2021

Applying Machine Learning Techniques to Improve Safety and Mobility of Urban Transportation Systems Using Infrastructure- and Vehicle-Based Sensors

Zubayer Islam
University of Central Florida

APPLYING MACHINE LEARNING TECHNIQUES TO IMPROVE THE SAFETY AND MOBILITY OF URBAN TRANSPORTATION SYSTEMS USING INFRASTRUCTURE- AND VEHICLE-BASED SENSORS

By

ZUBAYER ISLAM
BS, Bangladesh University of Engineering and Technology, 2017
M.S. University of Central Florida, 2020

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Civil, Environmental and Construction Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2021

Major Professor: Mohamed Abdel-Aty

ABSTRACT

The importance of sensing technologies in the field of transportation is ever increasing. Rapid improvements of cloud computing, Internet of Vehicles (IoV), and intelligent transport system (ITS) enables fast acquisition of sensor data with immediate processing. Machine learning algorithms provide a way to classify or predict outcomes in a selective and timely fashion. High accuracy and increased volatility are the main features of various learning algorithms. In this dissertation, we aim to use infrastructure- and vehicle-based sensors to improve safety and mobility of urban transportation systems. Smartphone sensors were used in the first study to estimate vehicle trajectory using lane change classification. It addresses the research gap in trajectory estimation since all previous studies focused on estimating trajectories at roadway segments only. Being a mobile application-based system, it can readily be used as on-board unit emulators in vehicles that have little or no connectivity. Secondly, smartphone sensors were also used to identify several transportation modes. While this has been studied extensively in the last decade, our method integrates a data augmentation method to overcome the class imbalance problem. Results show that using a balanced dataset improves the classification accuracy of transportation modes. Thirdly, infrastructure-based sensors like the loop detectors and video detectors were used to predict traffic signal states. This system can aid in resolving the complex signal retiming steps that is conventionally used to improve the performance of an intersection. The methodology was transferred to a different intersection where excellent results were achieved. Fourthly, magnetic vehicle detection system (MVDS) was used to generate traffic patterns in crash and non-crash events. Variational Autoencoder was used for the first time in this study as a data generation tool. The results related to sensitivity and specificity were improved by up to 8% as compared to other state-of-the-art data augmentation methods.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Background

The field of transportation engineering is rapidly advancing with the use of different innovative sensing technologies. Real-time sensor data is being used in research and development to improve the quality of life. Historically saved data can also help to understand the cause of certain unpredictable events like crash. Machine learning models can aid in this regard to predict and classify different events. In this dissertation, we focus on identifying research gaps in certain transportation applications that can be improved based on the use of machine learning models on sensor data from the infrastructure and/or vehicles.

There have been studies on two fronts in this regard. The first is mostly related with traffic patterns and roadway conditions. These are necessary to control traffic signals and investigate the reasons that could lead to safety concerns. The sensors can range from magnetic loop detectors to cameras at intersections as well as weather stations. These can be termed as infrastructure-based sensors. Secondly, it is also important to understand the individual vehicle dynamics to account for safety and mobility in a smart city environment. To this end, the research is concentrated on the real-time sensing units on vehicles; the On-board Units (OBUs). The technology is relatively new and therefore expensive to integrate into most cars as of today. While it is imperative that within a decade or two the system ultimately finds its way to all vehicles, the intermediate years are crucial in terms of performance that can be achieved later on. Smartphone sensors can provide us with similar information contained in an OBU and can be an effective alternative as will be discussed in Objectives 1 and 2a. Every year new chipsets are deployed in smartphones which results in more accurate and efficient sensing. With the advent of 5G, these can be taken to the next level with ultra-fast speed and connectivity. Objective 2b introduces a data augmentation method for crash data analysis. Since crash and non-crash samples in such dataset is hugely imbalanced, it is necessary to use a robust data augmentation method. The use of variational autoencoder is the first research to this end to the best of the knowledge of the author. Finally, Objective 3 relates to predicting signal phasing and timing from detector events. We believe this method can reduce the dependance on traditional signal retiming methods to a great extent.

Objectives

Objective 1: Using smartphone data to estimate vehicle trajectories with machine learning models:

As technology is moving rapidly towards automation and connectivity, it is of paramount importance to predict vehicle trajectories ahead of time. This not only enhances safety but also ensures mobility in a connected and automated environment. Previous studies have shown that, given the previous trajectory, the future trajectory can be estimated. But this method suffers from considerable drawbacks in case of intersections. This objective presents an integrated method of estimating vehicle trajectories for both general roadway segments and intersections. A lane change detection system is taken as an indicator of intersection turning movement estimation and corresponding vehicle trajectories are estimated accordingly. This method is implemented by a mobile application. Therefore, the system is of low cost but effective as it is able to bring partial automation and connectivity to the vehicles that are not automated. Sensor readings are taken periodically which are first filtered with a low pass filter to zero out any high frequency noise and then fed into a machine learning model to detect lane changes. Finally, vehicle trajectory is estimated using Chebychev's polynomial.

Objective 2a: Using novel technique to balance smartphone data and applying it to identify transportation modes:

A data augmentation technique is presented that can improve the classification of transportation modes when the training data is insufficient. The proposed method uses a variational autoencoder (VAE) based synthetic data generation algorithm for smartphone data. Often the data collected by individuals for research is limited due to practical constraints. The algorithm discussed would aid in generating similar data from a handful of collected data to give a substantial dataset for any machine learning models. We propose a VAE, the decoder of which can help generate this synthetic data. We show that the synthetic data closely follows the pattern of the real data. We also show that classification accuracy is improved with the use of this type of data. Our method would also be a useful tool to boost the samples of an underrepresented class in a dataset. The detection of activity recognition using smartphone sensors could be applied to multiple aspects of vehicle to pedestrian P2V systems and smart mobility.

Objective 2b: Evaluation of the data balancing technique to augment crash data:

In this objective, we present a data augmentation technique to reproduce crash data. The dataset comprising crash and non-crash events are extremely imbalanced. For instance, the dataset used in this objective consists of only 625 crash events for over 6.5 million non-crash events. Thus, learning algorithms tend to perform poorly on these datasets. We have used variational autoencoder to encode all the events into a latent space. After training, the model could successfully divide the crash and non-crash events in the latent space. To generate data, we sampled from the latent space containing crash data. The generated data was compared with the real data from different statistical aspects. It was also compared to some of the minority oversampling techniques like SMOTE and ADASYN. The results were also compared with

GAN framework for generating data. Crash prediction models based on Logistic Regression and Support Vector Machine were used to compare the generated data from the different models. Overall, variational autoencoder (VAE) showed excellent results compared to the other data augmentation methods.

Objective 3: Signal Phasing and Timing (SPaT) Prediction with high resolution detector data from ATSPM:

Signal retiming methods have been traditionally used to improve traffic flow at intersections by reducing delay and improving level of service at signalized intersections. This can often be a lengthy process that includes several iterations of various optimization methods. Since traffic volumes can be easily obtained at intersections, a real-time signal timing prediction based on deep learning algorithms is proposed that will take various traffic flow parameters as input and find optimal timing parameters. Detector data available has been used to calculate the traffic flow metrics at several intersections. This processed data is then used to predict the signal timing and phasing for the next six cycles with reasonable accuracy. Seventeen intersections from two distinct corridors have been used in this study. One of the corridors runs adaptive signal control and the other corridor runs actuated signal control. The proposed CNN-LSTM model shows that cycle length can be accurately predicted with an MAE of 7 to 16 seconds and phase duration can be predicted with an MAE of 3 to 8 seconds. The trained model was also successfully validated at an unknown intersection with an MAE of 12.7 seconds.

Dissertation Structure

In this dissertation, we will first discuss the relevant studies under the specified objectives in Chapter 2. Research related to objective 1, 2 and 3 have been completed and presented in Chapters 3, 4, 5 and 6, respectively. The results are also discussed extensively in each chapter and possible extensions of the work is also suggested. Finally, Chapter 7 presents the concluding remarks and takeaways from the dissertation.

CHAPTER 2: LITERATURE REVIEW

This chapter describes the previous effort relating to each objective presented in Chapter 1. The novel contribution proposed is then highlighted along with the key differences to earlier research.

Previous work related to Trajectory Estimation (Objective 1)

In this section, we will discuss the current state of trajectory estimation. This will be followed by reviewing the methods that have been used in detecting lane changes since this is a integral to our proposed trajectory estimation. Finally, the important contributions in our work will be underlined.

Trajectory estimation

Trajectory prediction has been studied in different branches of robotics as well as computer vision. Quite some methods are available that have been used to predict trajectories of vehicles (Lefèvre et al., 2014). The most popular of these models are the kinematic or dynamic models as presented by Ammoun and Nashashibi (2009). Some studies focused on low-dimensional manifold models to account for a minor subset of trajectories that are possible at intersections (Ammoun and Nashashibi, 2009). Recurrent neural network (RNN) has also been widely used particularly the long short term memory (LSTM) model to understand the complex dynamics of vehicle motion (Kim et al., 2017). The concept of occupancy grid map was taken a step further in these studies. Further increase in accuracy is obtained using multiple model filter involving the Bayesian technique (Kaempchen et al., 2004). The Monte Carlo sampling for assessing statistical threat is also noteworthy in this regard (Wiest et al., 2012). There are quite few machine learning models to predict vehicle trajectories as well. The Gaussian mixture model (Wiest et al., 2012) has been studied and also trajectories have been learned with the Gaussian process as well (Laugier et al., 2011; Tran and Firl, 2014) . Some other models include more sophisticated calculations (Brand et al., 1997; Gindele et al., 2015). Most of the machine learning techniques involve computer vision. For vehicles, this means separate on-board units with camera and some computing power as well. Smartphone data has been mostly neglected in this regard except in few cases. The importance of smartphone sensors and sensor fusion have been emphasized in several work nonetheless (Kanarachos et al., 2018). Transportation modes has been identified using such data (Xia et al., 2014). Human movement states have also been estimated using the accelerometer and magnetometer sensors of the smartphone (Iwashita and Shimanuki, 2018). Deep neural networks have been used to identify vehicle movement directions (Hernández Sánchez et al., 2018). There has been some recent work in the fields of connected vehicles that also involve trajectory prediction (Lin et al., 2018), most of these are in simulation environments (Rahman et al., 2019). Connected vehicle systems are also mostly implemented on simulation environments. Few research efforts use smartphone sensors to predict vehicle trajectories.

Detecting Lane Change Maneuvers

The first objective relies on identifying lane change. Therefore, we present some of the work that has been done to detect lane changes. Most of the related work has been in computer vision fields for automated vehicle studies. It involves finding a suitable time for lane change (Nilsson et al., 2016). Takahashi and Ninomiya (1996) used an extended Kalman Filter was used to model the lane markings to search within a specified area in the image so that far lane boundaries are searched with a smaller area than closer lane boundaries, thus reducing the impact of noise. Rose and Bevly augmented this work by fusing an Inertial Measurement Unit with camera vision in an Extended Kalman Filter (Rose and Bevly, 2009). Cameras on the smartphone has also been widely used in this regard even though this consumes substantial battery. Of the studies using low-power smartphone sensors, the activity recognition tool is the most prominent. Most of these also have been widely implemented by Android and iOS. Smartphone sensors have also been used to recognize road conditions and accidents (Engelbrecht et al., 2015). It has also been used to analyze driving behavior and to provide safety information to the drivers (Welch, 1967). The summary is presented in the Table 1.

Table 1 Summary of literature review relating objective 1

Reference	Objective	Method	Sensor
Ammoun and Nashashibi (2009)	Trajectory prediction and collision estimation	Bicycle Model based on linear Kalman Filter	GPS Module
Kim et al. (2017)	Predicting trajectories of surrounding vehicles	LSTM	Radar, Camera
Kaempchen et al. (2004)	Integration of individual filters to predict Stop and Go	Interacting Multiple Model	Laser scanner, Radar, Camera
Wiest et al. (2012)	Probabilistic model of trajectory	Gaussian Mixture Model	GPS
Laugier et al. (2011)	Risk estimation by estimating trajectories	Sensor fusion technique with FCTA	Onboard Unit
Tran and Firl (2014)	Trajectory estimation at urban intersection	Monte Carlo	Onboard Unit
Gindele et al. (2015)	Trajectory prediction to aid decision making	Expected Maximization of Bayesian Model	Simulation
Kanarachos et al. (2018)	Driver Monitoring	Review of important work	Smartphone
Hernández Sánchez et al. (2018)	Vehicle movement detection	DNN	Smartphone accelerometer

Reference	**Objective**	**Method**	**Sensor**
Lin et al. (2018)	Vehicle Trajectory Estimation	DNN	Smartphone
Nilsson et al. (2016)	Safe lane change maneuver possibility	Threshold based inter-vehicle gap estimation	Simulation
Rose and Bevly (2009)	Lane detection and tracking	Polynomial counding curve	IMU

Proposed Contribution

These aforementioned studies are an indicator that the use of smartphone sensors in the field of transportation is quite limited. In this objective, the authors have used sensors such as accelerometer, magnetometer, gyroscope, to develop a cost efficient, simplistic and user-friendly system. These sensors can be fused together to obtain useful information such as accurate driving behaviors as shown in (Kanarachos et al., 2018). It also shows how the smartphone can be used as an integrated platform in the fields of heterogeneous information sources and deep learning.

The author in this objective also uses some techniques to fuse the accelerometer and magnetometer sensors to obtain useful parameters such as azimuth, roll and pitch. Moreover, this objective also proposes a system that is readily applicable in most of the smartphones today. The aim of such a system is to design a car-independent system which does not need vehicle mounted sensors measuring turn rates, gas consumption or tire pressure. In contrast to state of the art, this work uses no external sensors, resulting in a cost efficient, simplistic and user-friendly system with the potential for high market penetration.

<u>**Literature Review on Activity Recognition (Objective 2a)**</u>

In this section, the previous work on activity recognition is summarized in three sections. We firstly discuss previous and recent works on activity recognition not limiting to smartphone sensors. Use of Variational Autoencoder to augment data in the computer vision field has been highlighted thereafter. Thirdly, relevant work to sensor data augmentation has been highlighted.

Transportation Mode Recognition

Mode or activity recognition has been studied under various branches under various names. It has been often called Human Activity Recognition (HAR), Locomotion Activity recognition (LAR), Transportation Mode, etc. HAR is mostly investigated under Applied Biological Science where it is important to identify the activity of a patient using wearable sensors Lau et al. (2018)**.** LAR is studied under Transportation where it is important to distinguish between different road users (Gu et al., 2017). Even though studied under different names from different perspectives, the underlying principle remain the same: recognize activity from sensor data. The earliest work go back to 1999 where Foerster et al. (1999) first tried to detect posture and motion by accelerometer. Most of the studies were after the extensive popularity of the smartphone in the year 2007 (Kakihara). Various sensors and devices have been used thereafter. These mostly include video/image sensors, wearable sensors, social network sensors and wireless signals *(26)*. Surveillance cameras were used by Bodor et al. (2003) to identify daily activities. With the wide availability of sensor chips, data was collected from wearable sensors and mobile phones and it was analyzed using different methods (Bhattacharya and Lane, 2016). There were studies based on statistical methods (Madabhushi and Aggarwal, 1999) initially which soon took a deep dive into learning algorithms (Lara and Labrador, 2012). Convincing accuracy was reported from many of the studies (Bulling et al., 2014; Chavarriaga et al., 2013; Hassan et al., 2018; Lara and

Labrador, 2012; Yin et al., 2008) . This was later followed by deep learning techniques (Wang et al., 2019a). Further studies were carried out to account for multimodal sensor technique (Chen et al., 2015; Zhang and Sawchuk, 2011). Stacked denoising autoencoders were used in (Fuqiang et al., 2018) to remove noise and train the model at the same time. We summarize the important works in the Table 2 below.

Table 2 Summary of literature review on activity recognition

Reference	**Objective**	**Method**	**Sensor**	**Field**
Lau et al. (2018)	Telemedicine middleware platform	KNN, DT, SMO	Smartphone Sensor	Biological Science
Lin et al. (2016)	Patient Handling Activity	KNN	Wearable Sensor	Biological Science
Gu et al. (2017)	Locomotion Activity Recognition	Neural Network	Smartphone Sensor	Indoor Localization
Bodor et al. (2003)	Activity Recognition	Kalman Filter	Surveillance Camera	Outdoor Localization, Tracking
Bhattacharya and Lane (2016)	Gesture, Transportation Mode	RBM	Smart Watch	DNN in resource limited fields
Madabhushi and Aggarwal (1999)	Tracking Activity	Bayesian	CCD Camera	Tracking human activity
Chen et al. (2015)	Human Activity	Single layer feature selection framework	IMU	Feature selection scheme
Fuqiang et al. (2018)	Locomotion Activity Recognition	Denoising Autoencoder	Smartphone Sensor	Fitness tracking, aged care
Ordóñez and Roggen (2016)	Human Activity	RNN, CNN	Body-worn Sensors	Hybrid model activity recognition
Zheng et al. (2016)	Indoor Activity	CNN, SAE	IMU	Hybrid time series modelling technique
Liu et al. (2016)	Indoor Activity	CNN, RBM	Smart watch, Smartphone	Automated generation of daily activity routine

Variational Autoencoder

Variational Autoencoders (VAE) were first introduced by Kingma and Welling (2014). The main challenges of why a vanilla autoencoder could not be used to generate data were addressed in this objective. It consists of an encoder and decoder just like an autoencoder, but the loss term and the bottleneck are modified. The reparameterization trick and the special loss function described in this study were noticeable to overcome the challenges. The loss function comprises two terms: the negative log likelihood with a regularizer. Since then, VAEs have found applications in different fields. It has been used to learn images, labels and captions (Pu et al., 2016), to detect anomaly (An and Cho, 2015), for text classification Xu et al. (2017) etc. Different improved version of the VAEs were also available (Sønderby et al., 2016, 2017 #367). (Kusner et al., 2017; Sønderby et al., 2016)In this study, we aim to use VAE for generating sensor data. To the best of the knowledge of the authors, this is the first time VAEs were studied from a sensor data standpoint.

Sensor Data Augmentation

Image data augmentation has been gaining popularity in recent time (Frid-Adar et al., 2018). Studies have shown much promise and that given a small number of image set, different synthetic and accurate images can be generated (Perez and Wang, 2017). Based on the success in computer vision fields, augmentation techniques have been studied for synthetic sensor data generation as well. CNN was used by Um et al. (2017) and LSTM by Steven Eyobu and Han (2018) for wearable sensor data augmentation. RNN was also suggested in (Wang et al., 2018) since these models have an internal memory state. GANs were also used in a couple of studies. The first by Alzantot et al. (2017) trained GANs on LSTM-based generator. The generator and discriminator were trained separately. Wang et al. (2018) first trained both generator and discriminator parallelly to achieve a more realistic approach to data augmentation. Therefore, we can classify previous work into two groups:

- Oversampling based on statistical modelling such as SMOTE, ADASYN
- Oversampling based on deep learning such as GAN, CNN

Our contribution is towards the deep learning architectures.

Proposed Novel Contribution

In this objective, we look at activity recognition from a different perspective of data augmentation. While notable accuracy is reported in the literature, there has been very few studies regarding data augmentation of sensor data. This can alleviate the problem of data scarcity that is quite common in this field. In fact, different studies rely on collected data which are often not enough for learning algorithms.

Survey of Previous Work Related to Crash Data Augmentation (Objective 2b)

Crash Prediction and Data Augmentation

Various methods have been used over the past two decades to predict real-time crash likelihood. In most of the previous studies, the traffic data is aggregated at 5-minute intervals. At the start of the century there were several statistical methods that began to gain popularity. Initially case control logistic regression was studied by Abdel-Aty et al. (2004) which was improved upon with log-linear and Bayesian logistic models (Wang et al., 2019c; Wang et al., 2015). Lately, there has been increasing work using learning algorithms like Support Vector Machine (SVM) (Basso et al., 2018; Yu and Abdel-Aty, 2013), Long Short Term Memory (LSTM) (Li et al., 2020b), Random Forest (Lin et al., 2015) to predict crashes. Huang et al. (2020) used CNN to predict crashes in Interstates and found better results than shallow models. Bao et al. (2019) also used CNN to model citywide short-term crash risk prediction and reported that CNN was able to capture local spatial correlation.

Traffic data related to crashes can be highly imbalanced since generally only none or a handful of crashes occur on a segment each year. With the advancement of more safety features and warning systems in recent vehicles, the crashes are expected to be reduced even further. If we downsample the non-crash data to get a sample size similar to a crash data sample, extensive non-crash events are neglected. Therefore, it is necessary to upsample the crash data to properly train it with any model, otherwise the model will be skewed to non-crash events. Some studies tend to upsample the data with Synthetic minority Oversampling Technique (SMOTE) (Li et al., 2020b). Another popular method Adaptive Synthetic Sampling (ADASYN) (He et al., 2008) has not been used to augment crashes to the best of the knowledge of the authors. In this objective we have also upsampled the crash data with ADASYN and we have compared our results from the VAE model to ADASYN as well as SMOTE. Qing et. al. recently applied DCGAN to augment crash data (Cai et al., 2020) and the results show how deep learning data augmentation techniques can be competitive to the statistical models like SMOTE. To summarize, we present Table 3. As can be interpreted from this table, most studies have used the conventional data balancing methods like SMOTE.

Table 3 Crash data augmentation techniques in the literature

Reference	**Classification Model**	**Balancing Method**
Ahmed and Abdel-Aty (2011)	LR	Matched-case control
Xu et al. (2012)	Conditional LR	Matched-case control
Zheng et al. (2010)	Conditional LR	Matched-case control
Shi and Abdel-Aty (2015)	Bayesian LR	Matched-case control
Yu et al. (2016)	LR, negative binomial	Matched-case control
Yu et al. (2018)	Bayesian LR	Matched-case control
Xu et al. (2013)	Binary Logit	Matched-case control
Basso et al. (2018)	SVM, LR	SMOTE
Parsa et al. (2019)	SVM, PNN	SMOTE
Yahaya et al. (2019)	RF	SMOTE
Li et al. (2020b)	LSTM-CNN	SMOTE
Li et al. (2020a)	LSTM-RNN	SMOTE
Zhou et al. (2019)	Regression	SMOTE-ENN
Cai et al. (2020)	SVM, LR, ANN, CNN	DCGAN
Yin et al. (2019)	LR	SMOTE, under-sampling, Matched-case control
Elamrani Abou Elassad et al. (2020)	SVM, MLP	SMOTE
You et al. (2017)	SVM,	SMOTE, ADASYN
Sun et al. (2020)	XGBoost	Random Over Sampler
He et al. (2018)	MLP	Supervised data synthesizing
Yuan et al. (2019)	LSTM-RNN	SMOTE, matched case control
Peng et al. (2020)	RF, MLP	Youden Index
Wang et al. (2019b); Wang et al. (2019c)	BPNN	SMOTE
Abou Elassad et al. (2020b)	BL, kNN, SVM, MLP	SMOTE
Elamrani Abou Elassad et al. (2020)	SVM, MLP	SMOTE
Ke et al. (2019)	SVM, AdaBoost	SMOTE

Reference	Classification Model	Balancing Method
Abou Elassad et al. (2020a)	SVM, MLP, RF	SMOTE
Parsa et al. (2019)	SVM, PNN	SMOTE
Basso et al. (2020)	SVM, LR	SMOTE, Random over sampling

Proposed Novel Contribution

In this objective, we propose Variational Autoencoder to augment crash data. We have compared our results to SMOTE and ADASYN from the metric of specificity and sensitivity, which are important parameters to gauge the performance of data augmentation methods. We also compared it to DCGAN method (Cai et al., 2020). Overall, VAE outperforms all three with respect to the performance metrics.

Literature Review on Signal Phasing and Timing Prediction (Objective 3)

This section presents the previous work related to signal phasing and timing prediction. It is one of the earliest areas of research in transportation and remains to be a debated topic. Historically, signal control has mostly been studied from an optimization point of view. The literature review shows extensive optimization techniques and various mathematical formulation of the signal timing. More recently, big data analytics has also been used in the mix to identify signal timing patterns. In this chapter, traditional methods of signal control are first discussed. Recent studies relating to signal control are presented thereafter which can be divided into two broad categories from a methodology point of view: mathematical modeling-based signal control and deep learning-based signal control. These methods are used whenever an intersection needs retiming due to underperformance from mobility standpoints. Several studies proposed optimizing methods directly relating to connected vehicles as well as autonomous vehicles. Furthermore, specific application-based studies are also found, such as those related to evacuation, reduction of fuel consumption, etc. These will also be discussed in separate sections.

Traditional Traffic Signal Control Systems

Traffic signal control systems have evolved tremendously over the years but broadly it can be divided into four types from an operation point of view (Gordon et al., 1996; Zhao and Tian, 2012):

- Pre-timed (or fixed time)
- Fully actuated
- Semi Actuated
- Adaptive

Pre-timed or fixed time signal controllers serve each of the available phases at an intersection for a set duration regardless of demand. While this is reasonable for off-peak hours, the throughput is considerably decreased during peak hour operation. Another ideal case for these types of operations could be an intersection in which all the phases have high demand.

Actuated traffic signal control systems utilize vehicle and pedestrian detectors to activate a phase only when there is a demand. The green time of the activated phase would also vary depending on the number of vehicles detected. Fully actuated traffic control systems have detectors for all phases, whereas semi actuated traffic signal control only has actuators for the minor street.

For fully actuated systems, the green intervals of each phase can vary within a range of predefined values. It can also be extended if a vehicle is detected. This is known as green extension or passage time. The phase will automatically terminate after a set time called gap time. Phases can also be skipped if there is no demand. Pedestrian phases are also served based on detection or pedestrian calls. Walk intervals are fixed unless there is coordination.

Semi actuated signals do not have actuators for one of the phases. Usually, this is the major through movement that is always served within a cycle while the other phases are served based on actuation. If there is any unused phase duration of the actuated phases, it is transferred to the major movement.

Although actuated signal control provides great flexibility than pre-timed control, it is restricted in certain parameters such as minimum/maximum green, passage time, etc. The impact of this restriction is that actuated signal control cannot always adapt to real-time changes in traffic. This resulted in the third-generation urban traffic control systems (UTCS) (Zhao and Tian, 2012; Zheng and Recker, 2013) such as Sydney Coordinated Adaptive Traffic System (SCATS), Split Cycle Offset Optimisation Technique (SCOOT), Optimization Policies for Adaptive Control (OPAC), real-time traffic adaptive signal control system (RHODES), etc. SCATS and SCOOTs optimize the signals at a central level to find optimal parameters for the entire network, while OPAC and RHODES try to optimize at the intersection level.

From an application perspective, there can be three types of signal control (Gordon et al., 1996):

- Isolated
- Arterial
- Grid

Isolated signal controls are used for intersections that are remote from other intersections and therefore do not benefit from coordination. Arterial signal controls are applied at the corridor level when intersections along a roadway benefit from coordination, while grid signal controls are found in urban areas. Based on the necessity, each of these signal control can use pretimed, semi-actuated or fully actuated systems.

Mathematical Modelling-Based Signal Timing Optimization

The research in traffic signal control can be divided into three broad categories such as, central, hierarchal, and decentralized signal timing approaches. Central signal timing approaches considers all signals in a given network and as such the optimization solution is a global minimum. Hierarchal approaches divide the network in several levels with several objectives at each level. This helps in achieving local optimization solutions at intersection level as well as global solutions at the network level. Whereas decentralized optimizations decompose the entire network into smaller regions and each of these are optimized individually. While this helps in finding the local minimum, but the network level performance is not considered.

Central Signal Timing Approaches

Central signal timing approaches has been studied historically with a view to improving throughputs in oversaturated conditions by dynamically controlling traffic lights (Abu-Lebdeh and Benekohal, 1997; Chang and Sun, 2004; Longley, 1968). Genetic algorithms were proposed in contemporary research work by Hajbabaie and Benekohal (2011). A traffic signal coordination method using ant colony optimization was also proposed by Putha et al. (2012). Some of the optimization methods (Cantarella et al., 2006) focuses on generating a uniform pattern of signal timings instead of real-time solutions. There were also several studies that tried to improve some other factors like lane layout, traffic assignment, etc. Certain studies also focused on improving signal timing and traffic assignment problems (Beard and Ziliaskopoulos, 2006). Signal timing optimization with route choice was studied by Sun et al. (2006). The route options were investigated to lower the travel cost along with optimizing signal timing by Ukkusuri et al. (2013).

Hierarchal Signal Timing Approaches

Hierarchical solutions factor the network into a multi-level optimization problem. Sims and Dobinson (1980) divided the control strategies into two classes: strategic and tactical. Strategic levels deal with ten intersections to determine suitable timings based on average prevailing traffic conditions while tactical controls only try to optimize one intersection. Mauro (1990) also proposed a two levels of control strategy. The area level controller handles medium- and long-term forecasting over the entire network while local level controller determines the optimum length of signal phases in real time. Three level controller was put forward by Head et al. (1992). Dynamic network loading level forecasts network level travel demand, network flow control level then uses this information to optimize green time while intersection control level determines appropriate phase intervals. Gartner et al. (2001) also proposed a three-layered control strategy. The synchronization layer is used for network level optimization, coordination layer adapts the cycle length based on real traffic conditions while local control layer adapts the phase timings within a cycle.

Decentralized Signal Timing Approaches

Decentralized approaches divide the network into multiple sub-levels with varying number of intersections. These approaches can be real-time but can find suboptimal solutions that are not scalable. Porche and Lafortune (1999) decomposed the network using dynamic programming. Each sub-level functioned on their own without coordinating with another but major streets could be provided higher weights. Priemer and Friedrich (2009) divided the network into individual intersections and tried minimizing the total queue length at each intersection. Lee et al. (2013) proposed an algorithm that allocates green signal time to the approaches with longer cumulative travel time. Goodall et al. (2013) proposed predictive microscopic simulation algorithm (PMSA) that decomposes the network into individual intersections. Each intersection is handles separately assuming connected vehicle environment. Coordination among different intersections were not considered.

Deep Learning-Based Signal Control

Artificial intelligence and machine learning have also been used to improve traffic control performance. Broadly the learning algorithms can be classified into reinforcement learning, neural network and fuzzy logic systems.

Reinforcement Learning for Signal Control

Reinforcement learning was first used by Thorpe and Anderson (1996) for traffic signal control. SARSA (Sutton, 1996) was applied in this study which was evaluated with three different representations of a specific state. Multi-agent reinforcement learning was also studied by Wiering et al. (2004); Wiering (2000). Each vehicle communicates its waiting time to the nearest traffic light. The goal of the system was to minimize the total waiting time. Abdulhai et al. (2003) applied Q-learning as a traffic controller. The reward function in this study was actually a penalty which was the total delay between two successive decisions. Wunderlich et al. (2008) used reinforcement learning as a tool to create distributed control since the proposed Longest Queue First (LQF) algorithm became increasingly complex in multi-intersection scenario. Feature based reinforcement learning was proposed by Prashanth and Bhatnagar (2010).

Neural Network for Signal Control

Neural network based adaptive controllers were also studied. Spall and Chin (1997) developed a neural network controller to optimize the system. Simultaneous perturbation stochastic approximation (SPSA) was used to model the weights of the neural network. Yin et al. (2002) developed a fuzzy neural network model to predict the traffic flows in an urban street network. The reported improvement was up to 30% against a conventional NN. Choy et al. (2003) used a simultaneous perturbation stochastic approximation neural network (SPSA-NN) that was tested in different complex situations in the Singapore Central Business District. Teodorović et al. (2006) developed a system based on neural network and dynamic programming that makes teal time decisions to extend current green time. Chao et al. (2008) used extended neural network (ENN) theory for crossroads. This work was more geared to detecting appropriate traffic patterns accurately. Nagare and Bhatia (2012) argued that NN usually obtains local solutions and therefore suggested using combined optimization methods.

Fuzzy Logic System for Signal Control

Fuzzy Logic System (FLS) was used on a single intersection by Pappis and Mamdani (1977). The system was evaluated with actuated method and better results were reported. Nakatsuyama et al. (1984) applied FLS to two adjacent intersections. Based on fuzzy logic controller (FLC), managing and controlling phase length dynamically was suggested by Lee et al. (1995). FLS was also used by Favilla et al. (1993), to control isolated intersection with two-way street. Some other studies that have used FLC are Wei et al. (2001), Murat and Gedizlioglu (2005), Hu et al. (2007), Zeng et al. (2007).

Signal Optimization in Connected Vehicle Environment

Several studies have also been proposed with respect to signal timing strategies in a connected environment. Quite a few these studies is devoted to using individual vehicle information such as speed, location, etc. to optimize signal timings (Beak et al., 2017; Choi et al., 2016; Day et al., 2017; Goodall et al., 2013; Gradinescu et al., 2007; He et al., 2012; Hu et al., 2015; Lee et al., 2013; Priemer and Friedrich, 2009; Yang et al., 2018). The main objectives also vary across the different papers. While some studies try to reduce the average vehicle or platoon delay (Beak et al., 2017; Goodall et al., 2013; He et al., 2012; Hu et al., 2015; Yang et al., 2018), other try to reduce the queue length (Priemer and Friedrich, 2009) or total travel time (Choi et al., 2016; Day et al., 2017; Lee et al., 2013). These considered the market penetration rates of CVs as well as the different traffic demands. Other studies have focused on improving the discharge time of vehicles such as minimization of evacuation time (Wu et al., 2007; Yan et al., 2009), travel time (Cai et al., 2012), queue length (Ahmane et al., 2013) , total delay (Pandit et al., 2013; Stebbins et al., 2016), etc.

Signal Optimization in Autonomous Vehicle Environment

Optimization using autonomous vehicles (Araghi et al., 2015) have also been studied. Dresner and Stone (2004) proposed requests from individual vehicles to reserve a time at an intersection with and without AVs (Dresner and Stone, 2006). This was extended to a network of intersection later on (Hausknecht et al., 2011). Lower levels of automation was also investigated from a similar reservation request policy (Au et al., 2015). Some non-reservation-based strategies were also studied (Dresner and Stone, 2006; Kamal et al., 2014; Lee and Park, 2012; Li et al., 2014; Zohdy and Rakha, 2012).

Signal Optimization for Platoon-Based Vehicles

Platoon based signal timing optimizations were also suggested to over the computational complexity of the CAV studies (Liang et al., 2018) that can also be used on real-time scenarios. A longitudinal trajectory control algorithm was proposed based on the leading vehicle of the platoon while also taking in consideration the vehicle dynamics such as acceleration and deceleration. Previous studies related to the same also considered non-CAV vehicles (Xie et al., 2012).

There are certain also certain works that uses Signal Phasing and Timing (SPaT) to find optimal fuel economy (Asadi and Vahidi, 2010) or to guide a vehicle through a signal (Koukoumidis et al., 2011). Vehicle planning scheme for energy efficiency has also been studied (Mahler and Vahidi, 2014) for arterials that predicts SPaT based on historical data. SPaT predictions based on GPS data from several buses has been studied by Fayazi et al. (2014) for fixed signal timings. Floating vehicle data from other sources have also been investigated to estimate fixed timing signal parameters by using speed measurements (Ban et al., 2009; Fayazi and Vahidi, 2015; Wang and Jiang, 2012). Smartphone camera has also been used to detect signal states (Koukoumidis et al., 2011) used historical SPaT from a single intersection to calculate future times.

