Urban Traffic Congestion Propagation Prediction Model: A case of non-recurrent congestion

Ansari Esfe, Reza

http://hdl.handle.net/1880/113606
master thesis

University of Calgary graduate students retain copyright ownership and moral rights for their thesis. You may use this material in any way that is permitted by the Copyright Act or through licensing that has been assigned to the document. For uses that are not allowable under copyright legislation or licensing, you are required to seek permission.

Downloaded from PRISM: https://prism.ucalgary.ca
Urban Traffic Congestion Propagation Prediction Model: A case of non-recurrent congestion

by

Reza Ansari Esfe

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN CIVIL ENGINEERING

CALGARY, ALBERTA

JUNE, 2021

© Reza Ansari Esfe 2021
ABSTRACT

In this thesis, a novel congestion propagation prediction framework is presented. This prediction framework can be used by navigation systems to mitigate congestion during collisions. In the first part of this thesis, a new deep neural network model with four learning layers is introduced to predict speeds of road segments in the vicinity of a collision. This prediction model uses a graph of connectivity information of road segments, recent speed information, historical speed, and weather information to predict speed. This model is tested on all collisions between 2014 to 2019 in downtown Calgary. In the second part of the thesis, a congestion quantification algorithm is developed to identify whether the given road segments are congested or not and to determine the congestion effect caused by recurring and non-recurring events. This model utilizes the speed prediction from the first section of the thesis to predict which road segments become congested over time. While the previous studies are not able to predict congestion propagation during non-recurring events, this study predicts both the growth and recovery of congestion during non-recurring events. A case study was conducted in downtown Calgary to show how the model predicted congestion propagation during collisions. The results of this study showed that congestion propagated not only downstream but in all directions in a network. However, congestion intensity depended on the directions of the road segments subjected to the accident compared to other road segments. In other words, if a collision happened in the vertical stream, the road segments in the vertical stream were more affected than the road segments in the horizontal stream.
ACKNOWLEDGMENT

I would like to show my gratitude and appreciation to my supervisor, Dr. Lina Kattan, for her valuable guidance and support during my studies at the University of Calgary. It has been an honor to work with Dr. Kattan.

I would also like to express my warm thanks to my dear friend and mentor, Dr. Mohammad Ansari Esfe. This study could not have been accomplished without his support.

I would like to thank INRIX and the City of Calgary for providing data because it would not have been possible to do this research without their contributions. In addition, I would like to thank NSERC for funding this project.

I want to thank my group members at AMA Active Traffic and Demand Management Laboratory for being supportive in the day-to-day life in the office.

And finally, I want to thank my best friend, my supporter, and my better half, Negar, helped me to complete this chapter of my life.
# TABLE OF CONTENTS

ABSTRACT ................................................................................................................................. ii

ACKNOWLEDGMENT .................................................................................................................. iii

TABLE OF CONTENTS ............................................................................................................... iv

LIST OF TABLES .......................................................................................................................... viii

LIST OF FIGURES ....................................................................................................................... ix

LIST OF ACRONYMS .................................................................................................................. xi

1. INTRODUCTION ..................................................................................................................... 1
   1.1 Overview .......................................................................................................................... 1
   1.2 Problem statement ......................................................................................................... 2
   1.3 Thesis objectives .......................................................................................................... 4
   1.4 Organization of thesis ................................................................................................. 7

2. LITERATURE REVIEW ........................................................................................................... 8
   2.1 Analysis of the evolution of traffic dynamics during collisions ...................................... 8
   2.2 Incident detection ....................................................................................................... 9
   2.3 Congestion propagation ............................................................................................ 14
      2.3.1 Recurring (daily or frequent) congestion propagation prediction ............................ 14
      2.3.2 Non-recurring (collision) congestion propagation prediction ................................. 16
      2.3.3 Visualization of congestion formation dynamics .................................................. 19
   2.4 Traffic prediction ....................................................................................................... 20
      2.4.1 Deep neural network models in traffic predictions .............................................. 22
   2.5 Summary of the literature .......................................................................................... 24

3. ARTIFICIAL NEURAL NETWORK ALGORITHM (ANN) ..................................................... 26
   3.1 Linear regression and logistic regression ...................................................................... 26
   3.2 Fully Connected Neural Networks (Feed Forward Neural Networks) ......................... 28
3.3 Recurring Neural Networks (RNNs) ........................................................................... 29
3.4 Convolution Neural Networks (CNNs) ......................................................................... 33
3.5 Graph Convolution Network (GCN) .............................................................................. 34

4. A SPATIOTEMPORAL FRAMEWORK for TRAFFIC SPEED PREDICTION .................. 38
4.1 Congestion propagation prediction solution steps .......................................................... 38
4.2 Build impact area graph (graph G)................................................................................ 39
4.3 Speed prediction model ................................................................................................. 41
4.3.1 The combined graph convolution network (GCN) and long short-term memory (LSTM) model structure ................................................................. 43
4.3.1.1 Spatial learning layer ......................................................................................... 46
4.3.1.2 Temporal learning layer .................................................................................... 47
4.3.1.3 Periodicity learning layer .................................................................................. 49
4.3.1.4 Context learning layer ....................................................................................... 50
4.3.1.5 Concatenation process ....................................................................................... 50
4.4 Application of the model to a case study for downtown Calgary, AB ......................... 51
4.4.1 Data description ........................................................................................................ 51
4.4.1.1 Speed data ......................................................................................................... 51
4.4.1.2 Spatial data ........................................................................................................ 53
4.4.1.3 Collision data .................................................................................................... 53
4.4.2 Data processing ......................................................................................................... 54
4.4.2.1 Geospatial dataset (calculating adjacency matrix) ........................................... 54
4.4.2.2 Temporal dataset .............................................................................................. 55
4.4.2.3 Periodicity dataset ........................................................................................... 57
4.4.2.4 Weather dataset ............................................................................................... 57
4.4.3 Model setting for training ......................................................................................... 59
4.4.4 Testing the Model........................................................................................................... 60
4.4.4.1 Case study ....................................................................................................................... 61
4.4.4.2 Model validation ............................................................................................................. 65

5. USING A CONGESTION QUANTIFICATION FRAMEWORK to model NETWORK CONGESTION...................................................................................................................... 67

5.1 Introduction ............................................................................................................................. 67
5.1.1 Binary congestion detection ............................................................................................... 67
5.1.2 New congestion quantification index .................................................................................. 71
5.2 Congestion value calculation ................................................................................................. 73
5.3 Hypothesis testing ................................................................................................................... 75
5.4 Congestion Quantification Algorithm..................................................................................... 78
5.4.1 $\lambda$ and CQA .................................................................................................................... 81
5.4.2 $\alpha$ and CQA....................................................................................................................... 84
5.5 Case Study ............................................................................................................................. 84
5.5.1 Calculating lower bound speed using hypothesis testing ....................................................... 86
5.5.2 Analyzing congestion with CQA......................................................................................... 89
5.5.3 Visualizing congestion propagation prediction .................................................................... 92

6. SUMMARY AND CONCLUSION .............................................................................................. 97
6.1 A spatiotemporal framework to predict traffic speed ............................................................... 97
6.2 Congestion quantification algorithm to model network congestion ...................................... 98
6.3 Research contributions .......................................................................................................... 98
6.4 Limitations and future work .................................................................................................. 99

REFERENCES ............................................................................................................................ 101

APPENDIX .................................................................................................................................... 110
A.1 Training the neural network .................................................................................................. 110
A.1.1 Loss function ........................................................................................................ 111
A.1.1.1 Mean absolute error (MAE) ........................................................................ 111
A.1.1.2 Mean absolute percentage error (MAPE) .................................................... 112
A.1.1.3 Mean squared error (MSE) ......................................................................... 112
A.1.1.4 Root mean squared error (RMSE) ............................................................... 112
A.1.2 Backpropagation (BP) algorithm ........................................................................ 113
A.1.3 Optimization algorithms .................................................................................... 115
LIST OF TABLES

Table 4-1 Process to build the impact area graph (graph G) ................................................................. 40
Table 4-2: Model error measures ........................................................................................................... 65
Table 5-1 Congestion detection methods ............................................................................................... 72
Table 5-2. Algorithm to identify the effect of collisions ......................................................................... 78
Table 5-3 The remaining traffic congestion percentages after passing the half_life ......................... 81
LIST OF FIGURES

Figure 1-1 Congestion propagation and clear path ................................................................. 3
Figure 1-2 Congestion propagation prediction solution steps ............................................... 6
Figure 2-1 Timeline of a traffic incident (Amer et al., 2013) ............................................. 9
Figure 2-2 Congestion propagation modeling (Pan et al., 2013) ........................................ 18
Figure 2-3 Speed prediction models Cui et al. (2020) ....................................................... 22
Figure 3-1: Linear regression model ...................................................................................... 27
Figure 3-2 Fully connected neural networks ......................................................................... 28
Figure 3-3 RNN example ....................................................................................................... 30
Figure 3-4 LSTM internal structure (Yuan et al., 2019) ...................................................... 31
Figure 3-5 Convolution neural network (CNN) structure .................................................... 34
Figure 4-1 Speed forecasting solution steps .......................................................................... 39
Figure 4-2 Deep neural network speed prediction model structure .................................... 45
Figure 4-3 Downtown Calgary road segments ......................................................................... 51
Figure 4-4 Sample of Raw INRIX traffic data ....................................................................... 52
Figure 4-5 Physical Road Information (INRIX) ...................................................................... 53
Figure 4-6 Collision Raw Data ................................................................................................ 54
Figure 4-7 Time series sliding window – learning intervals are red and tested intervals are green .......................................................................................................................... 56
Figure 4-8 Processing data and assigning speed data, weather data, periodicity data, and spatial data (adjacency matrix) to graph G ........................................................................ 58
Figure 4-9 Impact area for collision #1 (graph G) .................................................................. 61
Figure 4-10 Speed prediction results for segments in the collision #1 network .................... 64
Figure 5-1 First level upstream speed during collision.......................................................... 68
Figure 5-2 First level downstream speed during collision......................................................... 69
Figure 5-3 Threshold approach for first level upstream: 0 represents no congestion, and 1
represents congestion ............................................................................................................ 70
Figure 5-4 Hypothesis testing .................................................................................................. 76
Figure 5-5 Sensitivity analysis for different decay constant rates ........................................... 83
Figure 5-6 Accident impact area collision #1 ................................................................. 85
Figure 5-7. Hypothesis testing for five road segment samples in collision #1 ...................... 87
Figure 5-8 CQA testing on five road segments collision #1.................................................. 91
Figure 5-9 Congestion propagation of collision #1 from 12:00 to 13:35; green means the road
segment is not affected by the collision; yellow means mildly affected; orange means moderately
affected; and red means highly affected. .............................................................................. 95
# LIST OF ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>Akaike Information Criterion</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>ARIMA</td>
<td>Autoregressive Integrated Moving Average</td>
</tr>
<tr>
<td>BP</td>
<td>Back Propagation</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolution Neural Network</td>
</tr>
<tr>
<td>CQA</td>
<td>Congestion Quantification Algorithm</td>
</tr>
<tr>
<td>DBN</td>
<td>Dynamic Bayesian Network</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Network</td>
</tr>
<tr>
<td>FCN</td>
<td>Fully Connected Neural Network</td>
</tr>
<tr>
<td>FNN</td>
<td>Feedforward Neural Network</td>
</tr>
<tr>
<td>GAN</td>
<td>Generative Adversarial Network</td>
</tr>
<tr>
<td>GCN</td>
<td>Graph Convolution Network</td>
</tr>
<tr>
<td>GRU</td>
<td>Gated Recurrent Unit</td>
</tr>
<tr>
<td>IDA</td>
<td>Incident Detection Algorithm</td>
</tr>
<tr>
<td>KNN</td>
<td>(k)th Nearest Neighbor</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short-Term Memory</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>MAPE</td>
<td>Mean Absolute Percentage Error</td>
</tr>
<tr>
<td>MLF</td>
<td>Multi-Layer Feedforward</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>Pro_Graph</td>
<td>Propagation Graph</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Squared Error</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>STC</td>
<td>Spatiotemporal Congestion</td>
</tr>
<tr>
<td>STCCP</td>
<td>Spatiotemporal Co-location Congestion Pattern</td>
</tr>
<tr>
<td>STOTree</td>
<td>Spatiotemporal Outlier Tree</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

1.1 Overview

Transportation systems are the road map to the community. Roads move people from where they are to where they are going, connecting them to their family and institutions. However, every day traffic collisions and the resulting congestion result in fatalities, severe injuries, air pollution, and they waste energy and lower productivity. Motor vehicle crashes cost Canada almost CAD20.62 billion in 2010 (Canada Department of Transportation, 2011). A popular definition of congestion is when demand exceeds a road networks’ capacity (Aftabuzzaman et al., 2007). Congestion can be described based on three macroscopic traffic flow parameters: flow, speed, and density.

In an urban network when a major collision occurs, a bottleneck starts forming on the affected road segment and propagates to adjacent road segments, possibly blocking intersections and forming gridlocks. The impact of a collision is thus not limited to only the upstream and downstream road segments, but rather it disperses and forms an impact area in the vicinity of the link subject to the incident. The impact area can be defined as the network of roads in the vicinity of the link subjected to a collision where travel times are significantly affected by the collision. Short-term prediction of the congestion propagation in the impact area can provide a valuable tool that can be used to alleviate proactively gridlock formations by applying real-time remedial traffic responses, such as rerouting, providing advanced traveler information, and responsive signal control.
1.2 Problem statement

Congestion can be defined as recurring or non-recurring. Recurring congestion is periodic congestion formation that can predictably occur on a given day and time. Instances of recurring events include morning and evening rush hour traffic. Non-recurring congestion is caused by unpredictable events such as collisions, extreme weather, and disruptions due to construction sites. Therefore, non-recurring events need short term or no notice planning compared to recurring events.

An average of 1.35 million people die in road crashes annually in the world, which is an average of 3,700 deaths per day, making it the eighth leading cause of death in the world (World Health Organization, 2018). As a non-recurring event, collisions that result in severe injuries can cause death when emergency personnel cannot respond in a timely fashion. Thus, mitigating congestion and managing the timely arrival of emergency personnel to the collision scene is important. In addition, non-recurrent congestion, such as vehicular collisions and congestion due to construction and maintenance activities, account for 13%-30% of total traffic delay (Canada Department of Transportation, 2011). Mitigating congestion caused by non-recurring events is also important to significantly reduce traffic delays, fuel consumption, and emissions. Therefore, congestion propagation patterns should be modeled and predicted.

Predicting travel information is essential for building a robust traffic assignment model (Cascetta, 2009, Jia et al. 2016 and James (2021). Cascetta (2001) states that developing advanced traveler information requires assumptions on the information type and its strategy. Travel information can be made available before starting the trip (i.e., pre-trip information), during the trip (i.e., en-route information) or both en-route and pre-trip. This information can be: 1) historical based on all
previous time system states with similar characteristics (e.g., time of the day, day of the week, weather condition), 2) real-time on current traffic states, and 3) predictive of future traffic states.

To further illustrate this problem, a scenario is presented in Figure 1-1. The figure depicts a collision that has occurred on the given road segment. Current navigation systems, such as google, detect the location of the collision and inform vehicles passing this road to avoid congestion by proposing a re-route plan, as shown in Figure 1-1 (a). However, congestion propagates in all directions over time. By the time a vehicle that is informed about the collision approaches the rerouting turning point (Figure 1-1 (b), the blue triangle), the vehicle will still experience delays because the resulting congestion has propagated to the turning point as depicted in Figure 1-1 (b) with the black circle. If the navigation system had accurately predicted the congestion over the next few minutes and proposed an alternative route based on a prediction as shown in Figure 1-1 (c), the car would not be stuck in traffic. This congestion prediction process enables the navigation system to show a clear path for vehicles passing a collision area. Thus, the objective of this thesis is to predict 1)
how congestion from collisions propagates over time, 2) which roads become congested, and 3) which roads recover from congestion.

While several studies in the literature outline ways to predict congestion propagation, most of them focus on recurring congestion on freeways. To the best of my knowledge, no study examined the problem of congestion propagation during non-recurring events on other types of roads, such as arterials; the majority of the research on non-recurring congestion detection has focused on estimating the spatiotemporal impact of collisions on freeways. More precisely, the studies focus on examining the impact of congestion on the upstream links along the same corridor.

