# Freeway traffic incident detection using large scale traffic data and cameras

Pranamesh Chakraborty
*Iowa State University*

# Freeway traffic incident detection using large scale traffic data and cameras

by

**Pranamesh Chakraborty**

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Civil, Construction, and Environmental Engineering

Iowa State University

Ames, Iowa

2019

## DEDICATION

I would like to dedicate this dissertation to the memory of one of my first teachers, Sanat Chowdhury, who was instrumental in teaching me mathematics and helping me to grow my passion towards engineering in my early life. This work is also dedicated to my parents and sisters who have always supported me throughout my entire life and also during this dissertation.

# TABLE OF CONTENTS

**Page**

LIST OF TABLES . . . . . . . . . . v

LIST OF FIGURES . . . . . . . . . . vi

ACKNOWLEDGMENTS . . . . . . . . . . viii

ABSTRACT . . . . . . . . . . x

CHAPTER 1. INTRODUCTION . . . . . . . . . . 1
- 1.1 Problem Statement . . . . . . . . . . 1
- 1.2 Research Objectives . . . . . . . . . . 3
- 1.3 Research Methodology . . . . . . . . . . 3
- 1.4 Organization of Dissertation . . . . . . . . . . 6

CHAPTER 2. REVIEW OF LITERATURE . . . . . . . . . . 8
- 2.1 Introduction . . . . . . . . . . 8
- 2.2 Large-Scale Traffic Speed Data Driven Incident Detection . . . . . . . . . . 8
- 2.3 Traffic Congestion Detection from Camera Images . . . . . . . . . . 11
- 2.4 Freeway Incident Detection from Camera Videos . . . . . . . . . . 14
  - 2.4.1 Object Detection . . . . . . . . . . 16
  - 2.4.2 Multi-Object Tracking . . . . . . . . . . 18

CHAPTER 3. LARGE SCALE TRAFFIC SPEED DATA DRIVEN INCIDENT DETECTION 21
- 3.1 Introduction . . . . . . . . . . 21
- 3.2 Methodology . . . . . . . . . . 23
  - 3.2.1 Setup . . . . . . . . . . 23
  - 3.2.2 Univariate Threshold Computation . . . . . . . . . . 25
  - 3.2.3 Multivariate Spatio-Temporal Threshold Denoising . . . . . . . . . . 28
  - 3.2.4 Overall AID framework . . . . . . . . . . 33
  - 3.2.5 Performance Measures . . . . . . . . . . 34
- 3.3 Data Description . . . . . . . . . . 36
- 3.4 Results . . . . . . . . . . 38
  - 3.4.1 Univariate Thresholds Computation . . . . . . . . . . 38
  - 3.4.2 Multivariate Spatio-Temporal Threshold Denoising . . . . . . . . . . 42
- 3.5 Conclusions . . . . . . . . . . 49

CHAPTER 4. TRAFFIC CONGESTION DETECTION FROM CAMERA IMAGES . . . . 51
4.1 Introduction . . . . 51
4.2 Methodology . . . . 52
4.2.1 Deep convolutional neural networks (DCCNs) . . . . 52
4.2.2 YOLO (You Only Look Once) . . . . 54
4.2.3 Support Vector Machine (SVM) . . . . 56
4.3 Data Description . . . . 56
4.4 Results . . . . 59
4.4.1 Sensitivity Analysis . . . . 62
4.4.2 Real-time Implementation . . . . 63
4.5 Conclusions . . . . 66

CHAPTER 5. FREEWAY INCIDENT DETECTION FROM CAMERA VIDEOS . . . . 67
5.1 Introduction . . . . 67
5.2 Methodology . . . . 68
5.2.1 Trajectory-based Incident Detection . . . . 68
5.2.2 Pixel-based Incident Detection . . . . 74
5.3 Data Description . . . . 77
5.4 Results . . . . 79
5.4.1 Trajectory-based Incident Detection . . . . 79
5.4.2 Pixel-based Incident Detection . . . . 84
5.5 Conclusions . . . . 89

CHAPTER 6. CONCLUSIONS . . . . 90
6.1 Large-Scale Traffic Speed Data-Driven Incident Detection . . . . 90
6.2 Congestion Detection from Camera Images . . . . 92
6.3 Freeway Incident Detection from Camera Videos . . . . 93
6.4 Limitations and Future Work . . . . 94

BIBLIOGRAPHY . . . . 96

# LIST OF TABLES

| | | Page |
|---|---|---|
| Table 4.1 | DCNN model architecture used for congestion detection from images . . . . | 53 |
| Table 4.2 | YOLO model architecture used for congestion detection from images . . . . | 55 |
| Table 4.3 | Precision, recall and accuracy values obtained from the three algorithms for congestion detection from images . . . . . . . . . . . . . . . . . . . . . . . . | 61 |
| Table 5.1 | Trajectory-based incident detection performance on non-congested test videos | 83 |
| Table 5.2 | Pixel-based incident detection performance on congested and non-congested test videos . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 85 |

# LIST OF FIGURES

| | | Page |
|---|---|---|
| Figure 3.1 | Sample speed threshold heatmap (mph) | 30 |
| Figure 3.2 | Illustration of the AID algorithm framework | 34 |
| Figure 3.3 | Study region in Des Moines, Iowa | 36 |
| Figure 3.5 | Variation of (a) DR, (b) FAR, (c) MTTD, and (d) PI for different threshold constants $c$ for SND, MAD, and IQD methods | 40 |
| Figure 3.6 | Sample speed thresholds of SND, MAD, and IQD for a particular segment in a given weekday | 41 |
| Figure 3.8 | Variation of (a) DR, (b) FAR, (c) FAR count, (d) MTTD, and (e) PI for different threshold constants $c$ for SND, MAD, and IQD methods | 43 |
| Figure 3.10 | Speed threshold heatmap in mph (a) without denoising, (b) bilateral filter denoising, and (c) total variation denoising | 45 |
| Figure 3.12 | Contour plots showing variation of (a) DR (%), (b) FAR(%), (c) MTTD (mins), and (d) PI with $\sigma_s$ and $\sigma_r/\sigma_i$ | 46 |
| Figure 3.14 | Variation of (a) DR (%), (b) FAR(%), (c) MTTD (mins), and (d) PI with denoising parameter, $\sigma_s$ for bilateral filter and $\lambda$ for total variation | 48 |
| Figure 3.16 | Hourly change in $DR$ due to speed threshold denoising by (a) total variation and (b) bilateral filter | 49 |
| Figure 3.18 | Hourly change in $FAR$ due to speed threshold denoising by (a) total variation and (b) bilateral filter | 50 |
| Figure 4.1 | Original and smoothed occupancy of Wavetronix using Wavelet transform (db2 level 6) | 57 |
| Figure 4.3 | Congestion detection classification examples: (a-c) true positives, (d-f) false positives, (g-i) false negatives, (j-l): true negatives from YOLO, DCNN, and SVM respectively | 61 |
| Figure 4.5 | ROC curves under different prevalent conditions obtained from (a) YOLO; (b) DCNN; (c) SVM; and, (d) all conditions combined for each algorithm. | 64 |
| Figure 4.7 | (a) Sensor occupancy data and congestion alarm from a camera on a particular date; (b) - (c) Camera images of callouts (i-ii) shown in a. | 65 |
| Figure 5.1 | Confidence score prediction of bounding boxes by YOLO, with colors and bounding box widths indicating confidence score probabilities | 69 |
| Figure 5.3 | Sample images of vehicle detections using YOLOv3 object detector | 80 |
| Figure 5.4 | Vehicle detection and tracking sample | 81 |
| Figure 5.5 | Accuracy of the algorithms on number of labeled samples | 83 |
| Figure 5.6 | Incident and normal trajectories labeled by CPLE algorithm for 3 incident videos | 84 |
| Figure 5.7 | Sample images of stalled vehicle detected across 3 frames taken at 1-second interval | 84 |
| Figure 5.9 | Sample images of optical flow estimation using Gunnar-Farneback algorithm | 86 |

Figure 5.11 Sample images of pixel-based incident detection across 3 frames (a) Frame 210, (b) Frame 240, and (c) Frame 270 at 1-second interval in non-congested conditions . . . . 87

Figure 5.13 Sample images of pixel-based incident detection across 3 frames (a) Frame 210, (b) Frame 240, and (c) Frame 270 at 1-second interval in congested conditions . . . . 88

# ACKNOWLEDGMENTS

Acknowledging the persons whose contribution have helped you to shape your dream is difficult to express in words. However, I take this opportunity to thank them from the bottom my heart.

I would like to express my sincere gratitude to my major professors Dr. Anuj Sharma and Dr. Chinmay Hegde who showed me the way to think like a true researcher. Working 'with' them rather than 'under' them during my disserrtation helped me to grow up as a thinker. Their approach to solve any problem is something I would like to follow in my future life. Whatever I could achieve in future, I owe a lot to them.

I would also like to thank my committee members Dr. Jing Dong, Dr. Christopher Day, and Dr. Lily Wang for their continuous guidance throughout this research. In this context, I acknowledge the efforts of all the teachers of my entire lifetime who have showed me the path to follow.

Quality of any work depends on its place of origin. I am fortunate enough to share my workplace with some of my best friends and colleagues I have ever made. The discussions, debate, fun that I had done with them is something that have made my dissertation work more enjoyable.

The four years that I have spent in ISU are undoubtedly one of the best phase of my life. And its credibility goes to all the friends that I have made in this tenure. The moments that I have shared with Dikshant, Amrita, Animesh, Subhadipto, Anjana, Sneha, Saransh, Ashirwad, Sayani, and all other friends will remain in my life forever. They were like my family away from home. Life in ISU wouldn't have been so special without all my friends. And so a big thanks to ISU and Ames for giving such a great environment to me.

Not only this dissertation, but the entire 'me' would have been impossible without my parents and sisters. The support that they have given me in every part of my life, during my ups and downs have made it possible for me to reach this level and to continue further in my life. The sacrifices that they have given for my sake is something for which the word 'thank you' is too short to be

said. Surely, the best gifts that the Almighty gives to each of us in this world are our parents. So at last, Thank you Almighty for giving me this life to live in this beautiful world.

## ABSTRACT

Automatic incident detection (AID) is crucial for reducing non-recurrent congestion caused by traffic incidents. In this paper, a data-driven AID framework is proposed that can leverage large-scale historical traffic speed data along with the inherent topology of the traffic networks to obtain robust traffic patterns. Such traffic patterns can be compared with the real-time traffic data to detect traffic incidents in the road network. Our AID framework consists of two basic steps for traffic pattern estimation. First, we estimate a robust univariate speed threshold using historical traffic information from individual sensors. This step can be parallelized using MapReduce framework thereby making it feasible to implement the framework over large networks. Our study shows that such robust thresholds can improve incident detection performance significantly compared to traditional threshold determination. Second, we leverage the knowledge of the topology of the road network to construct threshold heatmaps and perform image denoising to obtain spatio-temporally denoised thresholds. We used two image denoising techniques, bilateral filtering and total variation for this purpose. Our study shows that overall AID performance can be improved significantly using bilateral filter denoising compared to the noisy thresholds or thresholds obtained using total variation denoising.

The second research objective involved detecting traffic congestion from camera images. Two modern deep learning techniques, the traditional deep convolutional neural network (DCNN) and you only look once (YOLO) models, were used to detect traffic congestion from camera images. A shallow model, support vector machine (SVM) was also used for comparison and to determine the improvements that might be obtained using costly GPU techniques. The YOLO model achieved the highest accuracy of 91.2%, followed by the DCNN model with an accuracy of 90.2%; 85% of images were correctly classified by the SVM model. Congestion regions located far away from the camera, single-lane blockages, and glare issues were found to affect the accuracy of the models. Sensitivity analysis showed that all of the algorithms were found to perform well in daytime conditions, but nighttime conditions were found to affect the accuracy of the vision system. However, for all conditions, the areas under the curve (AUCs) were found to be greater than 0.9 for the deep models. This result shows that the models performed well in challenging conditions as well.

The third and final part of this study aimed at detecting traffic incidents from CCTV videos. We approached the incident detection problem using trajectory-based approach for non-congested conditions and pixel-based approach for congested conditions. Typically, incident detection from cameras has been approached using either supervised or unsupervised algorithms. A major hindrance in the application of supervised techniques for incident detection is the lack of a sufficient number of incident videos and the labor-intensive, costly annotation tasks involved in the preparation of a labeled dataset. In this study, we approached the incident detection problem using semi-supervised techniques. Maximum likelihood estimation-based contrastive pessimistic likelihood estimation (CPLE) was used for trajectory classification and identification of incident trajectories. Vehicle detection was performed using state-of-the-art deep learning-based YOLOv3, and simple online real-time tracking (SORT) was used for tracking. Results showed that CPLE-based trajectory classification outperformed the traditional semi-supervised techniques (self learning and label spreading) and its supervised counterpart by a significant margin. For pixel-based incident detection, we used a novel Histogram of Optical Flow Magnitude (HOFM) feature descriptor to detect incident vehicles using SVM classifier based on all vehicles detected by YOLOv3 object detector. We show in this study that this approach can handle both congested and non-congested conditions. However, trajectory-based approach works considerably faster (45 fps compared to 1.4 fps) and also achieves better accuracy compared to pixel-based approach for non-congested conditions. Therefore, for optimal resource usage, trajectory-based approach can be used for non-congested traffic conditions while for congested conditions, pixel-based approach can be used.

# CHAPTER 1. INTRODUCTION

## 1.1 Problem Statement

Real time traffic incident detection, is a key component to reduce incident-related congestion and secondary crashes alleviate the waste of vehicles' fuel and passengers' time as well as to provide appropriate information in an earliest time frame for the field operation troops including highway maintenance, police department, and emergency group, in addition to the infrastructures and vehicles that would be connected and telecommunicated in near future. Traffic congestion has been defined by US Department of Transportation (USDOT) as "one of the single largest threats" to the economic prosperity of the nation (Owens et al., 2010). The cost of congestion in the year 2014 was calculated to be $160 billion for the top 471 urban areas in the United States. This included 6.9 billion hours of wasted time and 3.1 billion gallons of wasted fuel (Schrank et al., 2015). A major contributor to this congestion are traffic incidents. Schrank and Lomax (2007) showed that implementation of improved incident management procedures in 272 out of 439 urban areas resulted in reduction of 143.3 million hours of incident-related congestion and $3.06 million.

Early detection of incident is one of key step for improved incident management. Hence, significant efforts have been devoted in the past for development of accurate and fast automatic incident detection (AID) algorithms. Researchers have used pattern recognition algorithms, outlier mining methods, artificial neural networks, fuzzy set theory, genetic algorithms, wavelet transformation and other machine learning methods for traffic incident detection (4). However, a nationwide survey on deployment of AID algorithms in Traffic Management Centers (TMC) showed that 90% of survey respondents feel that the current AID algorithms are inappropriate for use either in present (70%) or in future (20%) (Williams and Guin, 2007). The two major reasons behind disabling of AID algorithms in TMCs are difficulty in algorithm calibrations and unacceptable false alarm rates when deployed in large scale. The complicated and time consuming calibration of AID algorithms make it difficult to use them by local TMC personnel. Thus, there is a significant need to revisit the AID algorithms and develop an algorithm which can address these major issues.

Automation of calibration process of AID algorithms can resolve one of the major hindrances of deployment of AID algorithms in TMCs. However, as pointed out by Castro-Neto et al. (2012), development of an incident dataset with accurate start and end time of incidents is time- consuming and often requires manual investigation. This makes the calibration of AID algorithms even more difficult for TMC personnels. In this paper, the main goal is to develop an AID algorithm that can extract maximum information from the traffic data to generate the normal travel pattern of each segment. Thereafter, the anomalous behaviour can be classified as incidents and hence sidestep the need for algorithm training with incident dataset. In the era of big data, traffic parameters (e.g. speed, volume, etc.) are stored for each and every segment across 24 $\times$ 7 hours and 365 days. For example, in Iowa State, probe vehicle data of 23,000 segments spread across the entire state are archived every day in one minute interval. This results in generation of approximately five gigabytes of daily traffic data, which in turn produce around two terabytes of traffic data in an annual basis. And, for traffic incident detection, traffic data needs to be collected and processed continuously for each segment. With the cheap data storage technologies now available, it makes more sense to store the entire dataset and use it to gain useful insights on the performance of the road network. These insights can help in developing more efficient AID algorithms. Thus, incident detection turns out to be an important field in the area of transportation which can get direct benefits from the big data analytics.

Also, with the widespread use of mobile phones and video surveillance systems, incident detection time has been reported to have decreased significantly in recent years, particularly in urban conditions. In general, it has been found that incidents are usually reported within 2 minutes and seldom within more than 5 minutes (Yang et al., 2018). However, this reporting mostly relies on either calls from people directly involved in the incidents or manual inspection of hundreds of cameras installed on the freeways, which hinders the scalability and reliability of the detection system. Cameras, however, when used for automatically detecting traffic anomalies, can report such anomalies within seconds. In the 2018 and 2019 AI City Challenge, traffic incident detection times were reported to be within 3 to 10 seconds (Naphade et al., 2018, 2019).

With the recent advancements in deep learning techniques and improvements in object detection accuracies from videos and images (Han et al., 2018), cameras installed on freeways can also be used to automatically detect traffic anomalies in significantly less time than other data sources. State departments of transportation (DOTs) typically install these roadside cameras on freeways and arterials for surveillance tasks such as incident detection. These cameras, when used effectively, can serve as useful sources for detecting traffic anomalies.

## 1.2 Research Objectives

This study aims at developing an efficient AID framework that can use the wealth of information available from large-scale traffic data, images and videos from closed circuit television (CCTV) cameras for quicker and accurate detection of traffic incidents in freeways. This study is divided into three broad research objectives.

1. Data-driven parallelizable traffic incident detection using spatio-temporally denoised robust thresholds
2. Deep-learning based traffic congestion detection in freeways using traffic camera images
3. Freeway traffic incident detection in congested and non-congested conditions using traffic camera videos

Each of these objectives and the research methodologies adopted are briefly discussed in the next section.

## 1.3 Research Methodology

**1. Data-driven parallelizable traffic incident detection using spatio-temporally denoised robust thresholds**

With the recent advancements in traffic data collection and data storage technologies, fixed sensors installed on roads or probe vehicles can provide useful information on the real-time traffic state over vast networks. One of the popular approach for detecting traffic incidents is to learn the traffic patterns using the past accumulated traffic data obtained from these data sources and detect incidents using real-time traffic data when they behave significantly different from the patterns (Dudek et al., 1974; Kamran and Haas, 2007; Zhang et al., 2016). The primary objective of the first part of this study is to propose and implement a massively parallelizable framework of freeway AID algorithm which can utilize the traffic information obtained from each sensor along with the inherent topology of the traffic network to obtain robust traffic pattern estimates. Such traffic patterns can be used to detect traffic incidents by comparing the real-time traffic data with the corresponding pattern and detecting the anomalies. Further, such a framework can be easily extended over large highway networks due to the its inherent parallelizable framework. This incident detection framework consists of two steps:

(a) *Univariate speed threshold determination:* This involves determination of robust summary statistics (thresholds) of each univariate time series resulting from each road-segment.

(b) *Multivariate spatio-temporal threshold denoising:* The thresholds determined in the previous step are now denoised using the spatio-temporal correlations of the adjacent thresholds.

We show in this study what improvements can be made in incident detection performance with such spatio-temporally denoised robust summary statistics.

**2. Deep-learning based traffic congestion detection in freeways from camera images**

State DOTs traditionally use sensor data and probe vehicle data for traffic state estimation. However, they have also installed a large number of roadside cameras on freeways and arterials for surveillance tasks such as incident detection. The second part of this study aims at developing a system that can detect traffic congestion from CCTV camera images. These cameras are used by traffic incident managers, who can zoom, tilt, and pan the cameras according to their need. Hence, the use of cameras for traffic-state estimation or congestion detection involves additional challenges due to frequent camera movement, which can alter the default calibrations. However, algorithms shouldn't rely on the exact placement of cameras and should be able to accurately detect traffic conditions for different placement scenarios. In this study, we used camera images from different locations, orientations, and weather conditions to successfully detect traffic congestion. Three different models are used for congestion detection tasks. Two of them are deep neural networks: deep convolution neural networks (DCNNs) and you only look once (YOLO) (Redmon et al., 2016). Because these models require time-consuming and costly Graphical Processing Unit (GPU) training, the support vector machine (SVM), a shallow learning model, is used as a comparison to determine the advantages of using deep models.

### 3. Freeway traffic incident detection in congested and non-congested conditions using traffic cameras

The third and final part of this study goes one step beyond congestion detection from images and attempts to detect traffic incidents from videos. We adopted a semi-supervised learning approach for trajectory classification to detect traffic incidents from videos during non-congested conditions. Traffic incident detection from videos using trajectory information comprises of 3 basic tasks: (a) vehicle detection (b) vehicle tracking and trajectory formation, and (c) trajectory classification. We used deep-learning based object detector, YOLO, to detect vehicles in video frames and kalman filtering based tracker SORT (Simple Online Realtime Tracking), to track the vehicles. Finally, the trajectories are classified into incident and non-incident ones using a semi-supervised classifier. This helps to do away with the step of manually annotating vehicle tracks in a video stream to prepare training dataset which is extremely labor intensive, expensive, and not scalable. Our experimental results on traffic video data provided by the Iowa DoT demonstrate that our framework achieves superior performance compared to supervised learning techniques with a comparable amount of labeled examples.

However, such trajectory-based incident detection cannot perform well in congested traffic conditions due to difficulties in tracking individual vehicles in crowded conditions (Li et al., 2014). Therefore, optical flow based approach is proposed in this study to detect traffic incidents in congested conditions. Traditional approaches in anomaly detection in crowded conditions involve detecting abnormal shapes or abnormal motions (Li et al., 2014). However, traffic incident detection involves vehicles (i.e., known shapes) rather than abnormal shapes. Therefore, we used deep-learning based object detector, YOLO, to detect vehicles in video frames and then supervised learning classifiers are used to detect abnormal motions among the detected vehicles. We used 'histogram of magnitude' of optical flow for each detected vehicle and then abnormal vehicles and tracked, using SORT tracker. We show that such optical flow based approach can detect incidents in both congested and non-congested conditions. However, due to computationally intensive optical flow calculation involved in this approach (less than 5 frames per second), we propose to use semi-supervised trajectory based incident detection during non-congested conditions and optical flow based incident detection during congested traffic conditions only. Congested traffic conditions are determined using the proposed congestion detection classifier, as described in our second research objective (Chapter 4).

## 1.4 Organization of Dissertation

This dissertation is divided into six chapters. Chapter I gives a brief introduction of the study problem, the three research objectives this study focuses on and the methodologies adopted to solve them. Chapter II provides literature review of the three research objectives. Chapter III provides detailed description of the first objective which deals with multivariate denoising of thresholds for freeway incident detection from large scale traffic data. The details of the methodology, data, results, and a short conclusion to the problem is provided in this chapter. Similarly, Chapter IV describes the details of the second objective of this study, detection of traffic congestion from camera images. Chapter V describes the methodology, data details, results, and a brief conclusion on the third research objective, incident detection from traffic camera videos for congested and non-congested conditions. Finally, Chapter VI provides conclusion on the work done in this study, limitations, and the future work that are planned to be done.

# CHAPTER 2. REVIEW OF LITERATURE

## 2.1 Introduction

Quicker traffic incident detection in freeways is critical for providing rapid incident response. Studies have shown that every seven minutes of delay in incident verification leads to an additional mile of queue build-up, thereby increasing the likelihood of secondary incidents (Wells and Toffin, 2005). Improved procedures for incident management resulted in reduction of $3.06 million and 143.3 million hours of incident-related congestion (Schrank and Lomax, 2007). Hence, significant efforts have been devoted towards development of accurate and fast automatic incident detection (AID) algorithms. Traditional AID algorithms rely on radar-based sensor data(Shi and Abdel-Aty, 2015), loop detector data (Xu et al., 2016), probe vehicle data (Asakura et al., 2015), cameras (Yuan et al., 2017), or fusing multiple multiple streams (Dia and Thomas, 2011; Houbraken et al., 2015) to detect traffic incidents from data streams. We discuss next the relevant literature on each of our three research objectives followed by the scope of work done to complement the existing research.

## 2.2 Large-Scale Traffic Speed Data Driven Incident Detection

Freeway traffic incidents are usually classified as anomalies or outliers in the traffic stream. AID algorithms aim to detect such anomalies using real-time traffic data along with historical data (whenever available). AID algorithms can be primarily classified into two categories based on the methodology adopted:

1. Real-time traffic data is compared with the traffic data observed in the immediate past (i.e., over the previous $T$ intervals) to detect abnormalities which can be classified as incidents.

2. Historical traffic data is used to generate "normal" traffic patterns, and significant deviation from the normal patterns are classified as incidents.

Several existing algorithms compare real-time traffic conditions with immediate past conditions to detect traffic incidents. Parkany and Bernstein (1995) detected traffic incidents based on the principle that when traffic conditions switch from incident-free to incident conditions, frequent lane switch maneuvers and temporal and spatial discrepancies in headway and travel conditions can be observed. Hellinga and Knapp (2000) proposed the Waterloo algorithm where it is assumed that travel-time is log-normally distributed and normal travel pattern is estimated from the traffic data of past $T$ intervals. Li and McDonald (2005) proposed the bivariate analysis model (BEAM) where the travel time difference between adjacent travel time intervals are used for detecting incidents. On the other hand, Zhu et al. (2009) used both spatially and temporally adjacent time intervals as features for incident detection purposes. Li et al. (2013) used weighted average and standard deviation of past $T$ intervals of traffic parameter values to detect traffic incidents from the data stream. False alarms due to fluctuations in traffic variables were handled by replacing the mean and standard deviation of current time intervals with those of the previous time intervals when the coefficient of variation of traffic variable was found to be less than a predetermined threshold. Recently, Asakura et al. (2017) used shock wave theory to probe-vehicle trajectories to detect traffic incidents. Although all the above algorithms have been used extensively for incident detection, they do not utilize the wealth of information available from past historical data. Rather, they rely only on the real-time and immediate past traffic data to detect the abnormalities in the data stream.

Using only real-time and immediate past traffic data for incident detection helps in avoiding the issues in storing and processing large scale traffic data. However, with the recent advancements in data storage and data processing technologies (Zhang et al., 2017b), it makes more sense to utilize information from historical traffic data to develop efficient AID algorithms. Along. For example, Sethi et al. (1995) used linear discriminant analysis to learn the linear relationship of predictor variables from historical traffic data that can differentiate between the incident and non-incident conditions. Balke et al. (1996) used standard normal deviates (SND) to generate confidence intervals for normal traffic conditions. Significant deviations from normal conditions can be labeled as traffic incidents. Historic average travel times and their normal deviates were determined by the time of day and day of the week for each road segment to denote incident-free conditions. Since mean and standard normal deviates are known to be prone to outliers, Balke et al. (1996) used an incident dataset for removing the outliers from historic data before summary statistics computation. So, this method also requires traffic incident dataset with accurate start and end time of the incidents (which are often hard to get) for incident detection. Snelder et al. (2013) used median instead of mean in order to achieve *robust* summary statistics computation and used it as a reference case to find out delays caused by incidents.

SND and such similar methods do not take into consideration the spatio-temporal relationship of the road network to generate normal traffic conditions. However, Chung (2011) applied SND over spatio-temporally connected cells to determine incident-induced delay. Chung and Recker (2012) further extended the method using an optimization method to determine optimal hyper-parameter $c$ in SND method ($\mu - c \times \sigma$, where $\mu$ denotes the mean and $\sigma$ denotes the standard deviation) to estimate the spatio-temporal extent of incidents. These studies, however, do not consider the spatio-temporal correlation of the thresholds. They rather use topological and temporal information to determine the impact region of traffic incidents based on the thresholds generated from univariate time-series. Similarly, Anbaroglu et al. (2014) used spatio-temporal clustering for detecting non-recurrent traffic congestion when link journey time of spatio-temporally connected cells exceed pre-determined threshold computed from historical data. Chen et al. (2016) also used spatio-temporal clustering to identify recurrent and non-recurrent congestion, their spatio-temporal extent and quantify the delay caused by such incidents. Similarly, Zhang et al. (2016) used dictionary-based compression theory for identifying the spatio-temporal traffic patterns at the detector, intersection, and sub-region level which can be used for anomaly identification.

With the rapid advancement in deep learning techniques and their success in supervised and unsupervised classification and anomaly detection problems, significant research has also been performed using such state-of-the-art techniques for traffic incident detection too. Convolution neural networks (CNN), deep belief networks (DBN), autoencoders, long short-term memory networks (LSTM) and their variations have been used for detecting incidents from spatio-temporal traffic data (Yuan et al., 2018; Hashemi and Abdelghany, 2018; Chen et al., 2016; Zhu et al., 2018), social media data (Chen et al., 2018; Zhang et al., 2018; Amin-Naseri et al., 2018), and traffic cameras (Singh and Mohan, 2018; Chakraborty et al., 2018b,a). Zhu et al. (2018) represented spatio-temporally correlated traffic data as images and used CNN for detecting incidents in traffic networks. A similar analysis was also performed by Yuan et al. (2018) to predict incidents using convolutional long short-term memory networks (Hetero ConvLSTM) model.

Thus, significant research has been conducted to develop accurate AID algorithms using the rich information obtained from historical traffic data. Spatio-temporal analysis has also been performed to find out the extent of traffic incidents and their impact regions. This study complements the existing research by extending the AID framework in two different ways. The first part involves the computation of robust summary statistics using parallel computation methods enabling application of the AID algorithm over large-scale traffic network. The second part involves the generation of accurate summary statistics by multivariate denoising which utilizes the spatio-temporal correlation of the parameters obtained from the first part. Our main objective in this study is to develop a data-driven approach which can utilize the large-scale traffic data to learn normal traffic patterns using parallel computation methods. The reduced dataset can be further refined using multivariate denoising techniques to develop more accurate and robust traffic patterns. This enables to implement the AID algorithm over large traffic networks. Overall, we show that our proposed AID framework can achieve improved overall detection performance. We next discuss the past studies done on our second research objective, traffic congestion detection from camera images.

## 2.3 Traffic Congestion Detection from Camera Images

Undoubtedly, dissemination of real-time traffic information to road users can significantly improve the efficiency of traffic networks. Hence, estimating real-time traffic states and thereby detecting network anomalies such as congestion and incidents, have been of significant interest to researchers for the last few decades.