Proposed Contribution

None of these previous studies have used real-time detector information for SPaT prediction which provides accurate granular traffic flow for all phases. The other drawback is each study is limited to a single intersection and as such the reproducibility of such methods on other intersections cannot be understood. Additionally, such methods fail to capture the real-time vehicle demand fluctuations in the field and would usually suggest an average solution. Based on our literature review, detector data has not been yet studied to predict SPaT. Due to directly correlating detector data with signal timing, this method has the potential to provide real-time accurate signal timing predictions than any of the previous studies.

CHAPTER 3: TRAJECTORY ESTIMATION USING SMARTPHONE SENSORS

Islam, Z., Abdel-Aty, M., 2021a. Real-Time Vehicle Trajectory Estimation Based on Lane Change Detection using Smartphone Sensors. *Transportation Research Record*, 0361198121990681.

<u>Introduction</u>

In a smart and connected environment, it is imperative to detect potential conflicts as early as possible and to warn drivers and pedestrians as soon as possible. This can only be achieved if the vehicle trajectory can be properly estimated several seconds in advance. By definition, vehicle trajectory information has the present location of the car as well as future location estimates. Given such estimates, it will be possible to detect V2V and P2V conflict scenarios. While most of the research revolves around getting vehicle trajectories at straight roadway segments, it is also important to have some predictive features for intersections. The authors of this objective present an integrated system that can predict vehicle trajectories at both. Separate calculations are involved for calculating through, left and right movements at intersections, and only through movements are calculated for other locations. The authors also decided not to use a base map since this brings the additional task of manually labeling different intersections and roadway segments. Instead, intersections are estimated based on previous lane changes and speed. Also, there are no origin and destination nodes in the proposed system, and therefore, predicting driving trajectories is a challenging task. It is more like reading the mind of the driver. For this case, this objective uses two features to estimate intersections and whether the driver will turn left and right. A vehicle that made a left lane change and slowed down more likely has a higher probability to be approaching an intersection and will make a left turn. Moreover, this system would work on any three or four-leg intersections since the calculations are independent of any basemap. The system is simple in itself so that it is possible to implement very quickly and efficiently. The reason for this simplicity is the use of smartphone sensors to achieve this prediction. It is a rare occasion to find a car without a smartphone. While there have been several studies involving on-board units (OBUs) and on-board equipment (OBEs), smartphone data has largely been neglected. The OBUs and OBEs have different sensors including microelectromechanical systems (MEMS) and Inertial Navigation System (INS) which are also present in smartphones nowadays. With the development of technology, these sensors in smartphones are also becoming more and more accurate. A study in 2012 (Blum et al., 2012) shows that the GPS error is within 10 meters and the compass error is 10 degrees. The authors experimented with some smartphones of 2019 (specifically the Samsung Galaxy Note 10, the OnePlus 6, and the iPhone XS) which are much more accurate, and the GPS error is under 1 meter in most cases. The error was calculated from the difference in reading between smartphone device and ground truth based on GNSS data. The expected error rate in GPS can increase when inside a building or near obstacles that are interfering with the GPS signal. The built-in error measurement tool of GPS chipsets in the phones can be used to get an understanding of the magnitude of the error. If the error is above a threshold, a warning can be issued to the driver that the measurements are not stable due to a large GPS error. The phones also come with multiple other sensors like the accelerometer, magnetometer, gyroscope, etc. which are also very precise. The usage of these sensors in smartphones eliminates the need for individual OBUs and is, therefore, a cost-effective solution for Connected Vehicles.

The chapter is organized as follows: The first section explains how the dataset for this objective was prepared and organized, the second describes the data preprocessing techniques that were used in generating the dataset. The methodologies and algorithms used to detect lane changes and predict trajectories are presented in the next along with the results. Finally, concluding remarks are made and more interesting topics of research that can be built from this objective are discussed. The work presented in this chapter has been published at Transportation Research Record (Islam and Abdel-Aty, 2021a).

Data Collection

This objective uses data from a smartphone device to conduct the analysis. Smartphone data was collected from 4 trips that covered 64 miles. The total number of data points in the raw data was 15504. The data was collected by two participants. In total the dataset had 94 lane changes. Readings from the accelerometer, magnetometer, gyroscope, and GPS were collected. The phone was placed on a mount to prevent unwanted jerking. Two Android phones were used to collect these data, the Samsung Galaxy Note 10 and the OnePlus 6. The authors developed their Android mobile app to get this data. The smartphone app architecture is shown in Figure 1.

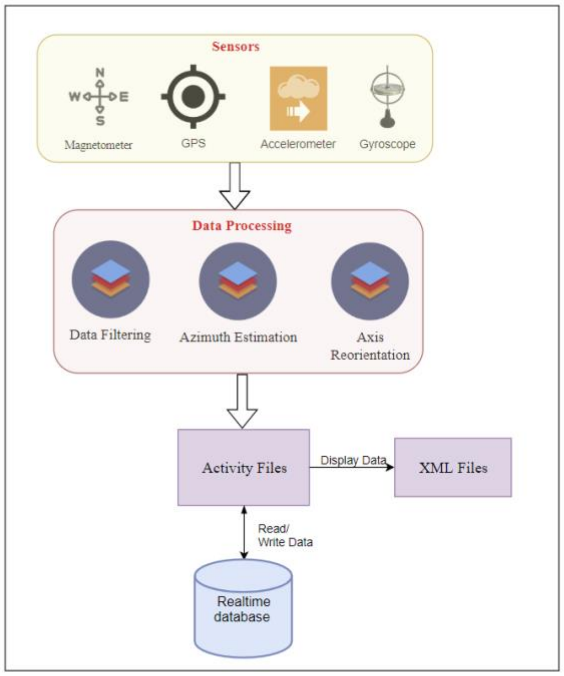


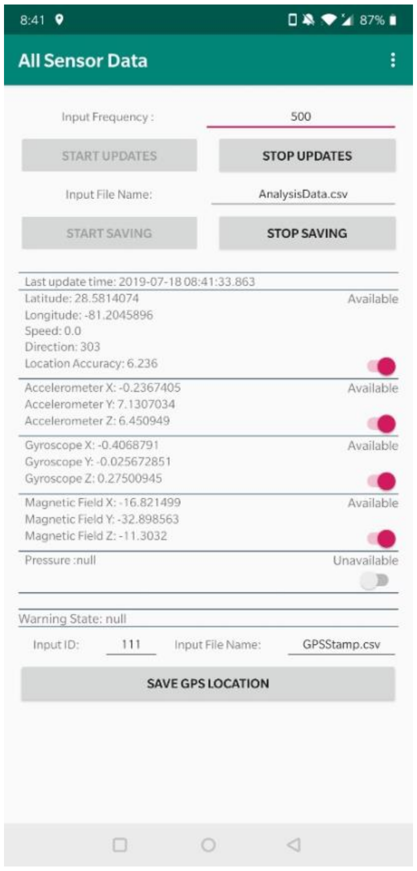
Figure 1 Android App Architecture for Data Collection

The mobile app did not only collect the data but also made other calculations whenever raw data were available. First, the app made the proper adjustment of the axis of the smartphone to the axis of the vehicle using rotation matrix. Second, the accelerometer and magnetometer data were filtered using a set of low pass filters. Finally, azimuth, pitch, and roll were estimated for each of the outputs of the low pass filters. The raw gyroscope values were also captured. A brief description of the data is summarized in Table 4. In addition, the filtered data for various sensors were also collected.

Table 4 Data collected from Smartphone

Sensor/Feature Name	**Description**	**Values Collected from Smartphone**
Accelerometer	Acceleration Profile of a vehicle	*accx, accy, accz*
Gyroscope	Rotating Profile	*gyrox, gyroy, gyroz*
Magnetometer	Strength of Earth's magnetic field	*magx, magy, magz*
GPS	Location of vehicle	*lat, lon*
Azimuth	The direction of the vehicle from the north pole	*azimuth*
Pitch and Roll	The tilt of the device	*pitch, roll*
Difference between consecutive azimuth	Rate of change of azimuth	*difference*
Magnitude	Azimuth conversion to binary	*magnitude*

The most difficult part of any data collection is the labeling. The common way of labeling a dataset is manual. The ImageNet dataset (Deng et al., 2009) was created by manually annotating images for 9 years. Even this is easier than labeling a lane change because, given a set of sensor readings, it is difficult to say which maneuver is a lane change. This research work introduces an unusual but effective way to achieve data labeling. The authors have used an extra device (called the Flic Wireless Smart Button) to label the lane changes while driving. This device was wirelessly connected to the smartphone. At the beginning of the lane change, a button was pressed to indicate a right lane change. The same button was pressed twice to indicate a left lane change. After the button press, the lane change maneuver took place. This could also be achieved if the mobile app had buttons to tap during a lane change. The main drawback of such an approach is that the accelerometer readings and gyroscope readings would change significantly due to taps on the screen. Therefore, using an external device to label the lane change was the best way to get real driving data and also to efficiently label the dataset. Figure 2 shows the app UI and the Flic wireless button used to collect data.



Input Frequency : 500

Input File Name: AnalysisData.csv

Last update time: 2019-07-18 08:41:33.863

Latitude: 28.5814074 Available
Longitude: -81.2045896
Speed: 0.0
Direction: 303
Location Accuracy: 6.236

Accelerometer X: -0.2367405 Available
Accelerometer Y: 7.1307034
Accelerometer Z: 6.450949

Gyroscope X: -0.4068791 Available
Gyroscope Y: -0.025672851
Gyroscope Z: 0.27500945

Magnetic Field X: -16.821499 Available
Magnetic Field Y: -32.898563
Magnetic Field Z: -11.3032

Pressure :null Unavailable

Warning State: null

Input ID: 111 Input File Name: GPSStamp.csv

(a) App UI (b) Flic Button

Figure 2 Android Application UI and the wireless Flic Button

Data Processing

This section explains the steps to process the raw data from the smartphone. First, the calculation of the rotation matrix is discussed since this is the primary step for getting consistent readings from the smartphone. Second, some signal processing techniques that help to eliminate noise in the sensor data are discussed. Fast Fourier Transform is introduced in the next section which shows the frequency components of any signal. Using this information a low-pass filter system can be designed that can eliminate unwanted noise in sensor data as discussed in the following section. Finally, a discussion is carried out with regards to using the processed data to get an estimation of the azimuth which is used to detect lane changes.

Rotation Matrix Calculation

The inherent problem with placing a phone in a vehicle is its orientation with respect to the vehicle. Each sensor has its own 3D axis and the vehicle also has its axis. To match these two axes in order to achieve the exact vehicle maneuver, a rotation matrix is used. The main advantage of using a rotation matrix in a smartphone application is that the phone can be placed in any orientation and still receive consistent readings. If ψ , θ and $\emptyset$ are the rotations about x-axis, y-axis and z-axis respectively, then the rotation vectors are defined as (Slabaugh, 1999)

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\psi & -sin\psi \\ 0 & sin\psi & cos\psi \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

$$R_x(\emptyset) = \begin{bmatrix} cos\emptyset & -sin\emptyset & 0 \\ sin\emptyset & cos\emptyset & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The angles ψ, θ and $\emptyset$ are also called Euler's angles. If analysis is carried out first at x-axis, then at z-axis and finally at y-axis then the resultant rotation matrix would look like

$$R = R_x(\psi)R_y(\theta)R_x(\emptyset)$$

In the above equation, R is a 3x3 matrix. This matrix, when multiplied with the sensor values along different axis converts the values to the corresponding axis of the vehicle. This ensures that the orientation in which a smartphone is placed in a vehicle does not affect the results.

Fast Fourier Transform

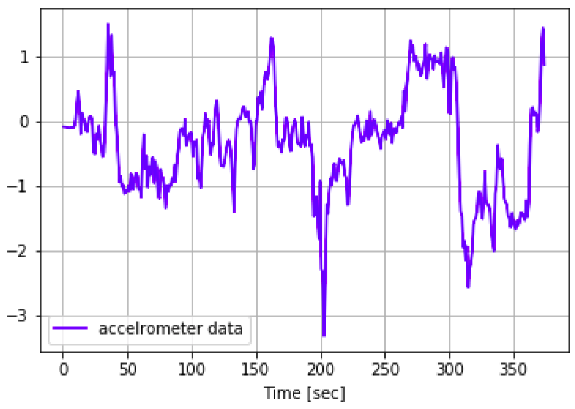
The accelerometers of smartphones are usually very sensitive and the same with magnetometers. This is an unwanted feature from the perspective of this objective because this inserts a large amount of noise to the sensors. To get an idea about the high-frequency noise of the accelerometers of the smartphone, the authors used Fast Fourier Transform (Welch, 1967) to convert the time series of accelerometer and magnetometer data to frequency series. Fast Fourier Transform (FFT) is an efficient way to calculate the Discrete Fourier Transform (DFT) of any signal in the time series. If the time series values of a signal are $x_0, x_1, \dots, x_{N-1}$, then the DFT is defined as

$$X_k = \sum_{n=0}^{N-1} x^n exp\,(-i2\pi kn/N)$$

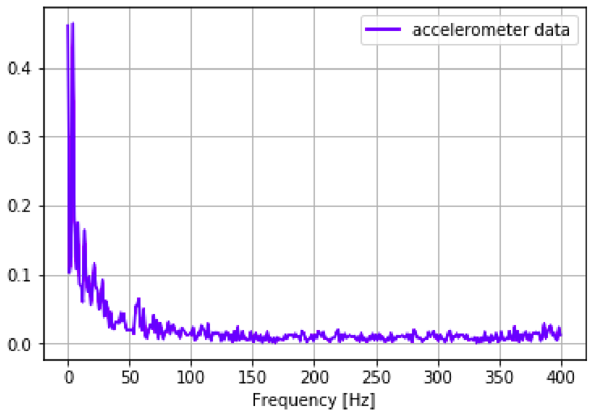
The value of k is in the range of $0,1,\dots,N-1$.

Figure 3 shows accelerometer readings before and after FFT has been applied. The graph on the left has x-axis as time and the one on the right has frequency as the x-axis. The frequency-domain graph gives an idea about the frequencies that are present in the accelerometer.

As can be seen from Figure 3 that the most significant part of the accelerometer data is that below 50 Hz. After that, it is mostly DC noise.



(a)



(b)

Figure 3 The graph on the left shows accelerometer readings in the time domain and the one on the right shows the same for the frequency domain

If we calculate the 3dB bandwidth from this plot, we get 3.28 Hz which means that the most significant part of the accelerometer data is below contained from 0 to 3.28 Hz. Formally, 3dB bandwidth is defined as the range of useful frequency when the signal amplitude gain is above 70.7% of the maximum gain possible.

Low Pass Filter

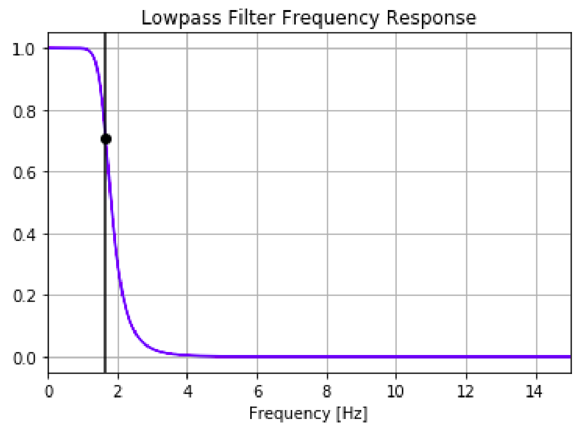
A low pass filter attenuates the high-frequency components and allows the low-frequency points to pass through (Fredendall, 1949). The transfer function of a low pass filter (LPF) is defined as

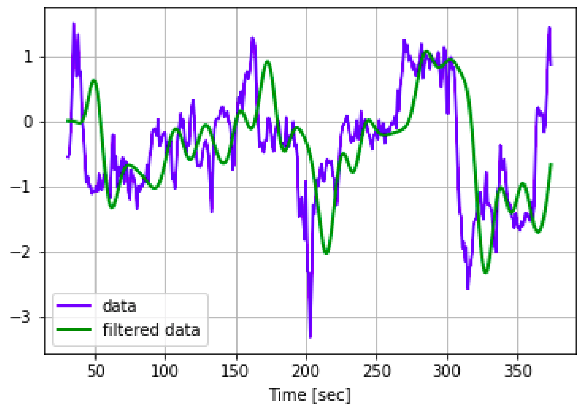
$$H_c = \frac{\omega_c}{s + \omega_c}$$

where,

$\omega_c = 2\pi f_c$ and f_c= the cut-off frequency.

The cut-off frequency for an LPF is the maximum frequency that will be passed through the filter. Figure 4 shows the LPF used in this objective. It has a cut-off frequency of 2 Hz. Figure 4 also shows the difference between accelerometer readings before and after filtering. The graph is smoother with the removal of high-frequency components.





(a) (b)

Figure 4 The graph on the left shows the Low Pass Filter used in this study and the one on the right shows the raw accelerometer and filtered data

Azimuth Estimation Using Accelerometer and Magnetometer

After proper filtering of the high-frequency components, the accelerometer and magnetometer data were used to calculate azimuth. The angle between the device's compass direction and the earth's magnetic north pole is defined as azimuth. This can be accurately estimated as shown in (Zhou et al., 2012).

As already mentioned, there are two coordinate systems; one for the vehicle (called the vehicle coordinate system, VCS) and the other is the co-ordinate system of the earth (also called geographic coordinate system, GCS). The relative rotation between these two coordinate systems can be described by three terms called azimuth, pitch, and roll. For the purpose of this objective only azimuth calculation is sufficient.

The transformation VCS and GCS are computed with the following cosine matrix

$$C_n^b = \begin{bmatrix} cos(N,X) & cos(E,X) & cos(G,X) \\ cos(N,Y) & cos(E,X) & cos(G,X) \\ cos(N,Z) & cos(E,X) & cos(G,X) \end{bmatrix}$$

where,

N, E, G are the three orthogonal axes of GCS and X, Y, Z are the three orthogonal axes of VCS.

The accelerometer readings x_a, y_a, z_a together with the magnetometer readings x_m, y_m, z_m can be converted back and forth with this formula. If the errors associated with the accelerometer is e_{ax}, e_{ay}, e_{az} and that of the magnetometer is e_{mx}, e_{my}, e_{mz}, then the conversion formulae would be

$$\begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} = C_n^b \begin{bmatrix} 0 \\ 0 \\ f_g \end{bmatrix} + \begin{bmatrix} e_{ax} \\ e_{ay} \\ e_{az} \end{bmatrix}$$

$$\begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} = HC_n^b \begin{bmatrix} cos\beta \\ 0 \\ sin\beta \end{bmatrix} + \begin{bmatrix} e_{mx} \\ e_{my} \\ e_{mz} \end{bmatrix}$$

The errors shown in these equations is mostly eliminated due to the techniques explained in previous sections.

Finally, the azimuth can be estimated using equation:

$$\psi = arctan(\frac{z_m sin\gamma - y_m cos\gamma}{x_m cos\theta + (y_m sin\gamma + z_m cos\gamma) sin\theta})$$

Where, θ is the pitch and can be calculated as

$$\theta = -arctan(x_a cos\gamma / z_a)$$

Figure 5 shows how the difference between azimuth reading from a smartphone device can aid in distinguishing between lane changing and non-lane changing instances. The fact that azimuth is the measure of direction from the north pole and that lane changing instances lead to a change in direction of driving, is an important indicator in this case. Evident from the Figure 5 is that the directions are opposite for left and right lane changing instances and it is almost constant for lane keeping instance.

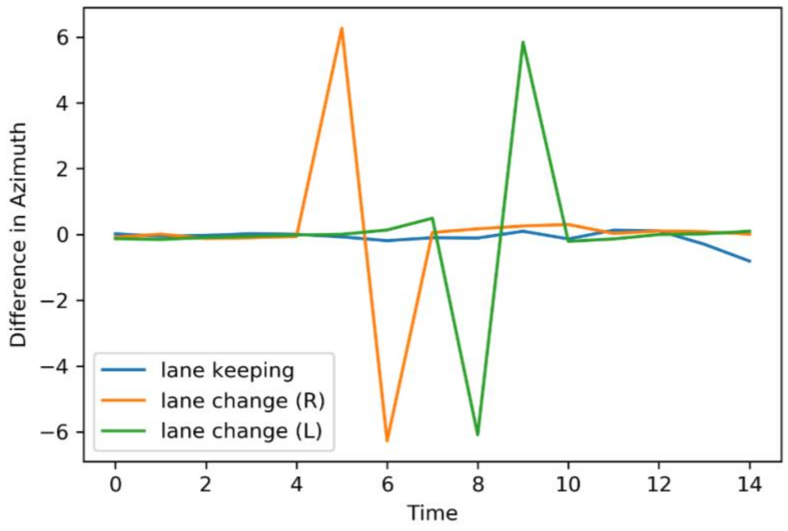


Figure 5 Difference in azimuth estimations from sensor values

<u>Methodologies and Discussion</u>

The methodologies are discussed in two sections; the first describes how various sensor values are used to identify lane changes and the next describes how the trajectory is estimated based on the previous GPS co-ordinates.

Lane Change Classification Using Machine Learning

Feature Extraction

The developed mobile application collected data from several sensors. The authors used machine learning models to detect lane changes in real time. The reason behind choosing machine learning to estimate lane changes is to capture the pattern in multiple sensors across multiple axes. In order to find out the best feature to identify lane changes, the authors carried out several feature extraction strategies.

Pearson coefficient *(78)* was evaluated on the entire dataset and the values are shown in Figure 6.

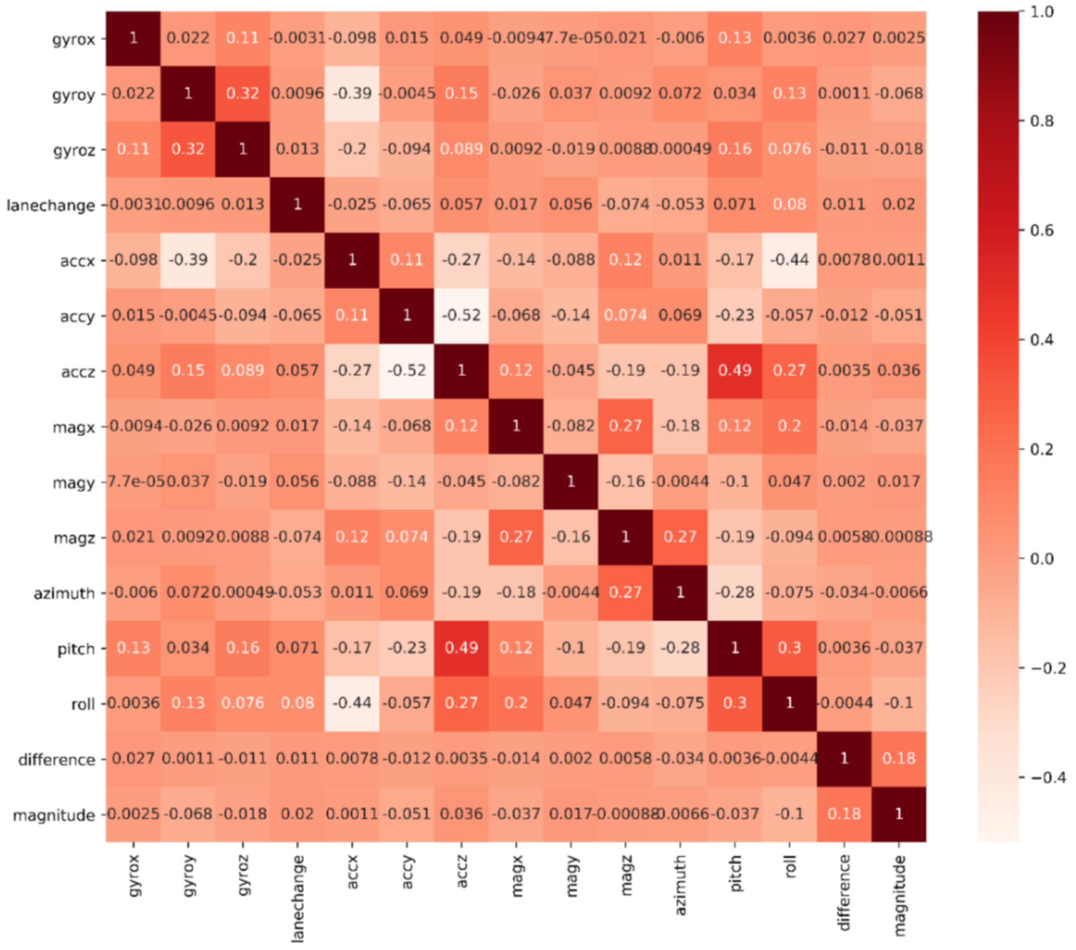


Figure 6 Pearson Correlation Coefficient across various sensors

The values for the row *lanechange* show that the most significant features are *gyroscope z, accelerometer z, magnetometer x, magnetometer y, pitch, roll, difference and magnitude.* This is an indication that identification of lane changing is dependent on the values of different sensors.

To get more conclusive results, several other feature extraction methods were carried out namely, Select Percentile, Select K-best and RFECV. Select Percentile and Select K-best are based on univariate feature selection statistical methods. Select K-best works by calculating ANOVA F-values of the provided sample and then selecting K features with highest F-values. Select Percentile also computes the F-values but selects all the features with the percentile of highest score. Recursive feature elimination (RFE) works by recursively considering smaller and smaller features. Given a model, RFE initially trains itself on the all the features and then on smaller subset of the features to find the desired number of features, RFE with cross-validation (RFECV), iterates over a loop to find the optimal combination of features. The features extracted with the above four methods are summarized in Table 5.

Table 5 Feature Extraction Methods Summary

Methods	**Optimal Features**
Pearson Coefficient	gyroscope z, accelerometer z, magnetometer x, magnetometer y, pitch, roll, difference and magnitude
Select Percentile (top 10%)	accelerometer x, accelerometer y, gyroscope y, magnitude, roll
Select K Best	accelerometer x, accelerometer y, gyroscope y
Recursive Feature Elimination	magnetometer x, magnetometer y, magnetometer z, azimuth, roll, pitch, accelerometer x, accelerometer y, magnitude

This leads us to the conclusion that accelerometer and gyroscope are an important indicator of lane change behavior. While some models suggest that magnetometer readings are also important, we can rule it out based on the fact that magnetometer readings measures the amount of magnetic flux at any place and cannot be an indicator of lane change indicator. It is also important to note that, the combination of the uncommon parameters like roll, azimuth and magnitude of azimuth are also important indicators. This can be explained with the logic that these parameters are nothing but sensor fusion of accelerometer and magnetometer. For training purposes, we have selected *gyroscope y*, *accelerometer y, difference of azimuth, magnitude of azimuth, pitch* as our features for lane change detection.

Model Description

The raw dataset that was used in this objective composed of 57 left lane changes, 37 right lane changes and 47 lane keeping instances. So that we do not make our model biased towards any individual outputs, it is essential to have a balanced dataset. There are several methods to balance dataset and we have chosen the ADASYN (He et al., 2008). Unlike undersampling or oversampling which merely copies of the same data, ADASYN generates synthetic data. It has an adaptive nature in the sense that it generates more samples for the minority class that are harder to learn. After this algorithm was applied, the new dataset had 57 left lane changes, 57 right lane changes and 53 lane keeping instances.

To successfully differentiate between lane change and non-lane change maneuvers, the authors have tried and tested different models. We tried out several models ranging including Stochastic Gradient Descent (SGD), Support Vector Classification (SVC), Gaussian Naive Bayes (GNB), AdaBoost (ADA), Decision Tree (DT), Random Forest (RF) and Extra Tree (EXTR). All the models were implemented by scikit-learn (Pedregosa et al., 2011).

Model comparison and Results

The results of all the models are summarized in Table 6. To get an understanding of the actual behavior of a model, it is necessary to not only take accuracy under consideration but also some other metrics. Precision is defined as the number of true positives to the number of total predicted positive while recall is the number of true positives to the number of total actual positive. F1 score is the weighted average of the precision and recall. In general, if precision is high, then the model does not classify anything negative as positive and if recall is high, then the model does not classify something positive as negative.

Table 6 Model Summary

Models	**Accuracy (%)**	**F1 Score (%)**	**Precision (%)**	**Recall (%)**
SGD	58	55	64	57
SVC	89	89	89	89
GNB	72	69	75	71
ADA	66	66	76	66
DT	95	95	95	95
RF	97	97	97	97
EXTR	98	98	98	98
GB	87	87	89	87
XGB	90	90	91	90

Based on all these values, Extra Tree Classifier outperforms all the other models by some margin. The precision and recall are also very high. The confusion matrix of this classifier is also shown in Figure 7. The confusion matrix helps to visualize if any of the classes has been misclassified. Based on the figure, it is clear that the model does well to classify each instance. In the labels, 0 indicate left lane changes, 1 indicate right lane changes and 2 indicates lane keeping instances

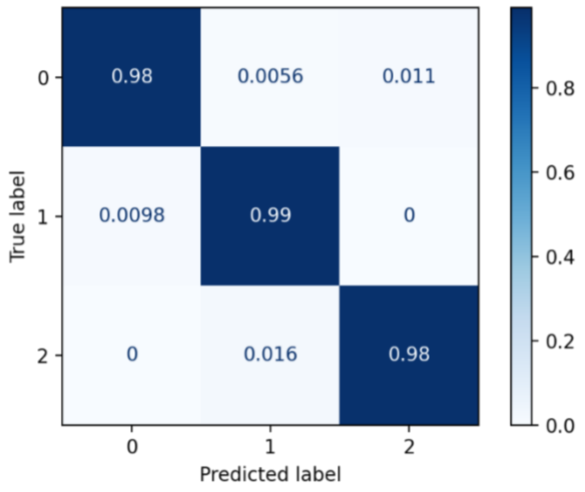


Figure 7 Confusion matrix for Extra Tree Classifier.

Another way to understand a model is also from its performance metrics across different folds of the dataset. The models were tried out with K cross fold validation (Kohavi, 1995). This gives us an idea about the performance of the model across 10 different folds of data. We are also able to obtain a mean and standard deviation from this analysis. The results are shown in Figure 8. In this plot, the mean accuracy of the cross-validation is shown with a yellow line. The box shows the standard deviation and whiskers show the entire accuracy range. SGD has a very high variance and should definitely be discarded. SVC does not have expected accuracy. ADA is performing pretty well in this case too but the most interesting find from this figure is the performance of EXTR, DT, RF, GB and XGB. All these models have low variance which indicates that the performance of these model is more consistent. Therefore, the conclusion is that each of DT, RF, EXTR, GB, and XGB can accurately classify lane-changing instances.

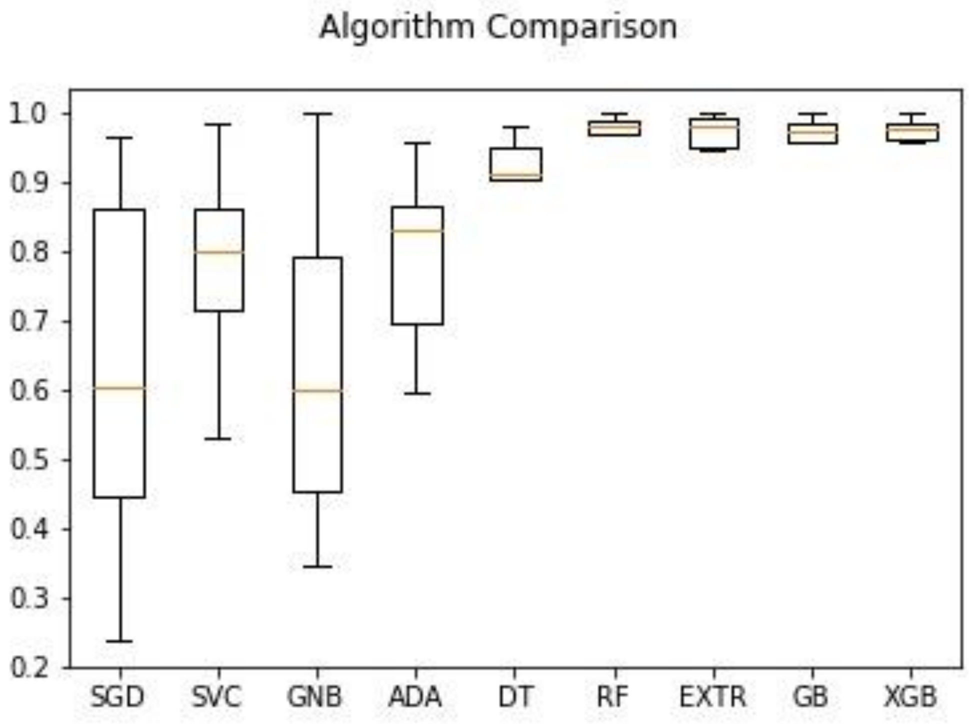
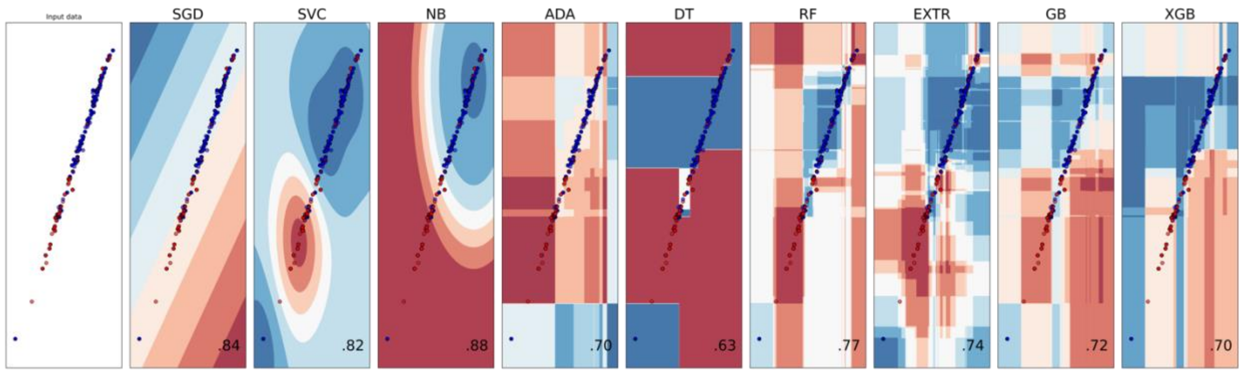


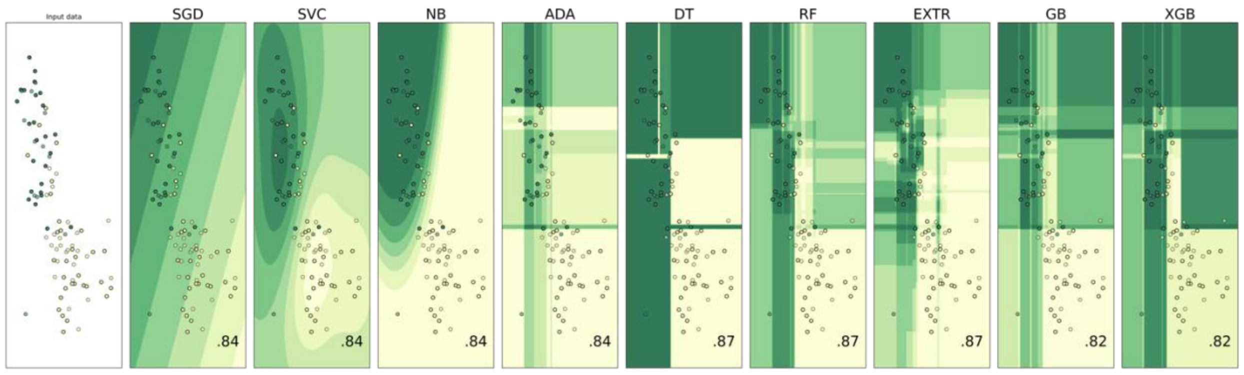
Figure 8 Comparison Between Different Models with Cross-Validation

The decision boundary of a model gives a visual understanding of the performance of a model. It also gives an idea about the cluster of the input data points and also how each model is creating a boundary to separate one from the other. This is shown in Figure 9(a) and Figure 9(b). The performance of each model is shown at the lower right of each plot. Subplot (a) shows the decision boundary for lane change and non-lane change instances. Red dots indicate lane-keeping instances and blue indicate lane-changing instances. Subplot (b) shows the decision boundary for left and right lane change instances. Red dots indicate left lane changing instances and blue indicates right lane changing instances.