This thesis presents several algorithms to understand and predict traffic propagation patterns that form as the result of a collision. The analysis uses three types of input information that is derived from three datasets: 1) observed real-time speed, 2) reported collisions, and 3) weather information. Deep learning techniques are used to capture the resulting complex traffic dynamics of congestion propagation and predict the short-term speed. Incidents are first detected when the observed traffic data in real-time deviates significantly from its historical traffic patterns. Then, congestion propagation patterns are predicted at the network level to identify the impact area. Accordingly, the resulting congestion effects are quantified.

1.3 Thesis objectives

The focus of this thesis is to develop a set of algorithms that can capture the spatial and temporal propagation of congestion resulting from a collision. The following research objectives are identified to achieve this goal:
1) Develop a deep learning network (DLN) based on a speed prediction model by considering the temporal and spatial interrelationships among road segments in a network.

2) Implement a new congestion quantification algorithm to investigate congestion patterns in recurring and non-recurring events.

3) Test and validate the proposed model using collision data from downtown Calgary to capture the propagation of congestion at the network level.

1.3.1 Overview of the methodology

To solve the congestion propagation prediction problem, the methodology used in this study is divided into two problems as shown in Figure 1-2: 1) the speed forecasting problem and 2) the congestion quantification problem. The speed forecasting problem predicts speed for road segments surrounding a collision during the collision time. To solve the problem, the time and location of the collision is detected. Then, a graph that represents the network of road segments in the vicinity of the collision is constructed. Next, speed data, adjacency matrix, weather data, and periodicity data (the historical speed corresponding to the speed predicted by the model) are extracted from datasets. In the next step, a spatiotemporal deep neural network model is developed by combining the graph convolution network (GCN) and long short-term memory (LSTM) layer to predict speed. Then, the model is trained using data from one year before the time of collision. In the final step, the speed is predicted for all road segments in graph G during the collision. The speed prediction results from the forecasting model are then used in the congestion quantification problem. The congestion quantification problem quantifies a congestion value and distinguishes recurring congestion from non-recurring congestion to identify which road segments will be affected by a collision. In addition, the proposed algorithm identifies the dynamics of a congestion (i.e., bottleneck formation or dissipation), thereby detecting the time when the congestion
propagation starts, reaches its maximum level, and completely dissipates. The congestion quantification algorithm relies on the density and speed changing during the collision. In addition, the congestion quantification algorithm utilizes statistical hypothesis testing to distinguish between non-recurring and recurring congestion, which makes the model capable of identifying congestion resulting from collisions. Combining these two problems enables us to predict how the effects of congestion, caused by a collision, propagate throughout the impact area.

Figure 1-2 Congestion propagation prediction solution steps
1.4 Organization of thesis

This thesis comprises five chapters as follows:

Chapter Two includes a literature survey on different algorithms that have been developed to detect collisions in road networks; the chapter also discusses the gaps that exist in investigating congestion propagation patterns for non-recurring events. Moreover, different approaches to modeling congestion propagation during recurring events are reviewed. Finally, studies related to traffic prediction are discussed.

Chapter Three and Four provides a description of the deep neural network implementation process. First, artificial neural network (ANN) modeling and training are discussed in chapter three. These models are utilized in developing the deep neural network’s aim to predict traffic speeds in chapter four. Secondly, the model’s structure is discussed in detail to clarify how its various components help to predict speed. Finally, a case study is performed to assess the performance of the proposed model to predict the road speeds at the network level.

Chapter Five introduces several methods to model congestion patterns. In addition, a new congestion quantification algorithm is proposed. This algorithm quantifies congestion and distinguishes recurring congestion from non-recurring congestion resulting from collisions. A case study is then conducted to estimate the time when each road segment becomes congested due to a collision. Finally, a map is presented to show the spatial and temporal propagation patterns of congestion.

Chapter Six is the last chapter of this thesis, and it outlines the major contributions and suggests future research directions.
2. LITERATURE REVIEW

This chapter introduces the phases of collisions and then reviews previous studies on three main interconnected topics: incident detection, congestion propagation, and traffic prediction. Since the focus of this thesis is on arterial networks, the first section reviews the incident detection algorithms (IDAs) that have been developed specifically for arterial road networks. The second section reviews previous studies on predicting congestion propagation for both recurring and non-recurring events. Finally, the third section reviews the deep neural network models that have been developed to predict future traffic states.

2.1 Analysis of the evolution of traffic dynamics during collisions

Figure 2-1 in Amer et al. (2013) divides the collision timeline into several phases, and this timeline defines the time between the incident occurrence time \(T_0\) and flow recovery time \(T_7\), which is the time to return to normal flow. Only two of phases of the timeline are discussed because they are more relevant to the work conducted in this thesis:

1) Incident detection: in traffic incident reports, the reported incident time of an incident \(T_1\) is usually recorded instead of the incident occurrence time. Early incident detection is important because it helps emergency systems to respond in a timely manner to reduce injury levels and fatality rates. In addition, early detection is crucial for managing and controlling traffic so that conditions can return to the before collision state.

2) Congestion propagation: propagation is the collision effect that disseminates from the road segment subject to collision to adjacent segments in the vicinity. Congestion propagation starts from the incident occurrence time \(T_0\) and continues until normal flow recovery time \(T_7\). In reporting collisions, either roadway clearance time \(T_5\) or incident clearance time \(T_6\) is usually
reported as the end time of an incident. However, even after the removal of an incident, the shockwave that results from the collision continues to propagate and congestion continues for a while. Therefore, the time between the reported start and end times does not fully represent the temporal impact of a given incident even if the incident has been removed.

![Timeline of a traffic incident](image)

Figure 2-1 Timeline of a traffic incident (Amer et al., 2013)

### 2.2 Incident detection

Anomalies in traffic flow can be caused by recurring or non-recurring events. Recurring events occur periodically and result from variations of vehicular demand in urban road networks, such as the congestion that occurs during morning or evening peak periods. Non-recurring events, however, are outcomes of unusual events caused by sudden disruptions in road networks. The possible sources of these non-recurring events can be either predictable or unexpected. For instance, scheduled events like roadworks, sport events, or concerts have an anticipated effect on the traffic of urban networks. In contrast, collisions or other disruptions (e.g., extreme weather)
differ from scheduled events because they happen abruptly and need short-term planning. Thus, it is beneficial to detect incidents quickly to prevent further disruption.

Incident detection algorithms (IDAs) have been developed to identify such unexpected events in road networks. While most studies have focused on using IDAs on highways, only a few studies apply IDAs in arterial road networks. Ahmed and Hawas (2012) stated that studying collisions in arterial road networks was challenging due to the complex topologies of arterial networks and because signals periodically interrupted traffic flow in arterials. Somerset intelligence (2017) found that 37% of fatalities in Somerset, England happened in arterial road networks, which highlighted the importance of studying collisions at the network level. Evans et al. (2020) divided IDAs for urban road networks into six groups: comparative algorithms, time series, machine learning techniques, image processing, social media, and data concatenation approaches. The first three algorithms rely on traffic data such as occupancy, flow, and speed data extracted from loop detectors, GPS, and Wi-Fi devices. The last three algorithms, which focus on image and video processing and semantic analyses of social media texts (e.g., Twitter), need more processing time and effort to convert the information into useful traffic data. In this study, only the first three groups of IDAs, implemented in arterial roads, are reviewed.

Comparative algorithms find the difference between real-time traffic data and historical traffic data (as threshold) to detect anomaly. Such traffic data (i.e., both historical and real-time data) are extracted from one single detector or a pair of detectors placed a short distance apart. Detection alerts occur when traffic data, such as occupancy, flow, or speed, reasonably differ from a preset threshold. For instance, Anbaroglu et al. (2014) used a preset threshold for travel time, and Bowers et al. (1996) used averaged traffic data as the threshold for each specific detector. Bowers et al. (1996) took the average of historical data by giving more weight to recent data. The authors
compared the real-time data with the calculated threshold and raised alerts when the flow was lower than the average flow and occupancy was greater than the threshold. While Bower’s algorithm utilized a pair of detectors to raise alerts, Cherrett et al. (2000) implemented a single-detector algorithm. Although Cherrett’s (2000) approach was less accurate than the two-detector algorithm, it could still be used in areas that have fewer detectors. Oskarbski et al. (2016) modified the existing highway IDA so that the IDA could be used in urban arterial networks. They extracted travel time from Wi-Fi and Bluetooth devices on vehicles, cyclists, and pedestrians. Travel time was defined as the time taken for a group of devices to pass between two set points in a network as measured by Bluetooth and Wi-Fi scanners. Then, historical reference values were calculated based on each mode of transportation (e.g. bicycle, bus, car) and time variables such as time of day and day of week. Finally, this algorithm raised alerts for incidents when real-time travel times were substantially different from the historical reference values. In addition, several studies compared real-time speed data with immediate past data to detect incidents. For instance, Li et al. (2013) utilized standard deviations and weighted averages of immediate recent traffic conditions as a threshold. They compared this threshold with real-time traffic conditions to raise alerts of incident occurrences. Chen et al. (2016), Zhang et al. (2016), and Chakraborty et al. (2019) developed a data-driven IDA that learned patterns of normal traffic conditions. They compared real-time traffic condition information with estimated patterns to detect anomalies. Although comparative algorithms are simple, they provide promising results in incident detection in arterial road networks. Most comparative algorithms use averaged historical traffic data, such as speed, occupancy, or flow, to define thresholds. However, average traffic data is not a good measure to determine the effects of collisions that result from daily traffic congestion on arterial roads since signalized intersections easily exceed the thresholds, which results in false alerts.
Time series algorithms suggest that traffic data follow a predictable pattern over a short period of incident-free time, but this pattern changes during incidents. Time series algorithms use historical traffic data to predict near future traffic conditions. If the expected values are sufficiently different from the real-time observed values, the IDA algorithm raises an incident detection alert. Time series algorithms differ from comparative algorithms because they use a time-dependent threshold. This threshold varies based on recent local traffic conditions. Thancanamootoo et al. (1998) used flow and occupancy data from downstream and upstream detectors to identify incidents. They developed an exponential smoothing scheme to determine the thresholds for flow and occupancy for each detector. Therefore, alerts were raised when occupancy and flow fell below their respective thresholds on the downstream detector and flow fell below and occupancy rose above their thresholds on the upstream detector. Lee and Taylor (1999) implemented a Kalman filter algorithm to find sudden fluctuations in traffic conditions. They applied linear Kalman filtering to traffic variables, such as speed and flow, to predict values and their upper and lower bounds. Then, for any real-time values that were outside the upper and lower boundaries, the algorithm raised an incident alert. Although the time series approach performs well, it is unable to identify anomalies that occur due to recurring or non-recurring events because the approach uses recent traffic variables as inputs.

Machine learning (ML) algorithms are a type of artificial intelligence, and they enable the training of datasets and test the accuracy of predictions. Most of these algorithms use macroscopic traffic parameters, such as occupancy, flow, and speed for several time intervals, as inputs to predict incident and non-incident states. Khan and Ritchie (1998) and K. Zhang and Taylor (2006) applied Bayesian belief networks to both highways and urban arterials. Machine learning algorithms enabled the calibration of the incident detection model based on prior knowledge instead of using
historical data. First, the algorithm mapped occupancy and flow to three groups (low, medium, high). Then, the Bayesian network was applied to the processed data. This method obtained 88% accuracy and gave false alerts 0.62% of the time. However, their Bayesian network used simulation data, which made the model outcomes less reliable. Yang et al. (2009) developed the first support vector machine (SVM) based on an IDA. This algorithm used traffic data, such as occupancy and flow from fixed downstream and upstream detectors and probe speed data, to classify incident and non-incident states in arterial road networks. Ghosh and Smith (2014) modified four machine learning-based highway IDAs to implement in urban roads network. They used a multi-layered feedforward (MLF) neural network, probabilistic neural network, SVM, and fuzzy-wavelet radial basis function neural network to detect incidents in signalized junction networks using simulated data. They found that the SVM performed better compared to the others. However, MLF was far less computationally intensive and easier to implement. Thus, they recommended the MLF algorithm as the most suitable choice. While the conventional MLF was successful for freeways, Zhu et al. (2018) developed a convolution neural network (CNN) model to detect incidents in urban networks. They used traffic flow data and incident data from London, UK, and their results were more accurate in terms of detecting incidents, especially in a large urban networks, compared to MLF models. Xiao (2019) developed an ensemble learning model to detect incidents on highways. While Xiao proposed individual SVM and k-nearest neighbor (KNN) models, he also combined these two methods to improve accuracy. Lin et al. (2020) developed an automated IDA using generative adversarial networks (GANs) to address the scarcity of incident data samples. They extracted spatiotemporal features using the random forest algorithm. Afterward, they generated a new sample using GANs. Finally, they developed an SVM classifier to detect incidents. While ML algorithms can be used to detect incidents, ML do not determine the effects
of congestion at the road segment level. To find the effects of congestion caused by collisions (non-recurring events) at the road segment level, a model needs to distinguish non-recurring from recurring congestion for each road segment.

2.3 Congestion propagation

Congestion propagation can be defined as the effect of a collision transferring over time from a particular road segment to adjacent segments. This propagation can occur in all directions including upstream and downstream. Congestion propagation algorithms can be classified based on their scope (i.e., single-corridor or network), event type (i.e., recurring or non-recurring), and goal (i.e., modeling, pattern identification, or prediction). In addition, congestion propagation has two phases (Figure 2-1): i) the time when congestion starts growing through the network and ii) the time when congestion starts dissipating and congestion on road segments reduces to initial conditions. None of the previous studies modeled the second phase of congestion (dissipation).

2.3.1 Recurring (daily or frequent) congestion propagation prediction

Congestion propagation during recurring events has been well studied and discussed in the literature. These studies mainly focused on finding the prevailing pattern in congestion propagation. For instance, Liu et al. (2011) developed an algorithm to find a casual pattern for traffic conditions. They divided the city of Beijing into several regions and graphed these regions. In the graph, nodes represented the regions, and edges depicted traffic flow between the regions. They proposed using the spatiotemporal outlier tree (STOTree) and a frequent subtree algorithm to reveal the casual anomaly pattern in road networks. Nguyen et al. (2016) introduced a solution
to detect spatiotemporal congested roads and causal relationships among roads. They developed the spatiotemporal congestion (STC) algorithm, which generated the most frequent sub-structures (subtrees) from all discoverable tree structures in a network to reveal the recurring propagation pattern. In addition, they used a dynamic Bayesian network (DBN) to estimate the probability of each propagation occurring. Combining these two approaches (STC_DBN) enabled them to discover effectively the congestion propagation pattern. Liang et al. (2017) presented a data-driven approach that captured the cascading patterns of traffic propagation by maximizing the likelihood function from the available data. They claimed that this model outperformed the STC_DBN algorithm in terms of accuracy and computational time. Later, He et al. (2018) improved the STC_DBN algorithms. They proposed a spatiotemporal congestion co-location pattern (STCCP) to discover the congestion propagation pattern. They built three-dimensional instances using the features of time, location, and congestion. Using the congestion features located in adjacent spaces and subsequent times, they mined the congestion pattern.

While the above studies mainly focused on determining the frequent congestion pattern at the network level using a tree-based algorithm, they were not able to predict congestion propagation. L. Wang and Zhou (2017) and X. Ji et al. (2019) used a mining algorithm to discover the spatiotemporal congestion propagation pattern. Further, they defined speed features based on taxi trajectory data to extract the spatiotemporal congested areas to build the frequent patterns. Combining the frequent congestion patterns with the congestion propagation rules enables the researchers to predict congestion propagation in recurring events. While recent studies focused on developing frequent trees or finding the most probable congestion propagation pattern to predict congestion propagation, Xiong et al. (2018) proposed using an efficient algorithm to predict where congestion propagates in the near future. They proposed the concept of propagation graph (Pro-
Graph) to model congestion propagation direction in networks. At each time interval, they predicted every Pro-Graph, which was discoverable in the network using empirical probabilities of propagation calculated based on historical data. Xiong et al. (2018) is one of the pioneering works in congestion propagation prediction. However, similar to previous works, they did not evaluate this model for non-recurring events.

More recently, Majumdar et al. (2021) utilized the LSTM networks with univariate data (speed) and multivariate data (speed and flow) to predict congestion propagation throughout road networks. First, they predicted vehicular speed for two sensor sites. Then, they compared the speed patterns to reveal similarity in the speed profile of both sites. The time lags between similar anomalies, such as a drop or increase in speed, were calculated. Time lags were then used to estimate the congestion propagation time. While this study predicts congestion propagation, this model cannot be used for arterial roads because arterials have signalized intersections. In addition, although this study models congestion propagation at the network level, it is not applicable to non-recurring events. Therefore, this study only considers congestion propagation for recurring events.