Traditionally, traffic-state estimation is conducted using point-based sensors, including inductive loops, piezoelectric sensors, and magnetic loops (Kotzenmacher et al., 2004). Recent advances in active infra-red/laser radar sensors have led to these devices gradually replacing the traditional point-based sensors (Zhong and Liu, 2007). Also, with the increasing usage of navigation-based GPS devices, probe-based data are emerging as a cost-effective way to collect network-wide traffic data (Feng et al., 2014). Video monitoring and surveillance systems also are used for calculating real-time traffic data (Ozkurt and Camci, 2009). Recent advances in image processing techniques have improved vision-based detection accuracy. Deep learning methods, such as convolution neural networks (CNNs), have been able to achieve human-level accuracy in image classification tasks (He et al., 2015). The basic advantage of these methods is that they don't require picking up hand-crafted features and hence can do away with the painstaking calibration tasks needed when using camera images for traffic-state estimation (Bauza et al., 2010).

Studies have also been performed fusing multiple sources of data for traffic state estimation. (Van Lint and Hoogendoorn, 2010) used extended generalized Treiber-Helbing filter for fusing probe-based and sensor based data. (Choi and Chung, 2010) used fuzzy regression and Bayesian pooling technique for estimating link travel times from probe data and sensor data. (Bachmann et al., 2013) investigated several multi-sensor data fusion based techniques to compare their ability to estimate freeway traffic speed. State DOTs also traditionally use sensor data and probe vehicle data for traffic state estimation. However, they have also installed a large number of roadside cameras on freeways and arterials for surveillance tasks such as incident detection. These cameras are used by traffic incident managers, who can zoom, tilt, and pan the cameras according to their need. Hence, the use of cameras for traffic-state estimation or congestion detection involves additional challenges due to frequent camera movement, which can alter the default calibrations. However, algorithms shouldn't rely on the exact placement of cameras and should be able to accurately detect traffic conditions for different placement scenarios.

In this study, we used camera images from different locations, orientations, and weather conditions to successfully detect traffic congestion. Three different models were used for congestion detection tasks. Two of these are deep neural networks: deep convolution neural networks (DCNNs) and you only look once (YOLO). Because these models require time-consuming and costly Graphical Processing Unit (GPU) training, the support vector machine (SVM), a shallow learning model, was used as a comparison to determine the advantages of using deep models.

During the last few decades, significant research efforts have been devoted to using closed-circuit television (CCTV) cameras to determine real-time traffic parameters such as volume, density, and speed (Zhang et al., 2017a; Darwish and Abu Bakar, 2015). These methods can be broadly divided into three categories: (a) detection-based methods, (b) motion-based methods, and (c) holistic approaches.

Detection-based methods use individual video frames to identify and localize vehicles and thereby perform a counting task. (Ozkurt and Camci, 2009) used neural network methods to perform vehicle counting and classification tasks from video records. Kalman filter-based background estimation has also been used to estimate vehicle density (Balcilar and Sönmez, 2008). In addition, faster recurrent convolution neural networks (RCNNs) have been used for traffic density calculation (Ren et al., 2017); however, they were found to perform poorly for videos with low resolution and high occlusion. Recent achievements in deep learning methods in image recognition tasks have led to several such methods being used for traffic counting tasks. (Adu-Gyamfi et al., 2017) used DCNNs for vehicle category classification. (Oñoro-Rubio and López-Sastre, 2016) used two variations of CNNs, namely counting CNN and hydra CNN, to conduct vehicle counting and predict traffic density. Recently, (Zhang et al., 2017a) used both deep learning and optimization-based methods to perform vehicle counts from low frame-rate, high occlusion videos.

Several motion-based methods have been suggested in the literature to estimate traffic flow utilizing vehicle tracking information. (Asmaa et al., 2013) used microscopic parameters extracted using motion detection in a video sequence. However, these methods tend to fail due to lack of motion information and low frame rates of videos; some vehicles appear only once in a video, and hence, it becomes difficult to estimate their trajectories.

Holistic approaches avoid the segmentation of each object. Rather, an analysis is performed on the whole image to estimate the overall traffic state. (Gonçalves et al., 2012) classified traffic videos into different congestion types using spatiotemporal Gabor filters. (Lempitsky and Zisserman, 2010) performed a linear transformation on each pixel feature to estimate the object density in an image; however, this approach was found to perform poorly in videos with a large perspective. Further, both these methods require manual annotation of each object in the images to perform the training of the counting task.

Overall, significant studies have been conducted in the past using various deep and shallow learning models to implement vehicle counting tasks and thereby determine congestion states. In this study, we adopted the holistic approach to label an image as either congested or non-congested. We also did away with counting each vehicle to determine the congestion state. Rather, we assigned labels to the images based on nearby benchmark sensors and then conducted the classification task. Next section provides detailed description of the studies relevant to the third research objective, semi-supervised learning approach for freeway incident detection from videos.

## 2.4 Freeway Incident Detection from Camera Videos

Traffic incident detection approaches from CCTV cameras can be broadly classified into two categories: (a) Explicit Event Recognition and (b) Anomaly Detection.

In explicit event recognition, the explicit knowledge of the events to be identified are used for incident detection. This requires *a priori* knowledge of all the recognizable events, which the associated AID systems use as predefined templates to parse the incoming data for incident detection. For example, (Ravinder et al., 2008) applied video image processing for traffic management where there is non-adherence to lane discipline . (Sadeky et al., 2010) used logistic regression over Histogram of Flow Gradients (HFG) to determine the probability of occurrence of accident in a video sequence. (Hui et al., 2014) used Gaussian Mixture Model (GMM) to detect traffic vehicles and tracked using Mean Shift Algorithm. Then traffic incident alarms were triggered when the velocity or acceleration of the detected vehicles exceed a pre-determined threshold. (Ren et al., 2016) used video based detection for analyzing the traffic state distribution characteristics in a cluster of cells dividing the lanes of a road segment . (Maaloul et al., 2017) used Farneback Optical Flow for motion detection in a video sequence, together with a heuristic threshold-selection approach for accident detection.

The other popular approach to incident detection is based on anomaly detection. In this approach, the system attempts to learn “typical” patterns in the incoming data; any irregularities in the observed data can be classified as an incident. For example, (Lou et al., 2002) used dynamic clustering techniques to cluster the normal trajectories and detect the abnormal ones. (Piciarelli et al., 2008) used one-class Support Vector Machine (SVM) for detecting anomalous trajectories. Recently, (Yuan et al., 2017) performed anomaly detection in traffic scenes using spatially-aware motion reconstruction. Such unsupervised modeling of the video sequences also traditionally involved a combination of sparse coding and bag-of-words (BOG) (Zhao et al., 2011). However, recent developments in deep learning techniques have resulted in new methods of learning normal video patterns and thereby detecting anomalies based on reconstruction error. (Hasan et al., 2016) used a fully convolutional feed-forward autoencoder to learn the spatio-temporal local features and thereby learn the temporal regularity in the video sequences. (Chong and Tay, 2017) also used a combination of spatial feature extractor and temporal sequencer based on Convolutional Long Short Term Memory (ConvLSTM) network for anomaly detection in videos.

The above two categories of traffic incident detection approaches can broadly be termed as *supervised* and *unsupervised* learning techniques. While supervised techniques can, in general, provide better results in detection or classification tasks, the main hindrance in their application is the scarcity of supervised data samples and the cost of manually annotating and labeling the dataset. In particular, manually annotating vehicle tracks in a video stream is extremely labor-intensive, expensive, and not scalable. In the present study, we established a new *trajectory-based* learning framework for traffic incident detection during non-congested traffic conditions using recent advances in semi-supervised learning (Loog, 2016). This framework can achieve the "best of both worlds." A small sample of normal vehicle tracks and the tracks of vehicles involved in an incident were manually annotated, and all other (unlabeled) vehicle tracks were used to improve the performance of the classification. However, with the increase in traffic density, it becomes infeasible to track individual vehicles. Then, *pixel-based approaches* are used to find out abnormal motions. It usually involves background subtraction or optical flow based motion estimation to detect abnormal motions. A brief overview of past research on vehicle detection and multi-object tracking are provided in the following sections.

### 2.4.1 Object Detection

In recent years, the evolution of CNN has resulted in significant improvements in the performance of object detection and classification. Results of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) point to dramatic improvements in object detection, localization, and classification (Russakovsky et al., 2015). Region-based convolutional neural networks (R-CNNs) were among the first modern developments in CNN-based detection (Girshick et al., 2014a). These developments involved cropping externally computed box proposals from an input image and running a neural net classifier on these crops. However, overlapping crops led to significant duplicate computations, which, in turn, led to low processing speeds. The development of Fast R-CNN involved pushing the entire input image only once through a feature extractor and cropping from an intermediate layer (Girshick, 2015). This led to the crops sharing the computation load for feature extraction and thereby increased processing speed.

Recent work has focused on generating box proposals using neural networks instead of relying on the external box proposals used in R-CNN and Fast R-CNN (Szegedy et al., 2013; Erhan et al., 2014; Ren et al., 2017; Redmon et al., 2016). Such approaches involve overlaying a collection of boxes on the image at different locations, aspect ratios, and scales. These boxes are called anchors or priors. Training is then performed to predict the discrete class of each anchor and the offset by which the anchor needs to be shifted to fit the ground truth bounding box. The accuracy and computation time of the object detection algorithm depends significantly on the choice of these anchors.

The following sections discuss four recent architectures for object detection and classification: Faster R-CNN (Ren et al., 2017), single-shot multibox detector (SSD) (Liu et al., 2016b), region-based fully convolutional networks (R-FCNs) (Dai et al., 2016), and YOLO (Redmon et al., 2016).

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

Faster R-CNN performs detection in two stages. Stage 1, called the region proposal network (RPN), involves processing images using a feature extractor (VGG-16), and the class-agnostic box proposals are predicted from the features obtained at some selected intermediate level (conv5).

In Stage 2, features from the same intermediate feature map are extracted using the box proposals and fed to the remainder of the feature extractor to predict the class and the class-specific box refinement for each proposal. Faster R-CNN is the basis on which most subsequent object detection algorithms, including SSD and R-FCN, were developed.

### Single-Shot MultiBox Detector (SSD)

SSD architecture is built on VGG-16 architecture. It uses a single feed-forward convolutional network to predict classes and anchor offsets, thereby evading the requirement for a second-stage per-proposal classification operation. In this approach, the output space of bounding boxes is discretized into a set of default boxes with different object scales and aspect ratios. During prediction, scores for the presence of an object in each default box are generated by the network, and, finally, adjustments are made to the box to match the object shape more accurately.

### Region-Based Fully Convolutional Networks (R-FCN)

R-FCN is fundamentally derived from Faster R-CNN, but it is designed to work much faster than Faster R-CNN. In R-FCN, crops are extracted from the last layer of features prior to prediction instead of cropping features from the layer where region proposals are predicted. This minimizes the per-region computation and has been shown to achieve comparable accuracy to Faster R-CNN with less computation time.

Previous research studies have proposed a position-sensitive cropping mechanism in place of the standard region of interest (ROI) pooling operation (Ren et al., 2017). A detailed comparison of these three algorithms (Faster R-CNN, SSD, and R-FCN), along with the speed-accuracy tradeoffs, can be found in a study by Huang et al. (2017).

#### You Only Look Once (YOLO)

YOLO frames object detection as a regression problem (Redmon et al., 2016). A single neural network is used to predict the bounding boxes and associated class probabilities in a single evaluation over the entire image. Thus, the entire pipeline can be optimized end-to-end based on detection performance. This makes the algorithm very fast, and images can be processed in real-time (45 frames per second [fps]).

### 2.4.2 Multi-Object Tracking

Multi-object tracking (MOT) aims to estimate the states of multiple objects while conserving their identification across time under variations in motion and appearance. This involves determining the locations, velocities, and sizes of the objects across time. With the recent advancements in object detection, tracking-by-detection has emerged as one of the predominant approaches for multi-object tracking. This approach generally involves associating the objects detected across multiple frames in a video sequence. The two broad categories in a tracking-by-detection framework are batch and online tracking.

Batch methods usually involve determining object trajectories in a global optimization problem and processing the entire video at once. Short tracklets are generated; individual detections are linked first, and then the tracklets are associated globally to form the object trajectory. Flow network formulations (Zhang et al., 2018; Berclaz et al., 2011) and probabilistic graphical models (Yang and Nevatia, 2012; Andriyenko et al., 2012) are the two broad classes of algorithms in a batch MOT problem. However, the intensive iterative computation required for generating globally associated tracks and the need for detection of the entire sequence beforehand limits the use of these batch MOT approaches in real-time applications.

Online methods build trajectories sequentially by using information provided up to the present frame and associating the frame-by-frame objects detected. Thus, this approach can be easily implemented for real-time tracking. However, these methods are prone to fragmented trajectory generation under occlusion and object detection errors.

Traditional online MOT methods are multiple hypothesis tracking (MHT) (Reid, 1979; Kim et al., 2015) and joint probabilistic data association filter (JPDAF) (Rezatofighi et al., 2015). The JPDAF method, first proposed by Fortmann et al. (1983), involves generating a single state hypothesis by weighting individual measurements with the association likelihoods. MHT, in contrast, involves tracking all possible hypotheses and then applying pruning schemes for computational tractability. Both of these approaches require significant computational and implementation complexity, thereby limiting their implementation in real-time applications.

Recently, Bewley et al. (2016) proposed simple online real-time tracking (SORT), which performs Kalman filtering in the image space and uses the Hungarian algorithm for frame-by-frame data associations. With a state-of-the-art object detection framework (Ren et al., 2017), SORT ranks higher than MHT in the MOT Challenge dataset (Leal-Taixé et al., 2015). However, SORT is known to perform poorly when state estimation uncertainty is high and is known to return substantially high identity switches.

To overcome this shortcoming, Wojke et al. (2017) proposed the Deep-SORT algorithm, which incorporates both motion and appearance information into the association metric, which, in turn, increases robustness against occlusions or detection errors. Even more recently, Bae and Yoon (2017) proposed a robust online MOT method that uses confidence-based data association for handling track fragmentation and deep appearance learning for handling similar object appearance in tracklet association.

Thus, significant efforts have been made till date using supervised and unsupervised techniques for traffic incident detection. While supervised techniques can in general provide better results in detection or classification tasks, the main hindrance in its application is the scarcity of enough supervised data samples and cost of manually annotating and labeling the dataset. In particular, manually annotating vehicle tracks in a video stream is extremely labor intensive, expensive, and not scalable. In this work, we establish a new learning framework for traffic incident detection using recent advances in semi-supervised learning (Loog, 2016). Via this framework, we try to achieve the "best-of-both-worlds"; we manually annotate only a small sample of normal vehicle tracks and tracks of vehicles involved in an incident, and then we used all other (unlabeled) vehicle tracks to improve the classification performance. For incident detection in congested traffic conditions, we used a novel Histogram of Optical Flow Magnitude (HOFM) feature descriptor to detect incident vehicles using SVM classifier based on all vehicles detected by YOLOv3 object detector. The next chapter provides detailed analysis and results of our first research objective, spatio-temporal threshold denoising techniques for freeway incident detection.

# CHAPTER 3. LARGE SCALE TRAFFIC SPEED DATA DRIVEN INCIDENT DETECTION

## 3.1 Introduction

With the wide-spread usage of mobile phones and video surveillance systems, incident detection time has been reported to reduce significantly in recent times, particularly in urban conditions. In general, incidents are found to be usually reported within 2 minutes and hardly exceeds 5 minutes (Yang et al., 2018). However, this reporting mostly rely on either receiving calls from people directly involved in the incidents or manually inspecting hundreds of cameras installed in the freeways, which hinders the scalability and reliability of the detection system. On the other hand, with the wide-spread usage of mobile phones used in navigation systems, probe vehicle data has emerged out to be a cost-effective way of providing state-wide real-time traffic speed data. Additionally, the state Department of Transportation usually installs fixed sensors (radar-based, loop-based, etc.) in major arterials and freeways for traffic monitoring purpose. These data sources when used effectively, can turn out to be an additional, complementary source for detecting traffic incidents reliably on a large scale. Thus, there lies a significant need for research to develop such a reliable and scalable AID algorithm.

Various data-driven algorithms and statistical models have been used to develop AID algorithms. A popular approach for detecting traffic incidents is to learn the traffic patterns using accumulated traffic data observed in the past and detect incidents when traffic data observed in real-time behaves significantly different from the learned patterns (Dudek et al., 1974; Kamran and Haas, 2007; Zhang et al., 2016). With the recent advancement in traffic data collection and data storage technologies, fixed sensors installed on roads or probe vehicles can provide useful information on the real-time traffic state over vast networks. These data sources, when compared and matched with corresponding historical datasets, can serve as useful indicators of traffic incidents.

However, two major challenges arise in such AID algorithm development. First, it is somewhat hard to model the dynamics of traffic patterns. Recurring congestion events can pair with the non-recurring events at the same time and location thereby making it difficult to separate the true positives (incidents) from false positives (caused by recurring congestion). Second, with the recent advancements in data storage technologies, the scale of traffic data stored from a traffic network makes it difficult to process it in real-time and detect traffic incidents. Considerable amount of computation is involved in parsing the data which makes it difficult for conventional computation methods to handle such massive data sources and develop real-time AID algorithm.

The primary objective of this paper is to propose and implement a massively parallelizable framework of freeway AID algorithm which can utilize the traffic information obtained from each sensor along with the inherent topology of the traffic network to obtain robust traffic pattern estimates. Such traffic patterns can be used to detect traffic incidents by comparing the real-time traffic data with the corresponding pattern and detecting the anomalies. Further, such a framework can be easily extended over large highway networks due to the its inherent parallelizable framework.

The primary building block of this framework is the road network which is subdivided into multiple smaller segments. Each segment produces a time-series of the traffic state (average speed). These time-series are extensively large-scale with thousands of data points being recorded daily for each one of them. So, we perform dimensionality reduction with robust summary statistics of each time-series across non-overlapping time windows. This summary statistics computation can be massively parallelized using MapReduce (Dean and Ghemawat, 2008) thereby making it feasible to apply the framework over large networks. However, this summary statistics computation do not take into account the spatio-temporal correlations of the time windows, and therefore maybe noisy. So, we leverage the knowledge of the topology of the road-network and construct a “heatmap” of the summary statistics. We then perform multivariate denoising of the “heatmap” assuming that the summary statistics of the topologically and temporally adjacent regions are likely to be similar. We show in this study what improvements can be made in incident detection performance with such spatio-temporally denoised robust summary statistics.

This chapter is organized as follows. Section 3.2 describes the methodology adopted in this study followed by the data description in Section 3.3. Section 3.4 provides the details of the results obtained using the proposed methodology. Finally, the conclusion of the study and the scope of future study are provided in Section 6.1.

This work is published in parts in Chakraborty et al. (2017b,a, 2019).

## 3.2 Methodology

Lane-blocking traffic incidents or incidents impacting the traffic stream often result in a significant drop in vehicle speeds and/or increase in the traffic densities. Hence AID algorithms often model traffic incidents as outliers or anomalies in the traffic data stream. The basic objective of the AID algorithm is to detect these anomalies by comparing the real-time traffic data with the immediate past data or with the historical past data. In this study, we extend the popular SND algorithm (Dudek et al., 1974; Balke et al., 1996) of incident detection which uses the historical traffic data to learn traffic pattern. Then the traffic pattern is compared with real-time traffic data to detect the anomalies in the streaming real-time data. Our incident detection framework consists of two steps:

1. *Univariate speed threshold determination:* This involves the determination of robust summary statistics (thresholds) of each univariate time series resulting from each road-segment.

2. *Multivariate spatio-temporal threshold denoising:* The thresholds determined in the previous step are now denoised using the spatio-temporal correlations of the adjacent thresholds.

Next, we set up our model, followed by a detailed description of each of the above two steps.

### 3.2.1 Setup

We first describe the mathematical abstraction of our data corpus. Let the weighted graph of the traffic network is denoted by $G = (S, E, W)$. Here, $S = \{s_i\}_{i=1}^{n}$ denotes the nodes of the graph, $E = \{e_i\}_{i=1}^{m}$ denotes the (undirected) edges, and $W = \{w_i\}_{i=1}^{m}$ denotes the weights of the graph.

In our framework, the nodes of the graphs correspond to the consecutive road-segments into which the freeway under consideration is partitioned, and an average vehicle speed ($x$) is reported each minute for each segment. The nodes corresponding to consecutive segments along the freeway are connected via appropriately weighted edges. In this study, the freeway segments considered are of approximately equal lengths, varying from 0.4 miles to 0.6 miles. Hence, we assume here that the weights of each node are equal to 1 (i.e., unweighted edge segments). However, conceptually these can be extended further to encode other spatial information. For example, unequal length freeway segments can be represented with weights proportional to its length. These weights can be used in the multivariate spatio-temporal threshold denoising, as explained in Section 3.2.3.

The topology of the road network (i.e., the order of the road segments) describes the connectivity of our graph system. We measure a (noisy) time series for each node $s$ on a given day (say $d$) of length $N$:

$$x_s^d = \left( x_s^{t_1,d}, x_s^{t_2,d}, ..., x_s^{t_N,d} \right) \tag{3.1}$$

Here, $t_i$ denotes the $i^{\text{th}}$ time instant, and $d$ denotes the day of the week. The observed time series are synchronized across different nodes, i.e., the time stamps $t_1, t_2, ..., t_N$ are same for all days across all segments.

Overall, we model the time series as a *third-order tensor* $x \in \mathbb{R}_+^{n \times N \times D}$. Here, $D$ denotes different days, $n$ refers to the total number of nodes in the graph, and $N$ denotes the length of the time series. For example, if the average speed of each segment is reported in 1-minute intervals (as in this study), then the length of the time series $N$ will be equal to $24 \times 60 = 1440$. Our objective here is to identify anomalous local patterns in this tensor.

However, a major challenge that we face is the scale of the traffic data. The number of road segments and the sampling rate of the segments are very high, thereby producing millions of traffic records daily. For example, the entire road network of Iowa, divided into approximately 54,000 segments, produce 4 gigabytes of daily traffic speed data, which aggregates to approximately 1.5 terabytes of annual traffic data.

To alleviate this issue, we first pre-process along the second dimension of the tensor. We perform robust summary statistics computation of each univariate time series across non-overlapping windows to determine (scalar) thresholds of each window in the time series, which will serve as key parameters in our AID framework. The details of this step are discussed next.

### 3.2.2 Univariate Threshold Computation

Our basic methodology for univariate threshold computation is based on the popular SND algorithm (Dudek et al., 1974; Balke et al., 1996) for incident detection. The algorithm involves modeling the univariate statistics of each non-overlapping windows of the time series data as a Laplace distribution with location parameter $\mu$ and scale parameter $\zeta$. More specifically, each univariate time series is divided into 15-minute non-overlapping windows ($p$), similar to study by Dudek et al. (1974), and the threshold speed ($\tau$) for each window is determined from the location and scale parameters. The threshold ($\tau_s^{p,d}$) speed is defined as:

$$\tau_s^{p,d} = \mu_s^{p,d} - c \times \zeta_s^{p,d} \tag{3.2}$$

where, the optimum constant $c$ is to be determined from the validation set. This auxiliary reduced tensor, speed threshold ($\tau_s^{p,d}$), can be compared with real-time speed data $X_s^{t_i,d}$ to detect anomalies or traffic incidents.

The SND algorithm uses the mean speed value ($\bar{x}$) as the location parameter ($\mu$) and the corresponding standard deviation ($\sigma$) as the scale parameter ($\zeta$) to model the normal traffic pattern. Normal traffic condition varies depending on the time of day and day of the week. Hence the location and scale parameters are determined for each day of the week, and 15-minute periods of the day for each segment. Threshold speed values over 15-minute intervals of each day of the week are determined using the previous 8 weeks of traffic data for the same segment for the given day of the week and period of the day, similar to the study of Balke et al. (1996). The mean speed values ($\bar{x}_s^{p,d}$), standard deviation $\sigma_s^{p,d}$, and threshold speed $\tau_{s,snd}^{p,d}$ for SND algorithm can be determined as follows:

$$\bar{x}_s^{p,d} = \frac{\sum_{\forall k} x_{k,s}^{p,d}}{\sum_{\forall k} k} \tag{3.3}$$

$$\sigma_s^{p,d} = \frac{1}{\sum_{\forall k} k} \sum_{\forall k \in (d,p,s)} \left( x_{k,s}^{p,d} - \bar{x}_s^{p,d} \right)^2 \tag{3.4}$$

$$\tau_{s,snd}^{p,d} = \bar{x}_s^{p,d} - c_{snd} \times \sigma_s^{p,d} \tag{3.5}$$

Although the SND algorithm is easy to calibrate and inherits good transferability (Li et al., 2013), a major drawback of the algorithm is that it is prone to outliers or anomalies (Balke et al., 1996). Both location and scale parameters (mean and standard deviation) are known to be highly susceptible to outliers (Pearson, 2005). This causes a severe challenge for traffic datasets that contain anomalies (in the form of incidents), and can lead to misrepresentation of the underlying normal traffic patterns due to these anomalies. One possible way to tackle this issue is to manually remove all time intervals that are known incidents (e.g. by referring to incident reports), before calculating the aforementioned threshold parameters. This requires complete knowledge of the traffic incident data during both training, testing, and implementation; however, it is often difficult to get accurate reports of incidents (Ren et al., 2012). In particular, it is hard to obtain accurate estimates for start time, duration, and impacted regions due to traffic incidents (Yue et al., 2016). To alleviate this issue, we propose to use *alternate* robust summary statistics (learned from the time series tensor itself) for the location and scale parameters so that the affect of the presence of anomalies can be minimized.

Outlier detection is an important task in statistical analysis and significant research has been performed for development of robust models to detect outliers from noisy data streams (Pearson, 2005; Aggarwal, 2007). Interested readers can refer to Gupta et al. (2014) for a detailed review of current outlier detection methods. In this study, we propose to use two such modifications of the SND summary statistics to calculate robust location and scale parameters. Specifically, we replace the standard deviation values by maximum absolute deviation ($MAD$) (Hampel, 1974) and inter-quartile deviation ($IQD$) for the scale parameter ($\zeta$). For both these methods, we use median speed values instead of mean values as location parameter ($\mu$). Studies have shown that median, IQD, and MAD provide more robust summary statistics compared to mean and standard deviation for data containing outliers (Pearson, 2005; Leys et al., 2013). Thus, the two modified univariate threshold computation techniques can now be expressed as:

1. MAD algorithm: $\mu$ = Median ($M$), $\zeta$ = Maximum Absolute Deviation ($MAD$),

$$\tau_{s,mad}^{p,d} = M_s^{p,d} - c_{mad} \times MAD_s^{p,d} \tag{3.6}$$

2. IQD algorithm: $\mu$ = Median ($M$), $\zeta$ = Inter-Quartile Distance ($IQD$),

$$\tau_{s,iqd}^{p,d} = M_s^{p,d} - c_{iqd} \times IQD_s^{p,d} \tag{3.7}$$

The median ($M$), maximum absolute deviation ($MAD$), and inter-quartile deviation ($IQD$) can be calculated as follows:

$$M_s^{p,d} = median_{\forall k}\left(x_{k,s}^{p,d}\right) \tag{3.8}$$

$$MAD_s^{p,d} = median_{\forall k}\left|x_{k,s}^{p,d} - M_s^{p,d}\right| \tag{3.9}$$

$$IQD_s^{p,d} = \left\{x_{k,s}^{p,d} : \Pr\left(X \leq x; \forall k\right) = 0.75\right\} - \left\{x_{k,s}^{p,d} : \Pr\left(X \leq x; \forall k\right) = 0.25\right\} \tag{3.10}$$

Although MAD and IQD are known to be robust to outliers, one of the major drawbacks of these statistics is they are susceptible to "swamping" problems where non-outliers are classified as outliers resulting in a significant number of false alarms. This implies that if more than 50% of the data values ($x_k$) are very similar (i.e., close to swamping breakdown point), then IQD and MAD are both equal to zero. So, any value different from median will be reported as an outlier. This has important consequences for incident detection because traffic data is usually concentrated along a particular value. For example, a freeway segment with speed limit of 70 mph will mostly report speed values around 70 mph. So, when more than 50% of speed values are equal to 70 mph, the $\zeta$ parameter (i.e., MAD or IQD) will be equal to zero and hence a speed value of even 69 mph will be reported as an outlier. This will result in a significant increase in the number of false alarms.

However, in traffic incident detection problems, we can take advantage of the fact that the capacity-reducing traffic incidents will significantly impact the traffic conditions and result in congested traffic conditions. To recall, AID algorithms relying on macroscopic traffic data solely (instead of cameras, probe vehicle trajectories or similar data sources) can only detect incidents which impact traffic flow and result in significant reduction in the capacity. So, the incident alarm can be triggered only when congested conditions exist and the observed speed is lower than the expected threshold (given by Equation 3.2). Federal Highway Administration guidelines state that congested conditions occur in freeways when average speed in a road-segment is less than 45 mph (Systematics, 2005). So, the modified threshold speed value can now be written as:

$$\tau_s^{p,d} = Min\left[45, \mu_s^{p,d} - c \times \zeta_s^{p,d}\right] \tag{3.11}$$

One of the major advantage of these summary statistics (mean, median, MAD, and IQD) is that their computation can be easily parallelized over multiple systems thereby enabling the AID framework to handle massively large datasets over wide traffic network. In this study, we used Apache Pig (2018), a Hadoop MapReduce framework for computation of the summary statistics values from raw traffic speed data. The optimal constant parameters, $c_{snd}, c_{mad}$, and $c_{iqd}$) in Equations 3.5, 3.6, and 3.7 respectively are determined using the incident validation data set. Also, similar to past studies (Ren et al., 2012; Li et al., 2013), we perform *persistence test* before triggering incident alarm. This means an incident alarm is triggered when observed speed values are lower than the threshold speed value (Equation 3.11) for three consecutive intervals. This is done to reduce false alarms due to spurious noisy low speed values.

### 3.2.3 Multivariate Spatio-Temporal Threshold Denoising

Univariate threshold computation, given by Equation 3.11, does not take into consideration the topology of the traffic network or temporal correlations between the time windows. Hence, the thresholds computed can be highly noisy and variable across contiguous roadway segments. To compensate for this, we propose to improve the quality of the estimated thresholds using the spatio-temporal information. We leverage both the underlying topology of the road network and the temporal coherence of the time windows and perform denoising to obtain improved estimates of the speed threshold values depicting the normal traffic pattern.

