

Journal of Advanced Transportation

Machine Learning in Transportation

Lead Guest Editor: Ali Tizghadam

Guest Editors: Hamzeh Khazaei, Mohammad H. Y. Moghaddam,
and Yasser Hassan






Machine Learning in Transportation

Journal of Advanced Transportation

Machine Learning in Transportation

Lead Guest Editor: Ali Tizghadam

Guest Editors: Hamzeh Khazaei, Mohammad H. Y. Moghaddam,
and Yasser Hassan



Copyright © 2019 Hindawi. All rights reserved.

This is a special issue published in "Journal of Advanced Transportation." All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Editorial Board

Francesco Bella, Italy
Abdelaziz Bensrhair, France
Cesar Briso-Rodriguez, Spain
María Calderon, Spain
Juan C. Cano, Spain
Giulio E. Cantarella, Italy
Maria Castro, Spain
Oded Cats, Netherlands
Anthony Chen, USA
Nicolas Chiabaut, France
Steven I. Chien, USA
Antonio Comi, Italy
Luca D'Acierno, Italy
Andrea D'Ariano, Italy
Alexandre De Barros, Canada
Stefano de Luca, Italy
Rocío de Oña, Spain
Luigi Dell'Olio, Spain
Cédric Demonceaux, France
Sunder Lall Dhingra, India
Vinayak Dixit, Australia
Yuchuan Du, China

Nour-Eddin El-faouzi, France
Juan-Antonio Escareno, France
David F. Llorca, Spain
Peter Furth, USA
Francesco Galante, Italy
Md. Mazharul Haque, Australia
Jérôme Haïri, France
Samiul Hasan, USA
S. P. Hoogendoorn, Netherlands
Hocine Imine, France
Lina Kattan, Canada
Victor L. Knoop, Netherlands
Alain Lambert, France
Ludovic Leclercq, France
Jaeyoung Lee, USA
Seungjae Lee, Republic of Korea
Zhi-Chun Li, China
Yue Liu, USA
Jose R. Martinez-De-Dios, Spain
Filomena Mauriello, Italy
Monica Menendez, UAE
Rakesh Mishra, UK




Andrea Monteriù, Italy
Giuseppe Musolino, Italy
Jose E. Naranjo, Spain
Aboelmagd Noureldin, Canada
Eneko Osaba, Spain
Eleonora Papadimitriou, Netherlands
Dongjoo Park, Republic of Korea
Paola Pellegrini, France
Luca Pugi, Italy
Nandana Rajatheva, Finland
Hesham Rakha, USA
Prianka N. Seneviratne, Philippines
Fulvio Simonelli, Italy
Richard S. Tay, Australia
Pascal Vasseur, France
Antonino Vitetta, Italy
Francesco Viti, Luxembourg
S. Travis Waller, Australia
Shamsunnahar Yasmin, Australia
Jacek Zak, Poland
Guohui Zhang, USA
G. Homem de Almeida Correia, Netherlands

Contents


Machine Learning in Transportation

Ali Tizghadam , Hamzeh Khazaei, Mohammad H. Y. Moghaddam, and Yasser Hassan
Editorial (3 pages), Article ID 4359785, Volume 2019 (2019)



Extracting Vehicle Trajectories Using Unmanned Aerial Vehicles in Congested Traffic Conditions

Eui-Jin Kim , Ho-Chul Park , Seung-Woo Ham, Seung-Young Kho, and Dong-Kyu Kim 
Research Article (16 pages), Article ID 9060797, Volume 2019 (2019)

EBOC: Ensemble-Based Ordinal Classification in Transportation

Pelin Yıldırım, Ulaş K. Birant, and Derya Birant 
Research Article (17 pages), Article ID 7482138, Volume 2019 (2019)



Spatiotemporal Traffic Flow Prediction with KNN and LSTM

Xianglong Luo , Danyang Li, Yu Yang, and Shengrui Zhang 
Research Article (10 pages), Article ID 4145353, Volume 2019 (2019)

Driver Distraction Identification with an Ensemble of Convolutional Neural Networks

Hesham M. Eraqi , Yehya Abouelnaga , Mohamed H. Saad , and Mohamed N. Moustafa 
Research Article (12 pages), Article ID 4125865, Volume 2019 (2019)


Public Transport Driver Identification System Using Histogram of Acceleration Data

Nuttun Virojboonkiate , Adsadawut Chanakitkarnchok, Peerapon Vateekul, and Kultida Rojviboonchai 
Research Article (15 pages), Article ID 6372597, Volume 2019 (2019)

Rapid Driving Style Recognition in Car-Following Using Machine Learning and Vehicle Trajectory Data

Qingwen Xue , Ke Wang , Jian John Lu , and Yujie Liu
Research Article (11 pages), Article ID 9085238, Volume 2019 (2019)

Optimizing Location of Car-Sharing Stations Based on Potential Travel Demand and Present Operation Characteristics: The Case of Chengdu

Yu Cheng , Xu Chen, Xiaohua Ding, and Linting Zeng
Research Article (13 pages), Article ID 7546303, Volume 2019 (2019)

A Hybrid Method for Traffic Incident Duration Prediction Using BOA-Optimized Random Forest Combined with Neighborhood Components Analysis

Qiang Shang , Derong Tan, Song Gao, and Linlin Feng
Research Article (11 pages), Article ID 4202735, Volume 2019 (2019)

A Machine Learning Method for Predicting Driving Range of Battery Electric Vehicles





Shuai Sun , Jun Zhang, Jun Bi , and Yongxing Wang 
Research Article (14 pages), Article ID 4109148, Volume 2019 (2019)

Developing a Travel Time Estimation Method of Freeway Based on Floating Car Using Random Forests

Juan Cheng , Gen Li , and Xianhua Chen 


Research Article (13 pages), Article ID 8582761, Volume 2019 (2019)

Evaluation and Application of Urban Traffic Signal Optimizing Control Strategy Based on Reinforcement Learning

Yizhe Wang , Xiaoguang Yang , Yangdong Liu , and Hailun Liang 


Research Article (9 pages), Article ID 3631489, Volume 2018 (2019)

An Improved Deep Learning Model for Traffic Crash Prediction

Chunjiao Dong , Chunfu Shao, Juan Li, and Zhihua Xiong 

Research Article (13 pages), Article ID 3869106, Volume 2018 (2019)

A SVM Approach of Aircraft Conflict Detection in Free Flight

Xu-rui Jiang, Xiang-xi Wen , Ming-gong Wu, Ze-kun Wang, and Xi Qiu

Research Article (9 pages), Article ID 7964641, Volume 2018 (2019)

Identifying Public Transit Commuters Based on Both the Smartcard Data and Survey Data: A Case Study in Xiamen, China

Shichao Sun  and Dongyuan Yang 

Research Article (10 pages), Article ID 9693272, Volume 2018 (2019)

Editorial

Machine Learning in Transportation

Ali Tizghadam ¹, **Hamzeh Khazaei**², **Mohammad H. Y. Moghaddam**³ and **Yasser Hassan**⁴

¹TELUS & University of Toronto, Toronto, Canada

²University of Alberta, Edmonton, Canada

³Ferdowsi University of Mashhad, Mashhad, Iran

⁴Carlton University, Ottawa, Canada

Correspondence should be addressed to Ali Tizghadam; ali.tizghadam@telus.com

Received 21 May 2019; Accepted 21 May 2019; Published 4 June 2019

Copyright © 2019 Ali Tizghadam et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Nation's economy, safety, and quality of life are influenced by a well-behaved transportation system. Yet, demands in transportation are ever increasing due to trends in population growth, emerging technologies, and the increased globalization of the economy which has kept pushing the system to its limits. The rate of increasing the number of vehicles is at points even more than the overall population increase rate, which leads to more congested and dangerous roadways. This problem is not going to be addressed by just adding to the number of roads anymore. The construction cost is very high and the time to return the result is too lengthy to catch up with the vehicle increase rate.

One way to improve upon the fleet management is by viewing the road as an information highway as opposed to highway for vehicles. The scale of ingested data in the transportation system and even the interaction of various components of the system that generates the data have become a bottleneck for the traditional data analytics solutions. On the other hand, machine learning is a form of Artificial Intelligence (AI) and a data-driven solution that can cope with the new system requirements. Machine learning learns the latent patterns of historical data to model the behavior of a system and to respond accordingly in order to automate the analytical model building.

The availability of increased computational power and collection of the massive amount of data have redefined the value of the machine learning-based approaches for addressing the emerging demands and needs in transportation systems.

Machine learning solutions have already begun their promising marks in the transportation industry, where it is proved to even have a higher return on investment compared to the conventional solutions. However, the transportation problems are still rich in applying and leveraging machine learning techniques and need more consideration. The underlying goals for these solutions are to reduce congestion, improve safety and diminish human errors, mitigate unfavorable environmental impacts, optimize energy performance, and improve the productivity and efficiency of surface transportation.

In this special issue, we present original research work aimed at reporting on new models, algorithms, and case studies related to the use of machine learning in the field of transportation and further analysis of the reliability and robustness of the whole transportation system. In particular, the special issue focuses on prediction methods in transportation, transport network traffic flows and signals, public transportation including air fleet, driving styles, electric cars, and car sharing.

In recent years, machine learning techniques have become an integral part of realizing smart transportation. In this context, using an improved deep learning model, the complex interactions among roadways, transportation traffic, environmental elements, and traffic crashes have been explored. The proposed model includes two modules, an unsupervised feature learning module to identify functional relationship between the explanatory variables and the feature representations and a supervised fine-tuning module to

perform traffic crash prediction. Another study leverages a hierarchical traffic signal control algorithm based on reinforcement learning, to solve the problem of “green wave bottleneck point” of the arterial and signal interference.

No effective travel guidance can be provided without a good travel time estimation strategy. Travel time can be achieved directly or indirectly. Direct methods measure travel time using probe vehicle, records at toll stations, tracking of cell phones, and many other technologies. Indirect methods infer travel time using measured traffic volume, speed, and occupancy in point sensors (e.g., loop detector, video camera) along the vehicle trajectory. In recent years, data mining and machine learning have gradually come into sight. The development of traffic information acquisition technologies (such as data of GPS trajectories) has provided us with a large amount of traffic data, which in turn paves the road to develop a more accurate travel time estimation based on data mining. Compared with traditional parametric models, data mining algorithms can explore implicit relationships between variables. To this end, in the special issue, a new data mining technique is introduced to leverage Random Forests for travel time estimation. It is shown that the influence of variables on travel time can be deeply excavated through Random Forests.

In Intelligent Transport Networks (ITS) context, accurate prediction of future traffic conditions is essential to mitigate traffic congestion and to respond to the traffic incidents. In the special issue, a hybrid traffic flow prediction method is proposed based on k-nearest neighbor (KNN) algorithm and long short-term memory network (LSTM). The improved accuracy in traffic flow prediction is illustrated through traffic flow prediction at two stations using detector data on major freeways/expressways in Minnesota.

Along the same lines, Unmanned Aerial Vehicles (UAVs) have great potential in daily traffic management and control as well as collection of traffic condition data at a low cost. However, UAV comes with its own challenges, primarily related to vehicle size in the images and potential occlusion by shadows or other road objects, especially in congested traffic conditions. In this context, two learning-based frameworks are proposed to detect vehicles and track vehicle trajectories in congested traffic conditions.

Statistical machine learning algorithms have also found their way in supporting smart transportation. In one of the research works reported in the special issue, using neighborhood components analysis and the Bayesian optimization algorithm, a random forest model has been trained to estimate the traffic incident duration with high accuracy. In another study in the context of statistical machine learning for transportation, a naive Bayesian classifier is used to identify public transit commuters' travel pattern based on both the smartcard and survey data sampled from commuters.

In a related work on public transit, a system architecture based on acceleration sensor data is proposed to determine the identity of the public transport driver. The proposed system is comprised of three major modules: data collection, data preprocessing and driver identification modules. The data is collected from active campus shuttle buses, the preprocessing module applies a filter to remove periods of inactivity from the data, and to extract the unique behavior of

the driver (which results in identifying the driver), histogram of acceleration sensor data is proposed.

Demand for air travel continues to grow at a rapid pace; consequently, air traffic control is becoming more complex. The main challenge in air travel remains to be the collision avoidance, and given the increase in demand, automated decision support technologies, mainly Conflict Detection Resolution (CDR), will be required to enable continued provisioning of safe and efficient services in increasingly congested skies. In the special case of Free Flight, pilots have the freedom to choose their trajectory and speed to optimize the flight duration and performance while maintaining safe separation from neighboring traffic. To this end, it is important to find a fast conflict detection method in Free Flight, which is applicable in terminals as well as en-route. An attempt to address this problem is presented in which the aircraft conflict detection is considered as a binary classification problem and a pattern recognition approach is used to solve it. The flight data features are extracted and fed into a classifier which has been trained by a large flight dataset. A support vector machine (SVM) method is employed to detect multi-aircraft conflict in Free Flight airspace and to detect the conflict probability. False alarm rate is the most important issue for air traffic controllers, and the proposed method uses Synthetic Minority Oversampling Technique (SMOTE) to handle imbalanced datasets.

Classification techniques have found their way in a more general sense in transportation networks. Learning the latent patterns of historical data in an efficient way to model a transportation system is a major need for making right decisions. However, many classification algorithms in the literature assume that the largest values in the datasets are unordered, so they lose inherent order between the class values. To tackle this problem, a research study is proposed to leverage Ensemble-Based Ordinal Classification (EBOC) approach, which suggests bagging and boosting, AdaBoost algorithm, methods as a solution for ordinal classification problem in transportation. The proposed method is compared with more traditional tree-based classification algorithms such as Decision Tree and Random Tree.

Distracted driving has been widely reported to be a major contributing factor to traffic collisions. With the emergence of new learning-based methods, addressing the driver's distraction problem is becoming a topic of interest among industry and academics. In the special issue, we investigate methods to detect and mitigate the driving style using deep learning and utilizing RGB images obtained from a camera mounted above the dashboard. In the popular case of rear-end collision crash, a study is presented that is focused on developing a driving style recognition method based on vehicle trajectory data extracted from surveillance camera. The study uses reference KPIs such as Inversed Time to Collision (ITTC) and Time-Headway (THW) to evaluate the collision risk level of vehicle trajectory for each driver and we leverage Support Vector Machine (SVM) to recognize driving style. Moreover, the SVM approach is compared with different learning methods including Random Forest (RF), K-Nearest-Neighbors (KNN), and Multilayer Perceptron (MLP) in terms of the precision of driving style recognition.

Car-sharing is becoming an increasingly popular travel mode around the globe. Placement problem is one of the most challenging and interesting issues to be addressed in the context of car sharing. Most of car-sharing station places are chosen either by experience or just randomly on places where the space has been available. This results in having low efficiency and consequently capital loss in many car-sharing stations. A study is included in the special issue aiming at using different data sources with statistical models and a machine learning algorithm to help car-sharing operator choose the optimal location of new stations and adjust the location of existing ones. In the study, the area of interest is assumed to be a grid, and the attention is given to find a model to estimate a potential travel demand value for each small grid and the demand volume is used to suggest best places for car-sharing stations. The study has compared different learning methods and concluded that Logistic Regression with LASSO (Least Absolute Shrinking and Selection Operator) is the best method to estimate the probability of existence of demand in all grids. In addition to demands per grid, the study focuses on a more competitive market and finds the influential factors on order number. Suggestions on the optimal location of stations are given in consideration of competitors.

In summary, we believe that the published contents in this special issue contribute in moving the smart transportation field forward and open new avenues for further research on how machine learning can be leveraged to build more sustainable and safer smart cities.

Conflicts of Interest

The editors declare that there are no conflicts of interest in publishing this editorial.

Ali Tizghadam
Hamzeh Khazaei
Mohammad H. Y. Moghaddam
Yasser Hassan

Research Article

Extracting Vehicle Trajectories Using Unmanned Aerial Vehicles in Congested Traffic Conditions

Eui-Jin Kim ¹, Ho-Chul Park ¹, Seung-Woo Ham,¹
Seung-Young Kho,² and Dong-Kyu Kim ²

¹Department of Civil and Environmental Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Republic of Korea

²Department of Civil and Environmental Engineering and Institute of Construction and Environmental Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Republic of Korea

Correspondence should be addressed to Dong-Kyu Kim; dongkyukim@snu.ac.kr

Received 5 October 2018; Revised 1 January 2019; Accepted 26 February 2019; Published 1 April 2019

Guest Editor: Yasser Hassan

Copyright © 2019 Eui-Jin Kim et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Obtaining the trajectories of all vehicles in congested traffic is essential for analyzing traffic dynamics. To conduct an effective analysis using trajectory data, a framework is needed to efficiently and accurately extract the data. Unfortunately, obtaining accurate trajectories in congested traffic is challenging due to false detections and tracking errors caused by factors in the road environment, such as adjacent vehicles, shadows, road signs, and road facilities. Unmanned aerial vehicles (UAVs), with incorporating machine learning and image processing, can mitigate these difficulties by their ability to hover above the traffic. However, research is lacking regarding the extraction and evaluation of vehicle trajectories in congested traffic. In this study, we propose and compare two learning-based frameworks for detecting vehicles: the aggregated channel feature (ACF), which is based on human-made features, and the faster region-based convolutional neural network (Faster R-CNN), which is based on data-driven features. We extend the detection results to extract vehicle trajectories in congested traffic conditions from UAV images. To remove the errors associated with tracking vehicles, we also develop a postprocessing method based on motion constraints. Then, we conduct detailed performance analyses to confirm the feasibility of the proposed framework on a congested expressway in Korea. The results show that Faster R-CNN outperforms the ACF in images with large objects and in those with small objects if sufficient data are provided. This framework extracts the vehicle trajectories with high precision, making them available for analyzing traffic dynamics based on the training of just a small number of positive samples. The results of this study provide a practical guideline for building a framework to extract vehicles trajectories based on given conditions.

1. Introduction

The trajectories of all the vehicles on the road provide very useful empirical data. To reduce traffic congestion and collisions, these data can be used to analyze various traffic phenomena such as drivers' characteristics, lane-changing behavior, traffic oscillation, capacity drops, and crash potential [1–4]. To conduct a reliable analysis of trajectory data, a framework is needed to efficiently and accurately extract the data. For more than 40 years, researchers have gathered such valuable data by applying the vision-based detection techniques to track vehicles from aerial platforms [5–9] and surveillance cameras mounted at elevated locations [10–12].

Although these studies have improved the overall quality of the data-gathering process, the need for improvements remains. In particular, the accurate tracking of vehicles in congested areas is known to be a challenging problem due to false detections and tracking errors caused by factors in the road environment such as adjacent vehicles, shadows, road signs, and road facilities. Nevertheless, a few studies have implemented vehicle detection and tracking scheme in congested traffic conditions [12, 13] which most traffic-flow studies have implemented [1–3].

To the best of our knowledge, Next Generation Simulation (NGSIM) data [12] are the only published vehicle trajectory data in congested traffic conditions that include

all of the vehicles on the road. These data were collected by automated vehicle tracking systems using surveillance cameras, which tracked approximately 75% of the vehicles. Subsequently, manual correction of false-positive and false-negative errors was required, which is inefficient [14]. Even with a fully automated system, camera-based detection has difficulty converting pixel coordinates into real-world coordinates (i.e., camera calibration). With the development of remote sensing technology, global positioning system-based (GPS-based) measurements, such as those acquired by probe cars, smartphones, and car navigation systems, have been used to extract vehicle trajectories. However, these devices can only provide trajectories for equipped vehicles, and the accuracy is insufficient for analyzing traffic dynamics [15, 16].

Unmanned aerial vehicles (UAVs), the use of which has increased in traffic-related research, can mitigate these difficulties [17–20]. The key advantages of UAVs as a tool for collecting vehicles' trajectories are that they can hover (stationary flight) above the traffic to obtain images without causing any disturbance or geometric image distortion. This approach can facilitate efficient supervised learning for vehicle detection with simplified preprocessing. However, the images provided by UAVs have the disadvantages such that vehicles appear as small objects in a large-scale image, and they may be partially occluded by shadows and road facilities. This problem can be further exacerbated in congested traffic. Although some studies have tried to extract vehicle trajectories from UAV images, they have limited success in improving accuracy due to the use of conventional vehicle detection methods, such as background subtraction, optical flow, and blob analysis. These approaches cannot robustly detect the exact location of vehicles in congested traffic, which results in false tracks in the tracking process [7–9, 18].