### 2.3.2 Non-recurring (collision) congestion propagation prediction

While researchers have studied congestion propagation during recurring events for various scopes (i.e., single-corridor or network), the research on congestion propagation during non-recurring events has not been extensively studied, and most existing studies have focused on single corridor (highways). In fact, traffic dynamics in urban networks and interactions among parking, adjacent land uses, transit, and non-motorized traffic make the analysis more complex. In addition, analyzing congestion propagation in arterial road networks requires congestion algorithms that can distinguish recurring from non-recurring congestion, which is very challenging because arterials have signalized intersections that periodically interrupt traffic. Therefore, almost all studies focus
on single corridor. For instance, Chiabaut and Faitout (2021) proposed a new method to predict congestion propagation and travel time prediction on highways. They used two different clustering methods, Gaussian mixture and k-means, to categorize traffic conditions and travel time in several groups. Their results showed that traffic conditions and travel time of the same group were similar. Real-time travel time and traffic condition data were then compared with the clustered groups to detect the most similar group to predict congestion propagation and travel time. This approach is fast and simple; however, it cannot consider congestion at the network level and on arterial roads. Pan et al. (2013) developed an algorithm to predict the spatiotemporal impact of incidents in urban networks. The authors’ goal was to find a clear path to help drivers avoid congested areas. The prediction algorithm determined the quantitative incident impact based on the time and location of the road segments. In the first step, they modeled the effect of each incident on road segments using the speed change ratio function and affected threshold, which are expressed in equations (2.1) and (2.2), respectively:

\[ \Delta v(l,t) = \frac{v_r(l,t) - v_c(l,t)}{v_r(l,t)} \]  \hspace{1cm} (2.1)

\[ \Delta v(l,t) \geq \lambda \]  \hspace{1cm} (2.2)

\( \Delta v(l,t) \): speed change ratio in time \( t \) and location \( l \)

\( l \): location

\( t \): time

\( v_c(l,t) \): current speed in time \( t \) and location \( l \)

\( v_r(l,t) \): average historical speed in time \( t \) and location \( l \)
where $\lambda$ is the impact parameter that defines the severity of speed change. Pan et al. (2013) partitioned the speed change diagram of each sensor using the $\lambda$ threshold (e.g., $\lambda = 60\%$) and generated scatter points in a 2D figure of space (the distance of a sensor from the incident location) and time for each sensor (one example is depicted in Figure 2.2a). Then, they trained the polynomial function to fit the scatter points at the onset of congestion propagation; this fitted function helped to interpolate the missing scatter points and capture the congestion propagation behavior in each time interval.

![Spatiotemporal diagram of collision propagation](a)

![Fitted result](b)

(a) Spatiotemporal diagram of collision propagation

(b) Fitted result

Figure 2-2 Congestion propagation modeling (Pan et al., 2013)

After modeling the congestion propagation for each collision occurrence, they classified collisions based on their features (e.g., time, location, and type of incidents). Therefore, collisions within the same class were trained to tune the weight in an easier way. Then, for new collisions, the model identified which class the collision belonged to and then used the weights assigned to that specific class to predict congestion propagation. While Pan et al. (2013) claimed that their model predicted congestion propagation at the network level, it only considered congestion propagation in the main
stream where congestion occurred. In other words, the model did not consider turning movements at intersections. Fei et al. (2017) developed an algorithm to predict congestion propagation boundaries caused by traffic incidents. They developed an analytical model based on kinematic wave theory and the Van Aerde model to estimate the speed of congestion propagation for each road type. They found different that propagation speeds were different depending on road type. In addition, they modeled the impact of turning movements at intersections on the propagation by considering the constant turning rate at each intersection. By estimating the speed and direction of congestion propagation, the model defined the boundaries of congestion propagation based on the time after collision. While their work predicted congestion propagation boundaries, Fei et al. (2017) could not model congestion recovery. In other words, the congestion propagation boundaries always increased over time. However, in the real world, congestion propagation stops after a period of time, and the network is restored to normal. In addition, this work, like previous other studies, did not evaluate its work on each individual road in the network; they only reported the maximum distance that the congestion reached.

2.3.3 Visualization of congestion formation dynamics

While the studies mentioned focused on finding congestion patterns to predict congestion propagation at the network level or on a single corridor, only a few focused on visualizing congestion dynamics. Liang et al. (2017) developed a visualization system to identify traffic jam propagation in road networks, and to do so, they used taxi trajectory data to find traffic jams (Ahmed & Cook, 1979). Moreover, Ji et al. (2014) developed a framework to identify hidden information in congestion formations using taxi trajectory data. They analyzed the spatiotemporal relationships between congested links to identify whether congestion propagates between two links. Then, they used the maximum connected component to capture congestion propagation at
the network level. While their approach successfully mapped congestion propagation at the network level, they did not provide a prediction framework for congestion propagation. In addition, Lee et al. (2020) and Wang et al. (2013) presented a visualization system that had the ability of congestion detection, surveillance, and prediction using vehicle detector data. Lee et al. (2020) used long short-term memory (LSTM) and embedding spatial graphs to predict congestion. In addition, they developed an algorithm to detect the causes of the congestion (i.e., when the initial road becomes congested), the direction of congestion propagation, and the severity of the congestion. While their frameworks covered a variety of congestion-related analyses, they did not provide a congestion propagation prediction framework.

2.4 Traffic prediction

With the availability of network-wide traffic data obtained from detectors, GPS, and Wi-Fi devices, researchers have access to a large database to predict changes in traffic conditions. Historical data are used to make short-term traffic predictions in a horizon from 5 to 60 minutes. These predictions can differ from the conventional time-series algorithms by considering spatial features along with temporal features. The prediction models can be classified based on the methodology and the databases used. In the traffic prediction problem, the learnable function takes previous values in the data series to predict traffic states in the future. Thus, the traffic prediction problem can be formulated as follows:

\[
\hat{y}_{t+T_h} = f([X_{t-T+1}, X_{t-T}, \ldots, X_t])
\]  (2.3)

\( f: \) learnable function

\( X_t: \) traffic speed at time \( t \)

\( T_h: \) length of prediction horizon
$T$: length of data used for prediction

$\hat{y}_{t+T_h}$: predicted speed for time horizon $T_h$

Studying traffic condition predictions started five decades ago. The first generation of studies was the autoregressive integrated moving average (ARIMA) family, which are classical statistical models. Lee and Fambro (1999) developed the first ARIMA algorithm to predict the volume and occupancy of a freeway in Los Angeles. They used 168 datasets from surveillance police cameras and showed that ARIMA outperformed the moving average in terms of mean absolute error (MAE). Tian et al. (2015) applied several time-series models to predict the volume and to investigate the accuracy of these models. They used the Akaike’s information criterion (AIC) with the conditional maximum likelihood to estimate the parameters. They found that all the time-series models performed with reasonable accuracy. However, the subset ARIMA model accomplished more stable and accurate results. Other researchers implemented various versions of ARIMA to predict speed, volume, and occupancy.

In contrast, machine learning methods are able to consider non-linearity compared to statistical methods like ARIMA that consider traffic to be stationery and have linear relationships in the data. Therefore, machine learning algorithms were developed in this research. Recently, researchers are increasingly interested in using deep neural networks (deep learning), which are a specific type of machine learning. The results from the deep learning models outperform the statistical models such as ARIMA and Kalman filtering and data-driven approaches such as Support Vector Regression (SVR) and Random forest in terms of accuracy and robustness (Yin et al., 2020, James, 2021, Cui et al. 2020, Jiang et al. 2021). Figure 2-3 shows the performance of statistical model (i.e., ARIMA) and machine learning model (i.e., SVR) compared to two simple deep neural network models (i.e., FNN and LSTM).
Deep neural network models extract features, such as spatial and temporal dependencies, automatically in a series of data. In other words, they build and discover spatiotemporal features by exploring patterns in databases. The automatic feature selection algorithms result in more robust traffic predictions. In the next section, three deep neural network methods such as (Recurrent Neural Network (RNN), Convolution Neural Network (CNN), and Graph Convolution Network (GCN)) that have been used by previous researcher in developing the spatiotemporal traffic prediction framework are discussed.

### 2.4.1 Deep neural network models in traffic predictions

Among the three deep neural network models, RNNs are usually used to capture the temporal patterns in traffic data. Long short-term memory is a subcategory of RNN and is the most popular model for traffic prediction. Ma et al. (2015) and Wu and Tan (2016) were the first researchers to implement the simple LSTM in the context of traffic prediction. Ma et al. (2015) used the Beijing Ring Road speed data with a granularity of 2 minutes. They used one recent time interval to predict one time-step ahead. In contrast, Tian and Pan (2015) applied their models to California flow data with a granularity of 5 minutes. Their model predicted flow for 15 to 60 minutes ahead. Both models were disadvantaged because they only considered temporal relationships to predict traffic states. This limitation motivated the development of hybrid algorithms that combined various
neural network structures to consider the spatial and temporal relationships to increase the accuracy of the prediction. Yu et al. (2017) combined two LSTM layers with a 1D CNN layer to predict the next 5 minutes. They applied 1D CNN on a small part of a highway to capture the spatial features. In addition, they utilized one LSTM layer to track the periodicity of data and another LSTM layer to capture short-term features. The outputs of these three layers were then linked and fed into a fully connected layer. In addition, they used 5 minutes granularity flow data and a 75-minute sequence length to extract the short-term features and a 30-minute sequence length to determine the daily/weekly periodicity patterns. Zhao et al. (2020) developed the mixture deep LSTM model (deep LSTM means stacking several layers of LSTM). They applied the deep LSTM to peak-hour traffic speed to capture the unique character of traffic data. In addition, they developed a stacked autoencoder decoder to predict post-collision traffic speed and found that the accuracy of the results was not acceptable. They suggested this error would happen since the reported time and real time of a collision occurrence were usually not the same (i.e., the reported time was after the real time). Although the previous model structures tried to aggregate all results from various neural network layers in the final fully connected layer, some studies implemented a structure that used each layer as an input of the next layer. Cheng et al. (2017) developed an LSTM layer that used outputs from CNN. They modeled a road network as a graph in which each vertex represented the traffic conditions of each road segment and each edge depicted the relationships between road segments in the network. Choosing a target road (vertex) for prediction, they classified vertices into several groups based on the order of their relationships with the target vertex (order is defined as the number of edges passed when moving from one vertex to another). Then, the historical traffic conditions of each group was sent to the independent CNN layers to capture the spatial features. These results were then fed into the LSTM layer to determine the temporal
relationships among the data. Finally, a weighted max-pooling layer was applied to aggregate the results. Cheng et al. (2017) was one of the first studies that represented road networks as a graph to determine spatial features. Spatial features indicate the connectivity among different road segments. Zhao et al. (2019) adopted the graph propagation rule from Kipf and Welling (2016) to capture the spatial patterns of road networks. They combined a graph convolution network (GCN) and gated recurring unit (GRU) to predict speed. They found that the accuracy of a prediction was enhanced compared to when only temporal features were considered to predict traffic speed. This study was one of the first that used graphs to model road network structures. The last and most complex type of hybrid model, the Cui et al. (2020) models, alters the internal structure of LSTM. Cui et al. (2020) applied GCN to consider the kth neighbor at each step.

2.5 Summary of the literature

In summary, incident detection algorithms use a variety of methods to detect incidents. Among these methods, comparative methods most closely relate to this thesis. In terms of congestion propagation, most of the previous studies focused on finding congestion propagation patterns during recurring events, and only a few studies predicted congestion propagation during non-recurring events. In addition, congestion propagation prediction can be categorized based on geographic scope (e.g., single-corridor or network) and propagation phases (e.g., growth or recovery). While a few studies focused on congestion propagation during non-recurring events, almost all these studies predicted congestion propagation in a single corridor. Moreover, none of these studies, predicted the evolution of and recovery from congestion. As speed is a reliable measure of congestion, this thesis uses changes in speed profiles to identify congestion due to collisions. Therefore, this thesis first outlines the development of a prediction framework to predict
speed at the network level. Second, it presents the development of new measurements to identify a collision’s congestion effect. The new measurements are capable of distinguishing the congestion effect caused by collisions from that caused by daily traffic in arterial road networks. These two sections together enable the prediction of congestion propagation and recovery at the network level during non-recurring events.
3. ARTIFICIAL NEURAL NETWORK ALGORITHM (ANN)

ANNs have been used in various fields of study to solve complex real-world problems. In traffic prediction problems, the approximation mapping function is developed to convert input variables to output variables based on historical traffic data. Deep neural networks (DNNs) are types of ANNs that have multiple layers between the input and output layers, and each layer consists of a set of mathematical operations to map the input to the output. In recent years, deep learning-based techniques (such as LSTM, CNN) have been widely utilized in time series analyses as the main part of traffic prediction. These methods can address the spatiotemporal problem in speed prediction. The structure of an ANN is analogous to the structure of a biological neural system. It comprises multiple processing units (called artificial nodes or neurons) connected in consecutive layers to work together and produce the final output.

3.1 Linear regression and logistic regression

Linear regression is one of the simplest types of ANN to use if, for example, the goal is to predict the speed for one step ahead using four recent time steps. Each ANN structure maps inputs to outputs. For example, in the above-mentioned problem, there are four time-steps as inputs, and ANN maps them to an appropriate output. In Figure 3-1a linear regression example is given that maps four inputs (x1, x2, x3, x4) to final outputs (y) with their respective weights (w1, w2, w3, w4).
w4) and the bias value (bias is similar to an intercept added to a linear equation to adjust the output along with the weighted sum of the inputs to a neuron).

Therefore, the problem in Figure 3-1 can be formulated as follows:

$$Y = f(W \cdot X_{\text{input}} + \text{bias})$$

(3-1)

Alternatively, equation (3-1) can be re-written in matrix form as follows:

$$Y = f\left( [w_1 \ w_2 \ w_3 \ w_4] \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \text{bias} \right)$$

(3-2)

$$Y = f(w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + w_4 \cdot x_4 + \text{bias})$$

(3-3)

where $f$ is the linear regression activation function defined as $f(x) = x$. By changing the activation function to $f(x) = \text{sigmoid}(x)$, the logistic regression model can be built. $W$ is the learning parameter in which the model adjusts the values during the training phase.
3.2 Fully Connected Neural Networks (Feed Forward Neural Networks)

Fully connected neural networks (FCN) are a combination of many neurons in consecutive layers. The neurons are connected in a way that enables the model to solve complex, non-linear problems.

In FCN, all neurons in one layer are connected to all neurons in the next layer, and the outcomes of each layer are fed into the next layer. Figure 3-2 represents a two-layer FCN mapping four inputs to one output. In each layer, a weight matrix is adjusted based on the inputs and outputs. In addition, each layer’s activation function enables FCN to handle complex and non-linear problems.

Thus, FCN can be formulated as follows:

\[
\begin{align*}
\text{Hidden1} &= f_1(W_1 \ast X_{\text{input}} + \text{bias}_1) \\
\text{Hidden2} &= f_2(W_2 \ast \text{Hidden1} + \text{bias}_2) \\
Y_{\text{output}} &= f_3(W_3 \ast \text{Hidden2} + \text{bias}_3)
\end{align*}
\]

W’s are the weight matrices for each layer, and \(f_1, f_2, f_3\) are the activation functions. Moreover, \(\text{Hidden1}, \text{Hidden2}, \text{and } Y_{\text{output}}\) are the outputs of each layer, respectively. In addition,
bias₁, bias₂, and bias₃ are the bias values for each layer. These weights and biases are updated during the training phase. The training phase is explained at the end of this thesis in the appendix. While FCNs are appropriate to model nonlinearity among data, they are not applicable for time series prediction.