First, we obtain a "heatmap" of the threshold speeds calculated using Equation 3.11. Figure 3.1 shows a sample speed threshold heatmap. The road segments are arranged according to their corresponding mileage, along with temporally consecutive time windows to form the heatmap. We argue that the spatially and temporally coherent time windows are likely to exhibit similar thresholds. So, we formulate our objective as to obtain a coherent, smoothed threshold heatmap by performing denoising of the raw heatmap. In this study, we used two specific procedures of image denoising techniques — bilateral filter (Tomasi and Manduchi, 1998) and total variation Rudin et al. (1992) — to obtain the denoised threshold map which can be used for improved incident detection performance. We chose these denoising techniques since they are known to preserve the edges during denoising. This is important given the fact the sharp edges in threshold heatmaps often indicate the regions of recurrent congestion (as shown in Figure 3.1) and preserving the edges will help to differentiate the recurrent congestion from non-recurrent congestion events such as traffic incidents. As explained in Section 3.2.1, we assume the weights of each node equal to 1 since the freeway segments are approximately equal length. For unequal length segments, weighted bilateral filter and weighted total variation can be used. Interested readers can refer to (Anantrasirichai et al., 2014; Liu et al., 2016a) for further details. Next, we discuss the details of bilateral filter and total variation in the context of this study.

#### 3.2.3.1 Bilateral Filter based Threshold Denoising

Bilateral filtering (Tomasi and Manduchi, 1998) is a popular image denoising technique which preserves edges while smoothing. In the most basic formulation, each pixel in an image is replaced by the average of its neighbors, keeping into account the geometric closeness (spatial and temporal correlations) along with the photometric similarity (speed threshold values). As stated in Section 3.2.1, the multivariate time series can be represented as a third-order tensor $x \in \mathbb{R}_{+}^{n\times N\times D}$, where $D$ denotes different days of the week (producing different heatmaps), $n$ refers to the total number

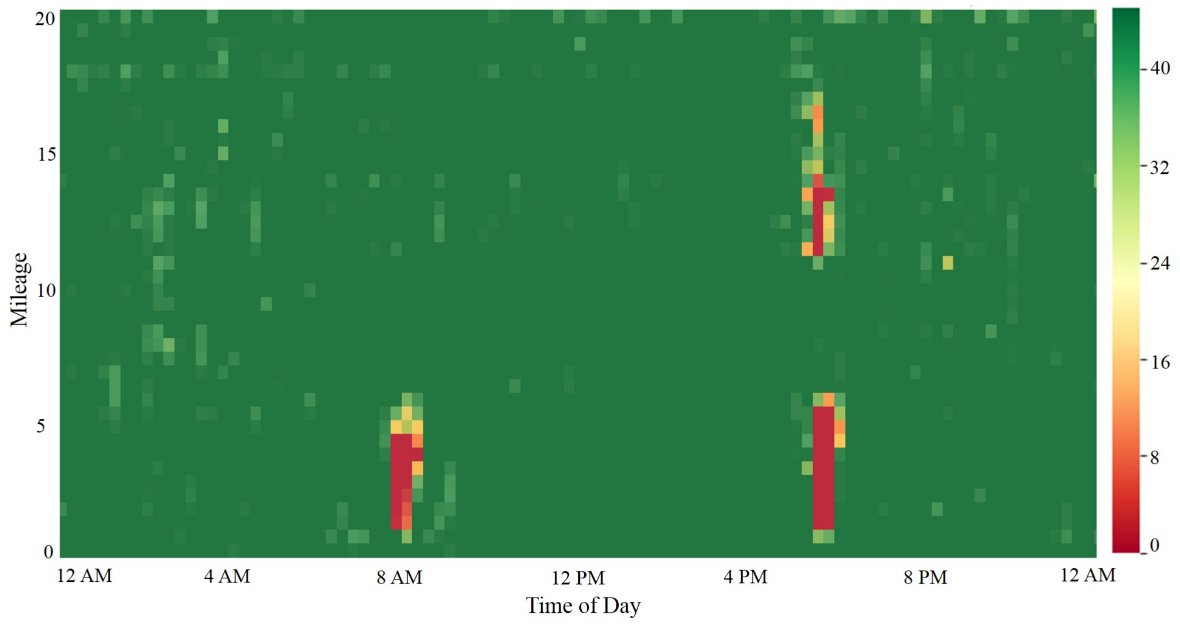


Figure 3.1: Sample speed threshold heatmap (mph)

of nodes in the graph, and $N$ denotes the length of the time series. So, each threshold heatmap can be represented as an image (a second-order tensor, $I : \tau \in \mathbb{R}_{+}^{n \times N'}$). Since we divide the time of the day into 15-minute time windows, so $N' = (60/15) \times 24 = 96$.

The Bilateral Filter, denoted by $BF\,[.]$, can be defined as a function that takes in an image $I$ as input, and returns an image $BF[I]$ whose $p^{\text{th}}$ pixel intensity is given by:

$$BF[I]_p = \frac{\sum\limits_{q \in S} G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(I_p - I_q)\, I_q}{\sum\limits_{q \in S} G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(I_p - I_q)} \tag{3.12}$$

where $S$ and $R$ denotes the space domain (set of possible pixel locations in an image) and range domain (set of possible pixel values) respectively. $G_{\sigma_s}$ is a two-dimensional spatial Gaussian kernel while $G_{\sigma_r}$ is a range Gaussian kernel where $\sigma_s$ and $\sigma_r$ are the spatial and range filtering parameters for the image I. The kernel can be represented as:

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3.13}$$

Thus, $G_{\sigma_s}(\|p-q\|)$, which defines the spatial distance, decreases influence of spatially distant pixels compared to the position $p$. The parameter $\sigma$ defines the extension of the neighborhood.

Similarly, $G_{\sigma_r}(I_p - I_q)$ decreases influence of $q$ pixels with color intensities different from that of $p$ $(I_p)$.

The range parameter $\sigma_r$ differentiates bilateral filter from Gaussian filter which only takes into consideration spatial (location) closeness for smoothing. As $\sigma_r$ increases, the bilateral filter converges to a Gaussian blur filter. As $\sigma_s$ increases, larger features are smoothed. Both spatial and range weights are multiplied in bilateral filter; thus, no smoothing occurs even if one weight approaches zero. For example, a narrow range Gaussian combined with large spatial Gaussian will produce limited smoothing in spite of large spatial extent. The contours are maintained by the range weight. This will help the bilateral filter to remove noisy thresholds from the heatmap while maintaining the sharp edges formed by recurrent congestion patches. Please refer to Paris et al. (2007) for a detailed review.

#### 3.2.3.2 Total Variation based Threshold Denoising

Total variation (TV), first proposed by Rudin et al. (1992), has been used extensively for image denoising problems since they are known to denoise the image without smoothing the object boundaries. Many algorithms have been developed in the past for total variation based image denoising. Rodríguez (2013) provides a detailed review of the different methods developed for TV-denoising. In this study, we adopted the algorithm proposed by Chambolle (2004) for solving the minimization problem of total variation of an image. As explained in Section 3.2.3.1, the image can be expressed as a second-order tensor, given by $I : \tau \in \mathbb{R}_+^{n\times N'}$. Denoting $X = \mathbb{R}_+^{n\times N'}$ and $Y = X \times X$, for any $u \in X$, the gradient $\nabla u \in Y$ is given by

$$(\nabla u)_{i,j} = \left((\nabla u)_{i,j}^x, (\nabla u)_{i,j}^y\right) \tag{3.14}$$

with

$$(\nabla u)_{i,j}^x = \begin{cases} u_{i+1,j} - u_{i,j} & \text{if } i < n, \\ 0 & \text{if } i = n, \end{cases} \tag{3.15}$$

$$(\nabla u)^y_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{if } j < N', \\ 0 & \text{if } j = N', \end{cases} \tag{3.16}$$

Then, the discretized total variation can be defined as

$$J(u) = \sum_{\substack{1 \le i \le n, \\ 1 \le j \le N'}} \left| (\nabla u)_{i,j} \right|_Y \tag{3.17}$$

where the standard Euclidean scalar product is given by

$$\langle p, q \rangle_Y = \sum_{\substack{1 \le i \le n, \\ 1 \le j \le N'}} p^1_{i,j} q^1_{i,j} + p^2_{i,j} q^2_{i,j}$$

for all $p, q \in Y$, $p = (p^1, p^2)$ and $q = (q^1, q^2)$. Then, Equation 3.17 can be written as

$$J(u) = \sup \{ \langle p, \nabla u \rangle_Y : p \in Y, |p_{i,j}| \le 1 \forall i, j \} \tag{3.18}$$

Now, a discrete divergence operator div:$Y \to X$ can be introduced $\forall p \in Y$ and $\forall u \in X$ defined by,

$$\langle -\text{div} p, \nabla u \rangle_X = \langle p, \nabla u \rangle_Y$$

It can be easily shown that the div is given by

$$(\text{div} p)_{i,j} = \begin{cases} p^1_{i,j} - p^1_{i-1,j} & \text{if } 1 < i < n, \\ p^1_{i,j} & \text{if } i = 1, \\ -p^1_{i-1,j} & \text{if } i = n, \end{cases} + \begin{cases} p^2_{i,j} - p^2_{i,j-1} & \text{if } 1 < j < N', \\ p^2_{i,j} & \text{if } j = 1, \\ -p^1_{i,j-1} & \text{if } i = N', \end{cases} \tag{3.19}$$

From the definition of divergence operator and Equation 3.18, we can write

$$J(u) = \sup_{v \in K_F} \langle u, v \rangle_X$$

where $K_F$ is given by

$$\{ \text{div} p :: p \in Y, |p_{i,j}| \le 1 \forall i, j \}$$

It can be shown that determination of nonlinear projection $\pi_{\frac{1}{\lambda} K_F}(f)$ leads to solving the following constrained minimization problem (Chambolle, 2004; Duran et al., 2013).

$$\min \left\{ \|\lambda \text{div} p - f\|_X^2 : p \in Y, |p_{i,j}|^2 - 1 \leq 0, \forall i,j \right\} \tag{3.20}$$

From Karush-Kuhn-Tucker optimality conditions (Hiriart-Urruty and Lemaréchal, 1993; Ciarlet, 1982), we get:

$$-\nabla(\lambda \text{div} p - f)_{i,j} + \alpha_{i,j} p_{i,j} = 0 \tag{3.21}$$

where either $|p_{i,j}| < 1$, $\alpha_{i,j} p_{i,j} = 0$, and $\nabla(\lambda \text{div} p - f)_{i,j} = 0$; or $|p_{i,j}| = 1$ and $\alpha_{i,j} p_{i,j} > 0$. Thus,

$$\alpha_{i,j} p_{i,j} = \left| \nabla(\lambda \text{div} p - f)_{i,j} \right|$$

Chambolle (2004) proposed a semi-implicit gradient descent algorithm for solving the minimization problem. For denoising parameter $\lambda$, tolerance parameter $t$, time-step $\tau$, and while $\max\limits_{\forall i,j} \left\{ \left| p_{i,j}^{n+1} - p_{i,j}^{n} \right| \right\} > t$,

$$p_{i,j}^{n+1} = \frac{p_{i,j}^{n} + \tau(\nabla(\text{div} p^n - f/\lambda))_{i,j}}{1 + \tau \left| (\nabla(\text{div} p^n - f/\lambda))_{i,j} \right|} \tag{3.22}$$

Chambolle (2004) showed that the algorithm converges for $\tau \leq 1/8$. In this study, we chose $t = 0.0002$ and our objective is to determine the optimal denoising parameter $\lambda$. Interested readers can also refer to Duran et al. (2013) for a detailed description of Chambolle's algorithm.

### 3.2.4 Overall AID framework

The flowchart of the proposed AID algorithm framework is shown in Figure 3.2. Historical traffic data are processed weekly in MapReduce for univariate speed thresholds computation of 15-minute windows for each segment and day-of-week. The speed thresholds can be generated using SND, MAD, or IQD method. Next, the thresholds are used to generate speed threshold heatmaps for each road and direction. These heatmaps are denoised using total variation or bilateral filter. The thresholds are matched with real-time speed data and when speed is less than the threshold value for 3 consecutive intervals, an incident alarm is triggered. This persistence test is performed to reduce false alarms generated due to spurious noise in real-time data. While increasing the duration of persistence test will result in further delay in detecting incidents, decreasing the duration can lead to an increase in false alarms due to the noisy real-time speed values. Hence, we used three intervals similar to the previous studies (Ren et al., 2012; Li et al., 2013).

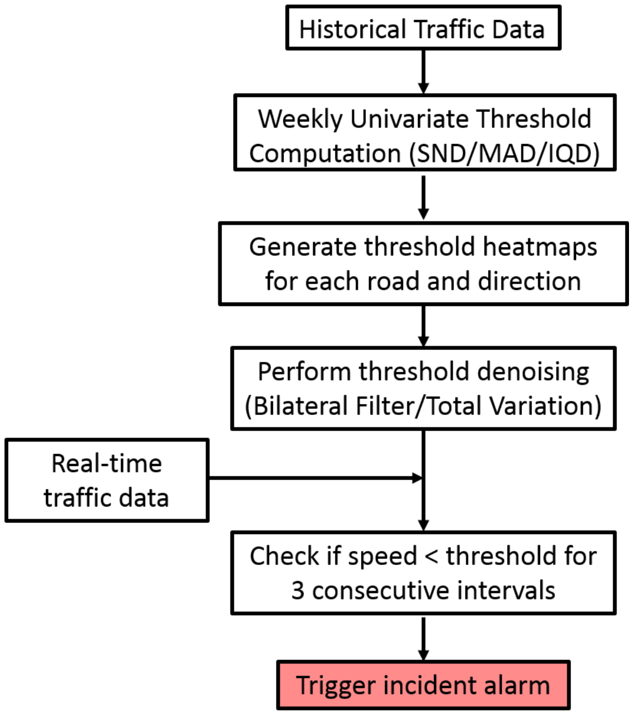

Historical Traffic Data
Weekly Univariate Threshold Computation (SND/MAD/IQD)
Generate threshold heatmaps for each road and direction
Perform threshold denoising (Bilateral Filter/Total Variation)
Real-time traffic data
Check if speed < threshold for 3 consecutive intervals
Trigger incident alarm

Figure 3.2: Illustration of the AID algorithm framework

### 3.2.5 Performance Measures

Incident detection performance has been evaluated in terms of the performance measures used in past studies (Parkany and Xie, 2005; Ren et al., 2012; Li et al., 2013). This involves determination of 4 performance measures:

1. Detection Rate ($DR$) is the ratio of number of incidents detected by AID algorithm to the total number of incidents occurred.

$$DR = \frac{\text{Total number of detected incidents}}{\text{Total number of actual incidents}} \times 100\% \tag{3.23}$$

2. False Alarm Rate ($FAR$) is used for penalizing false calls and is defined as the ratio of the number of false alarms reported to the total number of AID algorithm application. For example,

if traffic data is reported at one-minute interval for a particular segment and AID algorithm is applied for each record, then the number of algorithm application is equal to 60. Thus, if 5 records out of them is reported as false calls, then $FAR$ is equal to $[(5/60) \times 100]\,\% = 8.3\%$.

$$FAR = \frac{\text{Total number of false alarm cases}}{\text{Total number of algorithm applications}} \times 100\% \tag{3.24}$$

Besides this, we also calculate the average daily false alarms. Based on a nationwide survey on traffic management centers (TMC), Williams and Guin (2007) reports that a maximum number of ten false alarms per day is found to acceptable by TMCs. Higher false alarms are found to create negative impacts on TMC operators and managers and impacts the overall efficiency of the AID algorithm. In this study, we maintain this limit of maximum ten false alarms per day and reports whenever the number of daily false alarms are higher than this limit.

3. Mean Time to Detect ($MTTD$) takes into consideration the latency involved in the AID algorithm. It is defined as the average of time elapsed between the actual start of the incident and time when the incident is first detected by the algorithm.

$$MTTD = \frac{\text{Total time elapsed between detecting incidents}}{\text{Total number of incidents detected}} \tag{3.25}$$

4. Performance Index ($PI$), given by Equation 3.26, brings together all 3 performance measures ($DR$, $FAR$, and $MTTD$) into a single measure to find out the overall performance of the AID algorithm. $PI$ is believed to be one of the best possible measures to reflect the performance of AID during model selection (Ren et al., 2012; Weegberg et al., 2010; Cheu et al., 2003). Minimizing $PI$ is the optimization objective used during cross-validation. Since $DR$ can be 100% or $FAR$ can be 0% during training, the $PI$ measure is slightly modified with the constants (1.01 and 0.001) to handle such cases, similar to (Ren et al., 2012). As suggested by Baldi et al. (2000), any attempt to represent multiple measures to a single number will result in loss of information. PI is believed to one of the best measures to portray AID performance during model selection. However, federal and state transportation agencies can either use $PI$

for selecting model parameters or use their individual judgment to strike a balance between $DR$, $FAR$, and $MTTD$.

$$PI = \left(1.01 - \frac{DR}{100}\right) \times \left(\frac{FAR}{100} + 0.001\right) \times MTTD \tag{3.26}$$

## 3.3 Data Description

The data used in the study comprises traffic speed data and crash data from Interstate Freeways I-80/35 and I-235 of the Des Moines region, in Iowa, USA. The Des Moines region experiences the majority of Iowa's freeway congestion (62% of slow traffic events) along with one of the highest concentration of traffic incidents within the state (Kapsch, 2016). Hence, it is a challenging task to separate traffic incidents from recurring congestion events in such a network and develop a reliable AID framework.

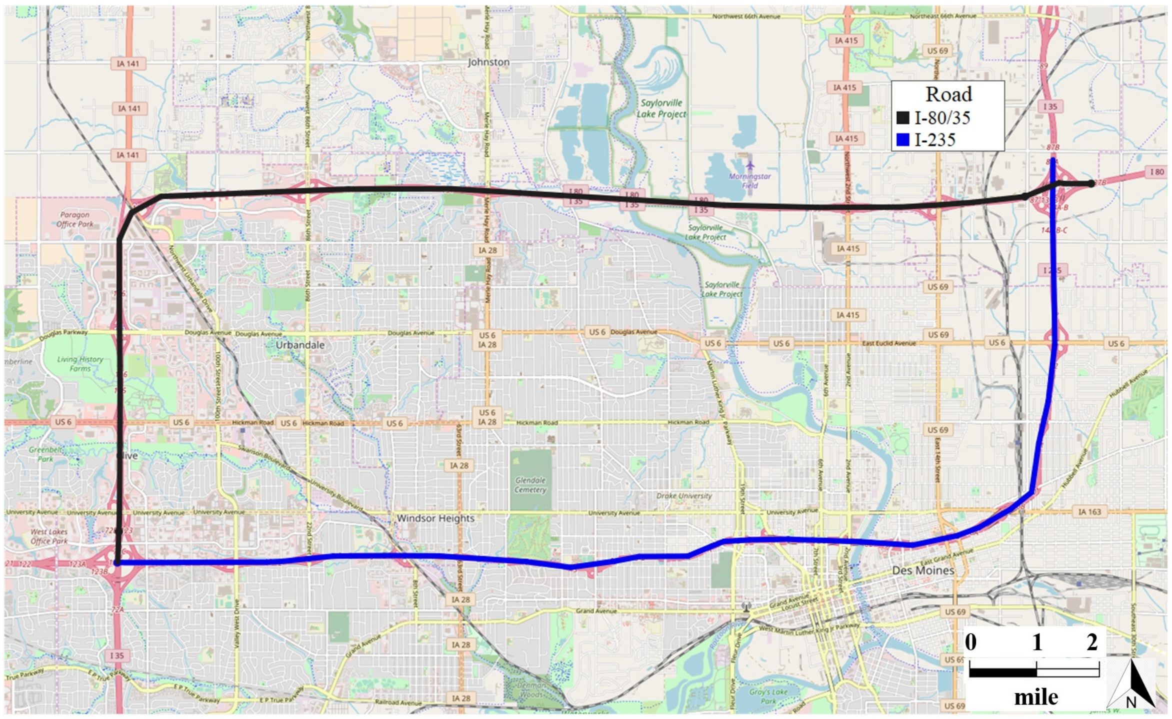


Figure 3.3: Study region in Des Moines, Iowa

Since traffic speed is the primary input in this AID algorithm, the incidents which didn't impact traffic speeds, such as stalled vehicles in shoulders, were not considered in this study. Thus, only lane-blocking incidents which caused an impact to traffic were included in the incident dataset. Each incident was manually verified using nearby cameras. The study period extended from April 2017 to October 2017. A total of 210 lane-blocking traffic incidents were reported during this period in the study region. The incident reports were obtained from the Traffic Management Center records in Ankeny, Iowa. The incident reports included information on the location of the incident (latitude, longitude, road, and direction), start and end time of the incidents, and also the incident type. The following incident classes were included in this study: 1-vehicle crash, 2-vehicle crash, 3+ vehicle crash, stalled vehicle, and debris. Construction reports and slow traffic events (not associated with any traffic incidents) were excluded from the incident database. All the incidents were manually verified with the cameras installed in the freeways.

High-resolution probe-based traffic speed data, provided by INRIX (INRIX, 2018) is used in this study. INRIX uses on-board GPS devices in trucks, taxis, buses, and passenger cars to estimate the real-time travel time and speed of freeways and arterial segments. The entire road network is divided into approximately 0.5 miles long segments and average speed data is reported in 1-minute interval. Since the quality of probe-based speed data depends on the number of probe vehicles available, INRIX reports two parameters, confidence score and c-value, to denote the reliability of each traffic record. Confidence score can take 3 values: 10, 20, and 30. Confidence score of 30 indicates that only real-time probe vehicles are used for reporting real-time speed. On the other hand, confidence score 10 indicates historical traffic speed data and is used to report traffic speed of a segment due to unavailability of probe vehicles. When a mix of real-time probe data and historical speed data is used, then confidence score of 20 is reported. C-value is an additional reliability parameter provided by INRIX only when confidence score is 30. C-value can range from 0-100 and provides a relative measurement of number of probe-vehicles used for real-time speed report. Interested readers can refer to Haghani et al. (2009); Sharma et al. (2017); Ahsani et al. (2018) for further details on INRIX data and its quality assessment. Since traffic incident detection requires real-time speed reports, we used traffic speed data corresponding to confidence score of 30 and c-value greater than 30, as suggested by Haghani et al. (2009). With the considerably high traffic volume in the study region (annual average daily traffic of 70,000 vehicles in 2017), more than 98% of the traffic speed records used in this study were found to be real-time. However, it will be interesting to find out the effects of missing data or the absence of real-time data on incident detection performance. This is important because such missing data can impact the AID performance in both training and implementation phases. Quiroga et al. (2005) and Dia and Rose (1997) performed thorough analyses on finding out the impacts of such data quality issues on incident detection performance. For example, sensitivity analysis can be performed to determine the impacts of such missing data. Also, random noise such as white noise can be inserted into the datasets to determine the stability in performance metrics while handling such noisy data. Besides this, a similar consideration of data quality must be performed when utilizing other types of data sources such as radar-based or loop-based data for developing similar AID algorithms. These data sources include additional traffic variables such as volume and occupancy along with the speed values which can be used for incident detection. Thus, assessing the importance of including or excluding one or more variables on AID performance can give useful insights when the detectors fail to capture all the traffic variables. Future studies can also look into such analyses to determine their effects on AID algorithm performance.

## 3.4 Results

As explained in Section 3.2, our incident detection framework consists of two distinct steps: (a) univariate speed threshold computation and (b) multivariate spatio-temporal speed threshold denoising. Next, we discuss the results obtained in each step.

### 3.4.1 Univariate Thresholds Computation

The first step for incident detection is the determination of speed thresholds from 8 weeks of historical traffic data. This involves determination of the optimum threshold constant $c$ for each the

3 methods, SND, MAD, and IQD (see Equations 3.5, 3.6, and 3.7 respectively). Figure 3.5 shows the variation of the different performance measures ($DR$, $FAR$, $MTTD$, and $PI$) for different values of the threshold constant ($c$) for the 3 AID methods (SND, MAD, and IQD). The main objective here is to determine the best of the three methods, SND, IQD, and MAD and also their corresponding optimum threshold constant $c$. As stated in Section 3.2.5, federal or state transportation agencies can choose the threshold constant ($c$) based on minimum $PI$ obtained here or use their individual judgment to strike a balance between $DR$, $FAR$, and $MTTD$.

Figure 3.4a shows that while MAD has the least variation of $DR$ with increasing values of $c$, $DR$ decreases sharply for SND for higher values of $c$. On the other hand, $FAR$ remains fairly high for all values of $c$ for MAD while the least $FAR$ is achieved for SND. This behavior is atypical in the machine learning community where better $DR$ comes at the cost of higher $FAR$. IQD tries to achieve "best of the two worlds" where $DR$ remains significantly high without producing too many false calls. $MTTD$ remains fairly constant for all 3 methods for higher values of $c$ except for SND where $MTTD$ increases for higher $c$ values. The combined performance with all 3 measures is depicted by $PI$ in Figure 3.4d. It shows that SND being sensitive to outliers or anomalies is very sensitive to the threshold constant $c$. The minimum $PI$ is obtained for IQD with $c_{IQD} = 2.2$. In the rest of this study, the optimum $c_{IQD} = 2.2$ is used for threshold determination. Speed thresholds for a sample segment with optimum $c$ value for each of the 3 methods are shown next to discuss the advantages and disadvantages of each method.

Figure 3.6 shows the speed threshold values for a particular segment of I-235 E for a given weekday. The recurring morning peak congestion resulted in low threshold values between 7 AM to 9 AM as shown in Callout (ii). However, MAD has higher speed thresholds ($\geq$ 30 mph) compared to both IQD and SND (20 mph to 30 mph threshold values). Evening peak, shown in Callout (iii), has a lower effect on traffic compared to morning peak primarily because evening peak affects mostly opposite direction of traffic (I-235 W). Nonetheless, speed thresholds are still highest for MAD and lowest for SND in evening peak time too. IQD follows closely to the SND thresholds during peak times. However, night-time construction during 1 out of 8 weeks of historical data

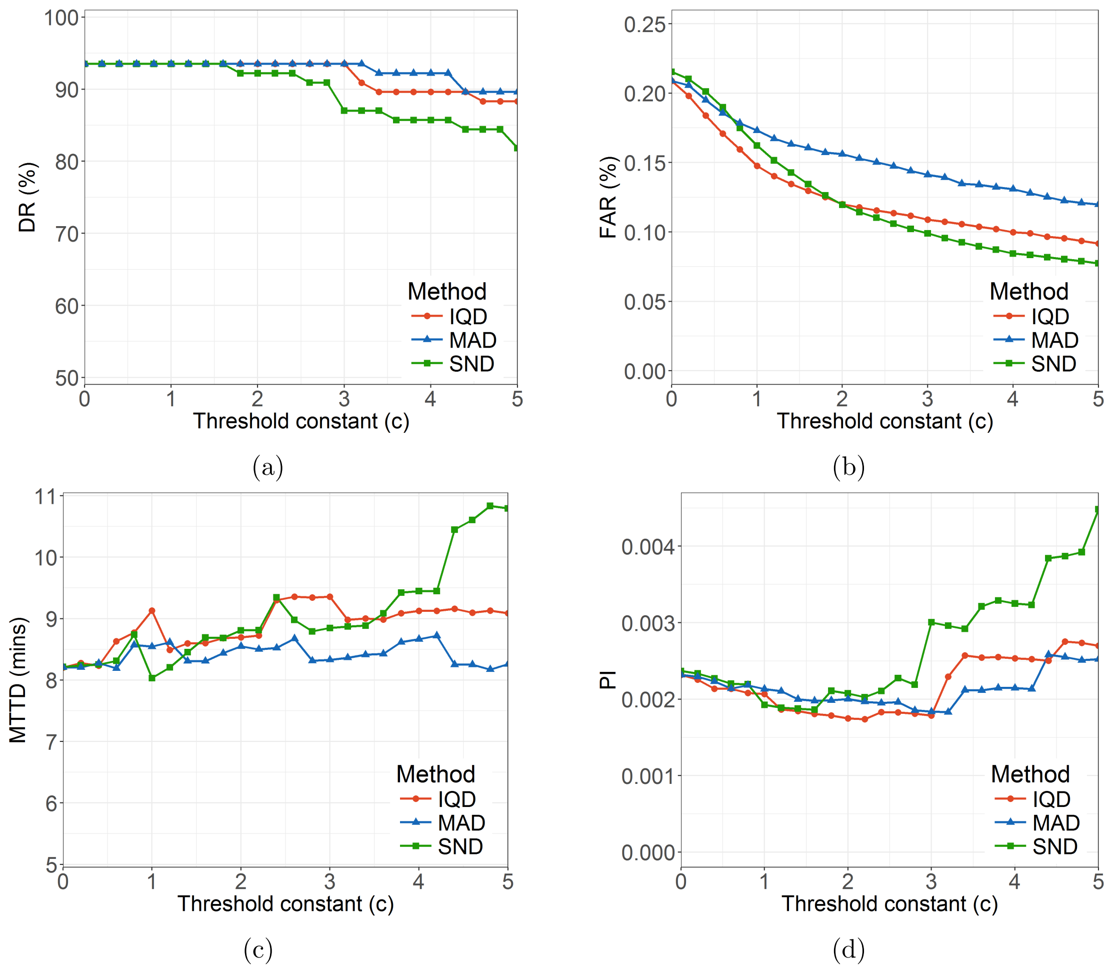


Figure 3.5: Variation of (a) DR, (b) FAR, (c) MTTD, and (d) PI for different threshold constants $c$ for SND, MAD, and IQD methods

resulted in low threshold values during night-time for SND, as shown in Callouts (i) and (iv). IQD and MAD, being robust to outliers, are not affected by such sporadic night-time congestion due to construction purposes and produces higher speed thresholds. This means that incident alarm will be triggered during night-time with IQD and MAD methods if congestion occurs due to an incident (which requires the attention of traffic incident managers) or due to some night-time construction (which traffic incident managers are already aware of from construction schedule).

SND, on the other hand, might miss such an incident due to lower threshold values. Similarly, low thresholds by SND during peak times will result in lower incident detection rates along with low false alarm rates. On the other hand, MAD having high threshold values even during peak times will lead to higher detection rates at the expense of high false alarm rates too. IQD attempts to achieve the "best of the two worlds" with low threshold values during peak times while maintaining high thresholds all other times. This is further demonstrated in Figure 3.5 where SND has lowest $DR$ along with the lowest $FAR$ while MAD has the highest $DR$ along with highest $FAR$. IQD, on the other hand, has significantly high $DR$ but with not too high $FAR$. Thus, IQD can be used a significantly better alternative than the popular SND algorithm for obtaining high incident detection rates without compromising too many false alarms.

