2020

Designing the next generation intelligent transportation sensor system using big data driven machine learning techniques

Tongge Huang
Iowa State University

Designing the next generation intelligent transportation sensor system using big data driven machine learning techniques

by

Tongge Huang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Civil Engineering (Intelligent Infrastructure Engineering)

Program of Study Committee:
Anuj Sharma, Major Professor
Chinmay Hegde
Christopher Day
Jing Dong
Soumik Sarkar

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2020

DEDICATION

I would like to dedicate this dissertation to my family without whose love and support I would not have been able to complete this work. A special feeling of appreciation to my parents Min Guo and Weixing Huang for their encouragement and guidance throughout my entire life. And to my girlfriend Chang Liu for her love and understanding.

I would also like to thank my friends and colleagues who have helped me during my research work at the Reactor Lab.

TABLE OF CONTENTS

Page

LIST OF TABLES v

LIST OF FIGURES vi

ACKNOWLEDGMENTS viii

ABSTRACT ix

CHAPTER 1. INTRODUCTION 1
1.1 Problem Statement 1
1.2 Research Objectives 3
1.3 Research Methodology 4
1.4 Organization of Dissertation 6

CHAPTER 2. REVIEW OF LITERATURE 7
2.1 Introduction 7
2.2 Large-Scale Data-driven Traffic Sensor Health Monitoring 7
2.3 Deep convolutional generative adversarial networks for traffic data imputation encoding time series as images 9
2.4 Technical and economic feasibility assessment of cloud-enabled traffic video analysis framework 12

CHAPTER 3. LARGE-SCALE DATA-DRIVEN TRAFFIC SENSOR HEALTH MONITORING 15
3.1 Introduction 15
3.2 Methodology 17
3.2.1 Setup 18
3.2.2 Data Completeness Test 19
3.2.3 AEVL Anomaly Test 20
3.2.4 Temporal Pattern Anomaly Test 22
3.2.5 Baseline Comparison 26
3.3 Results 27
3.3.1 Data Completeness Test Results 28
3.3.2 AEVL Anomaly Test Results 28
3.3.3 Temporal Pattern Anomaly Test Results 32
3.3.4 Baseline Comparison 34
3.4 Conclusion 40

CHAPTER 4. DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS FOR TRAFFIC DATA IMPUTATION ENCODING TIME SERIES AS IMAGES 43
4.1 Introduction 43
4.2 Methodology 46
4.2.1 Notation 46
4.2.2 Gramian Angular Summation Field 47
4.2.3 Generative Adversarial Networks 49
4.2.4 TSDIGAN Architecture 50
4.2.5 TSDIGAN Imputation Model 52
4.2.6 Large-Scale Implementation 56
4.3 Results 58
4.3.1 Data Description 58
4.3.2 Evaluation Criteria 58
4.3.3 Single VDS Performance 59
4.3.4 Imputation Performance on a Large-Scale Network 64
4.3.5 Model Comparison 69
4.4 Conclusion 70

CHAPTER 5. TECHNICAL AND ECONOMIC FEASIBILITY ASSESSMENT OF CLOUD-ENABLED TRAFFIC VIDEO ANALYSIS FRAMEWORK 73
5.1 Introduction 73
5.2 Methodology 76
5.2.1 Perception module 76
5.2.2 Analysis module 78
5.2.3 Cloud-enabled Implementation 79
5.3 Results 81
5.3.1 Economic assessment 81
5.3.2 Feasibility assessment 83
5.4 Conclusions 90

CHAPTER 6. CONCLUSIONS 92
6.1 Large-Scale Data-driven Traffic Sensor Health Monitoring 92
6.2 Deep convolutional generative adversarial networks for traffic data imputation encoding time series as images 93
6.3 Technical and economic feasibility assessment of cloud-enabled traffic video analysis framework 94
6.4 Limitations and Future Work 94

BIBLIOGRAPHY 96

LIST OF TABLES

		Page
Table 4.1	Performance summary for VDSs of 5 sample clusters at 20%, 50%, and 80% MR	68
Table 5.1	Summary of computation resources usage on GCP	82
Table 5.2	Experimental results for different cameras under different environment	85
Table 5.3	Absolute error (AE) of daily volume between camera and radar sensors	88
Table 5.4	Relative error (RE) of daily volume between camera and radar sensors	89

LIST OF FIGURES

		Page
Figure 3.1	Workflow of proposed TSHM module	18
Figure 3.2	Data completeness test result on the sample month of July	29
Figure 3.3	Sample missing data percentage heatmap of (a) Normal sensor, (b) Abnormal Sensor - Level 1, and (c) Abnormal Sensor - Level 2 from the data completeness test	29
Figure 3.4	AEVL anomaly test result for the sample month of July, 2017	30
Figure 3.5	Completeness Score plots for all sensors in step 1 and abnormal sensors in step 2	31
Figure 3.6	Sample CDF plots of abnormal sensor with their upstream(u/s) and downstream(d/s) sensors for abnormal sensor with: (a) low mean, (b) high mean, and (c) high variance	32
Figure 3.7	$AEVL$ time series distribution of a sample day for (a) Upstream sensor, (b) Middle anomaly sensor, (c) Downstream sensor and changepoint probability distribution for the same day for the (d) Upstream sensor, (e) Middle anomaly sensor, (f) Downstream sensor	33
Figure 3.8	Sample RPCA Decomposition: (a) Raw CP matrix, (b) Low rank CP matrix, (c) Sparse CP matrix	34
Figure 3.9	RPCA sparse temporal matrix clustering	35
Figure 3.10	RPCA Sparse Matrix Clustering heatmap	35
Figure 3.11	AEVL plots for all sensors in step 2 and abnormal sensors in step 3	36
Figure 3.12	Clustering comparison with CLM method	37
Figure 3.13	Sample monthly $AEVL$ heatmap of July, 2017 for sample (a) normal sensor and (b-d) abnormal sensors	38
Figure 3.14	MCB algorithm based error percentages of sensors with labels from the proposed method	39
Figure 3.15	$AEVL$ heatmap of (a) A5, (b) B3, (c) C2, and (d) D3	39
Figure 4.1	Architecture of GANs.	49
Figure 4.2	Proposed TSDIGAN (a) Discriminator and (b) Generator.	50
Figure 4.3	MMD Score Trace	52
Figure 4.4	Architecture of TSDIGAN Imputation Model.	53
Figure 4.5	Sample results of GASF encoded images training using the proposed GAN generator model.	60
Figure 4.6	Imputation performance for a single sample sensors for different missing rates.	62
Figure 4.7	Error distributions for 10% and 90% marginal missing rates: (a) absolute error and (b) relative error.	62
Figure 4.9	Sample synthetic traffic flow data plot: (a) Weekday and (b) Non-weekday.	63
Figure 4.10	(a) Traffic flow histograms and (b) deviation distribution between the real test data and synthetic data for 20% MR	64
Figure 4.11	Elbow plot to determine optimal number of clusters for VDSs	65

Figure 4.13 Weekday . . . 66
Figure 4.14 Nonweekday . . . 66
Figure 4.15 Median traffic flow patterns for (a) weekdays and (b) non-weekdays for 5 sample VDS clusters . . . 66
Figure 4.16 Imputation performance in terms of (a) MAE, (b) RMSE, and (c) MRE for all VDS in Cluster ID 2 . . . 67
Figure 4.17 Imputation performance accuracies (MAE, $RMSE$, and MRE) at 30% MR for all VDSs . . . 68
Figure 4.18 Comparison of imputation performance accuracies in terms of (a) MAE, (b) $RMSE$, and (c) MRE with respect to other benchmark imputation models 71
Figure 5.1 Perception module architecture . . . 77
Figure 5.2 Perception module architecture . . . 78
Figure 5.3 Overall Intelligent video analysis framework . . . 80
Figure 5.4 Main components latency of perception module with simultaneously running different number of cameras per GPU . . . 82
Figure 5.5 Daily operating cost for 160 cameras (a) with sustained use discount, and (b) without sustained use discount . . . 83
Figure 5.6 Sampled cameras with different viewing angles . . . 84
Figure 5.7 Camera views and trajectories plots under sunny and rainy day (a) Camera 5 sunny day, (b) Camera 5 rainy day, (a) Camera 9 sunny day, (b) Camera 9 rainy day . . . 86
Figure 5.8 Daily traffic volume data (10-mins agg) captured by cameras and nearby radar sensors on Monday at June 1st . . . 87
Figure 5.9 Anomaly events with pixel moving speed . . . 90

ACKNOWLEDGMENTS

I would like to take this opportunity to thank Dr. Anuj Sharma for his continuous support, patience, and motivation all these years. Without his inspiration, I would not have been able to complete this work. I would also like to thank my committee members Dr. Chinmay Hegde, Dr. Christopher Day, Dr. Jing Dong, and Dr. Soumik Sarkar for their suggestions throughout my study.

In addition, I would like to express thanks to my friends Dr. Pranamesh Chakraborty, Dr. Tingting Huang, Dr. Shuo Wang, and Subhadipto Poddar for their helps throughout my research work at the Reactor Lab.

I will cherish the moments that I have spent with all kindly faculties and friends at Iowa State University. Surely, not only the technical skills but also the active attitude I have learned here will help me to face any challenges in my future life.

ABSTRACT

Accurate traffic data collection is essential for supporting advanced traffic management system operations. This study investigated a large-scale data-driven sequential traffic sensor health monitoring (TSHM) module that can be used to monitor sensor health conditions over large traffic networks. Our proposed module consists of three sequential steps for detecting different types of abnormal sensor issues. The first step detects sensors with abnormally high missing data rates, while the second step uses clustering anomaly detection to detect sensors reporting abnormal records. The final step introduces a novel Bayesian changepoint modeling technique to detect sensors reporting abnormal traffic data fluctuations by assuming a constant vehicle length distribution based on average effective vehicle length ($AEVL$). Our proposed method is then compared with two benchmark algorithms to show its efficacy. Results obtained by applying our method to the statewide traffic sensor data of Iowa show it can successfully detect different classes of sensor issues. This demonstrates that sequential TSHM modules can help transportation agencies determine traffic sensors' exact problems, thereby enabling them to take the required corrective steps.

The second research objective will focus on the traffic data imputation after we discard the anomaly/missing data collected from failure traffic sensors. Sufficient high-quality traffic data are a crucial component of various Intelligent Transportation System (ITS) applications and research related to congestion prediction, speed prediction, incident detection, and other traffic operation tasks. Nonetheless, missing traffic data are a common issue in sensor data which is inevitable due to several reasons, such as malfunctioning, poor maintenance or calibration, and intermittent communications. Such missing data issues often make data analysis and decision-making complicated and challenging. In this study, we have developed a generative adversarial network (GAN) based traffic sensor data imputation framework (TSDIGAN) to efficiently reconstruct the missing data by generating realistic synthetic data. In recent years, GANs have shown impressive success in image

data generation. However, generating traffic data by taking advantage of GAN based modeling is a challenging task, since traffic data have strong time dependency. To address this problem, we propose a novel time-dependent encoding method called the Gramian Angular Summation Field (GASF) that converts the problem of traffic time-series data generation into that of image generation. We have evaluated and tested our proposed model using the benchmark dataset provided by Caltrans Performance Management Systems (PeMS). This study shows that the proposed model can significantly improve the traffic data imputation accuracy in terms of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) compared to state-of-the-art models on the benchmark dataset. Further, the model achieves reasonably high accuracy in imputation tasks even under a very high missing data rate ($> 50\%$), which shows the robustness and efficiency of the proposed model.

Besides the loop and radar sensors, traffic cameras have shown great ability to provide insightful traffic information using the image and video processing techniques. Therefore, the third and final part of this work aimed to introduce an end to end real-time cloud-enabled traffic video analysis (IVA) framework to support the development of the future smart city. As Artificial intelligence (AI) growing rapidly, Computer vision (CV) techniques are expected to significantly improve the development of intelligent transportation systems (ITS), which are anticipated to be a key component of future Smart City (SC) frameworks. Powered by computer vision techniques, the converting of existing traffic cameras into connected "smart sensors" called intelligent video analysis (IVA) systems has shown the great capability of producing insightful data to support ITS applications. However, developing such IVA systems for large-scale, real-time application deserves further study, as the current research efforts are focused more on model effectiveness instead of model efficiency. Therefore, we have introduced a real-time, large-scale, cloud-enabled traffic video analysis framework using NVIDIA DeepStream, which is a streaming analysis toolkit for AI-based video and image analysis. In this study, we have evaluated the technical and economic feasibility of our proposed framework to help traffic agency to build IVA systems more efficiently. Our study shows that the daily operating cost for our proposed framework on Google Cloud Platform (GCP) is less

than $0.14 per camera, and that, compared with manual inspections, our framework achieves an average vehicle-counting accuracy of 83.7% on sunny days.

CHAPTER 1. INTRODUCTION

1.1 Problem Statement

Intelligent transportation system (ITS) has been recognized as an efficient solution to make the transport network safer and smarter. As a crucial component of future Smart City (SC) framework, the ITS integrates a wide range of services, including traffic sensing, information communication, traffic management, and optimization. US Department of Transportation (USDOT) has expanded the resources pool by more than $500 million to support research-based initiatives related to ITS and SC (Gandy Jr et al., 2020). And, the ITS market is likely to grow to approximately $130 billion by 2025 (Mishra et al., 2019). As the most fundamental part of the ITS, traffic sensors including the inductive loop detectors and microwave radar sensors are widely installed on the road network to collect traffic data, such as volume, speed, and occupancy. However, these traffic sensors are often suffering from erroneous and missing data due to various reasons like communication loss, poor maintenance or calibration, and sensor malfunctioning (Lee and Coifman, 2011). According to California Performance Measurement System (PeMS), there are only 67% of the sensors in District 7 of southern California (Los Angeles) were found to be working as expected in December 2018 (Armanious, 2019). Yet, besides the traditional loop and radar sensors, there exists already a massive number of traffic surveillance cameras widely installed on the road network across the US to ensure traffic safety As mentioned by Ananthanarayanan et al. (2019), there was one camera per 8 people in US and UK. With the development of image/video processing techniques, traffic surveillance cameras have shown the great potential capability of providing insightful data directly from video frames. Therefore, an intelligent sensor system aims to address the failure of traffic sensors, interpolate the erroneous or missing data, and integrate the camera sensors is significantly needed to enable the future advanced ITS applications.

Detection of failure sensors which are reporting erroneous or missing data is one of the key steps to support the advanced ITS applications like traffic congestion detection, traffic speed prediction and decision-making (Chakraborty et al., 2018a; Ma et al., 2015; Mori et al., 2015). Typically, traffic sensor health monitoring (TSHM) is done by comparing the data readings with the predefined thresholds or neighboring sensors based on daily aggregate traffic data (Chen et al., 2003; Lu et al., 2014). However, such high-level aggregation fails to identify the within-day abnormalities or the erroneous from isolated sensors. And, monitoring sensor health state-wised using high-resolution data (say 20 seconds interval) requires the processing of large-scale traffic data which is beyond the traditional computation techniques. Thus, there is a need to develop the stepwise method to handle large-scale statewide traffic data sources to detect abnormal sensors based on overall and temporal abnormalities.

After addressing the failure sensors, the data from those sensors will be discarded from further usage. However, many traffic-related studies like traffic flow analysis require the data to be complete. An alternative approach is to impute the erroneous or missing data, so that the data from the failure sensors can still be used for subsequent analysis. Previously, traffic data imputation has been done using historical data (Nihan, 1997; Allison, 2001). However, these methods failed to capture the spatio-temporal correlations which lead to unreliable performance. Statistic modeling is another type of imputation method which requires the assumption of the probability distribution. And, statistic modeling is not always performing well to handle the large proportion of erroneous or missing data. In recent years, the traffic data imputation task has been treated as the corrupted data denoising problem (Duan et al., 2016; Asadi and Regan, 2019). However, modeling the strong time dependency of the time-series data is still a challenge. On the other hand, recurrent neural network (RNN) based methods like Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) were used to capture the time dependency features. However, training such RNN networks generally take a significantly longer time when handling the long time sequence (Li et al., 2019). Thus, building an easy-to-train, accurate, and robust traffic data imputation framework is needed for large-scale ITS sensor system.

To ensure the traffic safety, Department of Transportation (DOT) has installed the traffic surveillance cameras widely across the state. With the recent advancements in computer vision (CV) and Internet of Things (IoT) techniques, traffic surveillance cameras could be used as the connected "smart sensors" to make the transportation system more informative. Unlike the traditional loop sensors or radar sensors, camera sensors equipped computer vision techniques are able to recognize the traffic patterns more precisely by capturing the traffic data directly from video streams, also called intelligent video analysis (IVA). However, current research efforts on deep learning-based methods are focusing more on the model effectiveness rather than efficiency (Liu et al., 2018). For instance, Dai et al. (2019) proposed a video-based vehicle counting framework using the YOLOv3 model which achieved 87.6% counting accuracy, while the real-time rate (framework's run-time divided by the duration of test videos) is 1.3. Although counting accuracy is comparable, such offline approaches, with their high processing latency, is unfeasible for large-scale real-time ITS applications. Recently, pushing the CV algorithms close to the data sources or devices (edge computing) seems to be a possible solution to support large-scale real-time ITS applications using a massive number of traffic cameras. To address this problem, NVIDIA developed an accelerated video processing toolkit named Deepstream which enabled edge computing techniques for multi-GPU and multi-stream video analysis. In addition, cloud computing is another key concept of the future Smart City framework, which allows the efficiency of data sharing and deployment. Therefore, the cost and feasibility of developing such end to end, real-time, cloud-enabled traffic video analysis system are required to be estimated and evaluated for the real-world practical implementations.

1.2 Research Objectives

This study aims at developing an intelligent traffic sensor system to support the next generation ITS applications using big data and machine learning techniques. This study is divided into three broad research objectives mainly focus on sensor health monitoring, erroneous or missing data imputation, and camera sensor integration.

1. Data-driven large-scale stepwise method of anomalous sensor identification based on time series analysis and clustering analysis

2. Deep convolutional generative adversarial networks (DCGAN) based erroneous or missing traffic data imputation framework by encoding time series as images

3. Feasibility evaluation from aspects of economic and technical for an intelligent cloud-enabled traffic video analysis framework

The methodologies adopted for each of above research objectives are briefly discussed in the next section.

1.3 Research Methodology

1. Large-Scale Data-driven Traffic Sensor Health Monitoring

The first part of this study is to develop a data-driven, massively parallelizable traffic sensor health monitoring framework which can identify the failure sensors periodically at the state level to help the traffic agency maintain the devices. Such a stepwise framework can also help the subsequent analysis avoid using the erroneous data. This traffic sensor health monitoring framework consists of three steps:

(a) *Data Completeness Test:* Checks whether a traffic sensor has missing data issue by checking data readings for completeness.

(b) *AEVL Anomaly Test:* Identifies suspicious records by checking whether sensors have provided unreasonable volume, occupancy, or speed records.

(c) *Temporal Pattern Anomaly Test:* Checks whether sensors have provided fluctuating *AEVL* values over the daily time-series data.

We show that how we identify the failure sensors by clustering analysis and time series analysis with details in Chapter 3 .

2. Deep convolutional generative adversarial networks for traffic data imputation encoding time series as images

The second part of this study aims at developing an accurate and practical traffic data imputation framework to efficiently reconstruct the erroneous or missing data by generating realistic synthetic data. By taking the benefits of the generative model, our proposed framework in this study treats the data imputation problem as the data generation problem. We first adopted the novel GASF encoding method to embed the strong temporal dependency of the time-series data, thereby converting the time series generation problem into the image generation problem. Another benefit of such encoding methods is the precisely inverse operations, by that the synthetic image can be converted back to time series data easily. The deep learning-based generative adversarial network showed the remarkable ability in generating the high-quality image data by modeling the real data distributions. Thus, we then trained the deep convolutional generative adversarial networks (DCGAN) to generate realistic synthetic data for traffic data imputation. We show that how we encoding the time-series data and train the DCGAN with details in Chapter 4.

3. Technical and economic feasibility assessment of cloud-enabled traffic video analysis framework

The third part of this study introduced and evaluated an end to end, real-time, cloud-enabled IVA framework to analyze large-scale live video streams. And, we evaluate the economic and technical feasibility of such a practical framework to help the traffic agency build their own system to support future ITS and Smart City applications. The proposed framework consists of two modules, namely, perception module and analysis module. In the perception module, we used the Nvidia Deepstream which is an accelerated AI-powered framework for video processing and analysis (Nvidia Deepstream, 2020). The perception module mainly performs three tasks: (1) Decoding multiple live video streams, (2) analyzing the decoded video frames using deep learning-based detection and tracking model, and (3) formatting the resulted information of detected vehicles and

sending them to IoT based message exchange interface. In the analysis module, we used the big data processing platforms including Apache Kafka (Kafka, 2019), Apache Spark (Spark, 2020) and Elasticsearch combined with Kibana (Elastic, 2020) for data transmission, processing, and visualization, respectively. We deployed the proposed framework on the Google cloud platform (GCP) to estimate the detailed operating costs. And we evaluate the technical feasibility of the proposed framework by measuring the vehicle-counting accuracy. We show the framework architecture and feasibility assessment with details in Chapter 5.

1.4 Organization of Dissertation

This dissertation consists of six chapters. Chapter 1 gives a brief introduction of the background and problem statement, the three research objectives, and their methodologies involved in this study. Chapter 2 presents a comprehensive literature review of the three research objectives. Chapter 3 presents a detailed description of the first research objective which aims at addressing the fault traffic sensors from large-scale sensor networks. Chapter 4 describes the details of the second research objective, developing an accurate and robust traffic data imputation framework to reconstruct the erroneous or missing traffic data. And, Chapter 5 describes the third research objective which introduced an end to end, real-time, cloud-enabled traffic video analysis framework with the feasibility assessment. Finally, Chapter 6 concludes this study by summarizing the findings, limitations, and future work.

CHAPTER 2. REVIEW OF LITERATURE

2.1 Introduction

This chapter presents a broad literature review, focusing on the detection of failure traffic sensors, imputation of erroneous or missing data, and framework of intelligent traffic video analysis. In general. the methods used for the detection of failure traffic sensors can be divided into three categories: threshold-based method, comparison-based method and deep learning-based method. For erroneous or missing data imputation, the current methods can be summarized as prediction methods, interpolation methods, statistical modeling methods, and deep learning-based methods. To support the future Smart City applications, the traffic video analysis framework generally has three important components: video processing pipeline, efficient computer vision model, and cloud-enabled implementation. This chapter reviews the literature and backgrounds of each research objectives with categories.

2.2 Large-Scale Data-driven Traffic Sensor Health Monitoring

Over the last several decades, a number of studies have investigated this area. Traditionally, anomalous sensors have been identified by comparing individual traffic parameters to predetermined thresholds. For example, Payne and Thompson (1997) used predetermined thresholds of volume, occupancy, and speed, comparing these with 30-second and 5-minute aggregated traffic data to detect abnormal sensors. However, the unlikely assumption that traffic parameters are independent of one another is a primary concern regarding any single-parameter threshold algorithm. Therefore, Jacobson et al. (1990) used the relationship between traffic volume (q) and density (k), introducing the volume-occupancy ratio to check for anomalous data. If observed data fell outside accepted $k - q$ boundaries, it was flagged as erroneous data. However, the $k - q$ ratio algorithm labeled the data as erroneous when both k and q were equal to zero although such situations can also

arise when no vehicle passed by the sensor (Chen et al., 2003). Chen et al. (2003) have therefore suggested that the $k - q$ ratio region is sensitive to the threshold settings and developed a daily statistics algorithm (DSA) using density and volume time series data to generate four statistic factors to identify the anomalous loop detectors. Vanajakshi and Rilett (2004) have used the idea of conservation of vehicles for a series of sequential detectors, examining detector anomalies at the network instead of single-sensor level.

Most of the sensor anomaly detection algorithms listed above are mainly based on occupancy and volume, which can easily be obtained using single loop detectors. However, individual vehicle speed calculation requires paired loop detectors or advanced roadway sensors such as microwave radar sensors, and etc. Due to their low installation cost, high accuracy, and small size, such advanced sensors are now widely used for traffic data collection (Klein et al., 2006). Hence, algorithms have also been developed for faulty sensor detection using all three traffic parameters: speed, volume, and occupancy. Turochy and Smith (2000) have proposed average effective vehicle length ($AEVL$) as an approximate function of volume, occupancy, and speed. Their study showed that $AEVL$ can successfully capture a wide range of data anomaly types which single-parameter threshold-based methods cannot. Similar other studies using $AEVL$ for anomalous sensor identification have also shown its efficacy (Al-Deek et al., 2004; Wells et al., 2008).

Although $AEVL$ has proven a useful indicator for sensor health monitoring, most studies still use predetermined thresholds. However, different detectors such as dual loop detectors, laser sensors, microwave radar sensors, etc. record data in different ways, not all of which are well adapted for sensor analysis based on predetermined thresholds. For example, for sensors that can self-recover in the case of a temporal anomaly, threshold-based algorithms might be too sensitive. To eliminate these pre-defined $AEVL$ thresholds, Lu et al. (2014) have developed a temporal and spatial based sequential algorithm for sensor health screening. Their study proposed a Multiple-Comparisons-with-the-Best (MCB) model to compare $AEVL$ between adjacent lanes and stations to assess any target detector's data quality. However, such between-station comparisons might not work well if nearby stations also have data quality issues. Also, the sequential MCB algorithm cannot be applied

to isolated sensors or sensors with only one lane in each direction since it requires both within-station and between-station comparisons. To overcome these limitations, this study's proposed TSHM module introduces a data-driven approach to identifying faulty sensors based on clustering analysis for large-scale statewide sensor data.

Currently, most sensor-screening algorithms use daily aggregated traffic data to determine abnormal sensors, thereby ignoring temporal anomalies in the data stream. For example, Wu et al. (2017) have recently introduced a spatiotemporal pattern network (STPN) algorithm that can capture the time-series features of volume and speed data by a symbolization process. Their D-Markov model, trained on systematically ordered stations, calculates a mutual information matrix to identify anomalous sensors having low mutual information values. However, the well-ordered systematic sensor information required for training their STPN model is difficult to obtain on a statewide scale. Additionally, if any traffic sensor is added or removed, retraining the model becomes difficult. Finally, considering the large scale of traffic data collected from sensors at the statewide level, it is well beyond the capabilities of traditional computation techniques to train a complex STPN model for use at this level. Therefore, in this study, we propose a massively parallelizable TSHM approach that can extract temporal anomalies from large-scale traffic data streams and identify anomalous sensors showing frequent temporal abnormalities.

2.3 Deep convolutional generative adversarial networks for traffic data imputation encoding time series as images

Undoubtedly, many of ITS applications require high-quality and complete traffic data, a significant amount of studies have been done in the past on missing traffic data imputation. As summarized by Li et al. (2014), traffic data imputation methods can be broadly divided into three categories: prediction methods, interpolation methods, and statistical learning methods. Some recent studies have also applied deep learning-based methods like stacked denoising autoencoders (DSAE) to estimate missing traffic data.

Prediction based models such as the Autoregressive Integrated Moving Average (ARIMA) model (Nihan, 1997; Park et al., 1998), support vector regression (SVR) (Castro-Neto et al., 2009), and Bayesian networks (BNs) (Ghosh et al., 2007) use the historical observations to predict the future data points. These prediction-based models assume that the future data points follow the typical trace as the historical data. Prediction based methods can usually estimate data points effectively over the short-term and for samples whose missing data ratio is low. However, their performance drops significantly for long-term imputation problems. Further, these methods only use observations from the history before the missing data points, while any valuable information available after the missing data points is not utilized.

Interpolation based models, another popular method for missing data imputation, include temporal neighboring model or historical model (Yin et al., 2012; Smith et al., 2003; Allison, 2001) and k-NN model (Al-Deek et al., 2004). The basic idea of the historical model is to impute missing data points using the average historical value reported by the same sensor for the same time periods. Therefore, historical models assume that the daily traffic flow has strong consistency. However, they do not use the information of the inherent daily variation to improve the imputation performance. In contrast, the k-NN method utilizes the average historical value for a given day of the week from the same sensor or neighboring sensors to impute missing data, instead of considering the overall average. However, these interpolation-based models often fail to explicitly capture the spatial-temporal variations, which can lead to unreliable imputation results.

Statistical based models such as probabilistic principal component analysis (PPCA) (Qu et al., 2009) and Markov Chain Monte Carlo (MCMC) methods (Ni and Leonard, 2005) have been proposed to overcome the limitations mentioned above and improve imputation accuracy based on statistical modeling. These statistical methods assume a probability distribution model for the traffic data and impute the missing data using the observed data with optimized parameters. However, these models do not work well when the proportion of missing data are significantly high, especially for the case when an entire day of traffic data is missing (Anandkumar et al., 2014; Tan et al., 2013), a situation fairly common in real-world scenarios.

In recent years, deep learning based models using denoising stacked autoencoders (DSAE) have been studied to overcome the shortcomings of the traditional data imputation models (Duan et al., 2016; Ku et al., 2016). Such models have been found to be successful in obtaining reliable performance by converting the data imputation problem into a data cleaning/denoising problem. Typically, DSAE models extract the useful inherent correlations from the original data, recovering them from the high-level features with noise reduction. The basic idea of DSAE models is to train a "recovery tool" using both the raw data and imputed missing data. Therefore, a well-trained model will recover missing data with more reliable estimations compared to the conventional methods described above. However, such feature extraction compresses data into lower dimensions, which limits the variability of the model and makes model outcomes less interpretable.

Besides this, recurrent neural network (RNN) based GAN has also been used for time series data imputation. For instance, conditional Long Short-Term Memory (LSTM) based GAN has been used for medical data generation (Esteban et al., 2017) and traffic data prediction (Lv et al., 2018). On the other hand, Gated Recurrent Unit (GRU) based GAN has been used for multivariate time series generation (Luo et al., 2018). Similarly, Asadi and Regan (2019) adopted convolution recurrent autoencoder using bidirectional-LSTM layer as the encoder layer for spatial-temporal missing data imputation. However, training such RNN networks generally take significantly longer time, particularly when handling the long time sequence (>200) (Li et al., 2019). Recently, Chen et al. (2019) proposed parallel data based GAN model, which used the real data and synthetically generated data simultaneously for traffic data imputation to achieve state-of-the-art results. However, using the original daily traffic time series data as the latent space limits the generative ability of GAN based model thereby requiring each sensor to have its own generative model. This leads to training and managing a large number of models thereby making it difficult for large-scale application.

In this study, we propose a traffic data imputation framework based on generative adversarial network (TSDIGAN) encoding the time series into images. This enables us to treat the data imputation problem as image generation problem, thereby utilizing the significant developments

in DCGAN based image generation problems. We also compare our proposed model performance with the most recent state-of-the-art traffic data imputation based on GAN, proposed by Chen et al. (2019) to show the efficacy of our proposed model.

2.4 Technical and economic feasibility assessment of cloud-enabled traffic video analysis framework

As the conventional ITS sensors, the loop detector and microwave radar sensors are used to collect the traffic data periodically. However, maintaining and retrofitting these sensors is costly, and the data collected through these sensors are often aggregated so that the detailed information is lost (Dai et al., 2019). As another data source, traffic surveillance cameras are widely installed by Department of Transportation (DOT) to ensure the safety. However, monitoring the massive number of cameras manually is challenging and tedious (Liu et al., 2013). Thus, over the last decades, computer vision techniques for traffic images/videos analysis gained lots of attentions for the purpose of recognize the traffic pattern efficiently. In general, traffic recognition from videos streams comprises of 2 basic tasks: (1) vehicle detection and (2) vehicle tracking.