The main reason that EXTR, RF, and ADA outperformed the other models can be understood from this figure. These are methods from a branch of data science called ensemble techniques. They outperform general models because these models can generate a set of hypotheses to describe the data. Detailed descriptions can be found in (Dietterich and others, 2002). Different shades indicate different percentages of the surety of the model. For the model SGD, we see that the model tries to find a linear boundary between the different classes. While this is feasible for differentiating right and left LC (Figure 9(b) SGD), this cannot quite capture the decision boundary for LC and non-LC (Figure 9(a) SGD). The decision boundary of SVC and NB forms a contour. The ensemble methods form rectangular decision boundaries around groups of data points. The size of the rectangles depends on the number of data points that can be grouped. Due to the ability of these models to form these complex and intertwined decision boundaries, it has better performance than the models with linear or continuous boundaries.



(a)



(b)

Figure 9 The input data along with the decision boundary for different models

Trajectory Estimation Using Chebychev's Polynomial

After lane change detection, the GPS coordinates can be estimated. Different polynomial fitting methods have been used in previous studies to estimate trajectories (Deo et al., 2018; Houenou et al., 2013; Woo et al., 2017). To get an estimate of the future traveling coordinates the authors used Chebyshev's polynomial (Wiest et al., 2012) that has been shown to achieve good accuracy for probabilistic trajectory prediction. Initially, the authors have used this polynomial to predict through movements. To obtain turning movements, we used logarithmic functions thereafter. No base maps have been used in this study which makes it more robust and easily applicable to other intersections. The Chebychev polynomial is defined by

$$T_n(x) = cos(n \, arccos \, x)$$

The formula can be expanded to a polynomial, even though this looks trigonometric. The first terms of the polynomial yield explicit expressions of$T_n(x)$

$$T_o(x) = 1$$

$$T_1(x) = 0$$

$$\dots$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), n >= 1$$

Any function $f(x)$ in the interval $[-1,1]$ can be approximated by this polynomial. To get the polynomial, co-efficient $c_j, j = 0,1, \dots, N-1$are to be calculated first. The equation below is used to achieve this

$$c_j = \frac{2}{N} \sum_{0}^{N-1} f(x_k) T_j(x_k)$$

The approximate formula of $f(x)$ then can be calculated as

$$f_x \approx \sum_{0}^{N-1} f(c_k)T_k(x) - c_0/2$$

The details of the calculation can be found in Press et al. (2007). This method is suitable for estimating trajectories in general roadway segments, but it fails to capture the turning movement at the intersections. Therefore, the authors use two separate formulae for capturing the right and left-turn movements at intersections. The formula is a logarithmic function that is rotated and translated by $\theta,\ \delta$ to match the current vehicle coordinate.

$$x_r = (x+\delta)cos\theta - \log[b(x+\delta)]\,sin\theta$$

$$y_r = (x+\delta)sin\theta - \log[b(x+\delta)]\,cos\theta$$

The negative logarithm of the same equation is used to capture the left turns.

$$x_l = (x+\delta)cos\theta - \log[-a(x+\delta)]\,sin\theta$$

$$y_l = (x+\delta)sin\theta - \log[-a(x+\delta)]\,cos\theta$$

The mobile app collects GPS points every second, but these are some decimal numbers. There are several techniques to convert these into functions, the most common is the Web Mercator Projection (Janssen, 2009; Kessler et al., 2017) which is widely used in almost all maps available online. The Web Mercator uses spherical formula at all scales which is different from the way Mercator is defined. If $\gamma\ and\ \theta$ are the longitude and latitude respectively, then the Web Mercator is defined as

$$x = \frac{256}{2\pi} 2^{zoom-level}(\gamma + \pi)$$

$$y = \frac{256}{2\pi} 2^{zoom-level}\left(\pi - \ln\left[tan\left(\frac{\pi}{4}\right) + \frac{\phi}{2}\right]\right)$$

After this conversion, the Web Mercator points are now ready to be fed into Chebychev polynomial for prediction. The general system algorithm is shown in the flowchart in Figure 10.

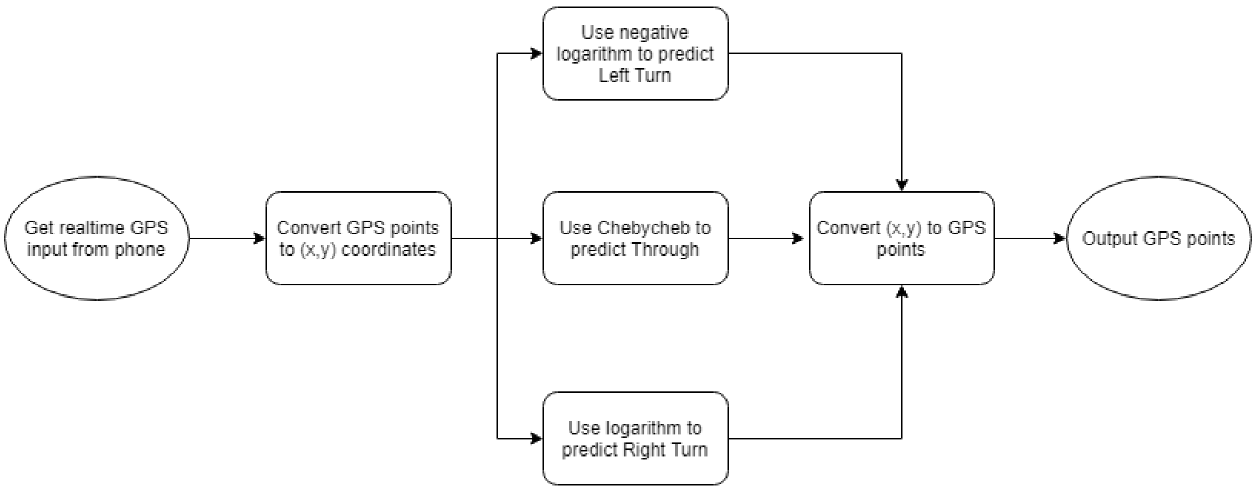

Get realtime GPS input from phone
Convert GPS points to (x,y) coordinates
Use negative logarithm to predict Left Turn
Use Chebycheb to predict Through
Use logarithm to predict Right Turn
Convert (x,y) to GPS points
Output GPS points

Figure 10 Flowchart of the system algorithm

Figure 11 shows an instance of the above algorithm based on a previous lane change. The green GPS points are the input to the system while the blue ones are the predicted trajectory

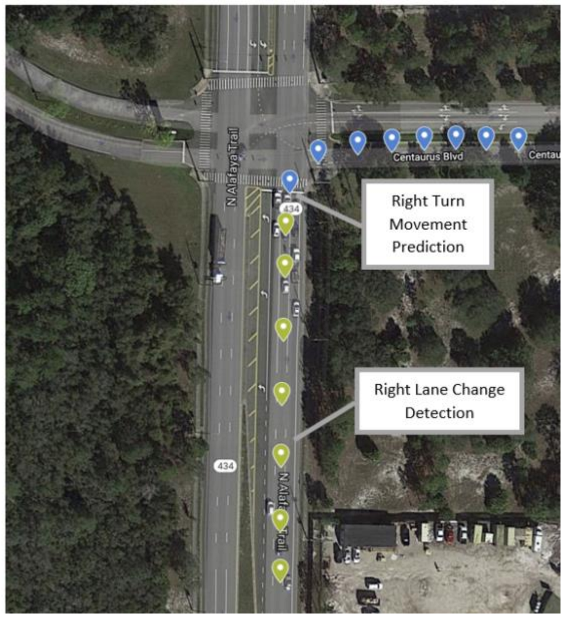


Figure 11 A particular case (the right turn prediction) of the trajectory prediction algorithm

The errors of the estimations were calculated based on the GPS error rate between the ground truth and the predicted paths. The overall summary is presented in Table 7. The results were obtained from three different intersections that had a total of nine different movements.

Table 7 GPS Error

Movements	GPS Error (in meters)
Through	9.77
Left	12.4
Right	4.64
Overall	9.27

The GPS error is plotted against time to visualize the effect of the error rates ahead of time. Figure 12 shows that the error rate for the three turning movements at a particular intersection. The error rate for the left turn is the highest. Individually, the errors tend to increase as the algorithm tries to predict more seconds into the future.

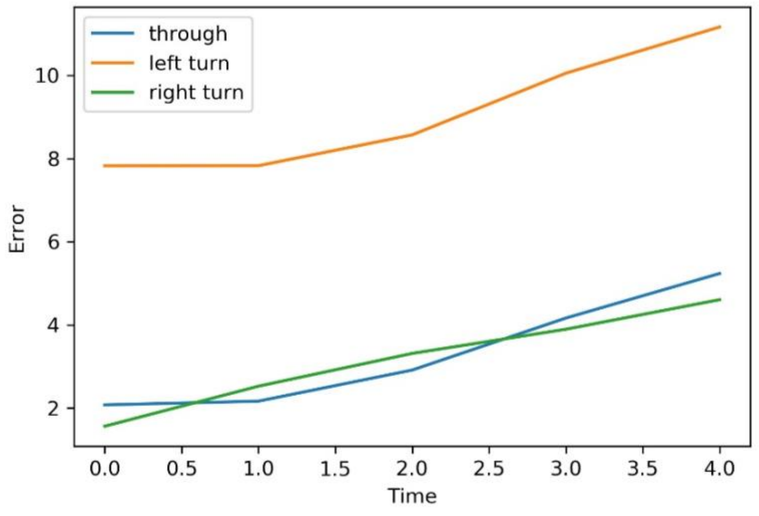


Figure 12 Error involved in future GPS estimates

Summary

This chapter introduces new techniques that have not yet been explored in the field of Connected and Automated Vehicles (CAV). The proposed system is simple in that it only uses smartphone sensors to estimate trajectories. Initially, the rotation matrix is used to match the orientation of the smartphone to the vehicle. Azimuth, pitch, and roll are also estimated from the raw sensor values. The dataset thus obtained was used to train models that can identify lane-changing instances. Experiments showed that ensemble methods like Extra Tree, Gradient Boosting perform better than linear models. Finally, the trajectory was estimated using GPS data and by taking lane-changing detection as an indicator of the turning movement at intersections. This study shows a realistic example of how a learning algorithm can be fused with traditional polynomial prediction algorithms to obtain more meaningful trajectory estimations. This type of method would be especially applicable to intersections with a large volume of turning vehicles as well as a good number of pedestrians. Parameters like time-to-collision can be calculated based on trajectory prediction to identify the safety concerns at an intersection.

The road to automation is still work in progress and while there are a few industry-grade vehicles that are level 3 automated, fully autonomous vehicles are still some time away. Even when level 5 vehicles are available, there would still be vehicles in the market that are only level 1 or 2. To address this gap in technology, a smartphone is presumably the best option. Also, since smartphones are updated every year, it is possible to get more accurate sensors with higher ratings that can adjust to extreme conditions in a car. Another important advantage of smartphones as OBUs is that they are readily movable from a vehicle and therefore are less subjected to extreme conditions of heat and cold than actual OBUs. For vehicles with lower levels of automation, such mobile-based systems could be a way to achieve partial automation.

The main contribution of this chapter is in identifying new sensor fusion data like azimuth, pitch, etc. Subtle changes in these data need to be captured to properly identify any maneuvers. Then again, too much sensitivity can ruin the original purpose of the system. The optimum method was to try different sensitivity and finally settle on the one that suits our purpose. The second part of the chapter is also significant in that it adds to the existing through movement for trajectory prediction. The algorithm can be used to get left turn and right turn trajectories at the intersection. Overall, this can be very insightful in several research fields in CV and safety. It can also be used to calculate potential conflicts between vehicles and pedestrians. Future studies could include trajectory prediction with the sensor data other than GPS as well as a comparison between more machine learning approaches for lane change detection. Moreover, the addition of a base map can further improve the trajectory prediction accuracy.

CHAPTER 4: ACTIVITY RECOGNITION WITH VARIATIONAL AUTOENCODERS

Islam, Z., Abdel-Aty, M., 2021b. Sensor-Based Transportation Mode Recognition Using Variational Autoencoder. *Journal of Big Data Analytics in Transportation* 3(1), 15-26.

Introduction

For almost the past two decades there has been tremendous progress in all forms of technologies. The development of microelectronics and chipsets made it possible to obtain highly accurate sensors that use the least of powers and that sit in the smallest of spaces. This allowed the move to ubiquitous sensing (Wang et al., 2019a) which is an active area of research that aims to extract useful information from pervasive sensors. The recognition of human activity is still at the forefront of these efforts because of its massive usefulness in the fields of medical, security, and transportation. The move to smart sensing and smart cities requires that the current mode or activity of a road user is accurately and quickly identified to ensure safety and efficient mobility. For example, if the activity can be quickly recognized, then potentially dangerous conflicts between vehicles and pedestrians or between vehicles and vehicles can be identified and the information can be relayed to the conflicting road users. Thus, the importance of classification of human activities in the transportation fields carries huge importance. While most work focus on either wearable sensor data or smartphone sensor data but there is research in the computer vision field as well (Song et al., 2010).

Although the first works on human activity recognition (HAR) go back to the ‘90s (Foerster et al., 1999), there are still many issues that motivate new algorithms. In this objective, we specifically look at one issue of the sensor data: limited data for learning algorithms. With the rapid change in technology, newer chipsets become available each year with more and improved sensors. Therefore, it often becomes necessary to repeat some earlier works with the new sensor data. Often it becomes necessary that data is collected with a new device that has never been used in the past. Most data collected by this process are usually limited and therefore, realistic accuracy cannot be obtained. In this objective, we propose a method to generate synthetic data from a sample of collected data. We also show that the synthetic data generated in this process are very similar to the real data. We demonstrate that the classification accuracy improves with the use of this synthetic data.

We propose using Variational Autoencoder (VAE) for generation of synthetic data. Previously, VAEs have been successfully used to generate samples of data in the computer vision field and our intuition was that it could prove to be useful for various sensor data as well. This objective is organized as follows: we first discuss the previous work followed by the basics of a VAE. Next, we show the architecture of our VAE and how it can generate data. Finally, the data is compared to the real data and evaluated based on several criteria.

We have used the openly available SHL dataset (Gjoreski et al., 2018; Wang et al., 2019d) for the purpose of this study. All through the objective, the term real data is used to refer to the data from the SHL dataset. The data generated by our VAE model is interchangeably referred to as generated data or synthetic data. The combination of the two is referred to as hybrid data. The work presented in this chapter has been published in the Journal of Big Data Analytics in Transportation (Islam and Abdel-Aty, 2021b).

Dataset

This objective uses the Sussex Huawei Locomotion (SHL) dataset which is openly available. The dataset is especially remarkable since it has 750 hours of labeled smartphone sensor data available from 3 different users. Each user uses three smartphones simultaneously to record data.

One is carried by the user in one hand, the second is placed in his pocket and the third in a backpack. Almost all sensor data that is available from a smartphone were collected. For this objective, we only used the sensors that are responsible for motion detection. These are an accelerometer, gyroscope, orientation, and linear acceleration. The labeled data is from eight different transportation modes; three are pedestrian modes (still, walk and run), one is bike mode and the rest three are vehicle mode (car, bus, train).

Proposed Method

Variational Autoencoders (VAE) are a type of autoencoder that has gained tremendous applications in computer vision recently. Proposed by Kingma and Welling (2014) it was quickly adapted to much of the available vision datasets. Since multiple sensor data with a fixed window is numerically the same as that of an image of the same dimension, the intuition was to use these models in sensor networks as well. To the best of the knowledge of the authors, this is the first time variational autoencoders are being investigated from a mobile sensor point of view. In this section, we will first present the basics behind a VAE and then describe the model that gave the best results for transportation modes generation.

An autoencoder is a neural net that maps its input to an output of exactly the same dimension. Towards the middle of the neural net, there is a bottleneck that is the unique feature of an autoencoder. These can help in reducing noise or getting a lower-dimensional representation of the input. A special variation of the autoencoder is the variational autoencoder. In general, all autoencoders have an encoder and a decoder. An encoder compresses the input into a latent space and a decoder decodes from this latent space. For an autoencoder, the latent space is discrete, and therefore sampling from this space would not result in anything meaningful. Variational autoencoder takes these discrete values and tries to find a normal distribution of the latent space. If we denote x as the input, z as the latent space representation and$\hat{x}$ as the output of an autoencoder, then $q(z|x)$ can be the encoder and $p(x|z)$ can be the decoder. The loss function consists of two terms; the first term penalizes the reconstruction error between the input x and the output $\hat{x}$ and the second term penalizes the error between the prior $p(z)$ and the learned distribution $q(z|x)$.

$$L = -D_{KL}\big(q(z|x)||\, p(z)\big) + \log\, p(z|x)$$

For the purpose of data augmentation, we used multiple VAEs; one for each mode of transportation. This was important since it would have been difficult to trace which latent space gives rise to which mode. From the SHL dataset, we excluded the train and subway modes because the main focus of the objective is the data from pedestrians and vehicles. The data was split into windows of 200 samples. Since the SHL dataset was sampled at 100Hz, this means that we have a window size of 2 seconds. The dataset was divided into training and testing subsets. Finally, after reshaping, there were 57784 'still' samples, 51280 'walk' samples, 6162 'run' samples, 25496 'bike' samples, 48608 'car' samples, and 28528 'bus' samples. We trained the VAEs on the training dataset and evaluated the generated data on the test dataset. The VAE model is shown in Figure 13. The number of neurons has been scaled down and actual numbers are shown above each layer.

We considered a two-layer densely connected encoder model for our VAE. The input to the encoder is the real sensor data. The real sensor data was normalized and flattened to a shape of 3200. It is then connected to the first layer of the encoder which has 800 neurons. The first layer is also densely connected to the second layer which has 800 neurons. The encoder ends in the latent layer and this is the bottleneck of an encoder. We decided to have a code size of 2 for this layer. To find the perfect fit, we also tried other densely connected layers of 20, 40, 50, 200, 400, 800, 1600, and 2000. We selected 800 because we noticed that the reconstruction loss to be the lowest for this case.

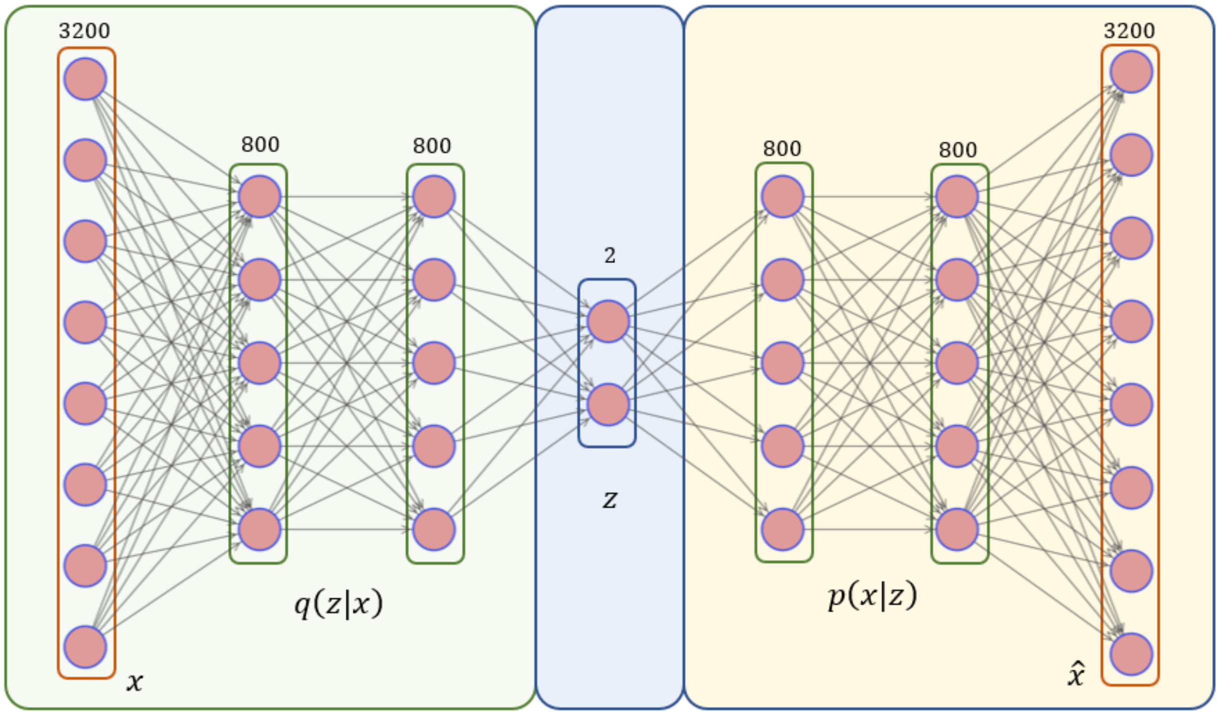


Figure 13 VAE model presented in the objective. The numbers above each layer represent the number of neurons.

The decoder takes the latent layer code and tries to reconstruct the original sensor data. For the decoder, we also had 800 neurons for the first layer and the same for the second layer. The output layer would be the same as the input which is 3200. All the neurons were densely connected to one another.

We used individual VAEs for synthetic data generation. This aided in the proper labeling of the generated data. The pseudocode for training and data generation of VAE is shown in Table 8**.** The input for training is the real data. The optimization technique used was Adam with a learning rate of 0.001. In each iteration, we also calculated loss on the test samples for visual evaluation. For data generation, we sampled from the latent space generated from the training phase and passed it through the decoder to obtain synthetic data.

Table 8 Algorithm for generating synthetic data

```
1:   **Input:** Real Sensor Data
2:   **Initialize:** decoder, encoder, prior, latent space dimensions
3:   Training:
4:   **for** the number of iterations **do**
5:       **for** the number of batches **do**
6:           input the batch into the encoder
7:           calculate loss using equation 1 and optimize over the training dataset
8:       calculate loss for the test dataset for evaluation
9:   Data Generation:
10:  **for** number of samples **do**
11:      sample from the latent space
12:      use the trained decoder from training step to obtain synthetic data
```

Experiments

Training Loss of VAE

The main objective of a VAE is the variational lower bound of the marginal likelihood of data. Marginal likelihood also includes the KL divergence loss and is the sum of overall likelihood of all data points.

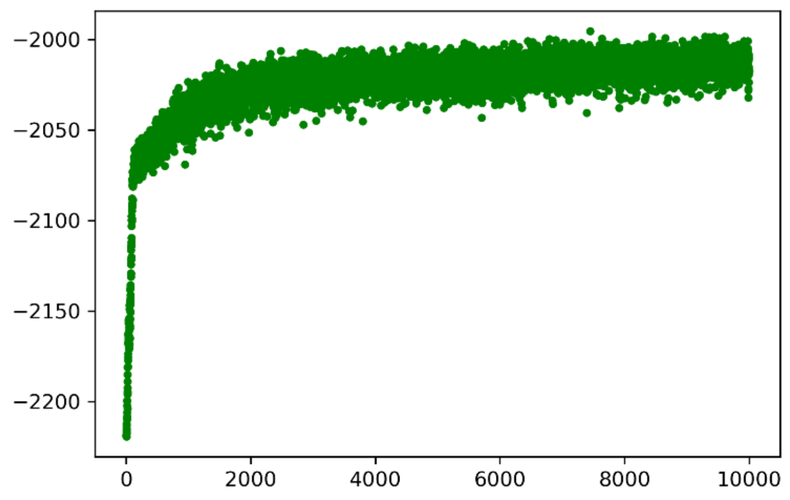


Figure 14 Training loss of the VAE model

The latter loss is also called the reconstruction loss and is responsible for generating similar data while the former is responsible to create a known distribution of the latent space. This slight restriction aids in generating data points from the latent space. The total loss is often called evidence lower bound objective (ELBO). The figure shows ELBO for 'run' data. The loss function for other activities is also similar and hence not presented. It is evident from Figure 14 that the loss function converges moving closer to zero.

Evaluation of Generated Data

In most data augmentation, the first step is to have a visual observation of the generated data (Bulling et al., 2014). We investigated our generated data from different visual criteria. Local visual evaluation is used first to see the pattern of the generated data during training of the VAEs. Secondly, global visual evaluation is used to understand the range of values between the synthetic data and real data. This is also referred to as the quality of the synthetic data (Bulling et al., 2014). Finally, we used memory independence evaluation to prevent overfitting of the data.

Local Visual Evaluation

In this evaluation, we randomly selected a synthetic sample of data and compared it with the test data. Since there are multiple sensors available, we decided to pick the gyroscope x-axis sensor values. The results from the other sensor axis are not shown but it follows the trend shown in Figure 15.

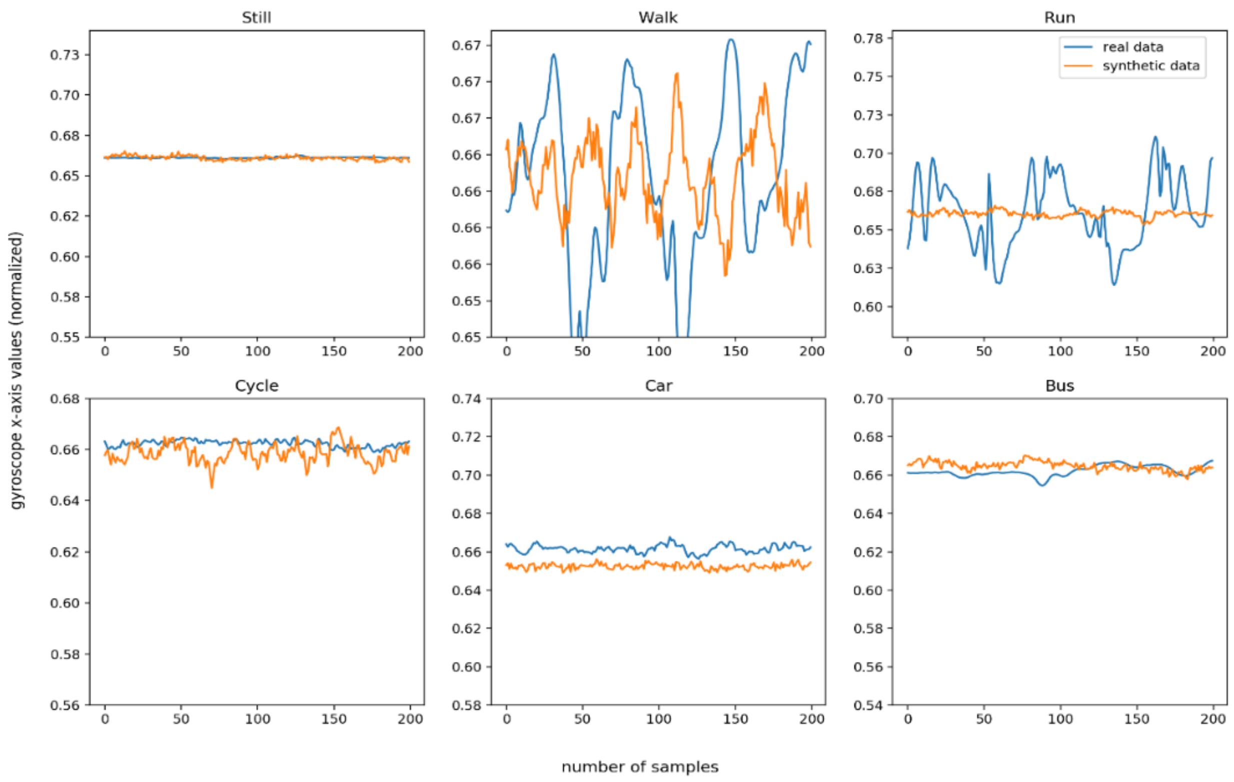


Figure 15 Comparison of real and synthetic data from VAEs

Real data samples from the SHL dataset are shown in blue and generated data samples are shown in orange. Almost all the synthetic data show some pattern of the real data except the data for 'run'. In fact, when 'run' and 'walk' data are plotted together, it seems that the decoder of our VAE generates 'walk' samples instead of 'run'. This can be attributed to the fact that walking and running are closely related activities (the main difference is the speed which we did not include in this objective) and the VAEs think both of these are the same. The other reason could be the smaller number of ‘run’ data. The sample size of ‘run’ data in the SHL dataset was three times lower than ‘walk’ data. The other observation from this data is that the most variation in sensor values is for ‘run’; the difference in peak to peak values is about 0.20 while for the other data it is 0.02.

Global Visual Evaluation

The global evaluation takes all the data into consideration and not just one sample as in the local evaluation. We illustrate this in Figure 16 with a grouped boxplot where we show the real data in blue and synthetic data in orange. This shows that the generated data has the same mean and interquartile range as with the real data. This figure also gives insight into the fact that vulnerable road users like pedestrians and bicyclists (with the modes ‘walk’, ‘run’ and ‘bike’) have higher standard deviation than the road users in ‘car’ or ‘bus’. This is also consistent with our real-life experience. Generally, phones are subject to most movements during the pedestrian phase and the least movements during the vehicle phase.

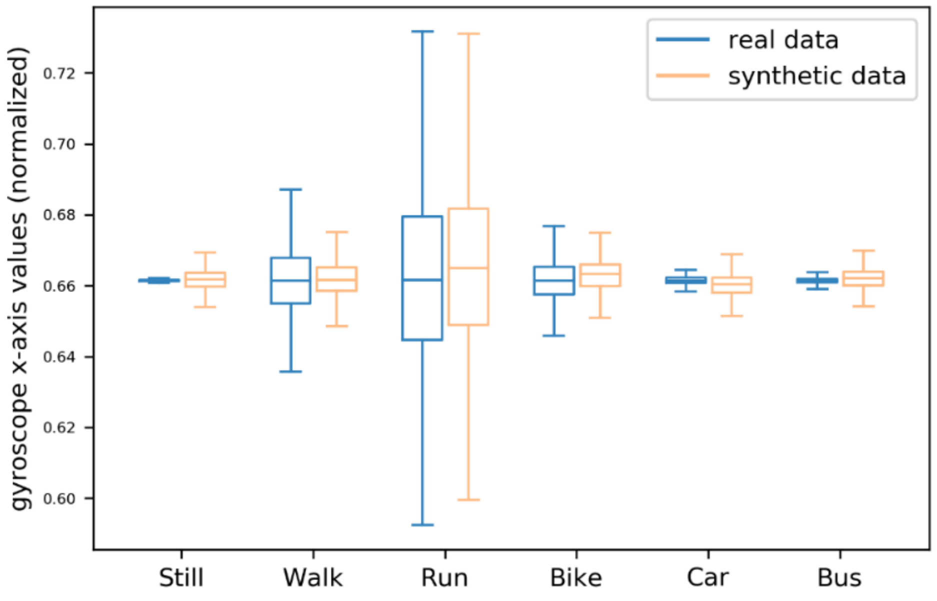


Figure 16 Boxplot showing the mean and standard deviation of real data and generated data

Memory Independence Visual Evaluation

This evaluation helps us to decide if there is any overfitting tendency of the synthetic data while training the VAE model. Figure 17 shows that VAEs can successfully discard overfitting when learning different data.

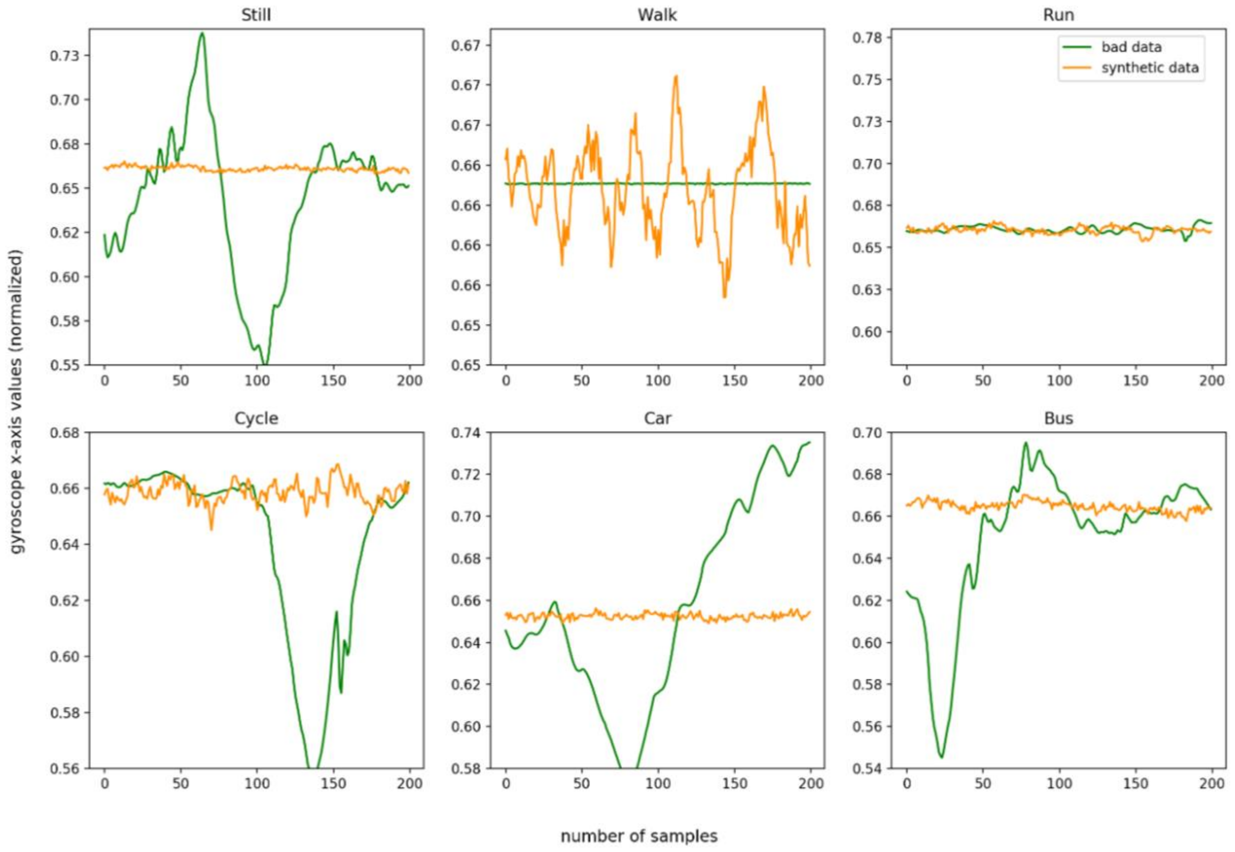


Figure 17 Memory Independence Evaluation of generated data and noisy data

The figure shows synthetic data in orange and bad data. The bad data shown was collected from the training dataset. This shows that our VAEs can avoid learning from the bad data. In general, time-series data that has multiple sensors involved tend to have the impurity problem where one data can be mixed with the other but VAEs can successfully overcome this given that the sample of bad data is only a handful.