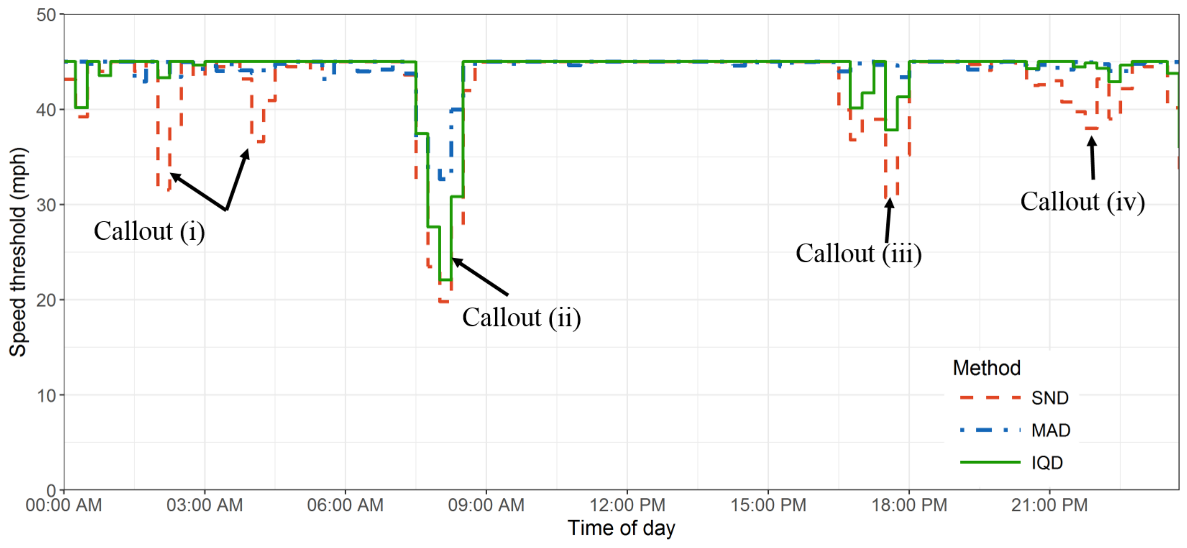
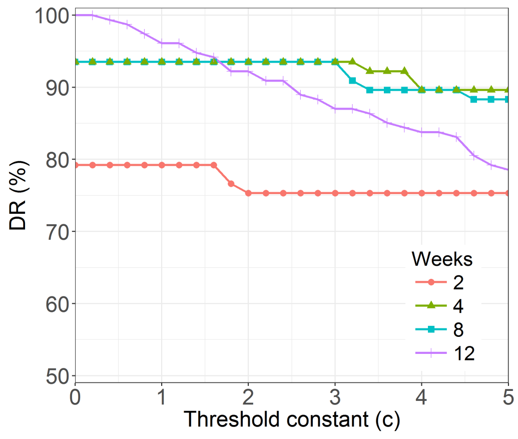


Figure 3.6: Sample speed thresholds of SND, MAD, and IQD for a particular segment in a given weekday

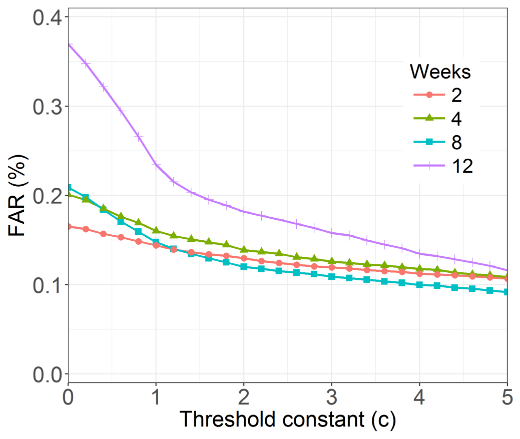
We also experimented with 2 weeks, 4 weeks, and 12 weeks of historical data to find out the optimal number of weeks to be considered for threshold computation from historical data. Figure 3.8 shows the variation of different performance measures ($DR$, $FAR$, Daily False Alarms count, $MTTD$, and $PI$) for different values of the threshold constant ($c$) for 2 weeks, 4 weeks, 8 weeks, and 12 weeks of historical data. Since IQD is found to be the best of the three methods (SND, MAD, and IQD) from the previous discussion, we only considered the IQD method for threshold computation here. From Figure 3.8, it can be seen that while 4 weeks and 8 weeks of historical data provide almost similar performance with 8 weeks being slightly better in terms of $PI$ and $FAR$, 2 weeks and 12 weeks of historical data usage result in two extreme performances. It can be recalled from Section 3.2.5 that lower $PI$ means better performance of AID algorithm. 2 weeks of historical data results in significantly low $DR$ (less than 80%) and hence higher $PI$. On the other hand, using 12 weeks of historical data produces almost 100% $DR$ and very low $MTTD$ values, but the $FAR$ increases significantly. This is evident in Figure 3.7c which clearly shows that 12 weeks of historical data with $DR \geq 90\%$ ($c \leq 3$) also produces average daily false alarm counts significantly higher than the acceptable limit of 10 false alarms per day, as discussed in Section 3.2.5. Hence, we do not consider 12 weeks data in our further analysis even though it achieves the minimum $PI$. This shows that, even though $PI$ is believed to one of the best measures to portray AID performance during model selection, it is clear that other performance measures ($DR$, $FAR$, and $MTTD$) should also be simultaneously monitored particularly when $DR$ is close to 100% or $FAR$ is close to 0%. Future studies can also look into developing alternate better performance measures which can handle these issues. Nonetheless, this discussion makes it clear that either 4 weeks or 8 weeks of historical data usage can be used for the optimal performance of our AID algorithm. On the other hand, 2 weeks and 12 weeks of data usage lead to either very low $DR$ values or very high $FAR$, which are unacceptable for real-world implementation. Since 8 weeks of historical data produced slightly better $PI$ in our analysis, our remaining analysis is performed with 8 weeks historical data.

### 3.4.2 Multivariate Spatio-Temporal Threshold Denoising

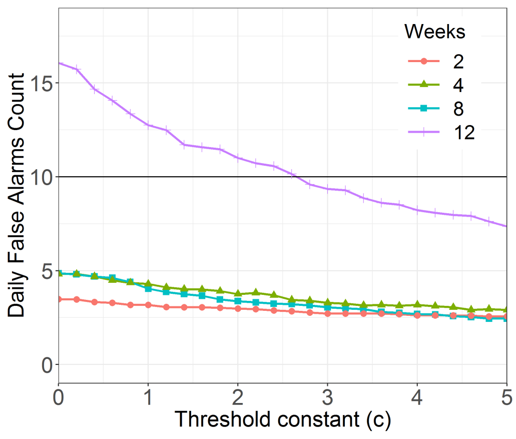
IQD and other univariate threshold computation methods (e.g., SND and MAD) do not take into consideration the spatio-temporal correlations of the 15-minute threshold windows. Thus, thresholds computed can be noisy and can be improved by considering the topology of the network. In this study, we considered the speed thresholds for all segments in a particular direction for a given day and week as a "speed heatmap" and performed two image denoising techniques, bilateral filter and total variation, to obtain smoother accurate threshold information. Figure 3.9a shows a



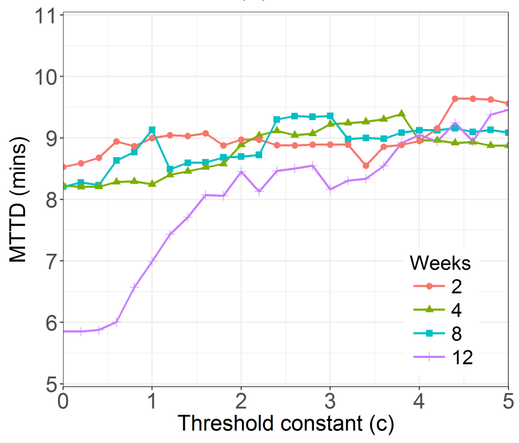
(a)



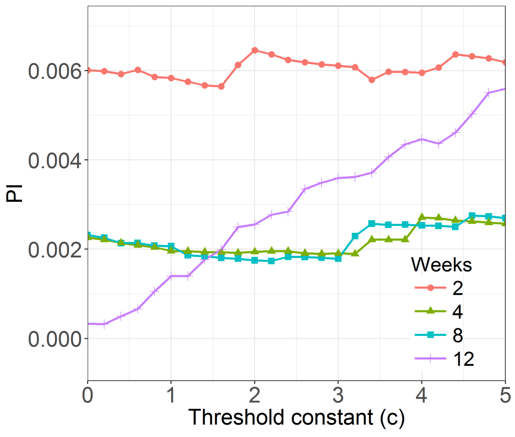
(b)



(c)



(d)



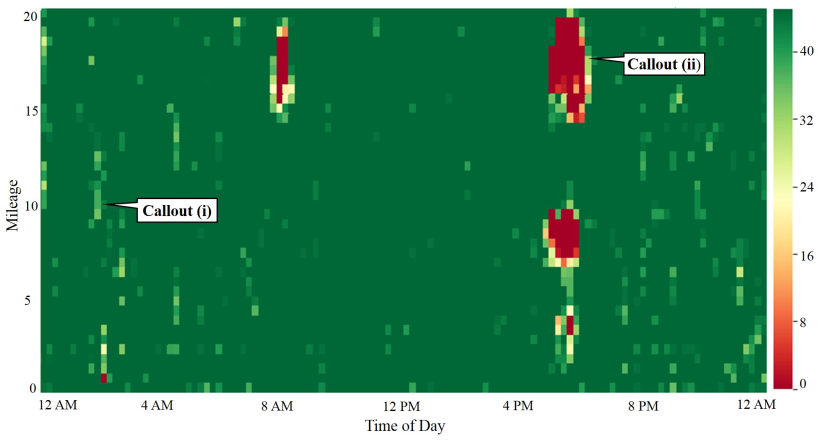
(e)

Figure 3.8: Variation of (a) DR, (b) FAR, (c) FAR count, (d) MTTD, and (e) PI for different threshold constants $c$ for SND, MAD, and IQD methods

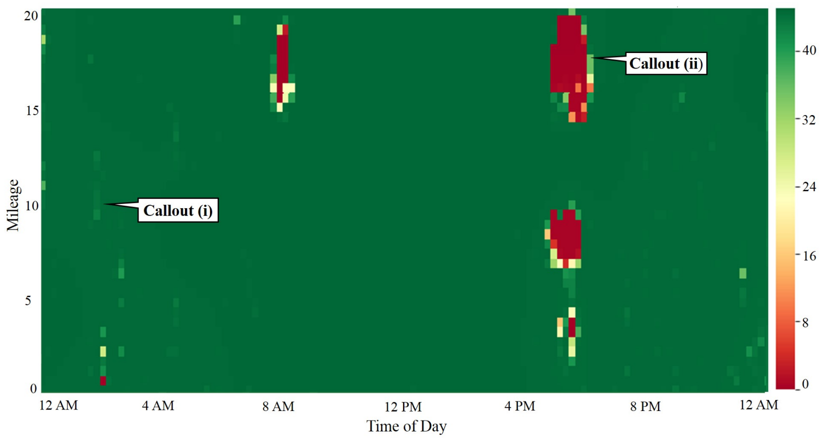
sample raw threshold speed heatmap, the denoised one with bilateral filter (3.9b), and also with total variation (3.9c).

It can be seen that the denoising techniques helps to remove the intermittent or sporadic low threshold values, shown in 'Callout (i)', generated typically during night times or non-peak hours either due to some outlying observations caused by incidents or other data issues. On the other hand, the threshold speed values at the edges of peak time congestion (as shown in 'Callout (ii)') gets increased slightly with total variation (3.9c) denoising compared to the raw threshold heatmap (3.9a). However, bilateral filter maintains the edges intact since it considers both the spatial and color range kernels for smoothing purposes. This attribute is important for incident detection because an increase in threshold values at edges of peak hours might result in missing incidents occurring during those times, which in turn will result in lowered incident detection rates. On the other hand, bilateral filter may sometimes fail to remove some spurious noise around the edges, which can be further looked upon in future to handle such edge cases using different denoising filters.

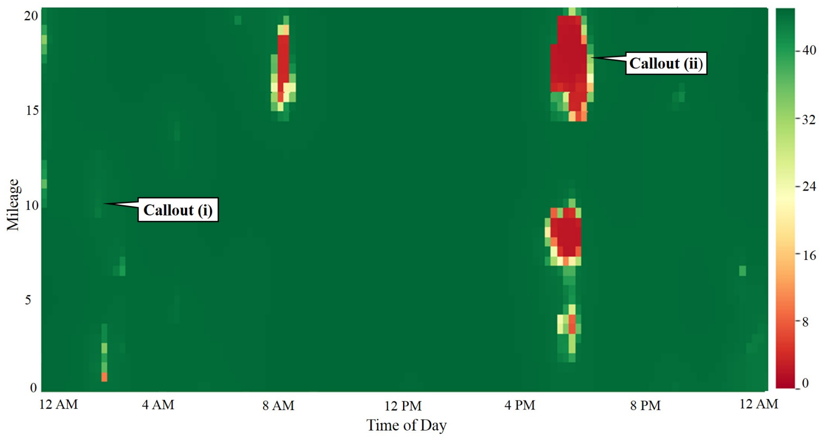
Incident detection performances will vary depending on the amount of denoising. Hence, our next objective is to find out the optimal hyperparameters for denoising. The two hyperparameters in bilateral filter denoising are $\sigma_s$ and $\sigma_r$, given in Equation 3.12. Optimal $\sigma_r$ usually depends on the amount of noise in the image and estimated in terms of $\sigma_r/\sigma_n$(Zhang and Gunturk, 2008), where $\sigma_n$ denotes the noise standard deviation. Since $\sigma_n$ is unknown for the speed threshold heatmaps, we used the ratio $\sigma_r/\sigma_i$ for hyper-parameter selection where $\sigma_i$ denotes the standard deviation of image. We assume here that the noise in the heatmap is proportional to color range in the heatmap, which can be considered as a standard normalization procedure (Zhang and Gunturk, 2008). Other than this heuristic-driven hyper-parameter selection, hyper-parameter selection can also be performed using advanced optimization techniques such as particle swarm optimization (Wang et al., 2017) or machine learning based methods (Dong et al., 2018). Such techniques can be applied in future studies to find out the improvements in AID performance.



(a)



(b)



(c)

Figure 3.10: Speed threshold heatmap in mph (a) without denoising, (b) bilateral filter denoising, and (c) total variation denoising

Figure 3.12 shows the contour plot of the variation of $DR$, $FAR$, $MTTD$, and $PI$ with $\sigma_s$ and $\sigma_r/\sigma_i$. Figure 3.11a shows that $DR$ is improved from 91.5% to 96% with increase in $\sigma_s$ and $\sigma_r/\sigma_i > 1$. Although $FAR$ also increases with denoising, however the increment is only from 0.12% to 0.136%. $MTTD$ is found to increase from 8.4 minutes to 9.1 minutes from higher $\sigma_s$ and $\sigma_r/\sigma_i$, except for very high $\sigma_s$ and $\sigma_r/\sigma_i$ ($\sigma_s > 6$ and $\sigma_r/\sigma_i > 2$). Although $FAR$ and $MTTD$ are found to increase marginally with increase in denoising, however the overall performance index is found to improve (lower $PI$) with denoising the speed threshold heatmaps, as shown in Figure 3.11d.

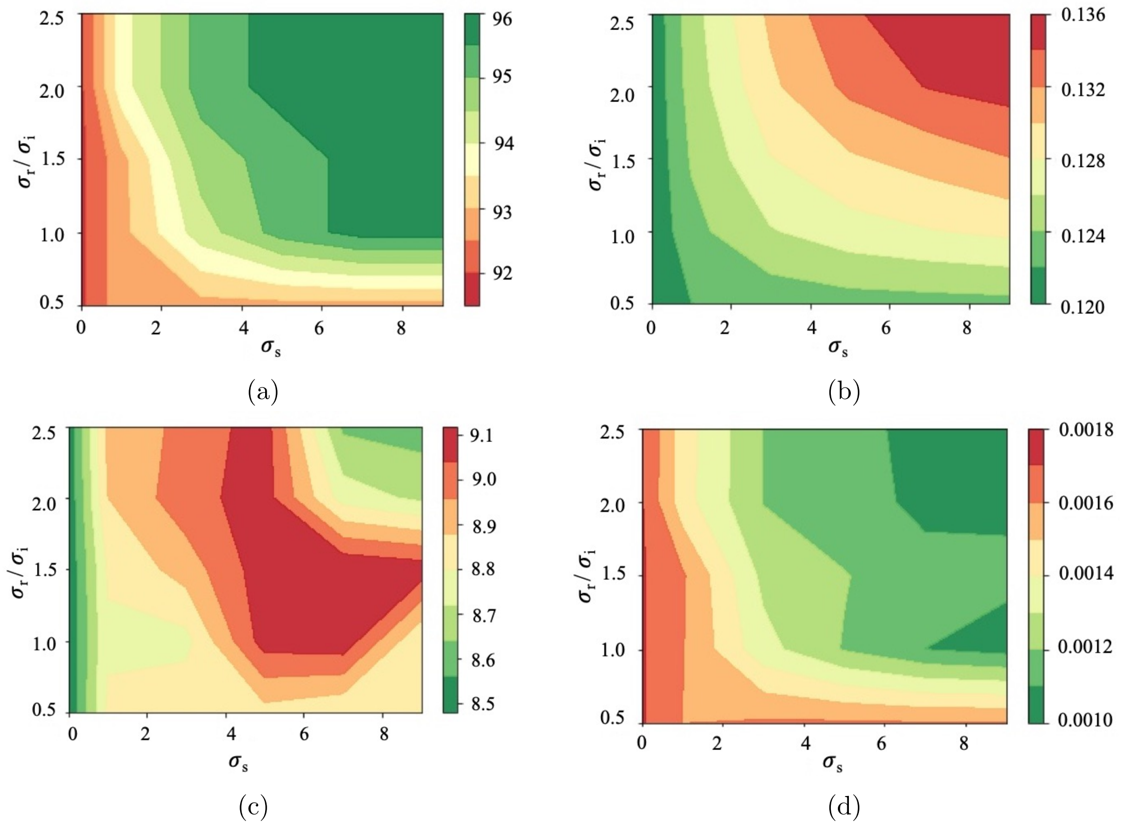


Figure 3.12: Contour plots showing variation of (a) DR (%), (b) FAR(%), (c) MTTD (mins), and (d) PI with $\sigma_s$ and $\sigma_r/\sigma_i$

Total variation denoising depends on the optimal hyper-parameter $\lambda$ (see Equation 3.22). To compare the performance of total variation (TV) with bilateral filter (BL) denoising simultaneously, Figure 3.14 shows the variation of $DR$, $FAR$, $MTTD$, and $PI$ w.r.t $\lambda$ for TV and $\sigma_s$ for BL. Since

BL has two hyper-parameters ($\sigma_s$ and $\sigma_r$), for each $\sigma_s$, the $\sigma_r/\sigma_i$ producing optimal $PI$ is used. Figure 3.13a shows that although $DR$ improves with denoising for BL, however it is found to decrease for TV denoising. On the other hand, while $FAR$ and $MTTD$ remains fairly constant with TV denoising, they are found to increase slightly with BL denoising (Figures 3.13b and 3.13b). Overall, $PI$ improvement (i.e., decrease in $PI$) is only obtained for BL denosing while for TV, $PI$ is found to increase primarily due to lowering of $DR$ with increased $TV$ denoising. Thus, bilateral filter (BL) denoising for speed threshold heatmaps helps to achieve improved incident detection performances while total variation (TV) denoising fails to achieve such improvement. To further investigate the reason for such behavior, we plotted the change in $DR$ and $FAR$ over the time of the day with increase in denoising by BL and TV.

To find out the effect of denoising of threshold heatmaps on $DR$ and $FAR$, the hourly improvement in $DR$ and $FAR$ with respect to raw heatmaps are shown in Figures 3.16 and 3.18 respectively. The $DR$ and $FAR$ change is obtained by subtracting hourly $DR$ and $FAR$ obtained using denoised heatmaps to those obtained using raw heatmaps. Figure 3.15a shows that although total variation denoising with $\lambda > 5$ helps in improvement of $DR$ during morning peak hours (7 AM), however $DR$ decreases during evening peak hours (4 PM and 5 PM) for all values of $\lambda$. This can be attributed to the increase in threshold speed values due to denoising by total variation in evening peak hours, shown in Callout (ii) of Figure 3.10. On the other hand, bilateral filter denoising helps in improving $DR$ in both morning and evening peak hours since it doesn't affect the low threshold values due to the color range parameter ($\sigma_r$). As expected, an increase in $DR$ also comes at the cost of an increase in $FAR$ and vice versa as shown in Figure 3.18. For total variation denoising, $FAR$ increases during morning peak hours and decreases during the transition from morning peak to normal conditions. During evening peak hours, although $FAR$ decreases in the beginning (4 PM), however, it increases again during the later phase of evening peak hours. As an end result, $FAR$ remains almost constant with total variation denoising as seen in Figure 3.13b. On the other hand, $FAR$ increase due to bilateral filter denoising, which is expected given the fact that $DR$ increased with denoising. Thus, it can be stated that the primary reason for

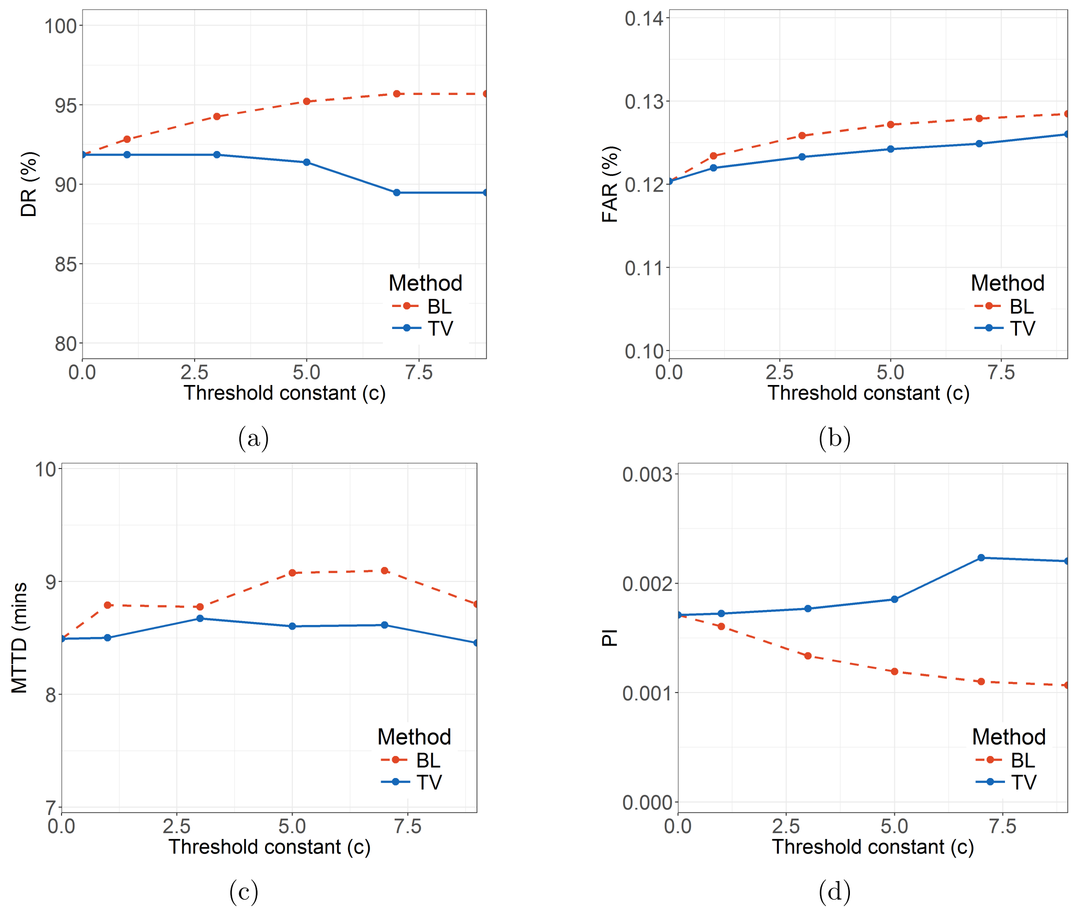


Figure 3.14: Variation of (a) DR (%), (b) FAR(%), (c) MTTD (mins), and (d) PI with denoising parameter, $\sigma_s$ for bilateral filter and $\lambda$ for total variation

the reduction of $DR$ and in turn, reduction of $PI$ for total variation denoising is primarily due to missing of detected incidents during peak hours congestion where speed threshold increases due to total variation denoising. Bilateral filter, on the other hand, maintains low threshold values intact even during peak hours with removal of spurious noisy thresholds and achieves better detection performance (higher $DR$ and $PI$) compared to raw threshold values generated by univariate threshold computation ($IQD$).

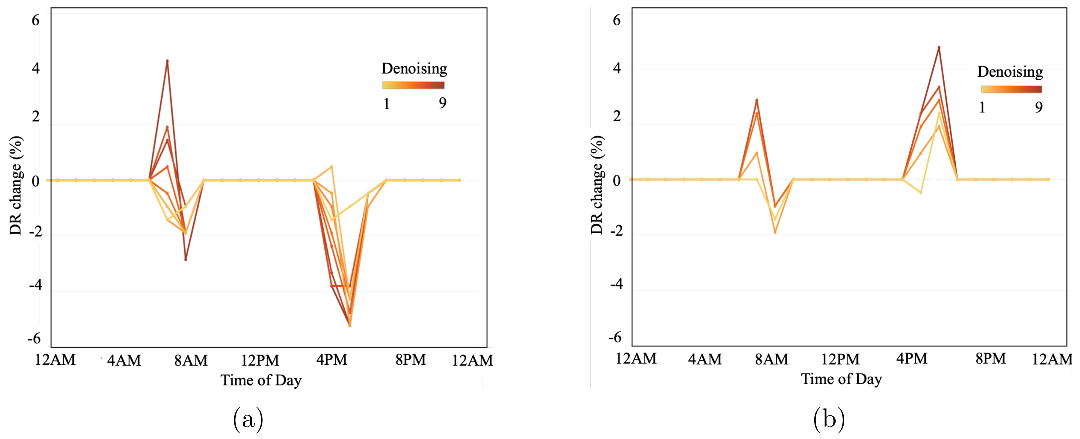
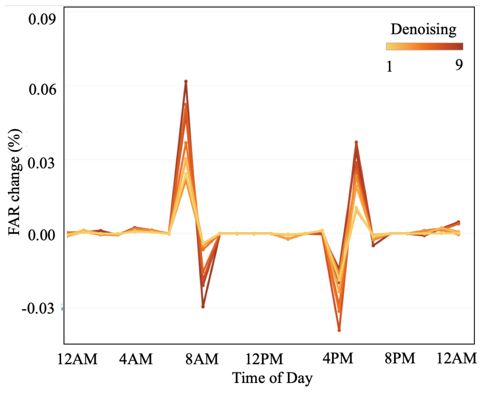


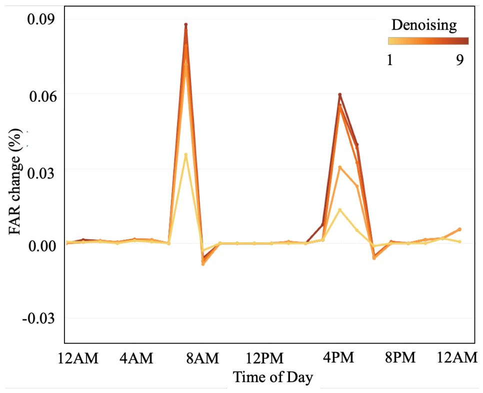
Figure 3.16: Hourly change in $DR$ due to speed threshold denoising by (a) total variation and (b) bilateral filter

## 3.5 Conclusions

This study involves development of an AID framework can be implemented over large traffic networks due to the inherent parallel computation involved in the framework. The proposed methodology consists of two major improvements to the AID algorithms developed in past literature. The first step of our AID framework consists of estimation of robust summary statistics (speed thresholds) across non-overlapping windows of time series data produced from each road segment. In our next step, we construct heatmaps with the IQD thresholds and perform image denoising of the heatmaps. We used two edge-preserving image denoising techniques, bilateral filter and total variation. Our research results show that the threshold denoising helps in achieving higher detection rates without increasing false alarms significantly thereby improving overall performance of the AID algorithm. The entire framework is data-driven in nature and can be easily extended in other data-set or traffic networks.



(a)



(b)

Figure 3.18: Hourly change in $FAR$ due to speed threshold denoising by (a) total variation and (b) bilateral filter

# CHAPTER 4. TRAFFIC CONGESTION DETECTION FROM CAMERA IMAGES

## 4.1 Introduction

Traditionally, traffic-state estimation is conducted using point-based sensors, including inductive loops, piezoelectric sensors, and magnetic loops (Kotzenmacher et al., 2004). Recent advances in active infra-red/laser radar sensors have led to these devices gradually replacing the traditional point-based sensors (Zhong and Liu, 2007). Also, with the increasing usage of navigation-based GPS devices, probe-based data are emerging as a cost-effective way to collect network-wide traffic data (Feng et al., 2014). Video monitoring and surveillance systems also are used for calculating real-time traffic data (Ozkurt and Camci, 2009). Recent advances in image processing techniques have improved vision-based detection accuracy. Deep learning methods, such as convolution neural networks (CNNs), have been able to achieve human-level accuracy in image classification tasks (He et al., 2015). The basic advantage of these methods is that they don't require picking up hand-crafted features and hence can do away with the painstaking calibration tasks needed when using camera images for traffic-state estimation (Bauza et al., 2010).

In this study, we used camera images from different locations, orientations, and weather conditions to successfully detect traffic congestion. Three different models are used for congestion detection tasks. Two of these are deep neural networks: deep convolution neural networks (DCNNs) and you only look once (YOLO). Because these models require time-consuming and costly Graphical Processing Unit (GPU) training, the support vector machine (SVM), a shallow learning model, is used as a comparison to determine the advantages of using deep models.

The outline of this chapter is as follows: The present section provides a brief introduction and the importance of traffic congestion detection, the next section gives a review of previous work done on using cameras for traffic-state estimation, the third section gives an overview of the proposed models used for traffic congestion determination, the fourth section provides description of the data used in this study and the data preprocessing steps adopted for further analyses, the fifth section includes a discussion of the results obtained from the analyses, and the final section provides the conclusion and recommendations for future work.