Vehicle detection model capture and localize the vehicles presented on the decoded individual video frame. There are two broad categories of detection model named one stage model and two stage model. The one stage detection model including the YOLOv3 (Redmon and Farhadi, 2018), SSD (Liu et al., 2016), RetinaNet (Lin et al., 2017) and so on are common used for object detection task. For two stage methods like R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2015) and Mask R-CNN (He et al., 2017) are also very popular methods for image understanding. After detecting and localizing the vehicles, multi-object tracking (MOT) model will then estimate trajectories of detected vehicles by associate their identifications across period of time in videos. The methods such as DeepSORT (Wojke et al., 2017), IOU (Bochinski et al., 2018), KLT (Lucas et al., 1981) and DCF (Lukezic et al., 2017) are often used for MOT tasks. By taking the benefits of advanced vehicle detection and tracking models, Dai et al. (2019) proposed the video-based vehicle counting framework using YOLOv3 and kernelized correlation

filters (KCF) tracker, which achieved 87.6% counting accuracy running at 20.7 fps. Meng et al. (2020) proposed a method using SSD and correlation-matched tracking algorithm to count the passing vehicle which achieved 93% counting accuracy running at 25 fps. Al-Ariny et al. (2020) adopted Mask R-CNN and KLT tracker for vehicle counting in videos which also achieved the high accurate result at 98.7% on test highway video. Similiarly, Lou et al. (2019) used Mask R-CNN combined with improved Kalman filter to detect and track the moving vehicles, and achieved the 95% accuracy running at 2.86 fps.

Although such advanced methods is promising for traffic recognition, such high computational intensity and processing delay is not tolerant for next generation real-time traffic video analysis framework, especially when handling the large-scale camera sensor system. To overcome this challenges, pushing the algorithm close to the sensor side, also called edge computing is a potential solution for analyzing large-scale live video streams in real time. Anjum et al. (2016) proposed cloud based video analytic framework for automated object detection and classification, which reduced the analyzing latency to 30.38 milliseconds per frame with full HD format by using GPUs mounted on compute servers in the cloud nodes. Liu et al. (2018) presented a real-time video analysis framework named EdgeEye which can analyze video stream (640x360 resolution) at 76 fps. Recently, Nvidia release the Deepstream 5.0 (Nvidia Deepstream, 2020), an AI-powered optimized video processing framework which enabled edge computing techniques to support the development of practical intelligent video analysis.

Future development of ITS applications are required to efficiently support the Smart City framework; thus, integrating the traffic video analysis framework and cloud server is needed (Khan et al., 2018; Petrolo et al., 2017). As summarized in Eltoweissy et al. (2019), running the ITS applications in the cloud server has the following benefits. First, cloud server gives users a massive pool of computation resources that eliminating the needs to install and maintain private clusters. Second, cloud server provide the flexible charging plan that allows users to pay the resources with short-term basis. In addition, a proper configured cloud server provide the worldwide remote access

that help the traffic agency, stakeholders, and governmental organizations access and understand the data easily.

In this study we introduced an end to end real-time cloud-enabled traffic analyais framework based on Nvidia Deepstream. And, we focused on evaluating the economic and technical feasibility of proposed framework to help the traffic agency build such practical framework efficiently. We deployed the proposed framework on Google cloud platform (GCP) to estimate the operating costs, and we evaluated the technical feasibility of proposed framework by measuring the vehicle-counting accuracy from various camera sensors with different viewing angles.

CHAPTER 3. LARGE-SCALE DATA-DRIVEN TRAFFIC SENSOR HEALTH MONITORING

3.1 Introduction

Intelligent Transport Systems (ITS) applications, such as for detection of traffic congestion or incidents (Chakraborty et al., 2018a,b) and for decision-making (Shi and Abdel-Aty, 2015; Ma et al., 2017; Mori et al., 2015) have shown great effectiveness for advanced traffic management. Implementation of these applications requires reliable, high-quality data collected from roadway sensors, such as radar sensors, loop detectors, and video detectors. However, inevitably these types of traffic sensors suffer from erroneous data due to communications loss and malfunctioning (Lee and Coifman, 2011). Therefore, it is important to determine sensor health conditions before data are used for real-time traffic operations purposes or planning/policy development.

Typically, traffic sensor health monitoring (TSHM) is done by matching sensor readings to predefined thresholds (Turochy and Smith, 2000; Chen et al., 2003). In these methods, thresholds are placed on the maximum volume or occupancy values observed by the sensors, number of sensors with zero volume and nonzero occupancy, average effective vehicle length, and other similar traffic statistics. Sensor health is determined based on the number of faulty records observed for each sensor. Other studies have also been performed where sensor health is determined at the network level by taking into consideration neighboring sensors (Sun et al., 2016; Lu et al., 2014).

However, thresholds determined by these methods are based on daily aggregate traffic data. Unfortunately, such high-level aggregation fails to capture sensors' frequent within-day temporal abnormalities, which can also indicate faulty sensor conditions, in part because detailed evaluation of sensors' temporal abnormalities requires the processing of large-scale traffic data, which is beyond the capabilities of traditional computation techniques. For example, data obtained from the 300 radar-based traffic sensors of Iowa account for 15 gigabytes monthly. In this study, we

have therefore developed a data-driven, massively parallelizable TSHM module that can handle large-scale statewide traffic data sources to detect abnormal sensors based on overall and temporal abnormalities.

Our TSHM module utilizes three steps for detecting abnormal sensors. First, based on clustering analysis, sensors with an abnormally high missing data percentage are labeled as faulty sensors. The module's second and third steps are based on the average effective vehicle length ($AEVL$) statistic. $AEVL$, proposed by Turochy and Smith (2000) combines volume, occupancy, and speed records using traffic flow theory to estimate vehicle dimensions. The second step of our TSHM module uses each sensor's $AEVL$ distribution to determine abnormal sensors using clustering analysis. $AEVL$ values, being representative of vehicles' physical dimensions, are robust to exogenous factors, such as traffic incidents and weather conditions. Therefore, the third step of our proposed TSHM module utilizes the assumption of constant vehicle length to determine changepoints in the temporal time-series data for each sensor. The temporal matrix of the changepoints obtained are then processed to extract the sparse matrix of all sensor abnormalities detected. Sensors that show frequent abnormalities can then be classified as abnormal sensors.

The major contributions of this study are as follows:

- We propose a data-driven stepwise method of anomalous sensor identification based on clustering analysis. This helps identify thresholds automatically for anomalous sensor detection. Our proposed method is similar to sieving analysis, wherein each sieve/step can be used to identify anomalous sensors by their distinct characteristics. This can enable authorities managing traffic sensors to easily identify sensor issues and take steps accordingly.

- The third step of our TSHM module. Based on the constancy of the vehicle length assumption, we identify the changepoints in the temporal matrix of the sensor data based on Bayesian analysis. And our entire method is scalable using massively parallelizable techniques, making it feasible to apply at a statewide level.

The rest of the paper is organized as follows. Section 2 provides a brief description of the relevant literature on sensor health monitoring followed by the details of the methodology adopted in this study in Section 3. Section 4 provides details of the data used and the results obtained. Finally, Section 5 provides a summary of our results and points to potential directions for future study.

3.2 Methodology

Sensor abnormalities can arise due to high missing data percentage or anomalous sensor readings. As explained earlier, according to traffic flow theory, $AEVL$ is an approximate function of speed, volume, and occupancy and is considered a useful indicator for monitoring traffic data quality. Therefore, in this study, we calculate $AEVL$ as follows to identify anomalous sensor readings:

$$AEVL = \frac{5280 \times S \times O}{V} \tag{3.1}$$

where, S and V denotes the average speed (miles/h) and the hourly flow rate (vehicles/h) during the time interval. O represents the occupancy, i.e., the fraction of the time the sensor is occupied with vehicles during a given time interval. The constant 5280 is the scalar conversion factor applied to the measurement unit (ft). Since $AEVL$ represents the physical dimensions of vehicles, it is robust to traffic anomalies such as incidents or bad weather. Further, Turochy and Smith (2000); Lu et al. (2014) have shown that AEVL can detect erroneous data records that cannot be detected using speed, occupancy, or volume individually. Therefore, our proposed TSHM module utilizes three steps to extract anomalous sensors from large-scale statewide $AEVL$ data based on missing data and erroneous sensor readings:

1. *Data Completeness Test:* Checks whether a traffic sensor has missing data by checking data readings for completeness.

2. *$AEVL$ Anomaly Test:* Identifies suspicious records by checking whether sensors have provided unreasonable volume, occupancy, or speed records.

3. *Temporal Pattern Anomaly Test:* Checks whether sensors have provided fluctuating $AEVL$ values over the daily time-series data.

Figure 3.1 shows the flowchart of our proposed large-scale data-driven TSHM screening algorithm.

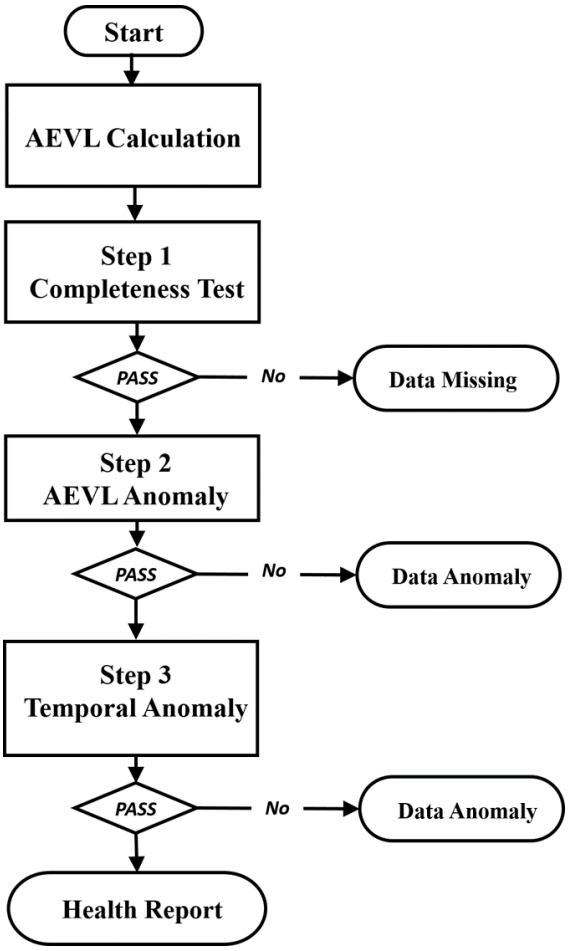


Figure 3.1: Workflow of proposed TSHM module

3.2.1 Setup

In the following subsections, N denotes the total number of sensors operating statewide and D the total number of days during the study period. In addition, the $AEVL$ for each sensor n on

each day d per 20-seconds interval (say t) with a total length of s is calculated as follows, with s being $3 \times 60 \times 24 = 4320$ for each sensor each day.

$$AEVL_n^d = \left(AEVL_n^{t_1,d}, AEVL_n^{t_2,d}, ..., AEVL_n^{t_s,d}\right) \tag{3.2}$$

3.2.2 Data Completeness Test

Long-time operation and various external reasons frequently lead to missing data issues in ITS traffic sensors (Turner et al., 2000). Therefore, our proposed TSHM module first attempts to detect sensors with an abnormally high missing data percentage, using completeness score (CS) as the metric for determining each sensor's missing data percentage Turner et al. (2000). CS is defined as the ratio of data readings received to the number expected. Thus, a lower CS score means a higher missing data percentage. The CS of each sensor for each day d was calculated with s the length of the time-series data:.

$$CS_{n,d} = \frac{s_{N_x,d}}{Max[s_{N_1,d}, s_{N_2,d}, ..., s_{N_x,d}]} \tag{3.3}$$

Then, the mean and standard deviation of the completeness score for a given sensor (n) over given days D was calculated and denoted as $S_n = (\mu_n^{CS}, \sigma_n^{CS})$. The features of the resulting 2-dimensional data were then used to determine anomalous sensors having abnormally high missing data percentages. This was accomplished via the unsupervised k-means clustering algorithm because of its computational efficiency (MacQueen et al., 1967; Berkhin, 2006). Here, $S \in \left\{\mu_n^{CS}, \sigma_n^{CS}\right\}_{n=1}^{N}$ defines the data points with length N, the total number of sensors. We first assign the data points to the K cluster centroids as $k_1, k_2, ..., k_j \in R$. In this step, each data sample will be assigned to the cluster c_i by calculating $L2$ distance between the point ($S_{(n)}$) and cluster centroid (k_j) as:

$$c^{(i)} = \arg\min_j \parallel S_n - k_j \parallel^2 \tag{3.4}$$

Then, the centroids k_j are recomputed and updated by taking the mean of all the data samples which were assigned to the cluster centered by that centroid:

$$k_j = \frac{\sum_{n=1}^{x} 1\{c^{(i)} = j\} S_n}{\sum_{n=1}^{x} 1\{c^{(i)} = j\}} \tag{3.5}$$

The algorithm then iteratively computes $c^{(i)}$ and k_j until convergence. In this study, we used the elbow method (Kodinariya and Makwana, 2013) in which K is chosen by drawing the sum of squared distances to provide the appropriate data separation and determine the optimal number of clusters K. (Kodinariya and Makwana, 2013). Ideally, for each given day over the month, the μ_{N_i} and σ_{N_i} of a normal sensor are expected to be close to 1 and 0 respectively. In other words, normal sensors are expected to have a high completeness score (close to 1) with low variance. Therefore, after convergence of the k-means clustering, any cluster with a centroid close to $(1, 0)$ was labeled a normal sensor group. The rest of the clusters were classified as abnormal sensors and assigned to anomalous sensor groups based on their different levels of distance from the normal sensor group in terms of missing data. This was based on the assumption that anomalous sensors with high levels of missing data are rare in the population while the majority of sensors perform normally, a fundamental assumption of unsupervised anomaly detection (Chandola et al., 2009). The next steps of the proposed TSHM module, the AEVL anomaly test, and temporal anomaly test, were then applied to sensors labeled as belonging to the normal sensor group to find any further issues.

3.2.3 AEVL Anomaly Test

The first step of our proposed TSHM module attempts to detect sensors with high levels of missing data. However, sensors with a low missing data ratio (and therefore classified as normal sensors in the data completeness test) can also suffer from abnormal, corrupted sensor readings due to calibration issues, double-counting of vehicles changing lanes, and other noise and errors. Hence, the second step of our proposed TSHM module, the $AEVL$ anomaly test, attempts to detect such sensors with frequently occurring noisy/corrupt readings.

To reduce noise in the raw AEVL data collected at 20-second to 5-minute intervals, data were first aggregated by taking the average, similar to Yao et al. (2017). Thus, $T = \{t_1, ..., t_z\}$ denotes the time-series data in 5-minute intervals of length z for each sensor on a given day. Thus, the $AEVL$ for sensor n at z^{th} 5-minutes interval can be denoted as $AEVL_n^{t_z,d}$. We then represented the

distribution of all $AEVL$ value for each sensor as a 2-dimensional feature vector $(\mu_n^{AEVL}, \sigma_n^{AEVL})$ using the aggregated AEVL records' mean and standard deviation.

The amount of data collected from state-wide traffic sensors is usually well beyond the capability of traditional computing techniques. For example, 500 MB of sensor data is collected daily in the state of Iowa, which aggregates to 15 GB monthly. A single conventional local machine cannot process such an enormous scale of data. To alleviate this issue, we used parallel computation techniques using Hadoop Distributed File System (HDFS) for storing the large-scale traffic data and Apache Pig Apache Pig (2018) for processing the data using MapReduce. While traditional single computer machines cannot process the monthly scale traffic data, usually the data processing can be completed in approximately 6 minutes. This enables abnormal sensor identification for statewide traffic data to be handled with ease.

Our methodology for detecting sensors with abnormal $AEVL$ records is based on the Density-based Spatial Clustering of Applications with Noise (DBSCAN) algorithm because of its great ability to handle outliers. The DBSCAN algorithm can be used to discover clusters of an arbitrary shape with good efficiency on large dataset with minimal requirements of domain knowledge to define clustering parameters (Ester et al., 1996). DBSCAN has been used as an efficient clustering algorithm in different applications, including both classification and outlier/anomaly detection problems((Erman et al., 2006), (Nisa et al., 2014)). The fundamental assumption for DBSCAN in anomaly detection problems is that while normal data instances belong to clusters, anomalous data do not belong to any cluster (Chandola et al., 2009) and can therefore be attributed to noise or anomalies. The basic steps to identifying anomalous sensors based on the 2-dimensional feature vector $VL_n = (\mu_n^{AEVL}, \sigma_n^{AEVL})$ using DBSCAN algorithm is as follows (Schubert et al., 2017):

1. The algorithm starts with a random point $VL_n = (\mu_{AEVL,n}, \sigma_{AEVL,n})$ calculated from the $AEVL$ records of sensor n from N number of total sensors. If the point p_n is unclassified, the algorithm continues to find neighbors based on the two basic DBSCAN functions, namely the 'range query' function against other points defined as:

$$dist(P_i, P_j) = \sqrt{(\mu_{AEVL,P_i} - \mu_{AEVL,P_j})^2 + (\sigma_{AEVL,P_i} - \sigma_{AEVL,P_j})^2} \tag{3.6}$$

and the 'distance function' with the searching radius to find neighbors defined as:

$$Neighbors_{\epsilon}(P_i) = \{P_j \in DB \,|\, dist(P_i, P_j) \leq \epsilon\} \tag{3.7}$$

If the number of neighbors is greater than $minPts$, they will be defined as a cluster. If point p_n does not belong to any cluster, it will be labeled as noise.

2. After defining the distance function and the -neighborhood function, two input hyperparameters need to be chosen: (a) $minPts$ which defines the minimum number of neighbors to build a cluster, and (b) ϵ which defines the searching radius to find any neighbors. We determined these two input parameters based on the sorted k-dist graph technique. We first define the $minPts$ as k, and drew the sorted k-dist curve, with the first *valley* of the curve being considered the threshold to divide noise from other points (Ester et al., 1996). This allowed us to perform iterated search on a wide range of real-world data to determine the optimal k through reaching stability.

3.2.4 Temporal Pattern Anomaly Test

The $AEVL$ anomaly test captures anomalous sensors with suspicious data readings based on high-level aggregated $AEVL$ data distributions, represented by $\mu_{AEVL,n}$ and $\sigma_{AEVL,n}$. However, sensors reporting abrupt $AEVL$ fluctuations cannot be detected using aggregated distributions. Thus, it is necessary to consider each individual sensor's time-series data to detect this latter type of abnormal sensor. Therefore, after severely abnormal sensors had been captured through the $AEVL$ anomaly test, we used the temporal pattern test to examine the remaining sensors to flag any with frequent temporal abnormalities.

The core idea of the temporal pattern anomaly test is based on the constancy of vehicle length. Unlike raw 20-second-interval data, aggregated 5-minute-interval $AEVL$ moving averages reduce data readings' inherent noise, so that $AEVL$ values over a given time period are more stable (Wells et al., 2008). Thus, $AEVL$ data from a given sensor at times T_i and T_{i+1} should, based on traffic operations theory, fall into the same distribution. In contrast, if many abrupt changes in the

$AEVL$ distribution exist over a given sensor's time-series data for a given day, it can be labelled as a potentially anomalous sensor.

Figure 3.7 shows the $AEVL$ time series plot of three consecutive sensors at a given day. The unanticipated "spikes" in the middle sensor (Figure 3.7b) compared to its upstream (Figure 3.7a) and downstream (Figure 3.7c) sensors indicate the middle sensor have recorded abnormal data, since its nearby sensors provide predictable data. The third step of our proposed TSHM module therefore attempts to detect abnormal sensors that report frequent abrupt fluctuations in $AEVL$ values. Its first task is to detect "spikes" (also referred to as changepoints) and then detect which sensors are reporting such frequent changepoints by extracting the temporal pattern of all changepoints detected.

3.2.4.1 Changepoint Detection

To detect abrupt $AEVL$ distribution changes over a given set of time-series data, we adopt changepoint detection, also known as time-series segmentation. Sensor faults in the real world are usually random and therefore unpredictable. Thus, we cannot know how many failures or changepoints will occur during a given time period. Therefore, abrupt $AEVL$ changepoint detection can be considered a multiple changepoint detection problem based on an exact Bayesian changepoint model (detailed mathematical description in (Fearnhead, 2006)).

In this study, we used the 5-minute $AEVL$ records ($AEVL_n^{t_z,d}$) determined from the step 2 of our TSHM module for change point detection. Let us assume that given time series has m changepoints with their positions referred to as $\tau_{1:m} = (\tau_1, \tau_2, ..., \tau_m)$ where $\tau_i < \tau_j$ for $i < j$ and the two change points at either end of the time series can be denoted $\tau_0 = 0$ and $\tau_{m+1} = z$. Note, $z = 288$ is the length of time series for a given day with 5-minute aggregated data. Further, the m changepoints will split the time series data into $m+1$ sub-segments and the observations of the j^{th} sub-segment will consist of $AEVL$ records from τ_{j-1} to τ_j. Then, an arbitrary prior distribution for the j^{th} sub-segment, which is mutual for all other sub-segments with a set of distribution parameters denoted as β. Further, the observations of any given subsegment being conditional

independent of other subsegments' observations can also be assumed (Fearnhead and Liu, 2007). Therefore, the probability that two time points say (t and s) will belong to the same sub-segment can be represented as:

$$\begin{aligned} \mathrm{P(t,s)} &= \mathrm{Pr(AEVL_{t:s} \,|\, t, s \text{ in the same sub-segment})} \\ &= \int \prod_{i=t}^{s} f(AEVL_i \,|\, \beta)\pi(\beta)d\beta \end{aligned} \tag{3.8}$$

where, $\pi(\beta)$ denotes the prior with the parameter β. Then the marginal likelihood of the segment at $AEVL_{t:z}$ given a changepoint at $t-1$ can be defined as:

$$Q\left(t\right) = \begin{cases} \sum_{s=t}^{z-1} P\left(t,s\right) Q\left(s+1\right) g\left(s+1-t\right) + P\left(t,z\right)\left(1-G\left(z-t\right)\right), & \text{if } t = 2, ..., z, \\ \\ \sum_{s=1}^{z-1} P\left(1,s\right) Q\left(s+1\right) g_0\left(s\right) + P\left(1,z\right)\left(1-G_0\left(z-1\right)\right), & \text{if } t = 1 \end{cases} \tag{3.9}$$

where $g(t)$ is the probability mass function of the time interval between two successive changepoints, and $g_0(t)$ is the probability mass function of the first changepoint after 0. Thus, the $G(t) = \sum_{s=1}^{t} g(s)$ and $G_0(t) = \sum_{s=1}^{t} g_0(s)$ denotes the probability distribution respectively. Then the posterior probability distribution of the first changepoint can be derived as:

$$\begin{aligned} \mathrm{Pr}\left(\tau_1 | \mathrm{AEVL_{1:z}}\right) &= \mathrm{Pr(AEVL_{1:z}, \tau_1)/Pr(AEVL_{1:z})} \\ &= \mathrm{Pr}(\tau_1)\mathrm{Pr(AEVL_{1:\tau_1} \,|\, \tau_1)Pr(AEVL_{\tau_1+1:z} \,|\, \tau_1)Q(1)} \\ &= \mathrm{P}(1,\tau_1)\mathrm{Q}(\tau_1+1)\mathrm{g_0}(\tau_1)/\mathrm{Q}(1) \end{aligned} \tag{3.10}$$

Then based on τ_{j-1} for $\tau \in (1, z-1)$, the remaining changepoints τ_j can be derived as:

$$\mathrm{Pr}\left(\tau_\mathrm{j} | \tau_{\mathrm{j}-1}, \mathrm{AEVL_{1:z}}\right) = \mathrm{Pr}(\tau_{\mathrm{j}-1}+1, \tau_\mathrm{j})\mathrm{Q}(\tau_\mathrm{j}+1)\mathrm{g}(\tau_\mathrm{j}-\tau_{\mathrm{j}-1})/\mathrm{Q}(\tau_{\mathrm{j}-1}+1) \tag{3.11}$$

In our case, the changepoints indicate changes in vehicle length distribution (i.e., a changing variance problem see (Fearnhead and Liu, 2007)). Specifically, for the normal upstream and downstream sensor in Figure 3.7, there is a higher chance to observe change points at around 5:30 AM and 11:30 PM which are recurrent. This observation follows standard traffic flow characteristics

on freeways, where the traffic volume at night is lower than during the day. Therefore, the aggregated 5-minute interval vehicle length moving average has larger variance at night due to a smaller sample size. In contrast, changepoints present in abnormal sensors occur arbitrarily and erratically throughout the entire day.

This observation follows the traffic flow characteristics on the freeways, where the traffic volume during the night time is lower than during the day time. Therefore, the moving average for vehicle length over 5-minutes aggregation will have larger variance during the night times due to smaller sample size. On the other hand, the change points present in the abnormal sensor are arbitrary and erratic occurring throughout the entire day.

3.2.4.2 Abnormal sensor detection in the temporal changepoint matrix

To detect sensors reporting an abnormal temporal pattern in the changepoint matrix, we used changepoint probabilities calculated as described above to form changepoint temporal matrices as in Figure 3.8. Specifically, changepoint probabilities calculated at each 5-minute interval (t_z) for sensor n at a given day d can be denoted as $CP_n^{t_z,d}$. We then accumulated the changepoint probabilities for each 5-minute interval for all days in the study period (each month of historical data) to obtain $CP_n^{t_z}$ as follows.

$$CP_n^{t_z} = \sum_{d=1}^{D} CP_n^{t_z,d} \tag{3.12}$$

This aggregated $CP_n^{t_z}$ was then used to create the following temporal matrix:

$$\begin{pmatrix} CP_1^{t_1} & CP_1^{t_2} & \dots & CP_1^{t_z} \\ CP_2^{t_1} & CP_2^{t_2} & \dots & CP_2^{t_z} \\ \dots & \dots & \ddots & \vdots \\ CP_n^{t_1} & CP_n^{t_2} & \dots & CP_n^{t_z} \end{pmatrix} \tag{3.13}$$

This change point matrix consists of two distinct data features comprised of (a) recurrent change point pattern and (b) abrupt presence of change points by anomalous sensors. Therefore, we need to extract enhanced anomalous data features that can be used to detect abnormal sensors

reliably. This is done in this study using Robust Principle Component Analysis (RPCA) (Candès et al., 2011). RPCA has been successfully used in literature to detect moving objects from the video surveillance system (Bouwmans and Zahzah, 2014). RPCA attempts to decompose the changepoint matrix CP into a low-rank temporal matrix (L) and a sparse temporal matrix S as follows.

$$CP = L + S \tag{3.14}$$

The low-rank matrix reflecting the recurrent changepoint pattern (say the background) and the sparse temporal matrix reflecting the abrupt presence of changepoints (say the foreground) are calculated as follows:

$$\min_{L,S} \text{rank}(L) + \lambda \parallel S \parallel_0 \quad s.t\ M - L - S = 0 \tag{3.15}$$

where $\lambda > 0$ is a balancing parameter. However, this is a non-convex problem for optimization and solving the $rank$ and $l_0 - norm$ are NP-hard. The relaxation function with the convex envelop is obtained by replacing the $l_0 - norm$ by the $l_1 - norm$ ($\parallel . \parallel_1$) and replacing $rank$ with nuclear norm ($\parallel . \parallel_*$):

$$\min_{L,S} \parallel L \parallel_* + \lambda \parallel S \parallel_1 \quad s.t\ M - L - S = 0 \tag{3.16}$$

where the $\lambda > 0$ is chosen by $\lambda = \frac{1}{\sqrt{\max(m,n)}}$. The RPCA algorithm to decompose the raw change point temporal matrix (CP) into the low-rank matrix and the sparse matrix. Then, we used the DBSCAN clustering algorithm described earlier to detect abnormal sensors. Clustering was done using the mean and standard deviation of the sparse temporal matrix values ($S_n^{t_z}$) for each sensor obtained from RPCA.

3.2.5 Baseline Comparison

To verify the feasibility and accuracy of our proposed TSHM module, we compared its algorithmic performance with 2 benchmark algorithms: 1) fixed-threshold based $AEVL$ control limit method (CLM) (Wells et al., 2008), and 2) the temporal and spatial comparison screening algorithm using multiple comparison with the best (MCB) technology (Lu et al., 2014).

The core idea of the CLM method is based on the 95% confidence interval calculated from the overall $AEVL$ distribution. The CLM method first identifies the experimental time period (31 days

in our study's example month of July 2017) and then calculates $AEVL$ for all sites to form the $AEVL$ distribution. Then, the mean $AEVL$ is obtained from the distribution, denoted as μ_{cl}. The upper and lower control limit boundaries can be obtained by $\mu_{cl} + 2\sigma_{cl}$ and $\mu_{cl} - 2\sigma_{cl}$ respectively. If the average $AEVL$ value of any individual sensor falls outside the control limits, the sensor will be flagged as potentially anomalous for further study.

he MCB algorithm flags anomalous sensors on the basis of temporal and spatial information by comparing $AEVL$ in three steps: (1) Data aggregation: Individual $AEVL$ data points are grouped into 30-minute intervals for each sensor and each lane, and represented by the mean $AEVL$ and variance. (2) Within station comparison: MCB are performed between different lanes in the same station, using the confidence interval created by MCB to check if there is a statistically significant difference between the target lane and best max lane. (3) Between station comparison: MCB are performed between nearby stations uses the fuzzy logic decision tree to label potential errors when target lane/station data are significantly higher/lower than that of comparison lanes/stations.

3.3 Results

Traffic sensor data were collected from Iowa's 338 freeway radar sensors statewide from April to December 2017. These radar sensors use digital radar beams (virtual lines) to record passing vehicles' speed, occupancy, volume, vehicle type, etc. The volume of each month's sensor data aggregated at 20-second intervals is approximately 15 GB of distributed storage (via a Hadoop Distributed File System or HDFS). In this paper, we show sample results and test our proposed module against the two benchmark algorithms using sensor data from a sample month (July 2017), utilizing our remaining data for sensitivity and stability analysis. In real-world applications, however, our module could be implemented with a sliding window of 1 month to detect sensors functioning abnormally over the last month. We next discuss the results of testing our proposed algorithm on aggregated data for each month of the study period via our TSHM module's 3 steps: (a) data completeness check, (b) $AEVL$ anomaly check, and (c) temporal pattern anomaly check. We then compare our proposed algorithm against the benchmark algorithms described previously.

3.3.1 Data Completeness Test Results

The first step of our proposed TSHM module is the data completeness test to detect sensors with abnormally high missing data percentages. As discussed in Section 3.2.2, this uses K-means clustering based on each sensor's completeness score (CS) mean and standard deviation. For 8 of the 9 months of the study period (April–December 2017), the elbow method described in Section 3.2.2 found the optimum number of clusters to be 3, while for August 2017, it found the optimum number of clusters to be 2. Figure 3.2 shows the clusters from the sample month July 2017. These 3 clusters can be labelled based on missing data severity as: normal sensor, abnormal sensor level 1 and abnormal sensor Level 2 based on their severity of data missing problem. In other words, the cluster with the mean CS closest to 1 and standard deviation close to 0 (i.e., $S_n = (\mu_n, \sigma_n) \approx (1, 0)$) can be labelled as normal and the other two labelled as anomalous. Thus, in July 2017, Iowa had 299 operating freeway radar sensors, 36 of which were abnormal.