To accurately detect vehicles, some researchers have suggested a supervised learning-based method for UAV images. These methods use a trained vehicle classifier that shows superior performance based on effective feature representation and a strong learning algorithm [13, 17, 19, 20]. Some of these methods extend the vehicle detection to tracking by matching the detected vehicles in sequential frames [13]. Recently, deep learning approaches, especially the convolutional neural network (CNN), have been applied with great success to object detection tasks. Whereas the possibility of the performance improvements based on human-made features has reached its limits, the structure of CNN and its variants continue to make a new breakthrough based on the data-driven features of the images [21]. Some of the CNN-based methods have been suggested for application in detecting vehicles from UAV images and have shown better performance than feature-representation schemes [22, 23]. However, these studies were conducted with sufficient amounts of training data and high-resolution images and involved high computational cost, which limits their practical application. In actual research, it is not easy to obtain a sufficient number of samples due to the limited operation history of UAVs and the large labeling effort required. In addition, the image resolution tends to be lower to cover a wide spatial range. The most effective detection method can vary depending on the given conditions. Therefore, it is

necessary to consider the frameworks available for extracting vehicles trajectories that are the most practical in the analysis of actual traffic flow.

In this paper, we present a framework for extracting vehicles trajectories from UAV images in congested traffic conditions. After performing simplified preprocessing, we trained two different vehicle classifiers using the human-made features and the data-driven features extracted from the CNN structure, respectively. We then compared these two classifiers regarding their detection performance according to the number of training samples and the image sizes. Next, we extended this detection to tracking by assigning vehicle identifications (IDs), and we identified some tracking errors by detailed performance analysis and corrected them using postprocessing based on motion constraint in congested traffic. Unlike previous studies which evaluated only detection accuracy, we evaluated tracking accuracy including mismatching and location errors, using manually labeled ground-truth locations of the trajectories. Finally, with the extracted vehicle trajectories from the proposed method, we calculated and evaluated the aggregated traffic data that are generally used for the traffic-flow studies.

The key contributions of this paper can be summarized as follows: (a) we propose and compare two different learning-based frameworks for detecting vehicles and extend these to vehicle tracking, to extract multivehicle trajectories in congested traffic from UAV images; (b) our framework shows promising performance using only a small number of training samples and involves a reasonable computation cost for constructing large-scale vehicle trajectories dataset; (c) we propose postprocessing for trajectories in congested traffic using motion constraints and conduct a detailed performance analysis to present the quality of the trajectories; and (d) evaluation with the aggregated traffic data confirms that the extracted trajectories are enough to be applicable for the traffic-flow analysis.

The remainder of this paper is organized as follows. First, we describe the existing frameworks used to extract vehicle trajectories and examine their advantages and shortcomings. In the next section, we discuss in detail the methodology of our framework. Then, we present the experimental process and describe the data we used. We then conduct a detailed performance analysis of the results, which confirm the excellence of the proposed framework. Lastly, we discuss our findings, make brief concluding remarks, and share future research plans.

2. Related Work

Many previous studies have conducted vision-based vehicle detection and tracking in the collection of traffic information. The vehicle tracking outputs include temporal information about the object in sequential frames, whereas vehicle detection outputs only localize the objects in images. For single-object tracking, many studies have mainly focused on designing a sophisticated appearance model [24, 25] or a motion model [26] to deal with challenging environments involving occlusion and illumination variations. For multiple-object

tracking, the key issue is to overcome the various interruptions such as interaction among crowded objects, the similar appearance among objects, and track initialization and termination as the objects appear in or disappear from each frame [27, 28]. Therefore, the density of the objects in images determines the tracking method. UAV images taken in the vertical direction from above congested traffic can be categorized as semicrowded images, which most researchers manage using the “tracking-by-detection” approach [28]. In this section, we mainly review existing methods for detecting vehicles that heavily affect the performance of “tracking-by-detection” approach, as well as some extensions for tracking vehicles in UAV images.

The use of surveillance cameras, which are deployed extensively around the world, has been studied for a long time. Coifman et al. [10] proposed a real-time system for vehicle tracking and tested this system on both free-flow and congested traffic. Their system exhibited good performance in obtaining the speeds of vehicles, but inferior performance in obtaining their flow, density, and trajectories. Subsequently, Wan et al. [11] suggested an improved framework that included a novel 3-D Hungarian algorithm for tracking vehicles. This algorithm detected the center of the mass of the vehicle and solved the assignment problem for sequential frames to identify and match each vehicle. This framework showed excellent performance for speed and volume, but no vehicle trajectories of the vehicles were extracted in this study. Since surveillance cameras gather tilted angles in their images, the camera must be calibrated to transform the pixels in the image into real-world coordinates. This procedure distorts the shapes of vehicles, which results in only the vehicles’ specific proportions being detected and tracked rather than providing bounding boxes around each vehicle. The camera calibration process also generates bias due to the sensitive parameters of the cameras [29].

Because of these fundamental difficulties associated with using surveillance cameras, UAV-based studies have been proposed as an alternative. Azvedo et al. [6] used median-based background subtraction and blob analysis to detect vehicles. The authors collected partial trajectories by aircraft flying over an extensive area. Ke et al. [18] suggested unsupervised, real-time vehicle detection and tracking systems that used interest-point tracking and clustering. Although the authors extracted no vehicle trajectories, their framework performed well in estimating speed but showed slightly lower performance in counting vehicles. Khan et al. [8] proposed an automated framework based on background subtraction using optical flow and blob analysis. These authors tested their framework on a four-street intersection but performed no quantitative evaluation of vehicle trajectories. The automated frameworks mentioned above that use unsupervised detection methods can be applied rapidly and conveniently without any training classifier. However, these methods are sensitive to the complexity of the scene, such as adjacent vehicles, shadows, road signs, and road facilities. Therefore, their performance decreases significantly in congested traffic in which vehicles are crowded together and moving slowly [13].

To overcome these limitations, supervised learning-based vehicle detection has been applied in computer vision based on the development of feature representation and a powerful learning algorithm. Since these methods detect vehicles as bounding boxes rather than points or blobs, they can be applied robustly in congested traffic conditions. Grabner et al. [17] were the first to suggest the supervised learning-based vehicle detection method using aerial images. The authors utilized the AdaBoost classifier with three feature types conducted tests on an urban road and a parking lot. Liu and Mattyus [20] proposed a multiclass vehicle detection process based on AdaBoost and integrated channel features. Their methodology showed higher accuracy in counting vehicles on the road than other relevant approaches. Xu et al. [13] extended the detection of vehicles using the Viola and Jones (V-J) object detection scheme [30] for the tracking of vehicles, and they showed that the V-J scheme outperforms many other methods based on feature representation. The authors trained the classifier using a vast number of training sets and conducted experiments at various sites, including those with congested traffic. Their framework performed well even in congested flow. However, they only evaluated the performance of this scheme using detection-based measures, such as precision and recall, and not tracking errors, such as mismatching and location errors. To confirm the usability of the vehicle trajectories for analyzing traffic dynamics, the measures are needed to consider the spatiotemporal context of the vehicles to account for tracking errors.

More recently, deep learning approaches based on CNN have shown impressive performance in object detection. Specifically, region-proposal-based detectors such as R-CNN [31], Fast R-CNN [32], and Faster R-CNN [33] have been proposed to precisely locate objects in an image. Xu et al. [22] applied the Faster R-CNN for vehicle detection from UAV images, and Faster R-CNN showed better performance than conventional methods with a sufficient amount of computation cost and training samples. However, the authors did not extend the detection to tracking, and a detailed comparison with a more advanced method based on human-made features is needed to support the conclusion of the performance excellence of the Faster R-CNN. Tang et al. [23] proposed a modified Faster R-CNN for small-sized vehicles from UAV images. They employed hierarchical feature maps to deal with features in small pixels. Their method showed better performance than that of the aggregated channel feature (ACF) detector. However, since the test was only conducted on urban streets and a parking lot, its performance in congested traffic should be determined.

Several implications can be drawn from the above review of previous studies. First, the learning-based vehicle detection method has shown superior performance compared with the conventional unsupervised method. CNN-based approaches, in particular, have shown promising performance in the detection of vehicles in UAV images. For practical application, however, evaluation in congested traffic conditions is necessary with respect to efficiency and accuracy. Second, although a few studies have extended vehicle detection to tracking, there is room for improvement by taking advantage of UAV images in congested traffic conditions. Therefore, an

overall framework for extracting vehicle trajectories should be specified. Third, more detailed analyses of tracking errors are needed to confirm the usability of vehicle trajectories.

3. Materials and Methods

3.1. Overview. Our framework for extracting vehicles trajectories comprises four sequential processes, as shown in Figure 1. First, we manually extract the road coordinates in the first frame and use these for all the frames, since the UAV can remain stable and balanced in flight. Then, we detect vehicles on the road using a supervised classifier trained by ACFs [34] and Faster R-CNN [33]. The locations of vehicles in each frame are collected during this process. To create the trajectories of vehicles over time, the detected locations of the same vehicles in sequential frames are matched by solving the assignment problem. Lastly, we perform postprocessing to remove three types of false positives that occurred in the tracking process, using motion constraints in congested traffic. Detailed descriptions of these processes are presented in the following sections.

3.2. Preprocessing: Road Extraction and Camera Calibration. Road extraction and camera calibration are the important preprocessing steps in the vision-based detection and tracking of vehicles. Road extraction can reduce the computation time required for the detection process and eliminate false-positive errors by limiting the search space. Camera calibration involves converting image coordinates into real-world coordinates. For this process, exact camera parameters are required, including the focal length, camera height, and tilt angle. These sensitive parameters influence the accuracy of the trajectory [29]. In this study, we simplified the two preprocessing steps described above by using UAV images. Since all images were obtained from a vertical direction above the traffic, this eliminated the need to calibrate the camera. In some cameras, the fish-eye effect must be removed [8], which was not necessary in our case. Also, since UAVs can hover and produce videos that maintain almost constant coordinates among frames, we were able to use the manually extracted road coordinates from the first frames for all the frames. Although, we focused on the automated process excluding preprocessing, real-time approaches require automated road extraction and camera calibration. Further details about these processes are presented in other work [8, 35].

3.3. Vehicle Detection

3.3.1. Aggregated Channel Features. ACFs [34], which were improved from integral channel features [36], have been shown to exhibit efficient and accurate performance in the detection of pedestrians [37]. Liu and Mattyus [20] recently showed that these features also enable the very fast and accurate detection of vehicles from UAV images. The integral image firstly suggested by Viola and Jones [38] is an intermediate representation of images for computing features, and, using this concept, Haar-like features, which are approximate local derivatives, can be computed at any

scale or locations. ACFs are calculated for several channels by linear and nonlinear transformations of the input images. We used normalized gradient magnitude, six channels of the histogram of oriented gradients, and LUV color as channels [36]. Single pixel lookups aggregated from each channel in a smoothed image can be computed extremely fast using the integral image, and we used these as first-order features for detection. Higher-order channel features (e.g., Haar-like features) can also be computed using multiple first-order features. Since these features naturally integrate data from different sources (e.g., lighting and weather conditions), they are suitable for collecting traffic data. We computed the ACFs using fast feature pyramid, which extracts sufficiently approximated features on finely sampled pyramids for rapid calculation [34].

To extract only the richer ACFs, only critical feature should be selected in the learning algorithm. We used AdaBoost as the training classifier, which combines weak classifiers, including a small number of features, to form a stronger classifier. In each iteration, t , a new weak classifier, $h_t(\mathbf{x})$, is trained using adaptively weighted samples that had been previously misclassified by a weak classifier. This combined classifier has the form $H(\mathbf{x}) = \sum_t \alpha_t h_t(\mathbf{x})$, where α_t is calculated by greedily minimizing a loss function in each iteration, t [39]. As a weak classifier, we used a decision tree with a maximum depth of five. We trained a boosted classifier with 1,024 weak classifiers. Since using all of the weak classifiers on every pixel in an image is inefficient, to improve efficiency, a cascade of classifiers is required, which removes negative subwindows using a smaller boosted classifier. We used a “soft cascade structure” as described in [38], to relax the distinct stages of the cascade using the rejection threshold. When performing the detections, multiple detections near one vehicle occurred because our trained classifier used a sliding window to detect vehicles in the image. To suppress these overlapping detections, we used nonmaximal suppression with an overlap threshold, whereby if the overlapped area of two detected vehicles was greater than 40% of their total area, only the vehicle with the higher detection score (i.e., higher probability of being detected as a vehicle) was retained.

3.3.2. Faster R-CNN. R-CNNs extract region proposals, which indicate candidate locations for objects from the raw image and then compute the CNN features using these locations as inputs [31]. Fast R-CNN features a major change in the method used to compute the CNN feature [32]. After extracting region proposals from the data, this method uses the coordinates of region proposals to select regions of interest (ROI). R-CNN and Fast R-CNN both use selective searches to generate region proposals, whereas Faster R-CNN uses a region proposal network to improve efficiency [33]. Faster R-CNN comprises two parts as shown in Figure 2(a): a region proposal network (RPN) and an object detection network. The object detection network has the same framework as the Fast R-CNN. The RPN uses a feature map from a convolutional neural network and creates up to 300 region proposals. It uses an anchor to identify the area where objects

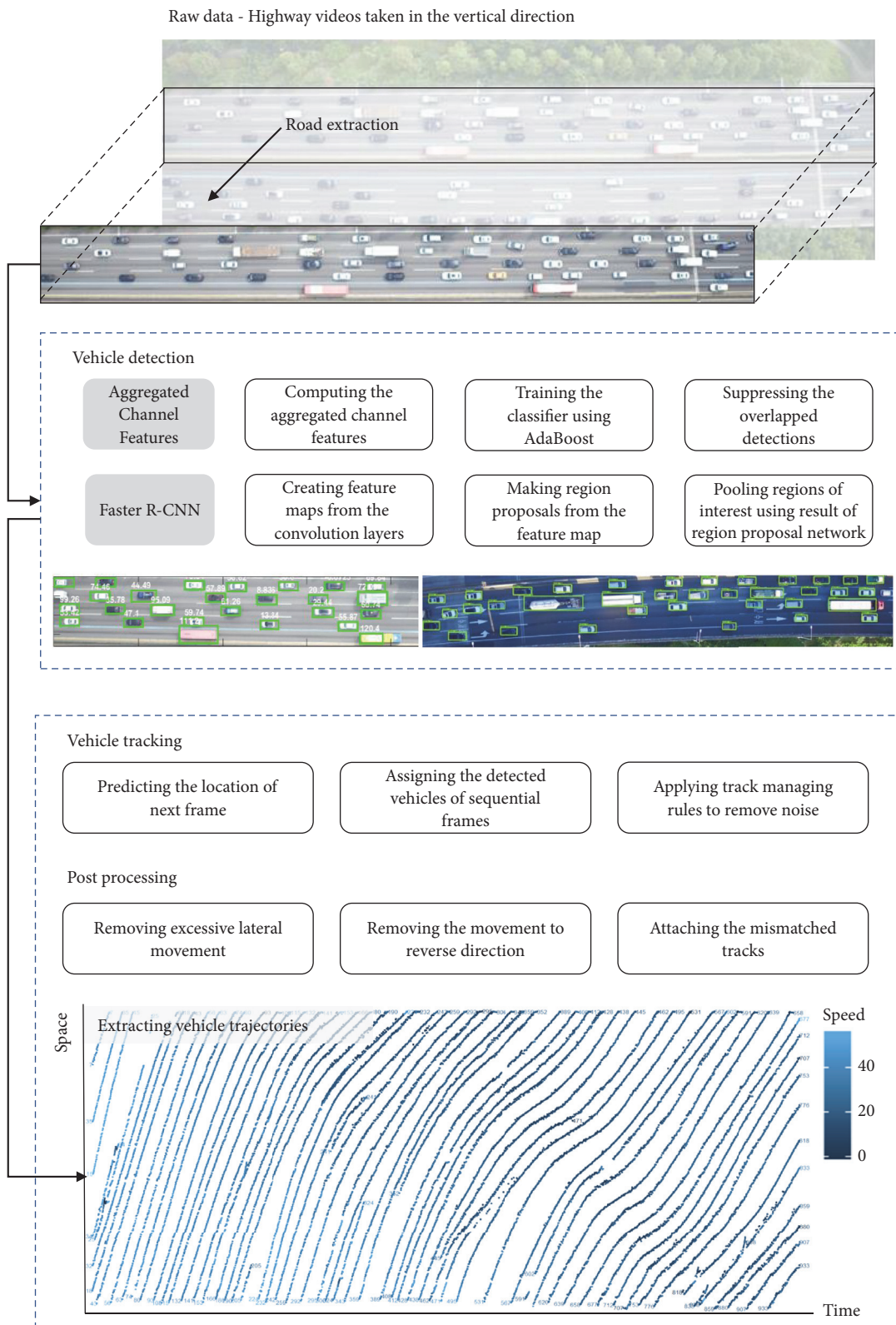


FIGURE 1: Flowchart of four sequential processes in the proposed framework for extracting vehicle trajectories, i.e., road extraction, vehicle detection, vehicle tracking, and postprocessing.

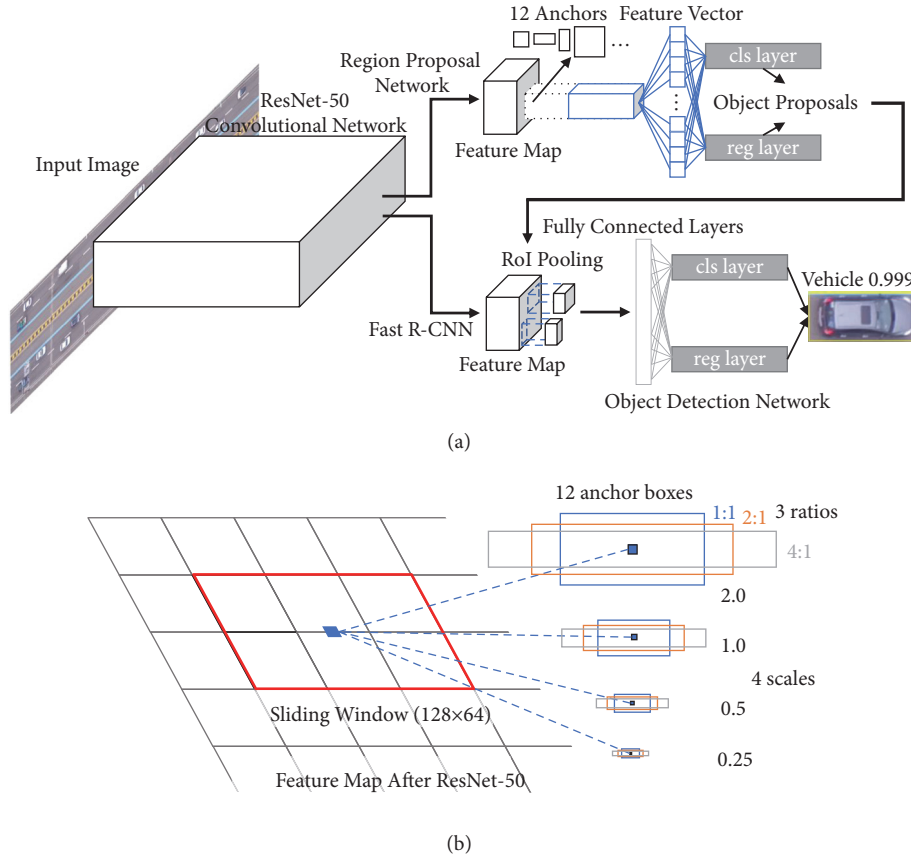


FIGURE 2: Architecture of Faster R-CNN: (a) overall architecture and (b) design of each anchor.

are likely to exist. The anchor is a type of sliding window with different sizes and ratios. We used an anchor with base size of 128×64 pixels, a scale from 0.25 to 4.0, and a ratio ranging from 1.0 to 4.0 as in Figure 2(b). We chose the hyperparameters based on background knowledge about the image which contain rectangular objects of relatively uniform size. The loss of each anchor is calculated by the cross-entropy loss, which has two classes: object and background. The result generated by the region proposal network is then given to the object detection network. ROI are pooled and each ROI is trained by a classifier and a bounding box regressor. The classifier loss and the bounding box regressor are calculated in integrated loss form, as follows:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (1)$$

where N_{cls} and N_{reg} are the minibatch size and number of anchor locations, respectively. p_i is the predicted probability of anchor i being an target object, and p_i^* is the ground truth, which has a value of 0 or 1, depending on whether anchor i is an object or not. t_i and t_i^* are modified forms of bounding-box and ground-truth-box coordinates. L_{cls} uses

cross-entropy loss and L_{reg} uses the smooth L_1 loss that has $t_i - t_i^*$ as an input.