This work is published in Chakraborty et al. (2018a).

## 4.2 Methodology

Traffic congestion detection from camera images can be conducted in two broad ways. With the first approach, the input image can be fed into an object recognition model to determine the number of vehicles in the image and, when the number of vehicles exceeds a threshold, the image can be labeled as congested. With the second approach, the entire image can be classified as either congested or non-congested. In our study, we used the second approach, as it is much simpler and also doesn't require time-consuming manual annotation of individual vehicles.

We used three different algorithms for the traffic congestion detection task: two based on deep neural networks, which require time-consuming GPU training, and one from the shallow learning algorithm class, which doesn't require GPU training. The shallow algorithm was adopted primarily to determine the advantages, if any, for using GPU for this classification task. The three algorithms used in this study were:

1. Traditional Deep Convolution Neural Network (DCNN)
2. You Look Only Once (YOLO)
3. Support Vector Machine (SVM)

A detailed description of each of these algorithms is provided next.

### 4.2.1 Deep convolutional neural networks (DCCNs)

Collectively, DCNNs are a state-of-art technique for object detection and image classification. We used a traditional ConvNet architecture consisting of convolution and pooling layers. The convolution architecture used in this study is shown in Table 4.1. Because images from different cameras were used in this study, the input images were of different sizes, the majority being 800×450 pixels. The images were then resized to 400×225 pixels to prevent memory allocation issues during the training of the model. Next, these images were fed into the model as two consecutive convolution layers 32×3×3 in size (i.e., kernel size 3×3) followed by a max pooling layer 2×2 in size. This was followed by two additional convolution layers 64×3×3 in size and then again max pooling with a 2×2 filter. Each max pooling layer was followed by dropout with a probability of 0.25 to prevent overfitting. Finally, two fully connected layers (dense) were used, the first one with 512 neurons and the final one with two neurons corresponding to the binary classes (congested and non-congested). A batch size of 32 was used throughout the model and Leaky-ReLU was used as an activation function.

Table 4.1: DCNN model architecture used for congestion detection from images

| **Layer** | **Kernel** | **Stride** | **Output Shape** |
|---|---|---|---|
| **Input** | | | [400, 225, 3] |
| **Convolution** | 3×3 | 1 | [400, 225, 32] |
| **Convolution** | 3×3 | 1 | [398, 223, 32] |
| **Max Pooling** | 2×2 | 2 | [199, 111, 32] |
| **Dropout** | | | [199, 111, 32] |
| **Convolution** | 3×3 | 1 | [199, 111, 64] |
| **Convolution** | 3×3 | 1 | [197, 109, 64] |
| **Max Pooling** | 2×2 | 2 | [98, 54, 64] |
| **Dropout** | | | [98, 54, 64] |
| **Dense** | | | 512 |
| **Dropout** | | | 512 |
| **Dense** | | | 2 |

DCNN models are computationally expensive and usually require millions of images to train the model to prevent overfitting. However, in our study, we had only 2400 images, of which 1400 were used for training. So, to prevent overfitting, along with dropout, we also used data augmentation, similar to Ahmed et al. (Ahmed et al., 2015). Here, we randomly flipped images horizontally with a probability 0.5 and also performed horizontal and vertical shifts in the range of 10% of the total height and width. It took 25 minutes to train the model on a NVIDIA Tesla K20m GPU with 4 GB RAM memory. Keras (Chollet et al., 2015), a deep learning library, was used to run the script in GPU.

### 4.2.2 YOLO (You Only Look Once)

We adopted the YOLO model (Redmon et al., 2016) for general purpose congestion detection and localization from CCTV video feeds. Current object detection systems repurpose powerful CNN classifiers to perform detection. For example, to detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in the test image. YOLO reframes object detection; instead of looking at a single image 1000 times to accomplish detection, it looks at an image only once (but in a clever way) to perform the full detection pipeline. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. This makes YOLO extremely fast and easy to generalize to difference scenes. YOLO is also a DCNN classifier, however in the rest of the analyses, we will denote it as YOLO and the traditional DCNN explained before as DCNN.

YOLO uses a simple CNN architecture shown in Table 4.2. This neural network uses only standard layer types: convolution with a $3\times3$ kernel and max pooling with a $2\times2$ kernel. The very last convolutional layer has a $1\times1$ kernel, which serves to reduce the data to the shape $13\times13\times125$. This $13\times13$ shape is the size of the grid into which the image gets divided. There are 35 channels for every grid cell. These 35 numbers represent the data for the bounding boxes and the class predictions, as each grid cell predicts five bounding boxes and a bounding box is described by seven data elements:

- x, y, width, and height for the bounding box's rectangle;
- the confidence score; and
- the probability distribution over the two classes (congested and non-congested)

The key implementation steps for YOLO are as follows:

Table 4.2: YOLO model architecture used for congestion detection from images

| **Layer** | **Kernel** | **Stride** | **Output Shape** |
|---|---|---|---|
| **Input** | | | [416, 416, 3] |
| **Convolution** | 3×3 | 1 | [416, 416, 16] |
| **Max Pooling** | 2×2 | 2 | [208, 208, 16] |
| **Convolution** | 3×3 | 1 | [208, 208, 32] |
| **Max Pooling** | 2×2 | 2 | [104, 104, 32] |
| **Convolution** | 3×3 | 1 | [104, 104, 64] |
| **Max Pooling** | 2×2 | 2 | [52, 52, 64] |
| **Convolution** | 3×3 | 1 | [52, 52, 128] |
| **Max Pooling** | 2×2 | 2 | [26, 26, 128] |
| **Convolution** | 3×3 | 1 | [26, 26, 256] |
| **Max Pooling** | 2×2 | 2 | [13, 13, 256] |
| **Convolution** | 3×3 | 1 | [13, 13, 512] |
| **Max Pooling** | 2×2 | 1 | [13, 13, 512] |
| **Convolution** | 3×3 | 1 | [13, 13, 1024] |
| **Convolution** | 3×3 | 1 | [13, 13, 1024] |
| **Convolution** | 1×1 | 1 | [13, 13, 35] |

- Resize the input image to 416×416 pixels.
- Pass the image through a CNN in a single pass. The architecture of the CNN is described in the following section.
- The CNN outputs a 13×13×k tensor describing the bounding boxes for the grid cells. The value of k is related to the number of classes as follows: k = (number of classes + 5)*5.
- Compute the confidence scores for all bounding boxes and reject all boxes that fall below a predefined threshold.

Because there are $13 \times 13 = 169$ grid cells and each cell predicts five bounding boxes, there are 845 bounding boxes in total. Ideally, the majority of these boxes would have very low confidence scores. In this study, a confidence threshold of 45% was used for congestion detection.

### 4.2.3 Support Vector Machine (SVM)

A SVM is one of the most widely used shallow algorithms for image classification task. It solves a constrained quadratic optimization problem to classify data into different categories. The resulting optimal hyperplane is determined by maximizing the largest minimum distance to training examples to make it least sensitive to noise. We utilized the Oriented FAST and Rotated BRIEF (ORB) (24) feature detector to detect the key points in each image, whereby the FAST (features from accelerated segment test) algorithm was used to extract the key points and the Harris corner distance was used to determine the top N points. The algorithm was run on the training data set with 10-fold cross-validation to determine the optimal penalty parameter and kernel. This algorithm was run on Windows 7 with Intel Core i7-4790 CPU with 8GB of RAM.

## 4.3 Data Description

Two different data sources were used in this study: camera images and radar-based Wavetronix sensors. Camera images were obtained from 121 cameras from the Iowa DOT CCTV camera database spread across the interstates and highways of Iowa. The database covered the major cities of Iowa: e.g., Des Moines, Sioux City, Cedar Rapids, Council Bluffs, Davenport, and Iowa City. Images were extracted from the cameras at 5-minute intervals from October 2016 to March 2017, resulting in a total of 3.5 million images during the study period. The task of assigning a label to an image (congested or non-congested) consisted of four sub-tasks:

1. Associating each camera with a nearby wavetronix sensor pair,
2. Smoothening the wavetronix data,
3. Extracting the details (camera name, timestamp) of each image, and
4. Assigning the label of the image based on sensor data.

Details of each of these tasks are discussed next.

Each camera was first associated with the two nearest Wavetronix sensor pairs covering both directions of the freeway on which camera was placed. If the sensor pair was located more than 0.5 miles away from the camera, then the particular camera was removed from analysis.

The next step was to assign the traffic data from the sensor to each image. However, sensor data obtained from Wavetronix in 20-second intervals included too much noise; so, we used Wavelet smoothing to remove the noise. In this study, among the several families of wavelets that could be used, such as Haar, Daubechies, Biorthogonal, Symlets, Coiflets, Morlet, Mexican Hat, Meyer, etc., we used Daubechies extremal phase wavelets. Daubechies family wavelets are also known "dbN," where N refers to the number of vanishing moments. The higher the value of N, the longer the wavelet filter and the smoother the wavelet. Based on our data, we used db2 with level 6 to achieve a smooth curve-like filter that followed most of the variations of the original signal. A sample of the original and smoothed data is shown in Figure 4.1.

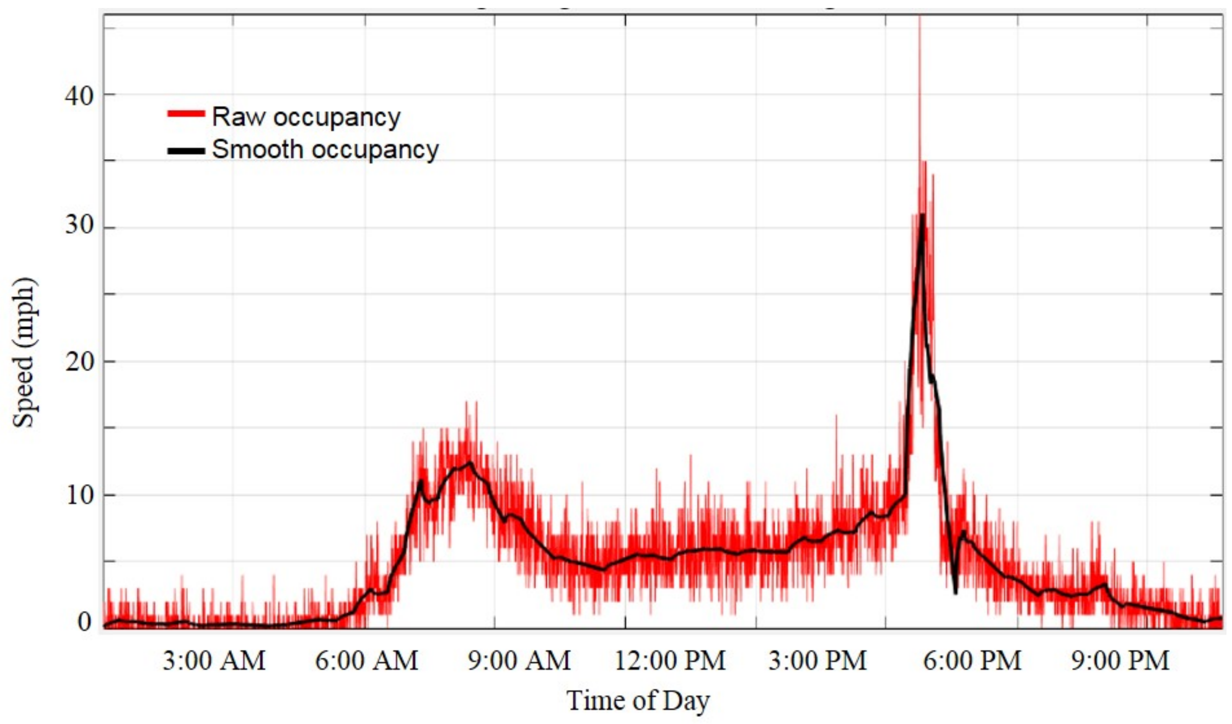


Figure 4.1: Original and smoothed occupancy of Wavetronix using Wavelet transform (db2 level 6)

The next step was to extract the details of each image. The top of each image showed the details of the image (direction, camera name, and timestamp). Optical character recognition (OCR) was used to extract the details from each image, which were then matched with the corresponding sensor data based on the camera's name and timestamp.

After obtaining the smoothed Wavetronix data, timestamp, and camera name for each image, we assigned the traffic data obtained from the sensor to the image. The traffic data comprised speed, volume, and occupancy observed at 20-second intervals. To assign the congested or non-congested label to the image, we used occupancy values, which are denoted by the percentage of the time the sensor is occupied by vehicles and have one-to-one mapping to traffic density or the number of vehicles in the unit distance. Persaud and Hall suggested that occupancy of 20% or more should be considered congested, whereas occupancy below that should be considered non-congested (Persaud and Hall, 1989). Thus, if no congestion (occupancy ¡20%) was observed in either direction of the wavetronix pair, then the image was classified as "non-congested"; if congestion was visible in any particular direction or in both directions, then it was labeled as "congested." We adopted this approach to do away with manual labeling of congested and non-congested images and to follow a uniform methodology for assigning labels to the images.

Finally, we obtained 1218 congested images and more than 3 million non-congested images. Due to class imbalance, we randomly chose 1200 non-congested images out of the 3 million images. This dataset consisting of a total 2418 images was then subdivided into a training set and a test set. The training set consisted of 1400 images with equal proportions of congested and non-congested images. However, as will be discussed later, the YOLO approach of congestion detection requires manually annotating the region of congestion. For this purpose, 100 congested images were extracted from the training set and manually annotated with the congested region. The test set consisted of 1018 images out of which 518 were congested and the rest were non-congested. Because sensor errors can occasionally cause misclassification of images, test set images were manually cross-checked and then the final labels were assigned. However, no manual cross-checking of labels was performed for the training set, as it was assumed that the algorithm itself should be able to determine the misclassification, if any, in the training set.

## 4.4 Results

The performance of each of the three algorithms were trained on 1400 images and tested on 1018 test set images (518 congested and 500 non-congested). YOLO was trained and tested on NVIDIA GTX 1080 Ti 8 GB RAM GPU while for DCNN, NVIDIA Tesla K20m GPU with 4 GB RAM was used. Intel Core i7-4790 with 8 GB RAM CPU was used for training and testing of SVM. The training times for YOLO, DCNN and SVM were 22 hours, 26 minutes, and 50.4 seconds respectively. The testing times for the 3 algorithms were 0.01, 0.01, and 0.03 seconds/frame respectively. The testing time does not include the time required for the model; rather, it includes only the time required to predict the class for each image. Because YOLO and DCNN are deep models, they had to be trained and tested using GPUs and which involved time-consuming and costly training compared to its shallow counterpart, SVM. The testing times for DCNN and YOLO were lower, but they required GPU during testing time as well.

The performance of the algorithms was evaluated using the standard performance metrics of precision, recall, and accuracy (Equations 4.1, 4.2, and 4.3 respectively). When a congested image was correctly labeled (i.e., the predicted label was also "congested"), it was classified as true positive ($TP$). Similarly, if a non-congested image was correctly labeled as "non-congested", then it was classified as true negative ($TN$). However, if the actual label was "congested" and the predicted label was "non-congested," it was classified as false negative ($FN$). And finally, if the actual label was "non-congested" and the predicted label was "congested," it was classified as false positive ($FP$).

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.3}$$

Some examples of true classifications and misclassifications obtained from each of the three algorithms (YOLO, DCNN, and SVM) are shown in Figure 4.3. Examples of true positives, for which YOLO, DCNN, and SVM correctly labeled congested images, are shown in Figures 4.2a, 4.2b, and 4.2c respectively (YOLO also gives the bounding box for the congested region). On the other hand, examples of false positives, for which the algorithms misclassified non-congested images as congested, are shown in Figures 4.2d, 4.2e, and 4.2f. It can be seen that YOLO misclassified an image as a congested region because of a group of vehicles located far away from the camera during nighttime (Figure 4.2d) and vehicles on a bridge led to misclassification by DCNN (Figure 4.2e). SVM, on the other hand, had misclassifications in adverse weather conditions (Figure 4.2f) because snow particles were detected as corners, which caused the image to be labeled as congested. Examples of false negatives, for which the algorithms failed to detect congested images correctly, are shown in Figures 4.2g, 4.2h, and 4.2i. Congestion quite distant from the camera led to misclassification by YOLO (Figure 4.2g), whereas DCNN failed to detect congestion in a single lane when the other lane was closed and hence empty (Figure 4.2h). Glare issues resulted in SVM misclassifications (Figure 4.2i). Finally, examples of true negatives, for which the algorithms correctly labeled non-congested images, are shown in Figures 4.2j, 4.2k, and 4.2l.

The precision, recall, and accuracy values obtained from each algorithm are shown in Table 4.3. YOLO achieved the highest precision, recall, and accuracy followed closely by DCNN. Because YOLO achieved better accuracy compared to DCNN, we didn't performed region-based CNN separately to determine the congested region of the results obtained from DCNN. YOLO, on the other hand, being a region-based classifier gives the congested region of the image by default (see Figures 4.2a and 4.2d). The accuracy obtained by SVM was comparatively lower (85.2%) than expected given the lower computation costs involved in such a shallow algorithm. In this context, it should be mentioned that a separate analysis, reported in a separate paper (24), using an ensemble of shallow learning algorithms (SVM with ORB, Shi-Tomasi, and Structured Edge Toolbox feature detector) gave an accuracy of 86.7% with the same dataset. Here, we used only SVM with ORB for comparison of deep learning models with a standard shallow model.

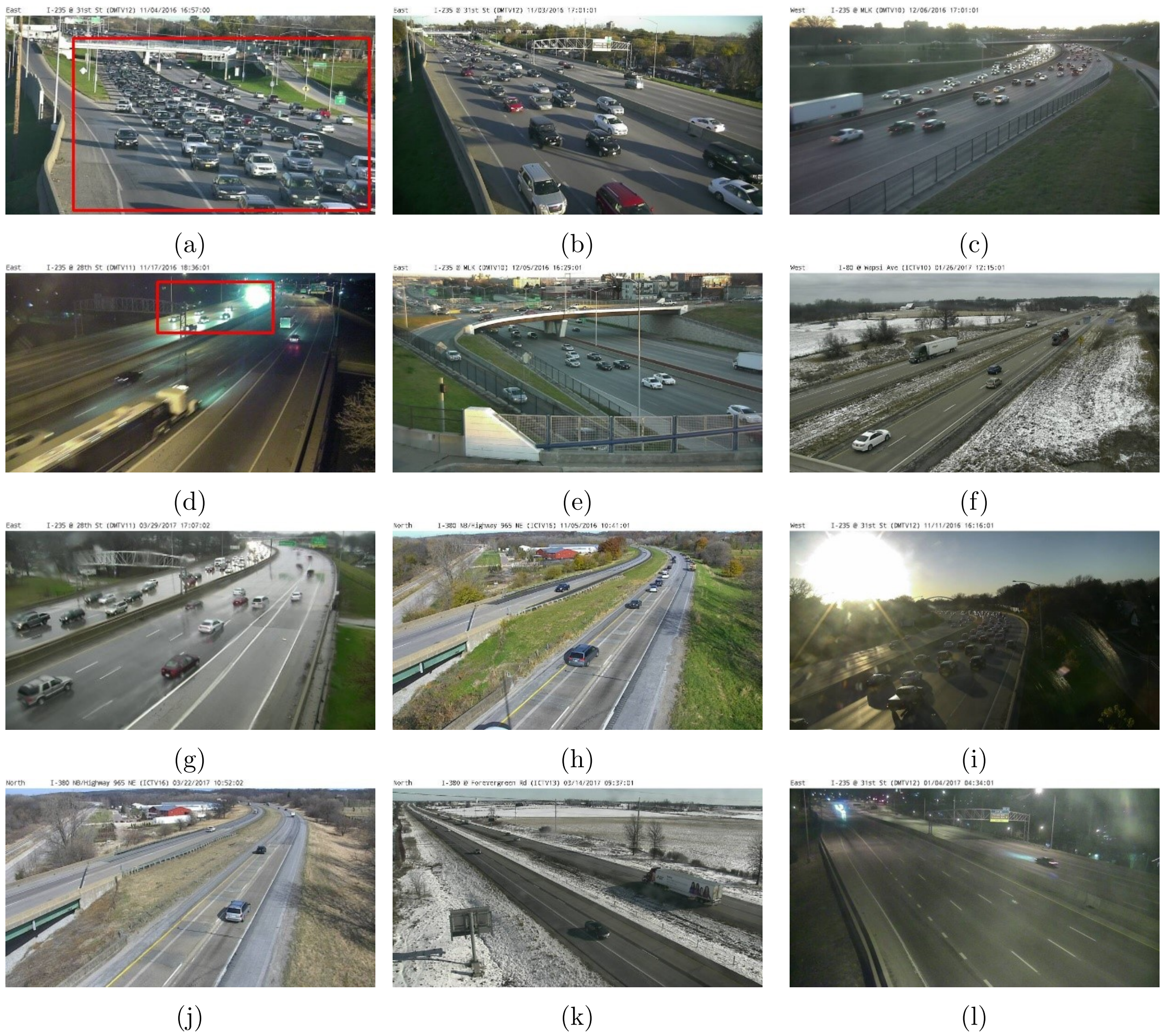


Figure 4.3: Congestion detection classification examples: (a-c) true positives, (d-f) false positives, (g-i) false negatives, (j-l): true negatives from YOLO, DCNN, and SVM respectively

Table 4.3: Precision, recall and accuracy values obtained from the three algorithms for congestion detection from images

| **Method** | **Precision (%)** | **Recall (%)** | **Accuracy (%)** |
|---|---|---|---|
| **YOLO** | 88.6 | 94.3 | 91.4 |
| **DCNN** | 86.9 | 93.9 | 90.2 |
| **SVM** | 82.8 | 88.5 | 85.7 |

### 4.4.1 Sensitivity Analysis

Sensitivity analysis was also performed to determine which factors might affect the performance of the congestion detection system developed here. We evaluated two factors that could influence the classification task: first, the time of day the image was captured (daytime versus nighttime) and, second, camera resolution (blurring, rain, snow, and glare). The test database was then divided into four subgroups according to the combination of the two factors, as follows:

1. D-G: daytime, good resolution (436 images);
2. N-G: nighttime, good resolution (147 images);
3. D-P: daytime, poor resolution (190 images); and
4. N-P: nighttime, poor resolution (245 images).

Receiver operating characteristics (ROC) curves were then used to compare the performance of each algorithm for each subgroup based on the true positive and false positive rates ($TPR$ and $FPR$, respectively), as defined in Equations (4) and (5), respectively:

$$TPR = \frac{TP}{TP + FN} \tag{4.4}$$

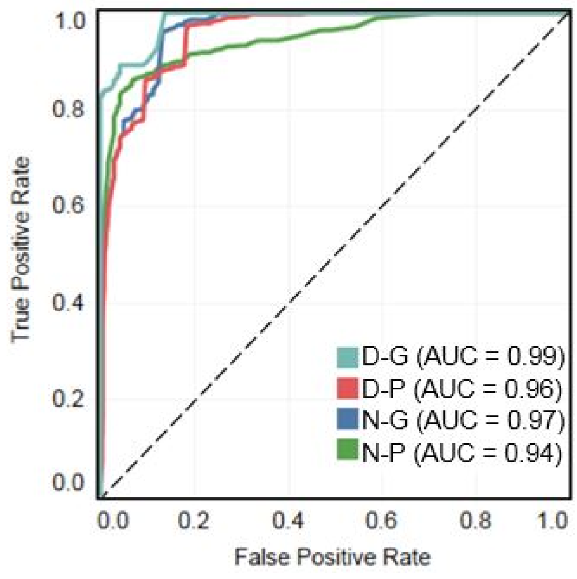
$$FPR = \frac{FP}{FP + TN} \tag{4.5}$$

For an efficient image classification model, the TPR should be higher than the corresponding $FPR$. On the other hand, for a poor-vision system model, when the sensitivity ($TPR$) increases, it loses the ability to discriminate between congested and non-congested images, which makes the TPR directly proportional to the FPR. The ROC curves for each subgroup obtained from the three models — YOLO, DCNN, and SVM are shown in Figures 4.4a, 4.4b, and 4.4c, respectively. The overall ROC curve for the three algorithms is shown in Figure 4.4d. Also, the area under each curve (AUC) is provided for each case. For all three algorithms, $TPR$s were higher than the corresponding $FPR$s irrespective of the prevailing conditions (daytime or nighttime, poor or good resolution). All of the algorithms performed well during the daytime, irrespective of the camera resolution. However, AUCs were found to be lowest for poor resolution images at night (N-P). Moreover, irrespective of the conditions, the AUCs from all algorithms for each subgroup were found to be mostly higher than 0.90, except for N-P conditions from SVM. This shows that the system works well even under challenging conditions.

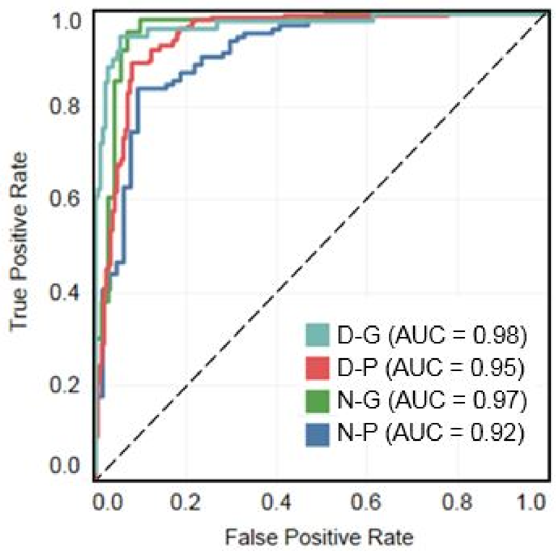
In addition, ROC curves can be used by traffic management centers (TMCs) to choose an optimal threshold between TPR and FPR. Previous studies have shown that too many false calls is a major reason for limited integration of automatic incident detection algorithms in the TMC. Hence, it is important for TMC personnel to know the accuracy that be achieved given a particular FPR. For example, if a TMC wants to restrict the FPR to lower than 0.1, then the TPR obtained by YOLO, DCNN, and SVM will be 0.92, 0.96, and 0.82, respectively, during good daytime (D-G) conditions. Obviously, the accuracy would be lower with poor camera conditions at night. Hence, TMC personnel can use the ROC curves to set the optimal threshold of TPR and FPR based on their specific needs.

### 4.4.2 Real-time Implementation

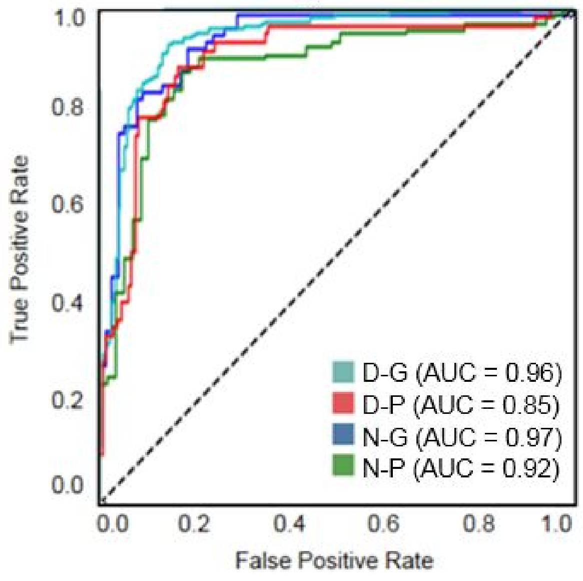
The congestion detection algorithms can also be implemented online easily. With a test time of 0.01 seconds per image, the algorithms can be adopted to detect traffic congestion of approximately 1000 cameras in every 10 seconds interval using a single GPU. Figure 4 shows an example of congestion detection by DCNN algorithm on images extracted from a camera on a single day (27th October, 2017) at every 10 seconds interval. The congestion alarm occurrences from camera is shown in the background of Figure 4 along with the occupancy data obtained from nearest radar sensors (both directions). However, due to sensor issues, sensor data were missing from 8:51 AM to 12:57 PM. So, a 2-vehicle crash reported at around 10:30 AM was missed by sensor, but was detected successfully by the camera. Thus, this example also shows that using multiple data sources (cameras, sensors, etc.) can increase the reliability in traffic state estimation. Callouts (i) and (ii) are also provided in Figure 4 to show samples of camera images when congestion alarms were triggered. To eliminate false alarms, alarms are triggered only when congestion is detected



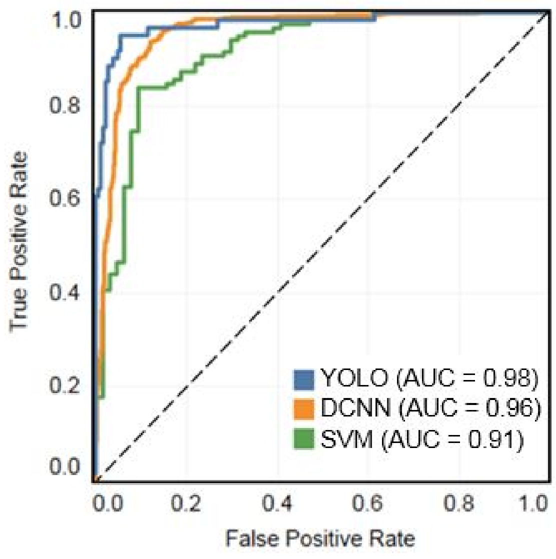
(a)



(b)



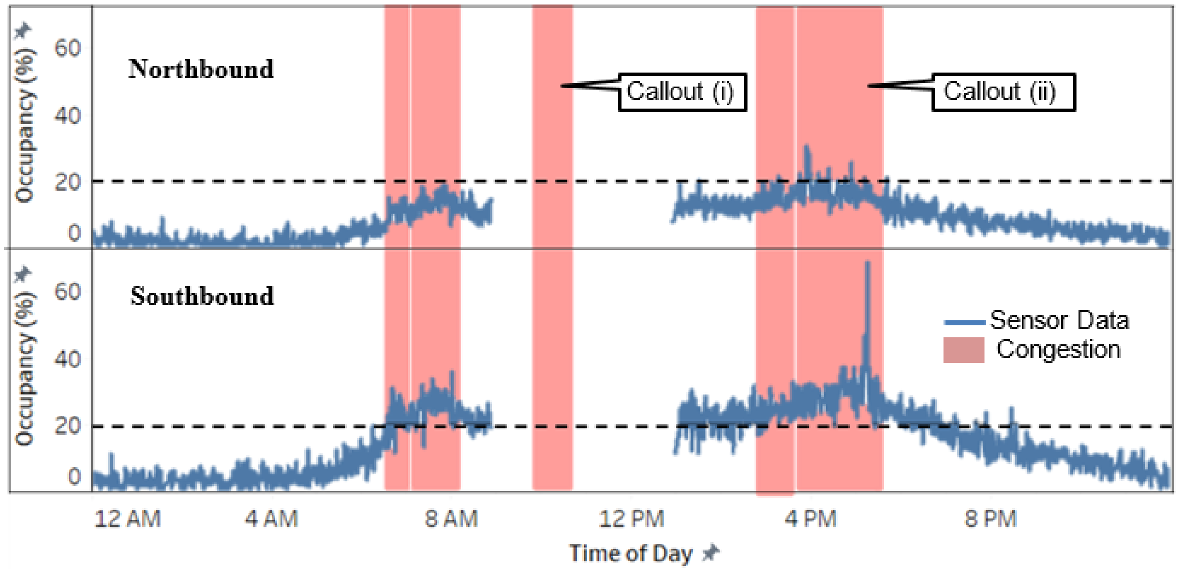
(c)



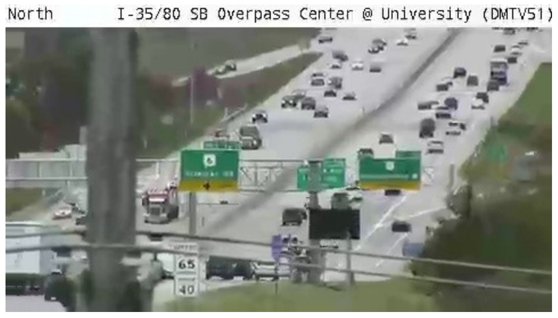
(d)

Figure 4.5: ROC curves under different prevalent conditions obtained from (a) YOLO; (b) DCNN; (c) SVM; and, (d) all conditions combined for each algorithm.