Figure 3.3 shows heatmaps of the 3 missing-data-percentage-based clusters from July 2017. As can be seen from Figure 3.3a, normal sensors rarely suffer missing data issues, while level 2 abnormal sensors have an exceptionally high percentage of missing data (Figure 3.3c), Although level 1 abnormal sensors' missing data rates are not excessive (Figure 3.3b), still they justify manual inspection and repairs. (It should be mentioned that although our proposed clustering-based method can automatically classify sensors as normal vs. abnormal, traffic operations agencies could also define anomalous sensors based on their individual requirements, choosing based on the cluster centers obtained the $S_n = (\mu_n, \sigma_n)$ for proposed method uses to identify abnormal sensors.)

It should be noted that sensors classified as normal in this data completeness test can nonetheless suffer from abnormal/atypical recorded values. Such "normal sensors" were therefore passed through step 2 of our proposed TSHM module, the AEVL anomaly test.

3.3.2 AEVL Anomaly Test Results

Sensors with no major missing data issues can still be abnormal by recording atypical sensor values. Therefore, the $AEVL$ anomaly test, step 2 of our proposed TSHM module, uses the

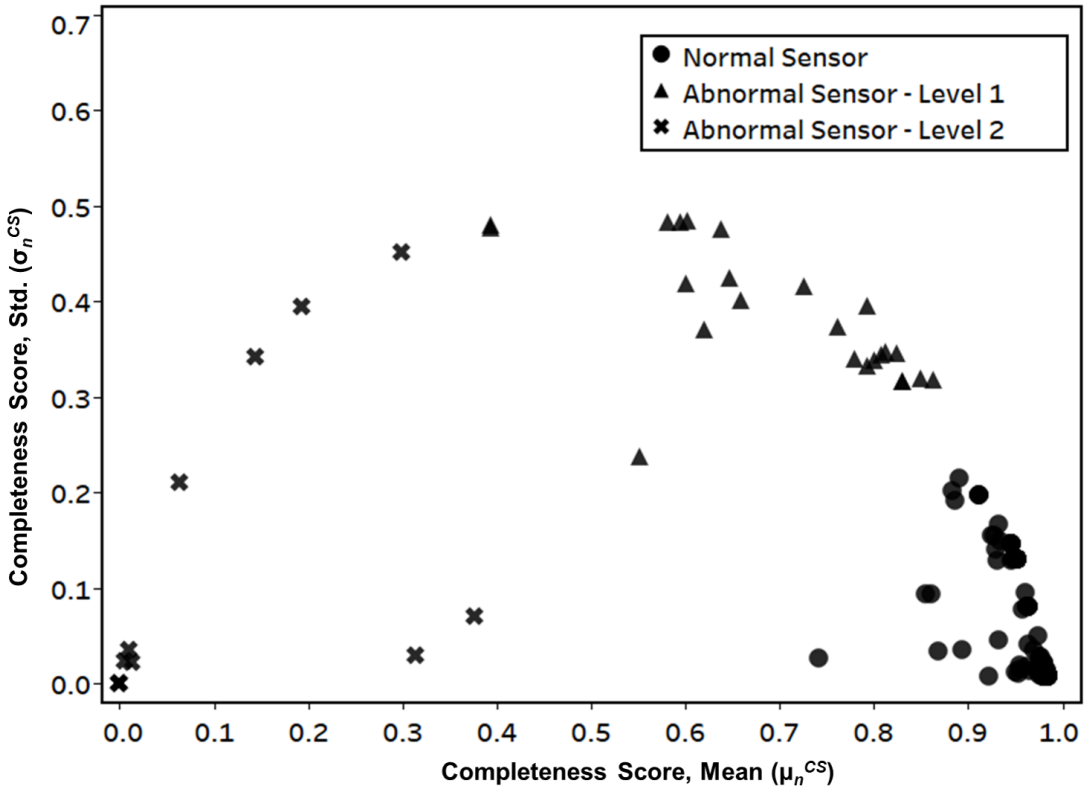


Figure 3.2: Data completeness test result on the sample month of July

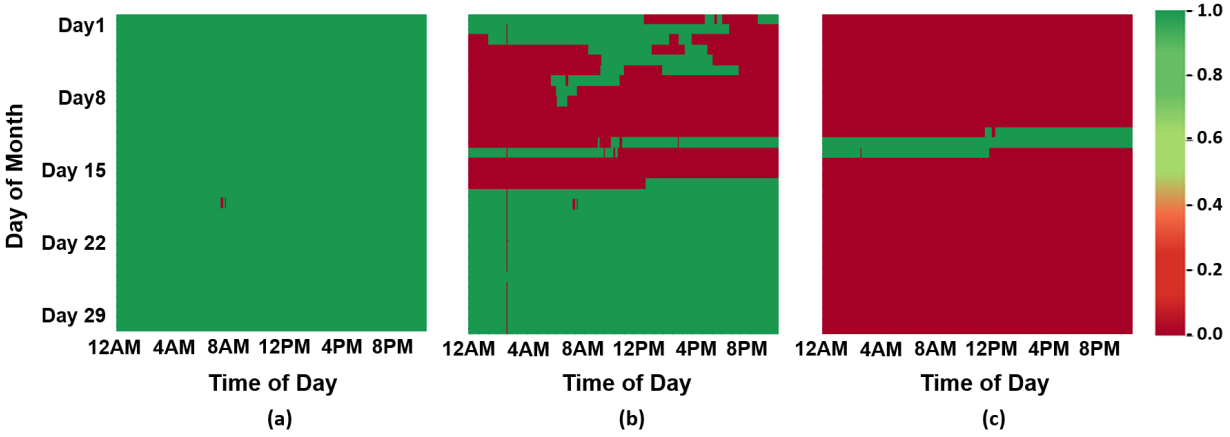


Figure 3.3: Sample missing data percentage heatmap of (a) Normal sensor, (b) Abnormal Sensor - Level 1, and (c) Abnormal Sensor - Level 2 from the data completeness test

DBSCAN algorithm described in Section 3.2.3 to detect sensors recording abnormal $AEVL$ values. Figure 3.4 shows abnormal sensors detected in the sample month of July 2017 based on the 2-d $AEVL$ distribution $(\mu_n^{AEVL}, \sigma_n^{AEVL})$. Out of the 263 sensors classified as normal sensors based on data completeness test for July 2017, 13 were classified as abnormal based on the $AEVL$ anomaly test.

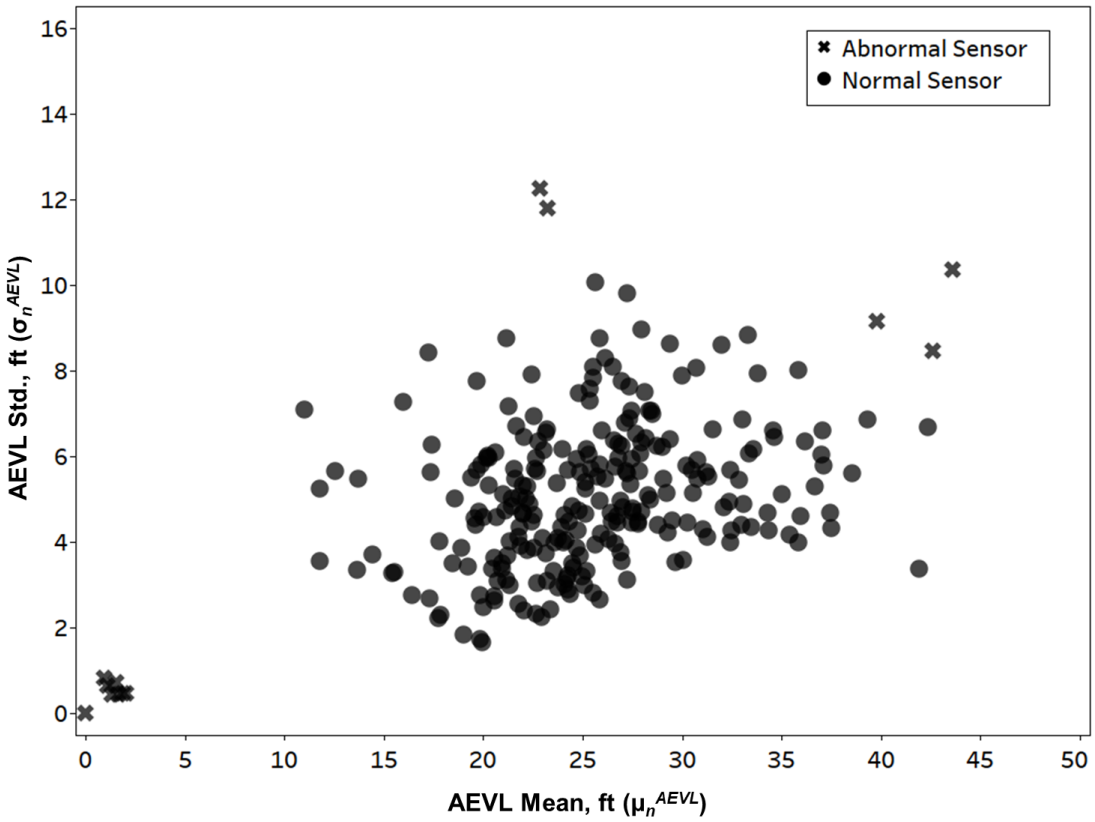


Figure 3.4: AEVL anomaly test result for the sample month of July, 2017

It can be seen in Figure 3.5, where we plot the 2-d completeness score $S_n = (\mu_n, \sigma_n)$ obtained from step 1's data completeness test with the sensor labels obtained from step 2's AEVL anomaly test, that the abnormal sensors detected in step 2 cannot be detected using step 1's CS feature vector S_n, since the sensors identified in step 2 didn't have any missing data issues. This demonstrates that both the data completeness and AEVL anomaly tests are required, since each identifies abnormal sensors having different issues that cannot be detected using a single test.

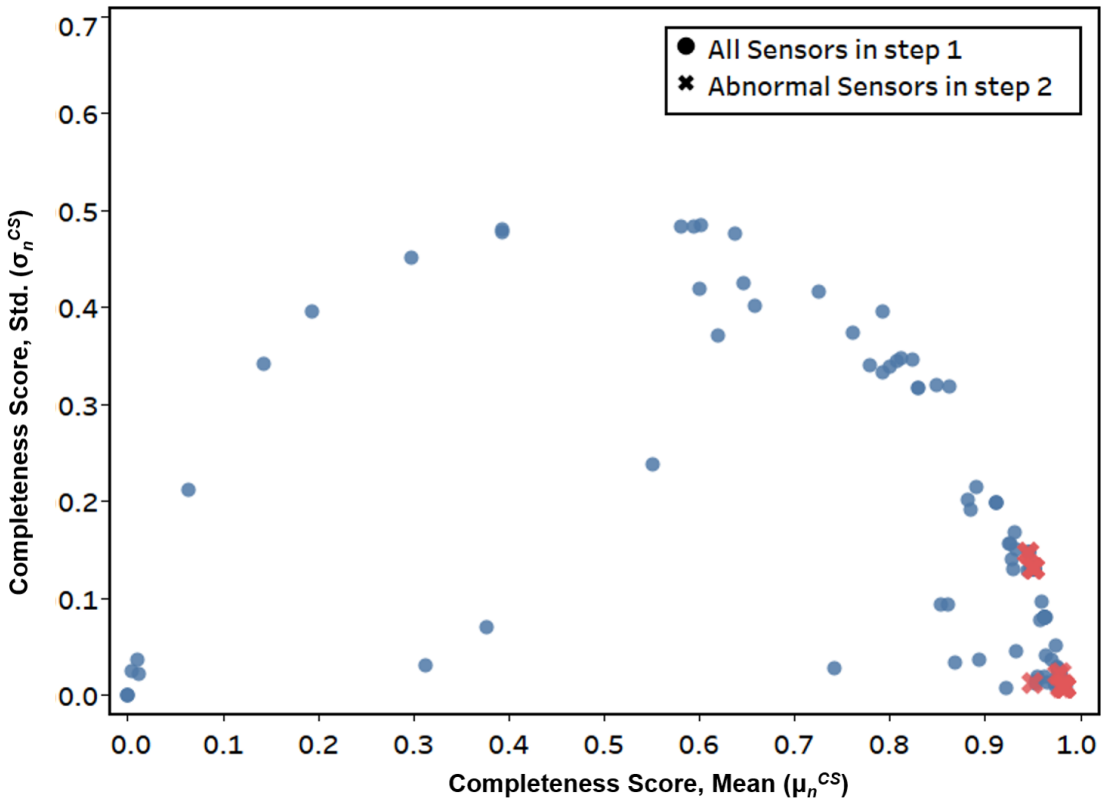


Figure 3.5: Completeness Score plots for all sensors in step 1 and abnormal sensors in step 2

To illustrate the difference in $AEVL$ distributions observed in anomalous vs. normal sensors, Figure 3.6 shows the different types of $AEVL$ cumulative distribution function (CDF) plots reported by three sample anomalous sensors along with their adjacent upstream (u/s) and downstream (d/s) "normal sensors." In Figures 3.6a and 3.6b, the abnormal sensors reported lower and higher means, respectively, compared to their adjacent u/s and d/s sensors. In contrast, the abnormal sensor shown in Figure 3.6c reported higher variance compared to its adjacent sensors. Such CDF visualization of abnormal sensors helps justify our proposed method of detecting sensors reporting abnormal sensor readings by clustering $AEVL$ distributions $(\mu_n^{AEVL}, \sigma_n^{AEVL})$.

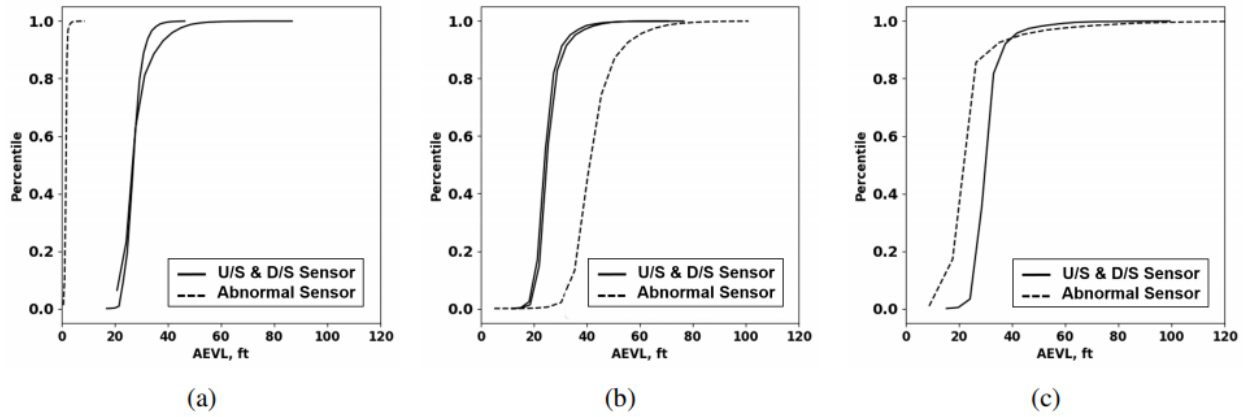


Figure 3.6: Sample CDF plots of abnormal sensor with their upstream(u/s) and downstream(d/s) sensors for abnormal sensor with: (a) low mean, (b) high mean, and (c) high variance

3.3.3 Temporal Pattern Anomaly Test Results

The temporal pattern anomaly test detects sensors showing abrupt fluctuations or "spikes" in $AEVL$ time-series values by using the Bayesian changepoint detection algorithm, as described in Section 3.2.4.1, and RPCA to denoise and enhance the changepoint matrix as described in Section 3.2.4.2. AEVL as a surrogate for vehicle length is not expected to be affected by time of day, unlike volume or speed. Therefore, frequent fluctuations in $AEVL$ values suggest sensor abnormality.

Figure 3.7 shows the raw $AEVL$ values and corresponding changepoint probability distributions of three consecutive sensors on a sample day. It can be seen that the middle sensor (Figure 3.7b)

shows a significant number of spikes in $AEVL$ values, resulting in an increase in changepoint probabilities (Figure 3.7e) compared to it's u/s and d/s sensors. Each sensor's changepoint temporal matrix was formed by accumulating its changepoint probability at each 5-minute interval over the days of the study as described in Section 3.2.4.1.

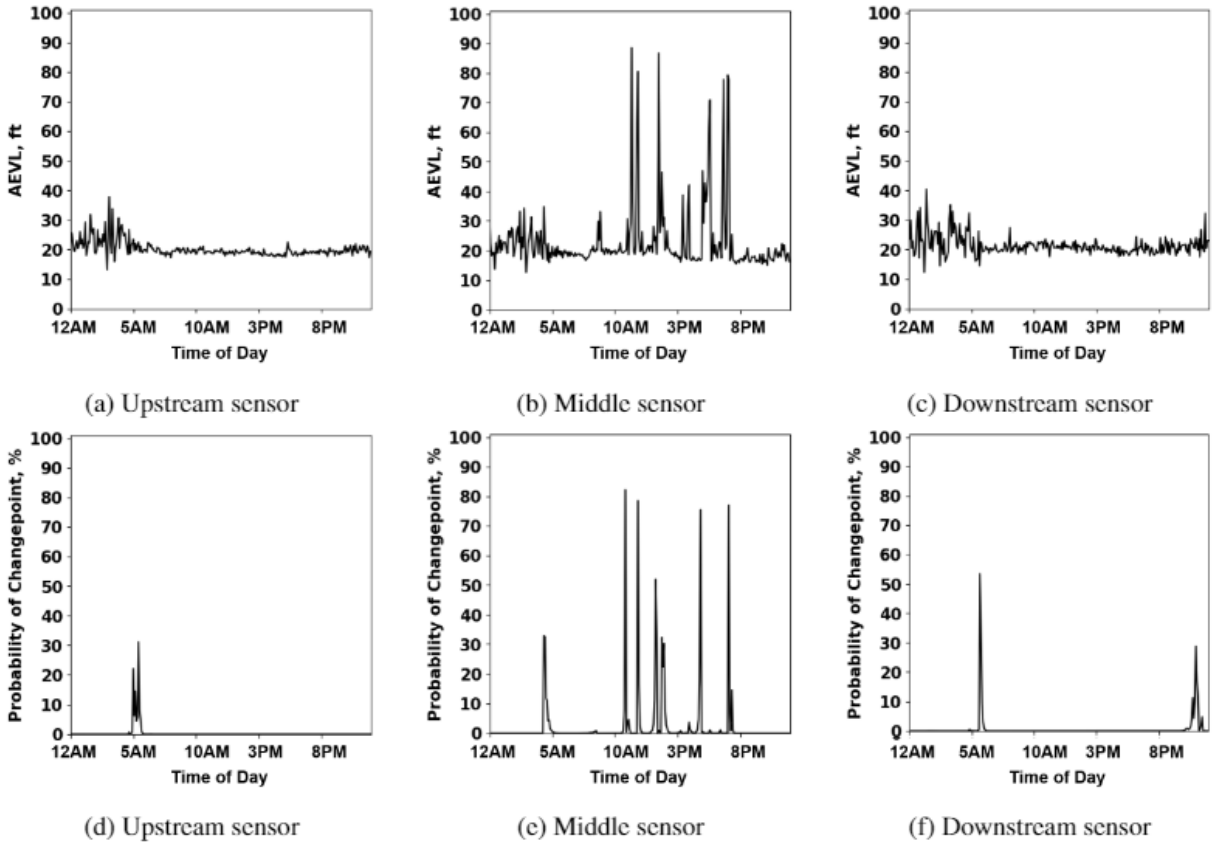
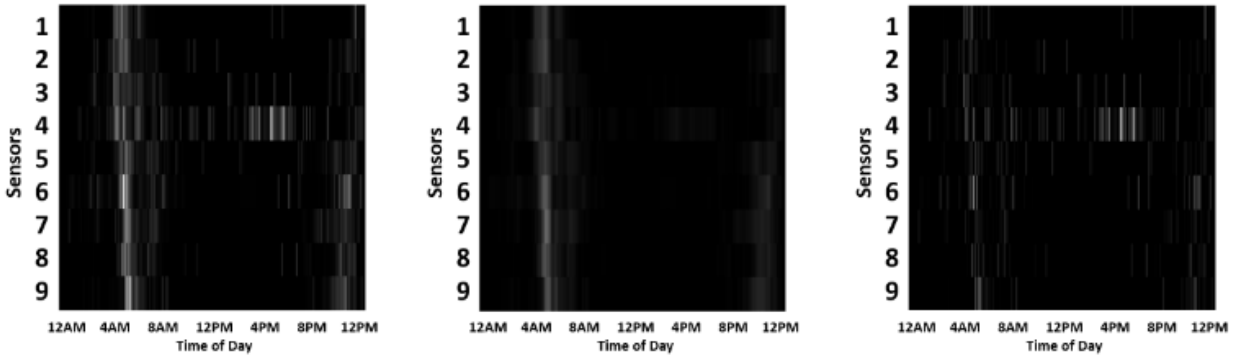


Figure 3.7: $AEVL$ time series distribution of a sample day for (a) Upstream sensor, (b) Middle anomaly sensor, (c) Downstream sensor and changepoint probability distribution for the same day for the (d) Upstream sensor, (e) Middle anomaly sensor, (f) Downstream sensor

Figure 3.8 shows sensors in a sample plot following their actual spatial order on the freeway. In the sparse matrix shown in Figure 3.8c, we can see sensor 4 shows abrupt changes in $AEVL$ over the study period, while its adjacent sensors are following the predictable temporal pattern.

Then, similarly to the AEVL anomaly test, we used DBSCAN clustering algorithm to detect the anomaly sensors by calculating each sensor's sparse matrix mean and standard deviation as a 2-d



(a) Raw CP matrix (b) Low rank CP matrix (c) Sparse CP matrix

Figure 3.8: Sample RPCA Decomposition: (a) Raw CP matrix, (b) Low rank CP matrix, (c) Sparse CP matrix

feature vector. Figure 3.9 shows the July 2017 sample month's clustering results in which 8 sensors (2.04%) were classified as abnormal. In Figure 3.10, we visualize the different characteristics of one randomly selected anomalous sensor and one randomly selected normal sensor using a heatmap of their raw $AEVL$ data It can be seen that compared with the normal sensor (Figure 3.10a), the abnormal sensor (Figure 3.10b) shows more "spikes" or temporal fluctuations, resulting in it having been flagged as anomalous.

Like before in Figure 3.5, in Figure 3.11 we verify the necessity of our TSHM modules's step 3 temporal pattern anomaly test by comparing its results with our step2 $AEVL$ anomaly test.Specifically, we plot the 2-d $AEVL$ distribution $(\mu_n^{AEVL}, \sigma_n^{AEVL})$ obtained from step 2 with the labels of the sensors obtained from step 3. Again, it can be seen that the abnormal sensors with frequent $AEVL$ fluctuations detected in step 3 cannot be identified in step 2, justifying that our temporal pattern anomaly test is required.

3.3.4 Baseline Comparison

In this section, we compare our proposed TSHM module with the baseline CLM and MCB methods described in Section 3.2.5.

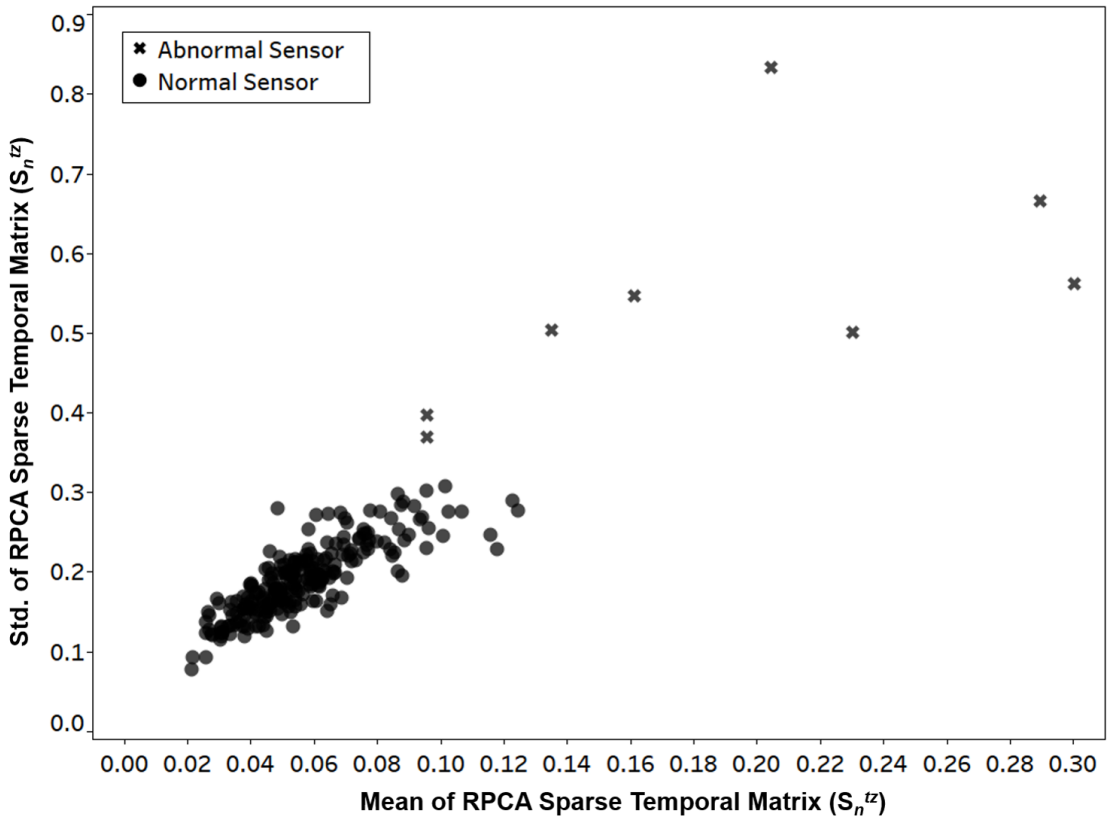


Figure 3.9: RPCA sparse temporal matrix clustering

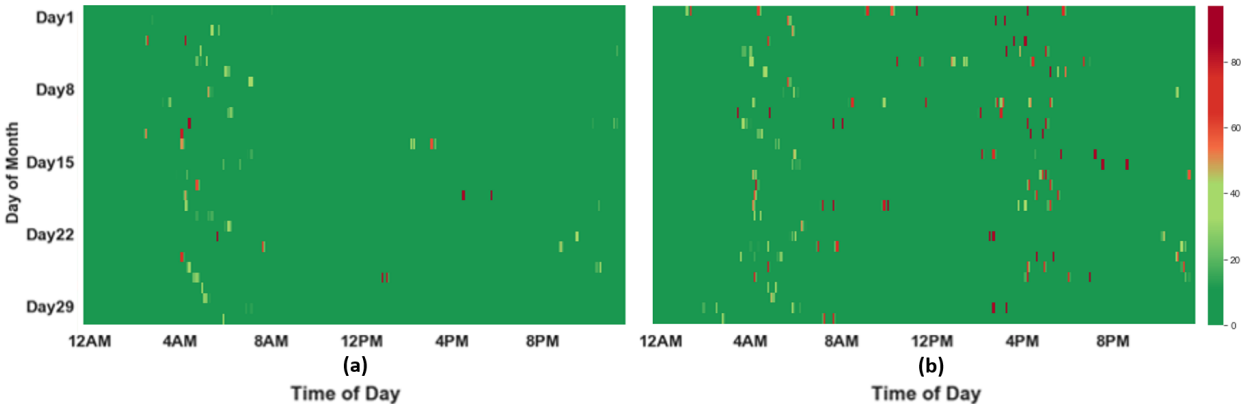


Figure 3.10: RPCA Sparse Matrix Clustering heatmap

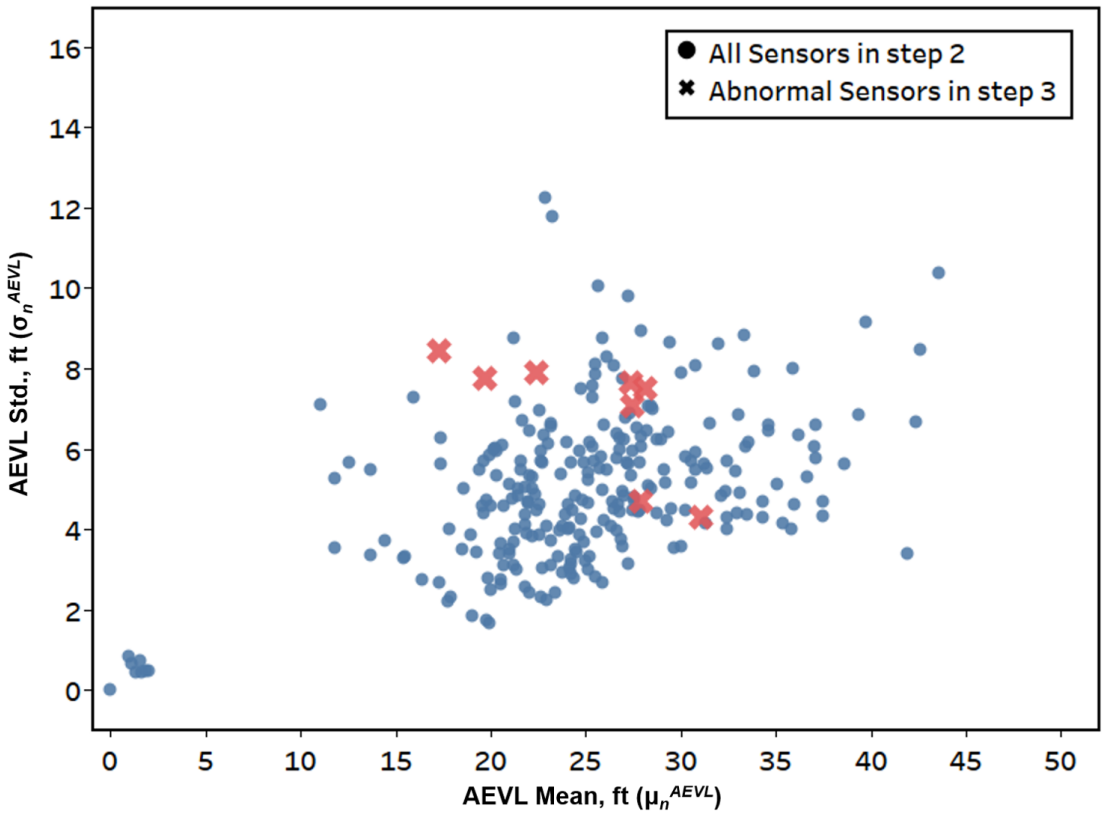


Figure 3.11: AEVL plots for all sensors in step 2 and abnormal sensors in step 3

3.3.4.1 CLM Comparison

The CLM method attempts to detect abnormal sensors based on $AEVL$ values. Since CLM method doesn't deal with any missing data issues, so we don't use the results obtained from our step 1 (data completeness test) in this comparison and only rely on the remaining two steps ($AEVL$ anomaly test and temporal pattern anomaly test) since these also use $AEVL$ as the primary variable. Figure 3.12 shows the comparison's results for the sample month July 2017. All anomalous sensors detected using the CLM method (3.04% or 8 out of 263) were also labelled anomalous by our method, but our proposed method also labelled an additional 13 sensors (4.94%) as anomalous, 5 in step 2 and 8 in step 3.