3.4. Vehicle Tracking. Based on the detection results, we implemented a simple motion-based tracker. The objective of tracking vehicles is to extract their trajectories over time by locating their positions. Of the various object-tracking methods [27, 28], we used simple point tracking based on individually detected vehicles. This method associates the vehicles in frame t to those in frame $t + 1$ and then assigns identifications (IDs) for extracting the trajectory of each vehicle. The vehicle tracking procedure has three steps. In the first, assuming constant velocity, we used previously tracked vehicles to predict their locations in frame $t + 1$ using a Kalman filter [40]. Although we could instead use the vehicle's location in frame t , the prediction process reduces the influence of noise associated with the detection error of the t frame and of some partially missed vehicles in one trajectory. In the second step, we associated the detected vehicles across frames and solved the optimization problem (i.e., assignment problem) to minimize the correspondence cost of predicted and detected vehicles in frame $t + 1$. We defined this cost as the Euclidean distance between the predicted and detected locations in the frame $t + 1$. To bound the search region in congested traffic conditions, we added the constraint that the instantaneous speed of the

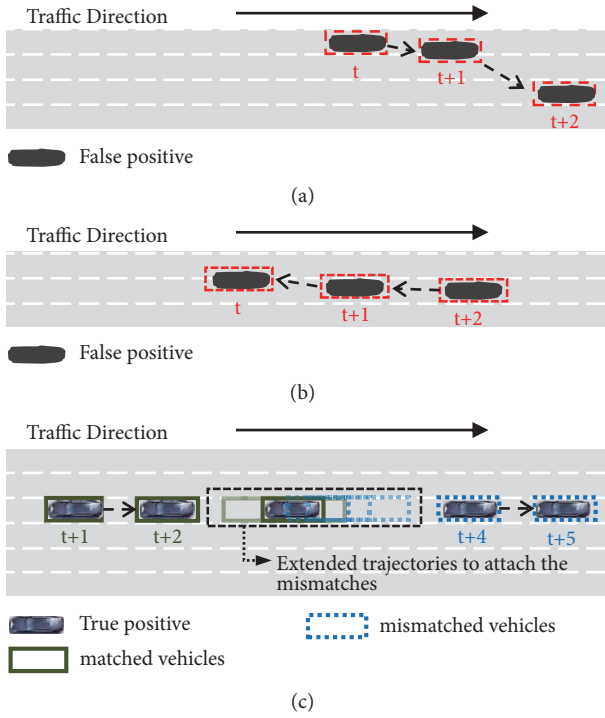


FIGURE 3: False tracks generated in congested traffic: (a) excessive lateral movement, (b) reverse direction, and (c) mismatching.

vehicles should be less than 85 km/h. We used the Hungarian algorithm to solve this optimization problem [41]. In the last step, we created and discarded tracks based on the assigned results and management rules. We created tracks when the visible count (i.e., the number of frames in which the vehicle was detected) was more than 15 frames, and we deleted them if they were not visible in at least 15 frames or if the visibility, which is the total number of visible tracks counted divided by the total number of frames, was less than 0.6.

3.5. Postprocessing. Vehicle tracking in congested traffic is significantly affected by false positives (i.e., detection of vehicles when none is present and detection of vehicles with incorrect size) and false negatives (i.e., detecting no vehicles when some are present). Duplicate detection, road facilities, shadows, and partially detected trucks are typical false positives in UAV images. Figure 3 shows three types of frequently occurring false tracks in the tracking process. Because false positives did not occur uniformly in sequential frames, their tracks appear to differ from the motion of a typical vehicle in a traffic jam. Excessive lateral movement and reverse direction are examples of cases in which tracks were generated by false positives, as shown in Figures 3(a) and 3(b). The mismatching in Figure 3(c) is an example of a track becoming disconnected in the middle and being converted to a new track. This error greatly influences the microscopic trajectory analysis, although it does not affect the collection of aggregated speeds or densities. Both false negatives and false positives can cause this disconnection.

Since vehicles in congested traffic have strong motion constraints, they can be used to remove false tracks. During postprocessing, we identify three types of false tracks and remove the errors associated with each. Three cases of postprocessing were proposed to remove each case of errors. Initially, we eliminated the trajectories of vehicles with excessive lateral movement and opposite-direction movements based on their speeds and directions. Subsequently, we extended the trajectories of each vehicle by three seconds before and after the vehicle, while assuming constant speed, and attached mismatches in cases where two extended trajectories were overlapped (i.e., closer than half of the vehicle's length).

3.6. Quantitative Evaluation. We evaluated the false positives (FP) and false negatives (FN) identified in the vehicle detection procedure based on the manually labeled ground-truth locations of the vehicles. The precision and recall are well-known measures for object detection [42] to quantify these errors. Precision is defined as the accuracy of predicting true positives from among all of the detected samples as in (2), and recall is the number of true positives detected among all the ground-truth locations of vehicles, as in (3). The F-measure is the trade-off between recall and precision, which have equal importance as in (4). When the detected area and the ground-truth area along the road were overlapped by more than 50% of their combined area, we considered this to be a true positive (TP). We computed the F-measure having same precision and recall to compare the detection methods in the following sections.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F - measure = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision} \quad (4)$$

We evaluated the vehicle tracking performance using the multiobject-tracking accuracy (MOTA) and multiobject-tracking precision (MOTP) metrics [43]. MOTA generates three error ratios with respect to ground truth (g_t) over time t , i.e., the ratio of misses (m_t), false positives (fp_t), and mismatches (mme_t), as shown in (5). Summing these three errors produces the total error. MOTP is the total position error for the overall frame ($\sum_{i,t} d_{i,t}$) of matched object-hypothesis pairs (i.e., detected true positives and ground-truth locations of vehicles) averaged by the total number of matches made ($\sum_t c_t$), as shown in (6).

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}$$

$$the \text{ ratio of misses} = \frac{\sum_t m_t}{\sum_t g_t}$$

TABLE 1: Description of video datasets.

Video	Site Locations	Number of Lanes	Spatial Range	Size of Vehicles	Traffic Density	Mean Speed	Lighting Condition
Test-Video 1	Korea Expressway No. 1	Four Lanes + Bus Exclusive Lane	188 m	40×91 pixels	50.2 veh/km/lane	22.3 km/h	Cloudy After Sunrise
Test-Video 2	Korea Expressway No. 120	Four Lanes + Entry and Exit Lane	137 m	60×132 pixels	56.1 veh/km/lane	20.8 km/h	Clear Before Sunrise

TABLE 2: Training and test sets for detection and tracking.

Video	Training Set	Test Set for Detected Vehicles	Test Set for Vehicle Trajectories
Test-video 1	4,000 labeled vehicles (one image every 6 sec)	1,000 labeled vehicles (one image every 6 sec)	32,800 labeled trajectories of 61 vehicles for two consecutive minutes (25 images every 1 sec)
Test-video 2	4,000 labeled vehicles (one image every 6 sec)	1,000 labeled vehicles (one image every 6 sec)	-

$$\text{the ratio of false positives} = \frac{\sum_t fP_t}{\sum_t g_t}$$

$$\text{the ratio of mismatches} = \frac{\sum_t mme_t}{\sum_t g_t} \quad (5)$$

$$MOTP = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t} \quad (6)$$

To evaluate the applicability of traffic-flow analysis, the five kinds of traffic data such as space-mean speed, density, flow, spacing, and the number of lane changes are calculated using the extracted trajectories from vehicle tracking and postprocessing. The space-mean speed, density, and flow are calculated by Edie's definition [44] as in (7).

$$\begin{aligned} \text{space mean speed} &= \frac{\sum_i x_i}{\sum_i t_i} \\ \text{density} &= \frac{\sum_i x_i}{|A|} \\ \text{flow} &= \frac{\sum_i t_i}{|A|} \end{aligned} \quad (7)$$

where x_i is the distance traveled by the i^{th} vehicle in the time-space region, t_i is the time spent by the i^{th} vehicle in the time-space region, and $|A|$ is the area of the time-space region. We also calculate the spacing of adjacent vehicles and the number of lane changes. All of these data except the number of lane change are aggregated by space-time resolution of 30m × 10s. A mean absolute percent error (MAPE) is used to evaluate these aggregated data. The number of lane changes is evaluated using a percent error (PE) after counting in total time and space.

4. Experiment and Results

4.1. Datasets. To verify the proposed frameworks, we recorded highway traffic videos in the vertical direction using a DJI Inspire Pro 1 equipped with a Zenmuse X5 camera, which is a quadcopter drone with 4K video and 3-axis gimbal stabilizers. The resolution of the video was 3,840 × 2,160 (25 fps). Table 1 shows details regarding the video data sets. We conducted the experiments in congested traffic with respect to two different conditions: lighting and the size of the vehicles. Those conditions have a significant impact on the vision-based approach even when using the learning-based method, which generally has the merit of generalized performance. We took the first test-video over a 188-m range of the four-lane Korea Expressway No. 1, which includes one exclusive bus lane, from 8:12 A.M. to 8:24 A.M on July 15, 2016. This was a cloudy morning at peak traffic after sunrise. The average traffic density was 50.2 vehicles/km/lane, and the average speed was 22.3 km/h. The average vehicle size was 40 × 91 image pixels. We took the second test-video on a 137-m section of Korea's four-lane Expressway No. 120, which includes entry and exit lanes, from 7:09 A.M. to 7:20 A.M. on September 20, 2016. This was a clear morning at peak traffic before sunrise, with a shadow covering the entire road. The average traffic density was 56.1 vehicles/km/lane, and the average speed was 20.8 km/h. The average vehicle size was 60 × 132 image pixels, which is approximately 1.5 times larger than that in the first video.

To train the detector, we used manually labeled training sets. We set the positive samples to include only vehicles, and the negative samples to include the background as well. As presented in Table 2, we constructed training sets with 4,000 positive samples in each test-video. We labeled these vehicles in more than 100 images at 6-second intervals to prevent one vehicle from being labeled more than twice. We tested the vehicle detection performance using images with 6-second intervals containing 1,000 labeled vehicles. In

vehicle tracking performance, we focused on evaluation of the proposed postprocessing with simple tracking algorithm, and, in particular, analyzing how tracking performance is improved by the postprocessing according to variation in detection performance. To reduce the large labeling effort, a detailed performance of tracking was evaluated using 2,880 images at 0.04-second intervals (two consecutive minutes) containing 32,800 carefully labeled trajectories of 61 vehicles only in test-video 1. We used a user-friendly labeling tool that utilizes cubic interpolation to perform this large-scale labeling task [37]. To implement the ACF, vehicle tracking, and postprocessing, we used the single-threaded MATLAB toolbox for computer vision [45] and an Intel i7-6700HQ CPU@2.6 GHz processor with 16 GB of memory, and Python programming language to implement only the Faster R-CNN, based on the tensor flow object detection API [46] with an Nvidia Geforce GTX 1050 TI, 4GB GDDR5.

4.2. Results of Vehicle Detection. After the simplified preprocessing of the UAV images, the vehicle classifier was trained using the ACF and Faster R-CNN on the labeled training data. To determine the efficiency of the training, we conducted a sensitivity analysis to identify the variation in the detection accuracy with the number of positive samples. To do so, we divided the training sets into sample subsets of 500, 1,000, 2,000, 3,000, and 4,000. Then, we randomly resampled each of these subsets ten times to reflect the performance variations. Figure 5 shows the F-measures of the ACF and Faster R-CNN with error bars for the two test videos. We calculated the F-measures of the error bars by adjusting the threshold for true positive to obtain the same precision and recall values. The error bar indicates the standard deviation of the measures.

In test-video 2, the performance of the Faster R-CNN was overwhelmingly superior to that of the ACF. The mean F-measure for the ACF was 84.7% for 500 samples and it peaked at 91.0% for 4,000 samples, with a standard deviation (SD) of 0.3%. The mean F-measure of the Faster R-CNN was 95.8% for 1,000 samples and it peaked at 97.1% for 4,000 samples, with an SD of 1.3%. These results indicate that the Faster R-CNN can capture richer vehicle features in images during clear weather. On the other hand, in test-video 1, the mean F-measure of the ACF was 80.9% for 500 training samples and it peaked at 93.0% for only 3,000 training samples, with an SD of 1.9%. It subsequently decreased slightly to 91.5% for 4,000 samples. The mean F-measure of the Faster R-CNN was 50.2% for 500 training samples and it gradually increased to 91.6% for 4,000 training samples, which was slightly lower than that of the ACF. Although using more training samples improves the performance of the Faster R-CNN in test-video 1, it shows comparable performance with the ACF for a small sample size. Performance degradation of the Faster R-CNN in lower-resolution images has been reported in previous studies [23, 47], and vehicles in UAV images can also suffer from these problems. This is because only the plain features that cannot be used to distinguish between vehicles are extracted from the ROI pooling layers due to the low-resolution feature map [47]. Therefore, if the

image resolution is not high enough to identify sophisticated features of vehicle, the use of ACFs, i.e., intuitive human-made features, may be more effective in detecting vehicles than the Faster R-CNN. Also, the results from two videos show the relation between model complexity and saturation. The complexity of the model determines whether the detector continues to improve as the number of training samples increases. When the model complexity is close to saturation, their accuracy does not increase with additional training data and can even lead to overfitting [48]. ACFs might be saturated with about 3,000 samples in test-video 1, whereas the Faster R-CNN, which is more structurally complex, was not yet saturated with 4,000 samples. In test-video 2, which contains more information in the vehicle images, neither model was saturated with 4,000 samples, and their performances could be improved with more training data.

Figure 6 shows the false positives of the best-performing detectors trained by the ACF and Faster R-CNN for each test-video. In test-video 1, the false positives of both of the detectors were caused by duplicated detections of vehicles. These errors can generate the false track and mismatch in tracking process, which is main factor reducing accuracy of vehicle trajectory. On the other hand, in test-video 2, the ACF suffered from the false positives caused by truck as well as other road environments whereas the Faster R-CNN successfully coped with those false positives. This explains the performance difference between two detectors in test-video 2.

4.3. Results of Vehicle Tracking and Postprocessing. In our framework, since we performed vehicle tracking based on the detection results, the tracking accuracy increased with increases in the detection accuracy. To present the sensitivity of the tracking accuracy based on the detection results, we performed vehicle tracking using four different vehicle classifiers with F-measures of 86.0%, 90.1%, 94.5%, and 97.4% in test-set for vehicle tracking. We also used postprocessing with motion constraints to remove the expected errors in the tracking process. Then, we conducted a detailed performance analysis using the large-scale labeled trajectories in test-video 1 where the duplicated detections were frequently observed. Those false positives significantly decrease accuracy of vehicle trajectory by generating the false track and mismatch in tracking process. Also, the detection results of ACF, which showed better performance than Faster R-CNN in test-video 1, was used for vehicle tracking and postprocessing because both of the detectors shared a similar cause of the false positives. We selected lane 3 (Figure 4(a)) as the target, where lane changes occur frequently. Table 3 presents the variation in tracking performance based on the detection accuracy and postprocessing. Each row shows the four different classifiers and whether or not postprocessing was performed, and each column shows the corresponding detection and tracking performances. Figures 7(a) and 7(b) show the changes in the trajectories before and after postprocessing. The numbers in Figures 7(c) and 7(d) are the start and end points of each identified vehicle, respectively. The shaded regions of Figure 7(c) are the false positives that showed a different pattern from the typical tracks and the mismatches divided

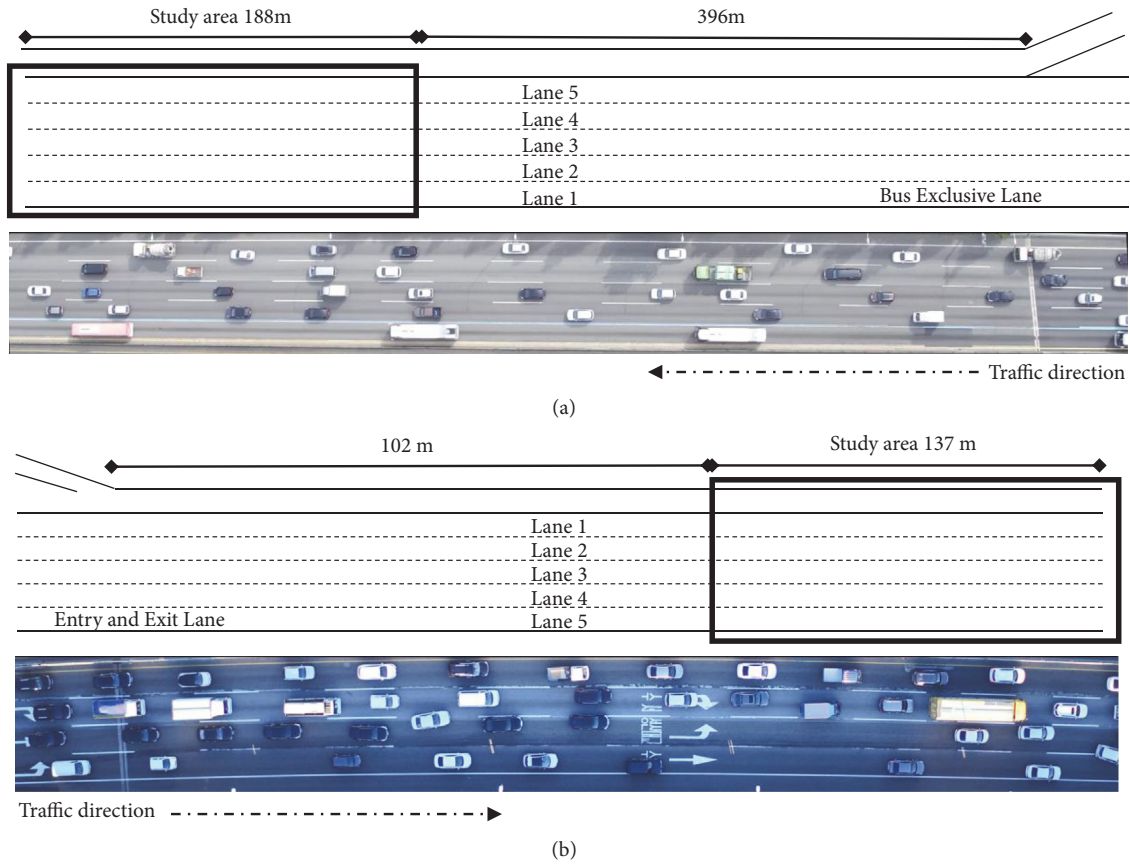


FIGURE 4: Images of the studied congested freeway from the UAV: (a) Korea Expressway No. 1 with cloudy after sunrise and (b) Korea Expressway No. 120 with clear before sunrise.