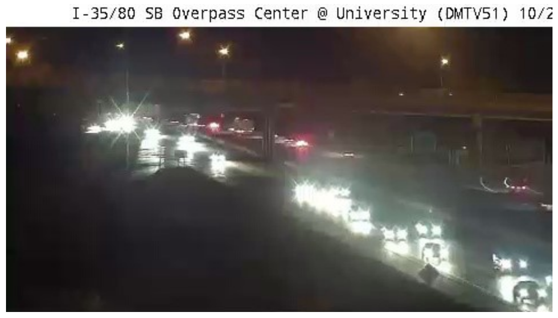
on 3 consecutive 10-seconds interval frames (persistency test). Also, alarms triggered within 5 minutes interval are combined together to form a single continuous alarm. These "signal smoothing" techniques help in decreasing false alarm rates (FAR) and increasing detection rates (DR). Future studies can be done implementing better smoothing techniques like Fourier Transforms or Wavelet smoothing and determining the DR and FAR on a network of cameras.



(a)



(b)



(c)

Figure 4.7: (a) Sensor occupancy data and congestion alarm from a camera on a particular date; (b) - (c) Camera images of callouts (i-ii) shown in a.

## 4.5 Conclusions

In this study, two modern deep learning techniques, the traditional DCNN and YOLO models, are used to detect traffic congestion from camera images. SVM is also used for comparison and to determine what improvements are obtained while using costly GPU techniques. To eliminate the time-consuming task of manual labeling and to maintain uniformity in congestion labeling, we used nearby Wavetronix sensors to correctly identify congested images. For testing purposes, we also labeled each image manually to remove misclassifications due to sensor errors. The YOLO model achieved the highest accuracy of 91.2% followed by DCNN with an accuracy of 90.2%; 85% of images were correctly classified by SVM. Congestion regions located far away from the camera, single-lane blockages, and glare issues were found to affect the accuracy of the models. To determine the sensitivity of the models to different camera configurations and light conditions, ROC curves are used. All the algorithms are found to perform well in daytime conditions, but night conditions are found to affect the accuracy of the vision system. However, for all conditions, the AUCs are found to be greater than 0.9 for the deep models. This shows that the models perform well in challenging conditions as well.

# CHAPTER 5. FREEWAY INCIDENT DETECTION FROM CAMERA VIDEOS

## 5.1 Introduction

Traffic incident detection approaches from CCTV cameras can be broadly classified into two categories: (a) trajectory-based approaches and (b) pixel-based approaches (Coşar et al., 2016). Trajectory-based approaches involve finding out individual vehicle trajectories and thereafter determining incident trajectories. Such methods are usually applicable when traffic density is low (i.e., non-congested conditions). However, with the increase in traffic density, it becomes infeasible to track individual vehicles. Then, pixel-based approaches are used to find out abnormal motions. It usually involves background subtraction or optical flow based motion estimation to detect abnormal motions. In this study, we show how optical flow based motion estimation can be used for incident detection for both congested and non-congested conditions. However, since optical flow estimation is computationally intensive (usually less than 5 fps), we propose to use optical flow based incident detection during congested conditions only. For non-congested traffic conditions, trajectory-based approach can be used to detect traffic incidents. Congested conditions are determined using the methodology stated in Chapter 4.

Trajectory-based event detection can be broadly classified into two categories: (a) explicit event recognition by supervised learning and (b) unsupervised learning based on anomaly detection. While supervised techniques can in general provide better results in detection or classification tasks, the main hindrance in its application is the scarcity of enough supervised data samples and cost of manually annotating and labeling the dataset. In particular, manually annotating vehicle tracks in a video stream is extremely labor intensive, expensive, and not scalable. In this work, we establish a new learning framework for traffic incident detection using recent advances in semi-supervised learning Loog (2016). Via this framework, we achieve the "best-of-both-worlds"; we manually annotate only a small sample of normal vehicle tracks and tracks of vehicles involved in an incident, and then we used all other (unlabeled) vehicle tracks to improve the classification performance. Our experimental results on traffic data provided to us by the Iowa DoT demonstrate that our framework achieves superior performance compared to supervised learning techniques with a comparable amount of labeled examples.

This chapter is structured as follows. Section 5.2 contains the details of the methodology adopted for trajectory-based and pixel-based incident detection. Section 5.3 gives the details of the data used in this study followed by the detailed results in Section 5.4. The final section provides a brief summary of the paper.

A part of this work is published in Chakraborty et al. (2018b).

## 5.2 Methodology

We first describe the trajectory-based incident detection which can be applied usually for non-congested traffic conditions followed by pixel-based approach for incident detection in both congested and non-congested conditions.

### 5.2.1 Trajectory-based Incident Detection

Traffic incident detection from videos using trajectory information comprises of 3 basic tasks: (a) vehicle detection (b) vehicle tracking and trajectory formation, and (c) trajectory classification. Each task is described next, with our primary focus geared towards trajectory classification using semi-supervised techniques.

#### 5.2.1.1 Vehicle Detection

In recent years, evolution of convolutional neural networks (CNN) have resulted in significant improvement in object detection and classification performance. Various state-of-art object detection algorithms over the fast years are based on CNN, which includes Region based CNN (RCNN) (Girshick et al., 2014b), Faster RCNN (Ren et al., 2015), Mask RCNN (He et al., 2017), Deformable

ConvNets (Dai et al., 2017), Single Shot MultiBox Detector (SSD) (Liu et al., 2016b), You Look Only Once (YOLO) (Redmon et al., 2016), YOLOv2 (Redmon and Farhadi, 2016), YOLOv3 (Redmon and Farhadi, 2018), etc.

In this study, we chose YOLOv3 (Redmon and Farhadi, 2018) for vehicle detection primarily because of its fast performance with reasonable accuracy which makes it suitable for real-time performance. Current object detection systems repurpose powerful CNN classifiers to perform detection. For example, to detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in the test image. In contrast, YOLO reframes object detection: instead of looking at a single image a thousand times to detect an object, YOLO only looks at an image once (but in a clever way) to perform the full detection pipeline (as shown in Figure 5.1).

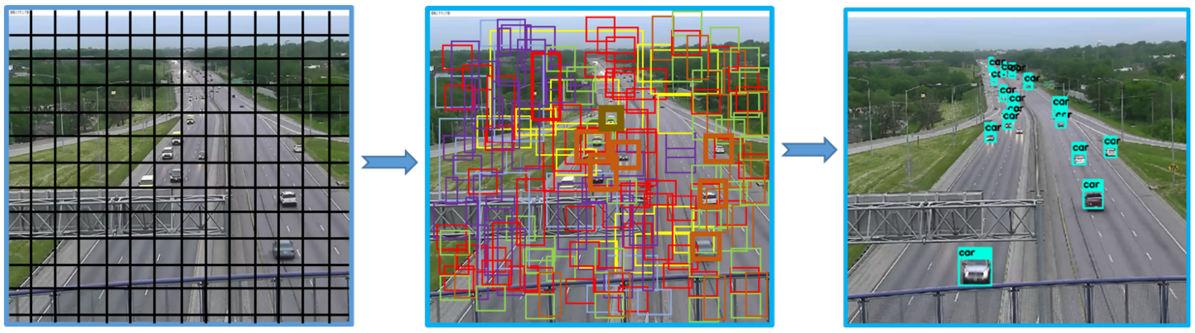


Figure 5.1: Confidence score prediction of bounding boxes by YOLO, with colors and bounding box widths indicating confidence score probabilities

A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. This makes YOLO extremely fast and easy to generalize to different scenes.

In this study, we used the YOLOv3-416 model trained on the Microsoft Common Objects in Context (COCO) dataset (Lin et al., 2014). We used the classes: ‘car’, ‘motorbike’, ‘bus’, and ‘truck’ out of the 80 classes in the COCO dataset for our vehicle detection module.

#### 5.2.1.2 Vehicle Tracking and Trajectory Formation

Recent improvements in object detection performances have led to tracking-by-detection as the leading paradigm for multi-object tracking (MOT). In MOT, multiple objects are detected in each frame and the aim is to associate the detections across frames in a video sequence. In this study we used the Simple Online and Realtime Tracking (SORT) algorithm for vehicle tracking. This is an online multi-object tracking algorithm which uses the Kalman Filter and the Hungarian algorithm for the data association problem. We chose this tracker because of its reasonable performance in online, realtime settings. SORT tracker updates at 260 Hz, making it suitable for realtime implementation.

Our object tracker module outputs a sequence of bounding box coordinates, X-center ($X^c$), Y-center ($Y^c$) for each unique vehicle id across the frames, thereby forming a trajectory. Thus, a trajectory can be defined as a sequence of 2-dimensional points, denoted by:

$$TR_i = (p_1, p_2, p_3 \ldots p_j \ldots p_{len_i}) \tag{5.1}$$

Here, each $p_j$ is a 2-dimensional point representing the bounding box coordinates. The length $len_i$ of a trajectory can be different for different trajectories. Note that, in our study, we consider only the bounding box center coordinates, but other features such as bounding box appearance descriptors can also be included.

#### 5.2.1.3 Semi Supervised Trajectory Classification

The aim of semi-supervised learning is to exploit the easily-available unlabeled data to improve the performance of supervised classifiers. However, it is not always the case that the semi-supervised classifiers achieve lower error rates compared to the supervised counterparts. On the contrary, empirical studies have observed severely deteriorated performances (Ben-David et al., 2008). Recently, Loog demonstrated how Maximum Likelihood (ML) can be used to improve classification performance in semi-supervised setting (Loog, 2016).

In this paper, we address the problem of trajectory classification in semi-supervised settings using the Contrastive Pessimistic Likelihood Estimation (CPLE) based on ML estimation (Loog, 2016). We present experimental results proving the validity of our approach and compare the results with the traditional semi-supervised classification techniques. We discuss next the details of the CPLE method for semi-supervised classification followed by a brief description of the traditional algorithms that we chose for comparison.

### Contrastive Pessimistic Likelihood Estimation (CPLE)

The two main concepts that form the core of CPLE are *contrast* and *pessimism*. The CPLE method is contrastive, meaning that the objective function explicitly controls the potential improvements of the semi-supervised classification over the supervised counterpart. CPLE is also pessimistic, which means that the unlabeled data is modeled as behave adversarially so that any semi-supervised learning mechanism least benefits from it. This makes it resilient to whatever form the true (unobserved) labels of the unlabeled data take.

For a $K$-class supervised classification, the log-likelihood objective function is given by:

$$L\left(\theta|X\right) = \sum_{i=1}^{N} \log p\left(x_i y_i|\theta\right) = \sum_{k=1}^{K}\sum_{j=1}^{N_k} \log p\left(x_{ij}, k|\theta\right) \tag{5.2}$$

where class $k$ contains $N_k$ samples, $N = \sum\limits_{k=K}^{N_k}$ is the total samples, $X = \{(x_i, y_i)\}_{i=1}^{N}$ is the set of labeled training pairs with $x_i \in \mathbb{R}^d d$-dimensional feature vectors, and $y_i \in C = \{1, ..., K\}$ are their corresponding labels.

The supervised ML estimate, $\widehat{\theta}_{\text{sup}}$, maximizes the above criterion:

$$\widehat{\theta}_{\text{sup}} = \underset{\theta}{\operatorname{argmax}}\, L\left(\theta|X\right). \tag{5.3}$$

In our study, we chose Linear Discriminant Analysis ($LDA$) as our classifier, similar to the approach of Loog (Loog, 2016). Here, the log-likelihood objective function is given by

$$\begin{aligned} L_{LDA}\left(\theta|X\right) &= \sum_{i=1}^{N} \log p\left(x_i, y_i|\pi_1, ..., \pi_k, \mu_1, ..., \mu_k, \Sigma\right) \\ &= \sum_{k=1}^{K}\sum_{j=1}^{N_k} \log p\left(x_{kj}, k|\pi_k, \mu_k, \Sigma\right), \end{aligned} \tag{5.4}$$

where $\theta = (\pi_1, ..., \pi_k, \mu_1, ..., \mu_k, \Sigma)$, $\pi_k$ are the class priors, $\mu_k$ are the class means, and $\Sigma$ is the class conditional covariance matrix. Let us define the fully labeled data set by

$$X_V = X \cup \{(u_i, v_i)\}_{i=1}^{M}$$

Then, $\widehat{\theta}_{\text{opt}}$ gives the parameter estimates of the classifier where the unlabeled data is also labeled.

$$\widehat{\theta}_{\text{opt}} = \underset{\theta}{\text{argmax}}\, L\left(\theta | X_V\right) \tag{5.5}$$

Since supervised parameters in $\widehat{\theta}_{\text{sup}}$ are estimated on a subset $X$ of $X_V$, we have

$$L\left(\widehat{\theta}_{\text{sup}} | X_V\right) \leq L\left(\widehat{\theta}_{\text{opt}} | X_V\right) \tag{5.6}$$

In semi-supervised setting, $V$ is unobserved, but we have $X$ (labeled data) and $U$ (unlabeled data). We have more information compared to supervised setting but less than the fully labeled case. Thus,

$$L\left(\widehat{\theta}_{\text{sup}} | X_V\right) \leq L\left(\widehat{\theta}_{\text{semi}} | X_V\right) \leq L\left(\widehat{\theta}_{\text{opt}} | X_V\right) \tag{5.7}$$

Now, we take the supervised estimate into account explicitly in order to construct a semi-supervised classifier than can improve upon its supervised counterpart.

Before doing so, we define $q_{ki}$ to be the hypothetical posterior of observing label $k$ given feature vector $u_i$. It can be also interpreted as the soft label for $u_i$. Since $\sum_{k \in C} q_{ki} = 1$, the $K$-dimensional vector $q_{\cdot i}$ can be stated as an element of the simplex $\Delta_{K-1}$ in $\mathbb{R}^K$:

$$q_{\cdot i} \in \Delta_{K-1} = \left\{ (\rho_1, \ldots, \rho_K)^T \in \mathbb{R}^K | \sum_{i=1}^{K} \rho_i = 1, \rho_i \geq 0 \right\} \tag{5.8}$$

Provided that the posterior probabilities are defined, the log-likelihood on the complete dataset for any parameter vector $\theta$ can be expressed as

$$L\left(\theta | X, U, q\right) = L\left(\theta | X\right) + \sum_{i=1}^{M} \sum_{k=1}^{K} q_{ki} \log p\left(u_i, k | \theta\right) \tag{5.9}$$

where the variable $q$ in left-hand side explicitly indicates the dependence on $q_{ki}$.

The relative improvement of the semi-supervised estimate $\theta$ over the supervised solution for a given $q$ can be expressed as

$$CL\left(\theta, \widehat{\theta}_{\text{sup}}|X, U, q\right) = L\left(\theta|X, U, q\right) - L\left(\widehat{\theta}_{\text{sup}}|X, U, q\right) \tag{5.10}$$

This enables us to check the extent of improvement of semi-supervised estimates in terms of log-likelihood, defined as *contrast*. Since $q$ is unknown, Equation 5.10 cannot be used directly in optimization. Hence, we choose the most pessimistic solution where we assume that the true (soft) labels achieve the worst-case among all semi-supervised solutions and consider the $q$ which minimizes the likelihood gain. Thus, our objective function can be written as:

$$CPL\left(\theta, \widehat{\theta}_{\text{sup}}|X, U\right) = \min_{q \in \Delta_{K-1}^{M}} CL\left(\theta, \widehat{\theta}_{\text{sup}}|X, U, q\right) \tag{5.11}$$

where $\Delta_{K-1}^{M} = \prod_{i=1}^{M} \Delta_{K-1}$ is the Cartesian product of $M$ simplices.

The objective function is strictly concave in $\theta$ and linear in $q$. The heuristic to solve the maximization problem is based on alternating between the following two steps:

1. Given a soft labeling $q$, the optimal $LDA$ parameters are estimated by

$$\widehat{\pi}_k = \frac{N_k + \sum_{i=1}^{M} q_{ki}}{N + M} \tag{5.12}$$

$$\widehat{\mu}_k = \frac{\sum_{j=1}^{N_k} x_{kj} + \sum_{i=1}^{M} q_{ki} u_i}{N_k + \sum_{i=1}^{M} q_{ki}} \tag{5.13}$$

$$\begin{aligned}\widehat{\Sigma} = \frac{1}{N+M} \sum_{k=1}^{K} [\sum_{j=1}^{N_k} \left(x_{kj} - \widehat{\mu}_k\right)\left(x_{kj} - \widehat{\mu}_k\right)^T \\ + \sum_{i=1}^{M} q_{ki} \left(u_i - \widehat{\mu}_k\right)\left(u_i - \widehat{\mu}_k\right)^T].\end{aligned}$$

2. The gradient $\nabla$ for $q$ given $\theta$ is calculated, and $q$ is changed to $q - \alpha\nabla$, with step size $\alpha > 0$. The step size $\alpha$ is decreased as one over the number of iterations and the maximum number of iterations to restricted to 3,000.

**Baseline Algorithms for Trajectory Classification** We compared the performance of our above CPLE-based framework for trajectory classification with respect to two baseline semi-supervised methods: Self Learning (Zhu and Goldberg, 2009) and Label Spreading (Zhou et al., 2004). Self Learning combines information from unlabeled data with the labeled data to iteratively identify the label of unlabeled data. The labeled training set is enlarged on each iteration until the entire dataset is labeled. LDA (Balakrishnama and Ganapathiraju, 1998) is used as the base model in Self Learning in this study. Label Spreading (Zhou et al., 2004), a modification of the traditional Label Propagation algorithm (Zhu and Ghahramani, 2002) uses an affinity matrix based on normalized graph Laplacian. It uses soft clamping for labeling and the loss function has regularization properties that make it robust to noise. Interested readers can refer to (Bengio et al., 2006) for further details. Besides these two baseline, we also compared the results of our algorithm to its supervised counterpart obtained from the LDA classifier trained on the labeled data.

**Feature Vector Generation** The trajectories obtained from the vehicle tracker module are of variable length (see Section 5.2.1.2). However, the semi-supervised techniques described above requires fixed-dimensional feature vectors. Hence, we first used trajectory subsampling to convert these variable length trajectories into fixed-length trajectories, similar to (Piciarelli et al., 2008). Each trajectory is subsampled to form a list of 2-D coordinates. We heuristically chose 75 as the fixed length of each of these lists. since the typical length of each trajectory is between 70 to 80. Thus, each trajectory can now be defined as $TR_i = p_1 p_2 p_3 ... p_j ... p_{75}$, where $p_j$ is the 2D vector representing $[X_j^c, Y_j^c]$. We normalized the feature vector to zero mean and performed Principal Component Analysis for dimension reduction. We found that 95% of variance (explained by top 3 principal components for $X^c$ and $Y^c$ each) is sufficient, similar to (Loog, 2016). Finally, the top 3 principal components for $X^c$ and $Y^c$ are concatenated to form a 6-D vector representing the trajectory information of each vehicle id. This 6-D feature vector is used for trajectory classification.

### 5.2.2 Pixel-based Incident Detection

Trajectory-based approaches for incident detection become infeasible with the increase in traffic density due to inherent difficulties in tracking individual vehicles in crowded conditions (Coşar et al., 2016). Hence, pixel-based approaches for traditional abnormal event detection are used in crowded conditions to detect abnormal shapes (e.g., vehicles in pedestrian zones) or abnormal motion (e.g., panic conditions, fighting conditions, etc.) or both. Since traffic incidents involves vehicles only, therefore we used YOLOv3 object detector (described in Section 5.2.1.1) to detect vehicles. Thereafter, optical flow motion estimation is extracted for each vehicle and then incident vehicles are identified based on supervised classification. The proposed methodology for pixel-based incident detection consists of four distinct steps: (a) vehicle detection, (b) optical flow estimation, (c) motion feature extraction, and (d) supervised motion feature classification for incident identification. As described before, YOLOv3 object detector is used for vehicle detection. We next describe the remaining three steps for pixel-based incident detection.

#### 5.2.2.1 Optical Flow Estimation

Optical flow estimation involves computation of the approximate 2-d *motion field* projecting the 3-d motion of objects onto the imaging surface (Verri and Poggio, 1989). Therefore, while *motion field* can be defined as the 2-d projection of the 3-d motion of surfaces, *optical flow*, on the other hand, is the *apparent* motion of brightness patterns in the image (Baker et al., 2011). In general, since we can only observe optical flow instead of the true motion field, we assume *optical flow* and *motion field* are not too different (Verri and Poggio, 1989).

Let the intensity of a pixel $(x, y)$ at time $t$ is given by $I(x, y, t)$. Similarly, let the flow is denoted by $u(x, y, t)$ and $v(x, y, t)$. With the assumption that when a pixel moves between two consecutive frames, its intensity and color remains same (*Brightness Constancy*), can we written as:

$$I(x, y, t) = I(x + u, y + v, t + 1) \tag{5.14}$$

Applying first-order Taylor expansion to Equation 5.14 yields:

$$I(x, y, t) = I(x, y, t) + u\frac{\partial I}{\partial x} + v\frac{\partial I}{\partial y} + 1\frac{\partial I}{\partial t} \tag{5.15}$$

which can be simplified to *Optical Flow Constraint Equation*:

$$u\frac{\partial I}{\partial x} + v\frac{\partial I}{\partial y} + 1\frac{\partial I}{\partial t} = 0 \tag{5.16}$$

*Brightness Constancy* and *Optical Flow Constraint Equation* equations provide one constraint for two unknowns $(u, v)$. In this study, we used Gunnar-Farneback algorithm (Farnebäck, 2003) to solve this problem and estimate the dense optical flow for each pixel in two consecutive frames. This consists of three main steps:

1. Each pixel's neighbors in both the frames are approximated
2. The inclusive displacement of the pixels within the image is used to built a new signal
3. Finally, the displacement is computed equating the coefficients in the quadratic polynomials

We used Farneback algorithm to extract optical flow for each vehicle, detected by the YOLOv3 object detector. This extracted vehicle optical flow is then used for motion feature extraction, as described next.

#### 5.2.2.2 Motion Feature Extraction: Histogram of Optical Flow Magnitude

In this study, we propose histogram of optical flow magnitude (HOFM) as the motion feature extractor. Traditional motion feature extractor such as histogram of optical flow (Perš et al., 2010), histogram of oriented optical flow (Chaudhry et al., 2009) attempts to find out the orientations of optical flow to extract abnormal events. However, magnitude of optical flow in detected vehicles is an useful feature instead of its orientation for incident detection. Therefore, we propose to use HOFM in this study as motion feature extraction. In HOFM, we bin the extracted optical flow magnitude of each detected vehicle into 10 classes [0-1, 1-2, ..., 8-9, ¿9] and normalize the frequencies to obtain relative frequency histogram, denoted as HOFM. This feature vector is then used for incident detection,as described next.

#### 5.2.2.3 Incident Detection using HOFM Feature Vector

In this study, we used Support Vector Machine (SVM) to detect incident vehicles in images using the extracted HOFM feature vector from annotated incident and non-incident vehicles.

SVM's principle is based on structural risk minimization by minimizing the upper bound of the generalization error (Cortes and Vapnik, 1995). Given data input $x_i, i = 1, 2, ..., N$, where $N$ is the number of samples. For binary classification of linearly separable data, the hyperplane $f(x) = 0$ can be determined such that:

$$f(x) = w^T x + b = \sum_{i=1}^{N} w_i x_i + b = 0 \tag{5.17}$$

where, $w$ is $N$-dimensional vector and $b$ is a scalar quantity.

This boundary hyperplane is determined such that the distance between the boundary and the nearest data point (also, referred to as support vectors) for each class is maximal. The optimal hyperplane determination can be obtained by solving the following optimization problem:

$$\text{minimize} \tfrac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i \tag{5.18}$$

$$\text{subject to} \begin{cases} y_i \left( w^T x_i + b \right) \geq 1 - \xi_i, i = 1, ..., N \\ \xi_i \geq 0, \ i = 1, ..., N \end{cases} \tag{5.19}$$

where, $C$ is the error penalty and $\xi_i$ is the distance between the boundary and the sample points.

For non-linear classification types, SVM can be used by mapping the data points onto a high-dimensional feature space using kernel functions such that linear classification is possible (Boser et al., 1992). Different kernel functions can be used in SVMs, for example, linear, polynomial, and Gaussian RBF. In this study, we used Gaussian RBF, linear, and polynomial kernels to classify incident and non-incident vehicles. Interested readers can refer to Scholkopf and Smola (2001) for further details on SVM. The detected vehicles in the videos which are classified as incident vehicles by SVM classifier are then tracked using SORT tracker, described in Section 5.2.1.2. This helps to remove noisy classification errors and false alarms.

## 5.3 Data Description

The primary source of data in this study are the traffic incident videos obtained from the CCTV cameras installed by the Iowa Department of Transportation (DoT) along the freeways of Iowa.

Our dataset consists of 201 traffic incident videos recorded from these cameras during the period of 2016-2017. Out of these 201 incident videos, 151 were non-congested videos while the remaining 50 videos were congested videos, detected using YOLO congestion detection classifier desribed in Chapter 4 and also manually verified. Each video is of two-minutes duration recorded at 30 frames per second and clearly captures the onset of the traffic incident. The resolution of videos varies from 800×480 pixels to 1920×1080 pixels depending on the camera resolution. The traffic incidents are caused due to car crashes or stalled vehicles. Out of the 151 non-congested incident videos, we manually annotated 11 videos with the bounding boxes of the vehicles involved in the incident. A javascript based video annotation tool (Bolkensteyn, 2016) based on VATIC (Vondrick et al., 2013) is used for annotating the vehicles. This resulted in a total of 15 unique trajectories of vehicles involved in incidents. These trajectories are then matched with the vehicle trajectories obtained from object detection and tracking modules used in this study (YOLOv3 for vehicle detection and SORT for vehicle tracking). For each frame, each manually annotated bounding box is matched with the detected bounding box with maximum overlap, setting a minimum threshold of 0.5 Intersection over Union (IoU).Each manually annotated incident trajectory was successfully matched with unique trajectory obtained from the tracking algorithm. These trajectories are henceforth referred to as incident trajectories. The remaining trajectories in those 11 incident videos are classified as normal trajectories. We randomly selected 15 such normal trajectories into our labeled dataset. Thus, our labeled dataset consists of 15 normal trajectories and 15 incident trajectories.

For trajectory-based incident detection, we randomly selected 90 incident videos from the 151 congested incident videos for our unlabeled dataset preparation. The 11685 trajectories obtained by the object detection and tracking algorithm from those 90 videos are included in the unlabeled dataset. The remaining 50 incident videos are equally divided into validation and test data sets, with 25 incident videos in each set. We also randomly selected 50 baseline videos without any incidents and split them equally into validation and test set. Thus, our validation dataset and test dataset consist of 50 videos each, 25 of them being incident videos and remaining 25 being normal baselines videos. Our validation and test datasets consist of 6333 and 5375 trajectories respectively.

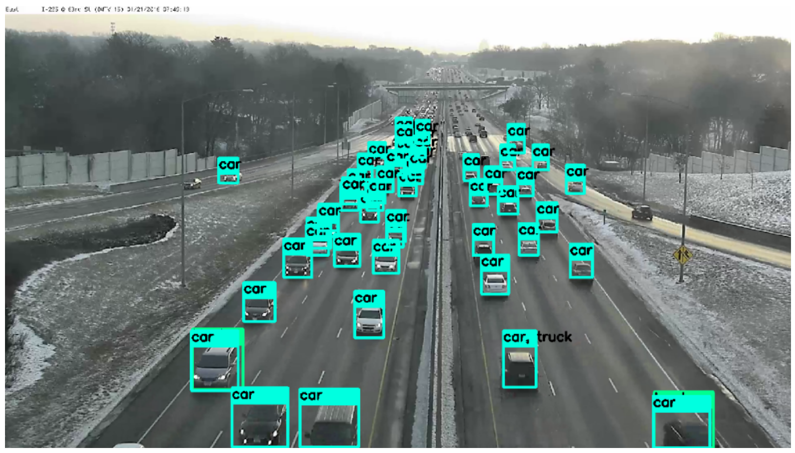
For pixel-based incident detection, the training sample was formed using the same 11 manually annotated videos, used for trajectory-based detection. We extracted 11,345 incident vehicle instances from the 15 unique trajectories of the 11 videos. The videos are sampled at 1 frames per second to handle over-representation of same vehicles within the training data. To test our model's performance similar to trajectory-based incident detection, we used 50 congested incident videos, equally divided into validation and test data sets, with 25 incident videos in each set. We also randomly selected 50 baseline congested videos without any incidents and split them equally into validation and test set.

