Adu-Gyamfi, "GC-GRU-N for Traffic Forecasting using Loop Detector Data" [submitted]

# Vita

Maged Shoman's research interests include intelligent transportation systems, deep learning, computer vision and data science. He recieved the B.Sc. degree in Civil Engineering from the American University of Sharjah (AUS), Sharjah, UAE in 2015, and the M.Sc. degree in Environmental Engineering from the Technical University of Munich (TUM), Munich, Germany, in 2019. As a graduate teaching and research assistant, Maged was responsible for conducting state-of-the-art research and publishing his findings in high-impact journals and conference proceedings. During his tenure at MU, Maged recieved the 2020 CEE Fellowship and 2021 CEE Outstanding PhD Student Award.