3.3 Recurring Neural Networks (RNNs)

The FCN model structure cannot consider the hidden relationships among time steps in a time series of data. In the example presented in the subsection on linear regression, the model uses the previous four time steps in the series to predict the next value in the series; FCN considers each of these four time steps as independent input, and so, the temporal relationships among the consecutive inputs are not examined. However, RNN considers that the data in the sequence are related to each other. As an analogy, to predict the final word in the sentence “I am tired; I need some…,” FCN considers each word in the sentence as independent input and predicts the last word. However, RNN considers the relationships between the words in the sentence. For instance, RNN considers “tired,” and predicts related words such as “rest” or “sleep.” Therefore, RNNs are suitable for dealing with time sequence data to capture the temporal correlation dependencies, which are called “hidden state” or “hidden information” in RNN structures. To clarify how RNN works using the example sentence, RNN units are represented as “A” in Figure 3-3. As shown, at each step, one word is input into the model. Then, the RNN unit predicts possible words. These possible words are those the model mostly sees during the training process. For example, in the first step, the words “am,” “want,” and “read” are the three most probable words after “I.” Therefore, the model sends these predicted values as the “hidden state” information to the next step. In the next step, the RNN unit receives two inputs: “am” from the previous unit and “am” from input data. Therefore, it bases its prediction on two pieces of information and chooses the
most probable word. For example, if “happy” is the next most probable word after “am,” “happy,” which is the hidden state information, is sent to the next RNN unit. However, the hidden information (“happy”) is not consistent with the input value (“tired”), and RNN assigns less weight to the hidden information. Then, the model predicts the word “rest.” This process is repeated until the model predicts the non-given word in the last step.

While RNN structures are appropriate for time series prediction, RNN can be defined as a very deep FCN with more time lags (hidden layers). Thus, RNN results in a vanishing or exploding gradient of the network, which means that the accuracy of a simple RNN may decrease as the sequence length increases because the earlier cells in the RNN get a small gradient update and stop learning in the backpropagation process. Thus, gated recurring neural network models, such as LSTM, have been proposed to overcome this issue.

The gated recurring neural network models are similar to the simple RNN model in that they consider correlation in the sequential data. Typical LSTM units consist of three gates: 1) input
gate, 2) output gate, and 3) forget gate and cell state. Therefore, the gates and the cell state manage whole procedures (explained in the next paragraph) in LSTM. LSTM transfers the cell state information and the hidden state information to the next time steps. In theory, the cell state can carry relevant information throughout the sequence processing. Thus, information from the earlier time steps can be stored in the memory of LSTM and be restored to later time steps. As the cell is passed to the next cells, the gates decide to add or remove this information. The gates are types of neural networks that decide which information should be allowed to pass onto the cell state, which means that gates can learn to distinguish relevant from irrelevant information and choose to keep or forget the information during training. Figure 3-4 shows the internal structure of LSTM.

The internal procedure in LSTM cells can be divided to four steps. First, there is a forget gate, which decides to keep or throw away information. The previous hidden state \( (h(t - 1)) \) and current input \( (x(t)) \) are passed on to the forget gate using the sigmoid activation function. The output is a range between 0 and 1. If the output is closer to 0, the gate will forget the cell state value. Second, the previous hidden state and current input are fed into the input gate, which chooses the values that will be updated based on the outputs of the sigmoid function.
Simultaneously, the current input and the previous hidden state are passed on to the tanh function. If the outcome of the sigmoid function is closer to 1, it is then important to use the new information from the tanh function outputs. Third, there exists enough information to compute the cell state. Thus, the input gate's output is taken, and a pointwise addition is applied to the output from the forget gate to update the cell state to a new value. Finally, an output gate decides what the next hidden state is. The hidden state carries information on previous inputs, and it is directly utilized for prediction. First, the current inputs and previous hidden states are passed on to the sigmoid function. The tanh function is applied to the newly updated cell state, and the result is multiplied by outputs of the sigmoid function. This procedure determines the information that the hidden states carry. Finally, the new hidden state $h(t)$ and cell state $C(t)$ are carried over to the next time-step. The final hidden state and cell state are calculated as follows:

$$i_t = \sigma(W_{ix} * X_t + W_{ih} * h_{t-1} + bi)$$ (3-7)

$$f_t = \sigma(W_{fx} * X_t + W_{fh} * h_{t-1} + bf)$$ (3-8)

$$o_t = \sigma(W_{ox} * X_t + W_{oh} * h_{t-1} + bO)$$ (3-9)

$$\tilde{c}_t = tanh(W_{cx} * X_t + W_{ch} * h_{t-1} + bc)$$ (3-10)

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1}$$ (3-11)

$$h_t = o_t \odot tanh(c_t)$$ (3-12)

where $X_t$ represents the input at time interval $t$; $h_{t-1}$ and $c_{t-1}$ represent the hidden state information and cell state information in time interval $t-1$, respectively; $W_{ix}$, and $W_{ih}$ represent the learnable weights in the input gate; $bi$ represents the bias in the input gate; $W_{fx}$ and $W_{fh}$ represent the learnable weights in the forget gate; $bf$ represents the bias in the forget gate; $W_{ox}$ and $W_{oh}$ represent the learnable weights in the output gate; $bO$ represents the bias in the output gate; $W_{cx}$ and $W_{ch}$ represent the learnable weights in the
cell state; and $bc$ represents the bias in cell state. The input gate, the forget gate, the output gate, and the cell state at time interval $t$ are defined as $i_t$, $f_t$, $O_t$, and $c_t$, respectively. Two element-wise functions $tanh$ (a hyperbolic tangent function) and $\odot$ (Hadamard product) are used to relate gate outputs.

3.4 Convolution Neural Networks (CNNs)

A CNN is an algorithm that performs well in image processing, computer vision, and the image recognition field Oquab et al., (2014). CNN structures have mostly been used in spatial problems due to their distance sensitivity. Each CNN structure consists of convolution layers, pooling layers, and fully connected layers. In the convolution layers, various filters (or kernels) extract different features, and these filters are sets of learnable weights that are adjusted during the training process to produce output features. First, the convolution filter is placed on the top left corner of the input matrix, then the product between the numbers at the same location in the input matrix and the filter are calculated, and these products are added together to obtain a single number, which is the convolution result of this operation. Then, the filter is moved to the right by one element [here the stride is (1,1), which means the filter moves by one element to the right and one element down until all values in the matrix are covered], and the convolution result is obtained. A pooling layer is then used to extract dominant features and reduce the number of parameters. Then, the results are sent to the fully connected layer, which makes a prediction. For example, in an input matrix with $N$ rows and $M$ columns, each row represents a sensor location, and each column shows the time-step. In addition, there are $K$ filters with size $s*s$. Thus, the convolution layer with such a setting converts the input matrix with size $N*M$ to $(N-s+1) * (M-s+1) * K$. Then, the pooling layer with a kernel (filter mask) size of $p*p$ is applied. Based on the type of pooling layer (average pooling or max pooling), the average or the maximum of the numbers at the same location in the
output of the convolution layer matrix and the pooling kernel are calculated. The output dimensions of the pooling layer depend on the stride setting. With a stride (2, 2), the output size is \(((N-s+1)/2) \times ((M-s+1)/2) \times K\). Finally, the matrix is flattened and is passed on to the fully connected layer. An example with ten sensors and eight time-steps is shown in Figure 3-5.

3.5 Graph Convolution Network (GCN)

While neural networks have been successful in recent years, they could only be implemented using Euclidean data. However, most real-world data are non-Euclidean and follow graph structures. Thus, recent studies have focused on graph-based structured data and the development of graph neural networks, of which GCN is one. GCN models learn features using the information from neighboring nodes. Each GCN operates on graphs. Thus, inputs contain a graph structure, e.g., an \(N \times N\) matrix representing an adjacency matrix \(A\) of given graph \(G = (V, E)\) with \(N\) nodes and the input features matrix \(X^{N \times f}\), where \(N\) is the number of nodes, and \(f\) represents the number of input features for each node. Therefore, the GCN propagation rule can be expressed as follows:
where $H^i$ represents the output matrix at layer $I$, and $A$ represents the adjacency matrix. In addition, $H^0 = XNf$ and $f$ represent the propagation rules that compute the feature representation of a node as an aggregate of the feature representations of its neighbors $g(A, H^{i-1})$ before it is transformed by applying the learnable parameters $W^{i-1}$ and activation function $\sigma$. In other words, each layer of GCN consists of an activation function, an aggregation function that defines the type of relationships among the nodes, and a learnable parameter that is adjusted during the training process. Three types of aggregation functions are discussed below: 1) Sum Rule, 2) Mean Rule, and 3) Spectral Rule.

**Sum Rule**

The Sum Rule aggregation function computes a node’s features as the sum of the features of the neighbor nodes:

$$g(A, H^{i-1}) = AH^{i-1} \quad (3-14)$$

However, this aggregation function cannot include its own features. Further, nodes with large degrees have large values in their feature calculation, while nodes with small degrees have small values, which can cause issues such as exploding gradients. In addition, training becomes more difficult when algorithms, such as stochastic gradient descent, are sensitive to feature scaling.

**Mean Rule**
The Mean Rule is used to overcome the shortcomings of the Sum Rule. First, the adjacency matrix is modified by adding the identity matrix to $A$. The new adjacency matrix $\hat{A} = A + I$ represents a self-loops matrix that considers the feature of its node among the neighbor nodes. Then, the feature representations are normalized by multiplying the self-loops matrix by the inverse degree matrix $D^{-1}$; this process generates a new aggregate function that calculates the mean value features of the neighboring nodes. Therefore, the aggregation function can be described as follows:

$$g(A, H^{i-1}) = D^{-1} \hat{A} H^{i-1}$$  \hspace{1cm} (3-15)

**Spectral Rule**

Kipf and Welling (2016) proposed a spectral propagation rule using the first-order approximate of localized spectral filters on graphs (Defferrard et al., 2016). The spectral rule differs from the previous rules discussed above because it chooses other aggregate function types. Although the normalization process is somewhat similar to the mean rule, which uses the degree matrix $D$ to the power of negative one, the spectral rule normalization is symmetric.

$$g(A, H^{i-1}) = \hat{D}^{-0.5} \hat{A} H^{i-1} \hat{D}^{-0.5}$$  \hspace{1cm} (3-16)

where $\hat{A} = A + I$ is the adjacency matrix of an undirected graph with the added self-loops, and $I$ is the identity matrix. $\hat{D}$ is the degree matrix of the adjacency matrix $\hat{A}$.

Kipf and Welling (2017) defined the spectral convolution as a product of filter $g = diag(N)$ with a scaler value for every node $X^N$; the formulation is shown below:

$$g * X = U g U^T X$$  \hspace{1cm} (3-17)
where $\lambda$ is the eigenvalues, and $U$ is the eigenvectors of the symmetrical, normalized graph Laplacian $L = I_N - D^{-0.5}AD^{-0.5} = U\lambda UT$. Therefore, the filter $g$ can be defined as the function of eigenvalues of $L$, i.e., $g(\lambda)$. Due to the complexity of calculating the equation, Kipf and Welling (2016) applied the Chebyshev polynomial approximations up to the first order. The Chebyshev approximation is as follows:

$$g \approx \sum_{k=1}^{K=1} \theta_k T_k(L) = \theta_0 T_0(L) + \theta_1 T_1(L) \quad (3-18)$$

$$\bar{L} = \frac{2}{\lambda_{max}} L - I_N \quad (3-19)$$

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \quad (3-20)$$

where $T_0(x) = 1, T_1(x) = x$, $\theta$ is the Chebyshev parameter, and $\lambda_{max}$ is the largest eigenvalue. By further approximation of $\lambda_{max} \approx 2$, the equation can be simplified as follows:

$$g \approx \theta_0 + \theta_1 (L - I_N) = \theta_0 + \theta_1 D^{-0.5}AD^{-0.5} \approx \theta (I_N + D^{-0.5}AD^{-0.5}) \quad (3-21)$$

Considering the parameter $\theta = \theta_0 = -\theta_1$ can be beneficial to address overfitting. In addition, Kipf and Welling (2016) applied the normalization trick $I_N + D^{-0.5}AD^{-0.5} \rightarrow \tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5}$ that leads to stability and prevents vanishing/exploding gradients. Thus, they used equation (3.16) for the aggregation function.
4. A SPATIOTEMPORAL FRAMEWORK FOR TRAFFIC SPEED PREDICTION

4.1 Congestion propagation prediction solution steps

As discussed in the introduction, solving the congestion propagation prediction problem has two steps as shown in Figure 1-2: 1) the speed forecasting problem and 2) the congestion quantification problem. This section is on the speed forecasting problem and first describes the algorithm that detects collision occurrences and accordingly generates an impact area denoted as graph G (step 1 and 2 in Figure 4-1). Afterward, this section describes the GCN_LSTM forecasting model structure that tracks and predicts speed profiles in the impact area during the collision timeline (step 4 in Figure 4-1). The developed algorithm is tested on a case study conducted in downtown Calgary because travel time data is available. Thus, the study area and the sources of data used are described in detail, followed by a description of the procedure adopted in constructing graph G based on multi-source data such as speed, weather, periodicity, and geospatial data (i.e., the adjacency matrix). The results of the model training and validation results for all collisions between 2014 to 2019 in downtown Calgary are presented.
In the first step, the collision location and time are extracted from the collision database. The latitude and longitude of the collision are utilized to locate the congestion on a map. The road segment corresponding to the collision is selected and tagged as the “main link subject to
collision.” The road segments in the vicinity of the collision are identified and extracted based on their connectivity with the “main link subject to collision” (step 1 and 2 in Figure 4-1). This process is depicted as the algorithm in Table 4-1 and used to build the impact area graph (graph G).

**Table 4-1 Process to build the impact area graph (graph G)**

<table>
<thead>
<tr>
<th>Procedure building network (graph G) around the collision</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Road segment subject to collision latitude and longitude</td>
</tr>
<tr>
<td><strong>Output:</strong> Graph G (Network around the link subject to collision)</td>
</tr>
<tr>
<td><strong>Step 1: Locate the collision in the corresponding road segment</strong></td>
</tr>
<tr>
<td>For all segments $S \in$ INRIX geospatial database</td>
</tr>
<tr>
<td>If $Start_{lat_S} &lt; collision_{lat} &lt; End_{lat_S}$</td>
</tr>
<tr>
<td>If $Start_{long_S} &lt; collision_{long} &lt; End_{long_S}$</td>
</tr>
<tr>
<td>Store latitude and longitude of this segment as M_segment</td>
</tr>
<tr>
<td>End for</td>
</tr>
<tr>
<td><strong>Step 2: Build the upstream graph for the link J</strong></td>
</tr>
<tr>
<td>For all segments $S \in$ INRIX geospatial database</td>
</tr>
<tr>
<td>If $(Start_{lat_J} = End_{lat_S}$ and $Start_{long_J} = End_{long_S}$) or $Start_{lat_J} = Start_{lat_S}$</td>
</tr>
<tr>
<td>Make list of upstream segments connected to link J</td>
</tr>
<tr>
<td>End for</td>
</tr>
<tr>
<td><strong>Step 3: Build the downstream graph for the link J</strong></td>
</tr>
</tbody>
</table>
For all segments $S \in$ INRIX geospatial database

If $(End_{lat_j} = Start_{lat_S}$ and $End_{long_j} = Start_{long_S})$

Make a list of downstream segments connected to link $J$

End for

The boundary of the area impact network for both upstream and downstream links can be changed based on the intensity and duration of a collision. For instance, for a collision with a duration of less than 1 hour, the area impact network for downtown Calgary encompasses four upstream and three downstream links. As the duration and severity of a collision increase, a larger boundary or impact area can be constructed. This graph is used to extract the adjacency matrix of graph $G$ as outlined in section 4.4.4.2.

4.3 Speed prediction model

With the advent of GPS technology, high-volume, high-resolution, heterogeneous traffic data can now be obtained from various transportation agencies. This high-resolution traffic data along with the collision data enable a closer examination of the spatiotemporal evolution of congestion patterns during the collision timeline. This thesis fuses three multi-source/multi-type data to investigate congestion propagation behavior: 1) speed data, 2) geospatial data, and 3) collision data. The spatiotemporal traffic prediction problem can be described as training the mapping
function $f$ using the road network topological feature $G$ and feature matrix $X_t, t \in \{t - n, ..., t - 1, t\}$ as illustrated as below:

$$[X_{t+1}, X_{t+2}, ..., X_{t+N}] = f(G; (X_{t-n+1}, ..., X_{t-1}, X_t))$$  \hspace{1cm} (4-1)

where $X_t$ represents all features assigned to graph $G$ (e.g., speed data, weather information, and periodicity data) at time interval $t$; $n$ is the sequence of previous time-steps in the series that the
model utilizes to predict the future speed profiles in the impact area, and \( N \) is the length of the horizon prediction.