## 5.4 Results

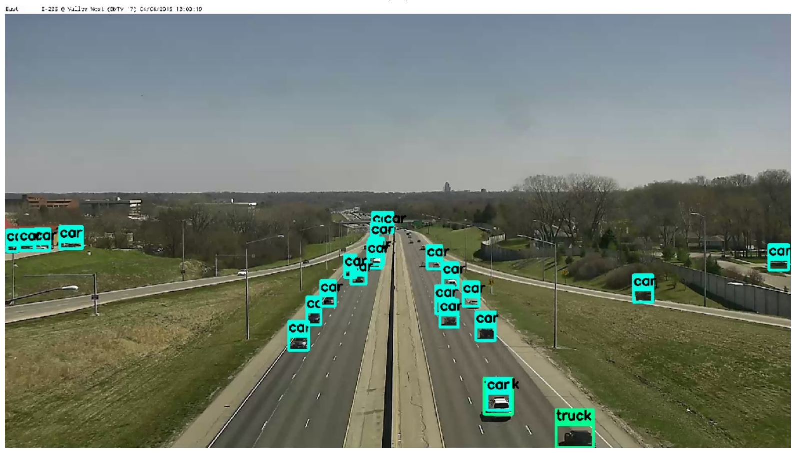
### 5.4.1 Trajectory-based Incident Detection

We used the state-of-the-art object detection algorithm YOLOv3 (Redmon and Farhadi, 2018) for vehicle detection and SORT (Bewley et al., 2016) for vehicle tracking. The object detection and tracking runs at around 45 frames per second (fps) on an NVIDIA GTX 1080Ti GPU, making it suitable for real-time performance. Figure 5.3 shows sample images of the vehicles detected. Figure 5.4 shows a sample image of vehicle trajectories, where each color represents a unique trajectory.

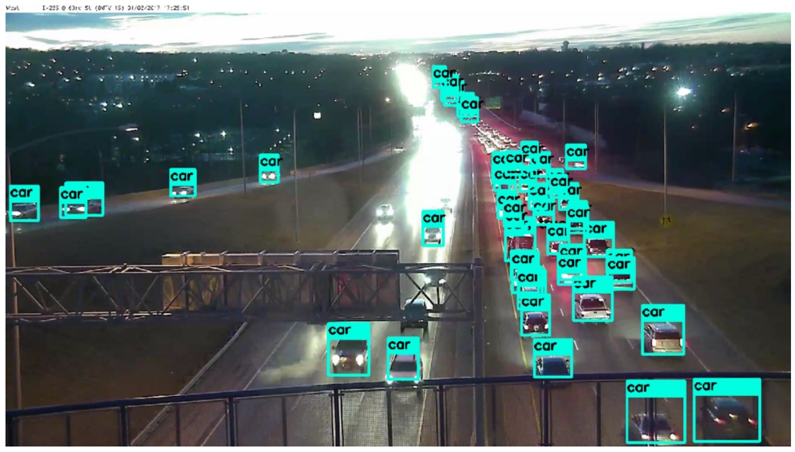
Our labeled trajectory dataset consists of 15 incident trajectories and 15 normal trajectories. To find out the sensitivity of the algorithm on the number of labeled examples, we ran each algorithm for label sample sizes varying from 5-15 trajectories for each class (normal and incident). The efficacy of the proposed model (CPLE) along with the comparison models (Label Spreading, Self Learning and Supervised Learning) are validated using the validation dataset and the final accuracy is reported for the test dataset. We label a video as an incident video if at least 1 trajectory in the video is classified as incident trajectory by the algorithm. The accuracy of the algorithm ($ACC$) is given by the accuracy of correctly classifying incident videos ($TPR$) and baseline videos ($TNR$), as shown in Equations 5.22, 5.20, and 5.21. $TP$ and $TN$ refer to the number of correctly identified



(a)



(b)



(c)

Figure 5.3: Sample images of vehicle detections using YOLOv3 object detector

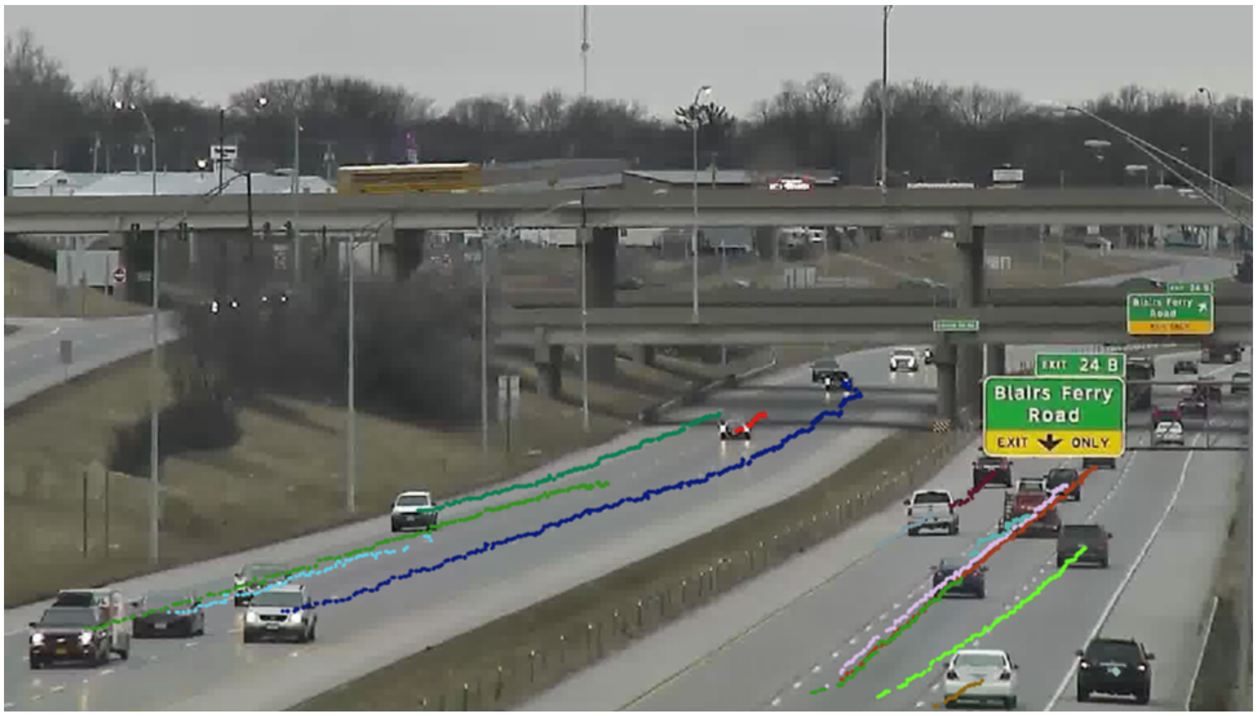


Figure 5.4: Vehicle detection and tracking sample

incident and baseline videos while $P$ and $N$ refer to total number of incident and baseline videos (25 each).

$$TPR = \frac{TP}{P} \tag{5.20}$$

$$TNR = \frac{TN}{N} \tag{5.21}$$

$$ACC = \frac{TP + TN}{P + N} \tag{5.22}$$

Figure 5.5 shows the accuracy of each algorithm on the validation dataset for different number of labeled samples. The experiments are repeated 20 times and the average accuracy is reported. We clearly see that CPLE performs superior compared to the other semi-supervised approaches and its supervised counterpart. On an average, 14% improvement is obtained on using CPLE compared to the second best algorithm (Label Spreading). Further, accuracy obtained by Self Learning, a baseline semi-supervised algorithm, drops below supervised classifier's accuracy even with the increase of the number of labeled samples beyond 18. This shows that, baseline semi-supervised algorithm cannot guarantee improvement in accuracy even with the increase of labelled samples, unlike CPLE. However, in future, number of samples can be increased further to determine the number of labeled samples required such that supervised classifiers can work better or at-least comparable to CPLE method.

The best model obtained from each algorithm is selected and applied on the test dataset. Table 5.1 shows the accuracy of each algorithm on the test dataset. It shows that while CPLE successfully identifies a large majority of the incident videos (21 out 25 incident videos), other algorithms fail to do so and perform poorly in $TPR$. However, since majority of trajectories are normal trajectories, all algorithms perform well in detecting the baseline videos correctly. This shows that the CPLE algorithm successfully extracts information regarding both incident and normal trajectories from the unlabeled dataset and hence achieves better performance.

Figure 5.6 shows a sample of incident and normal trajectories labeled by the CPLE algorithm for 3 incident videos (ID 1, 2, and 3). The X and Y coordinates of the bounding box center of each vehicle across the video frames is shown in the figure. It successfully detects the incident

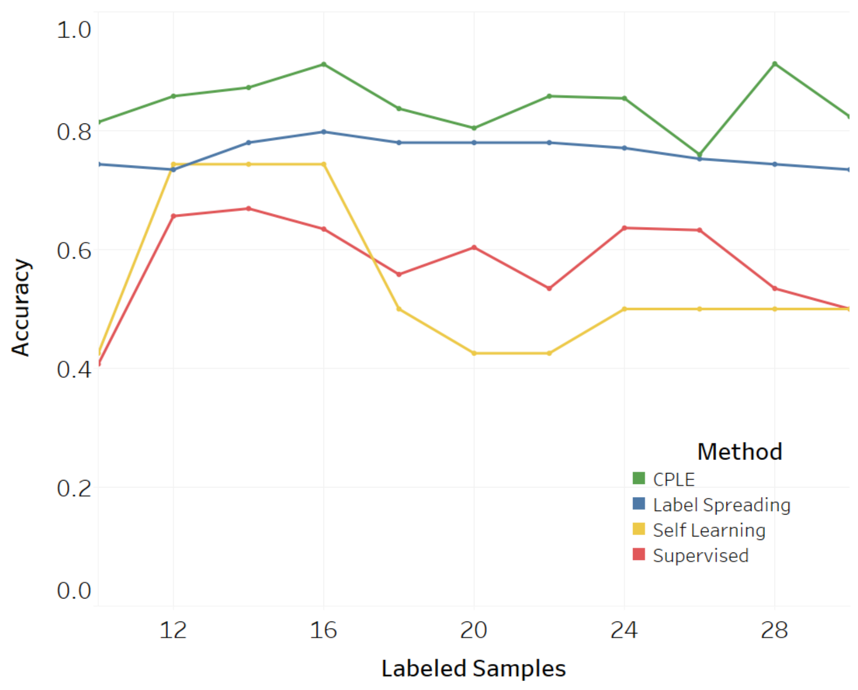


Figure 5.5: Accuracy of the algorithms on number of labeled samples

Table 5.1: Trajectory-based incident detection performance on non-congested test videos

| Method | TPR | TNR | ACC |
|---|---|---|---|
| CPLE | 0.83 | 0.92 | 0.88 |
| Label Spreading | 0.6 | 0.94 | 0.77 |
| Self Learning | 0.53 | 0.88 | 0.71 |
| Supervised | 0.28 | 0.96 | 0.62 |

trajectories in Video ID 1 and 2, but fails to detect any incident trajectory in ID 3, primarily due to missing object detection caused by poor video quality. A sample image of a stalled vehicle detected across 3 frames is shown in Figure 5.7.

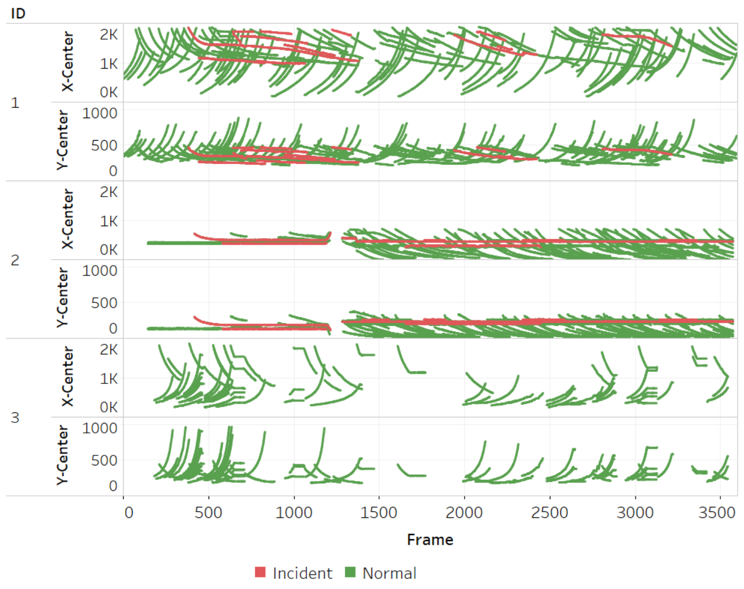


Figure 5.6: Incident and normal trajectories labeled by CPLE algorithm for 3 incident videos

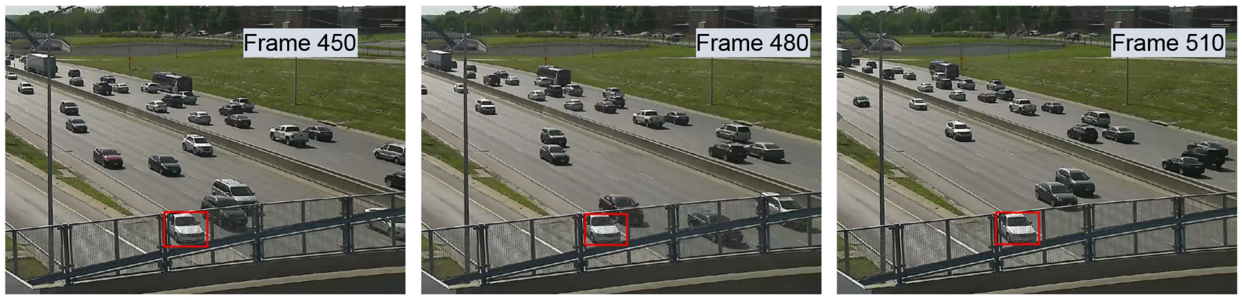


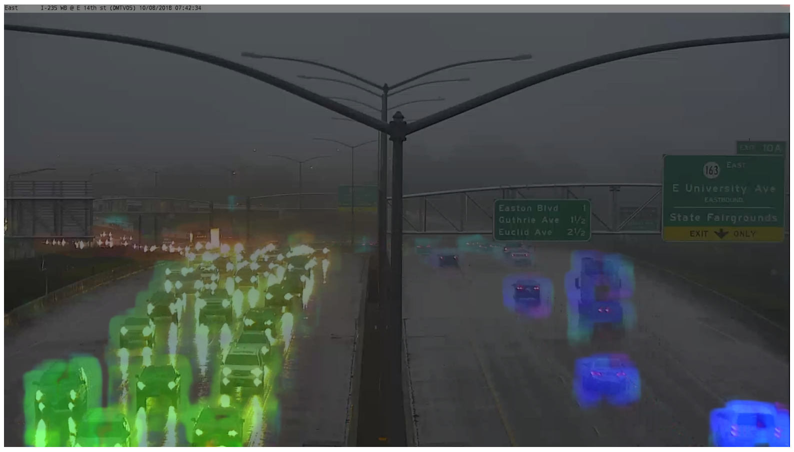
Figure 5.7: Sample images of stalled vehicle detected across 3 frames taken at 1-second interval

### 5.4.2 Pixel-based Incident Detection

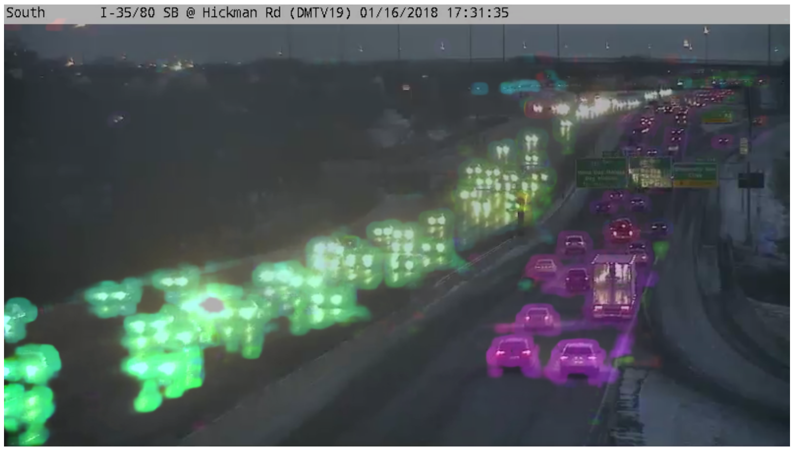
We used the state-of-the-art object detection algorithm YOLOv3 (Redmon and Farhadi, 2018) for vehicle detection and Gunnar-Farneback algorithm (Farnebäck, 2003) for optical flow estimation. The combined object detection and optical flow estimation runs at around 1.4 frames per second (fps) on an NVIDIA GTX 1080Ti GPU. Figure 5.3 shows sample images optical flow estimation. HOFM feature vector is calculated for each detected vehicle (sampled at 1 frames per second) and used to train the SVM classifier. We test our model on the similar 50 non-congested test videos (25 incident and 25 non-incident videos) and also on 50 congested test videos (25 incident and 25 non-incident videos). Figures 5.11 and 5.13 show samples of successful incident detection during non-congested and congested conditions respectively across 3 frames at 1-second interval using pixel-based approach. It can be seen that this approach can successfully detect such incident vehicles in both congested and non-congested conditions. Table 5.2 shows the performance of pixel-based incident detection on non-congested and congested test videos using $TPR$, $TNR$, and $ACC$ metrics. As it can be seen, that the proposed methodology achieves 0.72 accuracy for congested traffic conditions. Camera shaking, poor resolution, and extensively high congested conditions were found to some of the cases where the proposed methodology performed poorly. For non-congested conditions, the accuracy for pixel-based approach (0.83) is slightly less than that obtained using trajectory-based approach (0.88). Also, trajectory-based approach runs much faster (45 fps) compared to pixel-based approach (1.4 fps). Hence, trajectory-based approach can be used during non-congested conditions, while pixel-based approach can be used only during congested conditions to handle such difficult crowded conditions. This can help to optimize system performance and perform incident detection using cameras for large networks during both congested and non-congested conditions. In future, this can be extended to compute optical flow by creating masks outside the detected vehicle regions so that optical flow computation speed can be increased significantly.

Table 5.2: Pixel-based incident detection performance on congested and non-congested test videos

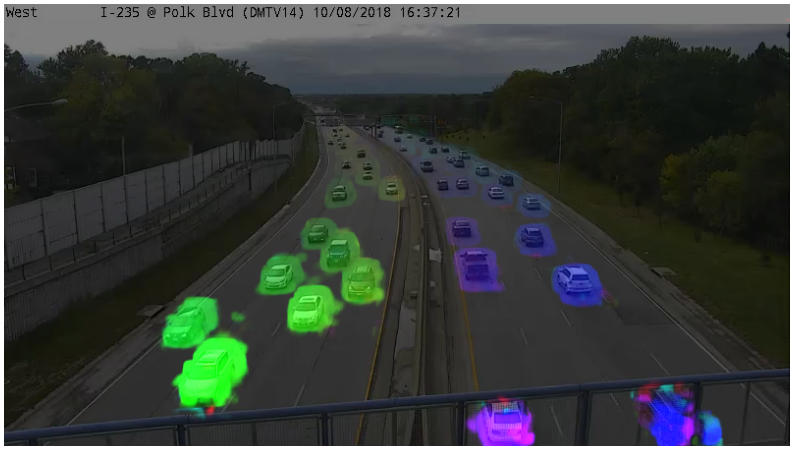
| Traffic Conditions | TPR | TNR | ACC |
|---|---|---|---|
| Non-Congested | 0.87 | 0.81 | 0.83 |
| Congested | 0.77 | 0.67 | 0.72 |



(a)

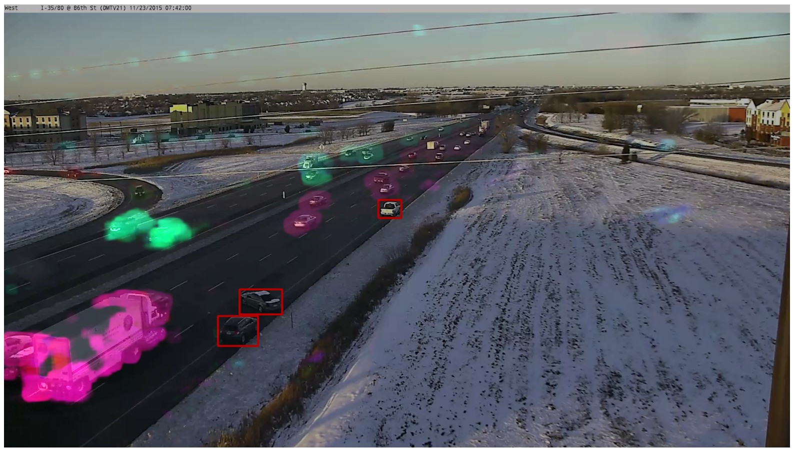


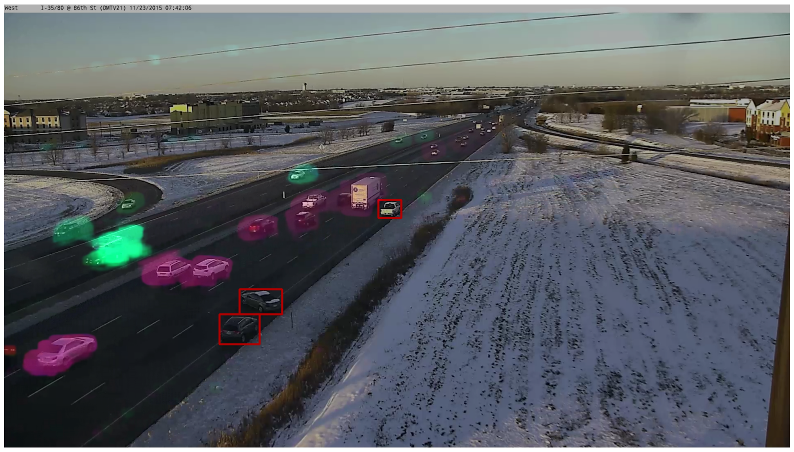
(b)



(c)

Figure 5.9: Sample images of optical flow estimation using Gunnar-Farneback algorithm


(a)


(b)

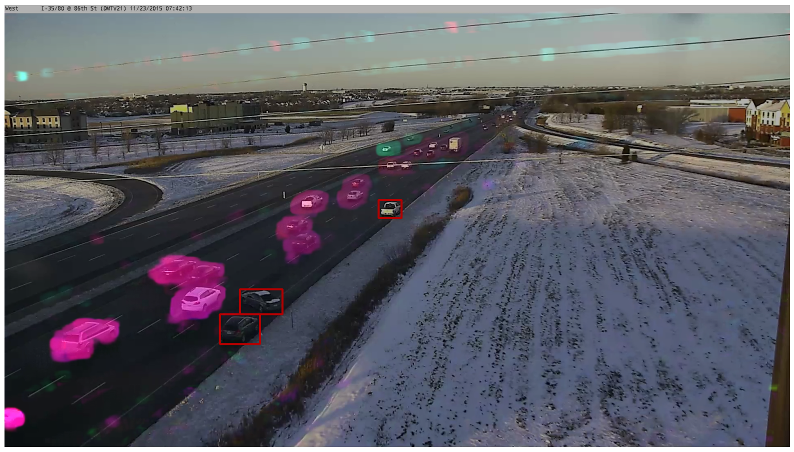
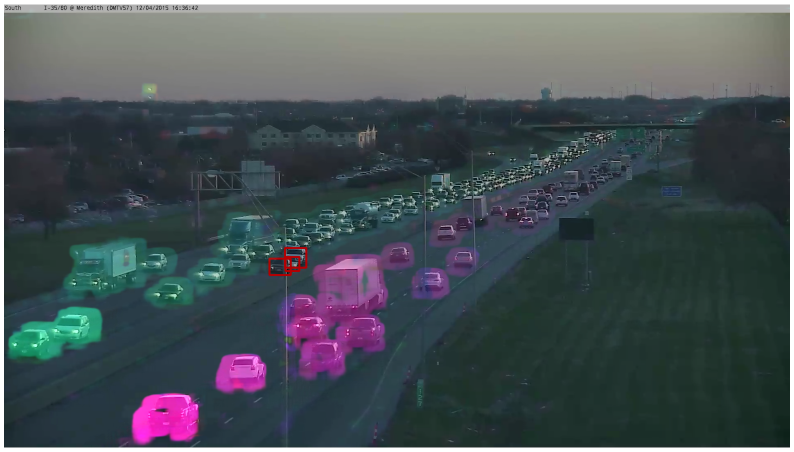
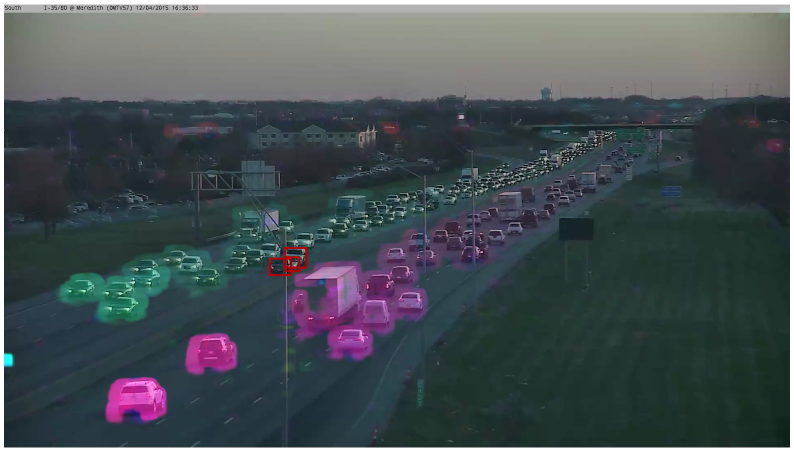

(c)

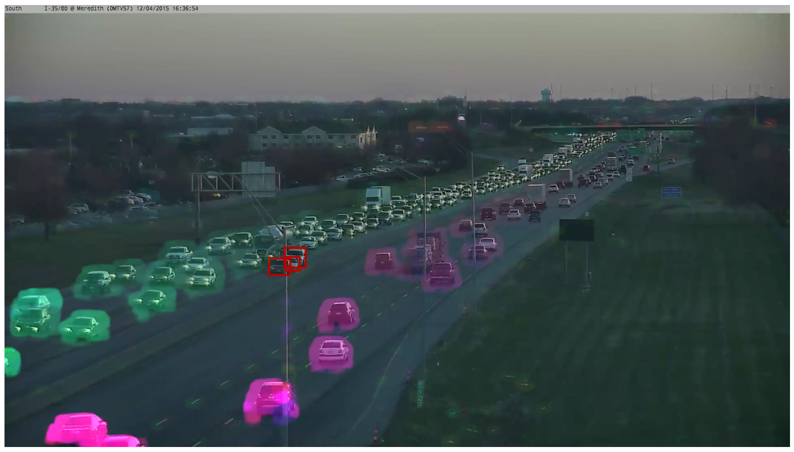
Figure 5.11: Sample images of pixel-based incident detection across 3 frames (a) Frame 210, (b) Frame 240, and (c) Frame 270 at 1-second interval in non-congested conditions



(a)



(b)



(c)

Figure 5.13: Sample images of pixel-based incident detection across 3 frames (a) Frame 210, (b) Frame 240, and (c) Frame 270 at 1-second interval in congested conditions

## 5.5 Conclusions

State Department of Transportation (DOTs) usually install a large number of CCTV cameras across freeways for surveillance purpose. However, it is virtually impossible to manually monitor such a large network of cameras constantly. Hence, there is a significant need to develop automatic incident detection algorithms using these these cameras.

In this study, we approached the incident detection problem using trajectory-based approach for non-congested conditions and pixel-based approach for congested conditions. Trajectory-based incident detection is handled using semi-supervised techniques. We used Maximum Likelihood Estimation based Contrastive Pessimistic Likelihood Estimation (CPLE) for trajectory classification and identification of incident trajectories. Vehicle detection is performed using state-of-art deep learning based YOLOv3 and SORT tracker is used for tracking. Results show that CPLE based trajectory classification outperforms the traditional semi-supervised techniques (Self Learning and Label Spreading) and also its supervised counterpart by significant margin.

For pixel-based incident detection, we used a novel Histogram of Optical Flow Magnitude (HOFM) feature descriptor to detect incident vehicles using SVM classifier based on all vehicles detected by YOLOv3 object detector. We show in this study that this approach can handle both congested and non-congested conditions. However, trajectory-based approach works considerably faster (45 fps compared to 1.4 fps) and also achieves better accuracy compared to pixel-based approach for non-congested conditions. Therefore, for optimal resource usage, trajectory-based approach can be used for non-congested traffic conditions while for congested conditions, pixel-based appraoch can be used.

# CHAPTER 6. CONCLUSIONS

Automatic traffic incident detection has been identified to be crucial for reduction of non-recurrent congestion caused by incidents. In this study, we proposed an AID framework utilizing large scale traffic data, images and videos from CCTV cameras. This study is divided into three broad topics involving detection of freeway traffic incidents from these data sources.

## 6.1 Large-Scale Traffic Speed Data-Driven Incident Detection

Our first research objective involves development of an AID framework can be implemented over large traffic networks due to the inherent parallel computation involved in the framework. Automatic traffic incident detection has been identified to be crucial for the reduction of non-recurrent congestion caused by incidents. In this study, we proposed an AID framework can be implemented over large traffic networks due to the inherent parallel computation involved in the framework. The proposed methodology consists of two major improvements to the AID algorithms developed in past literature. The first step of our AID framework consists of the estimation of robust summary statistics (speed thresholds) across non-overlapping windows of time series data produced from each road segment. This dimension reduction step can be parallelized using MapReduce framework making it feasible to apply the algorithm over large traffic networks. We then visualize these summary statistics as threshold heatmaps leveraging the spatio-temporal correlations of the time windows and perform multivariate denoising of the heatmaps. Such denoised thresholds can produce more accurate representations of the normal traffic patterns which in turn can help in increasing incident detection performance.

To produce robust summary statistics for the non-overlapping time windows of each time series traffic data, we proposed MAD and IQD methods in place of the popular SND algorithm. In these methods, we replace the mean and standard deviation measures which are known to be prone to outliers with the comparatively robust statistics such as median, MAD, and IQD. SND being prone to outliers is found to be affected by occasional low speeds caused due to incidents or traffic constructions. This results in lowering speed thresholds which in turn produces low incident detection rates while having false alarm rates comparable to IQD. MAD, on the other hand, is found to produce high threshold values even during peak hours thereby resulting in significantly higher false alarm rates than SND or IQD. Our results show that the IQD method, which uses median and IQD as summary statistics, achieves the best performance overall in terms of incident detection. It can achieve high incident detection rates without increasing false alarm rates significantly.