Data Distribution Evaluation

The distribution of the synthetic data should closely match that of the real data. We have used violinplot to compare the two. The mean and spread of the data can also be approximately understood from this plot. Figure 18 shows this comparison.

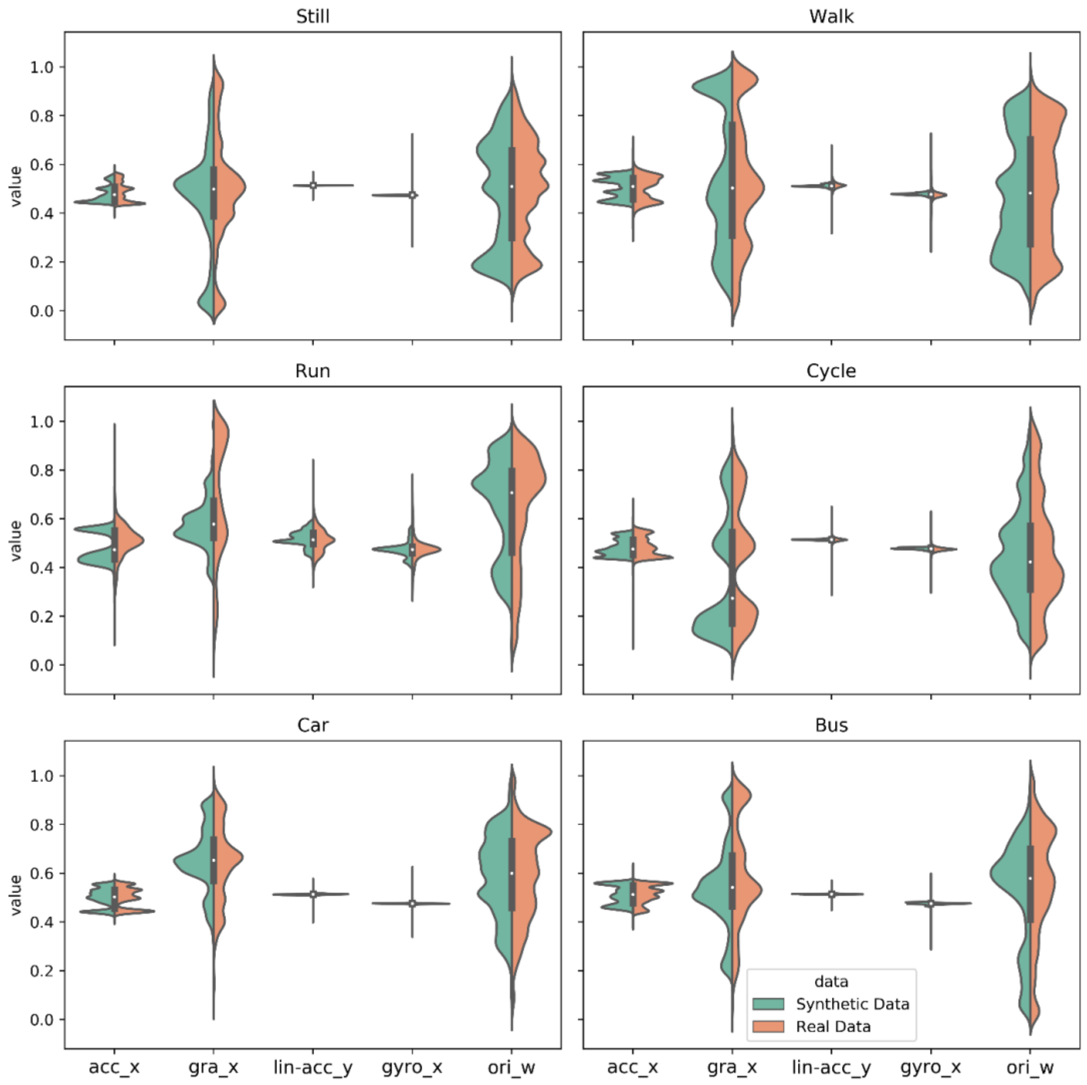


Figure 18 Data Distribution Evaluation

The real dataset is shown in green and the synthetic dataset in orange. We show the distribution of one axis of the five different sensors in Figure 18. The five values shown are accelerometer x-axis (acc_x), gravity x-axis (gra_x), linear acceleration y-axis (lin-acc_y), gyroscope x-axis (gyri_x) and orientation (ori_w). For each of the six different activities, we can see that all of the distributions closely match the real dataset.

Statistical Evaluation

To conclusively prove the similarity between the real data and synthetic data, three different statistical tests were also used. 20 samples were randomly selected from each dataset for the tests. t-Test was used to evaluate if the mean of the two datasets are statistically different while Levene test and Kolmogorov Smirnov test was used to compare variance and distribution. The p-values of all the tests for each of the 16 features used are presented in Table 9.

Table 9 Statistical Comparison between Real Data and Synthetic Data

	Real Data				**Synthetic Data**				**p-values**		
Variable	**Min**	**Max**	**Mean**	**STD**	**Min**	**Max**	**Mean**	**STD**	**t-test**	**Levene-test**	**Ks-test**
acc_x	0.11	0.96	0.51	0.08	0.27	0.59	0.48	0.06	0.13	0.17	0.21
acc_y	0.29	0.9	0.54	0.05	0.46	0.6	0.52	0.02	0.66	0.83	0.06
acc_z	0.28	0.72	0.49	0.03	0.44	0.52	0.48	0.01	0.84	0.34	0.13
gyro_x	0.51	0.83	0.66	0.03	0.57	0.74	0.67	0.02	0.68	0.17	0.01
gyro _y	0.23	0.61	0.42	0.03	0.37	0.47	0.42	0.01	0.07	0.08	0.04
gyro _z	0.28	0.77	0.47	0.04	0.4	0.58	0.47	0.03	0.76	0.34	0.07
ori_w	0	0.99	0.45	0.21	0.02	0.82	0.35	0.24	0.05	0.03	0.19
ori _x	0.06	0.99	0.64	0.23	0.06	0.93	0.64	0.2	0.36	0.57	0.97
ori _y	0	0.99	0.5	0.27	0.12	0.98	0.66	0.17	0.3	0.34	0.19
ori _z	0.01	1	0.53	0.21	0.14	0.92	0.53	0.17	0.66	0.57	0.92
gra _x	0	1	0.61	0.35	0.01	0.99	0.42	0.41	0.41	0.57	0.24
gra _y	0.04	1	0.67	0.24	0.34	1	0.58	0.1	0.54	0.34	0.27
gra _z	0	1	0.43	0.16	0.05	0.75	0.39	0.16	0.4	0.57	0.89
l_acc_x	0.13	0.94	0.53	0.06	0.4	0.67	0.53	0.03	0.46	0.57	0.13
l_acc _y	0.34	0.82	0.51	0.05	0.37	0.59	0.52	0.03	0.69	0.83	0.28
l_acc _z	0.35	0.78	0.54	0.03	0.47	0.62	0.53	0.03	0.88	0.98	0.86

It can be seen that the p-values are mostly greater than 0.05 which confirms that the two datasets are not statistically different. Besides, the minimum, maximum, mean, and standard deviation of the two datasets presented in Table 9, also show approximately similar values.

<u>Classification using real and synthetic data</u>

In this section, we train several classic activity recognition models on real data as well as synthetic data. We propose the flow chart in Figure 19 to show how we plan to evaluate real data and synthetic data.

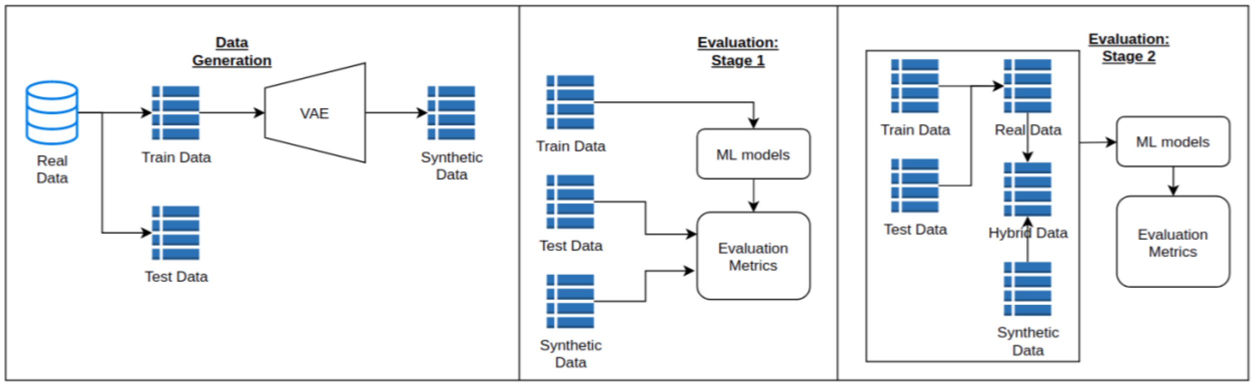


Figure 19 Flowchart for the evaluation of generated data

It shows that at first the dataset was divided into train and test subsets and only the train dataset was used to generate synthetic data. The test data was left for final evaluation. We then propose two-stage evaluation criteria to match the performance of the synthetic data and test data. The assumption is that the learning algorithms should be able to perform similarly on the synthetic and test datasets given that the models are trained with the training dataset. The evaluation metrics used are accuracy, precision, recall, and f1-score. We like to emphasize that the precision and recall are the most important characteristics that have been used in previous studies to measure the performance of synthetic data. Stage 1 compares the performance metrics applied to the test and synthetic datasets. Stage 2 fuses the real dataset with the synthetic dataset to create a hybrid dataset which is then used to compare if using synthetic dataset improves the performance metrics.

Data Generation

We used our trained VAE model to generate synthetic data. For each of the transportation modes ‘still’, ‘walk’, ‘run’, ‘bike’, ‘car’ and ‘bus’, we trained separate VAE models. Therefore, during each run, we generated data corresponding to one mode only. This way of training made it easier to label the generated data. In the final generated data sample, we had 31247 ‘still’ samples, 39179 ‘walk’ samples, 12155 ‘run’ samples, 37460 ‘bike’ samples, 35212 ‘car’ samples, and 25379 ‘bus’ samples. In the SHL dataset, the number of ‘run’ data was almost 4 times lower than the other data. After generating data, we have managed to double the total samples. Next, we divided our entire dataset into three groups. The first group contains real data from the SHL dataset only. The second group contains both real data and synthetic data and the third group contains only synthetic data.

Data Preprocessing

While it is customary to extract different features of the sensor data like mean, max, skew, kurtosis, etc., we decided to train some common activity recognition models on the raw sensor data. Our intuition is that if these models show improvement on raw values, it should definitely improve further on different hand-crafted features. The aim of the objective is primarily to show how useful synthetic data can be and does not focus on improving traditional classification accuracy.

For each of the three different data sets, we divided the data into 80-20 for training and testing, respectively. We also took a window size of 2 seconds or 200 samples. While most previous research usually takes longer window size like 5 seconds (Wang et al., 2018) or even 8, 16 seconds (Vavoulas et al., 2016), we would like to emphasize that in real-life situations and considering the move to smart mobility, a model should be able to classify pedestrians and vehicles as quickly as possible to make life-saving decisions.

Classification Results and Discussion

After taking this window size, we normalize the sensor values and feed them into some popular activity recognition algorithms like Extra Tree (EXTR), Random Forest (RF), Neural Net (NN). NN had a hidden layer size of (300,100,). The classifiers were implemented with scikit-learn (Pedregosa et al., 2011).

For stage 1, the algorithms were trained with the training dataset and then the performance metrics were evaluated on the test and synthetic dataset. The results are shown in Figure 20.

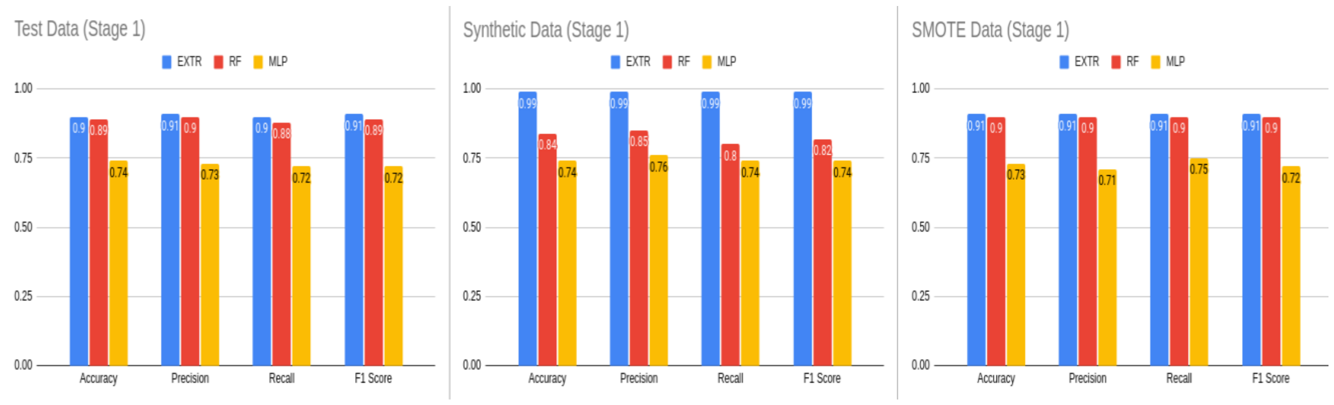


Figure 20 Performance metrics of the test and synthetic data

We see that the performance metrics of the EXTR classifier improves greatly on the synthetic dataset. The other two classifiers have similar performance metrics. The accuracy as well as precision and recall are improved to a near-perfect score. The performance of RF is similar across the two datasets, but it is somewhat lower for the NN dataset. The performance metrics were also compared to the synthetic data generation method SMOTE (Chawla et al., 2002). This can bring us to the conclusion that VAE generated data performs better on ensemble tree-based classifiers than on other models. This also shows that having a pipeline that has a data generation scheme improves the overall performance of the models.

For stage 2, the train data and synthetic data from VAE were combined to obtain the training dataset, and evaluation was performed on the test data. The results are presented in Figure 21. The accuracy, precision, recall, and f1-score of EXTR are similar to the test data in Figure 20. This leads us to the conclusion that the performance on the test data is relatively constant whether using real data or synthetic data. Therefore, the synthetic data closely mimics the real data.

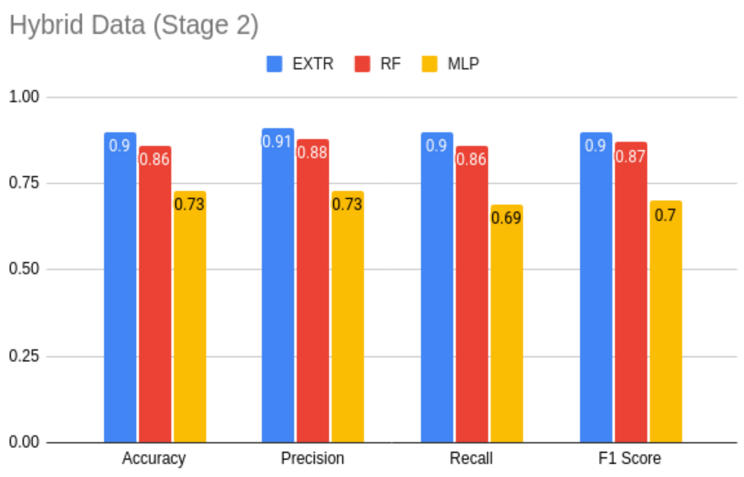


Figure 21 Performance metrics of hybrid data

All the three datasets were individually fed into the same classifier and compared. Table 10shows the obtained results. Synthetic data showed maximum efficiency since our VAE only captured the core sensor values related to any activity and discarded the noise. Real data always has some noise and therefore the classification accuracy is not always satisfactory. It depends on the individual model's capacity to filter the noise to obtain better accuracy in this case. Hybrid data showed optimal results in between the two extremes. The classification accuracy with hybrid data always shows improvement compared to the classification with real data.

Table 10 Accuracy of different classifiers

Classifier	**Real Data**	**Hybrid Data**	**Synthetic Data**
EXTR	0.90	0.94	1.00
RF	0.88	0.93	1.00
NN	0.78	0.83	0.98

Summary

In this chapter, we explored VAEs for sensor data generation purposes. Our motivation was the limitation of sensor data needed for research in this area. We showed that VAEs can be used as a data generation unit that can help overcome this limitation. We compared our generated data with the real data from SHL dataset. The mean, standard deviation, and interquartile range of the two datasets indicated that the generated data was closely related to the real data. The generated data was also used on several learning algorithms to test the performance and good results were obtained in terms of accuracy. We evaluated the datasets on certain criteria such as local and global visual evaluation and also memory independence visual evaluation. We also tested the dataset on common activity recognition models. The accuracy was improved considerably with the use of the generated data. We also noticed that hybrid data performed much better than real data. The improvement in accuracy was as high as 20% in some cases.

We can conclude that this work would be able to achieve optimal results in data-constrained studies not limited to activity recognition but in the applications of smartphone sensors in general. We also demonstrated that it can be used to upsample an underrepresented class in a dataset. This is especially helpful for cases where there could be an imbalanced dataset. The work can also be extended to other learning tasks that have these types of datasets like V2I vehicle maneuver classifications where for example the number of U-turns can be significantly lower than the left and right turns. We designed individual VAEs for each activity. For future work, we could explore a unified VAE network that can generate sensor data.

The detection of activity recognition using smartphone sensors can also be extended to wearable sensors. In general, the task of vehicle and pedestrian detection remains an important aspect of any type of P2V/V2P applications and smart mobility. Our work not only balances the data that might create bias in learning algorithms but also accurately classifies among different classes.

CHAPTER 5: CRASH DATA AUGMENTATION USING VARIATIONAL AUTOENCODER

Islam, Z., Abdel-Aty, M., Cai, Q., Yuan, J., 2020. Crash data augmentation using variational autoencoder. *Accident Analysis & Prevention* 151, 105950-105950.

Introduction

Real-time crash likelihood prediction has been an important area of study for the past two decades. Expressways, being a vital part of any roadway network, must be evaluated not only from a capacity standpoint but also from a safety stance. With modern sensors deployed along the expressways, it is convenient to get different roadway information in real-time. A major drawback with such data is that the data is highly imbalanced. The ratio of crash to non-crash data can be as imbalanced as 1:11,000 (Cai et al., 2020). Machine learning algorithms struggle to correctly predict outcomes with such data. As we will show later in this study, a model trained with such data always predicts non-crash events.

Crash prediction is an important tool to ensure safety in freeways. Figure 22 shows the number of deaths from motor vehicle accidents from 1992 to 2018 in a report (NSC-and-NHTSA-Report, 2019). It is evident that the number of fatal deaths has remained more or less around the 40,000 mark in these 26 years. In 2018 alone, 4.5 million of the crashes needed paramedic support. These has led to several research attempts in the field of safety and the recent values show that more and more work is necessary. Since crash like situations develop within short-term turbulence of traffic flow (Lee et al., 2003), it is necessary to have real-time crash risk monitoring systems. To reduce crashes there have been research from a planning perspective in which studies try to quantify how the demographic characteristics aids in a crash. From an engineering perspective, it can be the lighting of a roadway or its geometric design and from the viewpoint of control solutions, it can be countermeasures like ramp metering and variable speed limit. To identify which method to use, data is a basic requirement. While statistical methods can be tweaked to counter being biased, machine learning method can easily become biased. As more and more machine learning crash risk predictive schemes are being proposed (Li et al., 2020b; Yu and Abdel-Aty, 2013), it is important to balance these datasets. While most data augmentation techniques are applied to the computer vision dataset (Frid-Adar et al., 2018; Perez and Wang, 2017), there has been not much work in crash data until recently, where Qing et. al. used DCGAN to augment crash data (Cai et al., 2020).

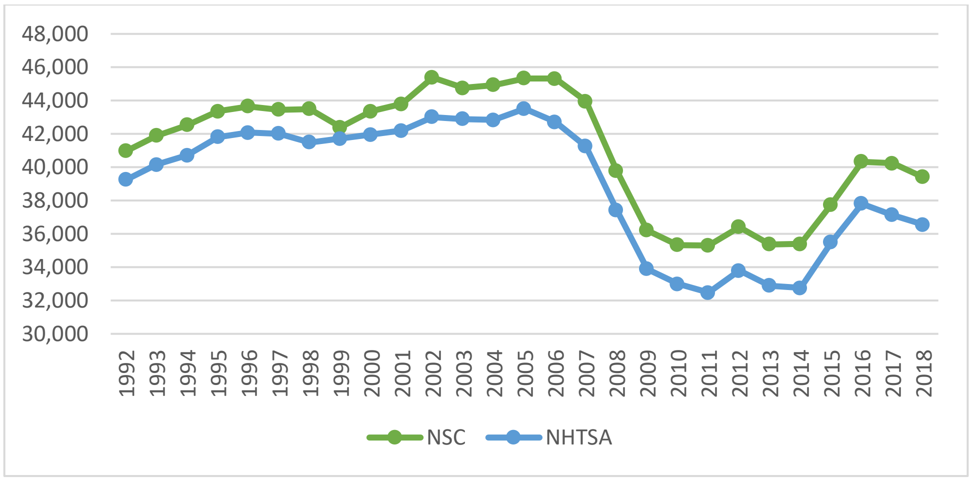


Figure 22 Motor vehicle deaths, National Safety Council (NSC) and National Highway Traffic Safety Administration (NHTSA), 1992-2018

In this objective, we propose to augment crash data using Variational Autoencoder (VAE). VAEs have been used to augment data successfully in the computer vision field. It was also helpful to balance smartphone dataset as described in Chapter 4. Our intuition was it can be used in other areas of research that have imbalanced datasets. In the following sections, we describe how we initialized the data for quick convergence and also how the normal distribution was used to encode the data. The generated data was then compared with the real data using t-test, Levene-test and Kolmogorov–Smirnov test to compare mean, variance and distribution respectively. To better understand the data, it was finally tried on three crash risk prediction algorithms: Logistic Regression (LR), Support Vector Machine (SVM) and Artificial Neural Network (ANN). The metrics used for evaluation was specificity, sensitivity and area under receiver operating characteristics curve (AUC). The generated data was also compared with SMOTE and ADASYN to assert the improvement in performance. The work presented in this chapter has been published in Accident Analysis and Prevention (Islam et al., 2020).

<u>Dataset</u>

This objective used the crash dataset obtained from processing the MVDS (Microwave Vehicle Detection System) data from expressway SR 408 in Orlando. It is 21.4 miles long with an average 417,000 vehicles using it each day (CFXWAY, 2021). It has a total of 110 MVDS detectors. The data from the year 2017 was used in this study. The roadway is shown in Figure 23.

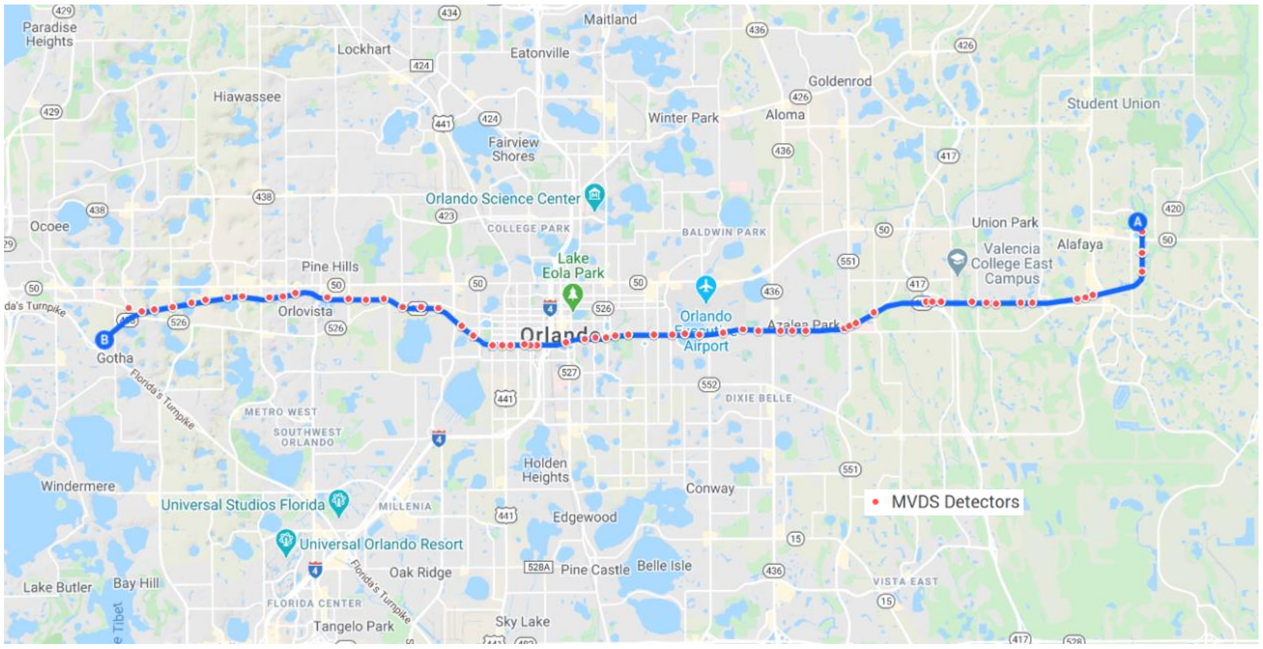


Figure 23 Study area showing SR 408 along with the MVDS detectors

The MVDS detectors in SR 408 are able to report speed, volume, lane occupancy in real time. This was further processed to obtain average and standard deviation of speed and volume. All of the 24 features that were derived from the MVDS data are summarized in Table 11**.** Each of the parameters has two values (as indicated by *1* and *2*) because for each event, two detectors upstream and two detectors downstream were considered. The data was aggregated in 5-minute intervals to prepare the features. There were 625 crash events in the year 2017 and about 6,749,447 non-crash events. For each of the crashes, 5-10 minutes before the event was labeled as a crash data. In addition, six hours prior to the crash were discarded since the data points before a crash may contain some traffic variation leading to the crash and therefore does not represent steady-state conditions. Moreover, six hours after the crash was also removed since it will take some time to reach steady state condition after a crash event.

Table 11 Feature Extracted from MVDS Dataset

Features	**Upstream**	**Downstream**	**Description**
Volume	*volume_up1, volume_up2*	*volume_down1, volume_down2*	*Directly obtained from MVDS*
Average Speed	*avg_speed_up1, avg_speed_up2*	*avg_speed_down1, avg_speed_down2*	*Directly obtained from MVDS*
Standard Deviation of Speed	*std_speed_up1, std_speed_up2*	*std_speed_down1, std_speed_down2*	*Calculated from average speed and actual speed*
Coefficient of Variation of Speed	*cv_speed_up1, cv_speed_up2*	*cv_speed_down1, cv_speed_down2*	*Calculated from average speed and actual speed*
Speed Difference	*speed_diff_up1, speed_diff_up2*	*speed_diff_down1, speed_diff_down2*	*Difference in speed between inner and outer lanes*
Congestion Index	*CI_up1, CI_up2*	*CI_down1, CI_down2*	$CI = \begin{cases} \frac{speedlimit - actualspeed}{speedlimit}, & CI > 0 \\ 0, & CI \leq 0 \end{cases}$

Proposed Method

Variational Autoencoders (VAE) are a type of autoencoder that has gained tremendous applications in computer vision recently. Proposed by Kingma and Welling (2014), it was quickly adapted to much of the vision available datasets. Since traffic data with a fixed window is numerically the same as that of an image of same dimension, the intuition was to use these models in traffic data as well. To the best of the knowledge of the authors this is the first time variational autoencoders is being investigated from crash data augmentation point of view. In this section, we will first present the basics behind a VAE and then describe the model that gave the best results for crash data generation.

An autoencoder is a neural net that maps its input to an output of exactly the same dimension. Towards the middle of the neural net there is a bottleneck that is the unique feature of an autoencoder. These can help in reducing noise or getting a lower dimensional representation of the input. A special variation of the autoencoder is the variational autoencoder. In general, all autoencoders have an encoder and a decoder. An encoder compresses the input into a latent space and a decoder decodes from this latent space. For an autoencoder, the latent space is discrete and therefore sampling from this space would not result in anything meaningful. Variational autoencoder takes these discrete values and tries to find a known distribution of the latent space. Therefore, a variational autoencoder not only tries to reconstruct its input, but also tries to form a distribution in the latent space. If we denote x as the input, z as the latent space representation and $\hat{x}$ as the output of an autoencoder, then $q(z|x)$ can be the encoder and $p(x|z)$ can be the decoder. The loss function consists of two terms; the first term penalizes the reconstruction error between the input x and the output $\hat{x}$ and the second term penalizes the error between the prior $p(z)$ and the learned distribution $q(z|x)$.

$L = -D_{KL}\big(q(z|x)||\,p(z)\big) + \log\, p(z|x)$

For crash data augmentation, we used the VAE model shown in Figure 24. The leftmost layer labeled x is the input and the rightmost layer labeled $\hat{x}$ is the output. The bottleneck is the layer z which we have chosen to have a dimension 3 since better latent space was obtained with 3 rather than 2. All the weights and biases of the encoder and decoder were initialized with Xavier Glorot initialization (Glorot and Bengio, 2010). This initialization is able to bring faster convergence for deep learning networks. The latent space was encoded to a normal distribution with a mean of 0 and standard deviation of 0.1.

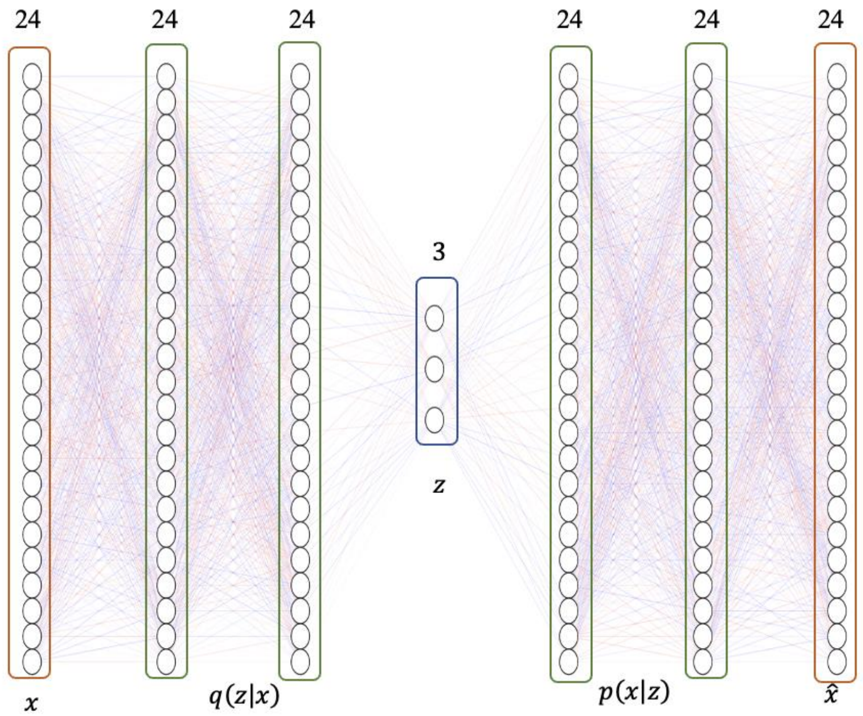


Figure 24 VAE model (the numbers above each layer represents the number of neurons)

Encoder Model of VAE

We considered a two layer densely connected encoder model for our VAE. The next two layers after the input x are the decoder. The number of neurons chosen was (24, 24). We also experimented with lower numbers such as (10,10), (15,15), (20,20) and (24,20). Best latent space visualization was obtained for (24,24).

Decoder Model of VAE

The decoder takes the latent layer code and tries to reconstruct the original crash data. For the decoder we also had 24 neurons for the first layer and the same for the second layer. At the end of the decoder a sigmoid activation function was used. The output layer would be the same as the input which is 24. All the neurons were densely connected to one another.

Data Generation Pseudocode

The pseudocode for training and data generation of VAE is shown Table 12**.** The input for training is the real data (70% was used for training). The optimization techniques used was Adam with a learning rate of 0.0001. The batch size was selected as 874. Of these 874 samples, 437 was crash and 437 was non-crash. Therefore, we have used the same crash values over the entire epoch. This was necessary otherwise the VAE model would overfit with only the non-crash values. In each iteration, we also calculated loss on the train samples for visual evaluation. It was trained for a total of 12,000 epochs since this is required to consider all the non-crash data. For data generation, we sampled from the latent space generated from the training phase and passed it through the trained decoder to obtain synthetic data at least once.

It is also important to determine the latent space boundary from which we have to sample to get crash data. Generally, VAEs generate the latent space in which there is overlap between classes of data. If we want to sample any particular class, it is necessary to estimate the cluster boundary of the class. Visual interpretation may often lead to false selection of the boundary. For our case, since we are interested to sample crash data, it is important to find the cluster volume of the crash data. We have used confidence ellipsoid to draw this latent space boundary. A confidence ellipsoid draws a cluster boundary based on the confidence level. An 80% confidence ellipsoid means that 80% of the observed data falls within the boundary of the ellipsoid. We have used three confidence levels: 80%, 85% and 90%. Next, we could easily sample from the space within the ellipsoid to get generated data.