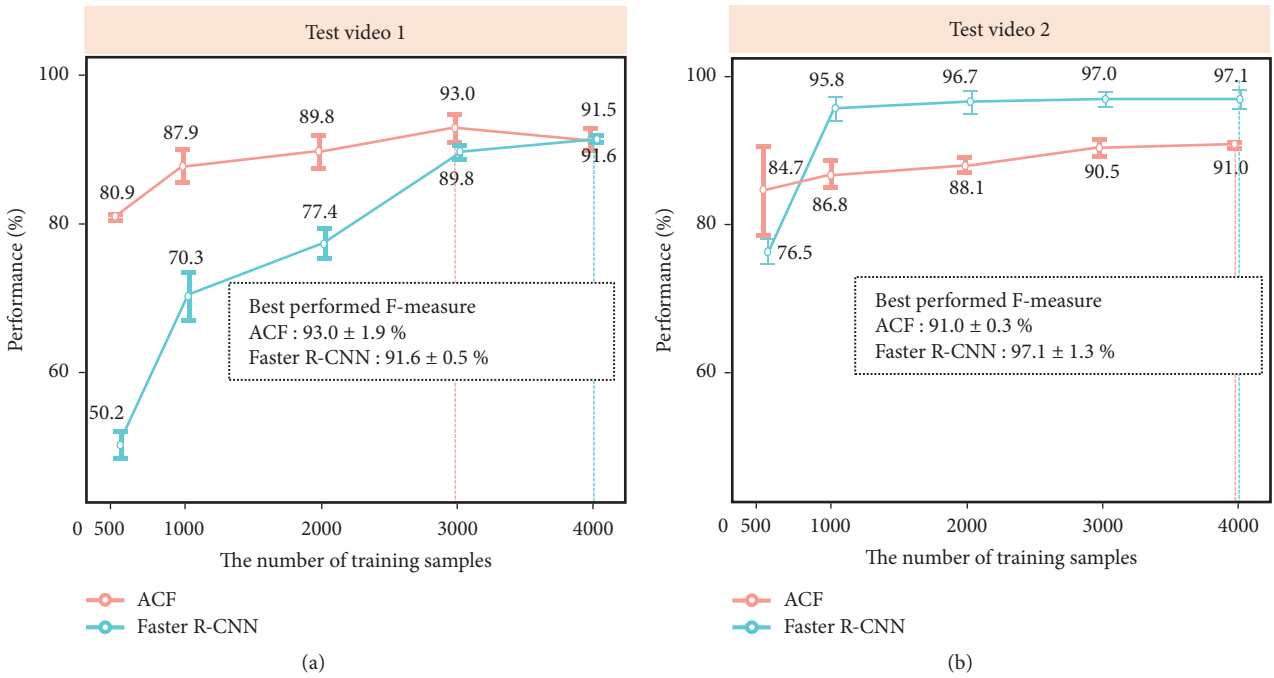


FIGURE 5: Variation in detection performance with the number of training samples. (a) Test-video 1 shows the best F-measure of 93.0% with a standard deviation of 1.9% using ACF with 3,000 samples, and (b) test-video 2 shows the best F-measure of 97.1% with a 1.3% standard deviation using Faster R-CNN with 4,000 samples.

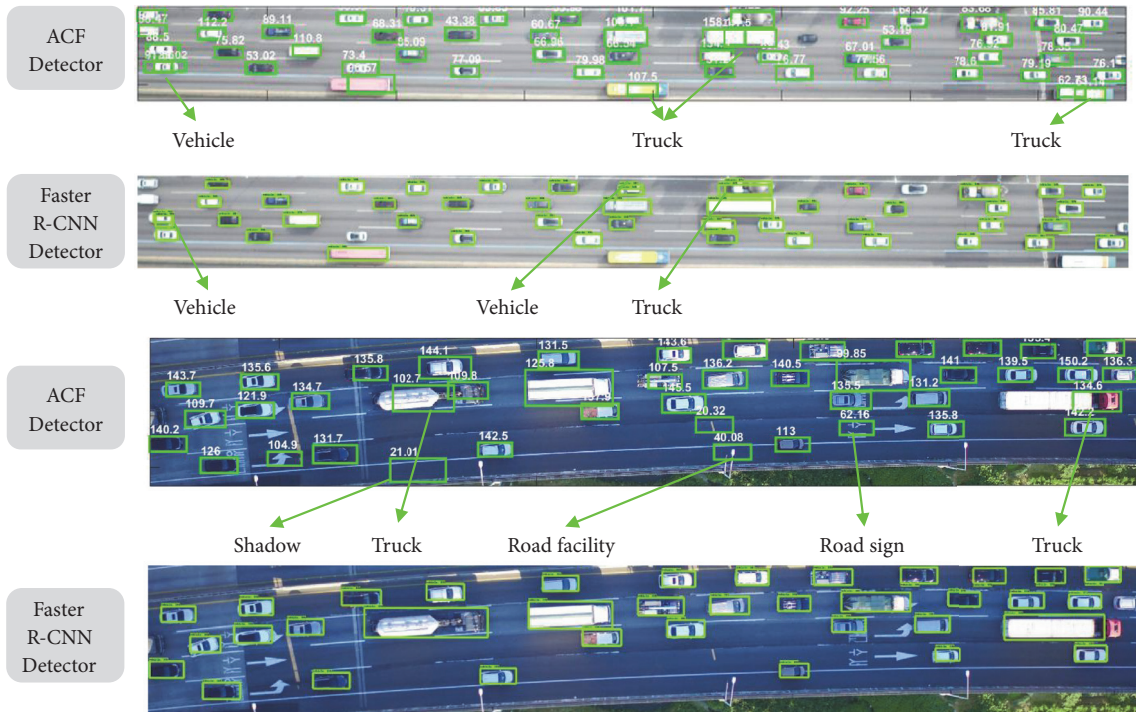


FIGURE 6: False positives from the two different detectors for each test-video.

into two different IDs. While postprocessing effectively removes the mismatched trajectories and false positives, it slightly increases the ratio of misses by removing some true positives. The postprocessing worked well for F-measures of 90.1%, 94.5%, and 97.4%, but it was less able to remove mismatches for an F-measure of 86.0%. This reduction was caused by difficulties in attaching mismatching trajectories when missed detection occurred to a greater extent in several consecutive frames. The MOTP was about 0.5–0.6 m regardless of the F-measure, which means that true positives provide reliable trajectories regardless of overall accuracy. In the best performance, the vehicle trajectories achieved location errors as low as 0.6 m with a MOTA of 89.9%. This suggests that the tracking process has a large impact on the overall quality of the vehicle trajectories as well as their detection.

4.4. Computation Time. The training and processing time is an important factor for extracting trajectories if a massive video dataset can be collected by UAVs. Table 4 shows the training times for detectors and the processing times for each step. In the table, we can see that the ACF and Faster R-CNN took about 31 minutes and 45 minutes, respectively, for 4,000 training samples using both the CPU and GPU. The training time of the Faster R-CNN is influenced by the hyperparameters, image sizes, and number of training samples. Therefore, the much shorter training time than that of a previous study [22] using a similar image size indicates the efficiency of the proposed method.

The total processing times of the ACF were 1.02 seconds and 1.34 seconds per image in the respective test videos,

whereas those of the Faster R-CNN were 0.61 seconds and 0.71 seconds per image, respectively. These results show that our model can be conducted with reasonable computation times to construct a dataset, although these times cannot be applied to a real-time approach. When we extracted one hour of trajectories (25 fps) using the Faster R-CNN from test-video 1, it took about 15 hours, and more powerful GPUs could significantly accelerate these overall processes.

4.5. Comparison of Methods to Obtain Aggregated Traffic Data. For application to the traffic-flow analysis, it is necessary to evaluate the reliability of aggregated traffic data obtained from vehicle trajectories such as space-mean speed, density, flow, spacing, and the number of lane changes. We compare the eight combinations of detection and tracking method including the four detectors and the two trackers. The original V-J detector [30] and Single Shot Multibox Detector (SSD) [49], which are two well-known object detectors using a human-made feature and a data-driven feature each, are used for comparison with the proposed methods, i.e., ACF and Faster R-CNN. The Kanade-Lucas-Tomasi (KLT) feature tracker [50] is also used for comparison with the proposed Kalman filter-based tracking method (KF).

The evaluation is performed on the same dataset which was used to evaluate the vehicle tracking and postprocessing. Excluding the number of lane changes, each value of the traffic data is calculated by space-time resolution of $30\text{m} \times 10\text{s}$ in lane 3. On the other hand, the number of lane changes is calculated in all the lanes over the entire period of time. Table 5 shows the testing results of eight combinations

TABLE 3: Variation in tracking performance based on the detection performance and postprocessing.

Detection, Tracking, and Postprocessing Performance							
Vehicle Classifier	F-measure	Postprocessing	Ratio of Miss	Ratio of False Positive	Ratio of Mismatch	MOTA	MOTP
Classifier 1	86.0%	✓	13.4%	6.8%	10.3%	69.5%	0.54m
			16.6%	4.8%	8.3%	70.3%	0.52m
Classifier 2	90.1%	✓	9.0%	7.7%	8.5%	74.8%	0.51m
			11.2%	3.9%	3.0%	81.9%	0.52m
Classifier 3	94.5%	✓	4.7%	8.6%	6.8%	79.9%	0.54m
			6.7%	7.4%	0.1%	85.8%	0.60m
Classifier 4	97.4%	✓	5.5%	3.7%	3.3%	87.5%	0.59m
			5.8%	3.2%	0.1%	89.9%	0.60m

TABLE 4: Computation time of detection, tracking, and postprocessing.

	Test-Video 1		Test-Video 2	
	ACF	Faster R-CNN	ACF	Faster R-CNN
Training Time for 4,000 samples	31 minutes	45 minutes	34 minutes	40 minutes
Processing Time per Frames	Vehicle Detection	0.83 seconds	0.41 seconds	1.15 seconds
	Vehicle Tracking	0.19 seconds		0.18 seconds
	Postprocessing	0.01 seconds		0.01 seconds
	Total	1.03 seconds	0.61 seconds	1.34 seconds

TABLE 5: Performance evaluation for obtaining aggregated traffic data.

Metric	Proposed Kalman Filter-Based Method (KF)				The Kanade-Lucas-Tomasi Tracker (KLT)			
	VJ	SSD	ACF	Faster R-CNN	VJ	SSD	ACF	Faster R-CNN
Processing Time (fps)	2.0	3.3	1.0	1.7	1.3	1.8	0.8	1.2
The Number of Lane Changes (PE, %)	+8.3%	+4.2%	+4.2%	+8.3%	+12.5%	+4.2%	+8.3%	+4.2%
Space-Mean Speed (MAPE, %)	3.0%	1.1%	2.3%	3.2%	3.5%	2.3%	3.4%	1.9%
Density (MAPE, %)	5.0%	6.7%	3.1%	5.8%	7.5%	2.7%	8.2%	2.3%
Flow (MAPE, %)	6.1%	6.8%	3.2%	7.1%	7.1%	2.8%	8.2%	2.2%
Spacing (MAPE, %)	7.4%	4.8%	5.1%	6.1%	10.1%	6.0%	11.1%	5.7%
Average (MAPE, %)	5.4%	4.9%	3.4%	5.6%	7.1%	3.5%	7.7%	3.0%

of detection and tracking method. With the proposed KF-based method, the ACF shows the best overall performance (3.4%) and the Faster R-CNN shows the lowest overall performance (5.6%). This is because the Faster R-CNN showed lower detection accuracy than ACF in test-video 1 although the detection accuracy is the critical factor for the KF-based method as shown in the prior section. The average performance of the SSD (4.9%) and the VJ (5.4%) is better than those of the Faster R-CNN, and the SSD achieved the fastest processing time (3.3 f/s). With the KLT feature tracker, however, the Faster R-CNN shows the best average performance (3.0%) with slightly slow speed (1.2 f/s) among

all combinations. This is because the tracking precision is more important for collecting the aggregated data than the tracking accuracy, and the Faster R-CNN has the advantage of precision through the bounding box regression [33]. Also, the KLT tracker, which extracts features from the detected bounding box, can effectively supplement the low tracking accuracy. The averaged performance of SSD, which uses the bounding box regression, is also improved from 4.9% to 3.5% with efficient speed (1.8 f/s) while the average performance of ACF and VJ decreases with KLT. These results indicate that it is important to find an optimal combination of detector and tracker depending on the data to be collected.

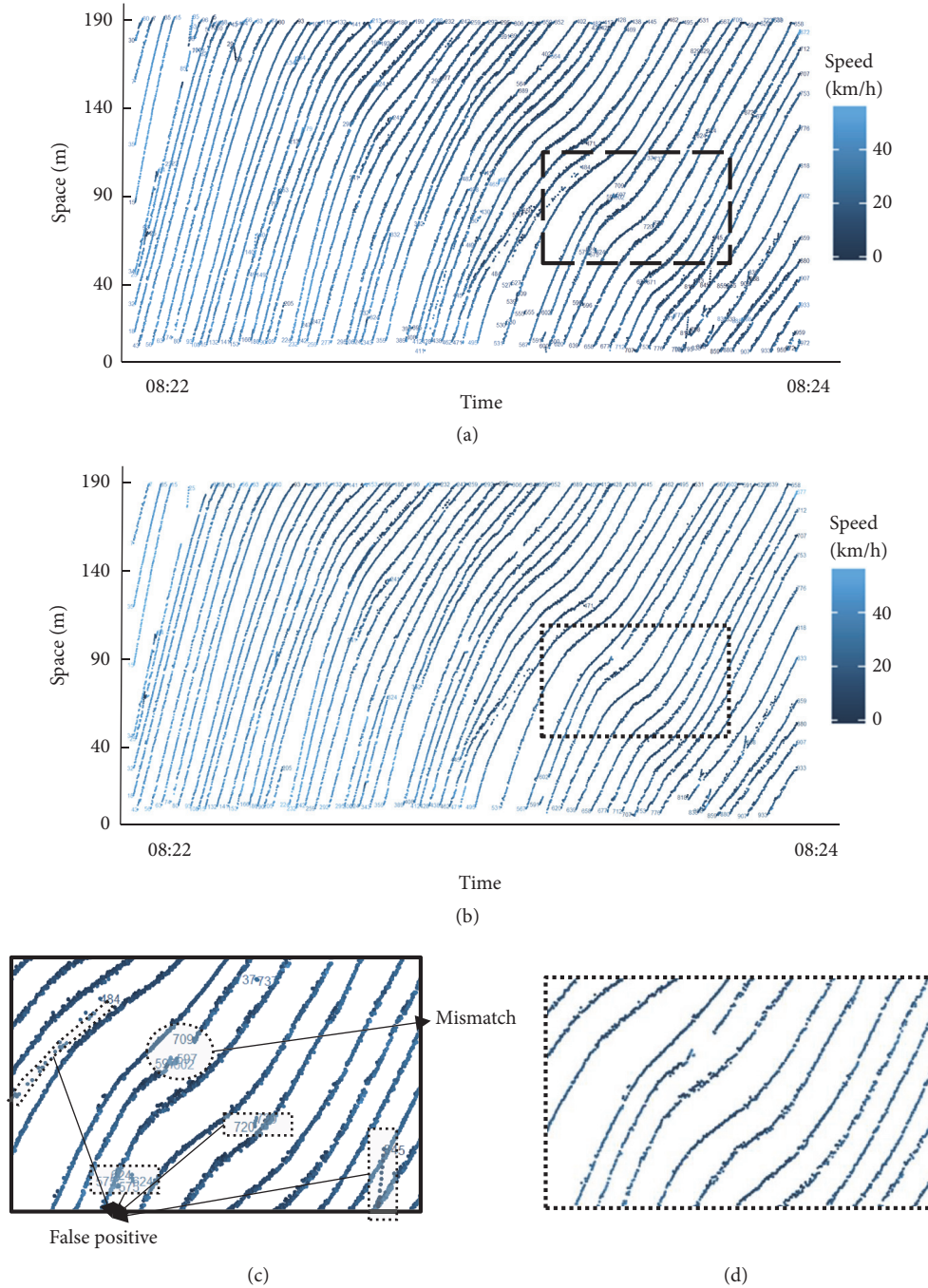


FIGURE 7: Extracted vehicle trajectories (a) before postprocessing by detection with 94.5% F-measure and (b) after postprocessing by detection with 94.5% F-measure. (c) Tracking errors including false positives and mismatches in trajectory data. (d) Results of postprocessing to remove tracking errors. The numbers in figures indicate the start and end points of each identified vehicle.

Within the test data, vehicle lane change occurred 24 times in total. The best-performed combinations of detectors and trackers, which are the ACF with KF, the SSD with KF, the SSD with KLT, and the Faster R-CNN with KLT, have 4.2% percent error (i.e., only one false positive). These results show that the lane changes can be well detected by the extracted trajectories.

5. Discussion and Conclusions

In this paper, we proposed an efficient and accurate framework for extracting vehicle trajectories recorded by UAVs to overcome the limitations reported in previous studies, i.e., low accuracy in congested traffic, lack of guideline for selecting proper detection method, and lack of detailed

performance analysis. We extended learning-based vehicle detection to tracking by combining postprocessing for vehicle trajectories in congested traffic. By evaluating the performance of the proposed framework on data from a highly congested highway, we found the Faster R-CNN to outperform the ACF in images with large objects and in images with small objects if sufficient data is provided. We note that the ACF showed comparable performance with the Faster R-CNN for small objects with a small sample size. Furthermore, we calculated and evaluated the various aggregated traffic data obtained from extracted vehicle trajectories to examine the applicability of traffic-flow analysis. The results of this study provide a practical guideline for building a framework to extract vehicle trajectories according to the given conditions.

The detection results indicate that the vehicle trajectories achieved as low as a 0.6-m location error with a MOTA of 89.9%, when using 97.4% of the detection results. False positives regarding vehicle detections, caused by road facilities, road signs, shadows, and trucks, were the main reason for errors in the extracted trajectories. In the tracking process, these not only became the false positives of trajectories but also caused mismatches with other trajectories. Their impacts are much greater in congested traffic, because there are so many interactions with other trajectories. In our postprocessing, we removed many of the false positives and mismatches, but some remained. Nevertheless, compared to the results of other studies, our study showed promising performance for extracting trajectories even in a traffic jam [13]. If we have a public dataset on congested traffic, we can readily compare the performance of our methods with those of other studies in future work.

Based on our detailed performance analysis, we discussed the usability and acceptability of the proposed framework for traffic-flow analysis. The sensitivity analysis, in which we varied the number of positive samples used to train a vehicle classifier, showed that it required only 1,000–4,000 positive samples to obtain a detector that performed well. This is a relatively small number of training samples compared with the number used in a previous study [13, 22]. The trajectories extracted based on the vehicle detections had acceptable accuracy. This indicates that the proposed framework may be used easily and efficiently in field studies without needing a vast amount of training data. Also, we showed that the MOTP, which indicates the position error of true positives, was about 0.5–0.6 m, regardless of the detection accuracy. This shows that our framework provided reliable results for true positives, even though it generated false positives, false negatives, and mismatches. In particular, the MOTP values in this study were acceptable for traffic-flow analysis. Considering that the average distance headway (i.e., the reciprocal of density) in this study was about 20 m, the location error was only about 2–3% of the headway. This value is lower than that of the GPS. When a vehicle equipped with a GPS travels, the location error ranges between 3–5 m if the vehicle had a differential GPS (DGPS), and the error is less than 1 m if the vehicle is equipped with a real-time kinetic GPS (RTK-GPS) [51]. Furthermore, we obtained more accurate vehicle trajectories by performing postprocessing to remove both the false positives and mismatches. The high precision

of macro- and microscopic traffic data obtained by extracted trajectories also supports that this framework is efficient for practical use and can also provide accurate trajectories for traffic-flow analysis. Also, our comparison results show that finding the appropriate combination of detector and tracker is a more efficient way to improve performance than finding the best detector and tracker, respectively. Therefore, it is necessary to study the detector and tracker with various characteristics to find the optimal combination for tracking vehicles.

Since the overall process of extracting vehicle trajectories is very extensive, we have not addressed all aspects of it in detail, so there is scope for improvement. As shown by the tracking accuracy based on the detection results, the tracking process is also an important factor for improving the quality of vehicle trajectories. We used simple motion constraints in congested traffic to remove the expected error of the tracking process, and this could be improved further by the use of a more sophisticated appearance or motion model, which we plan to do in future research. Details about recently developed models are described in other studies [28, 52]. As mentioned earlier, automated road extraction and camera calibration should replace our simplified preprocessing to realize a fully automated approach when traffic surveillance based on UAVs becomes practical. Also, although we presented a binary classification that distinguishes only whether or not a vehicle is present, multiclass detection and tracking could improve the tracking performance by reducing the number of false positives caused by trucks. This would also extend the relevance of this framework for future traffic dynamics research. A vision-based framework can be affected by weather and the geometric road features. Although we tested our method for two different lighting conditions, in future research, the generality of a UAV-based framework must be verified for use in harsh conditions, such as night-time, fog, and roads that are partially occluded by shadows.