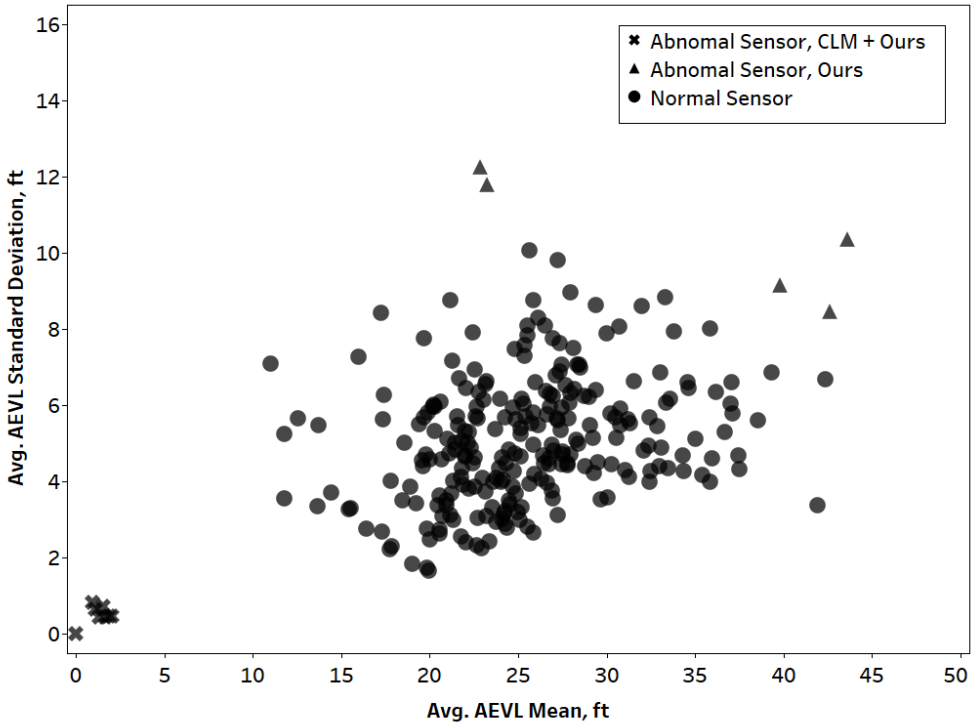


Figure 3.12: Clustering comparison with CLM method

Figure 3.13 shows heatmaps of the raw $AEVL$ values for the study month (July 2017) for a sample normal sensor and three different abnormal sensors. Figures 3.13a and 3.13b show heatmaps for sample sensors labeled normal and abnormal sensor, by both CLM and our proposed method.

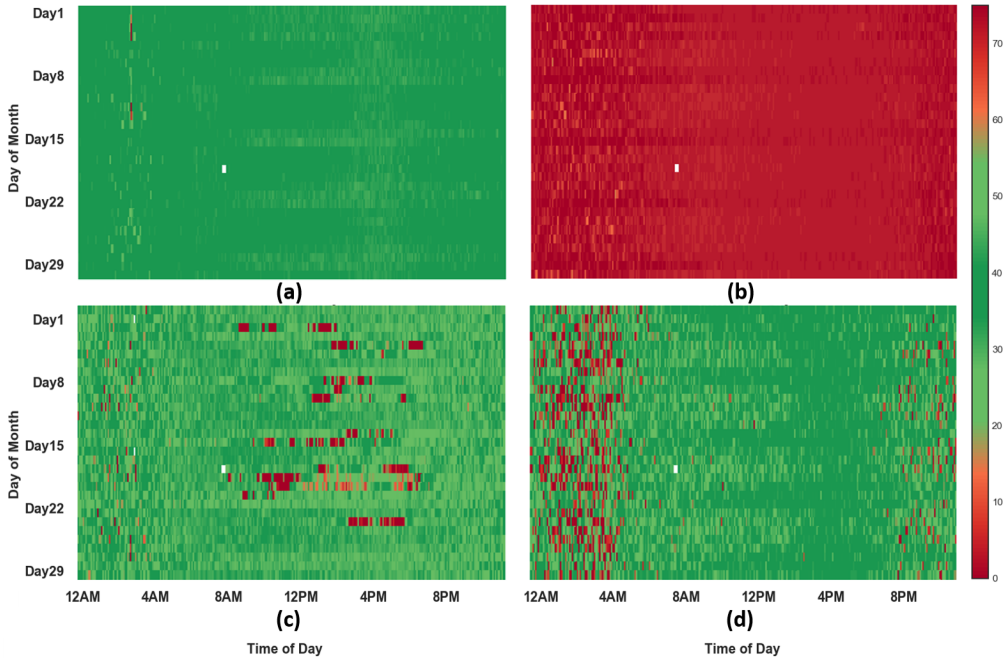


Figure 3.13: Sample monthly $AEVL$ heatmap of July, 2017 for sample (a) normal sensor and (b-d) abnormal sensors

Figures 3.13c and 3.13d show $AEVL$ heatmaps of sample sensors detected only by our proposed TSHM module (in step 2 and 3) that reported either intermittent abnormal $AEVL$ values or frequent $AEVL$ fluctuations, thereby justifying the efficacy of the proposed method.

3.3.4.2 MCB Method

The MCB algorithm requires sensors to be spatially ordered. However, generating the accurate spatial ordering of all sensors in Iowa is time-consuming. Therefore, we selected 4 different routes for evaluation, namely I-235 EB (with the 6 sensors A1–A6), I-35 NB (with the 6 sensors B1–B6), I-80 EB (with the 7 sensors C1–C7), and I-74 NB (with the 7 sensors D1–D7). Each route's head and tail sensors were ignored in the evaluation, since they do not have the u/s and d/s sensors required for MCB comparison. Figure 3.14 shows the remaining 18 sensors' normal/abnormal status according to our proposed method vs. MCB. Our proposed method labels 7 of these as anomalous (A3, A5, C2, C5, D3, D4, and D5), whereas MCB labels only 4 (A5, B3, C2, and C5)

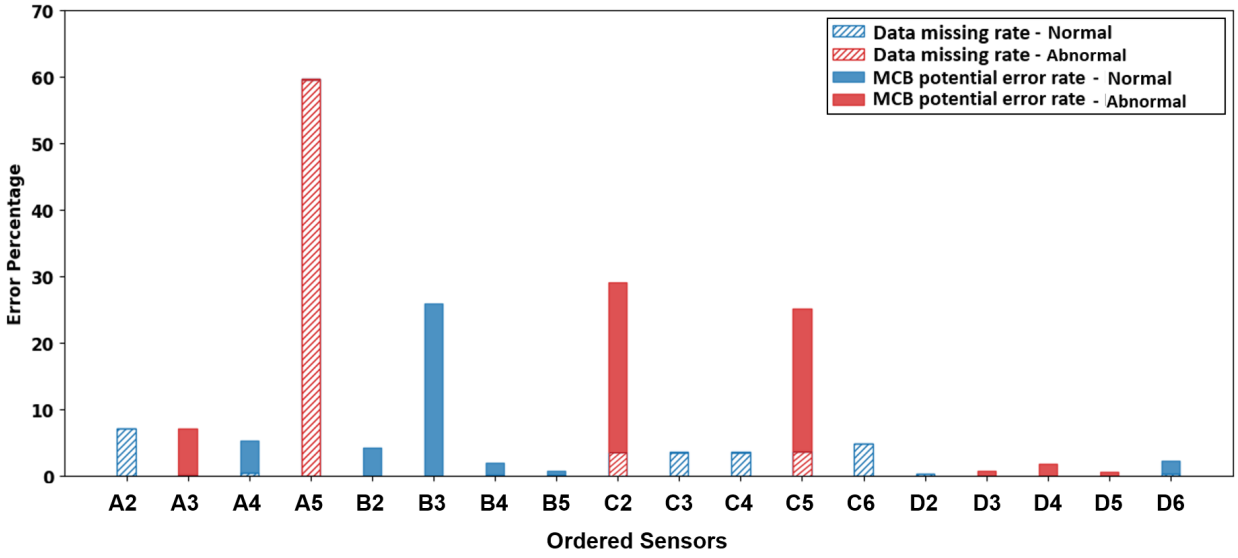


Figure 3.14: MCB algorithm based error percentages of sensors with labels from the proposed method

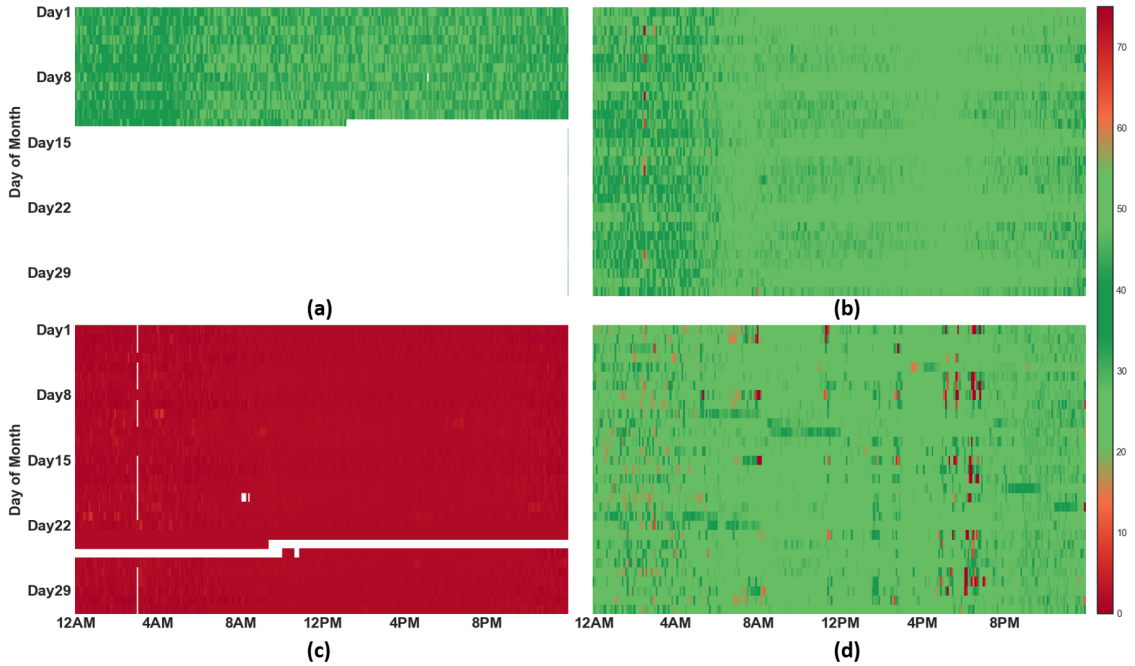


Figure 3.15: *AEVL* heatmap of (a) A5, (b) B3, (c) C2, and (d) D3

that have considerably higher missing data and potential error percentages as anomalous (under our assumed threshold of 20%, since Lu et al. (2014) did not propose any definite threshold for labelling anomalous sensors based on error percentages).

The greater efficacy of our proposed TSHM module compared to the MCB algorithm is additionally supported by Figure 3.15 raw $AEVL$ heatmap in that:

- Both the MCB algorithm and our proposed method detect as abnormal the $AEVL$ records for sensor A5, with its high missing data percentage observable in Figure 3.15a $AEVL$ heatmap, as well as sensors C2 and C5 (cf. Figure 3.15c), with their substantially different $AEVL$ both globally and with respect to their upstream (u/s) and downstream (d/s) sensors.

- The MCB algorithm labels sensor B3 as abnormal due to high potential error rate, but visual examination of B3's raw $AEVL$ heatmap in Figure 3.15b shows no substantial $AEVL$ abnormality. Further investigation reveals B3 and its nearest neighboring sensor B4 (2.1 miles away) to be located at two freeway interchanges where entering and exiting vehicles apparently affect vehicle composition and distribution (e.g., the B3 and B4 average hourly truck percentages were 9.34% and 16.12%, respectively). Therefore, the MCB method appears to be overly sensitive to vehicle composition varying between nearby locations.

- Our proposed TSHM module's step 3 temporal pattern anomaly test captures substantial temporal AEVL fluctuations (cf. Figure 3.15d raw AEVL plot for sensor D3), which led our method to report as abnormal the sensors D3, D4, and D5 that the MCB algorithm classifies as normal.

3.4 Conclusion

This study proposes a large-scale, data-driven traffic sensor health monitoring (TSHM) module involving massively parallelizable data processing techniques that make it feasible to deploy over large traffic networks. Our proposed TSHM module can be compared with sieving analysis, where each step identifies distinct sensor abnormalities, enabling traffic management authorities to take

the necessary steps to resolve. First, our module's data completeness test captures sensors with abnormally high missing data rates, providing a data completeness score (CS) that justifies assigning different levels of missing data severity. Second, reduced 2-d features from each sensor's aggregated $AEVL$ distribution are used to detect abnormal sensors based on DBSCAN clustering's anomaly detection logic. (We used the $AEVL$ metric since not only can it capture the variability of all three basic traffic variables (speed, density, and volume) simultaneously, but also it is a surrogate of vehicle length robust to daily or seasonal traffic variations and other external factors like inclement weather or traffic incidents.) Third, our novel temporal-pattern-based anomaly detection method utilizes the $AEVL$ assumption of constancy in the vehicle length distribution by introducing Bayesian changepoint detection in the temporal $AEVL$ matrix to detect sensors in the data stream reporting abnormally frequent spikes/fluctuations that suggest sensor problems requiring further attention.

One major challenge in abnormal sensor detection is the difficulty in obtaining groundtruth labels. Due to the absence of any explicit definition of abnormal sensors in the literature, this study has identified abnormal sensors by plotting sensor data along two different feature dimensions to identify points of agreement. For example, step 2 of our proposed TSHM module uses aggregated $AEVL$ records to detect abnormal sensors, but we also verify this cumulative feature vector's abnormality by comparing apparently abnormal sensors' CDF plots of with that of their adjacent u/s and d/s sensors. Similarly, we justify abnormal sensors identified by our step 3 temporal pattern anomaly test by demonstrating their raw $AEVL$ heatmaps also show frequent spikes. Finally, we compare our proposed module with two benchmark algorithms, the CLM and MCB. These baseline comparisons show our proposed method can successfully identify not only typical sensor error types, such as missing data or abnormal records, but also advanced error types such as frequent abrupt sensor data fluctuations. In addition, the efficacy of our proposed method is demonstrated in how, unlike the MCB algorithm, our method can successfully identify such abnormalities even for isolated or consecutive abnormal sensors.

However, our algorithm is an offline method that builds its model using historical traffic data. In future, our method can be extended to incorporate real-time sensor health monitoring to enable instant detection of abnormal sensors. Also, our proposed TSHM module's performance and reliability can likely be improved using traffic information from other sensor types (e.g., probe and camera data).

CHAPTER 4. DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS FOR TRAFFIC DATA IMPUTATION ENCODING TIME SERIES AS IMAGES

4.1 Introduction

Dissemination of accurate traffic data is an essential requirement for supporting advanced traffic management system operations. Different types of sensors, such as loop detectors, radar sensors, and video detectors, are installed in freeways and arterials for traffic data collection purposes. The data collected from these sensors can be used to detect traffic congestion or incidents (Chakraborty et al., 2018a,b, 2019), provide travel time information to road users (Lu et al., 2017; Gan et al., 2017), and support decision making at the traffic operation and planning levels (Shi and Abdel-Aty, 2015; Ma et al., 2017). However, missing data are quite common in traffic sensor data due to issues such as malfunctioning, poor maintenance or calibration, and intermittent communications (Lee and Coifman, 2011). According to the California Performance Measurement System (PeMS), only 67% of the sensors in District 7 of southern California (Los Angeles) were found to be working as expected in December 2018 (Armanious, 2019). While data from sensors with a high percentage of missing data can be discarded from further usage, an alternate approach is to impute the missing records, so that these sensors' data can still be used for subsequent analysis. This is particularly important for traffic-related studies that require traffic records to be complete, such as traffic flow analysis methods. Therefore, it is important to develop an effective traffic data imputation method which can handle missing traffic records even at a high percentage of missing data.

Traditionally, traffic data imputation has been done using prediction or interpolation methods that use historical traffic data or traffic data from adjacent sensors or time points to impute missing records (Nihan, 1997; Ghosh et al., 2007; Allison, 2001; Chang et al., 2012). However, these methods often fail to explicitly capture the spatio-temporal variations, which can lead to unreli-

able performance. Another class of imputation techniques relies on statistical learning models such as Markov chains or principal component analysis to learn the schema of the traffic data matrix (Lv et al., 2014; Ni and Leonard, 2005; Qu et al., 2009). However, these methods require the assumptions on the probability distribution of traffic data, which makes them difficult to apply in real-world scenarios. Also, these methods do not work well when handling large proportions of missing data, which is a common issue in real-world too. With the recent advancements in deep learning techniques and their success in image recognition and imputation tasks, traffic data imputation problem has been tackled using these new techniques that treat the data imputation problem as a corrupted data denoising problem (Duan et al., 2016; Ku et al., 2016; Asadi and Regan, 2019). However, modeling the strong time dependency of the time-series data is one of the major challenges in the application of these imputation techniques. To address this issue, we propose a novel Gramian Angular Summation Field (GASF) encoding method in this study to embed the traffic data for our model input, precisely preserving its time dependency. We then train a deep convolutional generative adversarial network (DCGAN) to generate realistic synthetic data for missing data imputation.

In recent years, deep learning based Generative Adversarial Networks (GANs) have been successful in generating impressively realistic synthetic data by modeling the real data distributions (Goodfellow et al., 2014, 2016). Further, by taking advantage of convolutional neural networks (CNNs), DCGANs (Radford et al., 2015) have shown remarkable ability in generating high quality synthetic image data for many applications such as image-to-image translation (Isola et al., 2017), audio generation (Donahue et al., 2018), and image super-resolution (Ledig et al., 2017). Such impressive performance in modeling the original data distribution has made DCGANs a strong candidate for data imputations (Yeh et al., 2017; Lee et al., 2019).

In this study, we have developed a traffic data imputation framework based on generative adversarial network (TSDIGAN) to efficiently resolve the missing data problem. Our proposed model treats the data imputation problem as a synthetic data generation problem. The novel GASF encoding method used in this study helps to embed the strong temporal dependency of the

time-series data, thereby translating the time-series imputation problem to an image imputation problem. We evaluate our proposed model using the benchmark PeMS dataset (PeMS, 2014) and compare its performance with other baseline statistical and deep learning models. We also investigate the capability of our proposed model for large-scale applications by clustering sensors into homogeneous groups and learning imputation models for each cluster of sensors. Thus, the major contributions of our study are as follows:

- Our proposed model takes advantage of deep learning based generative models, enabling users to treat the data imputation problem as a data generation problem. Such a generative framework can impute the missing data using the best-fitting generated realistic looking data, such that it is adaptive and robust in imputing the missing records, even at a high percentage of missing data.

- Our novel traffic time-series data-encoding technique using GASF method preserves the time dependency of traffic data without losing the underlying temporal dependency information. This proposed encoding method helps the model to learn the point-wise temporal relations between time-series traffic data.

- Our proposed model achieved reasonably high accuracy in imputation task for missing data ratios ranging from 5% to 90%, making it robust and reliable under challenging high missing data percentages. Additionally, training the proposed model using year-long traffic data takes less than 8 minutes, making it efficient and scalable for large-scale implementation. Therefore, we also evaluated our proposed model across extensive sensor groups of California District 5, showing the feasibility for large-scale practical applications. The proposed model has been found to improve the imputation accuracies in terms of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) compared to the state-of-the-art benchmark imputation models, while achieving comparable results in terms of Mean Relative Error (MRE).

The rest of this paper is organized as follows. Section 4.1 provides a brief description of related work regarding traffic missing data imputations, followed by the details of our methodology in Sec-

tion 4.2. Section 4.3 provides a detailed description of the data used in this study, results obtained using our proposed model compared to the baseline models. Finally, Section 4.4 summarizes the contributions of our study and its implications for future research.

4.2 Methodology

In this section, we provide the step by step details of our proposed TSDIGAN framework. We first explain the notation used in this study and then introduce the Gramian Angular Summation Field (GASF) time-series encoding and the basic concepts of GAN. The details of the proposed TS-DIGAN model framework is then discussed, followed by the large-scale implementation technique.

4.2.1 Notation

We first describe the abstract mathematical expressions used in this study. These notations will be used throughout the paper by default. Let us assume that the traffic flow time-series data for a given sensor s on a given day d is denoted by $X_d^s = \{x_1, x_2, x_3, ..., x_t, ..., x_T\}$, where x_t is the t^{th} observation of X. For 5-minute interval traffic flow data used in this study, T is given by $T = \{t_i\}_{i=1}^{288}$. The combined daily time-series data for each sensor s can then be represented as $\tilde{X}^s = \{X_d^s\}_{d=1}^{D}$, where D represents the total number of days involved. Finally, we use a binary corrupted mask as an indicator variable to flag whether the data for $X_{d,t}^s$ is present or missing. This leads to Equation 4.1.

$$I_{d,t}^s = \left\{ \begin{array}{l} 0, \text{ if } X_{d,t}^s \text{ is missing} \\ 1, \text{ if } X_{d,t}^s \text{ is not missing} \end{array} \right\} \tag{4.1}$$

$\tilde{X}^s$ can be divided into two subsets: (1) fully observed datasets, which do not have missing data points in any samples, denoted as $\tilde{X}^{s,f} = \{X_d^{s,f}\}_{d=1}^{D_1}$, and (2) corrupted datasets, which contain missing data points in each samples, denoted as $\tilde{X}^{s,m} = \{X_d^{s,m}\}_{d=1}^{D_2}$. For each sample, we flag whether the data points are missing or not using the corrupted mask $I_{d,t}^s$ which can also be divided into two subset matrices: $I_{d,t}^{s,f}$ (for fully observed datasets) and $I_{d,t}^{s,m}$ (for corrupted datasets). Next, we describe the first task in our proposed TSDIGAN framework: converting time-series traffic data

to GASF encoding, which enables treatment of the time-series imputation problem as an image imputation problem.

4.2.2 Gramian Angular Summation Field

Encoding time-series data as images have been widely used for time-series classification, audio data recognition, and similar other tasks. One of the popular approaches to tackle this problem is the spectrogram based method. For example, Cummins et al. (2017); Zhao et al. (2018) converted the speech/sound time series data into spectrogram using Short-time Fourier Transform (STFT) for recognition of emotional speech and locate image regions which produce sounds. Also, Lefebvre et al. (2017) estimated traffic flow by converting the spectrum features of the acoustic sensors signal data using Mel Frequency Cepstral Coefficients (MFCCs). However, such spectrogram based methods require careful parameter selection for precise inverse operation, and the imputation task of typical daily traffic volume data is unlikely to benefit from it (Wang and Oates, 2015). Another approach to encode the time series to images is to combine the spatial-temporal information as a 2D matrix. For instance, Zhuang et al. (2018); Kim et al. (2018) merged the ordered road segments/stations and time-series traffic data to form a 2D matrix that can be used by CNN to extract the spatial-temporal information. However, obtaining well-ordered sensor information in a complex, large-scale network is difficult and time-consuming. Further, the model needs to be retrained completely even when a single sensor is added or removed to the network.

To alleviate these issues, in this study, we adopted the Gramian Angular Summation Field (GASF), which has been demonstrated to improve CNN features extraction (Wang and Oates, 2015; Wang et al., 2017). The GASF has the following advantages. First, it helps to preserve and enhance the temporal correlations by considering the trigonometric sum between each time instance point. Second, the character of bijection provides directly and precisely inverse operation without the need for any specific parameter selection.

We rescaled the given preprocessed traffic flow time-series data $\tilde{X}^{s,f}$ to $[0, 1]$ such that we can represent the data in polar coordination system. Therefore, for daily traffic time-series data

$X_d^{s,f} = \{x_1, x_2, ..., x_t, ..., x_T\}$, the volume data x_t and time stamp t_i are encoded as angular cosine and radius (r) respectively, given by the following equation:

$$\begin{cases} \phi = \arccos(x_t), 0 \leq x_t \leq 1 \\ r = \frac{t_i}{C}, t_i \in \mathbb{N} \end{cases} \tag{4.2}$$

where, C is a constant regularization factor.

After transforming the traffic time-series data by using the above equation, the temporal correlation between each point can be identified by the GASF matrix denoted as $\hat{X}_d^{s,f}$:

$$\hat{X}_d^{s,f} = \begin{bmatrix} \cos(\phi_1 + \phi_1) & \cdots & \cos(\phi_1 + \phi_T) \\ \vdots & \ddots & \vdots \\ \cos(\phi_T + \phi_1) & \cdots & \cos(\phi_T + \phi_T) \end{bmatrix} \tag{4.3}$$

The main diagonal of $\hat{X}_d^{s,f}$ contains the original angular/value information. As mentioned above, the rescaled time-series data $X_d^{s,f}$ belongs to the set $[0, 1]$ so that the mapping between x_t and its corresponding angular cosine value is bijective. These characteristics allow us to precisely recover (inverse transform) the traffic time-series data from the GASF matrix $\hat{X}_d^{s,f}$ using the following equation:

$$X_d^{s,f} = \sqrt{\frac{\hat{X}_{d,diagonal}^{s,f} + 1}{2}} = \sqrt{\frac{\cos(2\phi) + 1}{2}}, \phi \in [0, \frac{\pi}{2}] \tag{4.4}$$

Here, we replace the suspect 0 value with 1 to avoid the zero division error and apply log transformation to avoid skewed distribution. Then, we rescale and transform the preprocessed 1-D daily traffic flow data $X_d^{s,f}$ into image-like GASF matrix $\hat{X}_d^{s,f}$ using Equations 4.2 and 4.3. The image-like matrix embedding with temporal dependencies helps convolutional neural networks (CNNs) to effectively extract the required features (LeCun et al., 1998). We used this image-like GASF matrix $\hat{X}_d^{s,f}$ as the input of our proposed GAN model. Finally, the daily traffic flow data $X_d^{s,f}$ can be recovered from the GASF matrix $\hat{X}_d^{s,f}$ using Equation 4.4.

4.2.3 Generative Adversarial Networks

Generative adversarial nets (GAN) were introduced as an effective tool for data augmentation and data generation. The basic GAN architecture is shown in Figure 4.1, which consists of two parts: a generator and a discriminator (Goodfellow et al., 2014). The generator (G) takes a random vector z as input, sampled from a noise distribution p_z, to output a corresponding synthetic data sample $G(z)$. The discriminator (D) takes a real sample x from the original dataset p_{data} and the synthetic sample $G(z)$ as inputs to estimate the probability that the generated and real sample comes from the same distribution. In our case, the original dataset p_{data} and the real sample x is given by $\tilde{X}_d^{s,f}$ and $\hat{X}_d^{s,f}$ respectively, which are obtained from GASF encoding. Both G and D usually consist of multi-layer perceptions (MLPs).

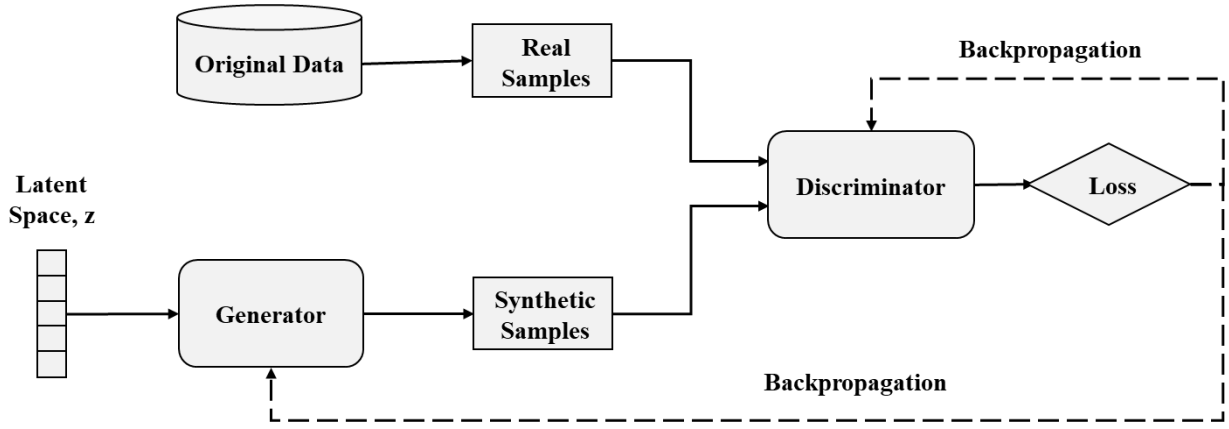


Figure 4.1: Architecture of GANs.

The discriminator and generator compete with each other like a two-player minimax game, where both the discriminator and the generator are trained simultaneously. During the training, the generator tries to fool the discriminator, while the discriminator tries to distinguish the real samples x from the synthetic samples $G(z)$ by solving the value function $V(G, D)$ formulated as:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{4.5}$$

After alternative training of both the discriminator and generator, the distribution of synthetic samples p_{syn} produced by the generator converges with the distribution of the original real data

p_{data}. In other words, the generator produces such realistic synthetic samples that the discriminator can no longer distinguish them from the original data samples.

4.2.4 TSDIGAN Architecture

In this subsection, we introduce the framework of our proposed model in details. The architecture of our proposed discriminator and generator is shown in Figure 4.2.

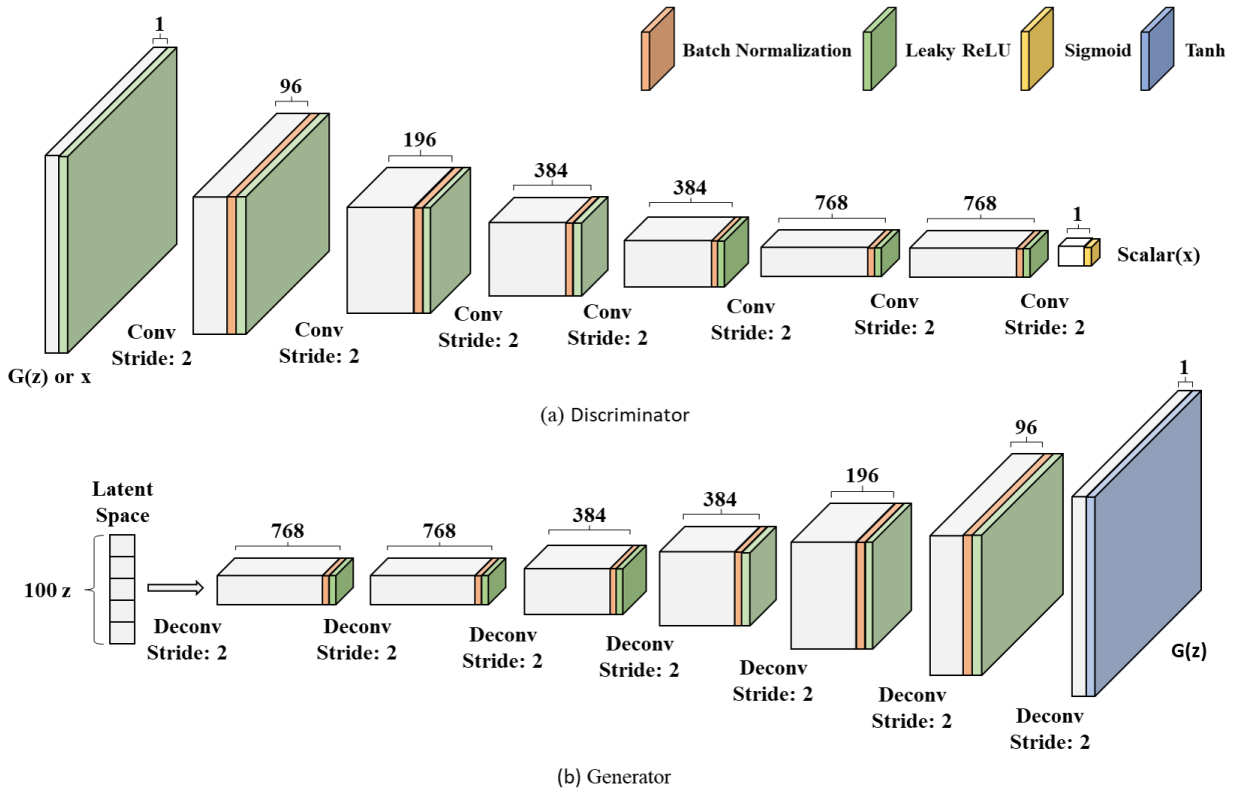


Figure 4.2: Proposed TSDIGAN (a) Discriminator and (b) Generator.