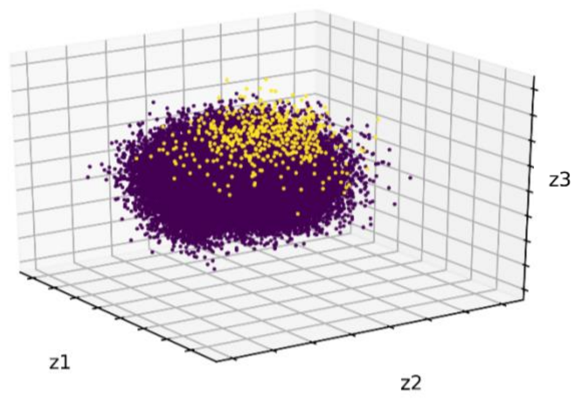
Table 12 Algorithm for generating synthetic data

1:	**Input:** Real Data
2:	**Initialize:** decoder, encoder, prior, latent space dimensions
3:	Training:
4:	**for** the number of iterations **do**
5:	**for** the number of batches **do**
6:	input the batch into the encoder
7:	calculate loss using equation 1 and optimize over the training dataset
9:	Data Generation:
10:	Determine crash boundary in latent space with confidence ellipsoids
11:	**for** number of samples **do**
12:	sample from the latent space
13:	use the trained decoder from training step to obtain synthetic data

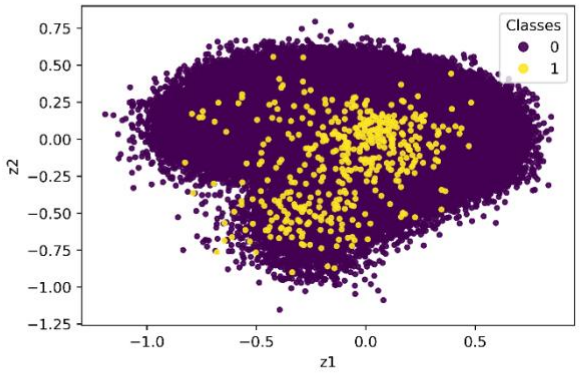
<u>Experiments</u>

Latent Space Visualization

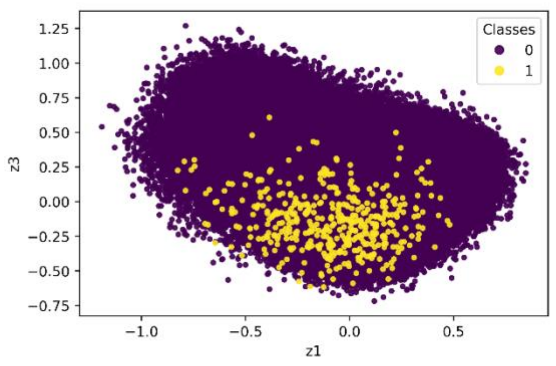
The latent space for a VAE gives the impression as to how effective a model is in separating a dataset into its target. By properly interpreting the latent space boundary it is possible to create new samples of data. It is also an important indicator as to how effective is the KL divergence loss. In the Figure 25, we display the results obtained from our VAE model.



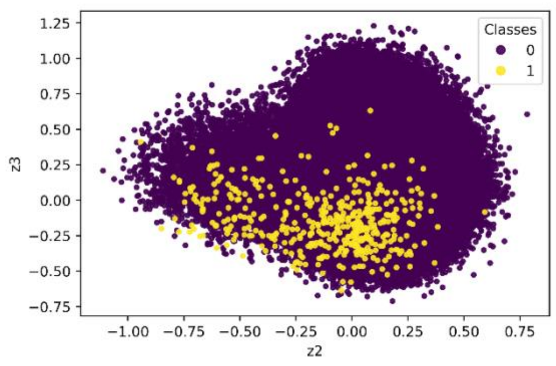
(a)



(b)



(c)

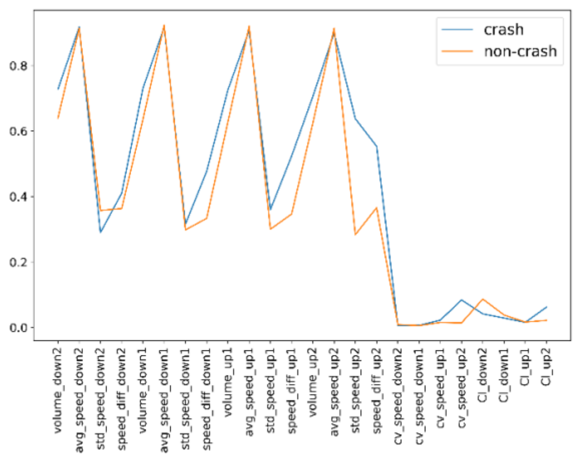


(d)

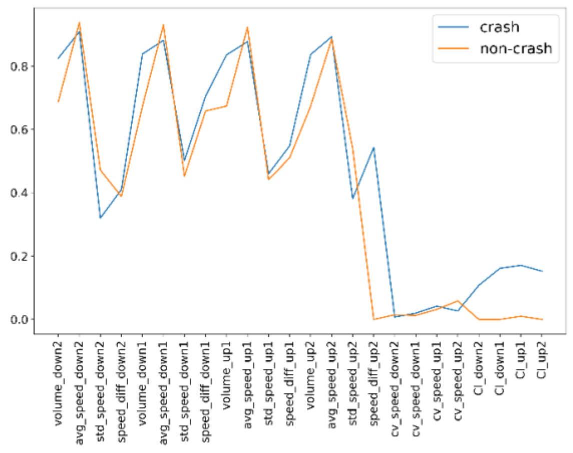
Figure 25 Latent Space Visualization of the VAE Model

Since there are three neurons in the bottleneck of the model, the latent space is best visualized in 3D as is shown in Figure 25 (a). $z1, z2, z3$ are the three neurons which are being displayed. This figure gives the idea that the system is able to create different clusters for the crash and non-crash data. The crash data, which is encoded in yellow, is at the surface of the 3D plot as shown in Figure 25 (a). We also show the 2D plots in Figure 25 (b), 3 (c) and 3 (d). All of these figures show that even though the surface of the latent space mostly contains the crash data, the overlap between crash and non-crash events is also significant. This can be explained from the fact that certain crash events may not result in a crash due to number of other factors like driver reaction, automatic breaking or other safety features that are becoming more and more popular in cars.

This is also clearly depicted in Figure 26 where we plot one crash and one non-crash sample in each sub-figure (a) and (b). It shows how close the values of the two events might be. Nevertheless, the overlap is an indicator that the conditions are getting risky and that a crash is likely.



(a)



(b)

Figure 26 Similarities between crash data and non-crash data

Figure 27 shows the 80% confidence ellipsoid that we have used to determine the latent space boundary of the crash data. This was a very important step because visual evaluation of the latent space may often lead to capturing false boundary and eventually false data. The yellow dots are the crash data and the purple ellipsoid is the 80% confidence ellipsoid. This region was selected to sample generated crash data. The same was done for two other confidence levels: 85% and 90%.

<u>**Synthetic Data Evaluation**</u>

After training the VAE, we can sample from the latent space shown in Figure 27 to generate more crash data. For our case, we have generated over 20 million crash events. In most data augmentation, the next step is to have a visual observation of the generated data (Wang et al., 2018). We evaluate the data from a global viewpoint of mean and standard deviation. We also plot the data distribution and analyze if both real and generated data show similar pattern.

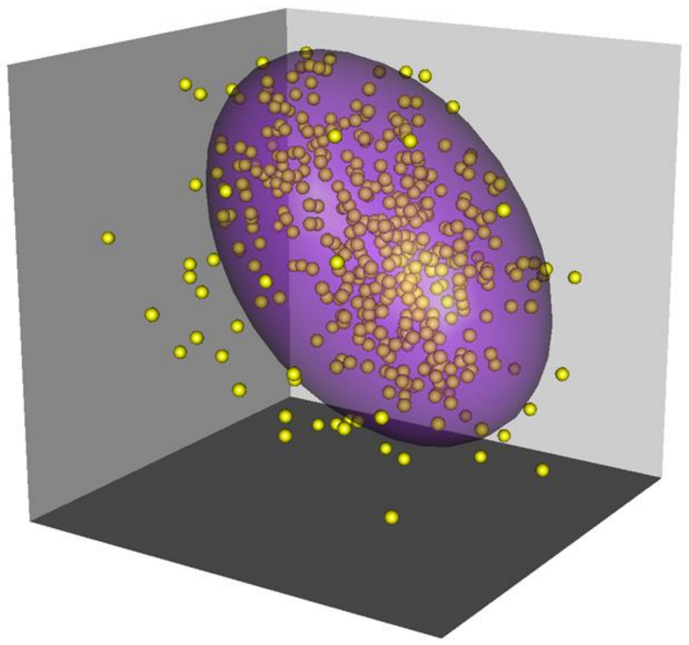


Figure 27 Latent space showing the crash data samples and 80% confidence ellipsoid

Global Mean and Standard Deviation Evaluation

We illustrate the mean and standard deviation of the generated and real data in the Figure 28. We have compared 2 million synthetic crash samples with the 437 crash samples in the training data. Each VAE has two losses. We showed in the previous section that the KL divergence loss limits the data within a fixed area. The other loss which is the reconstruction loss shows how close the generated data is to the real data. The mean and standard deviation gives an idea about this. As can be seen from Figure 28, the quartile range of values for the generated data mimics the real data. The mean of some of the values have shifted considerably. This can be attributed to the huge difference of data samples between the two sets. An outlier in the real data can easily push the mean aside which cannot happen for the 2 million generated data.

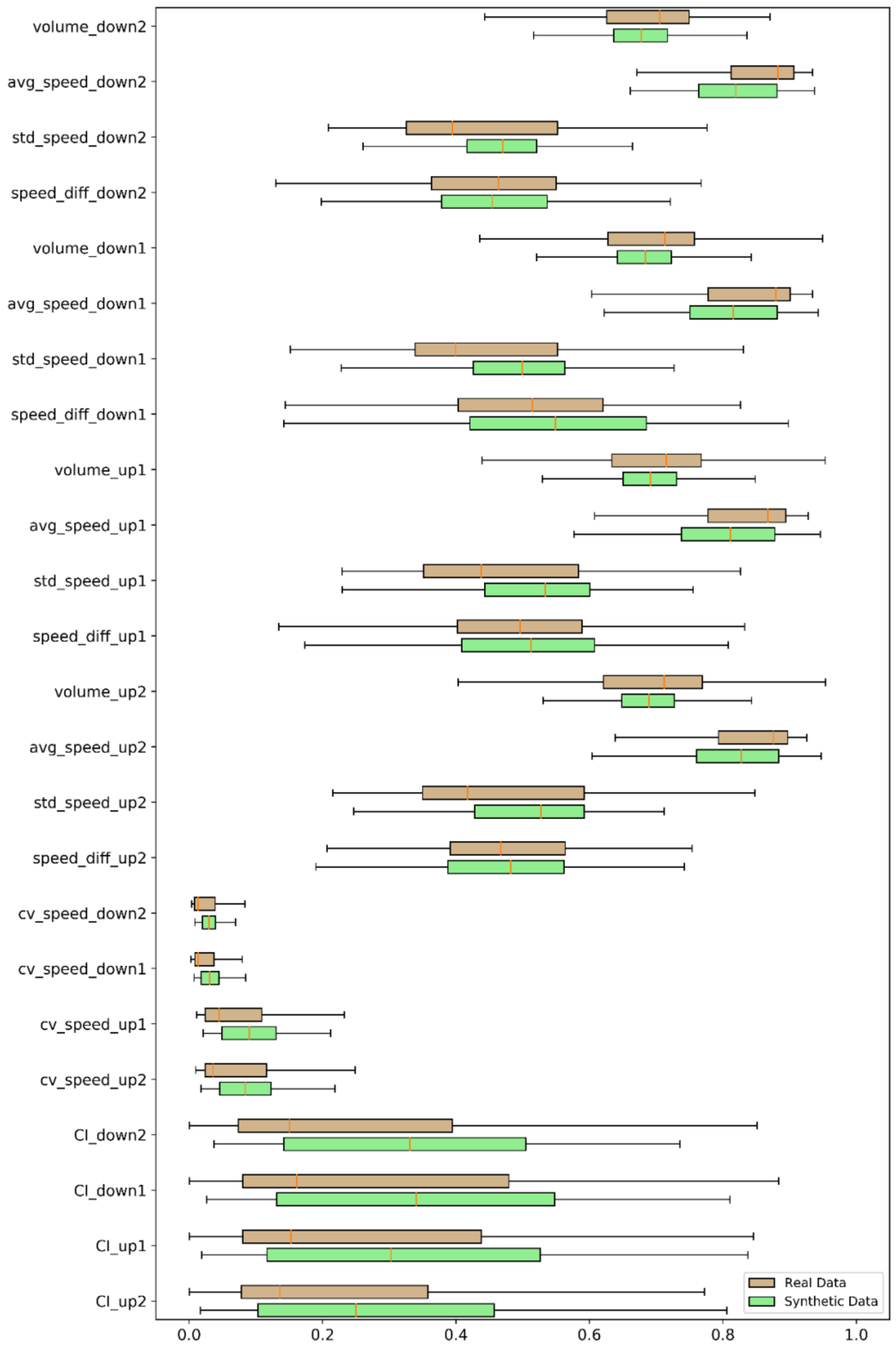


Figure 28 Boxplot showing the mean and standard deviation of real data and synthetic data

Data Distribution Evaluation

The distribution of each of the 24 variables used in this study is shown in Figure 29. We have used a violinplot to compare between the distributions of the generated data and the real data. This helps us to compare not only the distribution but also the gives an idea about the mean of the two datasets. Synthetic data is shown in green and real data in orange. For all of the variables, the generated data pattern closely resembles the distribution of real data. One important insight from this figure is that the spread of the generated dataset is always less than the spread of the real dataset. This is due to the reason that the real dataset only has 437 crash events whereas the synthetic dataset has 2,000,000 crash events. Also, while sampling, we intentionally left out some outliers that were too far from the crash cluster. Thus, the distribution standard deviation is lower for the larger dataset since it has less percentage of outliers.

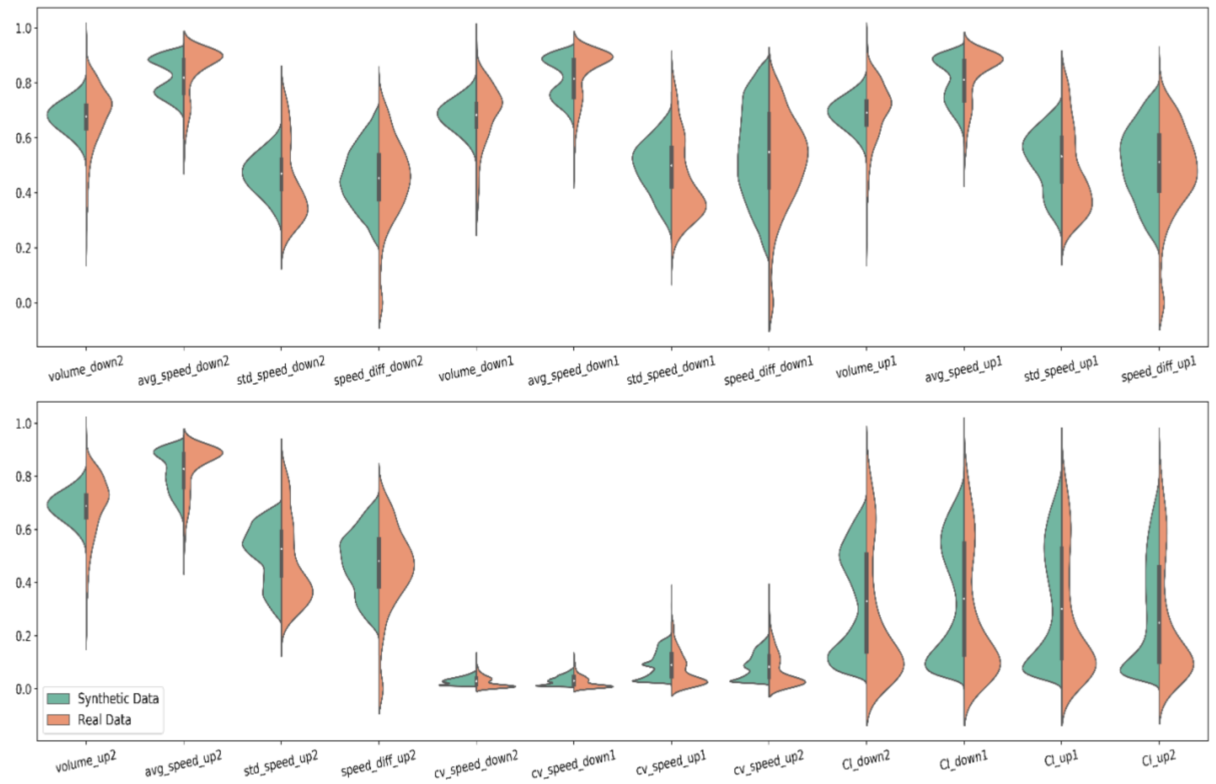


Figure 29 Data distribution evaluation of generated data and noisy data

Statistical Evaluations

The minimum, maximum, mean and standard deviation of all the variables in the real and synthetic data is summarized in Table 13. In addition, three different tests were carried out to confirm that the two datasets do not differ from each other statistically. 30 random samples were taken from the two datasets. The p-values of each of the tests are presented in the table. For t-test, it can be noted that, the p value is greater than 0.05 for all of the variables. This confirms that the mean of the two datasets are not significantly different. Levene's test was carried out to compare the variances between the two datasets. The p-value is greater than 0.05 for all variables except five. This indicates that the variance of the datasets is not statistically. Similarly, the p values from Kolmogrove Smirnov test are also greater than 0.05 indicating that the distribution of the variables in the two datasets are not statistically significant. Therefore, we can conclude that the two datasets are statistically similar.

Classification using real and synthetic data

In the previous section, we examined the data from a statistical point of view. It is also important to see how the data performs under learning algorithms that have used in the past to classify crash and non-crash data. We used Logistic Regression (LR) to classify crash and non-crash data since this was a common methodology used in previous work (Cai et al., 2020; Li et al., 2020b; Lin et al., 2015). We also used Support Vector Machine (SVM) that has also been previously studied (Basso et al., 2018; Yu and Abdel-Aty, 2013).

Table 13 Descriptive statistical features of real and synthetic data as well as comparison between the two based on t-test, Levene-test and Kolmogrove Smirnov test (ks-test)

	Real Data				**Synthetic Data**				**Statistical Tests between the real dataset and synthetic dataset**		
Variable	**Min**	**Max**	**Mean**	**STD**	**Min**	**Max**	**Mean**	**STD**	**t-test (p-value)**	**Levene-test (p-value)**	**ks-test (p-value)**
volume_down2	0.20	0.95	0.68	0.10	0.47	0.83	0.67	0.05	0.92	0.93	0.39
avg_speed_down2	0.52	0.93	0.84	0.08	0.66	0.93	0.82	0.06	0.26	0.18	0.80
std_speed_down2	0.20	0.77	0.43	0.14	0.24	0.66	0.46	0.07	0.43	0.15	0.39
speed_diff_down2	0	0.76	0.44	0.15	0.19	0.72	0.45	0.10	0.49	0.74	0.59
volumne_down1	0.31	0.94	0.68	0.11	0.48	0.84	0.68	0.05	0.43	0.45	0.59
avg_speed_down1	0.46	0.93	0.83	0.08	0.62	0.94	0.81	0.07	0.32	0.26	0.39
std_speed_down1	0.15	0.83	0.44	0.14	0.22	0.72	0.49	0.09	0.42	0.52	0.13
speed_diff_down1	0	0.82	0.50	0.17	0.14	0.89	0.54	0.16	0.63	0.29	0.23
volumne_up1	0.20	0.95	0.69	0.10	0.48	0.84	0.68	0.05	0.58	0.56	0.03
avg_speed_up1	0.47	0.92	0.82	0.09	0.57	0.94	0.80	0.08	0.18	0.26	0.59
std_speed_up1	0.22	0.82	0.47	0.15	0.22	0.75	0.51	0.10	0.83	0.60	0.59
speed_diff_up1	0	0.83	0.48	0.16	0.17	0.80	0.50	0.13	0.94	0.74	0.39
volumne_up2	0.21	0.95	0.68	0.11	0.49	0.84	0.68	0.05	0.90	0.62	0.39
avg_speed_up2	0.48	0.92	0.83	0.08	0.60	0.94	0.81	0.07	0.15	0.33	0.39
std_speed_up2	0.21	0.84	0.47	0.15	0.24	0.71	0.50	0.10	0.66	0.86	0.80
speed_diff_up2	0	0.75	0.45	0.15	0.18	0.74	0.47	0.11	0.43	0.18	0.39
cv_speed_down2	0.003	0.12	0.02	0.02	0.008	0.07	0.03	0.01	0.57	0.03	0.23
cv_speed_down1	0.002	0.12	0.02	0.02	0.007	0.08	0.03	0.01	0.70	0.64	0.07
cv_speed_up1	0.011	0.35	0.07	0.06	0.02	0.21	0.09	0.04	0.70	0.23	0.13
cv_speed_up2	0.009	0.35	0.07	0.06	0.01	0.21	0.08	0.04	0.65	0.37	0.59
CI_down2	0	0.85	0.25	0.23	0.03	0.73	0.32	0.19	0.26	0.19	0.59
CI_down1	0	0.88	0.26	0.23	0.02	0.81	0.34	0.21	0.39	0.32	0.59
CI_up1	0	0.84	0.26	0.23	0.01	0.83	0.32	0.22	0.26	0.25	0.39
CI_up2	0	0.84	0.23	0.22	0.01	0.80	0.28	0.20	0.13	0.32	0.13

Data Preparation

The real dataset was split into train and test datasets. 70% data was used for training and the rest for evaluation. It can also be mentioned here that a set of 100 different random splits of the dataset was generated and then evaluated with the chosen dataset. Statistical t-test, Levene test and Kolmogorov–Smirnov test was carried out. It was noted that the p-values of all the tests were greater than 0.05 which concludes that the chosen dataset and the other 100 datasets were similar with respect to mean, variance and distribution.

Only the 70% data was used to train our VAE model and it was tested on the rest of the data that the model did not see. This helps us to identify if the model can actually perform on real data that it has never been used before. We have also used this data to generate data with other popular data augmentation techniques like ADASYN (He et al., 2008) and SMOTE (Chawla et al., 2002). SMOTE generates minority samples along the line joining the *k* (mostly takes as 5) minority samples. On the other hand, ADASYN is a sampling approach that is focused on generating samples of data that are harder to learn. It assigns different weights to different minority samples. Higher weights mean higher difficulty in learning and therefore more data points are generated in that vicinity based on kNN. We have also used an undersampled dataset to see how sampling a little fraction of the non-crash data and all of the crash data perform to train a model. Random undersampling method was used to create the undersampled dataset. Finally, the data generated from the VAE model is also evaluated. The summary of the train dataset and test dataset are provided in Table 14. We have also included the sample size of each dataset in the table. We generated 3 sets of VAE data for three different confidence ellipsoids. The data generated from the latent space of 80% confidence ellipsoid is referred to as VAE Data A. The data from 85% and 90% confidence intervals are labeled as VAE Data B and VAE Data

C respectively. The number of crash samples increased from VAE Data A to C is because of the increase in volume of the confidence ellipsoids.

Table 14 Datasets used for evaluation

Train Dataset			Test Dataset		
Description	# of crash samples	# of non-crash samples	Description	# of crash samples	# of non-crash samples
Real Data (70% of the MVDS Data)	437	4,724,612	Real Data (30% of the MVDS Data)	188	2,024,835
Undersampled Data	437	437			
ADASYN Data	4,724,612	4,724,612			
SMOTE Data	4,724,612	4,724,612			
VAE Data A	5,583,020	4,724,612			
VAE Data B	6,738,161	4,724,612			
VAE Data C	8,336,539	4,724,612			

Evaluation Procedure

To evaluate whether the generated data performs as expected, we propose the methodology in Figure 30. The real data from the MVDS detectors are firstly divided into train and test sets. Only the trained data is used thereafter, and test data is left for final evaluation. The train data is processed into five different datasets. “Real Data” is exactly the train data and is used as a reference. “Undersampled Data” contains all the crash data in the train data and a sample of the non-crash data. It is perfectly balanced meaning that the number of crash and non-crash samples are exactly equal. “ADASYN Data” also contains a balanced dataset based on the work by He et al. (2008). The minority samples of crash data are increased in this method. SMOTE is also a minority oversampling technique as proposed by (Chawla et al., 2002). “SMOTE Data” contains the data from this algorithm. Finally, “VAE Data A” (also B and C) contains the samples generated by the VAE model described above and also the 437 real crash data. All these five datasets are fed into two learning algorithms namely Logistic Regression and Support Vector Machine. Finally, the model is evaluated based on three values: specificity, sensitivity and AUC (area under the ROC curve). These metrics have been used in different data augmentation techniques.

Specificity is defined as the ability of a model to predict true negative values. For our case, this is the ability to predict a non-crash as a non-crash.

$$Specificity = \frac{true\ negative}{true\ negative + false\ positive}$$

Sensitivity is the ability to predict true positive values. For the case of crash data, this shows how good a model is to predict a crash as a crash.

$$Sensitivity = \frac{true\ positive}{true\ negative + false\ positive}$$

AUC or area under the receiver operating characteristics curve is used to tell how effective a model is in distinguishing between classes. In this scenario, it refers to the ability to segregate crash and non-crash events.

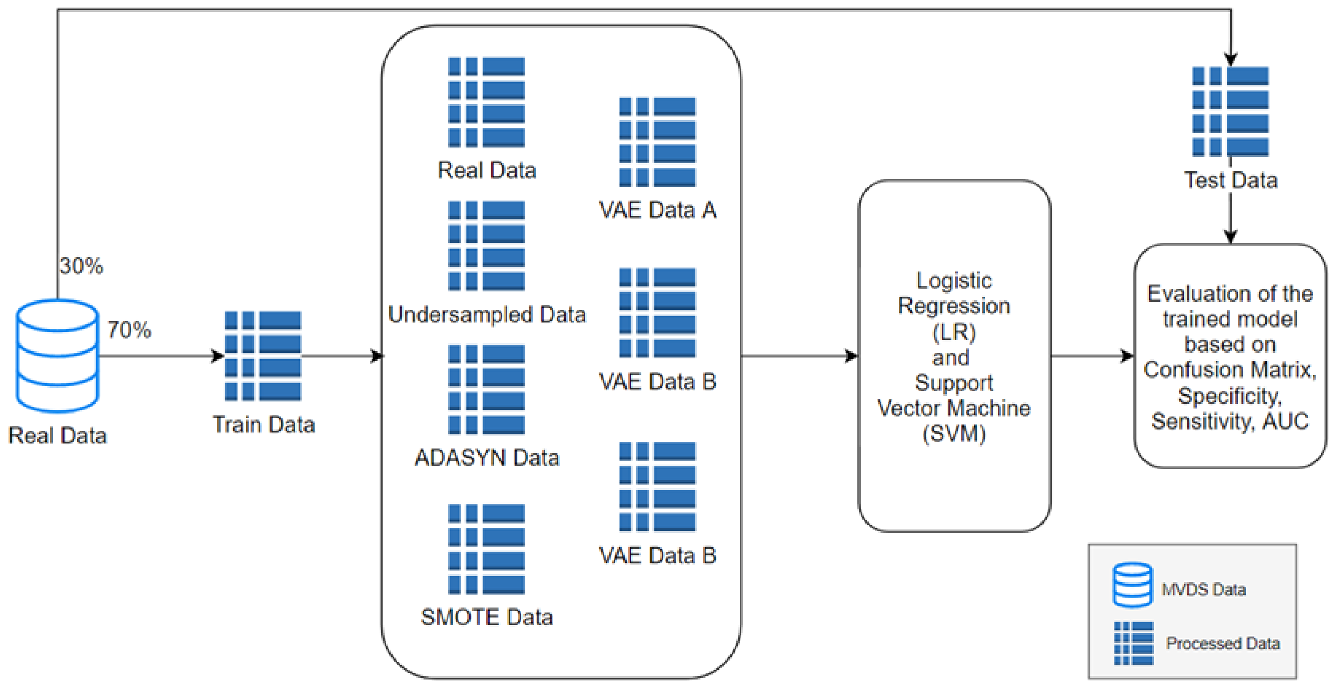

30%
70%
Real Data
Train Data
Real Data
Undersampled Data
ADASYN Data
SMOTE Data
VAE Data A
VAE Data B
VAE Data B
Logistic Regression (LR) and Support Vector Machine (SVM)
Test Data
Evaluation of the trained model based on Confusion Matrix, Specificity, Sensitivity, AUC
MVDS Data
Processed Data

Figure 30 Flowchart for the evaluation of generated data

Classification Results and Discussion

It is also important to look at the confusion matrix to better understand the true positives and the false positives. Confusion matrix lists all the predicted values and true values of the test dataset. Table 15 presents different confusion matrix calculated. The actual number of the classes are shown in each cell along with its percentage in italics.

The confusion matrix for the models LR and SVM are shown along with the different confidence ellipsoids. True labels presented along the rows are the labels obtained from the test data while predicted labels along the columns show the model predictions. Let us refer to the values "True = YES" and "Predicted = YES" as True Positives (TP) and "True = NO", "Predicted = NO" as True Negatives (TN). "True = NO", "Predicted = YES" is referred to as False Positives (FP) and the inverse as False Negatives (FN). As we increased the confidence level, it was seen that the

TP values steadily increased. Out of the 188 crashes in the test dataset, the LR model correctly classified 139, 150 and 154 crashes along the different confidence levels while the SVM model correctly classified 163, 165 and 169, respectively.

Table 15 Confusion Matrix of the models with different confidence

	80% Confidence Ellipsoid		85% Confidence Ellipsoid		90% Confidence Ellipsoid		
	Predicted = NO	Predicted= YES	Predicted= NO	Predicted= YES	Predicted= NO	Predicted= YES	
True = NO	1877900 0.93	146935 0.07	1816510 0.90	208325 0.10	1776680 0.88	248155 0.12	LR
True = YES	49 0.26	139 0.74	38 0.2	150 0.8	34 0.18	154 0.82	
True = NO	1792742 0.89	232093 0.11	1762788 0.87	262047 0.13	1693220 0.84	331615 0.16	SVM
True = YES	25 0.13	163 0.87	23 0.12	165 0.88	19 0.10	169 0.90	

Also, TN values decreased as the confidence level was increased. This means that even though the models have predicted crashes more accurately, it sometimes classified some non-crash event as a crash. This is the expected trend since increasing the confidence ellipsoid, increases the size of the ellipse and therefore it captures not only more of the crash data but also non-crash data that may be located between the newly captured crash events. We would also like to present our argument in favor of the fact that, these newly captured non-crash data that are predicted as crash lies in the grey boundary between the crash and non-crash data. These may or may not result in a crash owing to a number of other factors like driver skill or safer vehicles, but it is safe to consider that these events have a good chance of resulting in a crash. Therefore, the increase of FP values can also be an indicator that the models are performing towards our expectations. Based on this reasoning, we can argue that the 90% confidence ellipsoid should be selected as the method going forward. The best result for LR is marked green and that of SVM blue in Table 15.

Table 16 shows the comparison of the confusion matrix of VAE with SMOTE and ADASYN. From this table it can be seen that the maximum number of crashes correctly classified is 176 for the ANN model. While the performance of the LR model is better for SMOTE and ADASYN, for SVM and ANN, the performance of VAE data is better than the two minority oversampling methods.

We compared VAE model to the results obtained from the work of (Cai et al., 2020). The authors reported similar performance metrics as we have used in our study. The data used by them was also for SR 408 for the year 2017. The comparison on the test data is shown in Figure 31. In this case as well, we observe that the specificity, sensitivity and AUC improvement is remarkable for both the VAE models. The improvement in specificity is 8% for the LR model and 4% for the SVM model. The improvement of sensitivity is also increased by 6% and 5%, while that of AUC is 7% and 1%, respectively. Therefore, we can conclude that the VAE performs better for the LR and SVM models than the DCGAN method.

Table 16 Confusion Matrix of the models with VAE, SMOTE and ADASYN

	SMOTE		ADASYN		90% Confidence Ellipsoid (VAE Data C)		
	Predicted = NO	Predicted= YES	Predicted= NO	Predicted= YES	Predicted= NO	Predicted= YES	
True = NO	1744736 *0.86*	280099 *0.14*	1748582 *0.86*	276253 *0.14*	1776680 *0.88*	248155 *0.12*	LR
True = YES	24 *0.13*	164 *0.87*	24 *0.13*	164 *0.87*	34 0.18	154 *0.82*	
True = NO	1773173 *0.88*	251662 *0.12*	1762788 *0.86*	262047 *0.14*	1693220 *0.84*	331615 *0.16*	SVM
True = YES	24 *0.13*	164 *0.87*	22 *0.12*	166 *0.88*	19 *0.10*	169 *0.90*	
True = NO	1739254 *0.86*	285581 *0.14*	1709736 *0.84*	315099 *0.16*	1601701 *0.79*	423134 *0.21*	ANN
True = YES	20 *0.10*	168 *0.9*	18 *0.10*	170 *0.9*	12 *0.07*	176 *0.93*	

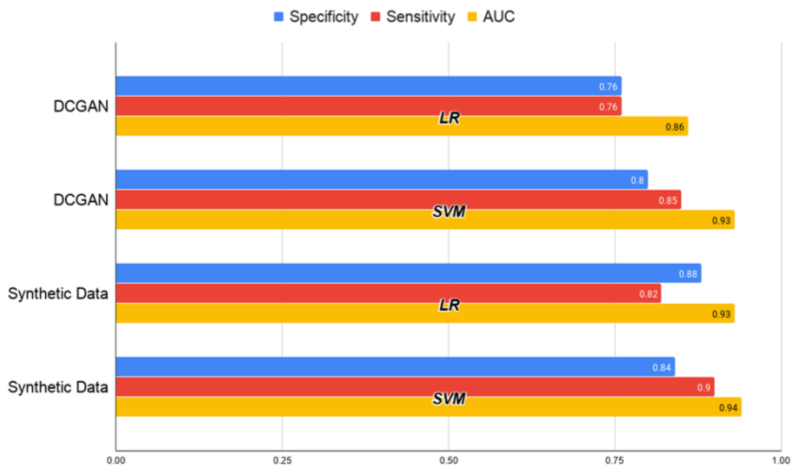


Figure 31 Comparison between DCGAN and VAE as a data augmentation method

We also compared the results with other data sampling methods. The results are shown in Figure 32. The real dataset is also shown for reference. For this dataset, we see that the model predicts everything as non-crash, since the specificity is 1 and sensitivity is zero. The undersampled dataset also performs well but the specificity is lower than the other techniques since it was only trained with a handful of non-crash data and therefore cannot recognize properly some of the non-crash events in the test data. ADASYN and SMOTE are also well recognized minority sampling techniques and therefore perform better than random undersampling. The performance of the VAE data is shown toward the right end of Figure 31.