Data Availability

The data used in this research are provided by the Trlab research programme conducted at the Seoul National University, Seoul, Korea. The data will be available when readers ask the authors for academic purposes.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2016RIC1B1008492).

References

- [1] D. Chen, J. Laval, Z. Zheng, and S. Ahn, "A behavioral car-following model that captures traffic oscillations," *Transportation Research Part B: Methodological*, vol. 46, no. 6, pp. 744–761, 2012.
- [2] D. Chen, S. Ahn, J. Laval, and Z. Zheng, "On the periodicity of traffic oscillations and capacity drop: The role of driver characteristics," *Transportation Research Part B: Methodological*, vol. 59, pp. 117–136, 2014.
- [3] S. Oh and H. Yeo, "Impact of stop-and-go waves and lane changes on discharge rate in recovery flow," *Transportation Research Part B: Methodological*, vol. 77, pp. 88–102, 2015.
- [4] C. Oh and T. Kim, "Estimation of rear-end crash potential using vehicle trajectory data," *Accident Analysis & Prevention*, vol. 42, no. 6, pp. 1888–1893, 2010.
- [5] J. Treiterer and J. A. Myers, "The hysteresis phenomenon in traffic flow," in *Proceedings of the 6th Symposium of Transportation and Traffic Theory*, pp. 13–38, 1974.
- [6] C. L. Azevedo, J. L. Cardoso, M. Ben-Akiva, J. P. Costeira, and M. Marques, "Automatic Vehicle Trajectory Extraction by Aerial Remote Sensing," *Procedia - Social and Behavioral Sciences*, vol. 111, pp. 849–858, 2014.
- [7] E. N. Barmounakis, E. I. Vlahogianni, and J. C. Golias, "Extracting kinematic characteristics from unmanned aerial vehicles," *Transportation Research Board 95th Annual Meeting Compendium of Papers*, vol. 16, no. 3429, 2016.
- [8] M. A. Khan, W. Ectors, T. Bellemans, D. Janssens, and G. Wets, "Unmanned aerial vehicle-based traffic analysis: Methodological framework for automated multivehicle trajectory extraction," *Transportation Research Record*, vol. 2626, pp. 25–33, 2017.
- [9] X. Cao, C. Gao, J. Lan, Y. Yuan, and P. Yan, "Ego motion guided particle filter for vehicle tracking in airborne videos," *Neurocomputing*, vol. 124, pp. 168–177, 2014.
- [10] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 4, pp. 271–288, 1998.
- [11] Y. Wan, Y. Huang, and B. Buckles, "Camera calibration and vehicle tracking: Highway traffic video analytics," *Transportation Research Part C: Emerging Technologies*, vol. 44, pp. 202–213, 2014.
- [12] "NGSIM—Next generation simulation," 2006, <http://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm>.
- [13] Y. Xu, G. Yu, X. Wu, Y. Wang, and Y. Ma, "An Enhanced Viola-Jones Vehicle Detection Method from Unmanned Aerial Vehicles Imagery," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1845–1856, 2017.
- [14] V. Alexiadis, J. Colyar, J. Halkias, R. Hranac, and G. McHale, "The next generation simulation program," *Institute of Transportation Engineers Journal*, vol. 74, no. 8, pp. 22–26, 2004.
- [15] J. Wang, X. Rui, X. Song, X. Tan, C. Wang, and V. Raghavan, "A novel approach for generating routable road maps from vehicle GPS traces," *International Journal of Geographical Information Science*, vol. 29, no. 1, pp. 69–91, 2015.
- [16] A. D. Patire, M. Wright, B. Prodhomme, and A. M. Bayen, "How much GPS data do we need?" *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 325–342, 2015.
- [17] H. Grabner, T. T. Nguyen, B. Gruber, and H. Bischof, "On-line boosting-based car detection from aerial images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 63, no. 3, pp. 382–396, 2008.
- [18] R. Ke, Z. Li, S. Kim, J. Ash, Z. Cui, and Y. Wang, "Real-Time Bidirectional Traffic Flow Parameter Estimation from Aerial Videos," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 890–901, 2017.
- [19] S. Tuermer, F. Kurz, P. Reinartz, and U. Stilla, "Airborne vehicle detection in dense urban areas using HoG features and disparity maps," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 6, pp. 2327–2337, 2013.
- [20] K. Liu and G. Mattyus, "Fast Multiclass Vehicle Detection on Aerial Images," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 9, pp. 1938–1942, 2015.
- [21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, 2012.
- [22] Y. Xu, G. Yu, Y. Wang, X. Wu, and Y. Ma, "Car detection from low-altitude UAV imagery with the faster R-CNN," *Journal of Advanced Transportation*, vol. 2017, p. 10, 2017.
- [23] T. Tang, S. Zhou, Z. Deng, H. Zou, and L. Lei, "Vehicle detection in aerial images based on region convolutional neural networks and hard negative example mining," *Sensors*, vol. 17, no. 336, 2017.
- [24] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. van den Hengel, "A survey of appearance model in visual object tracking," *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 58, 2013.
- [25] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16)*, pp. 4293–4302, 2016.
- [26] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011*, pp. 1265–1272, USA, June 2011.
- [27] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, 2006.
- [28] W. Luo, X. Zhao, and T.-K. Kim, "Multiple object tracking: A review," *Computer Vision and Pattern Recognition*, vol. 1, Article ID 1409.7618, 2014.
- [29] N. K. Kanhere and S. T. Birchfield, "A taxonomy and analysis of camera calibration methods for traffic monitoring applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 441–452, 2010.
- [30] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1511–1518, December 2001.
- [31] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 580–587, Columbus, Ohio, USA, June 2014.
- [32] R. Girshick, "Fast R-CNN," in *Proceedings of the 15th IEEE International Conference on Computer Vision (ICCV '15)*, pp. 1440–1448, December 2015.
- [33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

- [34] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.
- [35] H. Zhou, H. Kong, L. Wei, D. Creighton, and S. Nahavandi, "Efficient road detection and tracking for unmanned aerial vehicle," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 297–309, 2015.
- [36] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *Proceedings of the British Machine Vision Conference (BMVC '09)*, pp. 56–68, London, UK, September 2009.
- [37] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: an evaluation of the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.
- [38] C. Zhang and P. Viola, "Multiple-instance pruning for learning efficient cascade detectors," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pp. 1681–1688, 2008.
- [39] R. Appel, T. Fuchs, and P. Dollar, "Quickly boosting decision trees - pruning underachieving features early," in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, pp. 594–602, 2013.
- [40] G. Welch and G. Bishop, "An introduction to the Kalman filter," Tech. Rep., Department of Computer Science, University of North Carolina, Chapel Hill, NC, USA, 2004.
- [41] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society For Industrial & Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.
- [42] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1475–1490, 2004.
- [43] K. Bernardin, A. Elbs, and R. Stiefelwagen, "Multiple object tracking performance metrics and evaluation in a smart room environment," in *Proceedings of the in Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, vol. 90, 2006.
- [44] L. C. Edie, "Discussion of traffic stream measurements and definitions," in *Proceedings of the 2nd International Symposium on the Theory of Traffic Flow*, pp. 139–154, 1965.
- [45] P. Dollár, "Piotr's computer vision matlab toolbox (PMT)," 2014, <http://vision.ucsd.edu/pdollar/toolbox/doc/index.html>.
- [46] J. Huang, V. Rathod, C. Sun et al., "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17)*, pp. 7310–7319, 2017.
- [47] L. Zhang, L. Lin, X. Liang, and K. He, "Is faster R-CNN doing well for pedestrian detection?" in *Proceedings of the European Conference on Computer Vision (ECCV '16)*, pp. 443–457, 2016.
- [48] X. Zhu, C. Vondrick, D. Ramanan, and C. C. Fowlkes, "Do we need more training data or better models for object detection?" in *Proceedings of the 2012 23rd British Machine Vision Conference, BMVC 2012*, UK, September 2012.
- [49] W. Liu, D. Anguelov, D. Erhan et al., "SSD: single shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*, vol. 9905, pp. 21–37, 2016.
- [50] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 593–600, June 1994.
- [51] N. M. Thamrin, N. H. M. Arshad, R. Adnan et al., "Simultaneous localization and mapping based real-time inter-row tree tracking technique for unmanned aerial vehicle," in *Proceedings of the 2012 IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2012*, pp. 322–327, Malaysia, November 2012.
- [52] A. Milan, L. Leal-Taixe, K. Schindler, and I. Reid, "MOT16: A benchmark for multi-object tracking," *Computer Vision and Pattern Recognition*, vol. 1, Article ID 1603.00831, pp. 5397–5406, 2016.

Research Article

EBOC: Ensemble-Based Ordinal Classification in Transportation

Pelin Yıldırım,¹ Ulaş K. Birant,² and Derya Birant ²

¹Department of Software Engineering, Manisa Celal Bayar University, 45400, Manisa, Turkey

²Department of Computer Engineering, Dokuz Eylul University, 35390, Izmir, Turkey

Correspondence should be addressed to Derya Birant; derya@cs.deu.edu.tr

Received 3 October 2018; Revised 20 January 2019; Accepted 5 March 2019; Published 24 March 2019

Guest Editor: Mohammad H. Y. Moghaddam

Copyright © 2019 Pelin Yıldırım et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Learning the latent patterns of historical data in an efficient way to model the behaviour of a system is a major need for making right decisions. For this purpose, machine learning solution has already begun its promising marks in transportation as well as in many areas such as marketing, finance, education, and health. However, many classification algorithms in the literature assume that the target attribute values in the datasets are unordered, so they lose inherent order between the class values. To overcome the problem, this study proposes a novel ensemble-based ordinal classification (EBOC) approach which suggests bagging and boosting (AdaBoost algorithm) methods as a solution for ordinal classification problem in transportation sector. This article also compares the proposed EBOC approach with ordinal class classifier and traditional tree-based classification algorithms (i.e., C4.5 decision tree, RandomTree, and REPTree) in terms of accuracy. The results indicate that the proposed EBOC approach achieves better classification performance than the conventional solutions.

1. Introduction

Machine learning plays an important role in many prediction problems by constructing a model from explored dataset. The most common task in learning process is classification. *Classification* is the process of assigning an input item in a collection to predefined classes by discovering relationships among instances in the training set. Classification has a wide range of applications, such as document categorization, medical diagnosis, fraud detection, pattern recognition, sentiment analysis, risk assessment, and signal processing.

Transportation is a sector which focuses on replacement of humans, animals, and stuff from one position to another. Developments in the field of transportation reveal a need to discover associations and obtain complex and nonlinear relations underlying in a vast amount of transportation data. Because of this necessity, in recent years, machine learning techniques, especially classification algorithms, commenced to be used in transportation sector as an interdisciplinary approach. The underlying goals for these solutions are to predict traffic flow [1], classify vehicle images [2], identify different transportation modes [3], analyse traffic incident's severity [4], mitigate unfavourable environmental impacts

(i.e., to optimize energy usage [5]), develop autonomous driving system [6], and improve the productivity and efficiency of transportation systems.

The relationship between the class labels in the dataset to which the machine learning algorithms are applied influences the classification performance. In the literature, for the data of which class attribute values involve some order or a sort of ranking system, ordinal classification approach is proposed. *Ordinal classification* predicts the label of a new ordinal sample by taking ranking relation among the classes into consideration [7]. An example of an ordinal class attribute is one that has the values “large,” “medium,” and “small” for a size attribute. It is clear that there is an order among those values and that we can write $\text{large} > \text{medium} > \text{small}$. Although there are many classification studies [2–6] performed in transportation sector, to the best of our knowledge, there has been no prior detailed investigation for ordinal classification in transportation sector. Considering this drawback, the study presented in this article focuses on the application of ordinal classification algorithms on real-world transportation datasets. Some transportation studies contain an ordinal response variable; for example, the injury severity of traffic accidents can be categorized as fatal >

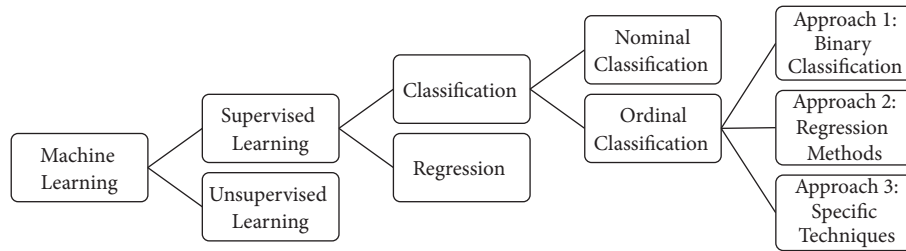


FIGURE 1: The classification methods of machine learning.

serious > slight; similarly, traffic volume can be classified as high > medium > low.

Meanwhile, ensemble learning has been recently preferred in machine learning for the classification task because of the high prediction ability it provides. *Ensemble learning* is a machine learning technique which combines a set of base learning models to get a single final prediction [8]. These learning models can be any classification algorithms, such as neural network (NN), Naive Bayes (NB), decision tree, support vector machine (SVM), regression, and k-nearest neighbour (KNN). Many studies in the literature have stated that ensemble learners improve prediction performance of individual learning algorithms. There exist various ensemble-learning methods: bagging, boosting, stacking, and voting. In this study, bagging and boosting (AdaBoost algorithm) methods are selected, due to their popularity, for the solution of ordinal classification problem in transportation sector.

The novelty and main contributions of this article are as follows: (i) it provides a brief survey of ordinal classification and ensemble learning, which has been revealed to improve prediction performance of the traditional classification algorithms, (ii) it is the first study in which the ordinal classification methods have been implemented in transportation sector, (iii) it proposes a novel ensemble-based ordinal classification (EBOC) approach for transportation, and (iv) it presents experimental studies conducted on twelve different real-world transportation datasets to demonstrate that the proposed EBOC approach shows better classification results than both ordinal class classifier and traditional tree-based classification algorithms in terms of accuracy.

The remainder of this article is structured as follows: in the following section, related literature and previous works on the subject are summarized. Section 3 gives background information about ordinal classification and ensemble learning. This section also explains utilized tree-based algorithms such as C4.5 decision tree, RandomTree, and REPTree in detail. In Section 4, the proposed ensemble-based ordinal classification (EBOC) approach for transportation sector is defined. Section 5 gives the description of transportation datasets used in this study. The application of traditional algorithms and proposed method on the transportation datasets and the experimental results of them with discussions are also presented in this section. Moreover, all obtained results were validated by three statistical methods to ensure the significance of differences among the classifiers on the datasets, including multiple comparisons (Friedman test and

Quade test) and pairwise comparisons (Wilcoxon signed rank test). Finally, the last section gives some concluding remarks and future directions.

2. Related Work

In machine learning, there are two main types of tasks: supervised learning and unsupervised learning. The classification process, which is one of the supervised learning techniques, is divided into two categories: *nominal classification* (where no order is assumed between the classes) and *ordinal classification* (where the ordinal relationship between different class labels should be taken into account). The difference between ordinal and nominal classification is not remarkable in the case of binary classification, owing to the fact that there is always an implicit order in “positive class” and “negative class.” In multiclass classification problems, standard classification algorithms for nominal classes can be applied to ordinal prediction problems by discarding the ordering information in the class attribute. However, this approach does not take advantage of the inner structure of the data and some information that can potentially improve the predictive performance of a classifier is lost since it ignores the existing natural order of the classes. The literature presents three different approaches for the ordinal classification problem: binary classification, regression methods, and specific techniques. The ordinal classification paradigm is summarized in the graph in Figure 1. In this article, a novel ordinal classification study was performed for the transportation sector.

The first approach for the ordinal classification process is to convert an ordinal classification problem into several binary classification problems. In this type of studies [9, 10], two-class classification algorithms are applied on ordinal valued datasets after transforming k class problem into a set of $k-1$ binary subproblems. For example, in the study presented in [10], the researchers proposed a novel approach which reduces the problem of classifying ordered classes to standard two-class problem. They introduced a data replication method which is then mapped into neural networks and support vector machines. In the experiments, they applied the proposed method on both artificial and real datasets for gene expression analysis. Li and Lin [11] developed a reduction framework for ordinal classification that consists of three steps: (i) extracting extended examples from training examples by using weights, (ii) training a classifier on the

extended examples with any binary classification algorithm, and (iii) constructing a ranking rule from the binary classifier.

As a second approach, regression methods [12, 13] can be used to deal with ordinal classification problem, since regression models have been thought for continuous data. In this method, categorical ordinal data is converted into continuous data scale and then regression algorithms are applied on this transformed data as a postprocessing step. However, a disadvantage of this approach is that the natural order of the class values is discarded and the inner structure of the samples is lost. In [12], two most commonly used ordinal logistic models were applied on medical ordinal data: proportional odds (PO) form of an ordinal logistic model and the forward continuation ratio (CR) ordinal logistic model. Rennie and Srebro [13] applied two general threshold-based constructions (the logistic and hinge loss) on the one million MovieLens dataset. The experimental results stated that their proposed approach shows more accurate results than traditional classification and regression models.

In the last approach, problem-specific techniques [14–18] were developed for ordinal data classification by modifying present classification algorithms. The main advantage of this approach is to retain the order among the class labels. However, some of them present some complexities in terms of implementation and training. These are complex and require nontrivial changes in the training methods such as modification of the objective function or using a threshold-based model. Keith and Meneses [14] proposed a novel technique called Barycentric Coordinates for Ordinal Classification (BCOC) which uses barycentric coordinates to represent ordinal classes geometrically. They applied their proposed method on the field of sentiment analysis and presented effective results for complex datasets. Researchers in another study [17] presented a novel heuristic rule learning approach with monotonicity constraints including two novel justifiability measures for ordinal classification. The experiments were performed to test the proposed approach and the results indicated that the novel method showed high prediction performance by guaranteeing monotone classification with low rule set increase.

Under favour of its high prediction performance, ensemble-learning techniques commenced to be preferred in ordinal classification [19–22] as well as nominal classification problems. Hechenbichler and Schliep [20] proposed an extended weighted k-nearest neighbor (*wkNN*) method for the ordinal class structure. In their study, weighted majority vote mechanism was used for the aggregation process. In the other study [21], an enhanced ensemble of support vector machines method was developed for ordinal regression. The proposed approach was implemented on the benchmark synthetic datasets and was compared with a kernel based ranking method in the experiments. Lin [23] introduced a novel threshold ensemble model and developed a reduction framework to reduce ordinal ranking to weighted binary classification by extending SVM and AdaBoost algorithms. The results of all these studies show that the ensemble methods perform well on the datasets and provide better performance than the individual methods.

Differently from existing studies, our work proposes a novel ensemble-based ordinal classification approach including bagging and boosting (AdaBoost algorithm) methods. Also, the present study is the first study in which an ordinal classification paradigm is implemented on real-world transportation datasets to model the behaviour of transportation systems.

3. Background Information

In this section, background information about ordinal classification, classification algorithms, and ensemble learning is presented to provide the context for this research.

3.1. Ordinal Classification. In most of classification problems, target attribute values which will be predicted usually assumed that they have no ordering relation between them. However, the class labels of some datasets have inherent order. For example, when predicting the price of an object, ordered class labels such as “expensive,” “normal,” and “cheap” can be used. The order among these class labels is clearly understood and denoted as “expensive” > “normal” > “cheap.” Because of this reason, classification techniques differ according to nominal and ordinal data types.

Nominal Data. The data which have no quantitative value is named nominal data. To label variables, nominal scales are utilized such as vehicle types (i.e., car, train, and bus) or document topics (i.e., science, business, and sports).

Ordinal Data. In ordinal data, the values have a natural order. In this data type, the order of values is significant such as data obtained from the use of a Likert scale.

Ordinal classification which is proposed for the prediction of ordinal target values is one of the most important classification problems in machine learning. This paradigm aims to predict the unknown class values of an attribute y that have a natural order. In the ordinal classification problem, the ordinal dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ has a set of m items with input feature space X and class attribute $y = \{c_1, c_2, \dots, c_k\}$ has k class labels with an order $c_k > c_{k-1} > \dots > c_1$, where $>$ denotes the relation of ordering. In other words, an example (x, y) is composed of an input vector $x \in X$ and an ordinal label (i.e., rank) $y \in Y = \{1, 2, \dots, K\}$. The problem is to predict the example (x, y) as rank k , where $k = 1, 2, \dots, K$.