4.3.1 The combined graph convolution network (GCN) and long short-term memory (LSTM) model structure

The architecture of the developed GCN_LSTM model that is used to predict speed for the horizon length is shown in Figure 4-2. The model consists of four learning layers: 1) spatial learning, 2) temporal learning, 3) periodicity learning, and 4) context learning. In the model structure, to capture the spatial relationship in a road network, the constructed graph corresponding to the impact area graph with the assigned speed features is transferred to the GCN layer at each time interval \( t \in \{ t - n, ..., t - 1, t \} \). Spatial relationships can be defined as the effects of the traffic conditions of the upstream road segments on downstream road segments (i.e., forward flow propagation) or the effects of the traffic conditions of the downstream road segments on upstream road segments (i.e., backward flow propagation). In addition to space, the traffic state of the road segments is interrelated over time. Thus, to capture this temporal correlation, the outputs from each time step's spatial layer are fed into the LSTM layer. In other words, future speeds on a given road segment are affected by the previous speed profile from both the upstream and downstream road segments. Using two layers of GCN enables the model to consider the effects of both the upstream and downstream road speeds up to two levels. Thus, combining the spatial layer with the temporal layer provides information on how upstream segment traffic conditions in recent time intervals can affect the traffic conditions in downstream segments in future time intervals and vice versa. The temporal layer's outcomes are then linked with the features extracted from periodicity learning and contextual learning.
Periodicity learning uses historical speed information for the same time of day and day of the week to predict speed because traffic conditions exhibit similar patterns for the same time of day and day of the week. In addition, contextual learning investigates the impact of weather information on traffic conditions. For instance, snowy weather and heavy rain precipitation reduces speed. Finally, the aggregated features are fed into the fully connected layer to predict the speed of the next time interval. This last layer works as a ranking process by giving more weight to the factors that have a significant impact on speed.
Figure 4-2 Deep neural network speed prediction model structure
4.3.1.1 Spatial learning layer

Spatial relationships, as mentioned above, are defined as the effects of forward flow propagation or the effects backward flow propagation. So, the output of this layer is the impact of traffic conditions of neighboring road segments on given road segments. While spatial features can be obtained using CNN, they can only be implemented using Euclidean data, which is data that is plotted in n-dimensional linear space; the direct distance between points represents how much these points differ from each other in terms of their features. However, in the real world, most data, such as road network data, are non-Euclidean and follow a graph structure. In non-Euclidean data, the degree of connectivity of nodes is what is important. For example, in analyzing traffic conditions, congestion propagates differently depending on the number of connecting roads. In recent years, various graph neural network algorithms, such as GCN, have been developed. GCNs use information from neighboring nodes to learn certain features (in this case connectivity level). Thus, the inputs contain a graph structure, e.g., an $N \times N$ matrix represents the adjacency matrix ($A$) of a given graph $G = (V, E)$ with $N$ nodes and the input feature matrix $X^{N \times f}$, where $N$ is the number of nodes and $f$ represents the number of input features for each node. Therefore, the GCN propagation rule can be expressed as follows:

$$H^i = f \left( H^{i-1}, A \right) = \sigma \left( g \left( A, H^{i-1} \right) * W^{i-1} \right)$$  \hspace{1cm} (4-2)

where $H^0 = X^{N \times f}$ and $f$ represent the propagation rule; $g$ represents the aggregate function; $W^{i-1}$ is the learnable parameters; and $\sigma$ represents the sigmoid activation function. In other words, each GCN layer consists of an activation function, an aggregation function that defines the type of relationship among nodes, and a learnable parameter (i.e., weight matrix and bias) that is adjusted during the training process (Afrin & Yodo, 2020). The spectral propagation rule using the first-
order approximation of localized spectral filters on graphs is explained in detail in section 3.5 and shown below:

\[ g(A, H^{l-1}) = \tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5} H^{l-1} \] (4-3)

where \( \tilde{A} = A + I \) is the adjacency matrix of an undirected graph with added self-loops, and \( I \) is the identity matrix. \( \tilde{D} \) is the degree matrix of the adjacency matrix \( \tilde{A} \). The layer-wise propagation rule can be stated as follows:

\[ H^l = \sigma(\tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5} H^{l-1} W^{l-1}) \] (4-4)

As shown in Figure 4-2, two sequential layers of GCN are used to capture the topological relationship between each road and its surrounding neighbors up to the 2\(^{nd}\) level. The input at each time interval is defined as \( X_t = \{V_t^1, V_t^2, ..., V_t^N\} \), where \( N \) is the number of graph nodes (i.e., number of roads), and the adjacency matrix \( A \) is sent to the GCN layers. Thus, the geospatial sequence features \( \{Y_{t-n}^S, Y_{t-n+1}^S, ..., Y_t^S\} \) at each time interval is ready to be fed into the temporal layer. The propagation rules for the first and last elements of the sequence are as follows:

\[ Y_{t-n}^S = \sigma(\tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5} \text{Relu}(\tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5} X_{t-n} W^{l-1}) W^{i}) \] (4-5)

\[ Y_t^S = \sigma(\tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5} \text{Relu}(\tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5} X_t W^{l-1}) W^{i}) \] (4-6)

\subsection{4.3.1.2 Temporal learning layer}

Traffic road segments are connected in time as they are in location. So, the outputs from each time step's spatial layer are fed into the LSTM layer, which enables the model to capture the temporal pattern that reveals the time dependency of traffic. Thus, in this layer, the sequence of spatial
features $Y_t^S$ is fed into the LSTM to capture the temporal dependency of speed with recent speeds. Typical LSTM units consist of three gates: 1) input gate, 2) output gate, 3) forget gate, and one cell state. Therefore, these various gates and the cell states manage whole procedures in LSTM. The cell states transfer relevant information down the sequence chain. In theory, the cell state can carry relevant information from previous time intervals throughout the processing of the sequence. Thus, information from the earlier time steps can be stored in the memory of LSTM and can be transferred to later time steps. With the input of spatial sequence features $\{Y_{t-n}^S, Y_{t-n+1}^S, ..., Y_t^S\}$, the output sequence $\{Y_{t-n}^{S_T}, Y_{t-n+1}^{S_T}, ..., Y_t^{S_T}\}$ iteratively obtained. The propagation rules for the first and last elements of the sequence are as follows:

\begin{align}
  i_{t-n} &= \sigma(W_{i,x} * Y_{t-n}^S + W_{i,h} * h_{t-n-1} + b_i) \quad (4-7) \\
  f_{t-n} &= \sigma(W_{f,x} * Y_{t-n}^S + W_{f,h} * h_{t-n-1} + b_f) \quad (4-8) \\
  o_{t-n} &= \sigma(W_{o,x} * Y_{t-n}^S + W_{o,h} * h_{t-n-1} + b_o) \quad (4-9) \\
  c_{t-n} &= \tanh(W_{c,x} * Y_{t-n}^S + W_{c,h} * h_{t-n-1} + b_c) \quad (4-10) \\
  Y_{t-n}^{S_T} &= h_{t-n} = O_{t-n} \circ \tanh(c_{t-n}) \quad (4-12)
\end{align}

\begin{align}
  i_t &= \sigma(W_{i,x} * Y_t^S + W_{i,h} * h_{t-1} + b_i) \quad (4-13) \\
  f_t &= \sigma(W_{f,x} * Y_t^S + W_{f,h} * h_{t-1} + b_f) \quad (4-14) \\
  o_t &= \sigma(W_{o,x} * Y_t^S + W_{o,h} * h_{t-1} + b_o) \quad (4-15) \\
  c_t &= \tanh(W_{c,x} * Y_t^S + W_{c,h} * h_{t-1} + b_c) \quad (4-16)
\end{align}
\[ c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \]  \hspace{1cm} (4-17)

\[ Y_t^{ST} = h_t = o_t \odot \tanh(c_t) \]  \hspace{1cm} (4-18)

All variables in the above equations are defined in section 3.3.

The last LSTM cell output \( Y_t^{ST} \), which carries spatiotemporal features, is then used in the concatenation process to predict speed. Combining the spatial layer with temporal layer provides information on how upstream segment traffic conditions in recent time intervals can affect the traffic conditions in downstream segments in future time intervals and vice versa.

4.3.1.3 Periodicity learning layer

Periodicity learning is used in this study to capture the periodicity of traffic behavior for a given time of day (within day dynamics) and day of the week (day to day dynamics). Traffic speed usually exhibits a recurring pattern because the speed of a particular road segment at a specific time is similar to the speed of that road segment at the same time on previous days or weeks. For example, speed starts decreasing at 6 AM due to morning traffic congestion, and it restores to the free flow speed around 9 AM. This behavior happens periodically at the same time on the same road segment. Traffic speed data of a road segment at a given time interval \( (X_{t+1}) \) can be extracted for \( n \) previous weeks to obtain the periodicity information of traffic speed in a time interval. The speed data are passed to the fully connected layer using the following propagation rule:

\[ Y^P = (W_p \ast X_{Historical}^{t+1} + bias_p) \]  \hspace{1cm} (4-19)

where \( W_p \) is the learnable weight, and \( X_{Historical}^{t+1} \) is \( n \) previous weeks of historical speed for given time interval \( t+1 \). The outcome of periodicity features \( Y^P \) is then used in the concatenation process.
4.3.1.4 Context learning layer

Along with the periodicity and spatiotemporal features, other contributing factors (context features), such as weather, holiday or non-holiday days, and day of the week, influence the speed variation in road networks. Thus, a fully connected layer is implemented inside the context learning layer to reflect the speed variations introduced by these external variables (weather, special holidays) with the following propagation rule:

\[ Y^C = (W_c * X_{context}^{t+1} + bias_c) \]  

(4-20)

where \( W_c \) is the learnable weight, and \( X_{context}^{t+1} \) is information on weather, holiday or non-holiday days, and day of the week for a given time interval t+1. The outcomes of this layer \( Y^C \) are integrated with previous features in the concatenation process.

4.3.1.5 Concatenation process

The spatiotemporal, periodicity, and context features affect speed in varying degrees. Thus, all features (\( Y^{Sr} \), \( Y^P \), and \( Y^C \)) are combined in one single matrix, and this matrix is fed into a fully connected layer. This strategy enables the model to adjust the learnable weight (\( W_{concat} \)) to reflect the impact of each feature based on its importance in the prediction.

\[ \hat{Y}_t = ReLU(W_{concat} * [Y^{Sr}, Y^P, Y^C] + bias_{concat}) \]  

(4-21)

The output of the concatenation process, \( \hat{Y}_t \), is the value predicted by the model, which is compared to the real speed, \( Y_t \), to compute the loss function. This model uses backpropagation to update learnable weights and to minimize the root mean square error (RMSE) between the predicted speed vectors and real speed.
4.4 Application of the model to a case study for downtown Calgary, AB

4.4.1 Data description

In this section, the proposed GCN_LSTM algorithm is tested in downtown Calgary. A total of 344 road segments are used in training and testing the model. The chosen segments in downtown Calgary are shown in Figure 4-3. As discussed in the previous section, this study predicts congestion propagation during a collision at the network level using predicted speed. This model utilizes speed and geospatial datasets from INRIX, collision data from the City of Calgary, and weather datasets from the Government of Canada website (2020) to predict speed.

![Figure 4-3 Downtown Calgary road segments](image)

4.4.1.1 Speed data

Speed data from INRIX, a company that manages traffic by analyzing data from road sensors and vehicles, is used. INRIX provides average speed data for each road segment, and each road segment is divided into several segments and assigned a unique segment ID. In addition, INRIX provides average speed in the granularity of 1, 5, 15, and 60 minute intervals. INRIX defines the accuracy measure (CValue), which depicts whether the reported average speeds are based on real-time data or historical data. In other words, this measure represents the confidence level of the
average speed. Therefore, INRIX reports traffic data (speed and travel time) in three ways: 1) real-time speed exists (Cvalue is equal to 100), 2) real-time speed partially exists (Cvalue is between 0 and 100), and 3) real-time speed does not exist because there are no sensors or any cars on the specific segment (Cvalue is 0 or Nan). A sample of raw data from INRIX is illustrated in Figure 4-4.

<table>
<thead>
<tr>
<th>Date Time</th>
<th>Segment ID</th>
<th>Speed(km/hour)</th>
<th>Hist Av Speed(km/hour)</th>
<th>Ref Speed(km/hour)</th>
<th>Travel Time(Minutes)</th>
<th>CValue</th>
<th>Pct Score30</th>
<th>Pct Score20</th>
<th>Pct Score10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021-02-01T00:00:00-07:00</td>
<td>284705466</td>
<td>35.0</td>
<td>35.0</td>
<td>35</td>
<td>0.15</td>
<td>NaN</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2021-02-01T00:05:00-07:00</td>
<td>284705466</td>
<td>35.0</td>
<td>35.0</td>
<td>35</td>
<td>0.15</td>
<td>NaN</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2021-02-01T00:10:00-07:00</td>
<td>284705466</td>
<td>35.0</td>
<td>35.0</td>
<td>35</td>
<td>0.15</td>
<td>NaN</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2021-02-01T00:15:00-07:00</td>
<td>284705466</td>
<td>35.0</td>
<td>35.0</td>
<td>35</td>
<td>0.15</td>
<td>NaN</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2021-02-01T00:20:00-07:00</td>
<td>284705466</td>
<td>35.0</td>
<td>35.0</td>
<td>35</td>
<td>0.15</td>
<td>NaN</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2021-02-01T00:25:00-07:00</td>
<td>284705466</td>
<td>35.0</td>
<td>35.0</td>
<td>35</td>
<td>0.15</td>
<td>NaN</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2021-02-01T00:30:00-07:00</td>
<td>284705466</td>
<td>37.0</td>
<td>37.0</td>
<td>35</td>
<td>0.15</td>
<td>NaN</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2021-02-01T00:35:00-07:00</td>
<td>284705466</td>
<td>37.0</td>
<td>37.0</td>
<td>35</td>
<td>0.15</td>
<td>NaN</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2021-02-01T00:40:00-07:00</td>
<td>284705466</td>
<td>37.0</td>
<td>37.0</td>
<td>35</td>
<td>0.15</td>
<td>NaN</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2021-02-01T00:45:00-07:00</td>
<td>284705466</td>
<td>35.0</td>
<td>35.0</td>
<td>35</td>
<td>0.15</td>
<td>NaN</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Figure 4-4 Sample of Raw INRIX traffic data

“Hist Av Speed” means the historical average speed, and “Ref Speed” is the free-flow speed, which is the speed when the road segment is unobstructed. This speed is based on INRIX historical data and is not the legal speed limit. “Travel Time” is the average travel time on a specific segment ID.

There is also some information on the coordinates of the segments, length of the segments, and the direction of the segment in the INRIX datasets. INRIX traffic data consist of approximately 16,000 segments in the City of Calgary. While INRIX covers most road segments in the City of Calgary, real-time traffic data are mostly reported for major roads.
4.4.1.2 Spatial data

Spatial data, also known as geospatial data, is information on a physical object that can be represented by numerical values in a geographical coordinate system. This thesis utilizes spatial data to build the network as a graph. As discussed in the previous section, INRIX provides physical road information, i.e., the start and endpoint coordinates of a segment, the length and direction of a segment (shown in Figure 4-5); the unique segment ID defines each road segment. How to use this information to build a graph is discussed in the next sections.

4.4.1.3 Collision data

The City of Calgary annually reports incident data that includes collisions and road maintenance data. This collision and construction data are available daily from 2014 to 2019. This study uses collision data to locate collisions and investigate propagation patterns in the vicinity of links subject to incidents. Collision data includes collision coordinates, start date and time, end date and time, description, and type. The description provides information about the type of collision or type of blockage that happened at that specific time and location. “Type” indicates if an event is a collision (Type 1) or due to road construction (Type 2).
This section represents the step 3 in Figure 4-1. There are four major steps in the data preparation process as shown in Figure 4-8: 1) graph construction (spatial dataset), 2) temporal dataset construction, 3) periodicity dataset construction, and 4) weather dataset construction. The processed data are then assigned to graph G as an input of model.

4.4.2.1 Geospatial dataset (calculating adjacency matrix)

After building the impact area graph (graph G) in section 4.2, the adjacency matrix is calculated. This study uses the unweighted and undirected graph to present the topological structures of a given road network. The undirected graph describes forward and backward congestion propagation in a network. The network is defined as a graph $G=(V, E)$ of the vicinity of a collision location; $V$ denotes the graph’s node and represents the set of roads $V\{v_1, v_2, ..., v_n\}$ in the study area, and $E$ shows whether two road segments are connected (i.e., $E=1$) or not (i.e., $E=0$). Thus, the
adjacency matrix $A^{N \times N}$ is used to present all connections between roads in network $G$. Each network contains the feature matrix $X^{N \times L \times F}$ where $N$ is the number of nodes (number of roads), $L$ represents the index of the time steps used for prediction $\{t - n, \ldots, t - 1, t\}$, and $F$ is the variables that each node (i.e., road) has (speed, day of the week, and weather). $X_t \in R^{N \times t \times l}$ is a variable $i$ of each node at time $t$. Because the day of the week and weather are maintained constant for all nodes at each time step, $X^{N \times L \times F}$ can be considered as $X^{N \times L \times 1}$; then, weather features and day of the week are added to a separate classification problem in the model. Therefore, the dimension of the input feature $X_t$ for each time interval $t$ is $R^{N \times 1 \times 1}$. This procedure is shown in Figure 4-8 and is highlighted in yellow.