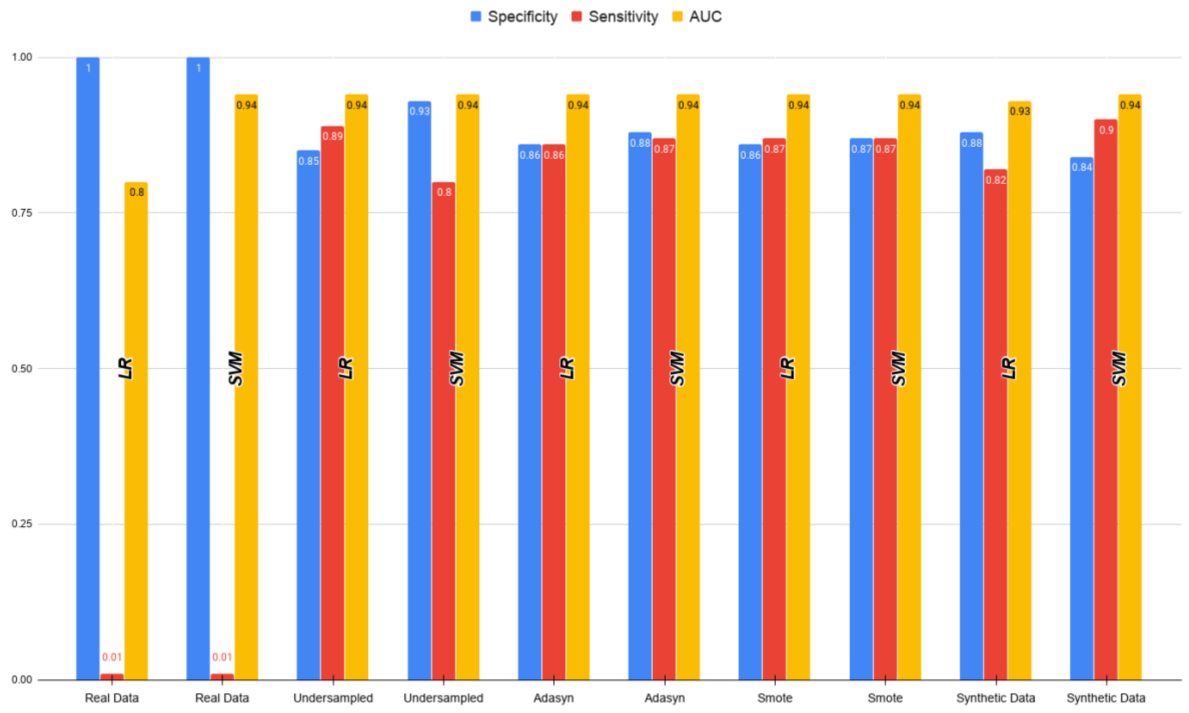


Figure 32 Comparison of the evaluation metrics across various models and proposed model

The VAE model has improved specificity for the LR model (about 2% increase) than that of both ADASYN and SMOTE while the sensitivity is lower by 4%. For the SVM model, VAE generated data has better sensitivity. The specificity is comparable to ADASYN and SMOTE while the sensitivity is improved by 3%. The AUC (for SVM) of the VAE generated data is also better than both ADASYN and SMOTE by 1%. The reason for this variation can be attributed to the selection of the confidence ellipsoids. A higher confidence level would mean better crash prediction, but also higher false alarms and lower confidence interval would lower the crash prediction and also lower false alarms. We are, therefore, convinced that the confidence level of the ellipsoid is an important hyperparameter to be tuned. From a crash prediction point of view, it is safe to have a higher confidence level since the false alarms would be a type of indicator that traffic conditions are likely to result in a crash. We also propose that, sensitivity is a better measure of the performance of generated crash data owing to this intuition. The SVM model outperforms the minority oversampling techniques with respect to sensitivity. For the ANN model, the sensitivity value peaks for all the models that have been trained. This model successfully classifies 93% of the crashes.

Summary

To predict crashes in real-time is of utmost importance in safety. It is also important to mark certain areas as unsafe where there are frequent crashes so that authorities can take long term measures to better the location. Due to insufficient data, the analysis of such metrics does not always provide conclusive results. Variational autoencoder can help leverage this by producing synthetic data. Given a small bunch of sample data, it can, theoretically, produce infinite data samples.

To summarize, the work in this chapter we have generated over 20 million crash data with only a handful of 437 crash samples. We also used the concept of confidence ellipsoid to accurately capture crash data from the latent space of a VAE. We then compared the generated data with the real data from statistical standpoints. The mean, standard deviation of the two datasets were similar. While it is known that autoencoders tend to reduce the noise in the data and thus the generated data could lose some important feature in the process, the data distribution of each of the 24 variables (in Figure 29) shows that the synthetic dataset closely follows the distribution of the real data. Finally, we evaluated the performance of the dataset with three crash prediction models: LR, SVM and ANN. The metrics chosen for evaluation was specificity, sensitivity and AUC. The real crash data was also augmented with state-of-the-art minority sampling techniques like SMOTE and ADASYN. These datasets were also used on LR, SVM and ANN to obtain the evaluation parameters. The results of the VAE model showed improvements from the view of specificity, sensitivity and AUC. The results were also compared with data augmentation technique involving DCGAN. Our VAE model outperformed most studies in the literature. The results from the confusion matrix, that was used to compare across different confidence ellipsoids, was also very insightful. It indicated that increasing the confidence ellipsoid increases the chances of getting false alarms but that is also an indicator that conditions are favorable for a crash. Overall, the results were encouraging, and can aid to balance an imbalanced crash dataset.

Future studies can train crash prediction models on more complex and non-linear algorithms that do not perform well on small datasets. Using VAE as a data generation tool in the pipeline would definitely aid in generating substantial data to train on non-linear models. Furthermore, there could be more work relating VAE that could have one more class in between crash and non-crash: crash-prone. The training data of this region could be derived from the false positives from our VAE model.

CHAPTER 6: SIGNAL PHASING AND TIMING PREDICTION AIDED BY DETECTOR DATA

Traffic signal control is one of the earliest areas of research in transportation and remains to be a debated topic. Traditionally, the problem of signal control can be decomposed into a two-stage solution. The first stage is to develop traffic flow models that help to estimate macroscopic traffic parameters at intersections. This can be flow-based analytical models or simulation-based iterative models. But for complex traffic conditions, the modeling costs and errors also have to be taken into account. Artificial intelligence can also be used in simulation to train macroscopic traffic flow. In this case too, the tuning of hyperparameters can prove to be a costly and time-consuming process (Li et al., 2016).

The next stage is to develop appropriate signal plans based on the estimated traffic flow parameters. Mathematical programming models can then be used to optimize delay or queue length (Liu et al., 2015; Yang and Jayakrishnan, 2015) Artificial intelligence can also be used to generate optimal signal plan which can learn the dynamics of the system and correlate it with the changes in traffic flow (Bingham, 2001; Li et al., 2016; Prashanth and Bhatnagar, 2010).

In this work, we focus on using deep learning architecture to model signal timing parameters directly from traffic flow. Each approach level volume and occupancy were also aggregated at the cycle level from high-resolution detector data which were then fed into the proposed model to obtain signal parameters appropriate for that traffic flow. These types of methodologies would aid in reducing the dependency on the optimization algorithms while simultaneously simplifying the traditional two-stage traffic signal control problem into a one-stage solution.

Data Preparation

Two distinct corridors in Orlando, Florida were selected: one of the corridors operates adaptive signal control (Corridor A) and the other corridor operates actuated signal control (Corridor B). As shown in Figure 33, Corridor A has 7 intersections and B has 10 intersections. All the 17 intersections are equipped with Automated Traffic Signal Performance Measures (ATSPM), which store high-resolution detector data of each intersection along with the signal timing parameters like cycle length, green time, red time, etc. Data from August to December 2019 were used in the study.

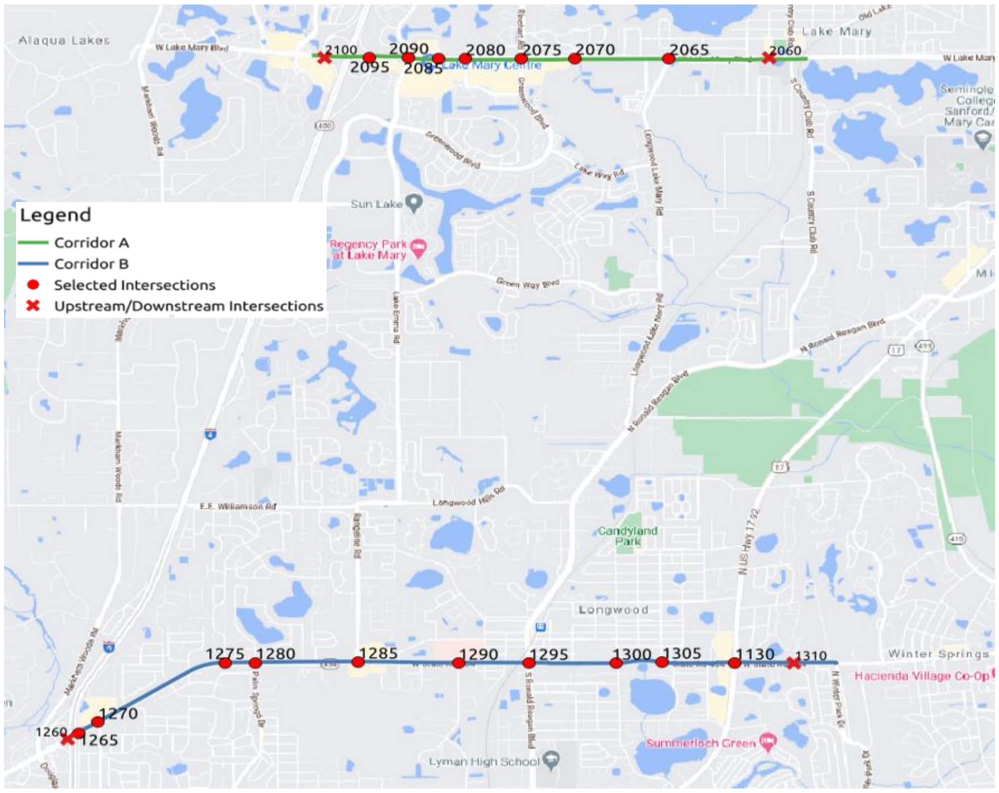


Figure 33 Study Area

For Corridor A, there are a total of seven intersections. Each is identified by the intersection ID as shown in Figure 33. The major movement is in the east-west direction. Intersections 2065, 2090, and 2075 are four-leg intersections that have protected left-turn movements for all approaches. Intersections 2085 and 2080 are also similar. However, they have an overlap phase as well, which circulates the major left turn movements twice per cycle. Intersection 2070 does not have protected left turns for the minor movement and 2095 is a T-intersection one leg of which merges into the ramp of a freeway, I-4. Since upstream and downstream count and occupancy were also used in the study, two more intersections 2100 and 2060 were added to the two ends of the corridor. On the other hand, Corridor B has 10 intersections, out of which two have all protected movements for all phases (1295 and 1130). Intersections 1270 and 1280 do not have protected left turns for the minor road but instead uses dummy phases to balance the ring. Intersection 1270 also has an overlap phase that circles the major through movement twice per cycle. Four intersections do not have protected left turns (1285, 1290, 1300, 1305) for the minor road. Intersections 1265 and 1270 are both T-intersections. Intersection 1265 merges into a ramp on the I-4 freeway. Similarly, as Corridor A, two more intersections 1260 and 1310 were added to the two ends of the corridor to calculate the upstream and downstream traffic parameters of 1265 and 1130, respectively.

Table 17 and Table 18 show the phases that are related to each intersection ID, as well as the direction each phase is tied to, as defined in the ATSPM signal controller database. Table 19 shows the ring diagram that is generally followed for each of the intersections. It should be noted that slightly different ring diagrams are also implemented within the same barrier depending on the demand at each approach. The overlap phases are special phases that circle the previous phases due to high demand during peak hours. For example, the overlap phases (OL 1 and OL 2 in Table 18) of intersection 1270 circle the major through movements (EB-T and WB-T) during weekday morning peak hours. OL 3 and OL 4 are special cases of concurrent pedestrian phase for intersection 2065.

Table 17 Phases for the Signals in Corridor A

ID	2095	2090	2085	2080	2075	2070	2065
WB-T	6	2	6	6	6	6	6
EB-T	2	6	2	2	2	2	2
WB-L	-	5	1	1	1	1	1
EB-L	-	1	5	5	5	5	5
SB-T	-	8	4	4	4	4	4
NB-T	-	4	8	8	8	8	8
SB-L	-	3	7	7	7	-	7
NB-L	3	7	3	3	3	-	3
OL 1	-	-	9	9	-	-	-
OL 2	-	-	13	13	-	-	-
OL 3	-	-	-	-	-	-	11
OL 4	-	-	-	-	-	-	16
Type	5	2	1	1	2	4	3

EB = Eastbound, WB = Westbound, NB= Northbound, OL = Overlap, SB = Southbound, T = Through, L = Left, OL = Overlap

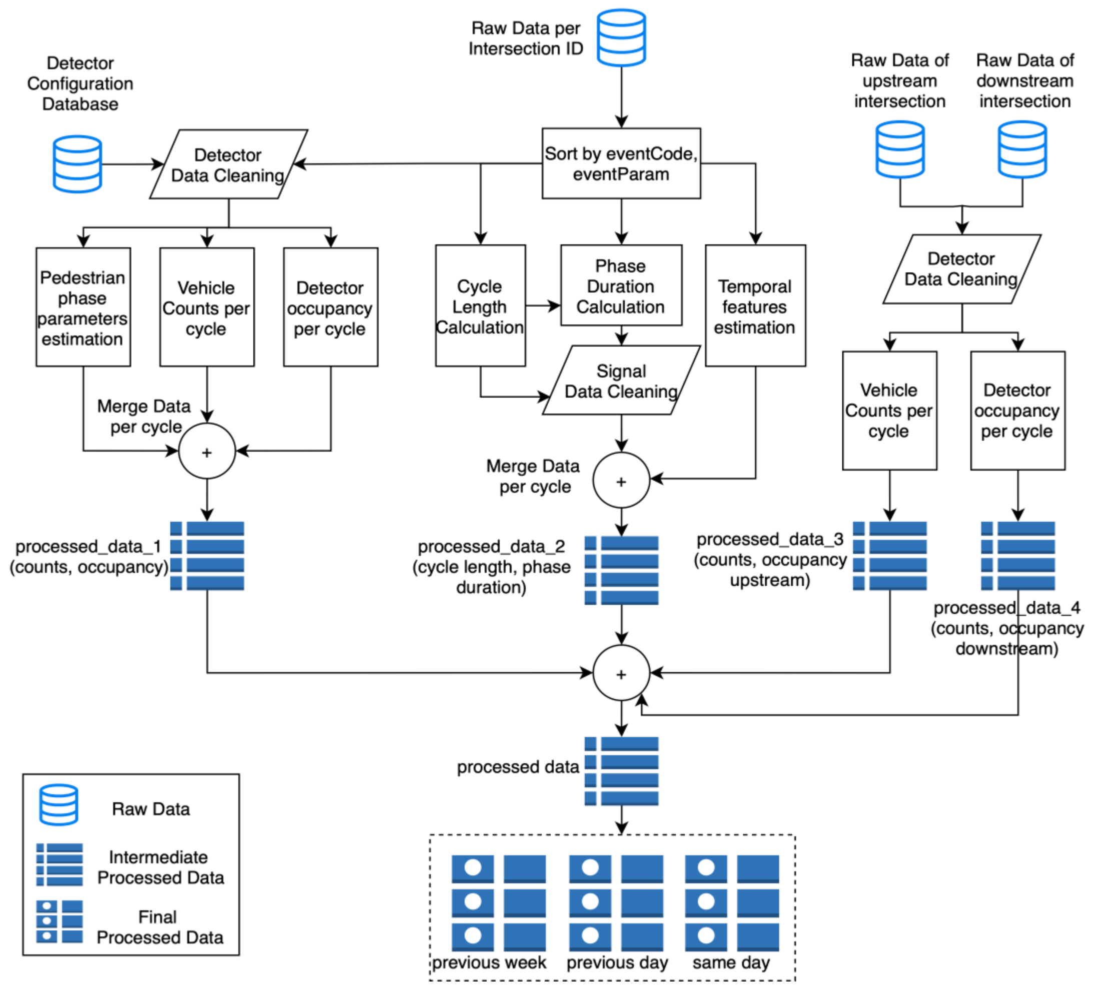
Table 18 Phases for the Signals in Corridor B

ID	1265	1270	1275	1280	1285	1290	1295	1300	1305	1130
WB-T	6	6	2	2	6	2	6	6	6	6
EB-T	2	2	6	6	2	6	2	2	2	2
WB-L	-	1	-	5	1	5	1	1	1	1
EB-L	5	5	1	1	5	1	5	5	5	5
SB-T	-	4	8	8	4	8	4	4	4	4
NB-T	8	8	-	4	8	4	8	8	8	8
SB-L	-	7	-	3	-	-	7	-	-	7
NB-L	-	3	-	7	-	-	3	-	-	3
OL 1	-	10	-	-	-	-	-	-	-	-
OL 2	-	14	-	-	-	-	-	-	-	-
Type	5	2	4	2	3	3	1	3	3	1

EB = Eastbound, WB = Westbound, NB= Northbound, SB = Southbound, T = Through, L = Left, OL = Overlap

Table 19 Ring Diagram of all Intersections in Study Area

ID	Ring Diagram							Notes
1265	Ring 1		2	8				T-intersection
	Ring 2	5	6					
1270	Ring 1	1	2	3	10	4		phases 10, 14 are repetition of phases 2, 6
	Ring 2	6	5	8	14	7		
1275	Ring 1	1	2	8				T-intersection
	Ring 2	6						
1280	Ring 1	1	2	3	4			
	Ring 2	5	6	8	7			
1285	Ring 1	1	2	3	8	7	4	
	Ring 2	5	6					
1290	Ring 1	1	2	3	8	7	4	
	Ring 2	6	5					
1295	Ring 1	1	2	4	3			
	Ring 2	6	5	7	8			
1300	Ring 1	1	2	4				No protected left for minor road
	Ring 2	5	6	8				
1305	Ring 1	1	2	4				No protected left for minor road
	Ring 2	5	6	8				
1130	Ring 1	1	2	3	4			
	Ring 2	6	5	7	8			
2095	Ring 1	2	3					T-intersection
	Ring 2	6						
2090	Ring 1	1	2	4	3			
	Ring 2	6	5	7	8			
2085	Ring 1	1	2	9	4	3		phases 9, 13 are repetition of phases 1, 5
	Ring 2	6	5	13	7	8		
2080	Ring 1	1	2	9	3	4		Phases 9, 13 are repetition of phases 1, 5
	Ring 2	5	6	13	7	8		
2075	Ring 1	1	2	3	4			
	Ring 2	6	5	7	8			
2070	Ring 1	1	2	4				No protected left for minor road
	Ring 2	6	5	8				
2065	Ring 1	1	2	3	4	11		Phase 11 is 1 and 16 is a concurrent pedestrian phase
	Ring 2	5	6	8	7	16		



Raw Data per Intersection ID
Detector Configuration Database
Detector Data Cleaning
Sort by eventCode, eventParam
Raw Data of upstream intersection
Raw Data of downstream intersection
Pedestrian phase parameters estimation
Vehicle Counts per cycle
Detector occupancy per cycle
Cycle Length Calculation
Phase Duration Calculation
Temporal features estimation
Detector Data Cleaning
Signal Data Cleaning
Vehicle Counts per cycle
Detector occupancy per cycle
Merge Data per cycle
Merge Data per cycle
processed_data_1 (counts, occupancy)
processed_data_2 (cycle length, phase duration)
processed_data_3 (counts, occupancy upstream)
processed_data_4 (counts, occupancy downstream)
processed data
Raw Data
Intermediate Processed Data
Final Processed Data
previous week
previous day
same day

Figure 34 Data Processing

The overall data preparation and preprocessing steps have been illustrated in Figure 34. Each intersection and its respective upstream and downstream were taken. After sorting, the data was passed through two separate channels; one channel processed the data to obtain counts and occupancy while the other channel was responsible to calculate signal timing parameters such as cycle length and phase length. The data was also cleaned to remove any missing detector information. The phase duration and cycle lengths were also matched for consistency. Finally, four immediate processed databases were obtained which were merged based on cycle start and end time. This database was then split into three sub-databases based on same-day data, previous-day data, and previous-week data. This was so done because travel demands across these time frames are largely similar. The details regarding calculating cycle volume, occupancy, cycle lengths, and phase durations are elaborated on in the subsections.

Cycle Volume Calculation

Cycle volume is calculated from the ATSPM data based on the detector ON and OFF events. All of the selected intersections have loop detectors for vehicle detection. The detectors stay on so long as it is occupied. It can also be used to calculate occupancy as will be shown later. The loop detectors can be divided into two types based on their location: stop bar detectors (SBD) are placed near the stop line and advanced count detectors (ACD) are placed at various lengths from the stop line. The selected 17 intersections have two sets of ACDs at each lane (for the major roads): the first set is placed at a distance of 75/150 feet from the stop line and the other is placed at 150/300/330 feet from the stop line. ACDs can be directly used to calculate the number of vehicles during the green time. During the red time, it can also indicate counts till a queue forms and reaches the ACD. SBDs are usually 30 feet in length and ACDs are 6 feet. The width of all detectors is almost equal to the lane width.

A typical setup with the different types of detectors at ATSPM controlled intersections is shown in Figure 35. For the study area, major roads are in the east-west direction and minor roads are in the north-south direction. The major through movements only has the ACDs whereas the left turn movements have only SBDs. All the minor movements have only SBDs. It should also be mentioned here that not all 17 intersections follow the detector configuration exactly as shown in Figure 35. The placement of detectors is often modified due to geometric and/or budget constraints. Therefore, in our study, we have separated the detectors based on the types per phase, and vehicle counts were aggregated per cycle per detector type per phase. For the cases, where there are multiple count detectors on the same lane (the EB-T, WB-T lanes in Figure 35), the detectors farthest away from the intersection were used to aggregate the count values.

In addition, the cycle volume at the upstream and downstream intersection of the target intersections was also calculated and added as a feature. As an example, the volume of intersections 1265 and 1275 was also added to the feature values when preparing the data for 1270.

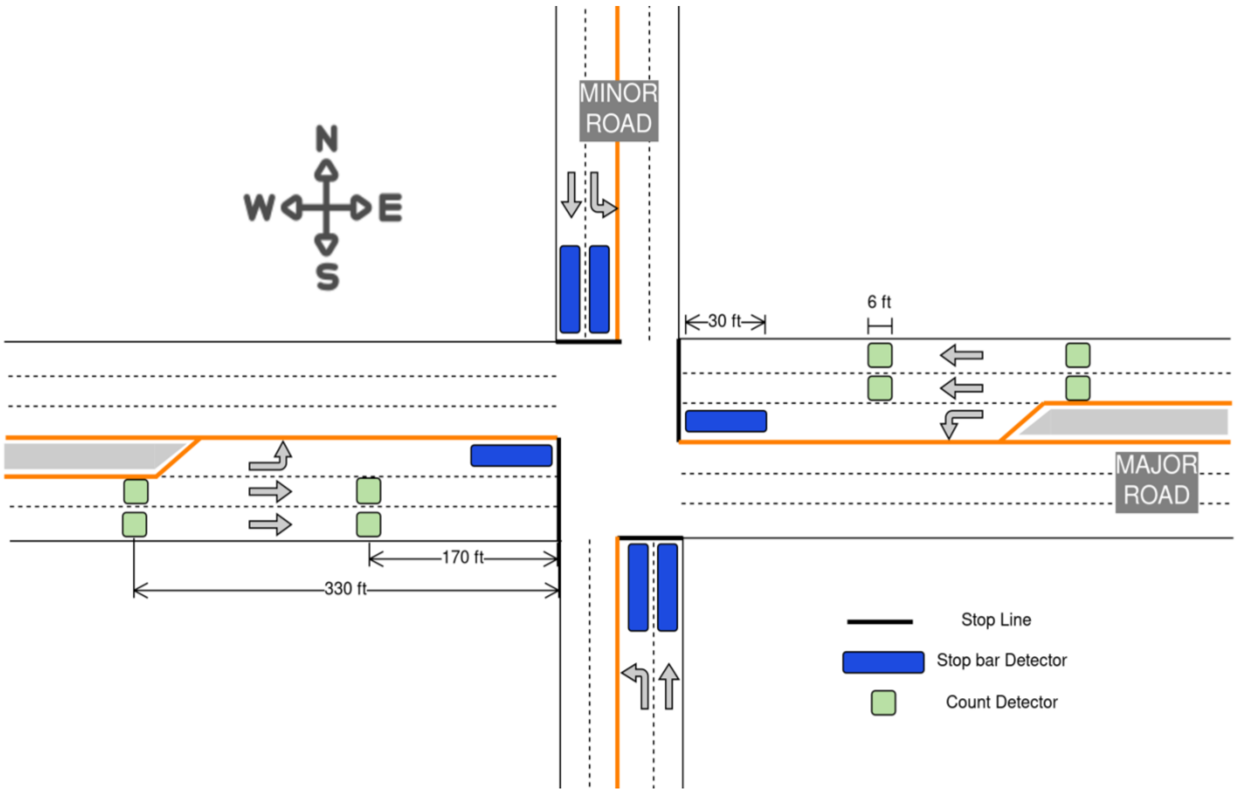


Figure 35 Detector configuration at intersections

Cycle Occupancy Calculation

In addition to cycle volume, occupancy was also calculated for each of the detector types. The time difference between consecutive ON and OFF state of a detector was used to calculate occupancy. The stop bar detectors report the best occupancy when the light is red, and occupancy directly equals to how long a vehicle is stopped, and count detectors report occupancy if the end of queue reaches the detector. Similarly, occupancy values of the upstream and downstream intersections were also used as features.

Signal Timing Parameters Calculation

The ATSPM controllers report the active green, yellow and red times of all the phases for an intersection. Usually, the yellow time, red time, and red clearance time for an intersection is fixed and depends on the geometric characteristics of the intersection. Therefore, it is enough to calculate the total phase length. The ATSPM database reports two parameters called *eventCode* and *eventParam* to indicate the phase and the associated state change, respectively. The phase number is indicated by the *eventParam* while the action associated with that phase is represented by the *eventCode.* Table 20 shows an example of the phases within a cycle at a morning peak. Based on this table, a ring diagram can be generated as shown in Figure 36 that can be used to calculate phase lengths. In this figure, phases 1, 2, 5, and 6 are the major movements while 4 and 8 are the minor movements. For example, the *eventCode* 0 represents when a phase turns ON and 12 when it turns OFF. Therefore, from the Table IV, phase 2 turns ON at 09:34:21 and turns OFF at 09:35:30.

The cycle length can also be calculated by dividing the phases in two different groups. Usually, the major movements belong to *phase group 1* (phases 2, 6, 1, 5) and the minor movement to *phase group 2* (phases 4, 8, 3, 7). Cycle length is calculated as the time difference between the start of *phase group 1* and the end of *phase group 2.*

After calculation of the cycle lengths and phase lengths, it was seen that there were some duplicate or missing phases within a cycle. The raw ATSPM data comes with several limitations which could contribute to the errors generated. Thus, a series of logical checks were performed with the data to confirm that each of the cycles and phases were calculated properly. For example, the ring lengths were checked to see if they would match the total cycle length, duplicate phases were removed, etc. Moreover, since the models were trained based on previous samples of data (as shown in Table 21), it is important that the data samples be continuous. Therefore, the processed data samples were matched by comparing the end of a cycle to the start of the next cycle.

Table 20 Phase Duration Calculation from Raw Data

timeStamp	eventCode	eventParam	Notes
2019-10-01 09:34:21	0	2	Phase 2 turns ON
2019-10-01 09:34:21	0	6	Phase 6 turns ON
2019-10-01 09:35:30	0	1	Phase 1 turns ON
2019-10-01 09:35:30	12	2	Phase 2 turns OFF
2019-10-01 09:35:48	0	2	Phase 2 turns ON
2019-10-01 09:35:48	12	1	Phase 1 turns OFF
2019-10-01 09:37:28	0	5	Phase 5 turns ON
2019-10-01 09:37:28	12	6	Phase 6 turns OFF
2019-10-01 09:37:43	0	6	Phase 6 turns ON
2019-10-01 09:37:43	12	5	Phase 5 turns OFF
2019-10-01 09:39:08	0	8	Phase 8 turns ON
2019-10-01 09:39:08	0	4	Phase 4 turns ON
2019-10-01 09:39:08	12	2	Phase 2 turns OFF
2019-10-01 09:39:08	12	6	Phase 6 turns OFF
2019-10-01 09:39:25	12	4	Phase 4 turns OFF
2019-10-01 09:39:25	12	8	Phase 8 turns OFF

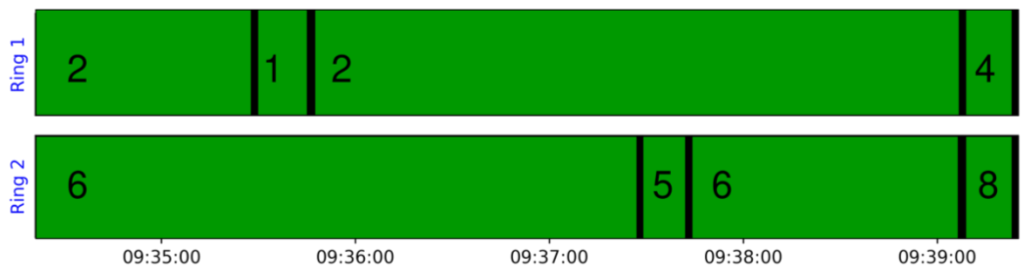


Figure 36 Ring Diagram for Table

Data Preprocessing

The data is then sliced into window sizes of various lengths and fed into deep learning algorithms which can predict the cycle lengths and phase lengths of future cycles. Input and output windows of sizes 1 to 6 were experimented with. Usually, signal timings are optimized based on not only the short-term traffic flow but also taking into account the long-term average traffic flow parameters. Therefore, instead of just taking the past cycles, the same cycles from the past day and past week were also taken into account. For example, if the cycle lengths and phase lengths of the next four cycles are to be predicted on a Thursday (October 17), the immediate few cycles of the same time are taken, as well as the same cycles during the previous day, October 16 and that of the same day of the previous week, October 10. This is illustrated in Table 21. Here i indicates the current cycle. It should be noted that $Cycle_{i-1}$ on each day will not occur at the same time and therefore, we calculated $Cycle_{i-1}$ of the previous day/week by averaging 2 or 3 cycles that are in close proximity to the current $Cycle_{i-1}$.

Table 21 *Illustration of feature selection for input and output window*

Thursday October 10 2019		Wednesday October 16 2019	Thursday October 17 2019	
$Cycle_{i-6}$	...	$Cycle_{i-6}$	$Cycle_{i-6}$	Input Features
$Cycle_{i-5}$	...	$Cycle_{i-5}$	$Cycle_{i-5}$	
$Cycle_{i-4}$	...	$Cycle_{i-4}$	$Cycle_{i-4}$	
$Cycle_{i-3}$	...	$Cycle_{i-3}$	$Cycle_{i-3}$	
$Cycle_{i-2}$	...	$Cycle_{i-2}$	$Cycle_{i-2}$	
$Cycle_{i-1}$	...	$Cycle_{i-1}$	$Cycle_{i-1}$	
			$Cycle_{i}$	Output Prediction
			$Cycle_{i+1}$	
			$Cycle_{i+2}$	
			$Cycle_{i+3}$	

Methodologies

In this section, details about the CNN-LSTM structure proposed in the chapter are discussed. Long short term memory (LSTM) networks (Hochreiter, 1997 #2337) belong to the recurrent neural network (RNN) family and are adept to overcome the shortcoming of conventional RNN: the vanishing gradient problem. RNNs are able to remember only recent information due to this limitation. LSTM can make use of both the short-term and long-term information to make a prediction. This is especially important in the prediction of signal timing because generally optimizations are done by considering both the short-term turbulent traffic flow and the long-term mean traffic parameters. LSTMs have an input layer, a hidden layer, and an output layer. While the input and output layers are traditional neurons, the hidden layers are specialized memory cells that can store information.

For the proposed model, a convolutional neural network (CNN) was also added on the top of the stacked LSTM structure. The input to the CNN layer is the processed data consisting of the cycle length, detector values, and other temporal parameters. The output of the CNN layer is flattened and fed into the stacked LSTM layer. If $x_1, x_2, \dots, x_n$ is the input vector, the output from the CNN layer can be represented by (1) where y_{ij}^1 is the output from the convolutional layer, σ represents the activation function, m is the index value, ω is the weight of the filter, x_{ij}^1 is the input layer and b_j^1 is the bias for the j^{th} feature.

$$y_{ij}^1 = \sigma(b_j^1 + \sum_{m=1}^{M} \omega_{m,j}^1 x_{i+m-1,j}^0) \qquad (1)$$

The output from the CNN layer is passed on to the stacked LSTM layers. Each cell of the LSTM layers consists of memory cells that have three multiplicative units: input, output and, forget gates. These memory cells aid in understanding the temporal relationships in a sequence. LSTM layers are also suitable because it addresses the vanishing gradient problem with traditional RNNs. The forget gate f controls when the previous memory will be forgotten, the input gate i controls the update of each unit and the output o gate decides the next hidden layer. The operation of each memory cell can be represented with (2), (3), and (4).

$$i_t = \sigma\,(W_{pi} \bullet y_t + W_{hi} h_{t-1} + W_{ci} \bullet c_{t-1} + b_i) \qquad (2)$$

$$f_t = \sigma\,(W_{pf} \bullet y_t + W_{hf} h_{t-1} + W_{cf} \bullet c_{t-1} + b_f) \qquad (3)$$

$$o_t = \sigma\,(W_{po} \bullet y_t + W_{ho} h_{t-1} + W_{co} \bullet c_{t-1} + b_o) \qquad (4)$$

Here W is the weight matrix, y_t is the flattened output from the CNN layer, b is the bias vector, $\bullet$ is used to indicate elementwise operation. The cell states c_t and hidden states h_t are calculated using (5) and (6).

$$c_t = f_t \bullet c_{t-1} + i_{t.}\sigma(W_{pc}p_t + W_{hc} \bullet h_{t-1} + b_c) \quad (5)$$

$$h_t = o_t.\sigma(c_t) \quad (6)$$

The final layer of the model consists of fully connected layer. Mathematically it can be represented by (7) where ω represents the weight of the i^{th} and j^{th} node for the layer $l-1$ and l respectively.

$$d_i^l = \sum_j \omega_{ji}^{l-1}(\sigma(h_i^{l-1}) + b_i^{l-1}) \quad (7)$$

<u>Network Architecture</u>

The overall network architecture is presented in Figure 37. The feature windows are concatenated to produce a 3D input structure similar to an RGB image which is then passed into the CNN layer. The 4x4 filters in this layer produce a 2D output which is flattened to input into the stacked LSTM layers. Three LSTM layers are used with each layer containing 150 cells. A similar architecture is followed for the phase duration prediction. It was important to separate the predictions of cycle length and phase duration since the mean and standard deviation between the two were significantly different and therefore the results involving MAEs would lead to incomplete conclusions. The output of the two branches of prediction gives the final signal timing prediction.