In a decision tree-based ordinal classification, data is transformed from a k -class ordinal problem to $k-1$ binary class problems that encode the ordering of the class labels, as the first step. The ordinal dataset with classes c_1, c_2, \dots, c_k is converted into binary datasets by discriminating c_1, \dots, c_i against c_{i+1}, \dots, c_k that represents the test $c_x > i$. In other words, the upward unions of classes are considered progressively in each stage of binary datasets construction. Then a standard tree-based learning algorithm is employed on the derived binary datasets to construct $k-1$ models. As a result, each model predicts the cumulative probability of an instance of belonging to a certain class. To predict the class

label of an unseen instance, the probability for each ordinal class $P(c_x)$ is estimated by using $k-1$ models. The estimation of the probability for the first ordinal class label depends on a single classifier $1-P(\text{target} > c_1)$. The probability of the last ordinal class is given by $P(\text{target} > c_{k-1})$. In the middle of the range, the probability is calculated by a pair of classifiers $P(\text{target} > c_{i-1}) - P(\text{target} > c_i)$, where $1 < i < k$. Finally, we choose the class label with the highest probability.

The core ideas of using decision tree algorithms for ordinal classification problems can be described in three folds.

First, Frank and Hall [7] reported that decision tree-based ordinal classification algorithm resulted in a significant improvement over the standard version on 29 UCI benchmark datasets. They also experimentally confirmed that the performance gap increases with the number of classes. Most importantly, their results showed that decision tree-based ordinal classification was able to generally produce a much simpler model with better ordinal prediction accuracy.

Second, to consider the inherent order of class labels, other standard classification algorithms such as KNN [20], SVM [21], and NN require modification (nontrivial changes) in the training methods. Traditional regression techniques can also be applied on ordinal valued data due to their ability to classify interval or ratio quantity values. However, their application to truly ordinal problems is necessarily *ad hoc* [7]. In contrast, the key feature of using decision tree for ordinal classification is that there is no need to make any modification on the underlying learning algorithm. The core idea is simply to transform the ordinal classification problem into a series of binary-class problems.

Third, owing to the advantages of fast speed, high precision, and ease of understanding, decision tree algorithms are widely used in classification, as well as ordinal classification. Ordinal classification is sensitive to noise in data. Even a few noisy samples exist in the ordinal dataset; they can change the classification results of the overall system. However, in the presence of noisy and missing data, a pruned decision tree can reflect the order structure information very well with good generalization ability. The benefits of implementing decision tree-based ordinal classification are not limited to these. It builds white box models, so the trees constructed from the algorithm can be visualized and the classification results can be easily explained by Boolean logic. In addition, the most important features, which are near to the root node, are emphasized via the construction of the tree for the ordinal prediction task.

3.2. Tree-Based Classification Algorithms. In this work, three tree-based classification algorithms (C4.5 decision tree, RandomTree, and REPTree) are used as base learners to implement ordinal classification process on transportation datasets that consists of ordered class values.

3.2.1. C4.5 Decision Tree. Decision tree is one of the most successful classification algorithms, which predicts unknown class attributes using a tree structure grown with depth-first strategy. The structure of decision tree consists of

nodes, branches, and leaves that represent attributes, attribute values, and class labels, respectively. In the literature, there are several decision tree algorithms such as C4.5, ID3 (iterative dichotomiser), CART (classification and regression trees), and CHAID (chi-squared automatic interaction detector). In this study, C4.5 decision tree algorithm was used, due to its popularity, for the ordinal classification problem in transportation sector.

The first step in C4.5 decision tree algorithm is to specify root of the tree. The attribute which gives the most determinant information for the prediction process is selected for the root node. To determine the order of features in the decision tree, information gain formula is evaluated for each attribute as defined in

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \quad (1)$$

where S_v is subset of states S for attribute A with value v . The entropy indicates the impurity of a particular attribute in the dataset as defined in

$$\text{Entropy}(S) = - \sum_{i=1}^n p_i \log_2 p_i \quad (2)$$

where i is a state, p_i is the possibility of outcome being in state i for the set S , and n is the number of possible outcomes. The attribute that has the maximum information gain value is selected for the root of the tree. Likewise, all attributes in the dataset are placed to the tree according to their information gain values.

3.2.2. RandomTree. Assume a training set T with z attributes and m instances, and RandomTree is an algorithm for constructing a tree that considers x randomly chosen features ($x \leq z$) with m instances at each node [24]. While, in a standard decision tree, each node is split using the best split among all features, in a random tree, each node is split using the best among the subset of attributes randomly chosen at that node. The tree is built to its maximum depth; in other words, no pruning procedure is applied after the tree has been fully built. The algorithm can deal with both classification and regression problems. In classification task, the predicted class for a sample is determined by traversing the tree from the root node to a leaf according to the question posed about an indicator value at that node. When a leaf node is reached, its label determines the classification decision. The RandomTree algorithm does not need any accuracy estimation metric different from the other classification algorithms.

3.2.3. REPTree. The reduced error pruning tree (REPTree) algorithm builds a decision / regression tree using information gain / variance and prunes it using a simple and fast technique [25]. The pruning technique starts from the bottom level of the tree (from leaf nodes) and replaces the node with most famous class. This change is accepted only if the prediction accuracy is good. By this way, it helps to minimize

the size of decision trees by removing sections of the tree that gives little capacity to classify instances. The values of numeric attributes are sorted once. Missing values are also dealt with by splitting the corresponding instances into parts.

3.3. Ensemble Learning. Ensemble learning is a machine learning technique that combines a set of individual learners and predicts a final output [26]. First, each learner in the ensemble structure is trained separately and multiple classification models are constructed. Then, the obtained outputs from each model are compounded by a voting mechanism. The commonly used voting method available for categorical target values is major class labelling and the voting methods for numerical target values are average, weighted average, median, minimum, and maximum.

Ensemble-learning approach constructs a strong classifier from multiple individual learners and aims to improve classification performance by reducing the risk of an unfortunate selection of these learners. Many ensemble-based studies [27, 28] proved that ensemble learners give more successful results than classical individual learners.

In the literature, the ensemble methods are generally categorized under four techniques: bagging, boosting, stacking, and voting. In one of our studies, we implemented bagging approach by combining multiple neural networks which includes different parameter values for the prediction task in textile sector [26]. In the current research, bagging and boosting (AdaBoost algorithm) methods with ordinal class classifier were applied on transportation datasets. For each method (bagging and boosting), tree-based classification algorithms (C4.5 decision tree, RandomTree, and REPTree algorithms) were used as base learners separately.

3.3.1. Bagging. Bagging (bootstrap aggregating) is a commonly applied ensemble technique which constitutes training subsets by selecting random instances from original dataset using bootstrap method. Each classifier in the ensemble structure is trained by different training sets and so multiple classification models are produced. Then, a new sample is given to each model and these models predict an output. The obtained outputs are aggregated, and a single final output is achieved.

General Process of Bagging. Let $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ be a set of m items and let $y_i \in Y = \{c_1, c_2, \dots, c_k\}$ be a set of k class labels; C denotes classification algorithm and n is number of learners.

- (1) Draw m items randomly with replacement from the dataset D , so generate bootstrap samples D_1, D_2, \dots, D_n .
- (2) Each dataset D_i is trained by a base learner and multiple classification models are constructed, $M_i = C(D_i)$.
- (3) Consensus of classification models is tested to calculate out-of-bag error.
- (4) New sample x is given to classifiers as input and the outputs y_i are obtained from each model, $y_i = M_i(x)$.

- (5) The outputs of models $\{M_1, M_2, \dots, M_t\}$ are combined as in

$$M^*(x) = \arg \max_{y \in Y} \sum_{i: M_i(x)=y} 1 \quad (3)$$

3.3.2. Boosting. In boosting method, classifiers are trained consecutively to convert weak learners to strong ones. A weight value is assigned to each instance in the training set. Then, in each iteration, while the weights of misclassified samples are increased, correctly classified ones are decreased. In this way, the misclassified samples' chances of being in the training set increase. In this study, AdaBoost algorithm was utilized to implement boosting method.

AdaBoost. AdaBoost, also known as adaptive boosting, is the most popular boosting algorithm which trains learners by reweighting instances in the training set iteratively. Then, the outputs produced by each learning model are aggregated using a weighted voting mechanism.

4. Proposed Method: Ensemble-Based Ordinal Classification (EBOC)

The proposed method, named ensemble-based ordinal classification (EBOC), combines ensemble-learning paradigm including its popular methods such as AdaBoost and bagging with the traditional ordinal class classifier algorithm to improve prediction performance. The ordinal class classifier algorithm which was proposed in [7] is used as a base learner for the ensemble structure of the proposed approach. In addition, tree-based algorithms such as C4.5 decision tree, RandomTree, and REPTree are also used as a classifier in ordinal class classifier algorithm.

The first step of the ordinal class classifier algorithm is to reduce multiclass ordinal classification problem to a binary classification problem. To realize this approach, the ordinal classification problem with k different class values is converted to $k-1$ two-class classification problems. Assume that there is a dataset with four classes ($k=4$); here, the task is to find two sides: (i) class C1 against classes C2, C3, and C4; (ii) classes C1 and C2 against classes C3 and C4; and finally (iii) classes C1, C2, and C3 against class C4.

For example, suppose we have car evaluation dataset [29] which evaluates vehicles according to buying price, maintenance price, and technical characteristics such as comfort, number of doors, person capacity, luggage boot size, and safety of the car. This dataset has an ordinal class attribute with four values: unacc, acc, good, and vgood. First, four different class values of the original dataset are converted to binary values according to these rules: Classes > "unacc", Classes > "acc" and Classes > "good" class valued attribute. If we consider "Classes > 'unacc'" rule, class values higher than "unacc" are labelled as 1 and the others are labelled as 0. In this way, three different transformed datasets that contain binary class values are obtained. In the next stage, a classification algorithm (i.e., C4.5, RandomTree, or REPTree) is applied on each obtained dataset separately. Figure 2 shows the

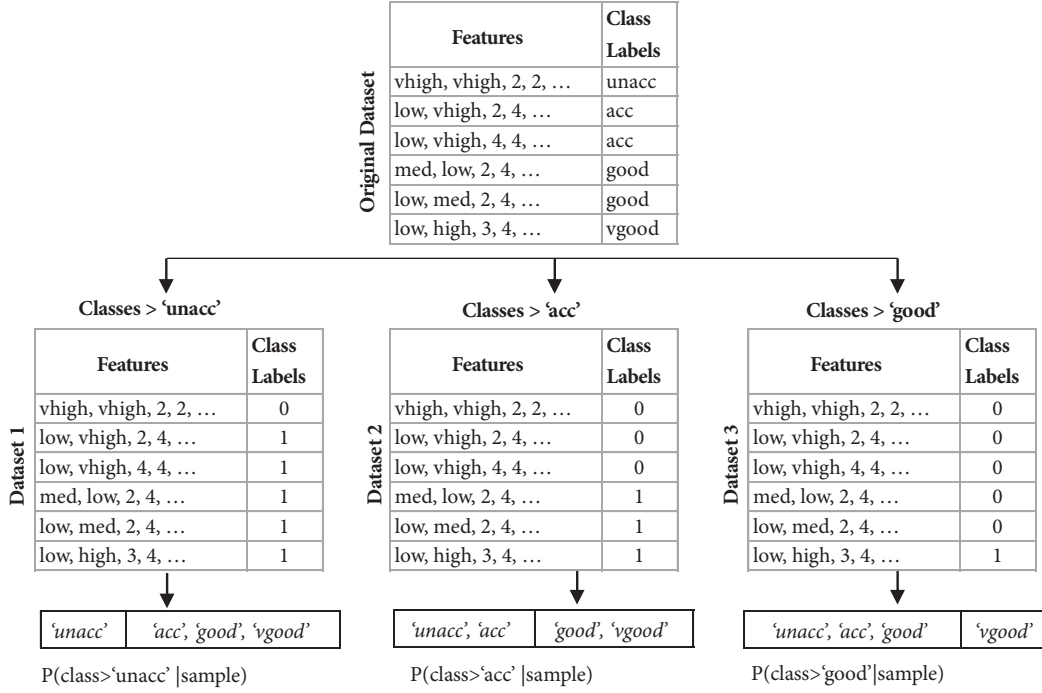


FIGURE 2: Application of ordinal class classifier algorithm on “car evaluation” dataset.

demonstration of how the ordinal class classifier algorithm works on the “car evaluation” dataset.

When predicting a new sample, the probabilities are computed for each k class value using $k-1$ binary classification models. For example, the probability of the “unacc” class value $P('unacc' | sample)$ in the sample car dataset is evaluated by $1 - P(class > 'unacc' | sample)$. Similarly, the other three ordinal class value probabilities are computed. And, finally, the class label which has the maximum probability value is assigned to the sample. In general, the probabilities of the class attribute values depend on “car evaluation” dataset which are calculated as follows:

$$\begin{aligned}
 P('unacc' | sample) &= 1 - P(class > 'unacc' | sample) \\
 P('acc' | sample) &= P(class > 'unacc' | sample) \\
 &\quad - P(class > 'acc' | sample) \\
 P('good' | sample) &= P(class > 'acc' | sample) \\
 &\quad - P(class > 'good' | sample) \\
 P('vgood' | sample) &= P(class > 'good' | sample)
 \end{aligned} \tag{4}$$

The proposed approach (EBOC) utilizes the ordinal classification algorithm as base learner for bagging and boosting ensemble methods. This novel ensemble-based approach aims to obtain successful classification results under favour of high prediction ability of ensemble-learning paradigm. The general structure of the proposed approach is presented in Figure 3. When bagging is considered, random samples are selected from the original dataset to produce multiple training sets, or when boosting is utilized, samples are selected with specified probabilities based on their weights. After that, the EBOC derives new datasets from the original dataset, each one with new binary class attribute. Each dataset is given to ordinal class classifier algorithm as an input. Then, a classification algorithm (i.e., C4.5, RandomTree, or REPTree) is applied on the datasets and multiple ensemble-based ordinal classification models are produced. Each classification model in this system gives an output label and the majority vote of these outputs is selected as a final class value.

The pseudocode of the proposed approach (EBOC) is presented in Algorithm 1. First, multiple training sets are generated according to the preferred ensemble method (bagging or boosting). Second, the algorithm converts ordinal data to binary data. After that, a tree-based classification algorithm (i.e., C4.5, RandomTree, or REPTree) is applied on the binary training sets and, by this way, various classifiers are produced. After this training phase, if boosting is chosen as ensemble method, the weight of each sample in the dataset is updated dynamically according to the performance (error rate) of the classifiers in the current iteration. To predict the class label of a new input sample, the probability for each ordinal class is estimated by using the constructed models. The algorithm chooses the class label with the highest

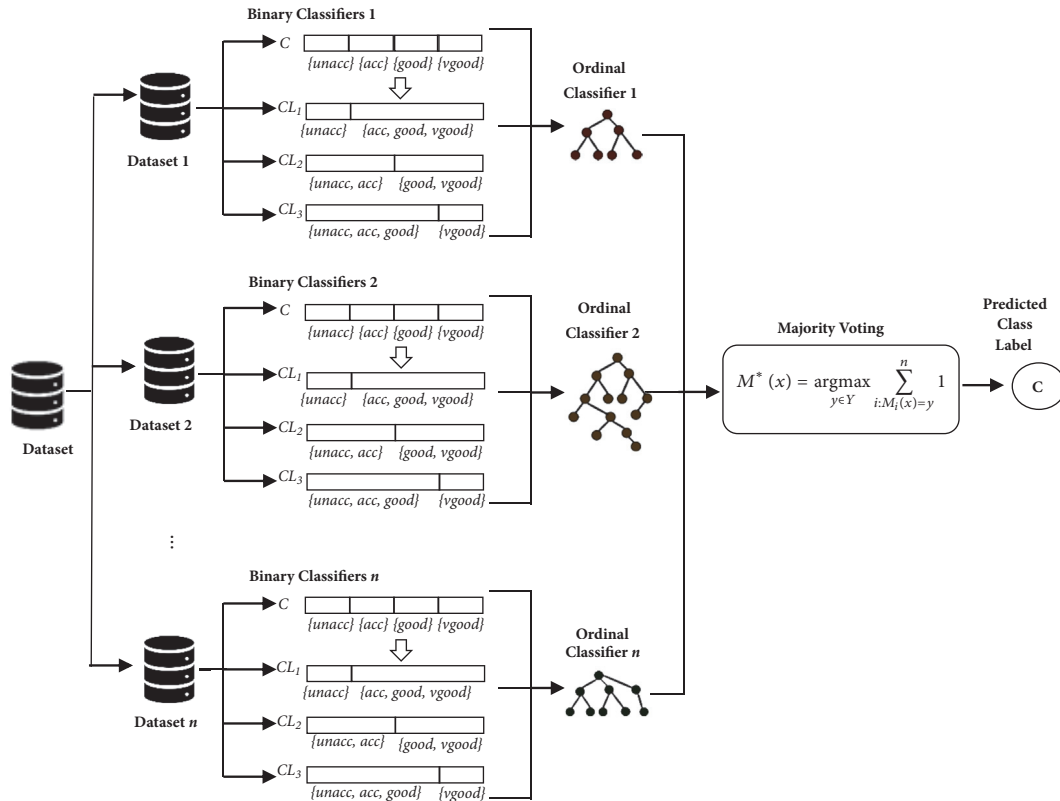


FIGURE 3: The general structure of the proposed approach (EBOC).

probability. Lastly, the majority vote of the outputs obtained from each ordinal classification model is selected as final output.

5. Experimental Studies

In the experimental studies, the proposed approach (EBOC) was implemented on twelve different benchmark transportation datasets to demonstrate its advantages over the standard nominal classification methods. The application was developed by using Weka open source data mining library [30]. Individual tree-based classification algorithms (C4.5, RandomTree, and REPTree), ordinal classification algorithm [7], and the EBOC approach were applied on the transportation datasets separately and they were compared in terms of accuracy, precision, recall, F-measure, and ROC area. In the EBOC approach, C4.5, RandomTree and REPTree algorithms were also used as a base learner in ordinal class classifiers separately. The obtained experimental results in this study are presented with the help of tables and graphs.

5.1. Dataset Description. In this study, 12 different transportation datasets which are available in several repositories for public use were selected to demonstrate the capabilities of the proposed EBOC method. The datasets were obtained from the following data archives: UCI Machine Learning Repository [31], Kaggle [32], data mill north [33], and NYC

open data [34]. The detailed descriptions about the datasets (i.e., what are they and how to use them) are given as follows.

Auto MPG. Auto MPG dataset, which was used in the American Statistical Association Exposition, is presented by the StatLib library which is maintained at Carnegie Mellon University. This dataset is utilized for the prediction of the city-cycle fuel consumption in miles per gallon of the cars according to their characteristics such as model year, the number of cylinders, horsepower, engine size (displacement), weight, and acceleration.

Automobile. The dataset was obtained from 1985 Ward's Automotive Yearbook. It consists of various automobile characteristics such as fuel type, body style, number of doors, engine size, engine location, horsepower, length, width, height, price, and insurance score. These features are used for the classification of automobiles by predicting their risk factors, six different risk ranking ranging from risky (+3) to pretty safe (-3).