### 4.4.2.2 Temporal dataset

INRIX provides real-time speed data in various granularities (1, 5, 15, 60 minutes). This study utilizes a granularity of 5 minutes, i.e., 12 time-intervals in one hour, and extracts data from April 2019 to April 2020 from 7 AM to 9 PM to train the GCN_LSTM model. The speed database with a speed matrix that has dimensions $N \times P$ is then constructed. $N$ represents the number of time intervals in one year, and $P$ represents the number of roads in graph $G$. Thus, each row represents a time interval, and each column represents a road segment. The segment's order in this matrix is the same as in the adjacency matrix. As discussed in the temporal learning layer, sequences of speed data are sent to the LSTM layer. Different values for horizon length are tested, and the most accurate is a horizon length of 10 intervals, and one time interval ahead is predicted. The sliding window (rolling horizon) is moved along consecutive time intervals in the speed matrix to build the training set. The speed values inside the time window are considered the features (the red box),
and the next value is represented as the label (the green box). These procedures are shown in the below figure:

![Figure 4-7 Time series sliding window – learning intervals are red and tested intervals are green](image)

As shown in Figure 4-7, this process is applied to all time intervals. Thus, with N time intervals in one year, (N-10) numbers of the samples are generated. All samples are used for training purposes. The dimensions of each sample are (10+1)*P, which contains ten features and one label to train the prediction model. Thus, X is the matrix of all samples that are entered into the GCN_LSTM model and has dimensions of (N-10)*(11)*P. This procedure is shown in Figure 4-8 and is highlighted in orange.
4.4.2.3 Periodicity dataset

This study uses a weekly period to capture periodicity. Therefore, the historical speed for five previous weeks for a given prediction horizon time interval \((t+1)\) for each individual segment in graph \(G\) are extracted from the speed dataset. Thus, the dimensions of this matrix are \((N-10) \times 5\). This procedure is shown in Figure 4-8 and is highlighted in green.

4.4.2.4 Weather dataset

Hourly weather data from the Calgary International Airport weather station, which is available on the Government of Canada website is used in this research to capture the effect of weather on speed prediction. The weather information includes cloudy, clear, rainy, snowy, and null value, which are considered as weather features. These features are the same for all road segments; therefore, this matrix has a dimension of \((N-10) \times 1\). However, weather features are categorical, and one hot encoding (binary encoding) is applied to it, which results in a matrix that has a dimension of \((N-10) \times 5\). These procedures are shown in Figure 4-8 and are highlighted in blue.
Figure 4-8 Processing data and assigning speed data, weather data, periodicity data, and spatial data (adjacency matrix) to graph G
4.4.3 Model setting for training

This study uses three different neural network models: FCN, GCN, and LSTM. In general, \((N-S_l)\) number of samples were used; \(N\) represents the total number of time intervals available, and \(S_l\) shows the sequence length (number of features). It is assumed that \(P\) road segments exist in graph \(G\). \(f\) represents an array that contains features where \(|f|\) is the number of features. The first entry of \(f\) is devoted to speed, the 2\(^{nd}\) to 6\(^{th}\) entries are weather features, i.e., 5 dimensions for 5 types of weather (clear, cloudy, rainy, snowy, and NA), and entries 7 to 12 are the periodicity features, i.e., speed values for five previous weeks at the same time of day. Using \(N\) as input leads to a high computation time; therefore, the model is trained in groups of data called batches. \(B\) and \(N_B\) are the batch size and number of batches, respectively.

In the spatial learning layer, two GCN layers are consecutively employed. As shown in Figure 4-2, the GCN cell is applied to only the speed feature (the first dimension among \(f\) dimensions) at each time interval in the sequence length. Thus, the input dimension of the GCN is \([N_B, B, S_l, P, I]\), and the output dimension is the same.

In the temporal learning layer, outcomes from the spatial learning layer are sent to an LSTM layer. The input dimension of the LSTM layer is \([N_B, B, S_l, P, I]\), which is the same as the previous layer. The number of neurons in each LSTM cell is assumed to be equal to the number of roads in graph \(G\). Each LSTM cell has an output size of \([N_B, B, I, P, I]\), and only the last LSTM cell outputs are used. Therefore, the output size of the LSTM layer is \([N_B, B, P, I]\).

All features of the label (the predicted value) except the first dimension of \(f\) are utilized in the weather and periodicity learning layer. For example, suppose the 11\(^{th}\) speed value is predicted in the time series using ten previous speed values, and the periodicity and weather features are only
meaningful for the 11th time interval. As a result, the weather learning layer input size is $[N_B, B, P, 5]$, and the weather features are sent to the FCN; FCN then generates an output that has size $[N_B, B, P, 1]$. However, as the weather features are consistent for all locations, the matrix is reduced to $[N_B, B, 1, 1]$ and then $[N_B, B, 1]$. The periodicity learning layer input size is $[N_B, B, P, 5]$. Historical data from five previous weeks are sent to the FCN layer, and FCN generates output that has size $[N_B, B, P, 1]$ and then is transformed into $[N_B, B, P]$.

The outputs from all layers are then linked to build a matrix of size $[N_B, B, 2P+1]$. The new matrix is passed to the FCN layer and generate the matrix of size $[N_B, B, P]$ to predict the speed of each segment in graph $G$.

All experiments were conducted in Python and used PyTorch nn.Module for the neural network models. All experiments were done on a system with GPU: NVIDIA GeForce GTX 1060. An RMSprop optimizer with a learning rate of $1e^{-5}$, root mean square error (RMSE) for the loss function, a batch size of 5, and 100 epochs was used to train the model.

4.4.4 Testing the Model

Downtown Calgary is chosen for the case study to investigate congestion propagation patterns at the network level because it is a dense urban network and real-time information is available. As previously mentioned, the model is first trained using one year of data from April 2019 to April 2020 for 333 road segments in downtown Calgary. Then, the model is tested to predict speed during the collisions in downtown Calgary before 2019.
4.4.4.1 Case study

For the case study, one example of a collision in downtown Calgary was used; the collision occurred on 26 May 2018. The occurrence time and clearance time of this collision were reported as 12:10 PM and 12:40 PM, respectively, by the City of Calgary. The network equivalent to graph $G$ for this collision was constructed using the algorithm in Table-3.1. The result is shown in Figure 4-9, and the road segment subject to the collision is marked with the red star. The road segment subject to the collision is a one-way road that stretches for 0.10 km north of Macleod Trail, a major arterial in downtown Calgary. In Figure 4-9, segments 9 Ave SW, 10 Ave SW, and 10 Ave SE were excluded from graph $G$ since the real-time speed information was not available. The road segments are labeled with their number in the INRIX database.

Figure 4-9  Impact area for collision #1 (graph G)
The predicted speeds versus true speeds (reported by INRIX) for several segments in downtown Calgary during collision #1 are plotted in Figure 4-10. The vertical and horizontal axes represent the speed and time interval (5-minute time intervals), respectively.

As shown in Figure 4-10, the GCN_LSTM model predicts the trend, i.e., ups and downs in speed, with high accuracy. For instance, the upstream segment (ID: 442964829) experiences two significant jumps in speed around time 11:25 AM and 11:50 AM. The proposed GCN_LSTM model can properly capture this behavior. The true speed value significantly decreases between 11:30 AM to 11:50 AM and 12:15 PM to 12:40 PM. The predicted speed also follows this trend during the same time intervals.

The mean absolute error (MAE) value for each segment is shown in Figure 4-10. The mean absolute error is higher for the upstream segments that are closer to the accident. For instance, MAE in segment ID:442981041 is 4.17, which is higher compared to segment ID: 442981040 because segment ID:442981041 is located on the first level vertical upstream where the collision happened. Thus, segment ID:442981041 is affected more than the other segment due to the collision and experiences more complex traffic dynamics, which increase the prediction error.

Moreover, MAE is smaller in the horizontal upstream. For instance, MAE of segment ID:431212995 is 1.67, which is less than MAE of segment ID:442981041 (4.17). This result is reasonable as segment ID:442981041 and the segment subjected to the collision are both in the vertical direction. Therefore, they are more affected by the collision and experience more complex traffic dynamics, which increase the prediction error. However, segment ID:431212995 is in the horizontal direction and experiences less disruption during the collision timeline.
In addition, the error term is smaller for the downstream segments compared to the upstream segments. For instance, the MAE value of segment ID:442964830 is smaller than that of segment ID:442981041 because upstream segments are more affected by the collision compared to the downstream segments. The error terms for all 40 segments in the network for collision #1 are calculated and listed in Table 4-2.
Figure 4-10 Speed prediction results for segments in the collision #1 network
4.4.4.2 Model validation

To evaluate the performance of the prediction model, MAE, root mean squared error (RMSE), and mean absolute percentage error (MAPE) are calculated for the examined collision. In addition, the model is tested on 152 recorded collisions that occurred in downtown Calgary between 2014 to 2019. The results are depicted in Table 4-2.

This thesis is the first study that predicts speed during non-recurring events, which makes it difficult to compare the results with that of other benchmark models. To validate the GCN_LSTM model, the error terms for the five previous top speed prediction models during recurring events are taken from Yin et al. (2020b) and listed in the third row of Table 4-2. For instance, the five speed prediction models predict speeds during recurring congestion with a minimum MAE of 2.69 to a maximum MAE of 4.86. The accuracy of the GCN_LSTM model is within an acceptable range compared to five previous models. In addition, since the five previous models only target recurring events, their reported errors are conducted under normal traffic conditions. In contrast, the error terms of the GCN_LSTM model are calculated during non-recurring events, which display more complex traffic conditions.

Table 4-2: Model error measures

<table>
<thead>
<tr>
<th>Error Measure</th>
<th>Mean absolute error</th>
<th>Root mean square error</th>
<th>Mean absolute percentage error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collision #1 (40 segments)</td>
<td>2.31</td>
<td>3.01</td>
<td>13.06%</td>
</tr>
<tr>
<td>152 downtown collisions</td>
<td>2.98</td>
<td>3.85</td>
<td>11.05%</td>
</tr>
<tr>
<td>5 top previous models (Yin et al., 2020b)</td>
<td>2.69-4.86</td>
<td>5.15-9.27</td>
<td>6.90% to 9.21%</td>
</tr>
</tbody>
</table>

As shown in Table 4-2, the MAE and RMSE errors of the GCN_LSTM model are within an acceptable range compared to the previous studies. However, the GCN_LSTM model has a higher MAPE compared to previous studies because the GCN_LSTM model is tested during the collisions, and speeds are very low compared to those of recurrent congestion conditions. For further clarification, suppose the model returns a predicted value of 100 km/h and a real value 90 km/h under the normal traffic conditions, and return a predicted value of 4 km/h and a real value of 2 km/h under the collision conditions. MAE and MAPE for the first and second cases are (10, 10%) and (2, 100%), respectively. Thus, MAPE is not a good performance measure to compare the performance of the proposed model with previous models because the proposed model is tested in cases of non-recurring events, which generally have significantly lower speed values compared to those in cases of recurring events.
5. USING A CONGESTION QUANTIFICATION FRAMEWORK TO MODEL NETWORK CONGESTION

5.1 Introduction

Congestion detection algorithms have been widely studied in the literature. These algorithms can be divided into two main approaches: 1) binary approaches and 2) continuous approaches. Binary approaches are algorithms that assign values of 0 and 1 to non-congested and congested conditions, respectively. And, continuous approaches are algorithms that assigns continuous values based on various congestion intensities.

5.1.1 Binary congestion detection

Afrin and Yodo (2020) introduced a speed reduction index (SRI) and speed performance index (SPI). In both indices, congestion is detected when the index value reaches a certain threshold. For example, the free flow speed in Figure 5-1. is 50 km/h, and the threshold is $\lambda=40\%$. When the speed drops below 20 km/h (calculated as $\lambda * free\_flow\_speed = 20\text{km/h}$), the segment is considered congested. While these detection approaches can distinguish between congested and non-congested conditions, the changing trends of speed cannot be recognized until the congestion index reaches the threshold values. For example, in Figure 5-1, speed starts decreasing at time
interval 12:15, but the binary approach is unable to detect congestion until time interval 12:25. Therefore, binary congestion detection approaches cannot identify congestion trends.

Fluctuation of speed below the threshold is common on arterials as arterial segments are periodically interrupted by traffic signals. Thus, traditional binary approaches are not able to capture these underlying dynamics and distinguish between periodic stop-and-go traffic at traffic lights and reduced speed caused by collisions. In fact, in the binary approaches, a road segment is considered congested by an anomaly when speed falls below the threshold for three consecutive time steps. For instance, in Figure 5-1, using a binary approach congestion detection method, anomalies happen and affect the road segment during three different time periods. First, speed starts decreasing at 11:10 AM and then falls below the threshold. Speed remains below the threshold from 11:15 AM to 11:25 AM (three consecutive time steps). Second, speed starts decreasing at 11:35 AM and stays below the threshold for an additional four consecutive time steps until 11:55 AM. Third, speed decreases at 12:15 PM and stays below the threshold until 2:10 PM. However, the reported collision time is between 12:10 PM and 12:40 PM.

Figure 5-1 First level upstream speed during collision
In addition, in Figure 5-2, the collision start time is shown with the orang vertical line. the lowest speed occurs at 11:20 AM, which is 50 minutes before the actual collision start time. Therefore, an efficient congestion detection framework should have several additional features. First, congestion measures should provide a more nuanced range of values to enable the model to properly reflect congestion dynamics.

Second, an efficient congestion detection framework should be capable of measuring the intensity of congestion. Intensity is defined as the degree to which congestion affects one road segment compared to other road segments. While binary approaches, such as SRI, determine congestion intensity based on various threshold values, including “severe,” “moderate,” and “low,” none of these approaches are able to compare congestion on different road segments; thus, a normalization procedure is needed to enable model to compare level of congestion among different road segments.
Third, the congestion detection framework should distinguish congestion caused by non-recurring events from congestion resulting from recurring events. This ability is an important feature that congestion detection frameworks must have to detect collisions, especially on arterial road networks. For example, Figure 5-3 shows the results of congestion binary approaches using the speed values in Figure 5-1. In this case, it is not possible to detect congestion resulting from the collision effectively. Based on congestion binary approaches, the time between 11:15 AM to 11:25 AM, 11:40 AM to 11:55 AM, and 12:25 PM to 13:10 PM are part of the collision; however, the reported collision occurrence time is between 12:10 PM to 12:50 PM. In other words, the congestion detected by binary approaches before time interval 12:10 PM and after time interval 12:50 PM should be investigated to identify whether this congestion results from the collision or the recurring congestion on this road.