It is also important to show that once the model predictions are obtained, it is straightforward to reconstruct a signal timing diagram. For example, let us take the ring diagram of Intersection 2075. If c is the cycle length predictions and, $p_1,\ p_2, p_3, p_4, p_5, p_6, p_7, p_8$ are the phase predictions for this intersection, then based on the ring diagram we can obtain (8) and (9).

$$p_2 + p_1 = kc, \qquad p_5 + p_6 = kc \tag{8}$$

$$p_4 + p_5 = (1-k)c, \qquad p_7 + p_8 = (1-k)c \tag{9}$$

Here k indicates the fraction of time *phase group 1* is active within a ring. Based on the predictions from the proposed method, (8) can be solved for k and phases p_1 and p_2 can be placed on Ring 1 (any order of the phases can be applied) and, p_5 and p_6 can be placed on Ring 2. Similarly, the ring diagram can be completed with (9). Therefore, given the predicted outputs from the proposed model in Figure 37 signal timing diagram can be generated accordingly.

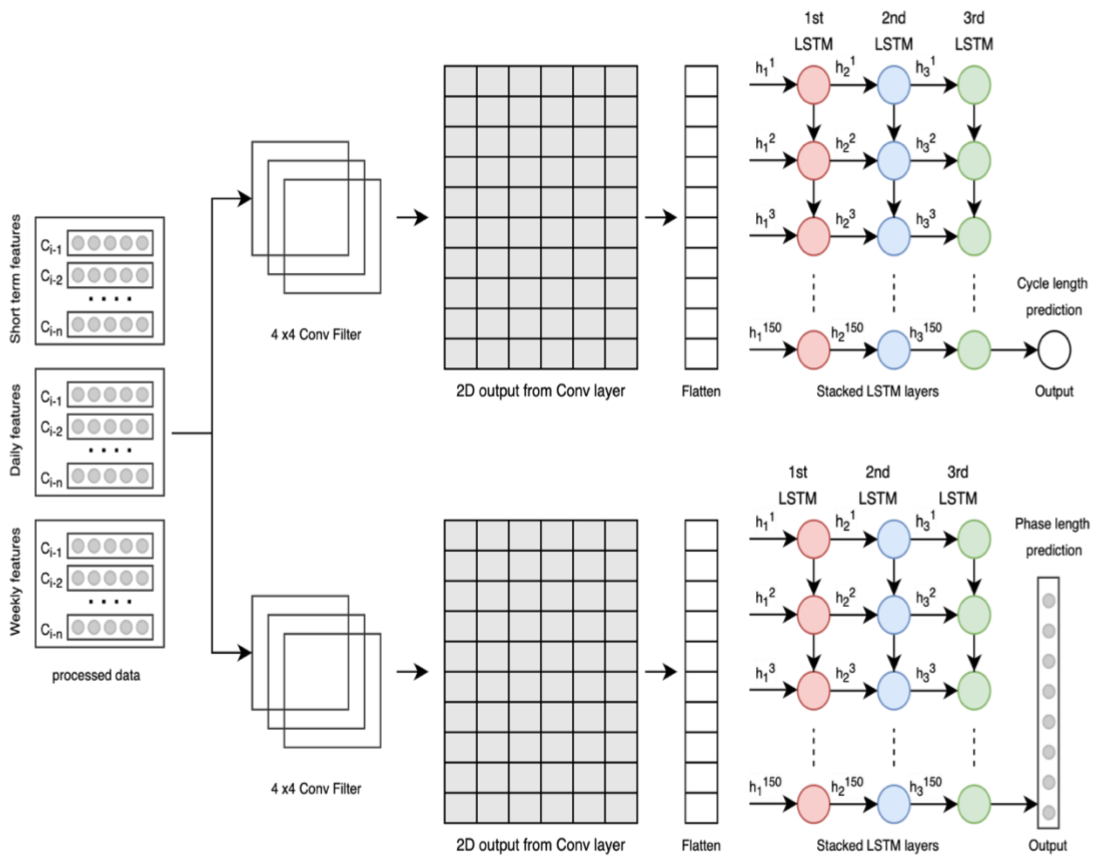


Figure 37 Model Architecture

Results

Performance metrics were evaluated for individual intersection models as well as combined intersection models. Based on the results, it seemed reasonable to group intersection based on different types to obtain better accuracy metrics. MAE and RMSE were taken as the metrics and were calculated with (10) and (11) where y_i is the predicted values and x_i is the actual values.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{10}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}} \tag{11}$$

The effect of window size will also be discussed along with the important features. The CNN-LSTM model was also compared with state-of-the-art time series forecasting models to show that it outperforms other methods in terms of performance. The data was divided into 70-30 ratio for training and testing. The trained model was also implemented at another intersection in a different corridor and impressive results were obtained.

Combined Model vs Model Based on Types of Intersection

Initially, the models were trained per corridor since it would be able to capture more variations within intersections and as such be an alternate to signal retiming with similar types of intersections. We also found that this combined method performs worse for some predictions since the model would average out the performance across different types of intersections. Due to this reason, it is worthy to separate the intersections into different types based on the phases as shown in Table 17 and Table 18. For example, the 7 intersections in Corridor A can be divided into five types. Intersections 2085 and 2080 have similar phase sequences and therefore can fall into the same type. Intersections 2090 and 2075 have eight phases each thereby belonging to the same type. The remaining three intersections are classified into individual types since 2095 is a T-intersection that merges into a freeway, 2070 does not have protected minor movements and 2065 has a concurrent pedestrian phase. The intersections in Corridor B can also be divided into 5 types.

Table 22 Combined MAE vs Type-Based MAE

		Type	1	2	3	4	5	Combined
Corridor A	Cycle length	MAE Train	12.69	8.78	**6.06**	133.62	10.36	18.08
		MAE Test	16.39	9.74	**6.34**	153.27	10.73	19.46
	Phase length	MAE Train	6.84	4.86	**2.73**	52.36	52.74	6.74
		MAE Test	7.01	5.04	**2.79**	52.74	31.84	6.76
Corridor B	Cycle length	MAE Train	**10.06**	16.54	19.10	62.94	17.42	19.29
		MAE Test	**11.18**	18.06	22.18	163.18	18.20	22.91
	Phase length	MAE Train	**5.46**	6.83	25.49	36.41	10.24	5.24
		MAE Test	**5.55**	6.86	25.75	37.37	10.58	5.28

As shown in Table 22, the combined model usually performs worse for all the intersection types except for type 4 in Corridor B. The reason being that type 4 is a T-intersection with very little demand on the minor roads. This essentially means that the intersection more or less runs in a free mode where demand is met whenever there is a vehicle on the minor road. This results in a large variation in raw cycle length values and therefore the prediction accuracy is also unsatisfactory. Furthermore, this also provides the insight that for such intersections (which have larger major road demand and lower minor road demand), traffic flow prediction from cycle to cycle is a difficult task.

The timing diagram of one such intersection is shown in Figure 38. Two cycles are shown here with the first cycle being considerably smaller than the second cycle.

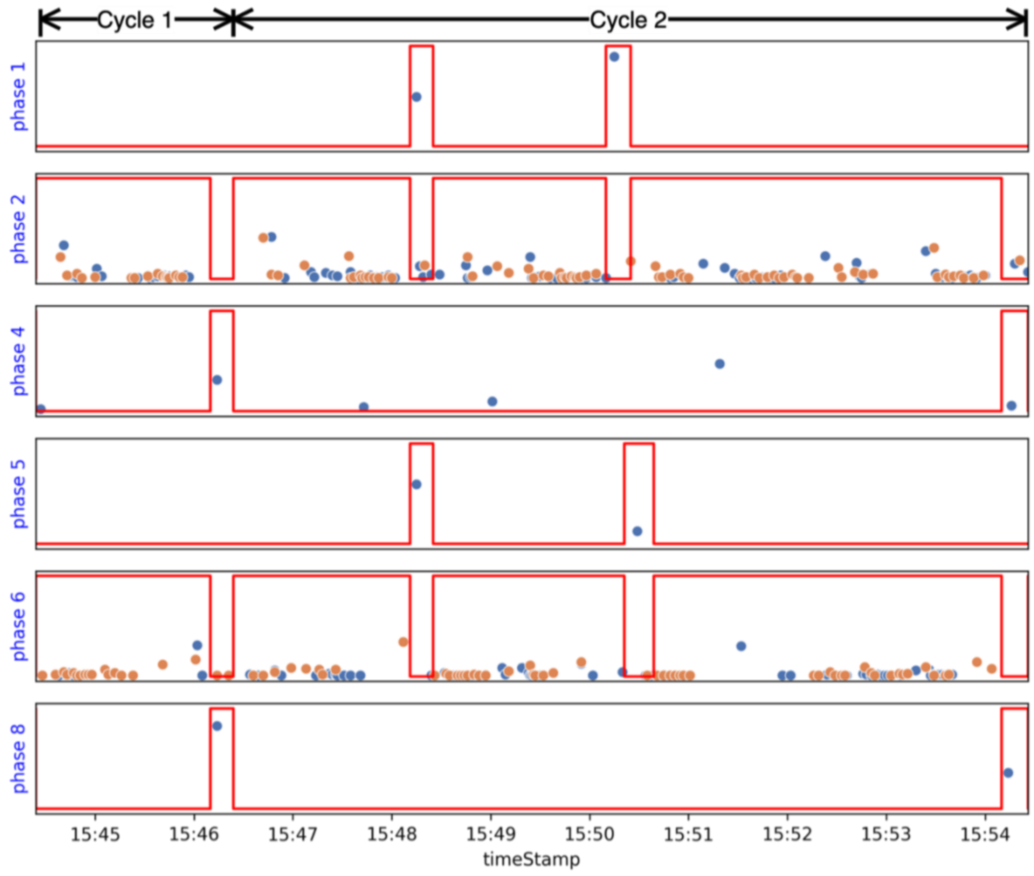


Figure 38 Phase Diagram along with Occupancy

The major roads have phases 1,2,5 and 6. Phases 4 and 8 are the minor road movements. Each of the scatter plots in the figure shows the occupancy and the corresponding time when the vehicle left the detector. As discussed before, the major through movements has predictable occupancy but the minor roads have vehicles intermittently passing. If we take phase 8, we see that one vehicle leaves the detector at 15.46 and the other vehicle at 15.54 and therefore one of the cycles is considerably longer than the other. For phase 4, we see that some vehicles leave the detectors during red. This indicates that those vehicles did not wait long enough to trigger the minor road phases. Understandably, those vehicles made a right turn movement.

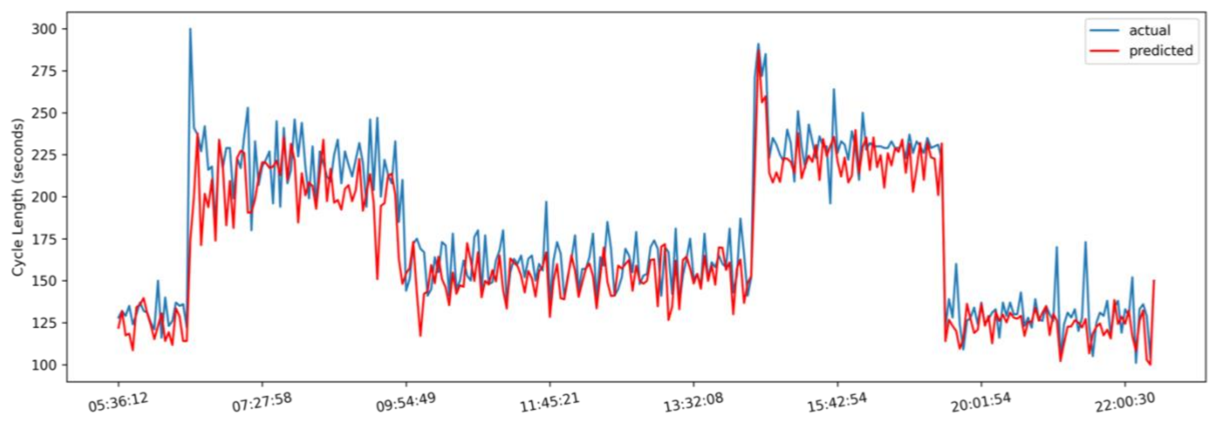


Figure 39 Actual vs Predicted Cycle Length for CNN-LSTM

Figure 39 shows the actual and predicted cycle lengths for one entire day for a type 1 intersection in Corridor B. The predicted cycle length closely follows the actual values for the test dataset.

Model Comparison

Predicting signal phasing and timing can be classified as a time series forecasting problem and therefore, the CNN-LSTM model has been compared with other popular models in this field. Initially, Gradient Boosting (GB) , Bagging Regressor (BR), Random Forest (RF), XGBoost (XGB), XGBoost-Random Forest (XGBRF) were tried among the tree-based ensemble models. Three deep learning models such as artificial neural network (ANN), long short-term memory (LSTM), and gated recurrent units (GRU) were also investigated. Of all the nine models, CNN-

LSTM had the least mean absolute error and therefore two more variants of the LSTM were also tried: CNN-LSTM and EMD-LSTM. CNN-LSTM performed better than the base LSTM model while EMD-LSTM performed worse and therefore not mentioned in the results. Table 24 and Table 23 show all the RMSE and MAE values for the individual models along with the window size. It is also seen that LSTM performs better than CNN-LSTM when the prediction horizon is smaller. For example, for the first row in Table 24(a) where input and output windows are 5 and 1 cycles respectively, LSTM has a better MAE and RMSE. As the output window size is increased, the performance of CNN-LSTM improves. Again, the RMSE values of CNN-LSTM are almost always lower than LSTM thereby making it the best model in this scenario.

Table 24 also compares all the models across all the different window sizes for the case of cycle length prediction. The window sizes are shown in the first column with the notation a, b where a is the number of past cycles used to train the model and b is the number of cycles in the prediction horizon. Table 24 (a) and (b) represents the two different corridors. The MAE values, as well as the RMSE values, are presented for comparison. It can be seen that the MAE values for Corridor A are lower than that of Corridor B. The reasoning can be attributed to the fact that A is adaptive signal controlled and therefore the signal timing patterns are coordinated with the traffic flow to some extent. Across the two tables, CNN-LSTM has the least MAE value: 19 seconds for Corridor A and 23.3 seconds for Corridor B which is better than the next best model RF by 2 seconds. GRU, XGB, and XGBRF report almost similar values of MAE. CNN-LSTM is also the model that reports the least RMSE. If RMSE is considered, the lowest value is 63.0 seconds for Corridor A and 62 seconds for Corridor B. Concerning the appropriate window size, it was noted that the MAE and RMSE values showed slight improvements when the window size was maximum at 6, 6. A larger window size gave the models a better idea about the past traffic pattern and helped to identify suitable signal timing for the next few cycles. But regardless the MAE values were not starkly different between different window sizes. Therefore, appropriate window size would depend upon the requirements of specific applications.

Table 23 MAE and RMSE for Cycle Length Prediction

Window Size	BR		GB		GRU		LSTM		CNN-LSTM		MLP		RF		XGB		XGBRF	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
5,1	24.8	89.4	23.8	80	26.4	89.7	20.8	87.9	22.2	89.4	50.5	106	22.2	79.3	22.9	80.3	23.8	81
5,2	24.1	76.2	24.1	76.7	25.4	84.8	21.1	81	20.9	78.3	52.1	106	22.4	76.7	22.8	76.6	23.6	77.3
5,3	24.2	76.8	23.5	73.4	25.1	80.3	20.5	77.8	20.6	73.5	50.5	99	22	72.3	22.7	73.6	23.4	73.7
5,4	24	73.3	23.2	68.1	24.4	74.6	19.9	71.2	20.0	70.3	51.4	94.4	21.5	67	22	68	22.8	68
5,5	24	72.6	23.3	70.6	28.4	76.8	20	72.3	20.1	73.5	29.1	73.9	21.7	69.5	22.3	70.6	23	70.5
6,1	24	75.5	23.7	80.3	41.8	93.8	21.3	90	22.4	87.8	50.6	108.1	22.3	81.2	22.9	81.4	23.7	81.6
6,2	24.1	78.7	23.5	75.4	24.9	82	20.4	77.6	19.8	72.7	50.2	99.6	22.2	74.4	22.7	75.6	23.4	75.5
6,3	23.8	75.4	23.1	69.7	23.6	75.7	19.6	71.2	**19.0**	**63**	49.8	94.6	21.5	68.2	22	69.2	22.7	69.2
6,4	24	74.2	23.2	71.6	24.9	77.3	19.9	72.5	19.3	67.4	50	97.1	21.7	70.3	22.3	71.3	23	71.4
6,5	24	70.7	23.2	68.4	25.2	75.4	20	72.3	19.5	67.3	49.3	92.9	21.3	67	21.9	68.1	22.6	68.3
6,6	23.9	68.1	22.7	66.9	24	72.6	19.5	70.7	19.7	69.6	49	91.3	21	65.8	21.7	66.9	22.5	67.5

(a) Corridor A

Window Size	BR		GB		GRU		LSTM		CNN-LSTM		MLP		RF		XGB		XGBRF	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
5,1	24.8	89.4	28.6	64.4	29.2	68.3	24.2	69	24.5	67.9	41.4	68.4	28.8	65	28.3	65	29.6	65.9
5,2	24.1	76.2	28.9	64.5	31.3	72	24.4	67.6	24.7	66.9	102.6	124.7	29.1	64.9	28.7	64.4	29.8	65.5
5,3	24.2	76.8	28.9	64.4	29.6	69	24	67	24.4	66.9	43.3	73.9	28.9	64.7	28.5	64.4	29.7	65.5
5,4	24	73.3	28.5	63.3	28.4	66.6	23.7	65.6	24.1	65.2	53.2	76.9	28.6	63.6	28.2	63.1	29.4	64.2
5,5	24	72.6	28.6	61.9	28.3	66.5	23.8	64.4	24.2	64.1	41.8	76	28.7	62.3	28.4	62	29.6	63.1
6,1	24	75.5	29	66.1	32.4	70.1	25.1	71.7	24.9	69.5	41	76.7	29.2	66.7	28.8	66	30.1	67.3
6,2	24.1	78.7	28.8	65.4	29.5	70.4	24.3	67.5	24.2	67	40.9	75.8	29	65.7	28.5	65.5	29.7	66.6
6,3	23.8	75.4	28.4	63.7	30.3	70.1	23.7	65.2	23.9	64.9	40.5	74.9	28.7	64.2	28.2	63.6	29.4	64.7
6,4	24	74.2	28.7	62.8	28.5	66.9	23.9	65.4	24.1	64.6	42.2	74.2	28.8	63.2	28.3	62.7	29.5	63.7
6,5	24	70.7	28.4	63.4	28.2	66.8	23.6	65.6	23.8	65.1	44	76.4	28.5	63.5	28.2	63.3	29.5	64.3
6,6	23.9	68.1	28.2	63.4	27.8	63.3	23.2	62.4	**23.3**	**62**	44.1	72.7	28.2	60.4	28	60.1	29.2	61.2

(b) Corridor B

Table 24 MAE and RMSE for Phase Duration Prediction

Window Size	BR		GB		GRU		LSTM		CNN-LSTM		MLP		RF		XGB		XGBRF	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
5,1	6.7	18.5	5.8	17.0	7.4	20.1	5.3	18.1	5.3	18.0	31.3	41.7	5.7	17.2	5.6	17.0	5.8	17.2
5,2	6.6	18.0	5.8	16.5	7.7	20.0	5.5	17.8	5.5	17.8	39.0	47.1	5.8	16.7	5.6	16.5	5.9	16.8
5,3	6.6	17.1	5.8	16.0	8.4	20.0	5.5	17.0	5.4	16.9	46.3	58.6	5.7	16.0	5.7	16.0	5.9	16.2
5,4	6.5	16.2	5.7	15.1	9.3	20.4	5.5	16.2	5.4	16.1	16.3	26.7	5.7	15.2	5.6	15.1	5.8	15.2
5,5	6.6	16.7	5.8	15.6	9.0	20.2	5.6	16.8	5.4	16.8	22.4	34.0	5.8	15.7	5.7	15.6	5.9	15.7
6,1	6.7	18.8	5.8	17.1	7.7	20.5	5.4	18.5	5.5	18.4	47.0	56.0	5.3	5.6	5.6	17.1	5.8	17.4
6,2	6.6	17.5	5.8	16.4	7.5	19.3	5.4	17.2	5.4	17.2	42.9	53.1	5.4	5.7	5.6	16.4	5.9	16.5
6,3	6.5	16.4	5.7	15.3	8.0	18.9	5.4	16.4	5.4	16.2	28.7	39.1	5.5	5.6	5.7	15.3	5.8	15.4
6,4	6.6	16.8	5.8	15.7	8.6	20.3	5.5	16.9	5.4	16.6	17.7	28.8	5.6	5.7	5.6	15.7	5.9	15.9
6,5	6.6	16.2	5.8	15.2	9.0	19.9	5.5	16.4	5.4	16.4	27.9	39.0	5.6	5.7	5.6	15.3	5.9	15.3
6,6	6.5	16.0	5.8	15.0	9.5	20.4	5.5	16.1	**5.4**	**16.1**	39.1	50.4	5.6	5.7	5.6	15.0	5.9	15.2

(a) Corridor A

Window Size	BR		GB		GRU		LSTM		CNN-LSTM		MLP		RF		XGB		XGBRF	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
5,1	9.0	18.6	7.4	16.8	9.3	20.6	6.8	17.9	6.7	17.6	64.5	70.0	7.9	17.4	7.4	16.8	7.8	17.4
5,2	9.0	18.5	7.7	16.8	10.1	21.4	7.1	17.8	7.0	17.8	29.3	40.0	7.9	17.1	7.6	16.8	7.9	17.2
5,3	9.0	18.6	7.7	17.0	10.3	21.4	7.0	18.1	7.0	18.0	15.1	25.9	7.9	17.3	7.6	17.0	7.9	17.4
5,4	9.0	18.4	7.7	16.8	10.5	21.4	7.0	17.9	7.0	17.9	15.2	24.7	7.9	17.1	7.6	16.8	7.9	17.1
5,5	9.0	18.0	7.7	16.4	11.4	22.1	7.1	17.5	7.0	17.5	15.0	24.5	8.0	16.8	7.7	16.4	8.0	16.8
6,1	9.1	18.9	7.6	17.0	9.8	20.9	6.9	18.0	6.8	17.9	47.9	55.2	7.9	17.6	7.4	17.0	7.9	17.5
6,2	9.0	18.9	7.6	17.2	9.8	21.2	7.0	18.3	7.0	18.2	53.4	61.7	7.9	17.5	7.5	17.1	7.9	17.6
6,3	8.9	18.4	7.6	16.8	10.2	21.2	7.0	17.8	7.0	17.8	15.0	24.7	7.9	17.1	7.5	16.8	7.8	17.2
6,4	9.0	18.2	7.7	16.6	10.9	21.8	7.1	17.7	7.0	17.6	44.4	57.5	8.0	17.0	7.6	16.6	7.9	16.9
6,5	9.0	18.4	7.7	16.9	11.2	22.2	7.1	18.0	7.0	17.9	23.1	46.2	7.9	17.2	7.6	16.9	7.9	17.2
6,6	8.9	17.7	7.7	16.2	11.1	21.4	7.0	17.3	**7.0**	**17.2**	22.3	31.1	7.9	16.5	7.6	16.2	7.9	16.5

(b) Corridor B

Transferability Analysis

To understand whether such models trained on several intersections would be able to predict signal timings correctly for other intersections, we have used the trained model on a separate intersection 2540. The intersection is shown in Figure 40. The raw data for this intersection was processed similarly to the other intersections. Type 1 intersection from Corridor B was used to train the model. The cycle length prediction result reported an MAE of 12.75 seconds which is close to the reported test MAE of 10.06 seconds (from Table 22). Therefore, we can conclude that the model can be successfully extended to other intersections.

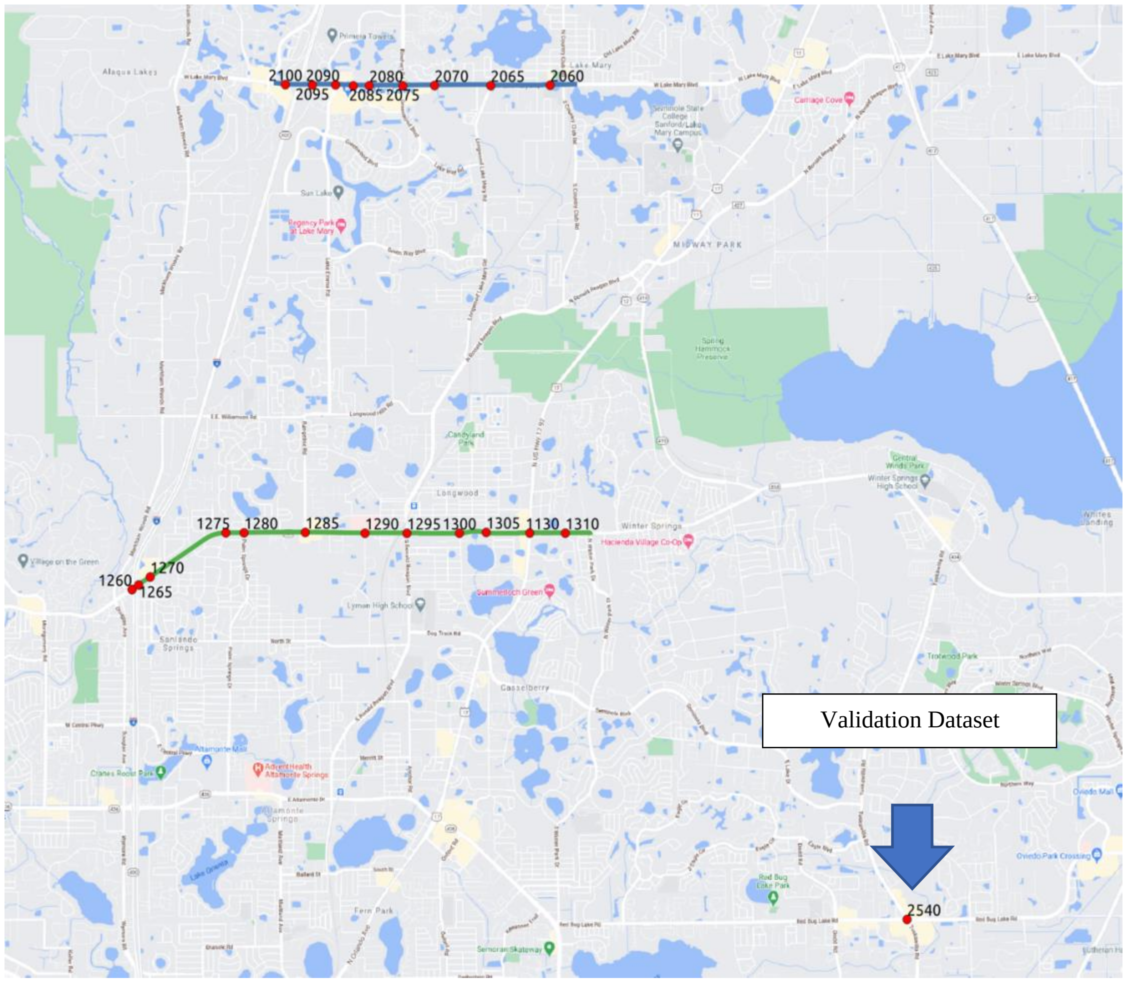
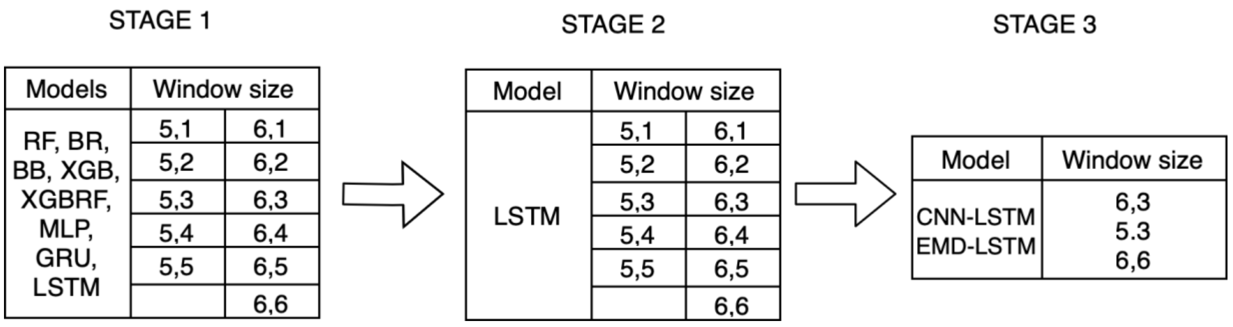


Figure 40 Validation Dataset

Intersection 2540 also serves as a validation set for the proposed model. Using a validation set to confirm the results of a machine learning model has been emphasized in several previous studies (Bleeker et al., 2003; Guyon, 1997; Larsen et al., 1996).

To summarize, we present the three-stage evaluation in Figure 41 that helped to refine the final proposed model. At Stage 1, eight different models were tried along with a combination of 11 different window sizes. This stage took the combined data of each corridor. The results from Stage 1 clearly showed that LSTM was the best model in terms of MAE. Therefore, at Stage 2, only LSTM was experimented with. Intersection type-based data was used in this stage. Stages 1 and 2 also helped to narrow down the window size to three values. Therefore, at Stage 3, only these three window sizes were tried along with modification of the LSTM model from Stage 2: CNN-LSTM and EMD-LSTM. CNN-LSTM showed lower MAE values.


STAGE 1

Models	Window size	
RF, BR, BB, XGB, XGBRF, MLP, GRU, LSTM	5,1	6,1
	5,2	6,2
	5,3	6,3
	5,4	6,4
	5,5	6,5
		6,6

STAGE 2

Model	Window size	
LSTM	5,1	6,1
	5,2	6,2
	5,3	6,3
	5,4	6,4
	5,5	6,5
		6,6

STAGE 3

Model	Window size
CNN-LSTM EMD-LSTM	6,3 5.3 6,6

Figure 41 Summary

Summary

The research work presented in this paper identifies a common problem in the signal timing practice: retiming, and proposes a solution based on deep learning architecture. Based on high resolution detector data the proposed architecture is able to predict signal timing sequences for the next few cycles. A comprehensive study was carried out comparing both adaptive and actuated signal control corridors. A total of 17 intersections were studied to show the feasibility of the application of the proposed method. Cycle length prediction reported MAE as low as 6.06 seconds and phase duration prediction showed MAE as low as 2.73 seconds. The proposed model also was validated on another intersection from a different corridor with good accuracy metrics.

There are many potential applications of predicting the SPaT timing of the upcoming cycles. It can make route planning and trajectory estimation (Islam and Abdel-Aty, 2021a) more efficient in a connected environment since future states of the signals can be predicted. This would aid in relevant studies where vehicle velocity is optimized to traverse intersections at green times and thereby reducing the carbon footprint (Asadi and Vahidi, 2010) or to find optimal velocity (Koukoumidis et al., 2011). The predicted signal timings can also be used to aid vehicles in the dilemma zone if the predictions can be transferred to the vehicles using on-board units. Most prominently, the signal retiming effort can be reduced to a great extent. Recent studies related to safety that try to predict pedestrian and vehicle conflicts also use signal timing information (Zhang et al., 2020a; Zhang et al., 2020b) can also benefit from having an extended SPaT information up to 6 cycles in the future.

CHAPTER 7: CONCLUSION

This dissertation focuses on the use of sensor-based data to improve safety and mobility in the field of transportation. All technologies worldwide are generating a lot of data with embedded sensors. The roadway networks have different types of sensors: intrusive sensors like loop detectors, pneumatic tube sensor, etc. and non-intrusive sensors like camera, radar, etc. Each new vehicle on the road is also generating data from on-board units, camera, GPS, etc. and storing data to cloud. Huge volume of data is being generated by all these sensors. It is becoming increasingly challenging to use this data for the betterment of human lives both from operations and safety point of view. Machine learning models can aid in this regard to help achieve critical objectives with sensor data.

In Chapter 3, we have used data generated from smartphone sensors to classify lane change and predict trajectories at intersections. Smartphone data contains several sensor information like accelerometer, magnetometer, gyroscope, etc. which can be further fused to obtain azimuth pitch and roll. Data cleaning was done using low-pass filter system which can be used to accurately identify lane changes. Different machine learning models were investigated like AdaBoost, Extra Tree, Gradient Boosting, Support Vector Classification, etc. Using this information, the trajectory of the vehicle was estimated using Chebychev polynomial. The main contributions of the chapter were in identifying new feature data like azimuth that can be used to detect lane changes. The turning movement prediction at intersections by using GPS can also be highlighted since most previous studies only estimate trajectory for road segments and not intersections. It also shows how traditional polynomial fitting algorithms can be used with machine learning methods. These types of systems can be especially useful for intersections in which there are a large volume of turning vehicles as well as pedestrians. Safety parameters like TTC, PET can be calculated using the trajectories. This work also shows how smartphones can be used as OBUs for the vehicles that have no connectivity. Since the road to full connectivity and automation is quite some time away, a large number of vehicles can benefit from OBU emulators, such as smartphones. The market penetration rate will also be very high. Building on this study, future work can concentrate on including safety performance measure with trajectory prediction.

In Chapter 4, transportation mode recognition was carried out. While this has been studied extensively recently, we have used variational autoencoder which has not been studied before to the best of the knowledge of the author. Variational autoencoder has been used as a data generation tool to upsample the minority class in the dataset since there can be a certain class(es) that are underrepresented in the dataset. SHL dataset has been used in this study which contains transportation mode data collected from smartphones. The data generated from variational autoencoder was compared to the real data. The statistical tests showed that the synthetic data was statistically similar to the real data. It was also trained on different learning algorithms to understand the performance on accuracy of classification. Hybrid data models were also experimented with and improvement in accuracy as high as 20% were obtained. The main contributions of this work could be attributed to the data generation of undersampled class using variational autoencoder. Usually, data generation schemes are of interest in the computer vision field, but this study shows that it can be used to replicate accelerometer and magnetometer data as well.

Chapter 5 also uses variational autoencoder (VAE) for data augmentation purposes. Crash data was collected from MVDS sensors which are radar sensors on freeway. Usually, the ratio of crash to non-crash events is highly imbalanced as such it is not possible to train machine learning models with such data. The approach most research take is that they reduce the non-crash samples to match the crash samples. Some studies use synthetic minority oversampling techniques as well. But in both cases, the entire non-crash sample is not used. Using variational autoencoder, we generate synthetic data from a handful of crash samples to match the non-crash samples. In this way, the entire real dataset is used to train the models. The mean, standard deviation, of the generated and real data were compared to show similarity between the two. The data distribution was also compared. The dataset was then evaluated on three crash prediction models such as logistic regression, support vector machine and artificial neural network. Specificity, sensitivity and AUC were measured. It was also compared to state- of-the-art minority sampling techniques like SMOTE and ADASYN and the results of the VAE model showed improvements in terms of the accuracy metrics. It was also compared do DCGAN and improved metrics were obtained. The insights obtained from the confusion matrix was also highlighted. VAEs can be used to generate samples from the latent space that represent conditions that the situation is favorable for a crash. Future studies can focus on using the synthetic dataset on complex and non-linear models. VAEs can be used in the pipeline that will aid in training more non-linear models.