In our next step, we construct heatmaps with the IQD thresholds and perform image denoising of the heatmaps. We used two edge-preserving image denoising techniques, bilateral filter and total variation. Our results show that speed thresholds increase in the recurrent congestion patches due to total variation denoising. As a result, the detection rate decreases and the overall performance, given by PI, deteriorates. Bilateral filter, on the other hand, preserves the edges and the low thresholds in recurrent congestion patches intact and simultaneously removes spurious noisy thresholds. This helps in achieving higher detection rates with bilateral filter denoising without increasing false alarms significantly thereby improving the overall performance of the AID algorithm.

One of the major contributions of the study is the development of an AID algorithm that can be easily transferred and implemented over large-scale traffic networks. With the advent of big-data tools, data-driven methodologies can now be applied to massive datasets instead of searching for statistical subsampling for data manipulation and aggregation. In this study, we have preferred to use simple data-driven approaches that essentially predicts a different threshold range for each sub-segment instead of finding a transferable statistically complex technique which can be generalized to multiple segments. In other words, it uses big-data manipulation tools to generate site-specific indices instead of searching for complicated models which essentially try to make the predictions transferable either to account for non-observability or reduced computation power during inferencing. The entire framework is data-driven in nature and can be easily extended in other data-set or traffic networks.

## 6.2 Congestion Detection from Camera Images

Recent advancements in machine-vision algorithms and high performance computing have improved image classification accuracy to a great extent. In In our second part of the study, two such deep learning techniques, the traditional DCNN and YOLO models, are used to detect traffic congestion from camera images. SVM is also used for comparison and to determine what improvements are obtained while using costly GPU techniques. To eliminate the time-consuming task of manual labeling and to maintain uniformity in congestion labeling, we used nearby Wavetronix sensors to correctly identify congested images. For testing purposes, we also labeled each image manually to remove misclassifications due to sensor errors.

The YOLO model achieved the highest accuracy of 91.2% followed by DCNN with an accuracy of 90.2%; 85% of images were correctly classified by SVM. Congestion regions located far away from the camera, single-lane blockages, and glare issues were found to affect the accuracy of the models. To determine the sensitivity of the models to different camera configurations and light conditions, ROC curves are used. All the algorithms are found to perform well in daytime conditions, but night conditions are found to affect the accuracy of the vision system. However, for all conditions, the AUCs are found to be greater than 0.9 for the deep models. This shows that the models perform well in challenging conditions as well.

An example of the real-time implementation of congestion detection using DCNN algorithm is also performed using a continuous set of images extracted from a camera. Simple persistence test methods are applied to reduce the false alarms and smoothen the output signal. Future studies can look into different smoothing techniques (Fourier Transform, Wavelets) to denoise the output obtained from the algorithm and determine the overall detection rate and false alarm rates on a network of cameras. Future studies can also be done using different model architectural designs to improve detection accuracies. Such models can also be used for determining different levels of congestion (high, medium, or low) and also more accurate traffic state determination (speed, volume and occupancy).

## 6.3 Freeway Incident Detection from Camera Videos

The third part of this study aims at detecting traffic incidents from CCTV videos. State Department of Transportation (DOTs) usually install a large number of CCTV cameras across freeways for surveillance purpose. However, it is virtually impossible to manually monitor such a large network of cameras constantly. Hence, there is a significant need to develop automatic incident detection algorithms using these these cameras. In this study, we approached the incident detection problem using trajectory-based approach for non-congested conditions and pixel-based approach for congested conditions. Usually trajectory-based incident detection from cameras have been approached using either supervised or unsupervised algorithms. One of the major hindrance in application of supervised techniques for incident detection is the lack of incident videos and the labor intensive, costly annotation tasks involved in preparation of labeled dataset. In this study, we approached the incident detection problem using semi-supervised techniques. We used Maximum Likelihood Estimation based Contrastive Pessimistic Likelihood Estimation (CPLE) for trajectory classification and identification of incident trajectories. Vehicle detection is performed using state-of-art deep learning based YOLOv3 and SORT tracker is used for tracking. Results show that CPLE based trajectory classification outperforms the traditional semi-supervised techniques (Self Learning and Label Spreading) and also its supervised counterpart by significant margin.

For pixel-based incident detection, we used a novel Histogram of Optical Flow Magnitude (HOFM) feature descriptor to detect incident vehicles using SVM classifier based on all vehicles detected by YOLOv3 object detector. We show in this study that this approach can handle both congested and non-congested conditions. However, trajectory-based approach works considerably faster (45 fps compared to 1.4 fps) and also achieves better accuracy compared to pixel-based approach for non-congested conditions. Therefore, for optimal resource usage, trajectory-based approach can be used for non-congested traffic conditions while for congested conditions, pixel-based approach can be used.

## 6.4 Limitations and Future Work

This study focused on using traffic speed data and cameras for detecting freeway traffic incidents. The data-driven AID framework can be easily extended in other data-set or traffic networks. However, real-world implementation of such an AID framework will also involve some thoughtful decision making and challenges which can be addressed in future studies. First, state or federal transportation agencies can decide to strike a balance between $DR$, $FAR$, and $MTTD$ based on their specific requirements. $PI$ is one such measure that has been used in this study for this purpose. However, the agencies can decide on the minimum $DR$ or maximum allowable $FAR$ based on their needs for optimal performance and reliability of the AID algorithm. Second, the present AID framework doesn't take into consideration abrupt weather changes or other scheduled events such as snowstorms, game days, etc. which can impact traffic significantly. Detecting traffic incidents during such extreme event days is challenging and requires further research. Third, the impacts of missing data and other data quality issues on AID performance needs to be investigated for real-time implementation purpose. Future studies can look into these aspects to build a more robust AID algorithm. Extending this AID framework to complicated traffic networks will also of significant interest and scope of research in future. There is also a scope of further improvements on the AID framework presented here. This study assumed uniform weights for the road segments while constructing threshold heatmaps. In future, it can be improved by using weights based on the road segment length or finding out the optimal weights using data-driven methods. The denoising framework can also be parallelized thereby unifying the entire system setup. And other advanced image denoising techniques can be applied to obtain better estimates of the thresholds from heatmaps.

Future studies can also be done using different model architectural designs to improve the proposed congestion detection accuracies. Such models can also be used for determining different levels of congestion (high, medium, or low) and also more accurate traffic state determination (speed, volume and occupancy). Different smoothing techniques (Fourier Transform, Wavelets) to denoise the output obtained from the algorithm and determine the overall detection rate and false alarm rates on a network of cameras. Congestion status obtained from the cameras can also be stored as historical data and used to determine traffic anomalies such as incidents. Incident detection using cameras can also be implemented on large-scale traffic camera network to determine the detection performance and false alarm rates. Also, the performance of the proposed camera-based incident detection algorithm can be compared with the algorithms developed by the winning teams of AI City Challenge 2018 and 2019 (Naphade et al., 2018, 2019). Finally, incident detection reliability can be improved significantly by fusing the decisions obtained from multiple data sources. Such an unified AID framework can use the different available data streams to detect incidents reliably with lower false alarm rates.

# BIBLIOGRAPHY

Adu-Gyamfi, Y. O., Asare, S. K., Sharma, A., and Titus, T. (2017). Automated vehicle recognition with deep convolutional neural networks. *Transportation Research Record: Journal of the Transportation Research Board*, 2645:113–122.

Aggarwal, C. C. (2007). *Data Streams: Models and Algorithms*, volume 31. Springer Science & Business Media.

Ahmed, E., Jones, M., and Marks, T. K. (2015). An Improved Deep Learning Architecture for Person Re-Identification. *Computer Vision and Pattern Recognition (CVPR)*, pages 3908–3916.

Ahsani, V., Amin-Naseri, M., Knickerbocker, S., and Sharma, A. (2018). Quantitative analysis of probe data characteristics: Coverage, speed bias and congestion detection precision. *Journal of Intelligent Transportation Systems*, 0:1–17.

Amin-Naseri, M., Chakraborty, P., Sharma, A., Gilbert, S. B., and Hong, M. (2018). Evaluating the reliability, coverage, and added value of crowdsourced traffic incident reports from waze. *Transportation Research Record: Journal of the Transportation Research Board*, 0(0).

Anantrasirichai, N., Nicholson, L., Morgan, J. E., Erchova, I., Mortlock, K., North, R. V., Albon, J., and Achim, A. (2014). Adaptive-weighted bilateral filtering and other pre-processing techniques for optical coherence tomography. *Computerized Medical Imaging and Graphics*, 38(6):526–539.

Anbaroglu, B., Heydecker, B., and Cheng, T. (2014). Spatio-temporal clustering for non-recurrent traffic congestion detection on urban road networks. *Transportation Research Part C: Emerging Technologies*, 48:47–65.

Andriyenko, A., Schindler, K., and Roth, S. (2012). Discrete-continuous optimization for multi-target tracking. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1926–1933. IEEE.

Apache Pig (2018). https://pig.apache.org/. accessed July 20, 2018.

Asakura, Y., Kusakabe, T., Long, N. X., and Ushiki, T. (2015). Incident detection methods using probe vehicles with on-board gps equipment. *Transportation Research Procedia*, 6:17–27.

Asakura, Y., Kusakabe, T., Nguyen, L. X., and Ushiki, T. (2017). Incident detection methods using probe vehicles with on-board GPS equipment. *Transportation Research Rart C: Emerging Technologies*, 81:330–341.

Asmaa, O., Mokhtar, K., and Abdelaziz, O. (2013). Road traffic density estimation using microscopic and macroscopic parameters. *Image and Vision Computing*, 31(11):887–894.

Bachmann, C., Abdulhai, B., Roorda, M. J., and Moshiri, B. (2013). A comparative assessment of multi-sensor data fusion techniques for freeway traffic speed estimation using microsimulation modeling. *Transportation Research Part C: Emerging Technologies*, 26:33–48.

Bae, S.-H. and Yoon, K.-J. (2017). Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):595–610.

Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M. J., and Szeliski, R. (2011). A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31.

Balakrishnama, S. and Ganapathiraju, A. (1998). Linear discriminant analysis-a brief tutorial. *Institute for Signal and Information Processing*, 18:1–8.

Balcilar, M. and Sönmez, A. C. (2008). Extracting vehicle density from background estimation using Kalman filter. In *2008 23rd International Symposium on Computer and Information Sciences, ISCIS 2008*.

Baldi, P., Brunak, S., Chauvin, Y., Andersen, C. A., and Nielsen, H. (2000). Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5):412–424.

Balke, K., Dudek, C., and Mountain, C. (1996). Using probe-measured travel times to detect major freeway incidents in Houston, Texas. *Transportation Research Record: Journal of the Transportation Research Board*, 1554:213–220.

Bauza, R., Gozalvez, J., and Sanchez-Soriano, J. (2010). Road traffic congestion detection through cooperative Vehicle-to-Vehicle communications. In *Proceedings - Conference on Local Computer Networks, LCN*, pages 606–612.

Ben-David, S., Lu, T., and Pál, D. (2008). Does unlabeled data probably help? Worst-case analysis of the sample complexity of semi-supervised learning. In *21st Annual Conference on Learning Theory COLT*, pages 33–44.

Bengio, Y., Delalleau, O., and Le Roux, N. (2006). *Label Propagation and Quadratic Criterion*. MIT Press.

Berclaz, J., Fleuret, F., Turetken, E., and Fua, P. (2011). Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1806–1819.

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE.

Bolkensteyn, D. (2016). Vatic.js. https://github.com/dbolkensteyn/vatic.js.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

Castro-Neto, M., Han, L., Jeong, Y.-S., and Jeong, M. (2012). Toward training-free automatic detection of freeway incidents: Simple algorithm with one parameter. In *Transportation Research Record: Journal of the Transportation Research Board, No. 2278*, pages 42–49. Transportation Research Board of the National Academies, Washington, D.C.

Chakraborty, P., Adu-Gyamfi, Y. O., Poddar, S., Ahsani, V., Sharma, A., and Sarkar, S. (2018a). Traffic congestion detection from camera images using deep convolution neural networks. *Transportation Research Record: Journal of the Transportation Research Board*, 2672(45):222–231.

Chakraborty, P., Hegde, C., and Sharma, A. (2017a). Trend filtering in network time series with applications to traffic incident detection. In *Time Series Workshop, 31st Conference on Neural Information Processing Systems (NIPS)*.

Chakraborty, P., Hegde, C., and Sharma, A. (2019). Data-driven parallelizable traffic incident detection using spatio-temporally denoised robust thresholds. *Transportation Research Part C: Emerging Technologies*, 105:81–99.

Chakraborty, P., Hess, J. R., Sharma, A., and Knickerbocker, S. (2017b). Outlier mining based traffic incident detection using big data analytics. In *Transportation Research Board 96th Annual Meeting Compendium of Papers*, pages 8–12.

Chakraborty, P., Sharma, A., and Hegde, C. (2018b). Freeway traffic incident detection from cameras: A semi-supervised learning approach. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1840–1845. IEEE.

Chambolle, A. (2004). An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1):89–97.

Chaudhry, R., Ravichandran, A., Hager, G., and Vidal, R. (2009). Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1932–1939. IEEE.

Chen, Y., Lv, Y., Wang, X., Li, L., and Wang, F.-Y. (2018). Detecting traffic information from social media texts with deep learning approaches. *IEEE Transactions on Intelligent Transportation Systems*, 0:1–10.

Chen, Z., Liu, X. C., and Zhang, G. (2016). Non-recurrent congestion analysis using data-driven spatiotemporal approach for information construction. *Transportation Research Part C: Emerging Technologies*, 71:19–31.

Cheu, R. L., Srinivasan, D., and Teh, E. T. (2003). Support vector machine models for freeway incident detection. In *2003 International Conference on Intelligent Transportation Systems (ITSC)*, volume 1, pages 238–243. IEEE.

Choi, K. and Chung, Y. (2010). A Data Fusion Algorithm for Estimating Link Travel Time. *Journal of Intelligent Transportation Systems*, 7(3-4):235–260.

Chollet, F. et al. (2015). Keras. https://github.com/fchollet/keras.

Chong, Y. S. and Tay, Y. H. (2017). Abnormal event detection in videos using spatiotemporal autoencoder. In *International Symposium on Neural Networks*, pages 189–196. Springer.

Chung, Y. (2011). Quantification of nonrecurrent congestion delay caused by freeway accidents and analysis of causal factors. *Transportation Research Record: Journal of the Transportation Research Board*, 2229:8–18.

Chung, Y. and Recker, W. W. (2012). A methodological approach for estimating temporal and spatial extent of delays caused by freeway accidents. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1454–1461.

Ciarlet, P. G. (1982). *Introduction à l'analyse numérique matricielle et à l'optimisation*. Dunod.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Coşar, S., Donatiello, G., Bogorny, V., Garate, C., Alvares, L. O., and Brémond, F. (2016). Toward abnormal trajectory and event detection in video surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(3):683–695.

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. (2017). Deformable convolutional networks. *CoRR, abs/1703.06211*, 1(2):3.

Darwish, T. and Abu Bakar, K. (2015). Traffic density estimation in vehicular ad hoc networks: A review. *Ad Hoc Networks*, 24(PA):337–351.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Dia, H. and Rose, G. (1997). Development and evaluation of neural network freeway incident detection models using field data. *Transportation Research Part C: Emerging Technologies*, 5(5):313–331.

Dia, H. and Thomas, K. (2011). Development and evaluation of arterial incident detection models using fusion of simulated probe vehicle and loop detector data. *Information Fusion*, 12(1):20–27.

Dong, J., Frosio, I., and Kautz, J. (2018). Learning adaptive parameter tuning for image processing. *Electronic Imaging*, 2018(13):1–8.

Dudek, C. L., Messer, C. J., and Nuckles, N. B. (1974). Incident detection on urban freeways. *Transportation Research Record: Journal of the Transportation Research Board*, 495:12–24.

Duran, J., Coll, B., and Sbert, C. (2013). Chambolle's projection algorithm for total variation denoising. *Image processing On Line*, 2013:311–331.

Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). Scalable object detection using deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2154.

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer.

Feng, Y., Hourdos, J., and Davis, G. A. (2014). Probe vehicle based real-time traffic monitoring on urban roadways. *Transportation Research Part C: Emerging Technologies*, 40:160–178.

Fortmann, T., Bar-Shalom, Y., and Scheffe, M. (1983). Sonar tracking of multiple targets using joint probabilistic data association. *IEEE journal of Oceanic Engineering*, 8(3):173–184.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014a). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580—-587.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014b). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 580–587.

Gonçalves, W. N., Machado, B. B., and Bruno, O. M. (2012). Spatiotemporal Gabor filters: a new method for dynamic texture recognition. *arXiv preprint arXiv*, pages 184–189.

Gupta, M., Gao, J., Aggarwal, C. C., and Han, J. (2014). Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267.

Haghani, A., Hamedi, M., and Sadabadi, K. F. (2009). I-95 corridor coalition vehicle probe project: Validation of INRIX data. Technical report, I-95 Corridor Coalition.

Hampel, F. R. (1974). The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393.

Han, J., Zhang, D., Cheng, G., Liu, N., and Xu, D. (2018). Advanced Deep-Learning Techniques for Salient and Category-Specific Object Detection: A Survey. *IEEE Signal Processing Magazine*, 35(1):84–100.

Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A. K., and Davis, L. S. (2016). Learning temporal regularity in video sequences. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 733–742. IEEE.

Hashemi, H. and Abdelghany, K. (2018). End-to-end deep learning methodology for real-time traffic network management. *Computer-Aided Civil and Infrastructure Engineering*, 33(10):849–863.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385v1*, 7(3):171–180.

Hellinga, B. and Knapp, G. (2000). Automatic vehicle identification technology-based freeway incident detection. *Transportation Research Record: Journal of the Transportation Research Board*, 1727:142–153.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (1993). *Convex analysis and minimization algorithms I, II*, volume 305-306. Springer Science & Business Media.

Houbraken, M., Audenaert, P., Colle, D., Pickavet, M., Scheerlinck, K., Yperman, I., and Logghe, S. (2015). Real-time traffic monitoring by fusing floating car data with stationary detector data. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 127–131. IEEE.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311.

Hui, Z., Yaohua, X., Lu, M., and Jiansheng, F. (2014). Vision-based real-time traffic accident detection. In *2014 11th World Congress on Intelligent Control and Automation (WCICA)*, pages 1035–1038. IEEE.

INRIX (2018). http://inrix.com/. accessed July 20, 2018.

Kamran, S. and Haas, O. (2007). A multilevel traffic incidents detection approach: Identifying traffic patterns and vehicle behaviours using real-time GPS data. In *2007 Intelligent Vehicles Symposium*, pages 912–917. IEEE.

Kapsch (2016). *Traffic management center annual report 2016*. Iowa Department of Transportation.

Kim, C., Li, F., Ciptadi, A., and Rehg, J. M. (2015). Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4696—-4704.

Kotzenmacher, B. J., Minge, E. D., and Hao, B. (2004). Evaluation of Portable Non-Intrusive Traffic Detection System. Technical report.

Leal-Taixé, L., Milan, A., Reid, I., Roth, S., and Schindler, K. (2015). MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942 [cs]*. arXiv: 1504.01942.

Lempitsky, V. and Zisserman, A. (2010). Learning To Count Objects in Images. *Advances in Neural Information Processing Systems*, pages 1324–1332.

Leys, C., Ley, C., Klein, O., Bernard, P., and Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766.

Li, T., Chang, H., Wang, M., Ni, B., Hong, R., and Yan, S. (2014). Crowded scene analysis: A survey. *IEEE transactions on circuits and systems for video technology*, 25(3):367–386.

Li, X., Lam, W. H., and Tam, M. L. (2013). New automatic incident detection algorithm based on traffic data collected for journey time estimation. *Journal of Transportation Engineering*, 139(8):840–847.

Li, Y. and McDonald, M. (2005). Motorway incident detection using probe vehicles. In *Proceedings of the Institution of Civil Engineers - Transport*, volume 158, pages 11–15.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *European Conference on Computer Vision ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

Liu, H., Gu, J., Meng, M. Q.-H., and Lu, W.-S. (2016a). Fast weighted total variation regularization algorithm for blur identification and image restoration. *IEEE Access*, 4:6792–6801.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016b). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Loog, M. (2016). Contrastive pessimistic likelihood estimation for semi-supervised classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):462–475.

Lou, J., Liu, Q., Tan, T., and Hu, W. (2002). Semantic interpretation of object activities in a surveillance system. In *Proceedings of 16th International Conference on Pattern Recognition*, volume 3, pages 777–780. IEEE.

Maaloul, B., Taleb-Ahmed, A., Niar, S., Harb, N., and Valderrama, C. (2017). Adaptive video-based algorithm for accident detection on highways. In *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 1–6. IEEE.

Naphade, M., Chang, M.-C., Sharma, A., Anastasiu, D. C., Jagarlamudi, V., Chakraborty, P., Huang, T., Wang, S., Liu, M.-Y., Chellappa, R., et al. (2018). The 2018 nvidia ai city challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 53–60.

Naphade, M., Tang, Z., Chang, M.-C., Anastasiu, D. C., Sharma, A., Chellappa, R., Wang, S., Chakraborty, P., Huang, T., Hwang, J.-N., et al. (2019). The 2019 ai city challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 452–460.

Oñoro-Rubio, D. and López-Sastre, R. J. (2016). Towards perspective-free object counting with deep learning. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9911 LNCS, pages 615–629.

Owens, N., Armstrong, A., Sullivan, P., Mitchell, C., Newton, D., Brewster, R., and Trego, T. (2010). Traffic incident management handbook. Technical report, U.S. Department of Transportation.

Ozkurt, C. and Camci, F. (2009). Automatic traffic density estimation and vehicle classification for traffic surveillance systems using neural networks. *Mathematical and Computational Applications*, 14(3):187–196.

Paris, S., Kornprobst, P., Tumblin, J., and Durand, F. (2007). A gentle introduction to bilateral filtering and its applications. In *ACM SIGGRAPH 2007 courses*. ACM.

Parkany, E. and Bernstein, D. (1995). Design of incident detection algorithms using vehicle-to-roadside communication sensors. *Transportation Research Record: Journal of the Transportation Research Board*, 1494:67–74.

Parkany, E. and Xie, C. (2005). A complete review of incident detection algorithms & their deployment: what works and what doesn't. Technical Report NETCR 37, The New England Transportation Consortium.

Pearson, R. K. (2005). *Mining imperfect data: Dealing with contamination and incomplete records*. SIAM.

Perš, J., Sulić, V., Kristan, M., Perše, M., Polanec, K., and Kovačič, S. (2010). Histograms of optical flow for efficient representation of body motion. *Pattern Recognition Letters*, 31(11):1369–1376.

Persaud, B. N. and Hall, F. L. (1989). Catastrophe theory and patterns in 30-second freeway traffic data- Implications for incident detection. *Transportation Research Part A: General*, 23(2):103–113.

Piciarelli, C., Micheloni, C., and Foresti, G. L. (2008). Trajectory-based anomalous event detection. *IEEE Transactions on Circuits and Systems for video Technology*, 18(11):1544–1554.

Quiroga, C., Hamad, K., and Park, E. S. (2005). Incident detection optimization and data quality control. Technical report, Texas Transportation Institute, The Texas A&M University System.

Ravinder, K., Velmurugan, S., and Gangopadhyay, S. (2008). Efficacy of video incident detection system for advanced traffic management under non-adherence of lane discipline. In *Intelligent Transportation Systems Connections: Saving Time. Saving Lives.*

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 779–788.

Redmon, J. and Farhadi, A. (2016). YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242.

Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *ArXiv e-prints 1804.02767*.

Reid, D. (1979). An algorithm for tracking multiple targets. *IEEE transactions on Automatic Control*, 24(6):843–854.

Ren, J., Chen, Y., Xin, L., Shi, J., Li, B., and Liu, Y. (2016). Detecting and positioning of traffic incidents via video-based analysis of traffic states in a road segment. *IET Intelligent Transport Systems*, 10(6):428–437.

Ren, J. S., Wang, W., Wang, J., and Liao, S. (2012). An unsupervised feature learning approach to improve automatic incident detection. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 172–177. IEEE.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99.

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.

Rezatofighi, S. H., Milan, A., Zhang, Z., Shi, Q., Dick, A., and Reid, I. (2015). Joint probabilistic data association revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3047—-3055.

Rodríguez, P. (2013). Total variation regularization algorithms for images corrupted with different noise models: A review. *Journal of Electrical and Computer Engineering*, 2013:1–18.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Sadeky, S., Al-Hamadiy, A., Michaelisy, B., and Sayed, U. (2010). Real-time automatic traffic accident recognition using hfg. In *2010 20th International Conference on Pattern Recognition (ICPR)*, pages 3348–3351. IEEE.

Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Schrank, D., Eisele, B., Lomax, T., and Bak, J. (2015). 2015 urban mobility scorecard. Technical report, Texas A&M Transportation Institute and INRIX.

Schrank, D. L. and Lomax, T. J. (2007). The 2007 urban mobility report. Technical report, Texas Transportation Institute, The Texas A&M University.

Sethi, V., Bhandari, N., Koppelman, F. S., and Schofer, J. L. (1995). Arterial incident detection using fixed detector and probe vehicle data. *Transportation Research Part C: Emerging Technologies*, 3(2):99–112.

Sharma, A., Ahsani, V., and Rawat, S. (2017). Evaluation of opportunities and challenges of using inrix data for real-time performance monitoring and historical trend assessment. Technical Report SPR-P1(14) M007, Nebraska Department of Roads.

Shi, Q. and Abdel-Aty, M. (2015). Big data applications in real-time traffic operation and safety monitoring and improvement on urban expressways. *Transportation Research Part C: Emerging Technologies*, 58:380–394.

Singh, D. and Mohan, C. K. (2018). Deep spatio-temporal representation for detection of road accidents using stacked autoencoder. *IEEE Transactions on Intelligent Transportation Systems*, 0:1–9.

Snelder, M., Bakri, T., and van Arem, B. (2013). Delays caused by incidents: Data-driven approach. *Transportation Research Record: Journal of the Transportation Research Board*, 2333:1–8.

Systematics, C. (2005). Traffic congestion and reliability: Trends and advanced strategies for congestion mitigation. Technical report, Federal Highway Administration, Texas Transportation Institute.

Szegedy, C., Toshev, A., and Erhan, D. (2013). Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561.

Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision*, pages 839–846. IEEE.

Van Lint, J. W. C. and Hoogendoorn, S. P. (2010). A Robust and Efficient Method for Fusing Heterogeneous Data from Traffic Sensors on Freeways. *Computer-Aided Civil and Infrastructure Engineering*, 25(8):596–612.

Verri, A. and Poggio, T. (1989). Motion field and optical flow: Qualitative properties. *IEEE Transactions on pattern analysis and machine intelligence*, 11(5):490–498.

Vondrick, C., Patterson, D., and Ramanan, D. (2013). Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1):184–204.

Wang, C., Xue, B., and Shang, L. (2017). Pso-based parameters selection for the bilateral filter in image denoising. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 51–58. ACM.

Weegberg, K., Luk, J., Han, C., and Chin, D. (2010). Freeway incident detection – technologies and techniques. Technical Report AP–R364/10, Austroads Ltd.

Wells, T. and Toffin, E. (2005). Video-based automatic incident detection on San-Mateo bridge in the San Francisco bay area. In *12th World Congress on Intelligent Transportation Systems, San Francisco*. Citeseer.

Williams, B. M. and Guin, A. (2007). Traffic management center use of incident detection algorithms: Findings of a nationwide survey. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):351–358.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple Online and Realtime Tracking with a Deep Association Metric. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*.

Xu, C., Liu, P., Yang, B., and Wang, W. (2016). Real-time estimation of secondary crash likelihood on freeways using high-resolution loop detector data. *Transportation Research Part C: Emerging Technologies*, 71:406–418.

Yang, B. and Nevatia, R. (2012). Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1918–1925. IEEE.

Yang, K., Wang, X., and Yu, R. (2018). A bayesian dynamic updating approach for urban expressway real-time crash risk evaluation. *Transportation Research Part C: Emerging Technologies*, 96:192–207.

Yuan, Y., Wang, D., and Wang, Q. (2017). Anomaly detection in traffic scenes via spatial-aware motion reconstruction. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1198–1209.

Yuan, Z., Zhou, X., and Yang, T. (2018). Hetero-ConvLSTM: A Deep Learning Approach to Traffic Accident Prediction on Heterogeneous Spatio-Temporal Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 984–992. ACM.

Yue, M., Fan, L., and Shahabi, C. (2016). Inferring traffic incident start time with loop sensor data. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2481–2484. ACM.

Zhang, M. and Gunturk, B. K. (2008). Multiresolution bilateral filtering for image denoising. *IEEE Transactions on Image Processing*, 17(12):2324–2333.

Zhang, S., Wu, G., Costeira, J. P., and Moura, J. M. (2017a). Understanding traffic density from large-scale web camera data. *arXiv preprint arXiv:1703.05868*.

Zhang, Y., Ren, J., Liu, J., Xu, C., Guo, H., and Liu, Y. (2017b). A survey on emerging computing paradigms for big data. *Chinese Journal of Electronics*, 26(1):1–12.

Zhang, Z., He, Q., Gao, J., and Ni, M. (2018). A deep learning approach for detecting traffic accidents from social media data. *Transportation Research Part C: Emerging Technologies*, 86:580–596.

Zhang, Z., He, Q., Tong, H., Gou, J., and Li, X. (2016). Spatial-temporal traffic flow pattern identification and anomaly detection with dictionary-based compression theory in a large-scale urban network. *Transportation Research Part C: Emerging Technologies*, 71:284–302.

Zhao, B., Fei-Fei, L., and Xing, E. P. (2011). Online detection of unusual events in videos via dynamic sparse coding. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3313–3320. IEEE.

Zhong, M. and Liu, G. (2007). Establishing and Managing Jurisdiction-wide Traffic Monitoring Systems: North American Experiences. *Journal of Transportation Systems Engineering and Information Technology*, 7(6):25–38.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328.

Zhu, L., Guo, F., Krishnan, R., and Polak, J. W. (2018). A deep learning approach for traffic incident detection in urban networks. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1011–1016. IEEE.

Zhu, T., Wang, J., and Lv, W. (2009). Outlier mining based automatic incident detection on urban arterial road. In *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*, Mobility '09, pages 29:1–29:6.

Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical report, School Comput. Science, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CALD-02-107.

Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.