In the discriminator module, we used strided convolution (expressed as "Conv Stride: 2" in Figure 4.2a). In contrast, for the generator module, we used fractional-strided convolutions (expressed as "Deconv Stride: 2" in Figure 4.2b). We then applied Batch Normalization (BatchNorm) layers to stabilize the learning process and prevent model collapse. BatchNorm was used for all layers, except the input layer of the discriminator and the output layer of the generator to avoid sample

oscillation. We used LeakyReLU as the activation function for all layers to provide non-linearity, except in the output layers of both the discriminator and generator, where we used the sigmoid and tanh functions as the activation functions to produce the scalar and synthetic data respectively. Moreover, the latent random vector z was sampled from the normal distribution.

To help the CNN model learn more effectively, we repeated the first and last traffic flow data points at the head and tail of the $X_d^{s,f}$ three times instead of doing the "zero-padding". Therefore, the dimensions of the input traffic data vector transformed to $294 = (3 + 288 + 3)$, with the GASF matrix $\hat{X}_d^{s,f}$dimensions being 294×294. This same padding value was removed after training. Additionally, we applied a Gaussian filter on the $\hat{X}_d^{s,f}$ to reduce the inherent noise and improve the quality of the GAN-generated synthetic data (Susmelj et al., 2017). The initial learning rate was set to 0.0002 along with the Adam optimizer. GANs are however known to suffer from mode collapse issues frequently, when the training model often sticks to only few modes of the true distribution ignoring the other modes (Radford et al., 2015). To prevent the potential mode collapse in this study, we randomly assigned the training label from 0.8 to 1.1 and 0.0 to 0.3 for positive and negative labels respectively. Also, we randomly flipped 10% of the training labels in each mini-batch (Salimans et al., 2016). These strategies helped to prevent the discriminator/generator from trapping into state of extremely high confidence and stabilize the training process during our experiments.

4.2.4.1 Maximum Mean Discrepancy

After training our TSDIGAN model, the generator can generate synthetic daily traffic flow dataset $\tilde{X}^{s,syn} = \{X_d^{s,syn}\}_{d=1}^{D_3}$ that looks "close" to the real data $X_d^{s,f}$ sampled from the original fully observed dataset $\tilde{X}^{s,f}$. During the training phase, the generator generates the same number of samples as used for training. Therefore, in this case $D1$ is obviously equal to $D3$. In general, a well-trained GAN can implicitly learn the distribution of the original fully observed dataset $\tilde{X}^{s,f}$. Visual inspection of synthetic traffic flow data is one recommended means of determining if a TSDIGAN

model is well-trained, which is also considered as the intuitive way to inspect GANs based models (Borji, 2019).

In addition, we utilized the maximum mean discrepancy (MMD) method as the quantifiable tool to measure the similarity between the two distributions $\tilde{X}^{s,f}$ and $\tilde{X}^{s,syn}$ (Gretton et al., 2007) using the following equation:

$$\widehat{MMD}_u = \left[\frac{1}{{D_1}^2} \sum_{i,j=1}^{D_1} k(\tilde{X}_i^{s,f}, \tilde{X}_j^{s,f}) - \frac{2}{D_1 D_3} \sum_{i,j=1}^{D_1,D_3} k(\tilde{X}_i^{s,f}, \tilde{X}_j^{s,syn}) + \frac{1}{{D_3}^2} \sum_{i,j=1}^{D_3} k(\tilde{X}_i^{s,syn}, \tilde{X}_j^{s,syn}) \right]^{\frac{1}{2}} \tag{4.6}$$

Here, $k(\tilde{X}_i^f, \tilde{X}_j^{syn})$ represent the kernel function, and we used the radial basis function (RBF) kernel for MMD score calculation as described in Esteban et al. (2017). We recommend interested readers refer to Esteban et al. (2017) for more details. Figure 4.3 shows the sample MMD score trace over 60 training epochs. Therefore, training of the proposed TSDIGAN was verified not only through visual inspection of its synthetic traffic flow data, but also by observing stable convergence of the MMD score.

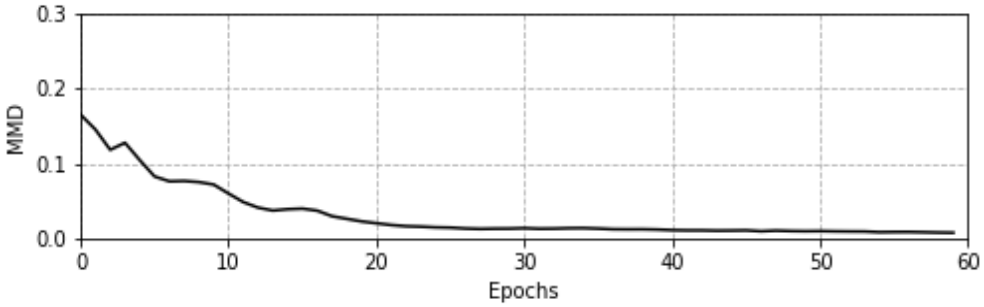


Figure 4.3: MMD Score Trace

4.2.5 TSDIGAN Imputation Model

After training our TSDIGAN using the fully observed dataset $\tilde{X}^{s,f}$ as described in Section 4.2.4, we used our trained model for missing traffic data imputation. In this subsection, we introduce the imputation (or inpainting) framework shown in Figure 4.4. The basic idea of our imputation framework is similar to GAN based image inpainting (Yeh et al., 2017; Yu et al., 2018) in that we

searched the most representative z from p_z as the input for the generator to use in generating a realistic synthetic data for each specific $X_d^{s,m}$.

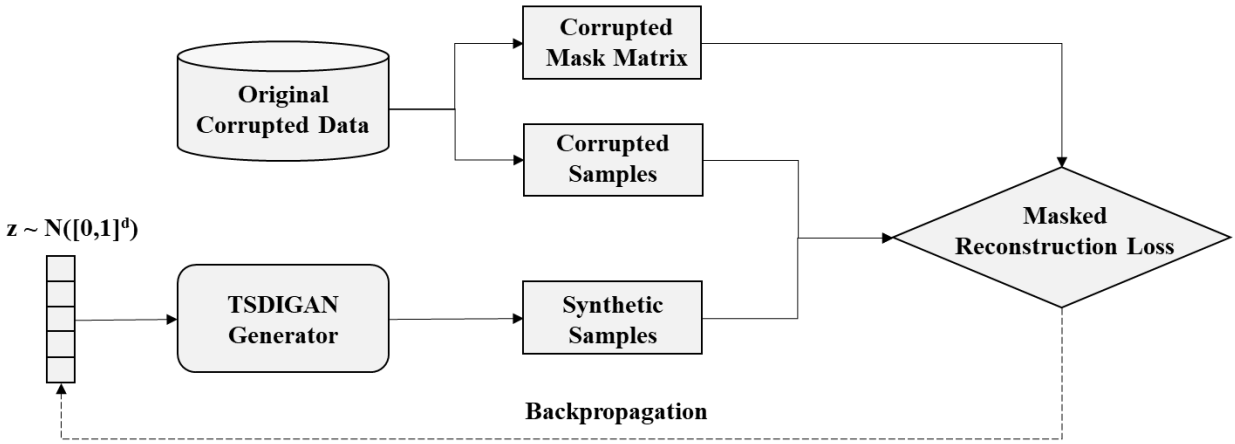


Figure 4.4: Architecture of TSDIGAN Imputation Model.

From the basic concept of the GAN, we know that the generator learns the distribution from original fully observed dataset $\tilde{X}^{s,f}$, and generate the synthetic dataset $\tilde{X}^{s,syn}$ without any missing points from the learned distributions. We could thus use the synthetic data sample $X_d^{s,syn}$ to fill in the missing points in $X_d^{s,m}$ via the corrupted mask vector $I_{d,t}^{s,m}$. However, for the given incomplete data vector $X_d^{s,m}$, which z should we use to generate the most reasonable data? The question above can be described as searching for the "closest" z_c from p_z to generate missing values constituting the best "overlay" synthetic traffic flow data $X_d^{s,syn}$, based on the observed part of the given data vector $X_d^{s,m}$. To accomplish this goal and inspired by Yeh et al. (2017); Luo et al. (2018), we designed the masked reconstruction loss denoted as ℓ_r. For any given $X_d^{s,m}$, the masked reconstruction loss can be formulated as:

$$\ell_r(z|X_d^{s,m}) = \frac{\left\|X_d^{s,m} \odot I_d^{s,m} - X_d^{s,syn} \odot I_d^{s,m}\right\|_1}{\sum_{t=1}^{T} I_{d,t}^{s,m}} \tag{4.7}$$

It should be noted that by multiplying the masked missing vector $I_d^{s,m}$, only the observed part of $X_d^{s,m}$ is used for calculating the masked reconstruction loss, and the $\odot$ represents element-wise multiplication. Thus, the "closest" latent vector z_c can be represented as:

$$z_c = \arg\min_z \ell_r(z|X_d^{s,m}) \tag{4.8}$$

Finally, we used the synthetic traffic flow data $X_d^{s,syn}$ generated from the "closest" z_c to fill in the missing part of $X_d^{s,m}$. We summarize this imputation module step by step as shown in Algorithm 1.

.

Require: (1) Corrupted traffic data vector $X_d^{s,m} = \{x_1, x_2, x_3, ..., x_t, ..., x_T\}$ which need imputation, (2) Corrupted missing vector $I_d^{s,m}$, (3) number of iterations w for back-propagation on latent space z, (4) learning rate α and (5) trained generator.

1: Initialize $z \sim N([0, 1])$ as the input for generator.

2: Fix the weights of trained generator and active the gradient descent on z

3: **for** w iterations **do**

4: Generate GASF matrix from the z, and extract the synthetic traffic time-series data using Equation 4.4:

$$\hat{X}_d^{s,syn} = G(z)$$

$$X_d^{s,syn} = \sqrt{\frac{diag(\hat{X}_d^{s,syn}) + 1}{2}}$$

5: Calculate masked reconstruction loss, and apply Back-propagation on z:

$$\ell_r(z|X_d^{s,m}) = \frac{\left\| X_d^{s,m} \odot I_d^{s,m} - X_d^{s,syn} \odot I_d^{s,m} \right\|_1}{\sum_{t=1}^{T} I_{d,t}^{s,m}}$$

$$z \leftarrow z - \alpha \times \nabla \ell_r(z|X_d^m)$$

6: Obtain the "closest" z_c by:

$$z_c = \arg\min_z \ell_r(z|X_d^{s,m})$$

7: **end for**

8: Impute the $X_d^{s,m}$ using the $X_d^{s,syn}$ generated from z_c:

$$X_d^{s,\text{imputed}} = X_d^{s,syn} \odot (1 - I_d^{s,m}) + X_d^{s,m} \odot I_d^{s,m}$$

Algorithm 1 Traffic data imputation module using TSDIGAN

4.2.6 Large-Scale Implementation

Deep learning models are promising for the traffic data imputation task; however, their practical applications on large-scale statewide level requires further investigation. In this subsection, we investigate the capability of our proposed TSDIGAN model for large-scale real world application. Typically, traffic sensors over extensive wide coverage involves a wide variation of traffic data characteristics. Grouping all the sensors together in a single cluster can make the model training significantly harder due to multiple modes present in the data generated from the distinct variations in traffic data generated across the different sensors. Further, this can lead to model instability and mode collapse, thereby making the training process significantly difficult and lead to poor model performance. On the other hand, training a model individually for each sensor leads to a significantly larger number of models training and maintenance/updates, making it difficult to large-scale application. For example, in the recent state-of-the-art traffic data imputation study by Chen et al. (2019) using parallel data based GAN model, a total of 294 models across the 147 districtwide sensors were developed for weekday and non weekdays. In this study, we chose a middle ground where we group the sensors based on their inherent traffic data characteristics such that models generated for each cluster can be focused towards the cluster-specific traffic variation characteristics. More specifically, we use k-means clustering (MacQueen et al., 1967; Berkhin, 2006) to group the sensors based on their daily traffic flow patterns because of it's simplified approach and computation efficiency.

Let us assume, we have S sensors in the traffic sensor networks. As mentioned in Section 4.2.1, the fully observed data provided by a given sensor s is denoted as $\tilde{X}^{s,f} = \{X_d^{s,f}\}_{d=1}^{D_1}$. Therefore, the traffic flow values for each sensor over a given set of days (D_1) can be denoted as a matrix with the dimensions $D_1 \times T$. At each time instance t, we extracted the features of this $D_1 \times T$ matrix by taking the quantiles q value along the D, and augmenting them into one long feature vector $X_{feature}^{s,f}$ with the shape 1×1440 as:

$$X_{feature}^{s,f} = \Big[q(\tilde{X}^{s,f},10),\ q(\tilde{X}^{s,f},30),\ q(\tilde{X}^{s,f},50),\ q(\tilde{X}^{s,f},70),\ q(\tilde{X}^{s,f},90)\Big] \tag{4.9}$$

We represent long feature vectors for all the sensors with $\tilde{X}^{s,f}_{feature} = \{X^{s,f}_{feature}\}^{S}_{s=1}$, and used the k-mean and elbow method (Kodinariya and Makwana, 2013) to divide the sensors into different groups. This sensor clustering procedure is summarized in Algorithm 2. We then trained our proposed TSDIGAN model for each group separately. This enabled us to simply identify which group any sensor belonged to, and use its corresponding trained model to produce appropriate synthetic data for imputation.

.

Input: Feature vectors for all the sensors $\tilde{X}^{s,f}_{feature} = \{X^{1,f}_{feature}, X^{2,f}_{feature}, ..., X^{s,f}_{feature}, ..., X^{S,f}_{feature}\}$

Output: Sensor groups

1: **for** i=1 to S **do**
2: Initialize: K=i, K clustering centroids $\mu_1, \mu_2 \in \mathbb{R}^{1440}$
3: **repeat**
4: Assign each feature vector to clusters based on the closest Euclidean norm.
5: Update the position of the centroids based on their mean distances to assigned points.
6: **until** Clustering converged
7: **end for**
8: Obtain the optimal K denoted as K_c using Elbow method.
9: Initialize: K_c clustering centroids $\mu_1, \mu_2 \in \mathbb{R}^{1440}$
10: **repeat**
11: Assign each feature vector to clusters.
12: Update the positions of the centroids.
13: **until** Clustering converged

Algorithm 2 k-means sensors clustering

4.3 Results

In this section, we evaluate our proposed model using traffic flow data obtained from the Caltrans Performance Management System (PeMS) (PeMS, 2014). We first evaluate the imputation performance for a single sample sensor followed by large-scale districtwide sensors. Then, we show the efficiency of our proposed model by comparing it with other benchmark baseline models, namely support vector regression (SVR), history average (HA), denoising stacked autoencoder (DSAE), and GAN based parallel data model (Chen et al., 2019).

4.3.1 Data Description

The Caltrans PeMS dataset used in this study, is one the most popular open source dataset for transportation research, consisting of more than 15,000 vehicle detector stations (VDSs) or sensors covering over the entire state of California. In this study, we used the 5-minute traffic flow data provided by the PeMS data warehouse for the year 2013 from District 5: Central Coast. There were 147 VDSs in this district, each of which had 363 days' worth of traffic flow data vectors, while no data was present for the remaining two days of the year. Hence, each individual VDS had 104,544 ($363 \times 24 \times 12$) traffic records. We divided the weekday data vectors and non-weekday data vectors following the work proposed by Duan et al. (2016) and Chen et al. (2019). This resulted in 245 days labeled as weekdays and 118 days labeled as non-weekdays for each individual VDS. It should be noted that our dataset is exactly the same dataset used in the Duan et al. (2016) and Chen et al. (2019) study for DSAE model and GAN based parallel data model respectively. This enabled us to directly compare the performance of our model with these benchmark models.

4.3.2 Evaluation Criteria

In order to evaluate the performance of our TSDIGAN model, we utilized three criteria: mean absolute error (MAE), root mean square error (RMSE), and mean relative error (MRE), given by the following equations:

$$MAE = \frac{\sum_{d=1}^{D_{test}} \sum_{t=1}^{T} I_{d,t}^{s} |y_{d,t} - \hat{y}_{d,t}|}{\sum_{d=1}^{D_{test}} \sum_{t=1}^{T} I_{d,t}^{s}} \tag{4.10}$$

$$RMSE = \sqrt{\frac{\sum_{d=1}^{D_{test}} \sum_{t=1}^{T} I_{d,t}^{s} (y_{d,t} - \hat{y}_{d,t})^2}{\sum_{d=1}^{D_{test}} \sum_{t=1}^{T} I_{d,t}^{s}}} \tag{4.11}$$

$$MRE = \frac{\sum_{d=1}^{D_{test}} \sum_{t=1}^{T} I_{d,t}^{s} \frac{|y_{d,t} - \hat{y}_{d,t}|}{y_{d,t}}}{\sum_{d=1}^{D_{test}} \sum_{t=1}^{T} I_{d,t}^{s}} \tag{4.12}$$

where, $y_{d,t}$ is the observed traffic flow data (groundtruth), while $\hat{y}_{d,t}$ is the imputed traffic flow data obtained using the proposed model. D_{test} is the total number of daily traffic flow vectors used for testing, T is the dimension of each traffic flow vector (equal to 288), and $I_{d,t}^{s}$ is the corrupted mask mentioned in Section 4.2.1.

To fairly evaluate our model's performance throughout the next steps of our study, we randomly corrupted the observed data with various random missing rates (MR), and distributed the missing data points equally for each test sample. The random missing rate (MR) can be defined as:

$$MR = \frac{\sum_{d=1}^{D_{test}} \sum_{t=1}^{T} I_{d,t}^{s}}{D_{test} T} \times 100\% \tag{4.13}$$

For the convenience of evaluation and comparison in the following sections, we used a single default ratio of 4:1 to split the training and test samples for each individual sensor. Therefore, 245 weekdays were split into 196 days for training and 49 days for testing, while the remaining 118 non-weekday patterns were split into 94 days for training and 24 days for testing. And for each MR, we repeated the experiments 25 times and took the average to ensure unbiased and reliable results. Next, we describe our model performance on a single sample sensor.

4.3.3 Single VDS Performance

As mentioned in Section 4.2, we trained our proposed model using fully observed training samples. Figure 4.5 shows the training process of the proposed model across different epochs along with the real observed data. The generator tends to learn to create realistic-looking GASF matrix images step by step. After 50 epochs, the generator was able to produce synthetic GASF images

that are similar to the real samples, as shown in the left-most sub-figure of Figure 4.5. We then used the imputation module described in Section 4.2.5 to impute our corrupted test samples from the optimal latent space. Here, we used the VDS 500010102 as our target VDS for demonstration, which is the same used in Duan et al. (2016). As mentioned in Section 4.3.1, we trained separate models for weekdays and non-weekdays. Therefore, we had 196/49 traffic flow data vectors for training/testing for weekdays, and 94/24 traffic flow data vectors for training/testing for non-weekdays. We then stacked our imputed result vectors $\hat{y}_{d,t}$ from both weekdays and non-weekdays together to evaluate the overall accuracy using the criteria equations mentioned in Section 4.3.2. This setup was used by default for all later experiments.

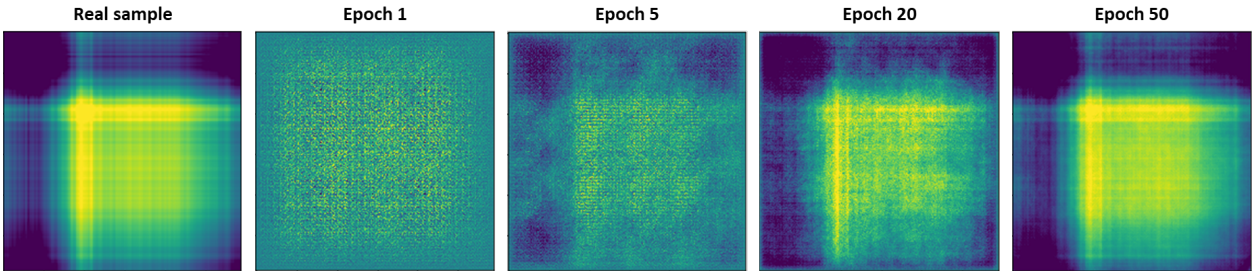


Figure 4.5: Sample results of GASF encoded images training using the proposed GAN generator model.

We conducted experiments to test the efficiency and robustness of our proposed model for MRs ranging from 5% to 90%. The results obtained for the sample sensor is shown in Figure 4.6. MAE was found to vary between 8.8 to 10.4 vehicles per 5 minutes (veh/5-mins), $RMSE$ from 13.3 to 15.4 veh/5-mins, while MRE ranged from 19.3% to 21.7%. As it can be seen from the figure, while the error trace increases with increase in MR, however, our proposed approach was able to perform reasonably well even in very high MR ($\geq$ 50%). Even under MR as high as 80%, our proposed model was still able to obtain decent imputation results with an MAE of less than 10.0 veh/5-mins and MRE of less than 21.0%. This shows that the proposed model is robust to high missing data percentages too.

To extend this further, Figure 4.7 shows the absolute error (AE) distribution and the relative error (RE) distributions for 10% and 90% MRs. This can help to understand the AE and RE variation range within two extreme MRs (10% and 90%). It can be observed that about 60% of AE was less than 8 veh/5-mins at 10% MR, while it is less than 9 veh/5-mins at 90% MR. Similarly, there is about 60% of RE less than 13.5% at 10% MR, while it is less than 15.5% at 90% MR. By taking advantage of the generative model and temporal dependency correlation GASF matrix, our proposed model can produce robust and reliable imputation results even for large variation of MRs. A sample weekday and non-weekday imputation results is shown in Figure 4.9a and 4.9b, respectively. The figure shows the imputed data obtained using the proposed model along with the corresponding actual "observed" data and corrupted data too. It can be seen that the sample synthetic traffic flow data produced by our proposed model perform reliably well in successfully overlaying the observed part, with its "closest" estimation of the corrupted data points.

To illustrate the efficiency of our proposed model in learning the traffic data distribution, we plot the traffic flow histograms generated using the real data and the synthetic data obtained from the model for 20% MR, as shown in Figure 4.10a. Further, Figure 4.10b shows the empirical distributions of the deviation time series of the real data and the synthetically generated data, similar to Chen et al. (2012); Li et al. (2013). The deviations are calculated as difference between simple average intra-day trend from the original and imputed data. This helps to check if the imputed data preserve the important statistical features of the original dataset. As it can be seen from Figure 4.10(a) and (b), the synthetic data distributions closely follow the original data distributions, thereby verifying that the proposed imputation technique has been able to retain the original data features successfully. In the next section, we discuss the details of our proposed model performance in a large-scale implementation over the entire District 5 of California instead of a single sensor imputation.

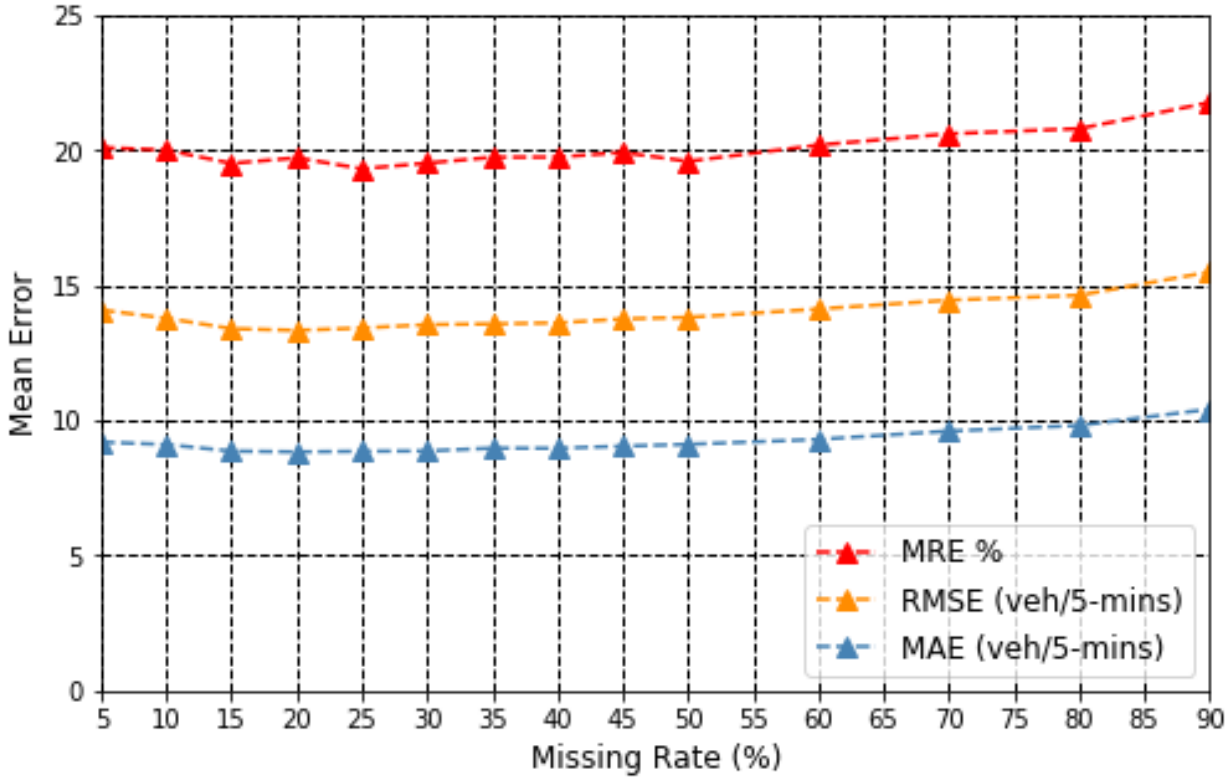


Figure 4.6: Imputation performance for a single sample sensors for different missing rates.

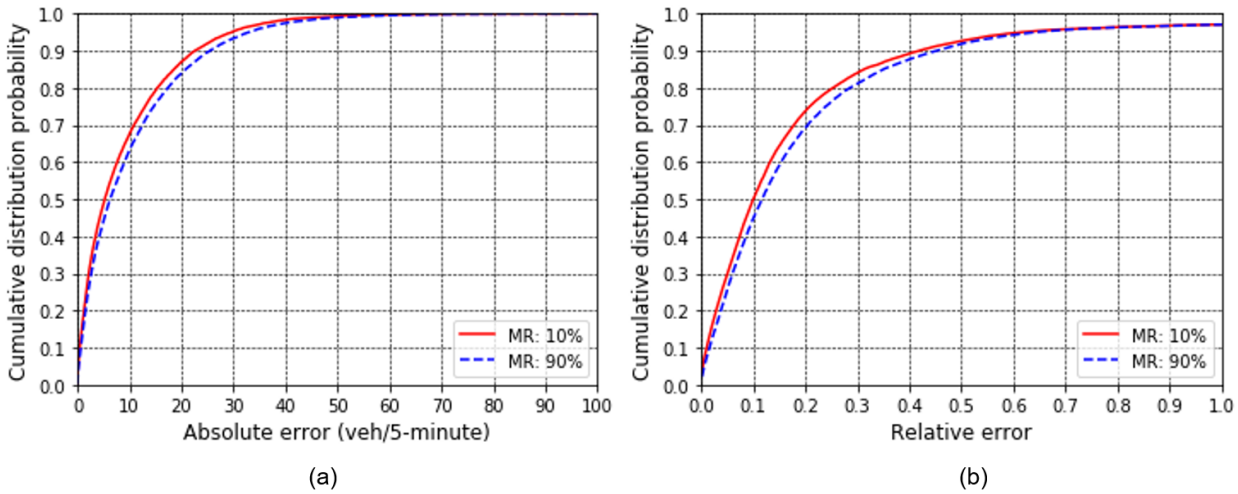
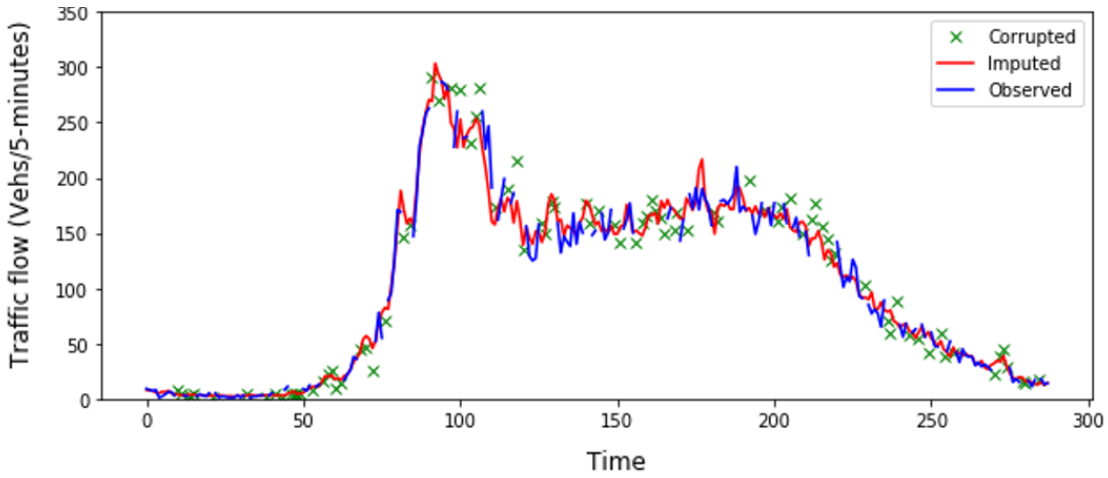
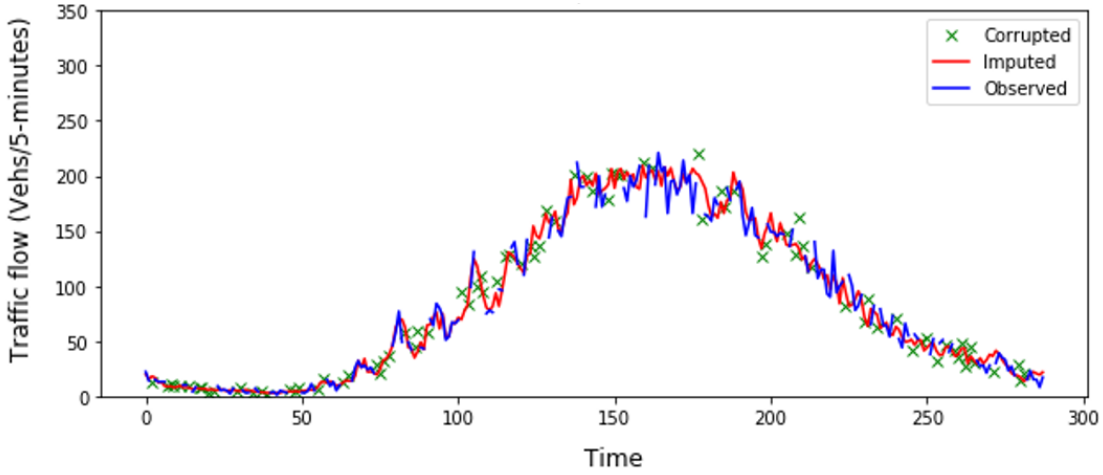


Figure 4.7: Error distributions for 10% and 90% marginal missing rates: (a) absolute error and (b) relative error.