Figure 5-3 Threshold approach for first level upstream: 0 represents no congestion, and 1 represents congestion
5.1.2 New congestion quantification index

So far, the shortcomings of the binary approach, which is the only method in the literature for detecting congestion, have been discussed. Therefore, to the best of the author’s knowledge, this study is the first to develop a congestion quantification framework to investigate the effect of congestion on arterial roads and the first to distinguish recurring from non-recurring congestion in road networks. The developed congestion quantification algorithm has the following improvements compared to the binary approach:

- provides a continuous range of values for congestion
- considers previous time interval congestion and how it affects in the current congestion
- provides a normalization technique that makes the algorithm globally comparative (i.e., values of congestion on different road segments can be compared and congestion intensity can be defined at the network level)
- incorporates hypothesis testing as an endogenous part of the algorithm to distinguish recurring congestion from non-recurring congestion
- models all congestion phases (i.e., start of congestion, peak, and recovery)

Afrin and Yodo (2020) introduced several criteria that described algorithms that measured congestion well: 1) The algorithm should be easy to understand for non-technical users. 2) It should provide a continuous range of values. 3) It should provide comparative values of different road types. 4) It should be applicable to different road types. In Table 5-1, the binary approach and the new congestion quantification algorithm (CQA) are compared based on the above criteria, and it shows that CQA outperforms the binary approach in detecting congestion.
In the next sections, the congestion value is calculated for CQA, and then, hypothesis testing is introduced to find the lower boundary for the speed that rejects the null hypothesis test. Afterwards, hypothesis testing is combined with the congestion value calculated in section 5.2 to build the CQA.
5.2 Congestion value calculation

Congestion can be considered the accumulated value of changing speed over time instead of an instantaneous value. The effect of congestion in earlier time intervals on congestion in the current time step fades away over time. So, congestion in recent time steps is greater compared to congestion in earlier time steps. To model this pattern, the amount of congestion that is passed from one time step to another is assumed to follow an exponential decay. The congestion value is defined as the sum of the portion of congestion in the previous step $C_v(t_{n-1})$ and the congestion effect $C_e(t_n)$ in the current time step. The congestion value in the current time step, $C_v(t_n)$, is obtained as follows:

$$C_v(t_n) = C_v(t_{n-1}) \cdot e^{-\lambda T} + C_e(t_n) \tag{5-1}$$

where $T$ is the time interval, and $\lambda$ is the constant decay parameter. $\lambda$ can be defined as the time required for congestion to decrease to half of its initial value, obtained by $\lambda = \ln(2) / t_{1/2}$. The half-life $t_{1/2}$ represents the time required for a quantity to reduce to half of its initial value; so, the effect of congestion directly relates to the change in speed. For instance, when speed changes from 100 km/h to 70 km/h on a freeway, it causes congestion, and when the speed changes from 70 km/h to 100 km/h, the congestion is relieved. Therefore, this change in speed can be formulated as follows:

$$C_e(t_n) \propto -\Delta v \tag{5-2}$$

where $\Delta v = v(t_n) - v(t_{n-1})$, which denotes the change in speed in consecutive time steps. $v(t_n)$ and $v(t_{n-1})$ are the speed at time interval $(t_n)$ and $(t_{n-1})$, respectively. $C_e(t_n)$ is the congestion effect at time interval $t_n$. In addition, the congestion effect directly relates to the density value: a
larger value of $k$ (i.e., density value in the fundamental diagram) indicates a higher value of congestion. Thus, these relationships can be formulated as follows:

$$C_o(t_n) \propto k$$  \hspace{1cm} (5-3)

In the fundamental diagram, speed ($u$) is related to density ($k$). The simplest relationship is the Greenshields model, which was introduced by Greenshields et al. (1935). Since then, several studies introduced new relationships to improve the over-simplified Greenshields model. These studies are categorized as single regime relationships (Greenberg, 1959; Newell, 1961; Underwood (1961) and multi regime relationships (Drake, 1967; Edie, 1961). Drake (1967), Greenberg (1959), and Underwood (1961) developed two parameter relationship functions, and Newell (1961) introduced a three parameter function. This thesis uses the two parameter speed-density function because density data is not available to calibrate a three parameter model. The Greenberg speed-density function is chosen as follows (Greenberg, 1959):

$$v = v_o, \ln \left( \frac{k_j}{k} \right)$$  \hspace{1cm} (5-4)

where $v$ is speed, $v_o$ is speed at capacity, $k$ is density, and $k_j$ is jam density. As this thesis models congestion as a function of speed, these formulations can be written as follows:

$$k = k_j \exp \left( -\frac{v}{v_o} \right)$$  \hspace{1cm} (5-5)

As $k_j$ is a constant parameter, equation (5-5) can be combined with (5-3):

$$C_o(t_n) \propto \exp \left( -\frac{v}{v_o} \right)$$
So, the congestion effect is as follows:

\[ C_e(t_n) \propto \exp \left( -\frac{v}{v_o} \right) \]  \hspace{1cm} (5-6)

\[ C_e(t_n) = -\Delta v \cdot \exp \left( -\frac{v}{v_o} \right) \]  \hspace{1cm} (5-7)

Equation 5-7 shows that when speed decreases, congestion increases and vice-versa. In addition, for two cases with the same \( \Delta v \), by including \( \exp \left( -\frac{v}{v_o} \right) \), the case with the lower current speed experiences more congestion as \( \exp \left( -\frac{v}{v_o} \right) \). For instance, suppose two collisions occur on two different road segments. In the first case, speed decreases from 50 km/h to 30 km/h, and in the second case, speed reduces from 25 km/h to 5 km/h. While \( \Delta v = 20 \) in both cases, the second case experiences more congestion (greater intensity) compared to the first case. Thus, congestion intensity is related to current speed \( v(t_n) \) in addition to change in speed \( \Delta v \).

### 5.3 Hypothesis testing

In the section, the congestion value formulation is combined with hypothesis testing to distinguish between recurring and non-recurring congestion.
As discussed in previous sections, since congestion can result from either recurring or non-recurring events, it is necessary to determine the source of congestion. In other words, the designed congestion function should detect congestion when vehicles stop at a signalized intersection. Most studies have compared current speed with average historical speed during the same time intervals to distinguish recurring congestion from non-recurring congestion. If the current speed is significantly less than the average speed, the algorithm considers the road segment as affected by a non-recurring event. This approach has been widely utilized to differentiate between congestion resulting from recurring and non-recurring events, but it cannot be applied to arterial road networks. Therefore, this thesis considers both the standard deviation of speed and average speed to perform a hypothesis test and determine whether speed follows the pattern of recurring events or that of non-recurring events.

Figure 5-4 Hypothesis testing
Consider a time series of speed data from one day of a week. Each time interval within the day contains multiple speed data points. Thus, each time interval has its own speed distribution (shown as 1 standard deviation) as indicated by the blue bars in Figure 5-4. The mean value of a sample is equal to the mean of the population, and the standard deviation of the population is calculated as $s / \sqrt{n}$ where $s$ represents the sample standard deviation, and $n$ is defined as the number of samples. The mean values and standard deviations are calculated for all time intervals. This thesis assumes that each time interval speed follows a normal distribution (Güner et al., 2012; Kim, 2018). According to the hypothesis test, the upper and lower bounds of $\alpha$ percent of confidence intervals are calculated as follows:

$$Upper_{bound} = \mu + Z_\alpha \ast s / \sqrt{n}$$

$$Lower_{bound} = \mu - Z_\alpha \ast s / \sqrt{n}$$

where $Z_\alpha$ is the z-value corresponding to $\alpha$ percent confidence interval of a standard normal distribution. In essence, the distribution of speed during normal traffic conditions (i.e., usual speed) and the distribution of speed collisions are significantly different. Thus, the upper and lower bounds of the usual speed represent whether a given speed data sample belongs to the typical speed distribution or not. In other words, if the sample speed is below the lower bound, the road is considered to be affected by a collision. In Figure 5-4, the mean values are the blue circles for each time interval, and the green and orange lines represent the upper and lower bounds, respectively. Therefore, this approach differentiates between recurring and non-recurring congestion by combining the congestion quantification function with the hypothesis test to model how the non-recurring congestion evolves through time.
5.4 Congestion Quantification Algorithm

The goal of this thesis is to develop a framework that assesses how congestion dynamics evolve during a collision. Recurring congestion and non-recurring congestion need to be identified to detect congestion caused by a collision. The framework for identification consists of two parts. First, using a congestion quantification function, corresponding values for all roads in graph $G$ are assigned based on predicted speed. The congestion value is computed from 1 hour and 30 minutes before the reported collision start time to one hour and 30 minutes after the reported collision clearance time (i.e., $T_w$) using predicted speeds calculated in chapter 3. Second, the lower bound speed value for $\alpha$ percent confidence interval is calculated for all road segments in graph $G$. As previously discussed, the upper and lower bound values are computed based on the historical values of each time interval during $T_w$. The lower bound values are sent to the congestion quantification function, and a congestion value is assigned to the lower bound speed value. The difference between the congestion values obtained from predicted speed and lower bound speed identifies whether an individual road segment is affected by the collision or not. In addition, due to the normalization step in the algorithm, the congestion values are comparative. The summary of the congestion quantification algorithm is shown in Table 5-2.

Table 5-2. Algorithm to identify the effect of collisions

<table>
<thead>
<tr>
<th>Procedure collision</th>
<th>congestion detection</th>
</tr>
</thead>
</table>

**Input:** Historical travel time data for links in network $G$ and predicted speed during collision for road segments in network $G$
**Output:** Recurring congestion value and congestion value during collisions for each link in network $G$

**Initialization:**

Assign an appropriate value to the confidence interval $\alpha$

Assign an appropriate value to the decay constant $\lambda$

Assign an appropriate value to the time interval size $T$

Set $C_v(t_0) = 0$

**Step 1:** Calculate congestion value during collision using predicted speed

For each link $l \in G$

For each time interval $t_n \in T_w$

Normalize speed between 0 and 1 $\Rightarrow \frac{v(t_n)}{v_{\text{max}}}$

Set $C_e(t_n) = -\Delta v \cdot \sqrt{-2 \cdot \ln \left( \frac{v}{v_f} \right)}$ or $C_e(t_n) = -\Delta v \cdot \exp \left( -\frac{v}{v_0} \right)$

Set $Step1_C_v(t_n) = C_v(t_{n-1}).e^{-\lambda T} + C_e(t_n)$

End for

End for

**Step 2:** Calculate congestion value for lower bound of historical speed data

For each link $l \in G$
For each time interval \( t_n \in T_w \)

Calculate \( \mu \) as the average of all historical data

Calculate \( s/\sqrt{n} \) as the standard deviation of all historical data

Set \( \text{Lower}_B t_n = \mu - Z_\alpha * s/\sqrt{n} \)

Normalize speed between 0 and 1 \( \Rightarrow \frac{\text{Lower}_B t_n}{v_{\max}} \)

Set \( C_v(t_n) = -\Delta v \times \sqrt{-2 \times \ln \left( \frac{\text{Lower}_B t_n}{v_f} \right)} \) or \( C_v(t_n) = -\Delta v \times \exp \left( -\frac{v}{v_o} \right) \)

Set \( \text{Step2}_C v(t_n) = C_v(t_{n-1}) \times e^{-\lambda T} + C_v(t_n) \)

End for

End for

**Step 3:** Compare Step 1 and Step 2 results

For each link \( l \in G \)

For each time interval \( t_n \in T_w \)

If \( \text{Step1}_C v(t_n) > \text{Step2}_C v(t_n) \) and \( \text{Step1}_C v(t_n) > 0 \)

Status: Congested because of collision

If \( \text{Step1}_C v(t_n) < \text{Step2}_C v(t_n) \) and \( \text{Step1}_C v(t_n) > 0 \)

Status: Congested because of daily congestion
If Step1 \( C_v(t_n) < 0 \)

Status: Uncongested

5.4.1 \( \lambda \) and CQA

As shown in Table 5-2, several parameters need to be calibrated before performing the algorithm. \( \lambda \), as the decay constant, defines how much the congestion in the previous time steps affects the congestion in the time intervals ahead. In other words, \( \lambda \) represents how long the effect of congestion remains in the network. The percentage of the remaining congestion after passing the number of half-life is illustrated in Table 5-3. The relationship between \( \lambda \) and half_life \( (t_{1/2}) \) is as follows:

\[
t_{1/2} = \frac{\ln(2)}{\lambda}
\]

(5-10)

By choosing different values for \( \lambda \), the congestion residual effect can be controlled. For example, by choosing \( \lambda = 0.4 \), \( t_{1/2} = \frac{\ln(2)}{0.4} = 1.73 \) hours, which means that after 1.73 hours and 1.73*2 hours, the remaining effect of congestion becomes 50% and 25% of the initial value, respectively.

Table 5-3 The remaining traffic congestion percentages after passing the half_life

<table>
<thead>
<tr>
<th>Number of half-lives elapsed</th>
<th>Fraction remaining</th>
<th>Percentage remaining</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( \frac{1}{1} )</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>( \frac{1}{2} )</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>( \frac{1}{4} )</td>
<td>25</td>
</tr>
</tbody>
</table>
In Figure 5-5, congestion values of a road segment associated with one day for various values of $\lambda$ are calculated. As equation (5-10) demonstrates, the recovery time from congestion $t_{1/2}$ decreases as $\lambda$ increases. In other words, the congestion effect is present for less time on the road segments, and congestion dissipates faster as $\lambda$ increases ($t_{1/2}$ decreases). For instance, in Figure 5-5, the congestion peak happens at 12:30 PM. When $\lambda = 2.8$, congestion dissipates immediately. However, by decreasing the $\lambda$ value, the congestion value returns more slowly to its initial value. Therefore, choosing a proper $\lambda$ is crucial. Hossain et al. (2020) utilized the same residual effect to model congestion. They assigned $\lambda$ the value of 3 for rush hour and for 2 hours off-peak. However, this thesis examines the congestion effect of collisions rather than congestion during peak or non-peak hours. Consequently, the value equal to the duration of a collision is used for $\lambda$, which is reasonable because assuming that congestion starts to dissipate once a collision is cleared and that the effects of congestion persist in the network, but less so, after the clearance time is reasonable. Therefore, if $t_{1/2}$ equals the collision duration, congestion is assumed to decrease to half of its initial value after one half-life and to a quarter after two half-lives.

\[
\begin{array}{ccc}
3 & \frac{1}{8} & 12.5 \\
4 & \frac{1}{16} & 6.25 \\
5 & \frac{1}{32} & 3.125 \\
6 & \frac{1}{64} & 1.15625 \\
7 & \frac{1}{128} & 0.78125 \\
\vdots & \vdots & \vdots \\
n & \frac{1}{2^n} & \frac{100}{2^n} \\
\end{array}
\]
Figure 5-5 Sensitivity analysis for different decay constant rates
5.4.2 $\alpha$ and CQA

As discussed in Table 5-2, $\alpha$ controls the significance level used to distinguish congestion that results from non-recurring and recurring events. Depending on the level of confidence of the model, different values of $\alpha$ are selected. As $\alpha$ increases, the difference between the mean and the lower and upper bounds also increases, which means that the probability of congestion caused by non-recurring events becomes smaller. In this study, a 95% confidence interval value is considered.

5.5 Case Study

In this section, the proposed CQA is tested on a real case study network. The model is tested on Calgary downtown, a dense urban area. The road segments of this collision and their associated IDs are depicted in Figure 5-6. The collision start time is 12:10 and continues until 12:40. The segment corresponding to the collision is marked with a red star. Moreover, the segment IDs of each segment are presented in the label in Figure 5-6. To test the model, first, hypothesis testing is applied for all time intervals during the collision. Hypothesis testing provides the lower bound speed. Then, CQA is applied to each road segment of graph $G$ for collision #1. In this step, the congestion value corresponding to the predicted speed during the collision and the lower bound speed calculated in the hypothesis testing section are calculated and compared for each road segment. When the congestion value is greater than zero, and congestion value is higher than the
congestion value corresponding to the lower bound speed, the segment is identified as congested due to a non-recurring event (e.g., collision).

Figure 5-6 Accident impact area collision #1
5.5.1 Calculating lower bound speed using hypothesis testing

In this section, the speed profiles of five road segments affected by collision #1 that occurred in downtown Calgary have been chosen as samples to demonstrate how lower bound speed is calculated and interpreted. Each segments ID is indicated at the top of each figure.
Previous studies compared the mean value of usual speed and the current speed to identify congested road segments. If the current speed is below the mean value in a given time interval, the road segment is considered congested. However, as shown in Figure 5-7, daily traffic without a disruptive event can fall below the typical speed line. For example, in Figure 5-7 (b), the speed values in all time intervals are below the mean historical speed values. Therefore, this comparative measure is not able to detect congestion and distinguish congestion caused by different sources. Consequently, this thesis uses a 95% confidence interval to draw a boundary for the lower bound of the hypothesis testing and distinguish non-recurring congestion from recurring congestion. Each point below the lower bound line (orange line) represents congestion caused by collisions. These points are shown inside the red oval.

Previous studies detect congestion resulting from collisions once speed falls below a certain threshold. However, this approach does not accommodate the within day fluctuations of the mean
speed value during different times of day. For example, in Figure 5-7 (b), the speed at 11:00 AM and 13:15 PM is around 10 km/h. However, the speed at 11:00 AM is below the lower bound, and the speed at 13:15 PM is far above the lower bound for that specific time interval. Therefore, the speed at 11:00 AM is significantly different from the typical daily traffic conditions at that time, but the speed at 13:15 PM is not significantly different from the daily normal traffic conditions at that time. This situation happens since the mean speed at 13:15 PM is less than the mean speed at 11:00 AM. In other words, daily congestion increases around 13:15 PM, which decreases the mean speed value. Therefore, it is necessary to define a modular minimum value for each time interval to distinguish the sources of congestion.