In Chapter 6, signal states were predicted based on detector data. Seventeen intersections from two corridors were used in this study. One of the corridors run adaptive signal control and the other run actuated signal control. The intersections were equipped with loop detectors and video cameras that reported sensor on and off states. These were processed to obtain volume, occupancy, etc. The signal timing data from controller was also collected. Cycle lengths and phase durations were calculated. Detector data can be used to predict the signal states up to six cycles in the future. Several learning algorithms like ensemble trees and deep reinforcement learning methods were investigated. CNN-LSTM showed the best results in terms of mean absolute and root mean square error. Cycle length can be predicted with MAE as low as 6.06 seconds and phase length can be predicted with MAE as low as 2.73 seconds. Signal phasing and timing prediction can be used for route planning and trajectory prediction. It would also solve the traditional signal retiming efforts at every intersection. A newly built signal or intersection can benefit from the signal timing prediction from another intersection with similar detector configuration. Knowing signal states of six cycles in the future would also aid in optimal velocity calculation and thereby reducing the carbon footprint. The predicted signal states can also aid in reducing the dilemma zone if it can be transferred to vehicles using OBUs or OBU-emulators. Pedestrian safety studies can also use predicted signal states to calculate potential conflicts.

In conclusion, the efforts in this dissertation show the use of various learning algorithms in traffic engineering safety and mobility applications. Starting with trajectory prediction with smartphone OBU-emulator, data augmentation of smartphone data and crash data, and ending with signal states prediction display the use of trained algorithms in urban transportation systems.

REFERENCES

Abdel-Aty, M., Uddin, N., Pande, A., Abdalla, M.F., Hsia, L., 2004. Predicting freeway crashes from loop detector data by matched case-control logistic regression. *Transportation Research Record* 1897(1), 88-95.

Abdulhai, B., Pringle, R., Karakoulas, G.J., 2003. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering* 129(3), 278-285.

Abou Elassad, Z.E., Mousannif, H., Al Moatassime, H., 2020a. A proactive decision support system for predicting traffic crash events: A critical analysis of imbalanced class distribution. *Knowledge-Based Systems* 205, 106314-106314.

Abou Elassad, Z.E., Mousannif, H., Al Moatassime, H., 2020b. A real-time crash prediction fusion framework: An imbalance-aware strategy for collision avoidance systems. *Transportation research part C: emerging technologies* 118, 102708-102708.

Abu-Lebdeh, G., Benekohal, R.F., 1997. Development of traffic control and queue management procedures for oversaturated arterials. *Transportation Research Record: Journal of the Transportation Research Board* 1603(1), 119-127.

Ahmane, M., Abbas-Turki, A., Perronnet, F., Wu, J., El Moudni, A., Buisson, J., Zeo, R., 2013. Modeling and controlling an isolated urban intersection based on cooperative vehicles. *Transportation Research Part C: Emerging Technologies* 28, 44-62.

Ahmed, M.M., Abdel-Aty, M.A., 2011. The viability of using automatic vehicle identification data for real-time crash prediction. *IEEE Transactions on Intelligent Transportation Systems* 13(2), 459-468.

Alzantot, M., Chakraborty, S., Srivastava, M., 2017. Sensegen: A deep learning architecture for synthetic sensor data generation, pp. 188-193.

Ammoun, S., Nashashibi, F., 2009. Real time trajectory prediction for collision risk estimation between vehicles. *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*, 417-422.

An, J., Cho, S., 2015. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE* 2(1), 1-18.

Araghi, S., Khosravi, A., Creighton, D., 2015. A review on computational intelligence methods for controlling traffic signal timing. *Expert Systems with Applications* 42(3), 1538-1550.

Asadi, B., Vahidi, A., 2010. Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time. *IEEE transactions on control systems technology* 19(3), 707-714.

Au, T.-C., Zhang, S., Stone, P., 2015. Autonomous intersection management for semi-autonomous vehicles. *Handbook of transportation*, 88-104.

Ban, X., Herring, R., Hao, P., Bayen, A.M.J.T.R.R., 2009. Delay pattern estimation for signalized intersections using sampled travel times. 2130(1), 109-119.

Bao, J., Liu, P., Ukkusuri, S.V., 2019. A spatiotemporal deep learning approach for citywide short-term crash risk prediction with multi-source data. *Accident Analysis & Prevention* 122, 239-254.

Basso, F., Basso, L.J., Bravo, F., Pezoa, R., 2018. Real-time crash prediction in an urban expressway using disaggregated data. *Transportation research part C: emerging technologies* 86, 202-219.

Basso, F., Basso, L.J., Pezoa, R., 2020. The importance of flow composition in real-time crash prediction. *Accident Analysis & Prevention* 137, 105436-105436.

Beak, B., Head, K.L., Feng, Y., 2017. Adaptive coordination based on connected vehicle technology. *Transportation Research Record: Journal of the Transportation Research Board* 2619(1), 1-12.

Beard, C., Ziliaskopoulos, A., 2006. System optimal signal optimization formulation. *Transportation Research Record: Journal of the Transportation Research Board* 1978(1), 102-112.

Bhattacharya, S., Lane, N.D., 2016. From smart to deep: Robust activity recognition on smartwatches using deep learning, pp. 1-6.

Bingham, E., 2001. Reinforcement learning in neurofuzzy traffic signal control. *European Journal of Operational Research* 131(2), 232-241.

Bleeker, S., Moll, H., Steyerberg, E., Donders, A., Derksen-Lubsen, G., Grobbee, D., Moons, K., 2003. External validation is necessary in prediction research:: A clinical example. *Journal of clinical epidemiology* 56(9), 826-832.

Blum, J.R., Greencorn, D.G., Cooperstock, J.R., 2012. Smartphone sensor reliability for augmented reality applications. *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, 127-138.

Bodor, R., Jackson, B., Papanikolopoulos, N., 2003. Vision-based human tracking and activity recognition.

Brand, M., Oliver, N., Pentland, A., 1997. Coupled hidden Markov models for complex action recognition. *CVPR* 97, 994-994.

Bulling, A., Blanke, U., Schiele, B., 2014. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys* 46(3), 1-33.

Cai, C., Wang, Y., Geers, G., 2012. Adaptive traffic signal control using wireless communications.

Cai, Q., Abdel-Aty, M., Yuan, J., Lee, J., Wu, Y., 2020. Real-time crash prediction on expressways using deep generative models. *Transportation Research Part C: Emerging Technologies* 117, 102697-102697.

Cantarella, G.E., Pavone, G., Vitetta, A., 2006. Heuristics for urban road network design: lane layout and signal settings. *European Journal of Operational Research* 175(3), 1682-1695.

CFXWAY, 2021. Central Florida Expressway Authority Webpage. https://www.cfxway.com/for-travelers/expressways/408/.

Chang, T.-H., Sun, G.-Y., 2004. Modeling and optimization of an oversaturated signalized network. *Transportation Research Part B: Methodological* 38(8), 687-707.

Chao, K.-H., Lee, R.-H., Wang, M.-H., 2008. An intelligent traffic light control based on extension neural network, *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, pp. 17-24.

Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S.T., Tröster, G., Millán, J.D.R., Roggen, D., 2013. The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters* 34(15), 2033-2042.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16, 321-357.

Chen, C., Jafari, R., Kehtarnavaz, N., 2015. UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor, pp. 168-172.

Choi, S., Park, B.B., Lee, J., Lee, H., Son, S.H., 2016. Field implementation feasibility study of cumulative travel-time responsive (CTR) traffic signal control algorithm. *Journal of Advanced Transportation* 50(8), 2226-2238.

Choy, M.C., Srinivasan, D., Cheu, R.L., 2003. Cooperative, hybrid agent architecture for real-time traffic signal control. *IEEE Transactions on Systems, Man,*

Cybernetics-Part A 33(5), 597-607.

Day, C.M., Li, H., Richardson, L.M., Howard, J., Platte, T., Sturdevant, J.R., Bullock, D.M., 2017. Detector-free optimization of traffic signal offsets with connected vehicle data. *Transportation Research Record: Journal of the Transportation Research Board* 2620(1), 54-68.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database, pp. 248-255.

Deo, N., Rangesh, A., Trivedi, M.M., 2018. How would surround vehicles move? a unified framework for maneuver classification and motion prediction. *IEEE Transactions on Intelligent Vehicles* 3(2), 129-140.

Dietterich, T.G., others, 2002. Ensemble learning. *The handbook of brain theory and neural networks* 2, 110-125.

Dresner, K., Stone, P., 2004. Multiagent traffic management: A reservation-based intersection control mechanism, *Autonomous Agents and Multiagent Systems, International Joint Conference on*. IEEE Computer Society, pp. 530-537.

Dresner, K., Stone, P., 2006. Human-usable and emergency vehicle-aware control policies for autonomous intersection management, *Fourth International Workshop on Agents in Traffic and Transportation (ATT), Hakodate, Japan*.

Elamrani Abou Elassad, Z., Mousannif, H., Al Moatassime, H., 2020. Class-imbalanced crash prediction based on real-time traffic and weather data: A driving simulator study. *Traffic injury prevention* 21(3), 201-208.

Engelbrecht, J., Booysen, M.J., van Rooyen, G.-J., Bruwer, F.J., 2015. Survey of smartphone-based sensing in vehicles for intelligent transportation system applications. *IET Intelligent Transport Systems* 9(10), 924-935.

Favilla, J., Machion, A., Gomide, F., 1993. Fuzzy traffic control: adaptive strategies, *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*. IEEE, pp. 506-511.

Fayazi, S.A., Vahidi, A., Mahler, G., Winckler, A.J.I.T.o.I.T.S., 2014. Traffic signal phase and timing estimation from low-frequency transit bus data. 16(1), 19-28.

Fayazi, S.A., Vahidi, A.J.I.T.o.I.T.S., 2015. Crowdsourcing phase and timing of pre-timed traffic signals in the presence of queues: algorithms and back-end system architecture. 17(3), 870-881.

Foerster, F., Smeja, M., Fahrenberg, J., 1999. Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring. *Computers in human behavior* 15(5), 571-583.

Fredendall, G.L., 1949. Low-pass filter system. *US Patent 2,472,798*.

Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., Greenspan, H., 2018. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing* 321, 321-331.

Fuqiang, G., Kourosh, K., Shahrokh, V., Jianga, S., Rui, Z., 2018. Locomotion Activity Recognition Using Stacked Denoising Autoencoders. *Ieee Internet of Things Journal* 5(3), 2085-2093.

Gartner, N.H., Pooran, F.J., Andrews, C.M., 2001. Implementation of the OPAC adaptive control strategy in a traffic signal network, *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*. IEEE, pp. 195-200.

Gindele, T., Brechtel, S., Dillmann, R., 2015. Learning driver behavior models from traffic observations for decision making and planning. *IEEE Intelligent Transportation Systems Magazine* 7(1), 69-79.

Gjoreski, H., Ciliberto, M., Wang, L., Morales, F.J.O., Mekki, S., Valentin, S., Roggen, D., 2018. The university of sussex-huawei locomotion and transportation dataset for multimodal analytics with mobile devices. *IEEE Access* 6, 42592-42604.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks, pp. 249-256.

Goodall, N.J., Smith, B.L., Park, B., 2013. Traffic signal control with connected vehicles. *Transportation Research Record: Journal of the Transportation Research Board* 2381(1), 65-72.

Gordon, R.L., Reiss, R.A., Haenel, H., Case, E., French, R.L., Mohaddes, A., Wolcott, R., 1996. Traffic control systems handbook. United States. Federal Highway Administration. Office of Technology Applications.

Gradinescu, V., Gorgorin, C., Diaconescu, R., Cristea, V., Iftode, L., 2007. Adaptive traffic lights using car-to-car communication, *2007 IEEE 65th vehicular technology conference-VTC2007-Spring*. IEEE, pp. 21-25.

Gu, F., Khoshelham, K., Valaee, S., 2017. Locomotion activity recognition: A deep learning approach, pp. 1-5.

Guyon, I., 1997. A scaling law for the validation-set training-set size ratio. *AT&T Bell Laboratories* 1(11).

Hajbabaie, A., Benekohal, R.F., 2011. Does traffic metering improve network performance efficiency?, *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 1114-1119.

Hassan, M.M., Uddin, M.Z., Mohamed, A., Almogren, A., 2018. A robust human activity recognition system using smartphone sensors and deep learning. *Future Generation Computer Systems* 81, 307-313.

Hausknecht, M., Au, T.-C., Stone, P., 2011. Autonomous intersection management: Multi-intersection optimization, *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4581-4586.

He, H., Bai, Y., Garcia, E.A., Li, S., 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning, pp. 1322-1328.

He, Q., Head, K.L., Ding, J., 2012. PAMSCOD: Platoon-based arterial multi-modal signal control with online data. *Transportation Research Part C: Emerging Technologies* 20(1), 164-184.

He, Y., Wu, D., Beyazit, E., Sun, X., Wu, X., 2018. Supervised Data Synthesizing and Evolving--A Framework for Real-World Traffic Crash Severity Classification, pp. 163-170.

Head, K.L., Mirchandani, P.B., Sheppard, D., 1992. *Hierarchical framework for real-time traffic control*.

Hernández Sánchez, S., Fernández Pozo, R., Hernández Gómez, L., 2018. Estimating Vehicle Movement Direction from Smartphone Accelerometers Using Deep Neural Networks. *Sensors* 18(8), 2624-2624.

Houenou, A., Bonnifait, P., Cherfaoui, V., Yao, W., 2013. Vehicle trajectory prediction based on motion model and maneuver recognition, pp. 4363-4369.

Hu, J., Park, B.B., Lee, Y.-J., 2015. Coordinated transit signal priority supporting transit progression under connected vehicle technology. *Transportation Research Part C: Emerging Technologies* 55, 393-408.

Hu, Y., Thomas, P., Stonier, R.J., 2007. Traffic signal control using fuzzy logic and evolutionary algorithms, *2007 IEEE congress on evolutionary computation*. IEEE, pp. 1785-1792.

Huang, T., Wang, S., Sharma, A., 2020. Highway crash detection and risk estimation using deep learning. *Accident Analysis & Prevention* 135, 105392-105392.

Islam, Z., Abdel-Aty, M., 2021a. Real-Time Vehicle Trajectory Estimation Based on Lane Change Detection using Smartphone Sensors. *Transportation Research Record*, 0361198121990681.

Islam, Z., Abdel-Aty, M., 2021b. Sensor-Based Transportation Mode Recognition Using Variational Autoencoder. *Journal of Big Data Analytics in Transportation* 3(1), 15-26.

Islam, Z., Abdel-Aty, M., Cai, Q., Yuan, J., 2020. Crash data augmentation using variational autoencoder. *Accident Analysis & Prevention* 151, 105950-105950.

Iwashita, S., Shimanuki, Y., 2018. Estimation of Human Movement State by Smartphone Sensor Information without GPS. *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, 554-558.

Janssen, V., 2009. Understanding coordinate reference systems, datums and transformations. *International Journal of Geoinformatics* 5(4), 41-53.

Kaempchen, N., Weiss, K., Schaefer, M., Dietmayer, K.C.J., 2004. IMM object tracking for high dynamic driving maneuvers. *IEEE Intelligent Vehicles Symposium, 2004*, 825-830.

Kakihara, M., 2014. Grasping a Global View of Smartphone Diffusion: An Analysis from a Global Smartphone Study, in: 2014 (Ed.), pp. 11-11.

Kamal, M.A.S., Imura, J.-i., Hayakawa, T., Ohata, A., Aihara, K., 2014. A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights. *IEEE Transactions on Intelligent Transportation Systems* 16(3), 1136-1147.

Kanarachos, S., Christopoulos, S.-R.G., Chroneos, A., 2018. Smartphones as an integrated platform for monitoring driver behaviour: The role of sensor fusion and connectivity. *Transportation research part C: emerging technologies* 95, 867-882.

Ke, J., Zhang, S., Yang, H., Chen, X., 2019. PCA-based missing information imputation for real-time crash likelihood prediction under imbalanced data. *Transportmetrica A: transport science* 15(2), 872-895.

Kessler, F.C., Battersby, S.E., Finn, M.P., Clarke, K.C., 2017. Map Projections and the Internet. Springer, pp. 117-148.

Kim, B., Kang, C.M., Kim, J., Lee, S.H., Chung, C.C., Choi, J.W., 2017. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 399-404.

Kingma, D.P., Welling, M., 2014. Auto-encoding variational bayes, Ml ed, pp. 1-14.

Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection, 2 ed, pp. 1137-1145.

Koukoumidis, E., Peh, L.-S., Martonosi, M.R., 2011. Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory, *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pp. 127-140.

Kusner, M.J., Paige, B., Hernández-Lobato, J.M., 2017. Grammar variational autoencoder, pp. 1945-1954.

Lara, O.D., Labrador, M.A., 2012. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials* 15(3), 1192-1209.

Larsen, J., Hansen, L.K., Svarer, C., Ohlsson, M., 1996. Design and regularization of neural networks: the optimal use of a validation set, *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*. IEEE, pp. 62-71.

Lau, S.L., König, I., David, K., Parandian, B., Carius-Düssel, C., Schultz, M., 2018. Supporting patient monitoring using activity recognition with a smartphone, pp. 810-814.

Laugier, C., Paromtchik, I.E., Perrollaz, M., Yong, M., Yoder, J.-D., Tay, C., Mekhnacha, K., Nègre, A., 2011. Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety. *IEEE Intelligent Transportation Systems Magazine* 3(4), 4-19.

Lee, J., Park, B., 2012. Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *IEEE Transactions on Intelligent Transportation Systems* 13(1), 81-90.

Lee, J., Park, B., Yun, I., 2013. Cumulative travel-time responsive real-time intersection control algorithm in the connected vehicle environment. *Journal of Transportation Engineering* 139(10), 1020-1029.

Lee, J.-H., Lee, K.-M., Lee-Kwang, H., 1995. Fuzzy controller for intersection group, *Proceedings IEEE Conference on Industrial Automation and Control Emerging Technology Applications*. IEEE, pp. 376-382.

Lefèvre, S., Vasquez, D., Laugier, C., 2014. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal* 1(1), 1-1.

Li, L., Lv, Y., Wang, F.-Y.J.I.C.J.o.A.S., 2016. Traffic signal timing via deep reinforcement learning. 3(3), 247-254.

Li, P., Abdel-Aty, M., Cai, Q., Yuan, C., 2020a. ⋆ This paper has been handled by associate editor Tony Sze. The application of novel connected vehicles emulated data on real-time crash potential prediction for arterials. *Accident Analysis & Prevention* 144, 105658-105658.

Li, P., Abdel-Aty, M., Yuan, J., 2020b. Real-time crash risk prediction on arterials based on LSTM-CNN. *Accident Analysis & Prevention* 135, 105371-105371.

Li, Z., Elefteriadou, L., Ranka, S., 2014. Signal control optimization for automated vehicles at isolated signalized intersections. *Transportation Research Part C: Emerging Technologies* 49, 1-18.

Liang, X., Guler, S.I., Gayah, V.V., 2018. Signal Timing Optimization with Connected Vehicle Technology: Platooning to Improve Computational Efficiency. *Transportation Research Record: Journal of the Transportation Research Board* 2672(18), 81-92.

Lin, F., Song, C., Xu, X., Cavuoto, L., Xu, W., 2016. Sensing from the bottom: Smart insole enabled patient handling activity recognition through manifold learning, pp. 254-263.

Lin, L., Gong, S., Li, T., Peeta, S., 2018. Deep learning-based human-driven vehicle trajectory prediction and its application for platoon control of connected and autonomous vehicles.

Lin, L., Wang, Q., Sadek, A.W., 2015. A novel variable selection method based on frequent pattern tree for real-time traffic accident risk prediction. *Transportation Research Part C: Emerging Technologies* 55, 444-459.

Liu, C., Zhang, L., Liu, Z., Liu, K., Li, X., Liu, Y., 2016. Lasagna: towards deep hierarchical understanding and searching over mobile sensing data, pp. 334-347.

Liu, H., Han, K., Gayah, V., Friesz, T., Yao, T., 2015. Data-driven linear decision rule approach for distributionally robust optimization of on-line signal control. *Transportation Research Procedia* 7, 536-555.

Longley, D., 1968. A control strategy for a congested, computer-controlled traffic network. *Transportation Research*.

Madabhushi, A., Aggarwal, J.K., 1999. A bayesian approach to human activity recognition, pp. 25-32.

Mahler, G., Vahidi, A., 2014. An optimal velocity-planning scheme for vehicle energy efficiency through probabilistic prediction of traffic-signal timing. *IEEE Transactions on Intelligent Transportation Systems* 15(6), 2516-2523.

Mauro, V.a.C.D.T., 1990. "UTOPIA," Control, Computers, Communications in Transportation: Selected Papers from the IFAC Symposium.

Murat, Y.S., Gedizlioglu, E., 2005. A fuzzy logic multi-phased signal control model for isolated junctions. *Transportation Research Part C: Emerging Technologies* 13(1), 19-36.

Nagare, A., Bhatia, S., 2012. Traffic flow control using neural network. *Traffic* 1(2), 50-52.

Nakatsuyama, M., Nagahashi, H., Nishizuka, N., 1984. Fuzzy logic phase controller for traffic junctions in the one-way arterial road. *IFAC Proceedings Volumes* 17(2), 2865-2870.

Nilsson, J., Brännström, M., Coelingh, E., Fredriksson, J., 2016. Lane change maneuvers for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems* 18(5), 1087-1096.

NSC-and-NHTSA-Report, 2019. Comparison of NSC and NHTSA Estimates. https://injuryfacts.nsc.org/motor-vehicle/overview/comparison-of-nsc-and-nhtsa-estimates/.

Ordóñez, F.J., Roggen, D., 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16(1), 115-115.

Pandit, K., Ghosal, D., Zhang, H.M., Chuah, C.-N., 2013. Adaptive traffic signal control with vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology* 62(4), 1459-1471.

Pappis, C.P., Mamdani, E.H., 1977. A fuzzy logic controller for a trafc junction. *IEEE Transactions on Systems, Man,*

Cybernetics 7(10), 707-717.

Parsa, A.B., Taghipour, H., Derrible, S., Mohammadian, A.K., 2019. Real-time accident detection: coping with imbalanced data. *Accident Analysis & Prevention* 129, 202-210.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., others, 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12, 2825-2830.

Peng, Y., Li, C., Wang, K., Gao, Z., Yu, R., 2020. Examining imbalanced classification algorithms in predicting real-time traffic crash risk. *Accident Analysis & Prevention* 144, 105610-105610.

Perez, L., Wang, J., 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Porche, I., Lafortune, S., 1999. Adaptive look-ahead optimization of traffic signals. *Journal of Intelligent Transportation System* 4(3-4), 209-254.

Prashanth, L., Bhatnagar, S., 2010. Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems* 12(2), 412-421.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 2007. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

Priemer, C., Friedrich, B., 2009. A decentralized adaptive traffic signal control using V2I communication data, *2009 12th International IEEE Conference on Intelligent Transportation Systems*. IEEE, pp. 1-6.

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., Carin, L., 2016. Variational autoencoder for deep learning of images, labels and captions, pp. 2352-2360.

Putha, R., Quadrifoglio, L., Zechman, E., 2012. Comparing ant colony optimization and genetic algorithm approaches for solving traffic signal coordination under oversaturation conditions. *Computer-Aided Civil*

Infrastructure Engineering 27(1), 14-28.

Rahman, M.H., Abdel-Aty, M., Lee, J., Rahman, M.S., 2019. Enhancing traffic safety at school zones by operation and engineering countermeasures: A microscopic simulation approach. *Simulation Modelling Practice and Theory* 94, 334-348.

Rose, C., Bevly, D., 2009. Camera vision and inertial measurement unit sensor fusion for lane detection and tracking using polynomial bounding curves. *Proc. ION GNSS*, 1849-1857.

Shi, Q., 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.

Sims, A.G., Dobinson, K.W., 1980. The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits. *IEEE Transactions on vehicular technology* 29(2), 130-137.

Slabaugh, G.G., 1999. Computing Euler angles from a rotation matrix. *Retrieved on August* 6(2000), 39-63.

Sønderby, C.K., Raiko, T., Maaløe, L., Sønderby, S.K., Winther, O., 2016. Ladder variational autoencoders, pp. 3738-3746.

Song, B., Kamal, A.T., Soto, C., Ding, C., Farrell, J.A., Roy-Chowdhury, A.K., 2010. Tracking and activity recognition through consensus in distributed camera networks. *IEEE Transactions on Image Processing* 19(10), 2564-2579.

Spall, J.C., Chin, D.C., 1997. Traffic-responsive signal timing for system-wide traffic control. *Transportation Research Part C: Emerging Technologies* 5(3-4), 153-163.

Stebbins, S., Kim, J., Hickman, M., Vu, H.L., 2016. Combining model predictive intersection control with green light optical speed advisory in a connected vehicle environment, *Australasian Transport Research Forum 2016*.

Steven Eyobu, O., Han, D.S., 2018. Feature representation and data augmentation for human activity classification based on wearable IMU sensor data using a deep LSTM neural network. *Sensors* 18(9), 2892-2892.

Sun, D., Benekohal, R.F., Waller, S.T., 2006. Bi-level programming formulation and heuristic solution approach for dynamic traffic signal optimization. *Computer-Aided Civil*

Infrastructure Engineering 21(5), 321-333.

Sun, S., Zhou, B., Zhang, S., 2020. Analysis of Factors Affecting Injury Severity in Motorcycle Involved Crashes, pp. 4207-4219.

Sutton, R.S., 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 1038-1044.

Takahashi, A., Ninomiya, Y., 1996. Model-based lane recognition. *Proceedings of Conference on Intelligent Vehicles*, 201-206.

Teodorović, D., Varadarajan, V., Popović, J., Chinnaswamy, M.R., Ramaraj, S., 2006. Dynamic programming—neural network real-time traffic adaptive signal control algorithm. *Annals of Operations Research* 143(1), 123-131.

Thorpe, T.L., Anderson, C.W., 1996. Tra c light control using sarsa with three state representations. Citeseer.

Tran, Q., Firl, J., 2014. Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression. *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 918-923.

Ukkusuri, S., Doan, K., Aziz, H.A., 2013. A bi-level formulation for the combined dynamic equilibrium based traffic signal control. *Procedia-Social*

Behavioral Sciences 80, 729-752.

Um, T.T., Pfister, F.M.J., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., Kulić, D., 2017. Data augmentation of wearable sensor data for parkinson’s disease monitoring using convolutional neural networks, pp. 216-220.

Vavoulas, G., Chatzaki, C., Malliotakis, T., Pediaditis, M., Tsiknakis, M., 2016. The MobiAct Dataset: Recognition of Activities of Daily Living using Smartphones, pp. 143-151.

Wang, C., Jiang, S., 2012. Traffic signal phases' estimation by floating car data, *2012 12th International Conference on ITS Telecommunications*. IEEE, pp. 568-573.

Wang, J., Chen, Y., Gu, Y., Xiao, Y., Pan, H., 2018. SensoryGANs: An Effective Generative Adversarial Framework for Sensor-based Human Activity Recognition. IEEE, pp. 1-8.

Wang, J., Chen, Y., Hao, S., Peng, X., Hu, L., 2019a. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters* 119, 3-11.

Wang, J., Kong, Y., Fu, T., 2019b. Expressway crash risk prediction using back propagation neural network: A brief investigation on safety resilience. *Accident Analysis & Prevention* 124, 180-192.

Wang, L., Abdel-Aty, M., Lee, J., Shi, Q., 2019c. Analysis of real-time crash risk for expressway ramps using traffic, geometric, trip generation, and socio-demographic predictors. *Accident Analysis & Prevention* 122, 378-384.

Wang, L., Abdel-Aty, M., Shi, Q., Park, J., 2015. Real-time crash prediction for expressway weaving segments. *Transportation Research Part C: Emerging Technologies* 61, 1-10.

Wang, L., Gjoreski, H., Ciliberto, M., Mekki, S., Valentin, S., Roggen, D., 2019d. Enabling reproducible research in sensor-based transportation mode recognition with the Sussex-Huawei dataset. *IEEE Access* 7, 10870-10891.

Wei, W., Zhang, Y., Mbede, J.B., Zhang, Z., Song, J., 2001. Traffic signal control using fuzzy logic and MOGA, *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236)*. IEEE, pp. 1335-1340.

Welch, P., 1967. The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. *IEEE Transactions on audio and electroacoustics* 15(2), 70-73.

Wiering, M., Vreeken, J., Van Veenen, J., Koopman, A., 2004. Simulation and optimization of traffic in a city, *IEEE Intelligent Vehicles Symposium, 2004*. IEEE, pp. 453-458.

Wiering, M.A., 2000. Multi-agent reinforcement learning for traffic light control, *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pp. 1151-1158.

Wiest, J., Höffken, M., Kreßel, U., Dietmayer, K., 2012. Probabilistic trajectory prediction with Gaussian mixture models. *2012 IEEE Intelligent Vehicles Symposium*, 141-146.

Woo, H., Ji, Y., Kono, H., Tamura, Y., Kuroda, Y., Sugano, T., Yamamoto, Y., Yamashita, A., Asama, H., 2017. Lane-change detection based on vehicle-trajectory prediction. *IEEE Robotics and Automation Letters* 2(2), 1109-1116.

Wu, J., Abbas-Turki, A., Correia, A., El Moudni, A., 2007. Discrete intersection signal control, *2007 IEEE international conference on service operations and logistics, and informatics*. IEEE, pp. 1-6.

Wunderlich, R., Liu, C., Elhanany, I., UrbanikII, T., 2008. A novel signal-scheduling algorithm with quality-of-service provisioning for an isolated intersection. *IEEE Transactions on Intelligent Transportation Systems* 9(3), 536-547.

Xia, H., Qiao, Y., Jian, J., Chang, Y., 2014. Using smart phone sensors to detect transportation modes. *Sensors* 14(11), 20843-20865.

Xie, X.-F., Smith, S.F., Lu, L., Barlow, G.J., 2012. Schedule-driven intersection control. *Transportation Research Part C: Emerging Technologies* 24, 168-189.

Xu, C., Liu, P., Wang, W., Li, Z., 2012. Evaluation of the impacts of traffic states on crash risks on freeways. *Accident Analysis & Prevention* 47, 162-171.

Xu, C., Tarko, A.P., Wang, W., Liu, P., 2013. Predicting crash likelihood and severity on freeways with real-time loop detector data. *Accident Analysis & Prevention* 57, 30-39.

Xu, W., Sun, H., Deng, C., Tan, Y., 2017. Variational autoencoder for semi-supervised text classification.

Yahaya, M., Jiang, X., Fu, C., Bashir, K., Fan, W., 2019. Enhancing Crash Injury Severity Prediction on Imbalanced Crash Data by Sampling Technique with Variable Selection, pp. 363-368.

Yan, F., Dridi, M., El Moudni, A., 2009. Autonomous vehicle sequencing algorithm at isolated intersections, *2009 12th International IEEE conference on intelligent transportation systems*. IEEE, pp. 1-6.

Yang, I., Jayakrishnan, R., 2015. Real-time network-wide traffic signal optimization considering long-term green ratios based on expected route flows. *Transportation Research Part C: Emerging Technologies* 60, 241-257.

Yang, K., Menendez, M., Guler, S., 2018. Multimodal Traffic Control using Connected Vehicle Technology. *Transportmetrica Part B.*

Yin, H., Wong, S.C., Xu, J., Wong, C., 2002. Urban traffic flow prediction using a fuzzy-neural approach. *Transportation Research Part C: Emerging Technologies* 10(2), 85-98.

Yin, J., Yang, Q., Member, S., Pan, J.J., 2008. Sensor-based Abnormal Human-Activity Detection. 1-24.

Yin, Y., Huang, Y., Zhang, L., Gao, Z., 2019. Influence of Different Sampling Techniques on The Real-time Crash Risk Prediction Model, pp. 1795-1799.

You, J., Wang, J., Fang, S., Guo, J., 2017. An optimized real-time crash prediction model on freeway with over-sampling techniques based on support vector machine. *Journal of Intelligent & Fuzzy Systems* 33(1), 555-562.

Yu, R., Abdel-Aty, M., 2013. Utilizing support vector machine in real-time crash risk evaluation. *Accident Analysis & Prevention* 51, 252-259.

Yu, R., Quddus, M., Wang, X., Yang, K., 2018. Impact of data aggregation approaches on the relationships between operating speed and traffic safety. *Accident Analysis & Prevention* 120, 304-310.

Yu, R., Wang, X., Yang, K., Abdel-Aty, M., 2016. Crash risk analysis for Shanghai urban expressways: a Bayesian semi-parametric modeling approach. *Accident Analysis & Prevention* 95, 495-502.

Yuan, J., Abdel-Aty, M., Gong, Y., Cai, Q., 2019. Real-time crash risk prediction using long short-term memory recurrent neural network. *Transportation research record* 2673(4), 314-326.

Zeng, R., Li, G., Lin, L., 2007. Adaptive traffic signals control by using fuzzy logic, *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*. IEEE, pp. 527-527.

Zhang, M., Sawchuk, A.A., 2011. A feature selection-based framework for human activity recognition using wearable multimodal sensors, pp. 92-98.

Zhang, S., Abdel-Aty, M., Cai, Q., Li, P., Ugan, J., 2020a. Prediction of pedestrian-vehicle conflicts at signalized intersections based on long short-term memory neural network. *Accident Analysis & Prevention* 148, 105799.

Zhang, S., Abdel-Aty, M., Yuan, J., Li, P., 2020b. Prediction of pedestrian crossing intentions at intersections based on long short-term memory recurrent neural network. *Transportation research record* 2674(4), 57-65.

Zhao, Y., Tian, Z., 2012. An overview of the usage of adaptive signal control system in the United States of America, *Applied Mechanics and Materials*. Trans Tech Publ, pp. 2591-2598.

Zheng, X., Recker, W., 2013. An adaptive control algorithm for traffic-actuated signals. *Transportation Research Part C: Emerging Technologies* 30, 93-115.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., Zhao, J.L., 2016. Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. *Frontiers of Computer Science* 10(1), 96-112.

Zheng, Z., Ahn, S., Monsere, C.M., 2010. Impact of traffic oscillations on freeway crash occurrences. *Accident Analysis & Prevention* 42(2), 626-636.

Zhou, B., Zhang, X., Zhang, S., Li, Z., Liu, X., 2019. Analysis of Factors Affecting Real-Time Ridesharing Vehicle Crash Severity. *Sustainability* 11(12), 3334-3334.

Zhou, Z., Zhu, R., Fu, X., Zhang, G., 2012. Microelectromechanical Sensor-Based System. Springer, pp. 619-651.

Zohdy, I.H., Rakha, H., 2012. Game theory algorithm for intersection-based cooperative adaptive cruise control (CACC) systems, *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, pp. 1097-1102.