(a)



(b)

Figure 4.9: Sample synthetic traffic flow data plot: (a) Weekday and (b) Non-weekday.

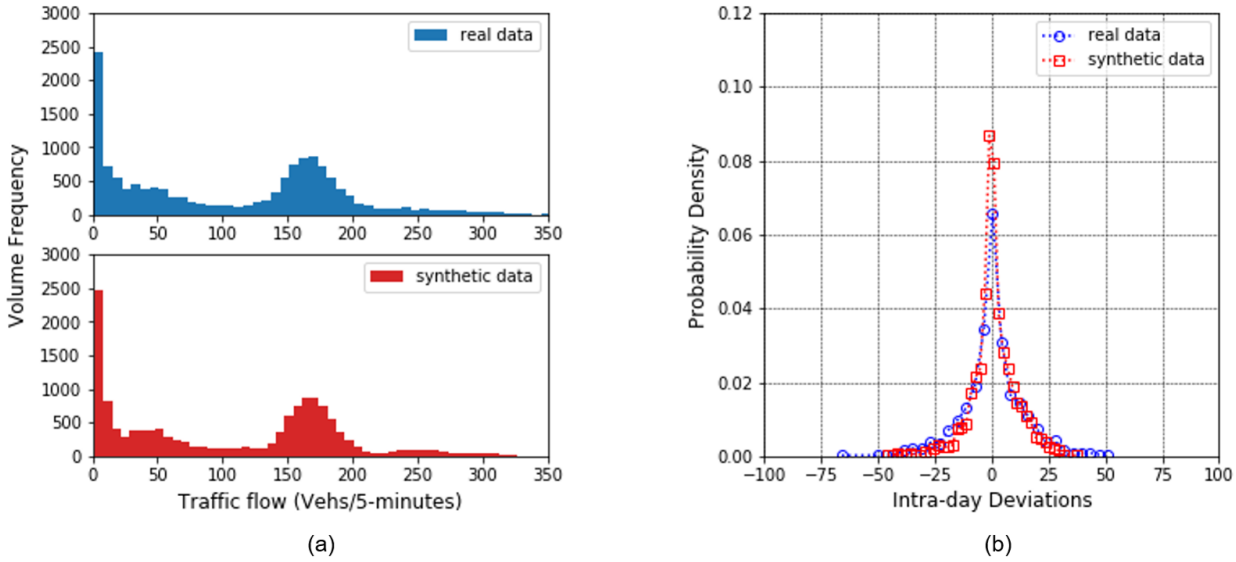


Figure 4.10: (a) Traffic flow histograms and (b) deviation distribution between the real test data and synthetic data for 20% MR .

4.3.4 Imputation Performance on a Large-Scale Network

In this subsection, we evaluate our proposed model using the entire 2013 data obtained from all VDSs of District 5 of California. As mentioned in Section 4.2.6, we first divided the 147 VDSs into different homogeneous groups based on their daily traffic flow patterns using the k-means and elbow method. As shown in Figure 4.11, we draw the sum of squared distances versus the possible number of clusters, and used the elbow method to determine the optimal number of clusters (K_c) (Kodinariya and Makwana, 2013). The optimal number of clusters were found to be 25 using the elbow point. However, the selection of the optimal groups can also be chosen based on the agency specific requirements.

To demonstrate the different daily patterns observed in the generated clusters, we plot the median daily traffic flow data of 5 sample clusters in Figure 4.15. The average daily traffic (ADT) for these 5 sample groups varied between 9,000 to 64,000 vehicles. It can be seen that there were both morning and evening peaks for groups 4 and 5, while groups 2 and 3 have either a morning peak or an evening peak. It can also be seen that group 1 had the lowest daily traffic flow. Therefore,

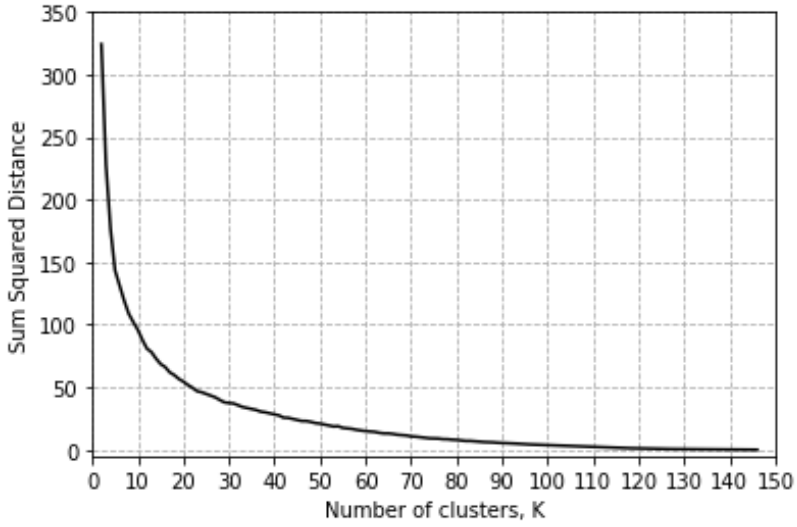


Figure 4.11: Elbow plot to determine optimal number of clusters for VDSs

the clusters generated having distinct traffic flow patterns help the model to learn those unique patterns and perform more efficiently in large-scale network.

In addition, to ensure fair and balanced results for each individual VDS, we selected 20% of the samples equally from each VDS for testing and used the remaining 80% for training. Figure 4.16 shows the performance of our proposed model for all VDS in a sample group 2 for MR ranging between 10% to 80%. It can be seen that MAE for all VDSs ranging from 9.1 to 10.6 veh/5-mins, $RMSE$ ranging from 12.9 to 15.3 veh/5-mins, and MRE ranging from 17.1% to 23.6%. This shows that the proposed model scales efficiently for larger number of VDSs, in addition to the single VDS model described in Section 4.3.3.

Table 4.1 summarizes the overall performance of our proposed TSDIGAN model for the 5 sample clusters, whose daily pattern is shown in Figure 4.15. It can be seen that the model performed reasonably well even in high MR of 80%, showing the efficacy of the model. Further, although the MAE and $RMSE$ increased for clusters with higher ADT, the MRE showed decreasing trend with increase in ADT, thereby showing it's performance is robust for different ranges of ADT too.

Figure 4.17 presents the imputation performance accuracies in terms of MAE, $RMSE$, and MRE for all 147 VDSs of the entire District 5 of California at 30% MR. For all VDSs, MAE

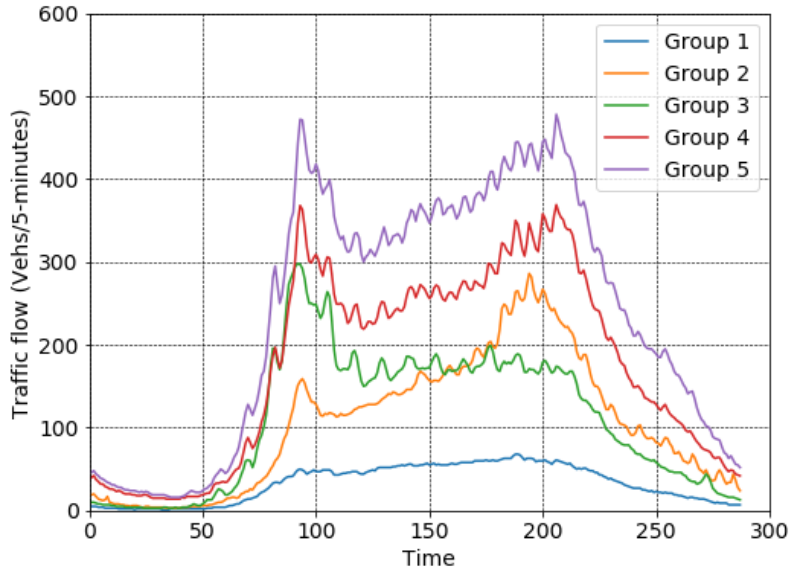


Figure 4.13 Weekday

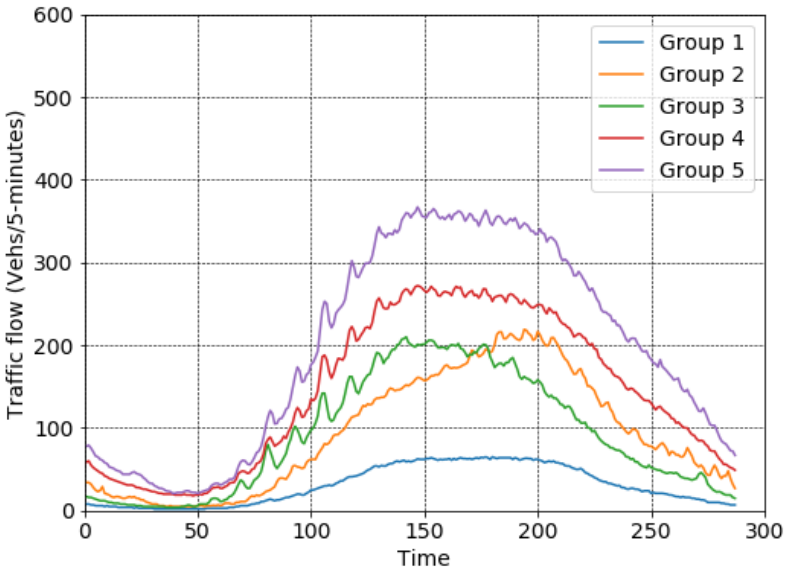


Figure 4.14 Nonweekday

Figure 4.15: Median traffic flow patterns for (a) weekdays and (b) non-weekdays for 5 sample VDS clusters

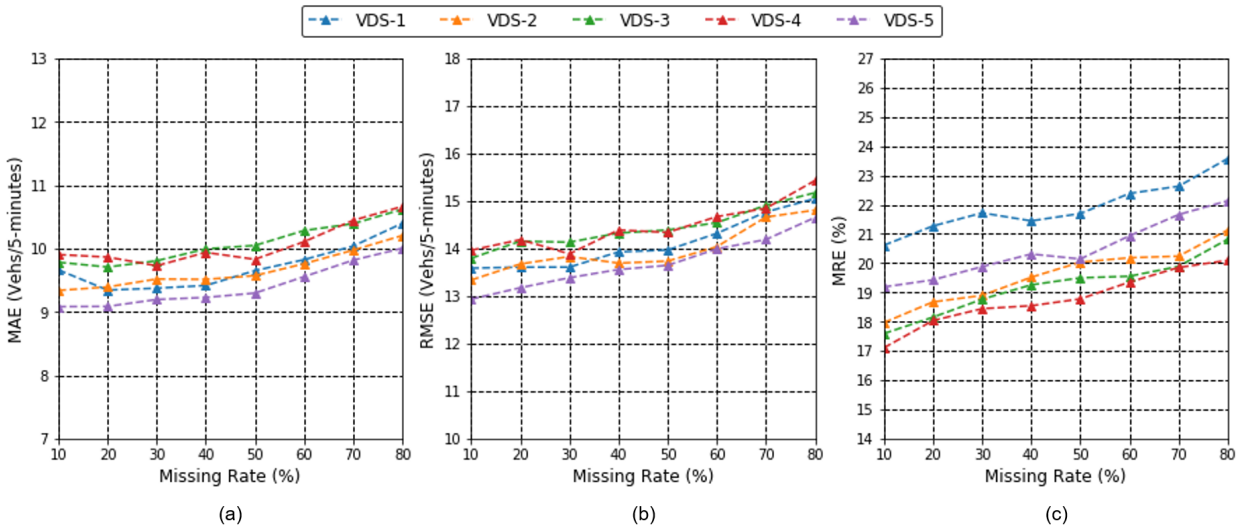


Figure 4.16: Imputation performance in terms of (a) MAE, (b) RMSE, and (c) MRE for all VDS in Cluster ID 2

was found to vary between 7.61 to 31.55 veh/5-mins with the median and mean values of 12.64 and 13.16 veh/5-mins respectively. Similarly, $RMSE$ varied between 10.91 to 52.9 veh/5-mins with the median and mean of 18.98 and 20.22 veh/5-mins respectively. Therefore, the proposed model performed reasonably well across large-scale sensor networks in terms of MAE and $RMSE$. However, MRE was found to vary between 12.4% to 469% with the median and mean values of 20.7% and 35.5%. Therefore, the MRE performance for the proposed model was found to be highly skewed, with exceptionally high MRE for VDS ID 126 and 127. On further investigation, it was observed that the input of raw training data of these sensors had more inherent noise and zero volume report which made learning of the sufficient representative "pattern" from such training data highly unstable. However, the median MRE across all sensors were found to be only 20.7%, suggesting reasonable overall performance across the sensors. These results indicates that the efficiency of our proposed TSDIGAN framework across districtwide sensor networks, thereby showing it's feasibility for large-scale practical implementations.

Additionally, efficiency of real-world implementation of the proposed model depends on the training and testing cost and time requirements. This is particularly important since training deep

Table 4.1: Performance summary for VDSs of 5 sample clusters at 20%, 50%, and 80% MR

Criteria		MAE (veh/5-min)			RMSE (veh/5-min)			MRE		
Group	ADT	20%	50%	80%	20%	50%	80%	20%	50%	80%
1	9000	5.0	5.2	5.6	7.2	7.5	8.1	0.318	0.327	0.350
2	29000	9.5	9.7	10.4	13.8	14.0	15.0	0.191	0.200	0.216
3	30000	9.8	10.0	10.6	14.5	14.9	15.5	0.200	0.210	0.221
4	47000	13.9	14.1	15.2	19.8	20.4	21.7	0.119	0.124	0.133
5	64000	14.6	14.9	16.4	20.7	21.4	22.9	0.103	0.108	0.116

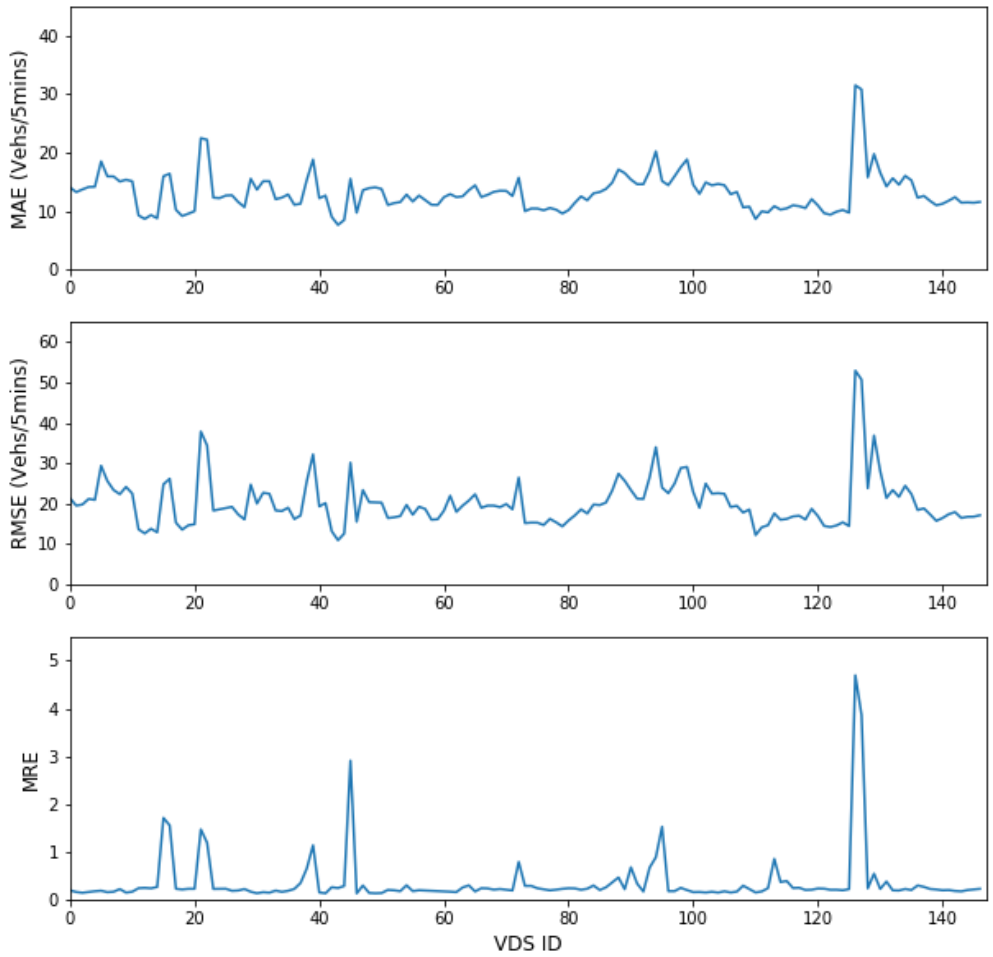


Figure 4.17: Imputation performance accuracies (MAE, $RMSE$, and MRE) at 30% MR for all VDSs

learning models is time-consuming and GANs in particular are well-known to suffer from vanishing gradients, mode collapse, and failure to convergence. Our proposed model was trained and tested using a single NVIDIA GTX 1080Ti GPU along with Intel(R) i7-8700 CPU, 32 GB RAM, and Windows 10 (64 bits) platform. All the frameworks used in this study were built using PyTorch 1.1 (Paszke et al., 2017). Our proposed model took approximately 8 minutes training time for a single VDS and 14 hours for the entire districtwide 147 VDSs using one year of historical traffic flow data for 50 epochs. The average training time for each group of VDS obtained using clustering method is found to be around 30 minutes. To find the optimal latent space using the imputation module during the test phase, the time taken varies depending on the iterations. In our experiments, we performed 200 iterations which was found to take approximately 2 seconds for generating daily traffic data generation for a single VDS. Therefore, it takes approximately 5 minutes for districtwide daily traffic data imputation across 147 VDS. This shows that the proposed model can be successfully applied to large-scale real-world implementation scenarios with the desired regular offline model retrain/update. Next, we present the results on comparison of our proposed model with other benchmark data imputation models.

4.3.5 Model Comparison

In this section, we compare the performance of our proposed TSDIGAN model with other benchmark traffic data imputation models to find out the efficiency of our proposed model. The benchmark models used in this study for comparison are support vector regression (SVR), history average (HA), denoising stacked autoencoder (DSAE), and GAN based parallel data model (Chen et al., 2019). It should be noted that the benchmark dataset used in our study was also used by Duan et al. (2016) for DSAE model and Chen et al. (2019) for GAN based parallel data model. This enabled us to directly compare the performance of our model with these benchmark models. The detailed default settings for model training and evaluation results for baseline models can be found in Chen et al. (2019). Figure 4.18 shows the average imputation performance accuracies across all VDS in terms of MAE, $RMSE$, and MRE for all comparison models. It can be seen

that our proposed model outperformed all other benchmark models in terms of MAE and $RMSE$. An overall improvement of 13.7 % and 16.3 % was observed by TSDIGAN model compared to the next best performing parallel data model. However, in terms of MRE, the proposed TSDIGAN model performed poorly to other benchmark models, except SVR. This can be contributed due to a few sensors for which MRE was found to be significantly higher, as shown in Figure 4.17 and described in Section 4.3.4. Further, while the benchmark models were trained individually for each sensor separately, we trained our models for each cluster or group of sensors which can be attributed to be one of the reason why our model performs poorly for sensors with significant noise or zero volume report compared to other sensors. It can be pointed out that while the mean MRE for TSDIGAN across all sensors for 30% MR was found to be 35.5%, the median MRE was only 20.7%. Also, the mean MRE for 95% of sensors was found to vary between 24.0% - 26.1% in comparison to 35.5% - 39.4% variation, when all sensors are considered. This implies that the mean value shown in Figure 4.18 was significantly affected by performance on few outlying sensors, which led to its poor performance compared to other benchmark models. In future, more efficient clustering techniques can be used either to remove such sensors from performance analysis or separate models can be trained for such sensors, depending on user specific requirements. Overall, the proposed TSDIGAN model outperformed all benchmark models in terms of MAE and $RMSE$, while performing reasonably well in terms of MRE too for majority of sensors.

4.4 Conclusion

In this study, we propose a traffic sensor data imputation framework based on generative adversarial networks (TSDIGAN) that treats the missing data problem as a data generation problem. Our study demonstrates that the generative model based method can perform accurately and robustly to impute missing traffic data under widely varying missing rates. Our proposed model first embeds traffic time-series data into GASF matrix images preserving the temporal correlations. This enables training of a deep convolutional generative adversarial network that can generate realistic-looking synthetic data for missing data imputation. We have also shown our proposed model's

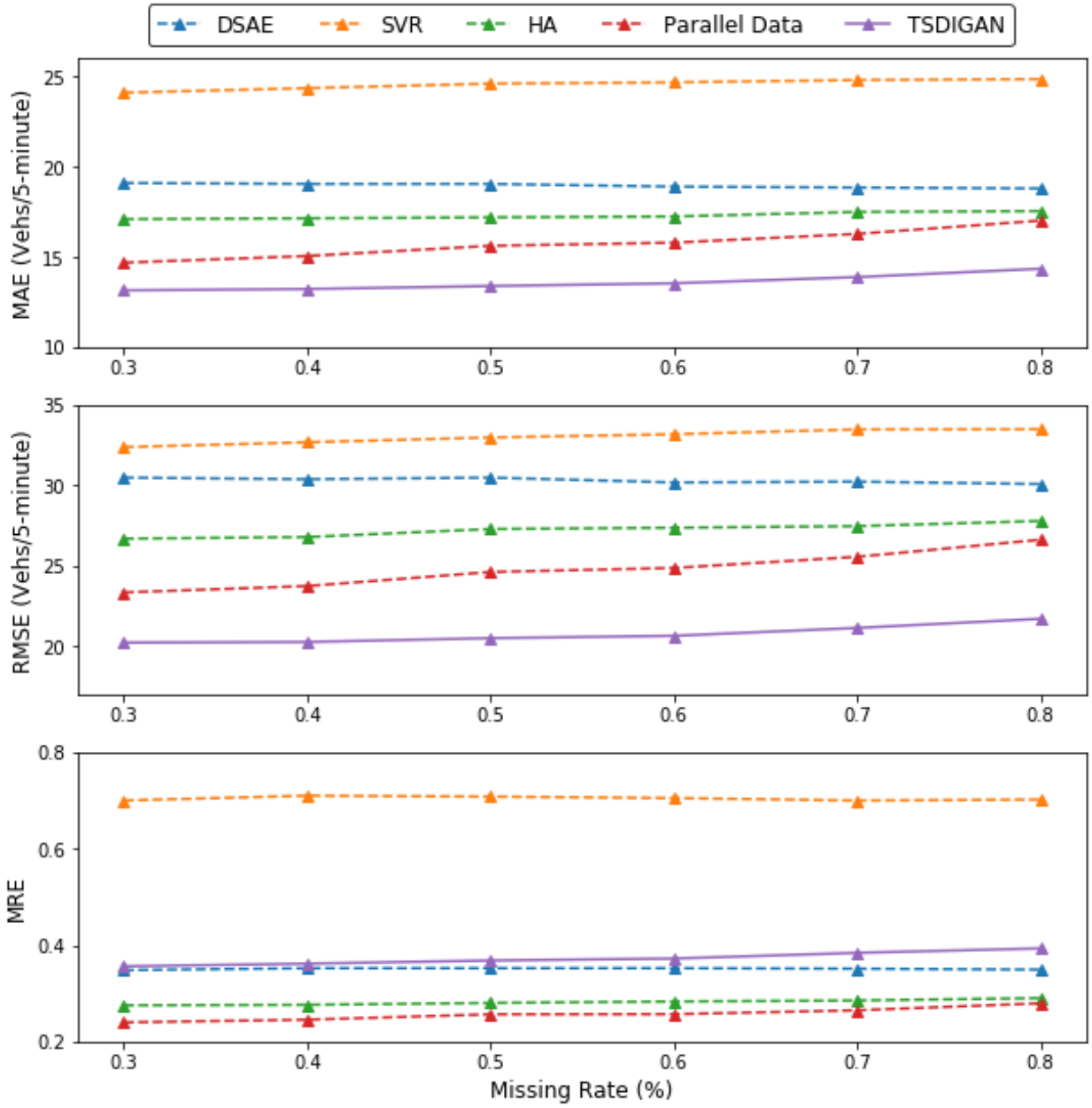


Figure 4.18: Comparison of imputation performance accuracies in terms of (a) MAE, (b) $RMSE$, and (c) MRE with respect to other benchmark imputation models

training process step by step, demonstrating how our model learns to generate its high-quality synthetic data. We have evaluated the performance of the proposed model using benchmark data from PeMS (PeMS, 2014) and further investigated it's capability for large-scale applications. We compared our proposed model performance with other benchmark models, including support vector regression (SVR), history average (HA), denoising stacked autoencoder (DSAE), and GAN-based parallel data model. Our results show that the proposed model can outperform the benchmark models in terms of MAE and $RMSE$, while achieving comparable accuracies in terms of MRE for majority of the sensors. Further, our proposed framework groups the sensors into clusters based on the similarity of their daily traffic patterns to learn the generative model which can be applied to the entire cluster. This can help to train fewer cluster-specific models instead of maintaining each sensor specific model, thereby handling the entire training, testing, and real-world application procedure more efficiently.

Our proposed framework can easily and cheaply generate a variety of realistic synthetic traffic data, which makes it a good choice when it is inconvenient or impossible to get sufficient real traffic data. In addition, the characteristics of our proposed framework offer the possibility of extended ITS applications like data analysis enhancement, anomaly detection, etc. In future, this can be integrated with external features such as weather, special events, and other factors that can impact traffic flow patterns to enable our model to provide more adaptive and accurate imputation performance to appropriately reflect different conditions. Further, in this study, we used k-means clustering to group the sensors based on their daily traffic patterns and develop models for each cluster. In future, this study can be extended to evaluate other efficient clustering techniques such as hierarchical clustering, density based clustering and even determining optimal variation of temporal and spatial traffic data characteristics which can be grouped and worked upon as a single cluster. Also, this can be extended to evaluate the suitability and effectiveness of such generative model based deep learning frameworks for traffic speed generation, prediction, and similar other ITS applications.

CHAPTER 5. TECHNICAL AND ECONOMIC FEASIBILITY ASSESSMENT OF CLOUD-ENABLED TRAFFIC VIDEO ANALYSIS FRAMEWORK

5.1 Introduction

The Smart City (SC) technology that makes our daily lives to be more connected and informed is growing rapidly throughout the world. It represents a new framework which integrates information and communication technology (ICT) and Internet-of-Things (IoT) technology to improve citizens' quality of life (Memos et al., 2018). Nowadays, many organizations and corporations have proposed innovative solutions to help the development of the Smart City, such as NVIDIA's Metropolis, Siemens' MindSphere, Huawei's OceanConnect, etc. As the crucial component of Smart City technology, the IoT-driven and cloud-enabled intelligent transportation systems (ITS) have drawn large amounts of attention to make driving safer, greener and smarter. The ITS market is likely to grow to approximately $130 billion by 2025 (Mishra et al., 2019). According to the USDOT's research-based initiative named "Beyond Traffic 2045: The Smart City Challenge", the department has expanded the resources pool by more than $500 million to support innovative Smart City projects, so as to improve the performance of the transportation system by addressing the issues of congestion, safety, etc. (Gandy Jr et al., 2020).

In traditional ITS, the loop detector and microwave sensors have been the primary choices to gather traffic information. However, the data collected by these traditional sensors is aggregated on the sensor side so that detailed information is lost, and installing such sensors is also costly (Dai et al., 2019). In addition, traditional video surveillance requires watchstanders to pay close attention to hundreds of screens simultaneously, which is challenging and tedious (Liu et al., 2013). Yet there exists already a massive number of traffic surveillance cameras widely installed on the road network across the US to ensure traffic safety. Thus, converting the existing traffic surveillance

cameras into connected "smart sensors", also called intelligent video analysis (IVA), has gained a large amount of attention over the past several years.

Typically, based on computer vision (CV) techniques, traffic cameras have been considered capable sensors for data extraction and analysis in various ITS applications, such as congestion detection, vehicle counting, traffic anomaly detection, etc. (Chakraborty et al., 2018a; Dai et al., 2019; Kumaran et al., 2019). Such ITS approaches mainly consist of vehicle detection and tracking. For vehicle detection, deep learning based methods have been proposed due to the development of convolutional neural networks (CNN) such as the two-stage methods like R-CNN, Fast R-CNN, Faster R-CNN and mask R-CNN and one-stage methods like YOLOv3, SSD, DSSD, RetinaNet, M2Det and RefineDet, as summarized in Jiao et al. (2019). For vehicle tracking, many tracking algorithms have been proposed such as DeepSORT (Wojke et al., 2017), IOU (Bochinski et al., 2018), KLT (Lucas et al., 1981) and DCF (Lukezic et al., 2017). Taking advantage of computer vision techniques, Dai et al. (2019) have proposed a video-based vehicle-counting framework using YOLOv3 and kernelized correlation filters (KCF) as detector and tracker respectively, which achieved an 87.6% counting accuracy running at 20.7 fps, and their video real-time rate was 1.3 (by taking the duration of their framework's run-time divided by the duration of their test videos). Also, Meng et al. (2020) have proposed a correlation-matched tracking algorithm combined with SSD for vehicle counting which reached 93% counting accuracy running at 25 fps on an expressway dataset.

However, current research efforts on deep learning based approaches have been more focused on model effectiveness instead of efficiency (Liu et al., 2018). Traditional analysis frameworks need to store and process video streams locally, which is unfeasible due to intensive computation cost and processing latency, especially for large-scale camera systems. Such offline approaches, with their high latency, lead to delayed decisions, which is not tolerable for IoT-driven ITS applications like accident detection, congestion reduction, etc. (Ferdowsi et al., 2019). Hence, by pushing the deep learning techniques close to their data sources, the edge computing frameworks show potential ability for real-time intelligent video analysis (Liu et al., 2018). To achieve this purpose, NVIDIA

has developed the AI-powered high throughput and scalable inference framework called NVIDIA Deepstream, which has enabled edge computing techniques for multi-GPU and multi-stream video analysis (Nvidia Deepstream, 2020).

Designing the next-generation ITS applications is required to efficiently support the development of Smart Cities. Thus, the cloud server is playing an important role in integrating the edge computing node or IoT devices horizontally, so as to provide real-time and dynamic message exchanges for supporting modern ITS applications (Khan et al., 2018; Petrolo et al., 2017). The benefits of deploying applications on cloud server can be summaried as: (1) massive computation resources, (2) flexible running plan, and (3) easy accessibility for data sharing. Therefore, edge and cloud computing enabled real-time video analysis framework deserved the further study to support large scale smart city based ITS applications. Further, the cost estimation of operating such cloud enabled IVA framework is needed to help the traffic agents build the system efficiently.