Bike Sharing. The data includes the hourly count of rental bikes between years 2011 and 2012. It was collected by Capital bike share system from Washington D.C., where membership, rental, and bike return are automated via a network of kiosk locations. The dataset is presented for the prediction of hourly bike rental counts based on the environmental and seasonal settings such as weather situation (i.e., clear, cloudy, rainy, and

Algorithm EBOC: Ensemble-Based Ordinal Classification**Inputs:**

D : the ordinal dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$

m : the number of instances in the dataset D

X : input feature space, an input vector $x \in X$

Y : class attribute, an ordinal class label $y \in \{c_1, c_2, \dots, c_k\} \in Y$, with an order $c_k > c_{k-1} > \dots > c_1$

k : the number of class labels

n : ensemble size

Outputs:

M^* : an ensemble classification model

$M^*(x)$: class label of a new sample x

Begin:

for $i = 1$ **to** n **do**

if (bagging)

D_i = bootstrap samples from D

else if (boosting)

D_i = samples from D according to their weights

 // Construction of binary training sets, D_{ij}

for $j = 1$ **to** $k-1$ **do**

for $s = 1$ **to** m **do**

if ($y_s \leq c_j$)

D_{ij} .Add($x_s, 0$)

else

D_{ij} .Add($x_s, 1$)

end for

end for

 // Construction of binary classifiers, BC_{ij}

for $j = 1$ **to** $k-1$ **do**

BC_{ij} = ClassificationAlgorithm(D_{ij})

end for

if (boosting)

 update weight values

end for

// Classification of a new sample x

for $i = 1$ **to** n **do**

 // Construction of ordinal classification models, M_i

$P(c_1) = 1 - P(y > c_1)$

for $j = 2$ **to** $k-1$ **do**

$P(c_j) = P(y > c_{j-1}) - P(y > c_j)$

end for

$P(c_k) = P(y > c_{k-1})$

$M_i = \max(P)$

end for

// Majority voting

if (bagging)

$$M^*(x) = \arg \max_{y \in Y} \sum_{i: M_i(x)=y}^n 1$$

else if (boosting)

$$M^*(x) = \arg \max_{y \in Y} \sum_{i: M_i(x)=y}^n weight_i$$

End Algorithm

ALGORITHM 1: The pseudocode of the proposed EBOC approach.

snowy), temperature, humidity, wind speed, month, season, and the day of the week and year.

Car Evaluation. This dataset is useful to evaluate the quality of the cars according to various characteristics such as buying

price, maintenance price, number of doors, person capacity, size of luggage boot, and estimated safety of the car. The target attribute in the dataset indicates the overall scores of the cars as unacceptable, acceptable, good, and very good.

Car Sale Advertisements. The data was collected from private car sale advertisements in Ukraine in 2016. It was proposed for the prediction of seller's price (denominated in USD) in the advertisement according to well-known car features such as model, body type, mileage, engine volume, engine type, and drive type.

NYS Air Passenger Traffic. The data was collected monthly by the Port Authority of New York State between years 1977 and 2015. The aim is to predict the total number of domestic and international passengers for five airports (ACY, EWR, JFK, LGA, and SWF).

Road Traffic Accidents (2017). The dataset contains the records of traffic accidents across the City of Leeds, UK, reported in 2017 with the information of location, number of people and vehicles involved, the type of vehicle, road surface, weather situations, lighting conditions, and the age and sex of casualty. The aim is to predict the severity of casualty as slight, serious, or fatal.

SF Air Traffic Landings Statistics and SF Air Traffic Passenger Statistics. These two separate datasets include aircraft landings and passenger statistics of San Francisco International Airport recorded during the period from July 2005 to March 2018. These datasets are used to predict monthly landing and passenger counts, respectively.

Smart City Traffic Patterns. This dataset includes the traffic patterns of the four junctions of a city collected between 2015 and 2017. The aim is to manage the traffic of the city better and to provide input on infrastructure planning for the future to improve the efficiency of services for the citizens. To serve this purpose, the number of vehicles in the traffic is predicted to implement a robust traffic system for the city by being prepared for traffic peaks.

Statlog (Vehicle Silhouettes). The Statlog dataset contains the features of vehicle silhouettes extracted by Hierarchical Image Processing System (HIPS). The vehicle may be viewed from one of many different angles. The aim is to classify a given silhouette using a set of features extracted from the image.

Traffic Volume Counts (2012-2013). The dataset includes the hourly traffic volume counts collected by New York Metropolitan Transportation Council between years 2012 and 2013. The traffic counts were measured on various roads from one intersection to another with a specified direction (NB, SB, WB, and EB). The attribute that will be predicted in this dataset is traffic volume at 11:00 - 12:00 AM.

Before the application of the nominal and ordinal classification algorithms, generally the datasets have passed through data preprocessing steps. In this study, the ID attributes in the transportation datasets were eliminated in the data reduction step. The date attributes were split into three triplets: day, month, and year, because they provide more useful information for the prediction task. In addition, continuous class values were discretized into 3 bins using equal

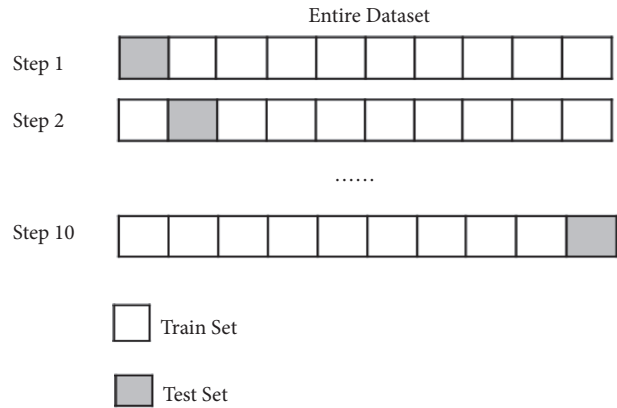


FIGURE 4: The 10-fold cross-validation process.

frequency technique, since ordinal classification algorithms require categorical ordinal data.

Basic characteristics of the investigated transportation datasets are given in Table 1. These are the number of instances, attributes and classes, target attribute, data preprocessing operations, and class distributions in each bin.

5.2. Experimental Work. In each experiment, four methods were compared on the transportation datasets: (i) individual tree-based classification algorithms, (ii) ordinal class classifier, (iii) boosting-based ordinal classification, and (iv) bagging-based ordinal classification. They were compared by using n -fold cross-validation technique selecting n as 10. In this validation technique, the entire dataset is divided into n equal size parts, $n-1$ of them is chosen for training phase of the classification model and the last part is used in testing phase. This process is repeated n times with changing parts for training and testing phase. When the cross-validation process is terminated, the accuracy values obtained in each step are averaged and a single accuracy value is produced as a success sign of the algorithm. Figure 4 represents 10-fold cross-validation process.

In this study, alternative tree-based ordinal classification algorithms were compared according to accuracy, precision, recall, F-measure, and ROC area measures. The metrics are explained with their abbreviations, formulas, and definitions in Table 2.

5.3. Experimental Results. In this study, three different experiments with three different tree-based classification algorithms (C4.5, RandomTree, and REPTree) were performed to compare the classification success of the proposed EBOC approach with the existing individual algorithms and ordinal class classifier algorithm. To evaluate the classification performances of the algorithms on a specific transportation dataset, accuracy rate values were calculated using n -fold cross-validation technique selecting n as 10.

In each experiment, one of the tree-based classification algorithms was used as base learners. For example, in the first experiment, four methods were applied and compared on 12 different transportation datasets: (i) C4.5: the individual

TABLE 1: The basic characteristics of the transportation datasets. (I: number of instances, A: number of attributes, C: number of classes).

Dataset	I	A	C	Class Distribution	Target Attribute	Data Preprocessing
Auto MPG	398	8	3	131-134-133	mpg	Removing the columns with unique values (i.e., car name)
Automobile	205	26	7	0-3-22-67-54-32-27	symboling	-
Bike Sharing	17379	13	3	5797-5783-5799	cnt	Removing the columns “instant,” “dteday,” “casual,” and “registered”
Car Evaluation	1728	7	4	1210-384-69-65	class	-
Car Sale Advertisements	9309	10	3	3112-3080-3117	price	Removing rows with zero price value
NYS Air Passenger Traffic	1584	4	3	528-528-528	total passengers	Removing the columns “Domestic” and “International Passengers”
Road Traffic Accidents (2017)	2203	13	3	1879-309-15	casualty severity	(i) Removing columns that hold reference numbers (ii) Splitting date column into day and month
SF Air Traffic Landings Statistics	21105	14	3	6941-7103-7061	landing count	-
SF Air Traffic Passenger Statistics	18398	12	3	6132-6133-6133	passenger count	-
Smart City Traffic Patterns	48120	6	3	15687-16436-15997	vehicles	(i) Removing ID column (ii) Splitting date column into day, month, and year
Statlog (Vehicle Silhouettes)	846	19	4	212-217-218-199	class	-
Traffic Volume Counts (2012-2013)	5945	31	3	1983-1989-1973	traffic volume at 11:00 -12:00AM	(i) Removing the columns with unique values (i.e., ID, Segment ID) (ii) Splitting date column into day, month, and year

TABLE 2: The detailed information about classifier performance measures. (TP: true positives, FP: false positives, TN: true negatives, and FN: false negatives).

Abbreviation	Measure	Formula	Description
Acc	Accuracy	$Acc = \frac{TP + TN}{TP + TN + FP + FN}$	The ratio of the number of correctly classified instances to the total number of records.
Prec	Precision	$Prec = \frac{TP}{TP + FP}$	The ratio of correctly classified positive instances against all positive results.
Rec	Recall	$Rec = \frac{TP}{TP + FN}$	The ratio of correct answers for a class over all the answers correctly belonging to this class.
F	F-measure	$F = \frac{2 * precision * recall}{precision + recall}$	The harmonic mean of precision and recall.
ROC Area	Receiver Operating Characteristic Area	The area under the curve which is generated to compare correctly and incorrectly classified instances.	

decision tree algorithm; (ii) *Ord.C4.5*: ordinal class classifier with C4.5; (iii) *Ada.Ord.C4.5*: AdaBoost based ordinal classification with C4.5 as base learners; and finally (iv) *Bag.Ord.C4.5*: bagging based ordinal classification with C4.5 as base learners. In the second and third experiments, *RandomTree* and *REPTree* algorithms were utilized as base learners, respectively. The default parameters of the

algorithms in Weka were used in all experiments, except ensemble size (the number of members) parameter which was set to 100. As a result of the experiments, the accuracy rates of each algorithm were evaluated. Tables 3, 4, and 5 show the comparative results of implemented techniques on twelve benchmark datasets in terms of the accuracy rates. The accuracy rates which are higher than individual classification

TABLE 3: The comparison of C4.5 based ordinal and ensemble learning methods in terms of classification accuracy.

Dataset	C4.5 (%)	Ord.C4.5 (%)	Ada.Ord.C4.5 (%)	Bag.Ord.C4.5 (%)
Auto MPG	80.40	80.90 *	83.67*	81.91 *
Automobile	81.95	66.34	84.39 *	73.66
Bike Sharing	87.75	87.72	89.29 *	89.79 *
Car Evaluation	92.36	92.19	98.73*	94.27 *
Car Sale Advertisements	83.65	81.09	83.82 *	85.00 *
NYS Air Passenger Traffic	84.91	85.42 *	86.05 *	86.81 *
Road Traffic Accidents (2017)	85.29	85.29 *	81.03	84.38
SF Air Traffic Landings Statistics	98.74	98.31	99.37*	98.69
SF Air Traffic Passenger Statistics	90.68	90.82 *	90.41	91.43 *
Smart City Traffic Patterns	85.66	85.98 *	85.07	86.94 *
Statlog (Vehicle Silhouettes)	72.46	68.91	78.13*	74.82 *
Traffic Volume Counts (2012-2013)	88.36	88.12	89.77 *	89.10 *
<i>Average</i>	86.02	84.26	87.48	86.40

TABLE 4: The comparison of RandomTree based ordinal and ensemble learning methods in terms of classification accuracy.

Dataset	Random Tree (%)	Ord.Random Tree (%)	Ada.Ord. RandomTree (%)	Bag.Ord. RandomTree (%)
Auto MPG	77.64	81.66 *	82.66 *	81.41 *
Automobile	76.59	74.63	83.41 *	84.88 *
Bike Sharing	80.07	80.56 *	86.98 *	87.73 *
Car Evaluation	83.16	85.94 *	97.22 *	95.83 *
Car Sale Advertisements	82.38	82.12	83.90 *	86.34 *
NYS Air Passenger Traffic	86.11	86.17 *	85.61	86.49 *
Road Traffic Accidents (2017)	78.76	78.44	82.21 *	84.48 *
SF Air Traffic Landings Statistics	97.38	97.43 *	98.93 *	98.83 *
SF Air Traffic Passenger Statistics	89.22	89.27 *	89.05	89.17
Smart City Traffic Patterns	82.26	82.79 *	84.72 *	85.91 *
Statlog (Vehicle Silhouettes)	70.92	65.60	76.71 *	76.00 *
Traffic Volume Counts (2012-2013)	82.96	83.45 *	89.77 *	89.69 *
<i>Average</i>	82.29	82.34	86.76	87.23

algorithm's accuracy rate in that row are marked with *. In addition, the accuracy rates which have the maximum value in each row are made bold. The experimental results show that the EBOC approaches (Ada.Ord.C4.5, Bag.Ord.C4.5, Ada.Ord.RandomTree, Bag.Ord.RandomTree, Ada.Ord.REPTree, and Bag.Ord.REPTree) generally provide higher accuracy values than individual tree-based classification algorithms. For example, on 12 datasets, Bag.Ord.RandomTree (87.23%) is significantly more accurate than RandomTree (82.29%). Similarly, both Ada.Ord.REPTree and Bag.Ord.REPTree win against plain REPTree on 11 datasets and lose on only one. The results also show that the performance gap generally increases with the number of classes. When the average accuracy values are considered in general, it is possible to say that the proposed EBOC approach has the best accuracy score in all three

experiments: Ada.Ord.C4.5 is 87.48%, Bag.Ord.RandomTree is 87.23%, and Bag.Ord.REPTree is 85.67%.

The matrices presented in Tables 6, 7, and 8 give all pairwise combinations of the tree-based algorithms with ordinal and ensemble-learning versions. Each cell in the matrix represents the number of wins, losses, and ties between the approach in that row and the approach in that column. For example, in the pairwise of C4.5 and Bag.Ord.C4.5 algorithms, 2-10-0 indicates that standard C4.5 algorithm is better than Bag.Ord.C4.5 only on 2 datasets, while Bag.Ord.C4.5 is better than the other on 10 datasets. Their accuracies are not equal in any dataset. When all matrices are examined, it is clearly seen that our proposed approach EBOC outperforms all other methods.

The graphs given in Figures 5, 6, and 7 show the average ranks of the tree-based algorithms (C4.5, RandomTree, and

TABLE 5: The comparison of REPTree based ordinal and ensemble learning methods in terms of classification accuracy.

Dataset	REPTree (%)	Ord.REPTree (%)	Ada.Ord. REPTree (%)	Bag.Ord. REPTree (%)
Auto MPG	75.38	76.63 *	81.91 *	81.16 *
Automobile	63.41	66.83 *	83.90 *	72.20 *
Bike Sharing	84.96	85.27 *	88.75 *	88.12 *
Car Evaluation	87.67	90.91 *	98.67 *	93.63 *
Car Sale Advertisements	80.70	80.42	82.52 *	82.30 *
NYS Air Passenger Traffic	84.34	84.91 *	86.74 *	87.31 *
Road Traffic Accidents (2017)	84.66	85.02 *	80.07	84.57
SF Air Traffic Landings Statistics	98.04	98.10 *	99.12 *	98.57 *
SF Air Traffic Passenger Statistics	90.39	90.49 *	90.97 *	91.25 *
Smart City Traffic Patterns	84.66	85.06 *	85.33 *	86.38 *
Statlog (Vehicle Silhouettes)	72.34	69.62	77.54 *	73.17 *
Traffic Volume Counts (2012-2013)	80.57	88.75 *	86.39 *	89.37 *
<i>Average</i>	82.26	83.50	86.83	85.67

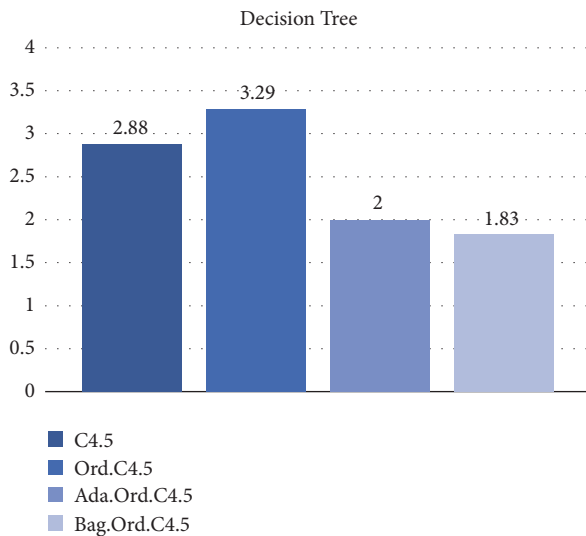


FIGURE 5: The average ranks of C4.5 algorithm with ordinal and ensemble-learning versions.

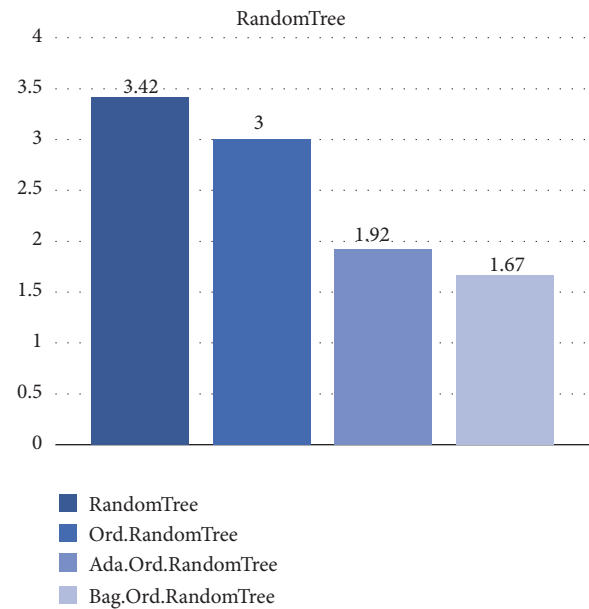


FIGURE 6: The average ranks of RandomTree algorithm with ordinal and ensemble-learning versions.

REPTree) with ordinal and ensemble-learning (AdaBoost and bagging) versions, respectively. In the ranking method, each approach used in this study is rated according to its accuracy score on the dataset. This process is performed by starting with giving rank 1 to the classifier with the highest classification accuracy and continues to increase the rank value of the classifiers until assigning rank m to the worst one of the m classifiers. In the case of tie, average of the classifiers' rankings is assigned to each classifier. Then, mean values of the ranks per classifiers on each datasets are computed as average ranks. According to the comparative results, EBOC approach with bagging method has the best performance among the others in Figures 5 and 6, because it gives the lowest rank value.

The proposed algorithms (Ada.Ord.* and Bag.Ord.*) were applied on the transportation datasets and were compared with ordinal and nominal (standard) classification algorithms in terms of accuracy, precision, recall, F-measure, and ROC area metrics. Additional measures, besides accuracy, are required to evaluate all aspects of the classifiers and they are useful for providing additional insight into their performance evaluations. Table 9 shows the average results obtained from 12 transportation datasets at each experiment. Except accuracy, the metrics listed in Table 9 give a degree ranging from 0 to 1. The algorithm with higher rate means that it is more successful than others. Among C4.5 decision

TABLE 6: The pairwise combinations of the C4.5 algorithm with ordinal and ensemble learning versions.

	Bag.Ord.C4.5	Ada.Ord.C4.5	Ord.C4.5
C4.5	3-9-0	3-9-0	7-4-1
Ord.C4.5	1-11-0	3-9-0	
Ada.Ord.C4.5	6-6-0		
Bag.Ord.C4.5			

TABLE 7: The pairwise combinations of the RandomTree algorithm with ordinal and ensemble learning versions.

	Bag.Ord. RandomTree	Ada.Ord. RandomTree	Ord.RandomTree
RandomTree	1-11-0	2-10-0	4-8-0
Ord.RandomTree	2-10-0	2-10-0	
Ada.Ord.RandomTree	5-7-0		
Bag.Ord.RandomTree			

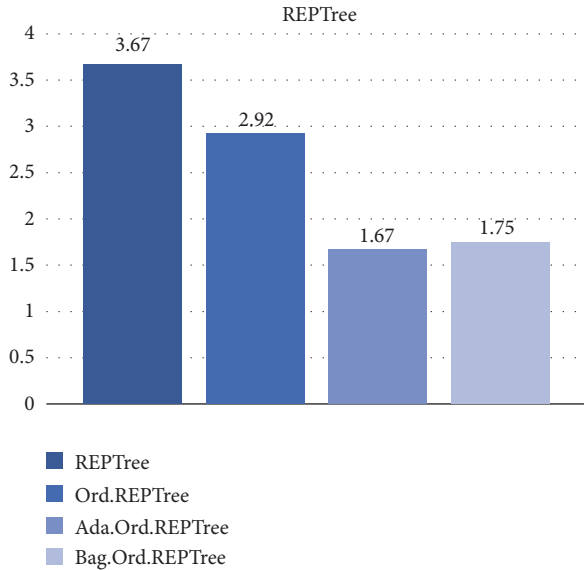


FIGURE 7: The average ranks of REPTree algorithm with ordinal and ensemble-learning versions.

tree-based algorithms, it is clearly seen that Ada.Ord.C4.5 algorithm has the best accuracy (87.467), precision (0.874), recall (0.875), and F-measure (0.874) results. While bagging has the highest values for the RandomTree algorithm, boosting is the best among REPTree-based algorithms. When the ROC area is considered, Bag.Ord.* algorithms have the highest scores according to the experimental results. As a result, it is possible to say that ensemble-based ordinal classification (EBOC) approach provides more accurate and robust classification results than traditional methods.