The reported collision time in Figure 5-7 is between 12:10 PM and 12:40 PM. Figure 5-7 (a) shows that speed starts decreasing at 12:10 PM and reaches a minimum value at 12:50 PM, and it starts reverting to normal conditions between 13:00 PM to 13:20 PM. In Figure 5-7 (c), the reduction in speed starts at 12:10 PM and stays below the lower bound, with few fluctuations, for more than 4 consecutive time intervals. This pattern is similar to the binary approach on highways, which defines congestion that occurs when speed falls below the threshold for 4 consecutive time steps. Therefore, the segment is shown to be affected by collision. However, the lower bound speed before 12:00 PM is not stabilized, which means that speed is not affected by collision congestion. This approach is in the category of binary approaches, and it needs information and analysis to interpret congestion since it does not define the values for congestion directly. In addition, this approach cannot show the intensity of congestion. In contrast, CQA defines the values for congestion, which allows for easy interpretation of congestion, and it measures the intensity.
5.5.2 Analyzing congestion with CQA

The predicted speed value during a collision and the lower bound speed for all segments in graph G (collision impact area) have been calculated. In this section, the CQA algorithm is used to quantify congestion and identify the congestion trends and congestion intensity. The CQA is applied to the same road segments in collision #1 to discuss the results.

As illustrated in Table 5-2, CQA is applied to both predicted speed and lower bound speed during the collision. Therefore, the difference between these two values is extracted to represent the effect of congestion resulting from collisions. In Figure 5-8, the orange and green lines represent the CQA outcomes for the predicted speed and lower bound speed during the collision, respectively. A congestion value greater than zero means that the road becomes congested due to an anomaly, and if the congestion value corresponding to the predicted speed is greater than the congestion value corresponding to the lower bound speed, then the segment is affected by the collision. All road segments in Figure 5-8 show congestion trends. In Figure 5-8 (d) and (e), the CQA algorithm generates different results from those of the hypothesis testing algorithm, which are indicated by the red ovals in Figure 5-7 (d) and (e), because CQA considers the residual effects of congestion from previous time intervals. For instance, Figure 5-7 (e) shows that the collision resulting from congestion is detected at 12:50 PM, and there is no congestion before and after this time resulting from a collision. However, Figure 5-8 (e) indicates that congestion starts at 12:45 PM, reaches its maximum at 12:55 PM, and recovers to the initial conditions around 13:10 PM. The CQA results do more make sense because speed between 12:50 PM to 13:10 PM in Figure 5-7 (d) is not restored and still is affected by collision congestion.
For all diagrams in Figure 5-8, the Y-axis defines the value of congestion in CQA. As all speeds inside CQA became normalized before calculating the congestion values, the congestion values can be used to compare congestion on all road segments. For example, the maximum congestion value in Figure 5-8 (c) and (e) are around 0.25 and 0.1, respectively, which means that segment (c) is affected by the collision more than segment (e) is affected. This comparison is reasonable as segment (c) is in the same direction as the segment that is subject to the collision, which is shown in Figure 5-6. Comparing values allows for the ranking of segments based on the congestion during each time interval.

Therefore, the CQA method provides more information compared to the binary approaches algorithms. First, CQA generates continuous values for congestion, which enables the model to track congestion increases and decreases during each time interval. This feature also provides information about whether a given road segment becomes congested or not. In contrast, hypothesis testing can only distinguish between congestion sources, but it needs the binary approach to identify whether congestion happened or not. Second, CQA directly generates congestion values; in contrast, hypothesis testing needs expert knowledge to interpret congestion from speed. Thus, a non-technical audience can easily understand and interpret CQA. Third, CQA shows the recovery period of congestion after a collision. In other words, CQA defines three states for collisions: start, the maximum point, and recovery. Fourth, CQA allows for the comparison of congestion intensity on road segments.
Figure 5-8 CQA testing on five road segments collision #1
5.5.3 Visualizing congestion propagation prediction

The reported collision occurrence time of collision #1 is 12:10 PM, and it ended at 12:40 PM. The location of the accident is identified by the star shape in Figure 5-9 (a). The color on each road segment represents the level of congestion caused by non-recurring event. The color of the road indicates the state of the road segment. The results are presented below in five minutes intervals from 12:00 PM to 1:35 PM.
As shown in Figure 5-9, congestion from the collision starts at 12:20 PM. This start time is reasonable as road segments becomes congested several minutes after a collision. Therefore, the collision happened at 12:10 PM, and it took 10 minutes for the first segments to be affected. Congestion propagated until police cleared the collision. In other words, the intensity of congestion started increasing at 12:10 PM, and the network became fully congested at 12:40 PM, which was the incident clearance time. The congestion in the network recovered to its initial conditions at 13:35 PM as congestion decreased in the network after the collision was cleared.
The congestion started propagating in the vertical stream at 12:20 PM, but it started at 12:30 PM in the horizontal stream with lower intensity. In other words, congestion affected the upstream road segments in the vertical direction sooner than the upstream road segments in the horizontal direction. This result is logical because the collision happened on a vertical road segment. The downstream segment ID: 442981041 was blocked by the collision, but the downstream segment ID: 431212995 was free. Congestion was resolved in the horizontal and vertical streams at 13:15 PM and 13:35 PM, respectively, which was also reasonable because the vertical segments were affected more by the collision. Consequently, congestion dissipation takes more time on the vertical road segments (e.g., segment ID: 442981041) compared to the horizontal segments (e.g., segment ID: 431212995).

Moreover, congestion propagates in all directions on the upstream and downstream road segments. But, it affects the downstream less than the upstream. For instance, segment ID: 442981041 (the first level vertical upstream) was affected from 12:20 PM until 1:35 PM with sustained high intensity (as red indicates); however, segment ID: 431212995 (the first level horizontal upstream) was affected from 12:30 PM until 13:35 PM with lower intensity (as orange and yellow indicate).

Inside the network, the links that are farther away from the collision experience less congestion. For instance, the fourth level downstream segment ID:442981037 was yellow at its maximum congestion, but the first level downstream segment ID: 442981041 was red during the entire collision timeline.
6. SUMMARY AND CONCLUSION

This research predicts congestion propagation patterns that result from non-recurring events at the network level. This research is divided into two parts. In the first part, a spatiotemporal speed prediction algorithm is developed to predict speed in all neighboring road segments in the vicinity of a collision. In the second part, the congestion quantification algorithm (CQA) is developed to identify the impact area that consists of the set of roads that becomes congested during the accident. This chapter presents concluding remarks in sections (6.1) and (6.2), which correspond to parts one and two. Section (6.3) discusses the contributions of this work. Lastly, future research suggestions are discussed in section (6.4).

6.1 A spatiotemporal framework to predict traffic speed

A new prediction model with four learning layers was developed to predict speed on road segments in the vicinity of a collision. This prediction model utilized road connectivity information, recent speed information, historical speed information, and weather information to predict speed during collisions. This model was tested on 152 collisions that occurred in downtown Calgary between 2014 and 2019. The prediction error for this case study was reasonably accurate (MAE of 2.98, which was comparable to the range of previous works, such as Yin et al., 2020b, who reviewed all traffic prediction models). While this model was validated during collisions, the previous models were validated during normal traffic conditions. Collisions cause more complex traffic dynamics and increase the error terms compared to normal conditions.

While this thesis mainly focused on predicting speed during collisions, the developed methods can also be used to predict speed during recurring events. In terms of scalability, the prediction model can be applied to any part of the City of Calgary and to any other city worldwide. This model can
also be applied to all road types as it considers road segments as a graph. However, the model requires training for each city separately, which means that it is not easily transferable.

### 6.2 Congestion quantification algorithm to model network congestion

A novel congestion quantification algorithm (CQA) was developed to generate continuous congestion values. CQA utilizes a combination of change in speed and current speed values to quantify congestion. Thus, this model can be used to model each road segment's congestion level during recurring and non-recurring incidents. In addition, hypothesis testing is implemented inside CQA to distinguish recurring from non-recurring sources of congestion. Comparing the recurring and non-recurring congestion values from CQA enables the model to capture the direct effects of collisions in road networks. Thus, the predicted speed in the first section is used in the CQA algorithm to predict congestion propagation at the network level.

This model was tested during a collision in downtown Calgary. The results of the model showed that the time that the first road segment was affected by the collision was approximately the same time as was reported by the City of Calgary. In addition, the model showed that congestion cleared 45 minutes after the reported collision clearance time. The model, moreover, ranked segments based on the intensity of the congestion. In terms of scalability, the CQA model can be applied to all road types and during recurring and non-recurring events. Finally, this framework can be used to provide re-routing guidance in navigation systems to avoid congested roads.

### 6.3 Research contributions

The research contributions are as follows:
• Developed a speed prediction model to predict speed during recurring and non-recurring events
• Defined a novel formulation to quantify congestion using change in speed and current speed value
• Developed a congestion quantification formula that can be applied to all road types
• Developed a novel CQA that can carry out the following:
  o distinguishes recurring congestion from non-recurring congestion
  o identifies the three phases of congestion: congestion start time, the time congestion reaches its maximum, and recovery time
  o determines the intensity of congestion, which allows for comparison of congestion on road segments
• Developed a congestion propagation prediction framework that predicts the following:
  o congestion propagation during non-recurring events
  o congestion propagation at the network level
  o congestion growth and recovery
• Developed a congestion propagation prediction framework that evaluates congestion propagation at the road segment level

6.4 Limitations and future work

These results of the prediction framework show that error terms increase for outliers. To solve this problem, future studies should define a customized loss function to improve model prediction for outliers. In addition, this study predicts speed for a five-minute horizon. Future studies should consider a larger horizon window for prediction, which would enable navigation systems to
propose more robust re-routing plans for vehicles passing through the collision impact area. In addition, future works can develop online incident detection algorithms (IDAs) to detect collisions based on the results from the CQA algorithm for arterial networks. Moreover, future research can develop a re-routing guidance algorithm that can use the results of this study to assess the impact of re-routing on congestion formation and intensity.
REFERENCES


APPENDIX

A.1 Training the neural network

The training process contains four main parts: 1) feedforward calculation, 2) loss function calculation, 3) backpropagation calculation, and 4) optimization. The feedforward process explained in chapter 4. This section elaborates on the rest of parts.

In the training process, the objective is to minimize the difference between actual values and predicted values. Therefore, the parameters of the model are adjusted using data in the training dataset so that the best set of parameters is identified: i.e., the weights and bias of the model are chosen so that a smaller error in prediction is generated. In this way, the training process involves running the model in both directions. First, the model makes a prediction using the current model parameters (initial random values) by feeding data into the model. Then, the predicted values are compared with the actual values, and the error function is calculated. Afterward, the partial derivative of the error function (gradient) is computed for each layer. In the next step, the parameters of the model in the opposite direction to the gradient are updated to minimize the error function value. Once the model's error rate converges or the epochs reach a defined maximum number, the model is prepared for prediction. In other words, in each training step, the error function is computed, and the parameters of the neural network are adjusted by a backpropagating procedure from the output layer to the input layer; thus, the error in the next iteration is reduced. The backpropagation algorithm works based on a gradient descent technique to train neural networks.
A.1.1 Loss function

The loss function is the most important part of training the neural network. It determines how accurate the predictions of the model are. A high loss means the model is not accurate, and a low loss indicates the model performs well. Choosing the right type of loss function leads to accurate and more stable results. The loss function can be classified based on the type of output, and it can be either categorical or continuous. In this appendix, only the loss function for continuous values is discussed.

A.1.1.1 Mean absolute error (MAE)

MAE is the absolute difference between actual and predicted values. MAE is not used for outlier prediction as it only considers linear effects based on the difference between real labels and predicted value. For example, ten units deviation have twice the impact as five units deviation. In addition, the optimal prediction is the median target value. In MAE, the gradient magnitude is independent of the error size, and it depends on the sign of the real value minus predicted value. In other words, the gradient magnitude is sensitive to small error values; however, it can lead to convergence problems. The MAE formulation is as follows:

\[
MAE = \frac{1}{n} \sum_{1}^{n} |\hat{Y} - Y|
\]  

(1)

where \(\hat{Y}\) represents the predicted value, \(Y\) is the real value, and \(n\) is the number of samples.
A.1.1.2 Mean absolute percentage error (MAPE)

MAPE is similar to MAE, but the error percentage is calculated by dividing the absolute error value by the real value:

\[
MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{Y} - Y|}{Y}
\]  

(2)

where \(\hat{Y}\) represents the predicted value, \(Y\) is the true value, and \(n\) is the number of samples.

A.1.1.3 Mean squared error (MSE)

MSE is the mean of the squared difference between the actual and predicted value and the most commonly used loss function for regression. MSE is sensitive towards outliers and gives several examples with the same input feature values; the optimal prediction is the mean target value:

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{Y} - Y)^2
\]  

(3)

where \(\hat{Y}\) represents the predicted value, \(Y\) is the real value, and \(n\) is the number of samples.

A.1.1.4 Root mean squared error (RMSE)

RMSE is calculated using the square root of the average squared differences between the predicted and real value. This method, in contrast to MAE, gives a relatively high weight to large errors since the errors are squared before they are averaged. Consequently, RMSE is more useful when large errors are particularly undesirable.
\[ MSE = \sqrt{\frac{1}{n} \sum_{1}^{n} (\hat{Y} - Y)^2} \] (4)

A.1.2 Backpropagation (BP) algorithm

As discussed, the backpropagation algorithm is applied during the training process to adjust the model parameters to minimize the error. BP computes the partial derivative of the cost function over the learnable parameters, i.e., weight and bias. The sign and the number of this derivative are used in optimization algorithms to update parameters to reduce error (the loss function). Due to the complexity of BP for large networks, a simplified model is discussed below. Suppose there are two layers of a network Figure 1:

![Diagram](image)

Figure 1: Forward and backward propagation

L is the layer’s number, and \( a \) is the output of each layer and the input of the next layer; \( w \) and \( b \) are the model’s parameters that need to be optimized. The blue arrays represent
forward propagation, and the red arrays depict backward propagation. Forward propagation is defined below:

\[ z^L = w^{L-1} \cdot a^{L-1} + b^{L-1} \]  
\[ a^L = \sigma(z^L) \]  

Next, how much \( C \) changes as the model’s parameters change needs to be determined. In other words, the derivative of the cost function (\( C \)) over \( w \) and \( b \) must be calculated. The chain rule is used to calculate \( \frac{\partial C}{\partial w_{L-1}} \):

\[ \frac{\partial C}{\partial w_{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial w_{L-1}} \]  

\[ \frac{\partial C}{\partial a^L} = 2(a^L - y) \]  
\[ \frac{\partial a^L}{\partial z^L} = \sigma(z^L) \]  
\[ \frac{\partial z^L}{\partial w_{L-1}} = a^{L-1} \]

\[ \Rightarrow \frac{\partial C}{\partial w_{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial w_{L-1}} = 2(a^L - y) \cdot \sigma(z^L) \cdot a^{L-1} \]  

The cost function gradient is \( 2(a^L - y) \). The second term is the gradient of the activation function (sigmoid) of the final layer, and w.r.t is the weighted sum of the final layer. The last part Equation 8 is the derivative of the weighted sum with respect to the weight that connects two layers (\( L-1 \)) and (\( L \)). Then, the computed gradients are passed to Equation 9 to update the weight.
A.1.3 Optimization algorithms

Gradient descent technique is an optimization method that finds the maxima or minima of a differentiable function. It minimizes the loss function by updating the parameters iteratively using the direction of the gradient of the loss function with respect to the parameters, as shown in the equation below:

\[ W = W - \eta \times \text{gradient}(f, D, W) \]  \hspace{1cm} (9)

where \( \eta \) is the learning and represents how much the parameters are updated in each step. Choosing a value for \( \eta \) that is either too small or too large affects the results. In addition, \( f \) and \( D \) are the loss function and training samples in each iteration, respectively.

There are three types of gradient descents, and they differ in terms of the size of data used to update the parameters. Vanilla gradient descent calculates the gradient of the cost function with respect to the parameters using an entire training dataset. This method is prolonged when the size of a dataset is large. Stochastic gradient descent (SGD) updates the parameters for each training sample. For large datasets, SGD solves the optimization problem by performing an update at each iteration, which results in faster learning, but the frequent updates can cause the cost function to fluctuate more. Third type of gradient decent, is Mini-batch gradient descent which updates the parameters for each mini-batch training sample, which results in a smaller memory requirement and leads to more stable convergence.