In this study, we introduce an end-to-end, real-time, cloud-enabled traffic video analysis framework using NVIDIA Deepstream, which is recognized as the state-of-the-art AI-powered video analysis toolkit. For this study, hundreds of live video streams recorded by traffic CCTV cameras across Iowa were decoded and inferred using NVIDIA Deepstream. The model inference results generated by NVIDIA Deepstream were processed and analyzed using the big data processing platforms Apache Kafka (Kafka, 2019), Apache Spark (Spark, 2020), Elasticsearch and Kibana (Elastic, 2020) for data transmitting, processing, indexing, and visualizing respectively. Toward the goal of efficient access and standardized deployment, each component of the framework is containerized using Docker (Docker, 2020). Taking advantage of the benefits of cloud servers, we have deployed the proposed framework on Google Cloud Platform (GCP) and estimated the operating costs in detail from associated billing reports. Then, we evaluated the technical feasibility of our proposed framework by measuring its vehicle-counting accuracy. Further, we present the extensibility of our proposed framework with the discussion of its limitations for supporting future real-time ITS applications. The main contributions of this study can be concluded as:

1. We introduce a practical, end-to-end traffic video analysis framework using a state-of-the-art multi-GPU and multi-stream analysis toolkit

2. The detailed computational resources usage and operating costs are presented to help the traffic agents develop such IVA framework efficiently.

3. The technical feasibility of proposed framework for real-time large-scale implementations are evaluated by discussing vehicle-counting accuracy under various scenarios.

The remainder of this paper is organized as follows: The next section provides the background and materials for our proposed IVA framework in detail. Then, Section 5.3 discusses the capability and efficiency of our proposed framework, followed by the economic assessment of operating the framework on GCP. Also, the technical feasibility of applying this framework is discussed. Finally, we discuss the potential abilities, limitations and scope of future work on the proposed framework in Section 5.4.

5.2 Methodology

In this section, we present the background and materials we have used in this study. Following the recent AI-based IoT projects, especially NVIDIA smart parking (Nvidia IOT, 2019), NVIDIA real-time video redaction (Shah et al., 2020), and real-time analysis of popular Uber locations (Carol McDonald, 2018), we have introduced a practical cloud-enabled traffic video analysis framework for real-time, large-scale traffic recognition. Our proposed framework consists of two modules, namely, a perception module and an analysis module, with end-to-end implementation on GCP as discussed below.

5.2.1 Perception module

As mentioned in Section 5.1, NVIDIA Deepstream is an accelerated AI-powered framework based on GStreamer, which is a multimedia processing framework (Gstreamer, 2019). The key features of NVIDIA Deepstream can be summarized as follows. First, it provides a multi-stream

processing framework with low latency to help developers to build IVA frameworks easily and efficiently. Second, it delivers high throughput model inference for object detection, tracking, etc. Also, it enables the transmission and integration between model inference results and the IoT interface such as Kafka, MQTT or AMQP. Combined with the NVIDIA Transfer Learning Toolkit (TLT), NVIDIA Deepstream provides highly flexible customization for developers to train and optimize their desired AI models.

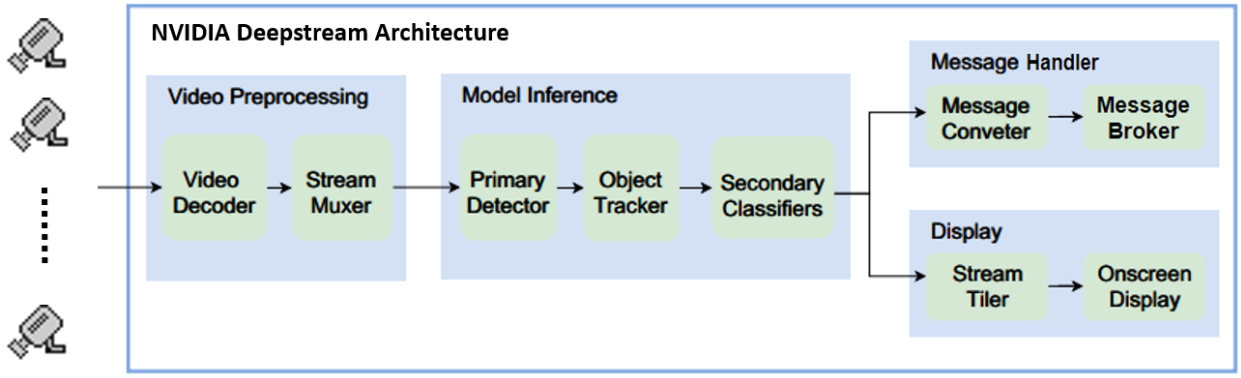


Figure 5.1: Perception module architecture

Therefore, NVIDIA Deepstream is used as our perception module as shown in Figure 5.1. The main components of perception module are: (1) Video pre-processing, which decodes and forms the batches of frames from multiple video stream sources; (2) Model inference, which loads TensorRT-optimized models for vehicle detection; (3) OpenCV-based tracker, which tracks the detected vehicles to provide detailed trajectories; (4) model inference results like bounding boxes and vehicle IDs drawn on composite frames for visualization in onscreen display; and (5) model inference results converted and transmitted through the IoT interface by the message handler using the JSON format.

The perception module supports various video stream inputs like the Real-Time Streaming Protocol (RTSP), recorded video files and V4L2 cameras. In this study, we used live video streams recorded by traffic cameras through the public RTSP managed by the Iowa Department of Transportation. For the primary vehicle detector, we used TrafficCamNet (Nvida TLT, 2020), which

utilizes ResNet18 as its backbone for feature extraction (He et al., 2016). The model was trained to detect four object classes, namely car, person, roadsign, and two-wheeler. A total of 200,000 images, which includes 2.8 million car class objects captured by traffic signal cameras and dashcams, were involved in the training. The model's accuracy and F1 score, which were measured against 19,000 images across various scenes, achieved 83.9% and 91.28%, respectively.

For the multi-object tracking (MOT) task, occlusion and shadow are inevitable issues, especially for vehicle tracking in complex scenarios. Thus, to recognize traffic patterns more robustly and efficiently, we used the NvDCF tracker, which is a low-level tracker based on the discriminative correlation filter (DCF) and Hungarian algorithm. We recommend interested readers refer to Nvidia Deepstream (2020) for more details.

5.2.2 Analysis module

As mentioned above, the model's inference results are delivered to the data transmission interface. Here, the analysis module processes and analyzes the resulting metadata for further ITS applications as shown in Figure 5.2.

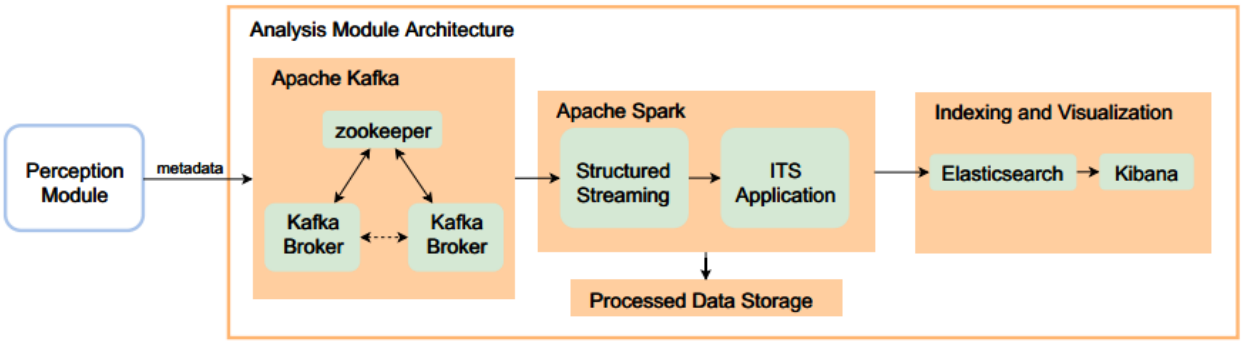


Figure 5.2: Perception module architecture

In the analysis module, we used Apache Kafka to handle the data transmissions on a large-scale basis. Kafka has been introduced as a real-time, fault-tolerant distributed streaming data platform. It takes input data from various sources like mobile devices, physical sensors, web servers, etc. The

data collected from edge devices (say traffic cameras) are published to the Kafka broker with predefined Kafka topics. Each Kafka topic is maintained in many partitions to achieve the purpose of high throughput and fault tolerance. In addition, data from multiple sources can be connected to different Kafka brokers which are managed by a zookeeper.

For large-scale batch data processing in streaming environments, Apache Spark was used in this study. Spark is recognized as a computationally efficient, large-scale data processing platform. By using Resilient Distributed Datasets (RDD) and Directed Acyclic Graphs (DAG), Spark provides a high speed and robust fault-tolerant engine for big data processing. In analysis module, Spark acts like a "consumer," subscribing to the streaming data from Kafka for further data cleaning, data analysis, trend predictions with machine learning, etc. More specifically, Spark structured streaming is used, which is an exactly-once operation built on the Spin a unbounded table which allows the append operation for "micro-batch" processing.

5.2.3 Cloud-enabled Implementation

As mentioned in Section 5.1, deploying a large-scale IVA framework on a cloud server is flexible and powerful as a cloud server provides easy accessibility for traffic agency to maintain and manage IVA systems remotely. In this study, we have implemented our proposed framework on the Google Cloud Platform (GCP), which is a scalable and powerful cloud computing service. Also, Docker (version 19.03) has been used to deploy the proposed framework efficiently. Each component mentioned in Section 5.2.1 and Section 5.2.2 has been containerized and is managed by Docker and Docker Compose. The entire proposed framework is shown in Figure 5.3.

In summary, the perception Docker (NVIDIA Deepstream) takes the live video streams from multiple cameras as inputs through the RTSP. Then, the decoded video frames are inferred by the trained AI model, and the model inference results are sent to the analysis Docker for further data processing and visualization. The entire framework is operated in GCP, which can be remotely accessed by local machines for the purposes of maintenance, inspection, diagnosis, etc.

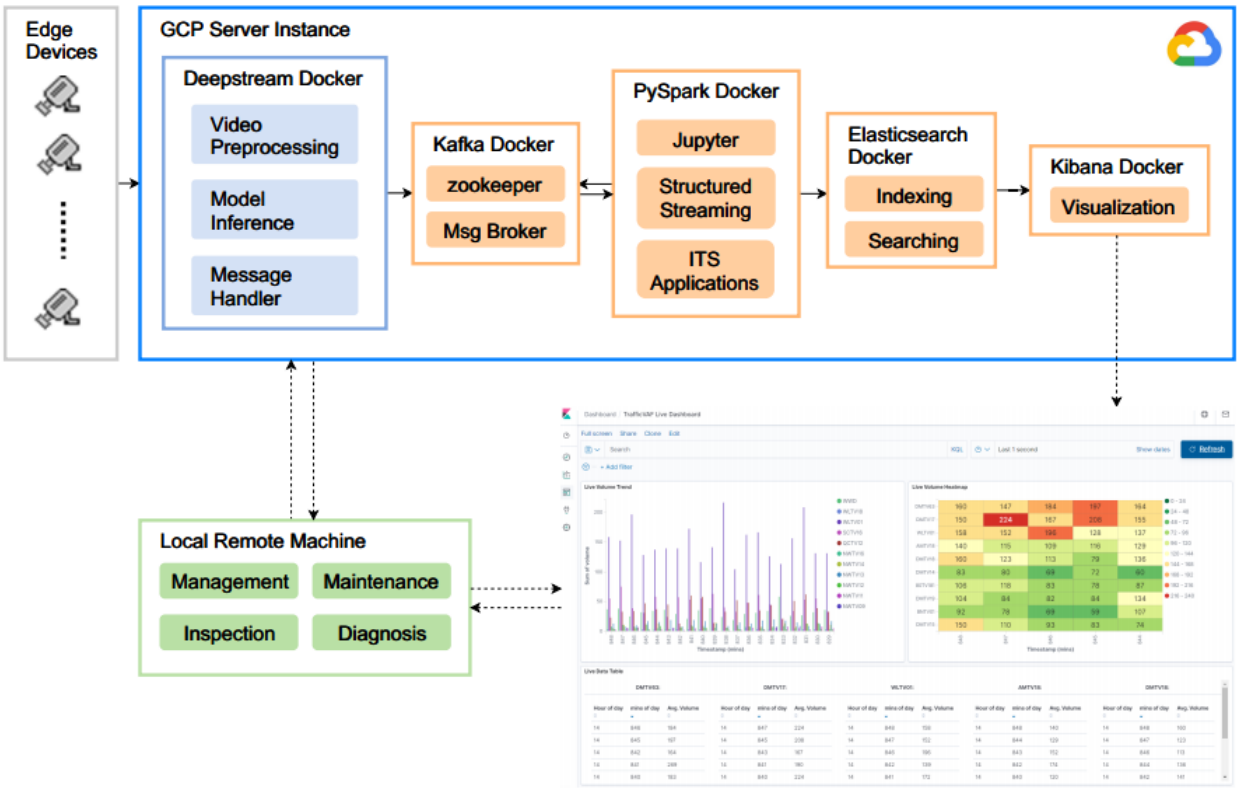


Figure 5.3: Overall Intelligent video analysis framework

5.3 Results

In this study, there were 160 traffic surveillance cameras involved. We accessed their live video streams by using the public RTSP managed by the Iowa Department of Transportation (Iowa DOT). These live videos are encoded following the H.264 standard with the resolution of 480 × 270. In the following sections, we present in detail the resources usage, operating costs and technical feasibility of proposed framework running on GCP.

5.3.1 Economic assessment

The GCP instance we used belongs to the N1 series powered by the Intel Skylake CPU platform. We used 8 virtual CPU (vCPU) cores with 30 GB memory and Ubuntu 18.04 LTS operating systems for evaluation. In addition, we used 2 NVIDIA T4 GPUs for video analysis and a 3 TB persistent disk (PD) for data storage. As mentioned in Section 5.2.1, the perception module can take multiple video streams as inputs, so it requires developers to twiddle the number of video streams (traffic cameras) they need to process simultaneously in each single GPU so that they achieve a balance between performance and cost. We used built-in NVIDIA Deepstream functionality to determine the processing latency of three main components with various numbers of traffic cameras. Results of this latency analysis are presented in Figure 5.4.

For the purpose of processing data in near real time, we distributed 80 cameras to each T4 GPU so that the overall processing latency caused by the perception module was less than 1 second. Thus, the total of 160 cameras was split into two T4 GPUs where the average GPU memory usage was around 65%. In other words, the number of cameras could be decided by a traffic agency based on their tolerance of processing delay, without necessarily exceeding the total GPU memory.

In the perception module, there are 6 consecutive frames skipped at the stage of model inference for computational efficiency. The coexisting CUDA 10.2 and CUDA 10.1, GPU driver version 418+, and TensorRT 7.0 were used to improve the performance of the NVIDIA T4 GPU according to the T4 workaround specification (Nvidia Deepstream, 2020), and Kafka 2.4, Spark 2.4.5, and

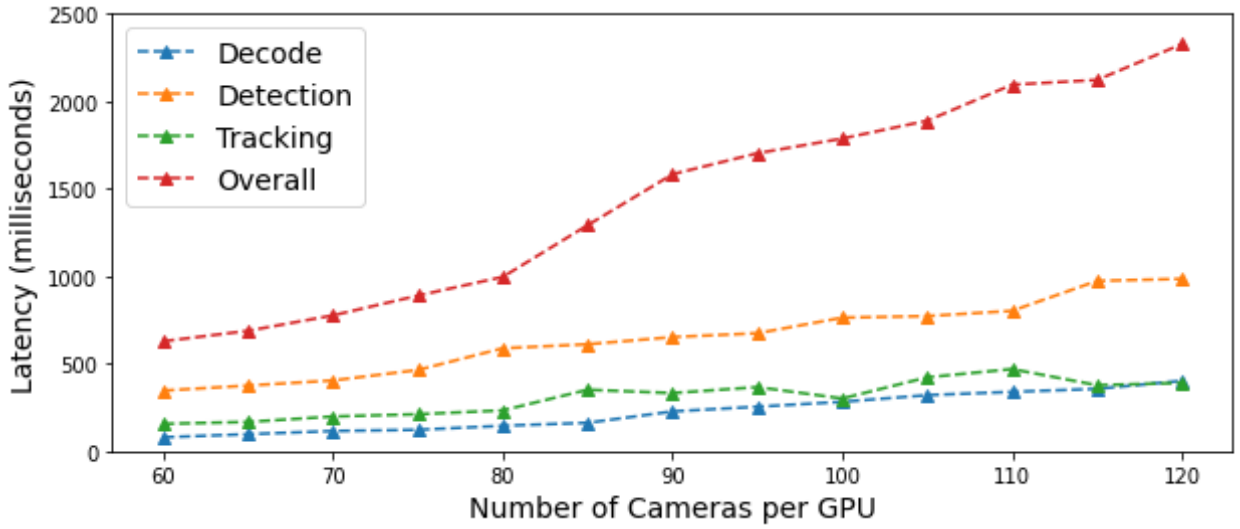


Figure 5.4: Main components latency of perception module with simultaneously running different number of cameras per GPU

Elasticsearch/Kibana 7.5.2 were used in the analysis module. In Table 5.1, we summarize the computational resources usage for running our framework on GCP.

Table 5.1: Summary of computation resources usage on GCP

Container	CPU	Memory	GPU memory (Per GPU)	GPU Util (Per GPU)
Perception0	22.3%	6.9%	65.5%	71.0%
Perception1	21.6%	6.5%	64.1%	69.0%
Kafka	3.8%	2.2%	0%	0%
Spark	30.6%	29.8%	0%	0%
ElasticSearch	10.3%	3.3%	0%	0%
Kibana	0.18%	0.4%	0%	0%

To measure the operating costs in detail, we implemented the proposed framework in GCP for 5 days. The GCP provides a sustained use discount based on monthly usage level. In Figure 5.5a and Figure 5.5b, we show the average daily cost with and without the sustained use discount. The daily operating cost for 160 cameras was $21.59 with the sustained use discount (assuming such a traffic video analysis framework would be operating for the long term). The GPUs and vCPUs are

the two major costs, which respectively took 46.4% and 19.7% of the total cost, while the persistent storage, network, and RAM together took 33.9% of the total cost. Thus, the daily cost of proposed framework for each camera was $0.135, leading to the yearly cost per camera of $49.30. In Iowa, there are 390 operating traffic surveillance cameras on the road network. Thus, turning all these surveillance cameras into connected "smart sensors" would cost $1601.40 per month using such a cloud-enabled system, with the benefit of eliminating additional infrastructure costs.

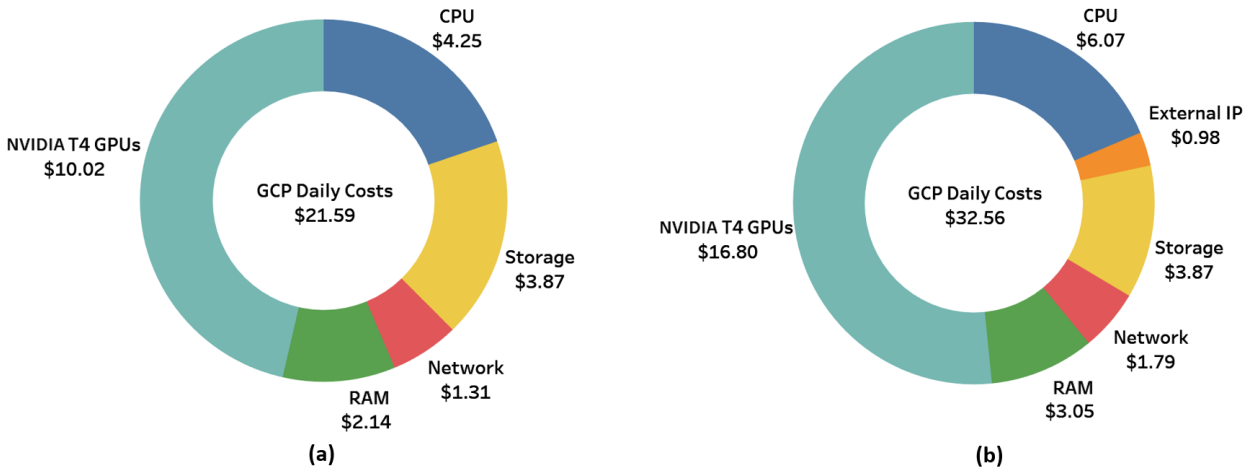


Figure 5.5: Daily operating cost for 160 cameras (a) with sustained use discount, and (b) without sustained use discount

5.3.2 Feasibility assessment

In this section, we present the feasibility of our proposed framework by measuring its vehicle counting accuracy compared with manual inspection and counting via nearby radar sensors. There were 12 cameras with different viewing angles selected for this evaluation, as shown in Figure 5.6. The selected cameras are representative, since they cover the most common camera viewing angles across Iowa.

For each camera, we manually counted the vehicles that passed a predefined region of interest (ROI) for 5 minutes on a sunny day and 4 minutes on a rainy/cloudy day around 11 am as the

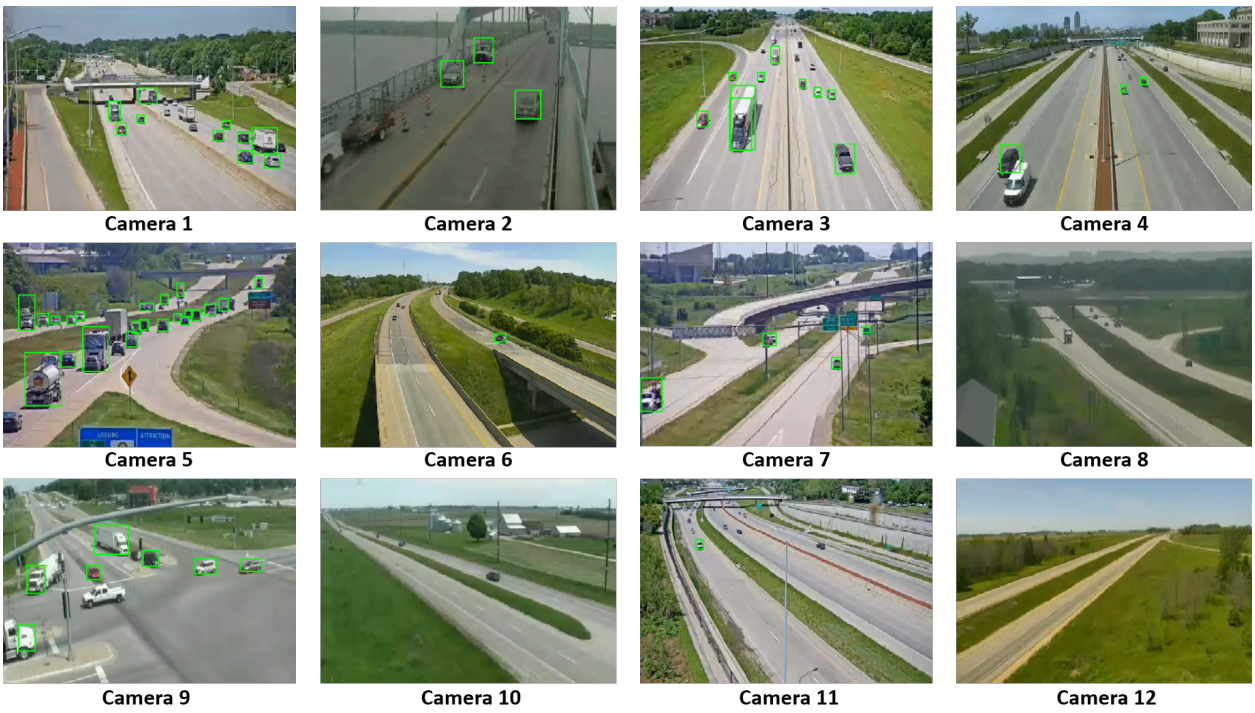


Figure 5.6: Sampled cameras with different viewing angles

ground truth, thus there were 108 minutes of video involved in our evaluation. The ROI for these cameras was set from 90 to 260 pixels counting from the bottom left along the y-axis. For instance, the ROIs for camera 1 and camera 2 were from 10 to 90 pixels and from 10 to 200 pixels along the y-axis respectively. The accuracy of measurement of our vehicle counting application was defined as:

$$Cr = \frac{C_g - |C_g - C_e|}{C_g} \tag{5.1}$$

Where C_r, C_g, and C_e denote the (1) correct counting rate, (2) counting ground truth, and (3) counting estimated by proposed framework. The results of this evaluation of our vehicle counting application for different cameras under both sunny and cloudy/rainy environments are shown in Table 5.2.

Table 5.2: Experimental results for different cameras under different environment

	Sunny				Rainy/Cloudy			
Scenes	Cg	Ce	$\|Cg - Ce\|$	Cr (%)	Cg	Ce	$\|Cg - Ce\|$	Cr (%)
Cam 1	511	539	28	94.5	344	401	57	83.4
Cam 2	108	103	5	95.4	92	98	6	93.5
Cam 3	381	470	89	76.6	268	391	123	54.1
Cam 4	326	336	10	96.9	223	221	2	99.1
Cam 5	129	148	19	85.3	97	102	5	94.8
Cam 6	126	138	12	90.5	91	113	22	75.8
Cam 7	107	113	6	94.4	101	104	3	97.0
Cam 8	65	56	9	86.2	44	38	6	86.4
Cam 9	179	211	32	82.1	90	292	202	-124.4
Cam 10	106	112	6	94.3	73	31	42	42.5
Cam 11	359	291	68	81.1	259	234	25	90.3
Cam 12	51	14	37	27.5	31	10	21	32.3

In addition, we show trajectory plots for 2 sample cameras under the sunny and rainy day scenarios in Figure 5.7. For sunny days, the results show that the proposed framework is working as expected in that its average counting accuracy achieved 90.1%, except with the cameras 3 and 12. During the test periods, camera 3 was suffering from the network lag issue, so the video frames transmitted to the perception module were nonconsecutive. Such a lag issue will make the tracking

model produce labels repeatedly for the same vehicle. For camera 12, the perception module barely captured the passing vehicles, which indicates that the model needs to be fine-tuned to fit this scenario involving distant viewing angles. In addition, the results show that the counting accuracy was affected on rainy days for camera 6, camera 9, and camera 10 which, as shown in Figure 5.7d, were exposed to the rain.

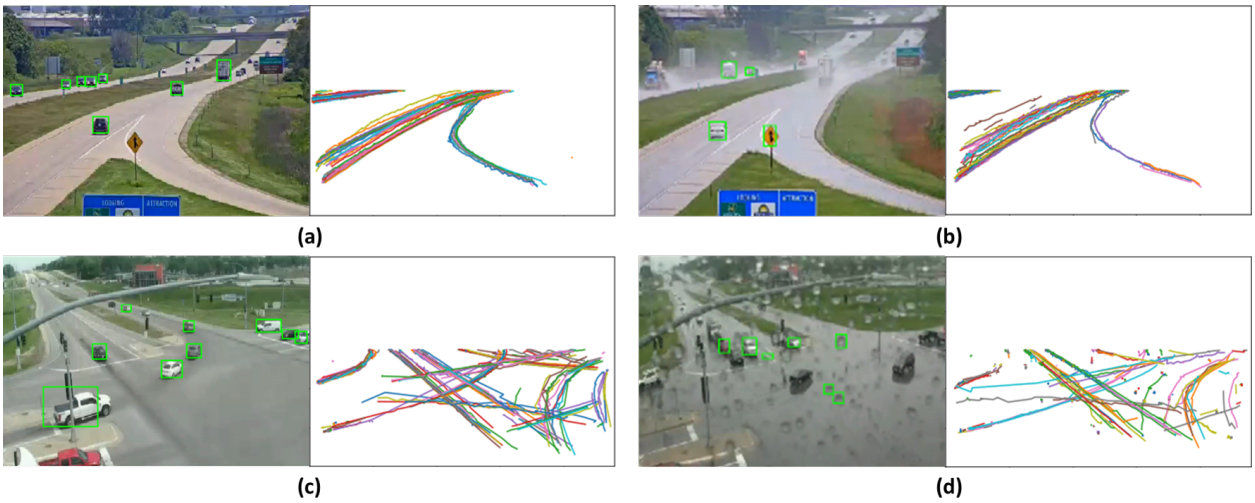


Figure 5.7: Camera views and trajectories plots under sunny and rainy day (a) Camera 5 sunny day, (b) Camera 5 rainy day, (a) Camera 9 sunny day, (b) Camera 9 rainy day

To further measure the feasibility of our proposed framework, we compared the daily volume pattern captured by our testing cameras versus by their nearby microwave radar sensors, as shown in Figure 5.8. However, it is worthy of mention that matching the detection areas of camera sensors and radar sensors point by point is hard, since our camera sensors were zoomed into the ROI with ramps, interchanges and minor roads included, as discussed in Section 5.3.2. Therefore, the overall traffic pattern comparison is more meaningful in showing the agreement between these two types of sensors.

The data of the daily volume plots were aggregated in 10-minute intervals. For camera 1 and camera 11, the volume trend after 3 pm is not presented, since these cameras turn the opposite direction after 3 pm, which is beyond the scope of this study. Also, the comparison of camera 2,

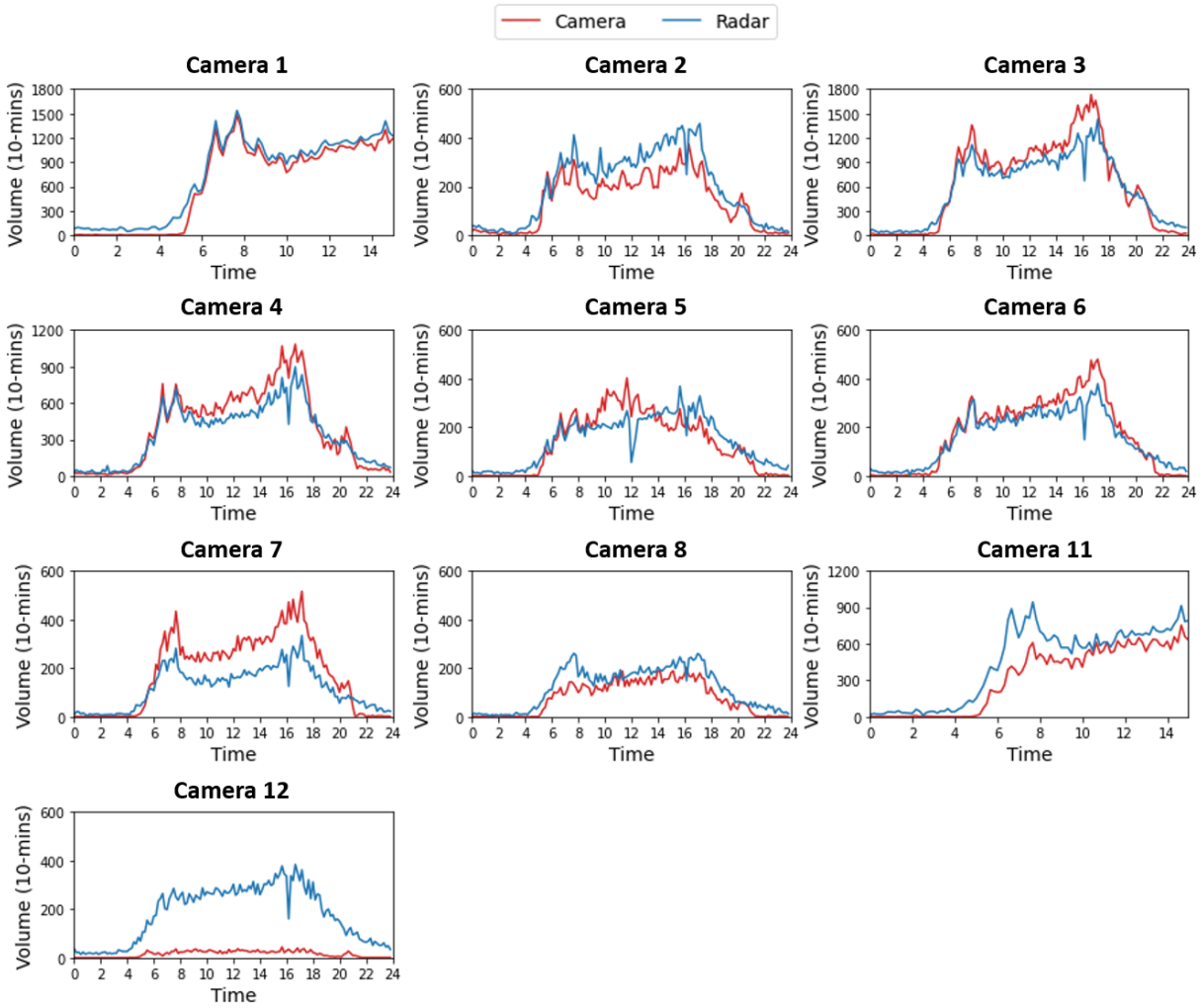


Figure 5.8: Daily traffic volume data (10-mins agg) captured by cameras and nearby radar sensors on Monday at June 1st

camera 5 and camera 7 was impacted by the nearby interchanges and ramps, especially for camera 7 whose closest radar sensor had failed (thus we had to use the radar sensor located 2 interchanges above instead). In addition, for camera 11, the vehicles on a minor road which was far from the camera missed detection. Similarly, as described earlier, our model only occasionally captured the passing vehicles within distant view of camera 12. In Table 5.3 and Table 5.4, we aggregate the data in 3-hour intervals and present the differences using the absolute error (AE) and relative error (RE) defined as:

$$AE = |V_{camera} - V_{radar}| \tag{5.2}$$

$$RE = \frac{|V_{camera} - V_{radar}|}{V_{radar}} \tag{5.3}$$

where V represents the 3-hour aggregated traffic volumes.

Overall, the traffic patterns captured by the testing cameras using computer vision techniques versus by their nearby radar sensors are comparable, except with camera 12 where, as noted earlier, model fine-tuning for distant viewing angles is needed. Also, the results shown in both Figure 5.8 and Table 5.4 demonstrate that the model does not perform well during the night (between 9 pm and 6 am), which indicates the model also requires fine-tuning for night-time applications.

Table 5.3: Absolute error (AE) of daily volume between camera and radar sensors

	Time of Day							
Scenes	0 - 3	3 - 6	6 - 9	9 - 12	12 - 15	15 - 18	18 - 21	21 - 24
Cam 1	1276	2413	1277	1397	1394	38	NA	NA
Cam 2	220	313	1266	1406	1860	1946	619	359
Cam 3	619	1135	2417	1690	2772	5520	496	1814
Cam 4	331	218	678	1848	2672	3701	319	877
Cam 5	241	480	270	1622	554	1168	781	735.5
Cam 6	264	267	341	499	795	1776	434	597
Cam 7	176	240	1653	1829	2285	2964	1265	685
Cam 8	151	450	1405	474	661	1104	698	569
Cam 11	553	1914	5051	1573	1868	160	NA	NA
Cam 12	357	1048	3857	4282	4678	5206	3002	1150

Table 5.4: Relative error (RE) of daily volume between camera and radar sensors

	Time of Day							
Scenes	0 - 3	3 - 6	6 - 9	9 - 12	12 - 15	15 - 18	18 - 21	21 - 24
Cam 1	95.9	60.3	6.2	7.8	6.5	3.1	NA	NA
Cam 2	52.0	23.4	24.4	27.9	30.2	27.9	22.5	59.0
Cam 3	78.0	43.8	16.2	12.1	16.7	27.1	4.5	59.8
Cam 4	49.5	11.4	7.3	22.8	27.9	30.0	5.4	41.1
Cam 5	90.9	53.9	8.0	42.6	13.6	24.1	29.6	77.5
Cam 6	83.9	29.4	9.0	12.2	16.8	33.5	16.3	64.7
Cam 7	86.8	33.3	47.8	69.3	69.7	69.4	61.5	84.8
Cam 8	91.0	65.4	40.6	16.6	19.2	27.8	38.4	89.0
Cam 11	89.8	74.2	40.1	14.5	14.4	20.2	NA	NA
Cam 12	100.0	89.8	91.0	90.0	90.6	91.0	93.3	96.5

As mentioned above, one advantage of our proposed framework is its large-scale production of vehicle trajectories on a real-time basis, which has potential for use in ITS applications like anomaly detection, speed violation detection, wrong-way detection , etc. In Figure 5.9, we present the extensibility of our proposed framework by showing three anomalous events on a freeway with their corresponding image coordinates' moving speed calculated from the associated vehicle trajectories. The mean and minimum image coordinates' moving speed was calculated from all the detected vehicles and their trajectories during a certain time period (say 1 second). It is clearly observable that, by applying appropriate thresholds or advanced algorithms, such real-time trajectories generated by our proposed framework could help traffic agency to quickly address anomalous events.

For long-term, real-world operations, it should be noted that the current GCP does not support live migration for cloud instances with GPUs during the host maintenance period, which, as tested in our experiments, typically occurs once a week. In other words, the running programs will be disrupted automatically by the GCP system for the purpose of host maintenance. Therefore, to minimize the disruption of their programs, users need to prepare for their workload's transition through this system restart by monitoring the maintenance schedule, which is usually announced 1 hour a head of system termination.

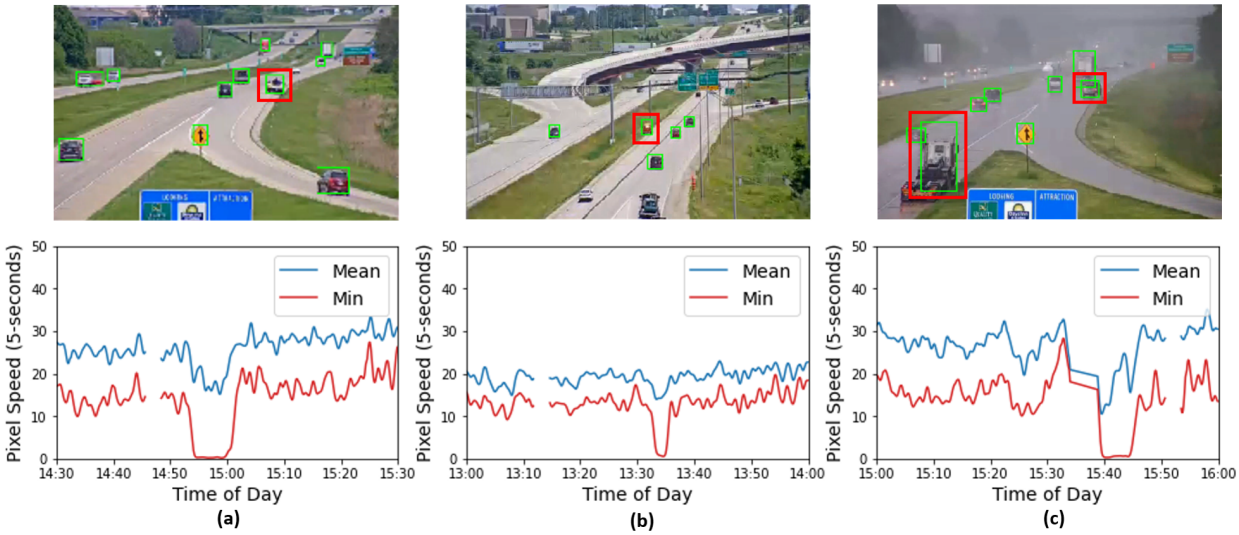


Figure 5.9: Anomaly events with pixel moving speed

5.4 Conclusions

In this study, we have introduced an end-to-end, real-time, cloud-enabled traffic video analysis framework using NVIDIA Deepstream, and we have evaluated the proposed framework from two perspectives: economics and feasibility. The proposed framework consists mainly of a perception module and analysis module. The perception module takes multiple live video streams as inputs, and outputs insightful inferred metadata which are produced by the embedded AI model. Then, these resulting metadata are transmitted to the analysis module through Kafka. In the analysis module, Spark consumes the data and forms a dynamic unbounded table for batch processing. Finally, the processed data are indexed and visualized by Elasticsearch and Kibana.

Our study demonstrates the proposed framework is both economically efficient and technically feasible. From the perspective of economics, the results show that the daily operating cost for each camera is less than $0.14, so the yearly operating cost per camera is less than $50. In addition, we have presented the processing latency of our framework's main components and its usage of cloud computational resources, which can help developers design AI-powered traffic video analysis frameworks accordingly based on their requirements. From the perspective of technical feasibility,

we have measured the accuracy of a vehicle-counting application by using the live video streams recorded by 12 cameras with different viewing angles. The results show that for most of the tested viewing angles, the proposed framework is able to produce the expected vehicle-counting results compared with both manual inspections and the count from nearby microwave sensors. Further, we have demonstrated the potential ability of our proposed framework for future real-time ITS applications by showing sample anomalous events with image coordinates' moving speed calculated in seconds.

It should be noted that such real-time ITS applications will likely always suffer from the issue of network lag. As discussed in Section 5.3, the nonconsecutive frames caused by network lag affect the performance of our proposed framework for both vehicle detection and tracking modeling. Thus, developers need to work with their local department of transportation to optimize network bandwidth and address network lag problems. In addition, from our experiments, GCP cloud instances will restart once a week for the purpose of host maintenance, thus a proper transition plan is needed for long-term operations. In addition, model fine-tuning is required to improve performance for scenarios like distant viewing angles, nighttime and rainy days.

Future work will pursue: (1) More images from different viewing angles to fine-tune the current vehicle detection model, so as to improve our framework's performance in various environments, (2) A real-time alert system for deployment to help traffic operation centers address anomalous events quickly, (3) Calibration of the image coordinates of detected vehicles with the real-world geometric coordinates of the roadway network for more precise travel speed estimation and data visualization, and (4) The creation of modules using machine learning libraries for prediction applications like travel speed estimation, congestion prediction, etc.

CHAPTER 6. CONCLUSIONS

Accurate and sufficient high-quality traffic data are crucial component of various advanced Intelligent Transportation System applications. In addition, by taking the advantages of advanced image/video processing techniques, integrating camera sensors is a key step to support the development of future Smart Cities based ITS. This study consists of three broad topics to design the intelligent traffic sensor system using big data and machine learning techniques to (1) detecting the failure traffic sensors; (2) imputing the erroneous or missing data; and (3) integrating camera sensors.

6.1 Large-Scale Data-driven Traffic Sensor Health Monitoring

Our first research objective aims to develop a large-scale, data-driven traffic sensor health monitoring (TSHM) framework using massively parallelizable data processing techniques for large traffic sensor network. The proposed stepwise framework consists of three major step, where each step identifies distinct sensor abnormalities. First, our proposed framework captures the traffic sensors with high missing data rates by assigning different levels of missing data severity. Second, we adopted reduced feature from each sensor's aggregated $AEVL$ distribution to detect failure sensors based on the logic of anomaly detection using clustering analysis. Finally, a novel temporal-pattern-based anomaly detection method using Bayesian changepoint detection (time series segmentation) to capture the failure sensors which reporting unexpected frequent fluctuations over daily data stream. However, one major challenge in large-scale abnormal sensor detection is the difficulty in obtaining groudtruth. Due to the leak of explicit definition of anomalies, we here identified the abnormal sensors by showing sensor data along different feature dimensions. For instance, we compared the detected abnormal sensors with their nearest adjacent upstream and downstream sensors, and we present the heatmaps of raw data to show points of agreements. Our results show

the efficacy of proposed TSHM framework to identify the failure sensors from the large scale traffic sensor network.

6.2 Deep convolutional generative adversarial networks for traffic data imputation encoding time series as images

Undoubtedly, future traffic sensor system need the practical module to impute erroneous or missing traffic data, since many advanced ITS applications need the complete and high quality data. Our second research objective is to develop a accurate and practical traffic sensor data imputation framework based on generative adversarial networks (TSDIGAN). Our proposed model first encodes traffic time-series data into GASF matrix images preserving the temporal correlations. Such encoding enables training of deep convolutional generative adversarial network which can generate synthetic data for imputation. We have evaluated the performance of the proposed model using benchmark data from PeMS (PeMS, 2014) which is recognized as the benchmark dataset for transportation research. We demonstrate how our proposed model learn to generate realistic-looking synthetic data by showing the model training process step by step. Then, we group the sensors into clusters based on their daily traffic patterns to learn the generative model from each of clusters. Such clustering based implementation can improve the framework efficiency by avoid maintaining each sensor' specific model, thereby handling the entire training and imputation more efficiently. We compared our proposed model performance with other benchmark method including support vector regression (SVR), history average (HA), denoising stacked autoencoder (DSAE), and GAN-based parallel data model. Our results show that the proposed model can outperform the benchmark models in terms of MAE and $RMSE$, while achieving comparable accuracies in terms of MRE for majority of the sensors. In addition, one advantage of our proposed framework is to easily and cheaply generate a variety of realistic synthetic traffic data, which makes it a good choice when it is inconvenient or impossible to get sufficient real traffic data. Such characteristics of generative model extend the possibility of other ITS application like data enhancement, anomaly detection and etc.

6.3 Technical and economic feasibility assessment of cloud-enabled traffic video analysis framework

The third part of this study aims at introducing an end to end, real-time, cloud-enabled traffic video analysis framework with the evaluation of its economic and technical feasibility to help the traffic agency converting the current traffic camera sensors in to the connected "smart sensors". The proposed traffic video analysis framework consists of two module, namely, perception module and analysis module. Perception module aims at decoding and analyzing live video streams recorded by traffic cameras across the state of Iowa. Then, the results of detection and tracking also called metadata will be delivered to the analysis module through IoT enabled big data transmission platform. Analysis module takes metadata as inputs and perform the subsequent data processing and analysis to support further ITS applications. We deployed the entire proposed framework on google cloud server which supporting the cloud computing and data exchange. Our results show that the daily operating cost for each camera is less than $0.14, so the yearly operating costs is less than $50. And, our proposed framework is able to produce the expected and comparable vehicle counting results compared with both manual inspections and the count from nearby microwave sensors. Further, we have demonstrated the potential extensibility of our proposed framework for other real-time ITS applications like anomaly detection by showing the sample events with their image coordinates' moving speed calculated in seconds.

6.4 Limitations and Future Work

Our entire study focused on developing an intelligent sensor system to support the future ITS applications using big data driven machine learning techniques. However, implementing such large-scale system involve some challenges which can be addressed in the future work. First, our method for detecting the failure traffic sensors is an offline method which is built using historical traffic data. To enable instant detection of abnormal sensors, the method can be extended to meet the real-time basis. Also, the model performance can be improved by using traffic information from other data sources like probe data and camera data.

For our data imputation module, the cluster-specific models are trained offline using historical yearly data, and the external features like weather and special events which might impact the traffic pattern have not been involved in this study. In future work, the model performance can be improved by integrating the external features to provide more adaptive and accurate imputation results. And, other efficient clustering techniques like hierarchical clustering and density based clustering can be explored to improve the classification of traffic pattern. Further, our imputation module can be extended to fit other ITS applications like traffic speed prediction and anomaly detection.

For our camera integration module, such real-time traffic video analysis framework will likely always suffer from the issue of network lag, so that the nonconsecutive frames will be delivered to the proposed framework which will affect both the detection and tracking accuracy. And, from our experiments, long-term operating the framework on cloud server need a proper transition plan to prevent the host interruption. In future study, the detection model can be fine-tuned using more data to improve the performance, especially for those cameras with distant viewing angles. To build a more reliable and precise video-based real-time alert system, one additional model, converting image coordinates of detection vehicle into real-world geometric coordinates will be developed.

BIBLIOGRAPHY

Al-Ariny, Z., Abdelwahab, M. A., Fakhry, M., and Hasaneen, E.-S. (2020). An efficient vehicle counting method using mask r-cnn. In *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, pages 232–237. IEEE.

Al-Deek, H. M., Venkata, C., and Chandra, S. R. (2004). New algorithms for filtering and imputation of real-time and archived dual-loop detector data in i-4 data warehouse. *Transportation research record*, 1867(1):116–126.

Allison, P. D. (2001). *Missing data*, volume 136. Sage publications.

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. (2014). Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832.

Ananthanarayanan, G., Bahl, V., Cox, L., Crown, A., Nogbahi, S., and Shu, Y. (2019). Demo: Video analytics-killer app for edge computing. In *ACM MobiSys*.

Anjum, A., Abdullah, T., Tariq, M., Baltaci, Y., and Antonopoulos, N. (2016). Video stream analysis in clouds: An object detection and classification framework for high performance video analytics. *IEEE Transactions on Cloud Computing*.

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

Armanious, A. (2019). District 07 mobility performance report: 2018 fourth quarter. Technical report, California Department of Transportation.

Asadi, R. and Regan, A. (2019). A convolution recurrent autoencoder for spatio-temporal missing data imputation. *arXiv preprint arXiv:1904.12413*.

Berkhin, P. (2006). A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer.

Bochinski, E., Senst, T., and Sikora, T. (2018). Extending iou based multi-object tracking by visual information. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE.

Borji, A. (2019). Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65.

Bouwmans, T. and Zahzah, E. H. (2014). Robust pca via principal component pursuit: A review for a comparative evaluation in video surveillance. *Computer Vision and Image Understanding*, 122:22–34.

Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11.

Carol McDonald (2018). Real-time analysis of popular uber locations using apache apis: Spark structured streaming, machine learning, kafka and mapr database. https://mapr.com/blog/. accessed April 10, 2020.

Castro-Neto, M., Jeong, Y.-S., Jeong, M.-K., and Han, L. D. (2009). Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert systems with applications*, 36(3):6164–6173.

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. (2019). Data-driven parallelizable traffic incident detection using spatio-temporally denoised robust thresholds. *Transportation research part C: emerging technologies*, 105:81–99.

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.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15.

Chang, G., Zhang, Y., and Yao, D. (2012). Missing data imputation for traffic flow based on improved local least squares. *Tsinghua Science and Technology*, 17(3):304–309.

Chen, C., Kwon, J., Rice, J., Skabardonis, A., and Varaiya, P. (2003). Detecting errors and imputing missing data for single-loop surveillance systems. *Transportation Research Record*, 1855(1):160–167.

Chen, C., Wang, Y., Li, L., Hu, J., and Zhang, Z. (2012). The retrieval of intra-day trend and its influence on traffic prediction. *Transportation research part C: emerging technologies*, 22:103–118.

Chen, Y., Lv, Y., and Wang, F.-Y. (2019). Traffic flow imputation using parallel data and generative adversarial networks. *IEEE Transactions on Intelligent Transportation Systems*.

Cummins, N., Amiriparian, S., Hagerer, G., Batliner, A., Steidl, S., and Schuller, B. W. (2017). An image-based deep spectrum feature representation for the recognition of emotional speech. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 478–484.

Dai, Z., Song, H., Wang, X., Fang, Y., Yun, X., Zhang, Z., and Li, H. (2019). Video-based vehicle counting framework. *IEEE Access*, 7:64460–64470.

Docker (2020). Docker 19.03. https://www.docker.com/. accessed March 10, 2020.

Donahue, C., McAuley, J., and Puckette, M. (2018). Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*.

Duan, Y., Lv, Y., Liu, Y.-L., and Wang, F.-Y. (2016). An efficient realization of deep learning for traffic data imputation. *Transportation research part C: emerging technologies*, 72:168–181.

Elastic (2020). elasticsearch-kibana 7.5.2. https://www.elastic.co/elasticsearch/. accessed January 21, 2020.

Eltoweissy, M., Azab, M., Olariu, S., and Gračanin, D. (2019). A new paradigm for a marketplace of services: Smart communities in the iot era. In *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 1–6. IEEE.

Erman, J., Arlitt, M., and Mahanti, A. (2006). Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286. ACM.

Esteban, C., Hyland, S. L., and Rätsch, G. (2017). Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Fearnhead, P. (2006). Exact and efficient bayesian inference for multiple changepoint problems. *Statistics and computing*, 16(2):203–213.

Fearnhead, P. and Liu, Z. (2007). On-line inference for multiple changepoint problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):589–605.

Ferdowsi, A., Challita, U., and Saad, W. (2019). Deep learning for reliable mobile edge analytics in intelligent transportation systems: An overview. *ieee vehicular technology magazine*, 14(1):62–70.

Gan, Q., Gomes, G., and Bayen, A. (2017). Estimation of performance metrics at signalized intersections using loop detector data and probe travel times. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):2939–2949.

Gandy Jr, O. H., Nemorin, S., et al. (2020). Transportation and smart city imaginaries: A critical analysis of proposals for the usdot smart city challenge. *International Journal of Communication*, 14:21.

Ghosh, B., Basu, B., and O'Mahony, M. (2007). Bayesian time-series model for short-term traffic flow forecasting. *Journal of transportation engineering*, 133(3):180–189.

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. (2014). 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.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. J. (2007). A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520.

Gstreamer (2019). Gstreamer open source multimedia framework. https://devblogs.nvidia.com/real-time-redaction-app-nvidia-deepstream-part-1-training/. accessed Dec 18, 2019.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Jacobson, L. N., Nihan, N. L., and Bender, J. D. (1990). *Detecting erroneous loop detector data in a freeway traffic management system*. Number 1287.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., and Qu, R. (2019). A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868.

Kafka (2019). Kafka 2.4. https://kafka.apache.org/. accessed December 16, 2019.

Khan, M. A., Abbas, S., Hasan, Z., and Fatima, A. (2018). Intelligent transportation system (its) for smart-cities using mamdani fuzzy inference system. *no. January*.

Kim, Y., Wang, P., Zhu, Y., and Mihaylova, L. (2018). A capsule network for traffic speed prediction in complex road networks. In *2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–6. IEEE.

Klein, L. A., Mills, M. K., Gibson, D. R., et al. (2006). Traffic detector handbook: Volume i. Technical report, Turner-Fairbank Highway Research Center.

Kodinariya, T. M. and Makwana, P. R. (2013). Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95.

Ku, W. C., Jagadeesh, G. R., Prakash, A., and Srikanthan, T. (2016). A clustering-based approach for data-driven imputation of missing traffic data. In *2016 IEEE Forum on Integrated and Sustainable Transportation Systems (FISTS)*, pages 1–6. IEEE.

Kumaran, S. K., Dogra, D. P., and Roy, P. P. (2019). Anomaly detection in road traffic using visual surveillance: A survey. *arXiv preprint arXiv:1901.08292*.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

Lee, D., Kim, J., Moon, W.-J., and Ye, J. C. (2019). Collagan: Collaborative gan for missing image data imputation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2487–2496.

Lee, H. and Coifman, B. (2011). Quantifying loop detector sensitivity and correcting detection problems on freeways. *Journal of Transportation Engineering*, 138(7):871–881.

Lefebvre, N., Chen, X., Beauseroy, P., and Zhu, M. (2017). Traffic flow estimation using acoustic signal. *Engineering Applications of Artificial Intelligence*, 64:164–171.

Li, D., Chen, D., Jin, B., Shi, L., Goh, J., and Ng, S.-K. (2019). Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*, pages 703–716. Springer.

Li, L., Li, Y., and Li, Z. (2013). Efficient missing data imputing for traffic flow by considering temporal and spatial dependence. *Transportation research part C: emerging technologies*, 34:108–120.

Li, Y., Li, Z., and Li, L. (2014). Missing traffic data: comparison of imputation methods. *IET Intelligent Transport Systems*, 8(1):51–57.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Liu, H., Chen, S., and Kubota, N. (2013). Intelligent video systems and analytics: A survey. *IEEE Transactions on Industrial Informatics*, 9(3):1222–1233.

Liu, P., Qi, B., and Banerjee, S. (2018). Edgeeye: An edge service framework for real-time intelligent video analytics. In *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, pages 1–6.

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

Lou, L., Zhang, Q., Liu, C., Sheng, M., Liu, J., and Song, H. (2019). Detecting and counting the moving vehicles using mask r-cnn. In *2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 987–992. IEEE.

Lu, L., Wang, J., He, Z., and Chan, C.-Y. (2017). Real-time estimation of freeway travel time with recurrent congestion based on sparse detector data. *IET Intelligent Transport Systems*, 12(1):2–11.

Lu, Y., Yang, X., and Chang, G.-L. (2014). Algorithm for detector-error screening on basis of temporal and spatial information. *Transportation Research Record*, 2443(1):40–48.

Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision.

Lukezic, A., Vojir, T., Cehovin Zajc, L., Matas, J., and Kristan, M. (2017). Discriminative correlation filter with channel and spatial reliability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6309–6318.

Luo, Y., Cai, X., Zhang, Y., Xu, J., et al. (2018). Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1596–1607.

Lv, Y., Chen, Y., Li, L., and Wang, F.-Y. (2018). Generative adversarial networks for parallel transportation systems. *IEEE Intelligent Transportation Systems Magazine*, 10(3):4–10.

Lv, Y., Duan, Y., Kang, W., Li, Z., and Wang, F.-Y. (2014). Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873.

Ma, D., Luo, X., Li, W., Jin, S., Guo, W., and Wang, D. (2017). Traffic demand estimation for lane groups at signal-controlled intersections using travel times from video-imaging detectors. *IET Intelligent Transport Systems*, 11(4):222–229.

Ma, X., Tao, Z., Wang, Y., Yu, H., and Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187–197.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Memos, V. A., Psannis, K. E., Ishibashi, Y., Kim, B.-G., and Gupta, B. B. (2018). An efficient algorithm for media-based surveillance system (eamsus) in iot smart city framework. *Future Generation Computer Systems*, 83:619–628.

Meng, Q., Song, H., Zhang, Y., Zhang, X., Li, G., and Yang, Y. (2020). Video-based vehicle counting for expressway: A novel approach based on vehicle detection and correlation-matched tracking using image data from ptz cameras. *Mathematical Problems in Engineering*, 2020.

Mishra, S., Patel, S., Panda, A. R. R., and Mishra, B. K. (2019). Exploring iot-enabled smart transportation system. In *The IoT and the Next Revolutions Automating the World*, pages 186–202. IGI Global.

Mori, U., Mendiburu, A., Álvarez, M., and Lozano, J. A. (2015). A review of travel time estimation and forecasting for advanced traveller information systems. *Transportmetrica A: Transport Science*, 11(2):119–157.

Ni, D. and Leonard, J. D. (2005). Markov chain monte carlo multiple imputation using bayesian networks for incomplete intelligent transportation systems data. *Transportation research record*, 1935(1):57–67.

Nihan, N. L. (1997). Aid to determining freeway metering rates and detecting loop errors. *Journal of Transportation Engineering*, 123(6):454–458.

Nisa, K. K., Andrianto, H. A., and Mardhiyyah, R. (2014). Hotspot clustering using dbscan algorithm and shiny web framework. In *2014 International Conference on Advanced Computer Science and Information System*, pages 129–132. IEEE.

Nvida TLT (2020). Nvidia transfer learning toolkit. https://developer.nvidia.com/transfer-learning-toolkit. accessed March 10, 2020.

Nvidia Deepstream (2020). Deepstream 5.0. https://developer.nvidia.com/deepstream-sdk. accessed April 28, 2020.

Nvidia IOT (2019). Smart parking detection. https://ngc.nvidia.com/catalog/containers/nvidia:deepstream$_$360d/. accessed Nov 27, 2019.

Park, B., Messer, C. J., and Urbanik, T. (1998). Short-term freeway traffic volume forecasting using radial basis function neural network. *Transportation Research Record*, 1651(1):39–47.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

Payne, H. J. and Thompson, S. (1997). Malfunction detection and data repair for induction-loop sensors using i-880 data base. *Transportation Research Record*, 1570(1):191–201.

PeMS (2014). Pems. http://pems.dot.ca.gov/. accessed June, 2014.

Petrolo, R., Loscri, V., and Mitton, N. (2017). Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms. *Transactions on Emerging Telecommunications Technologies*, 28(1):e2931.

Qu, L., Li, L., Zhang, Y., and Hu, J. (2009). Ppca-based missing data imputation for traffic flow volume: A systematical approach. *IEEE Transactions on intelligent transportation systems*, 10(3):512–522.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

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.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):19.

Shah et al. (2020). Building a real-time redaction app using nvidia deepstream. https://devblogs.nvidia.com/real-time-redaction-app-nvidia-deepstream-part-1-training/. accessed Feb 12, 2020.

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.

Smith, B. L., Scherer, W. T., and Conklin, J. H. (2003). Exploring imputation techniques for missing data in transportation management systems. *Transportation Research Record*, 1836(1):132–142.

Spark (2020). Spark 2.4.5. https://spark.apache.org/docs/latest/index.html. accessed Feb 08, 2020.

Sun, Z., Jin, W.-L., and Ng, M. (2016). Network sensor health problem. *Transportation Research Part C: Emerging Technologies*, 68:300–310.

Susmelj, I., Agustsson, E., and Timofte, R. (2017). Abc-gan: Adaptive blur and control for improved training stability of generative adversarial networks. In *International Conference on Machine Learning (ICML 2017) Workshop on Implicit Models*, volume 5.

Tan, H., Feng, G., Feng, J., Wang, W., Zhang, Y.-J., and Li, F. (2013). A tensor-based method for missing traffic data completion. *Transportation Research Part C: Emerging Technologies*, 28:15–27.

Turner, S., Albert, L., Gajewski, B., and Eisele, W. (2000). Archived intelligent transportation system data quality: Preliminary analyses of san antonio transguide data. *Transportation Research Record: Journal of the Transportation Research Board*, 1719(1):77–84.

Turochy, R. E. and Smith, B. L. (2000). New procedure for detector data screening in traffic management systems. *Transportation Research Record*, 1727(1):127–131.

Vanajakshi, L. and Rilett, L. (2004). Loop detector data diagnostics based on conservation-of-vehicles principle. *Transportation research record*, 1870(1):162–169.

Wang, Z. and Oates, T. (2015). Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Wang, Z., Yan, W., and Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *2017 international joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE.

Wells, T. J., Smaglik, E. J., and Bullock, D. M. (2008). Implementation of station health monitoring procedures for its sensors, volume 1.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE.

Wu, L., Liu, C., Huang, T., Sharma, A., and Sarkar, S. (2017). Traffic sensor health monitoring using spatiotemporal graphical modeling. In *Proceedings of the 2nd ACM SIGKDD Workshop on Machine Learning for Prognostics and Health Management*, pages 13–17.

Yao, B., Chen, C., Cao, Q., Jin, L., Zhang, M., Zhu, H., and Yu, B. (2017). Short-term traffic speed prediction for an urban corridor. *Computer-Aided Civil and Infrastructure Engineering*, 32(2):154–169.

Yeh, R. A., Chen, C., Yian Lim, T., Schwing, A. G., Hasegawa-Johnson, M., and Do, M. N. (2017). Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5485–5493.

Yin, W., Murray-Tuite, P., and Rakha, H. (2012). Imputing erroneous data of single-station loop detectors for nonincident conditions: Comparison between temporal and spatial methods. *Journal of Intelligent Transportation Systems*, 16(3):159–176.

Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. S. (2018). Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5505–5514.

Zhao, H., Gan, C., Rouditchenko, A., Vondrick, C., McDermott, J., and Torralba, A. (2018). The sound of pixels. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 570–586.

Zhuang, Y., Ke, R., and Wang, Y. (2018). Innovative method for traffic data imputation based on convolutional neural network. *IET Intelligent Transport Systems*, 13(4):605–613.