5.4. Statistical Significance Tests. The field of inferential statistics offers special procedures for testing the significance of differences between multiple classifiers. Although the

ensemble-based ordinal classification (EBOC) algorithms (Ada.Ord.* and Bag.Ord.*) have lower ranking value, we should statistically verify that these algorithms are significantly different from the others. For verification, we used three well-known statistical methods [35]: multiple comparisons (Friedman test and Quade test) and pairwise comparisons (Wilcoxon signed rank test).

The null hypothesis (H_0) for a statistical test is one in which provided classification models are equivalent; otherwise, the alternative hypothesis (H_1) is present when not all classifiers are equivalent.

H_0 : There are no performance differences among the classifiers on the datasets

H_1 : There are performance differences among the classifiers on the datasets

The p -value is defined as the probability, which is a decimal number between 0 and 1, and it can be expressed as a percentage (i.e., 0.1 = 10%). With a small p -value, we reject the null hypothesis (H_0), so the relationship between the classification results is significantly different.

5.4.1. Statistical Tests for Comparing Multiple Classifiers.

Multiple comparison tests (MCT) are used to establish a statistical comparison of the results reported among various classification algorithms. We performed two well-known MCT to determine whether the classifiers have significant differences or not: Friedman test and Quade test.

Friedman Test. The *Friedman test* is a nonparametric statistical test that aims to detect significant differences between the behaviors of two or more algorithms. Initially, it ranks the algorithms for each dataset separately between 1 (smallest) and 4 (largest). In case of ties, the average rank is assigned to them. After that, the Friedman test statistics (χ_r^2) is calculated according to the following:

$$\chi_r^2 = \frac{12}{d * t * (t + 1)} * \sum_{i=1}^t R_i^2 - 3 * d * (t + 1) \quad (5)$$

TABLE 8: The pairwise combinations of the REPTree algorithm with ordinal and ensemble learning versions.

	Bag.Ord.REPTree	Ada.Ord.REPTree	Ord.REPTree
REPTree	1-11-0	1-11-0	2-10-0
Ord.REPTree	1-11-0	2-10-0	
Ada.Ord.REPTree	7-5-0		
Bag.Ord.REPTree			

TABLE 9: The comparison of algorithms in terms of accuracy, precision, recall, F-measure, and ROC area.

Compared Algorithms	Accuracy (%)	Precision	Recall	F-measure	ROC Area
C4.5	86.02	0.861	0.860	0.866	0.897
Ord.C4.5	84.26	0.843	0.842	0.848	0.884
Ada.Ord.C4.5	87.48	0.874	0.875	0.874	0.933
Bag.Ord.C4.5	86.40	0.859	0.864	0.860	0.947
RandomTree	82.29	0.823	0.823	0.823	0.860
Ord.RandomTree	82.34	0.825	0.823	0.824	0.861
Ada.Ord.RandomTree	86.76	0.866	0.868	0.866	0.928
Bag.Ord.RandomTree	87.23	0.868	0.872	0.869	0.947
REPTree	82.26	0.817	0.823	0.817	0.903
Ord.REPTree	83.50	0.831	0.835	0.829	0.908
Ada.Ord.REPTree	86.83	0.868	0.868	0.867	0.925
Bag.Ord.REPTree	85.67	0.852	0.857	0.852	0.939

where d is the number of datasets, t is the number of classifiers, and R_i is the total of the ranks for the i th classifier among all datasets. The Friedman test is approximately chi-square (χ^2) distributed with $t-1$ degrees of freedom (df) and the null hypothesis is rejected, if $x_r^2 > x_{t-1,\alpha}^2$ in the corresponding significance α .

The obtained Friedman test value for the four C4.5-based classifiers is 10.525. Since the obtained value is greater than the critical value ($x_{df=3,\alpha=0.05}^2 = 7.81$), null hypothesis (H_0) is rejected, so it has been concluded that the four classifiers are significantly different. The same situation is also valid for RandomTree-based and REPTree-based algorithms, which have 15.3 and 20.1 test values, respectively. The obtained p -values for the C4.5, RandomTree, and REPTree related EBOC results (Tables 3, 4, and 5) are 0.01459, 0.00158, and 0.00016, which are extremely lower than 0.05 level of significance. Thus, it is possible to say that the differences between their performances are unlikely to occur by chance.

Quade Test. Like the Friedman test, the *Quade test* is a nonparametric test, which is used to prove that the differences among classifiers are significant. First, the performance results of each classifier are ranked within each dataset to yield $R_{i,j}$, in the same way as the Friedman test does, where d is the number of datasets and t is the number of classifiers, for $i = \{1, 2, \dots, d\}$ and $j = \{1, 2, \dots, t\}$. Then, the range for each dataset is calculated by finding the difference between the largest and the smallest observations within that dataset. The obtained rank for dataset i is denoted by Q_i . The weighted average adjusted rank for dataset i with classifier j is then

computed as $S_{i,j} = Q_i * (R_{i,j} - (t + 1)/2)$. The Quade test statistic is then given by

$$\hat{F} = \frac{(d-1)(1/d) \sum_{j=1}^t S_j^2}{\sum_{i=1}^d \sum_{j=1}^t S_{i,j}^2 - (1/d) \sum_{j=1}^t S_j^2} \quad (6)$$

where $S_j = \sum_{i=1}^d S_{i,j}$ is the sum of the weighted ranks for each classifier. The \hat{F} value is tested against the F-distribution for a given α with $df_1 = (k-1)$ and $df_2 = (b-1)(k-1)$ degrees of freedom. If $\hat{F} > F_{(k-1),(b-1)(k-1),\alpha}$ then null hypothesis is rejected. Moreover, the p -value could be computed through normal approximations.

The p -values computed through the Quade statistical tests for the C4.5, RandomTree, and REPTree related EBOC algorithms are 0.013398, 0.000018, and 0.000522, respectively. Test results strongly suggest the existence of significant differences among the considered algorithms, at the level of significance $\alpha = 0.05$. Hence, we reject the null hypothesis which indicates that all classification algorithms have the same performance. Thus, it is possible to say that at least one of them behaves differently.

5.4.2. Statistical Tests for Comparing Paired Classifiers. In addition to *multiple comparison tests*, we also conducted *pairwise comparison test* to demonstrate that the proposed algorithms (Ada.Ord.* and Bag.Ord.*) have significant differences from others when individually compared. Wilcoxon signed rank test was applied to see pairwise performance differences.

TABLE 10: Wilcoxon signed ranks test results.

Compared Algorithms	W^+	W^-	W	z -score	p -value	Significance Level
Ada.Ord.C4.5 vs C4.5	63	15	15	-1.882715	0.059739	moderate
Ada.Ord.C4.5 vs Ord.C4.5	65	13	13	-2.039608	0.041389	strong
Bag.Ord.C4.5 vs C4.5	61	17	17	-1.725822	0.084379	moderate
Bag.Ord.C4.5 vs Ord.C4.5	75	3	3	-2.824072	0.004742	very strong
Ada.Ord.RandomTree vs RandomTree	75	3	3	-2.824072	0.004742	very strong
Ada.Ord.RandomTree vs Ord.RandomTree	75	3	3	-2.824072	0.004742	very strong
Bag.Ord.RandomTree vs RandomTree	77	1	1	-2.980965	0.002873	very strong
Bag.Ord.RandomTree vs vs Ord.RandomTree	75	3	3	-2.824072	0.004742	very strong
Ada.Ord. REPTree vs REPTree	71	7	7	-2.510287	0.012063	strong
Ada.Ord. REPTree vs Ord.REPTree	64	14	14	-1.961161	0.049860	strong
Bag.Ord.REPTree vs REPTree	77	1	1	-2.980965	0.002873	very strong
Bag.Ord.REPTree vs Ord.REPTree	77	1	1	-2.980965	0.002873	very strong
<i>Average</i>	71.25	6.75	6.75	-2.602996	0.022918	

Wilcoxon Signed Rank Test. This statistical test consists of the following steps to check the validity of the null hypotheses: (i) calculate performance differences between two algorithms for each dataset, (ii) order the differences according to their absolute values from smallest to largest, (iii) rank the absolute value of differences, starting with the smallest as 1 and assigning average ranks in case of ties, (iv) calculate W^+ and W^- by summing the ranks of the positive and negative differences separately, (v) find W as the smaller of the sums, $W = \min(W^+, W^-)$, (vi) calculate z -score as $z = W/\sigma_w$ and find p -value from the distribution table, and (vii) determine the significance level according to the computed p -value [36] and reject the null hypothesis if p -value is less than a certain risk threshold (i.e., 0.1 = 10%):

Significance level

$$= \begin{cases} p > 0.1 & \text{weak or none} \\ 0.05 < p \leq 0.1 & \text{moderate} \\ 0.01 < p \leq 0.05 & \text{strong} \\ p \leq 0.01 & \text{very strong} \end{cases} \quad (7)$$

Table 10 demonstrates the W , z -score, and p -values computed through Wilcoxon signed rank tests. Based on the reported p -values, most of the tests strongly or very strongly prove the existence of significant differences among the considered algorithms. As the table states, for each test, the ensemble-based ordered classification algorithms (Bag.Ord.* and Ada.Ord.*) always show a significant improvement over the traditional versions, with a level of significance $\alpha=0.1$. In general, it can be observed that the average p -value of proposed Bag.Ord.* variant algorithms (0.0171) is a little less than Ada.Ord.* variants (0.0288).

Based on all statistical tests (Friedman, Quade, and Wilcoxon signed rank test), we can safely reject the null hypothesis (i.e., there are no performance differences among the classifiers on the datasets), since the p -values computed through the statistics are less than the critical value (0.05). Thus, the proposed approach (EBOC) has a statistically

significant effect on the response at the 95.0% confidence level.

6. Conclusions and Future Works

As a result of developments in the field of transportation, enormous amounts of raw data are generated every day. This situation creates the potential to discover knowledge patterns or rules from it and to model the behaviour of a transportation system using machine learning techniques. To serve the purpose, this study focuses on the application of ordinal classification algorithms on real-world transportation datasets. It proposes a novel ensemble-based ordinal classification (EBOC) approach. This approach converts the original ordinal class problem into a series of binary class problems and use ensemble-learning paradigm (boosting and bagging) at the same time. To the best of our knowledge, this is the first study in which ordinal classification methods and our proposed approach were applied on transportation sector. In the experimental studies, the proposed model with the tree-based learners (C4.5, RandomTree, and REPTree) was implemented on twelve benchmark transportation datasets that are available for public use. The proposed EBOC method was compared with ordinal class classifier and traditional tree-based classification algorithms in terms of accuracy, precision, recall, f-measure, and ROC area. The results indicate that the EBOC approach provides more accurate classification results than them. Moreover, statistical test methods (Friedman, Quade, and Wilcoxon signed rank tests) were used to prove that the classification accuracies obtained from the proposed algorithms (Ada.Ord.* and Bag.Ord.*) are significantly different from the traditional methods. Therefore, the proposed EBOC method can assist in making right decisions in transportation. Our findings demonstrate that the improvement in performance is a result of exploiting ordering information and applying ensemble strategy at the same time.

As future work, different types of ensemble-based ordinal classifiers can be developed using different ensemble methods

such as stacking and voting, and using different classification algorithms such as Naive Bayes, k-nearest neighbour, neural network, and support vector machine. In addition, ensemble clustering models can be improved to cluster transportation data. Furthermore, a comparative study which implements ensemble-learning and deep learning paradigms can be performed in transportation fields.

Data Availability

The “Auto MPG,” “Automobile,” “Bike Sharing,” “Car Evaluation,” and “Statlog (Vehicle Silhouettes)” datasets used to support the findings of this study are freely available at <https://archive.ics.uci.edu/ml/datasets>. The “Car Sale Advertisements,” “NYS Air Passenger Traffic,” “Smart City Traffic Patterns,” “SF Air Traffic Landings Statistics,” and “SF Air Traffic Passenger Statistics” datasets used to support the findings of this study are freely available at <https://www.kaggle.com/datasets>. The “Road Traffic Accidents (2017)” dataset used to support the findings of this study is freely available at <https://datamillnorth.org/dataset>. The “Traffic Volume Counts (2012-2013)” dataset used to support the findings of this study is freely available at <https://opendata.cityofnewyork.us/data>.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.



References

- [1] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, “Traffic flow prediction with big data: a deep learning approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.
- [2] X. Wen, L. Shao, Y. Xue, and W. Fang, “A rapid learning algorithm for vehicle classification,” *Information Sciences*, vol. 295, pp. 395–406, 2015.
- [3] S. Ballı and E. A. Sağbaşı, “Diagnosis of transportation modes on mobile phone using logistic regression classification,” *IET Software*, vol. 12, no. 2, pp. 142–151, 2018.
- [4] H. Nguyen, C. Cai, and F. Chen, “Automatic classification of traffic incident’s severity using machine learning approaches,” *IET Intelligent Transport Systems*, vol. 11, no. 10, pp. 615–623, 2017.
- [5] G. Kedar-Dongarkar and M. Das, “Driver classification for optimization of energy usage in a vehicle,” *Procedia Computer Science*, vol. 8, pp. 388–393, 2012.
- [6] N. Deepika and V. V. Sajith Variyar, “Obstacle classification and detection for vision based navigation for autonomous driving,” in *Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*, pp. 2092–2097, Udupi, India, September 2017.
- [7] E. Frank and M. Hall, “A simple approach to ordinal classification,” in *Proceedings of the European Conference on Machine Learning (ECML)*, vol. 2167 of *Lecture Notes in Computer Science*, pp. 145–156, Springer, 2001.
- [8] E. Alpaydin, *Introduction to Machine Learning*, The MIT Press, 3rd edition, 2014.
- [9] S. Destercke and G. Yang, “Cautious ordinal classification by binary decomposition,” in *Proceedings of the Machine Learning and Knowledge Discovery in Databases - European Conference ECML/PKDD*, pp. 323–337, Nancy, France, 2014.
- [10] J. S. Cardoso and J. F. Pinto da Costa, “Learning to classify ordinal data: the data replication method,” *Journal of Machine Learning Research (JMLR)*, vol. 8, pp. 1393–1429, 2007.
- [11] L. Li and H. T. Lin, “Ordinal regression by extended binary classification,” in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pp. 865–872, Canada, 2006.
- [12] F. E. Harrell, “Ordinal logistic regression,” in *Regression Modeling Strategies*, Springer Series in Statistics, pp. 311–325, Springer International Publishing, Cham, Switzerland, 2015.
- [13] J. D. Rennie and N. Srebro, “Loss functions for preference levels: regression with discrete ordered labels,” in *Proceedings of the IJCAI Multidisciplinary Workshop Adv. Preference Handling*, pp. 180–186, 2005.
- [14] B. Keith, “Barycentric coordinates for ordinal sentiment classification,” in *Proceedings of the 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, Halifax, Canada, 2017.
- [15] E. Fernández, J. R. Figueira, J. Navarro, and B. Roy, “ELECTRE TRI-nB: A new multiple criteria ordinal classification method,” *European Journal of Operational Research*, vol. 263, no. 1, pp. 214–224, 2017.
- [16] M. M. Stenina, M. P. Kuznetsov, and V. V. Strijov, “Ordinal classification using Pareto fronts,” *Expert Systems with Applications*, vol. 42, no. 14, pp. 5947–5953, 2015.
- [17] W. Verbeke, D. Martens, and B. Baesens, “RULEM: A novel heuristic rule learning approach for ordinal classification with monotonicity constraints,” *Applied Soft Computing*, vol. 60, pp. 858–873, 2017.
- [18] W. Kotlowski and R. Slowinski, “On nonparametric ordinal classification with monotonicity constraints,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2576–2589, 2013.
- [19] K. Dembczynski, W. Kotlowski, and R. Slowinski, “Ordinal classification with decision rules,” in *Proceedings of the Third International Conference on Mining Complex Data (MCD’07)*, pp. 169–181, Warsaw, Poland, 2007.
- [20] K. Hechenbichler and K. Schliep, “Weighted k-nearest-neighbor techniques and ordinal classification,” *Collaborative Research Center*, vol. 399, pp. 1–16, 2004.
- [21] W. Waegeman and L. Boullart, “An ensemble of weighted support vector machines for ordinal regression,” *International Journal of Computer and Information Engineering*, vol. 1, no. 12, pp. 599–603, 2007.
- [22] K. Dembczynski, W. Kotlowski, and R. Slowinski, “Ensemble of decision rules for ordinal classification with monotonicity constraints,” in *Proceedings of the International Conference on Rough Sets and Knowledge Technology*, vol. 5009 of *Lecture Notes in Computer Science*, pp. 260–267, 2008.
- [23] H. T. Lin, *From Ordinal Ranking to Binary Classification [PhD thesis]*, California Institute of Technology, 2008.
- [24] C. K. A. Cuartas, A. J. P. Anzola, and B. G. M. Tarazona, “Classification methodology of research topics based in decision trees: J48 and randomtree,” *International Journal of Applied Engineering Research*, vol. 10, no. 8, pp. 19413–19424, 2015.
- [25] C. Parimala and R. Porkodi, “Classification algorithms in data mining: a survey,” in *Proceedings of the International Journal of Scientific Research in Computer Science*, vol. 3, pp. 349–355, 2018.

- [26] P. Yildirim, D. Birant, and T. Alpyildiz, "Improving prediction performance using ensemble neural networks in textile sector," in *Proceedings of the 2017 International Conference on Computer Science and Engineering (UBMK)*, pp. 639–644, Antalya, Turkey, October 2017.
- [27] H. Yu and J. Ni, "An improved ensemble learning method for classifying high-dimensional and imbalanced biomedicine data," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 4, pp. 657–666, 2014.
- [28] D. Che, Q. Liu, K. Rasheed, and X. Tao, "Decision tree and ensemble learning algorithms with their applications in bioinformatics," in *Software Tools and Algorithms for Biological Systems*, H. R. Arabnia and Q-N. Tran, Eds., vol. 696, pp. 191–199, Springer, 2011.
- [29] "UCI Machine Learning Repository: Car Evaluation Data Set," Archive.ics.uci.edu, 2018, <https://archive.ics.uci.edu/ml/datasets/car+evaluation>.
- [30] "Weka 3 - Data Mining with Open Source Machine Learning Software in Java," Cs.waikato.ac.nz, 2018, <https://www.cs.waikato.ac.nz/ml/weka/>.
- [31] "UCI Machine Learning Repository," Archive.ics.uci.edu, 2018, <https://archive.ics.uci.edu/ml/index.php>.
- [32] "Datasets — Kaggle," Kaggle.com, 2018, <https://www.kaggle.com/datasets>.
- [33] "Data Mill North," Datamillnorth.org, 2018, <https://datamillnorth.org/dataset>.
- [34] "N.: City of New York, "NYC Open Data"," Open-data.cityofnewyork.us, 2018, <https://opendata.cityofnewyork.us/>.
- [35] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.
- [36] N. A. Weiss, *Introductory Statistics*, Addison-Wesley, 9th edition, 2012.

Research Article

Spatiotemporal Traffic Flow Prediction with KNN and LSTM

Xianglong Luo ^{1,2}, Danyang Li,² Yu Yang,² and Shengrui Zhang ¹

¹School of Highway, Chang'an University, Xi'an 710064, China

²School of Information Engineering, Chang'an University, Xi'an 710064, China

Correspondence should be addressed to Xianglong Luo; xlluo@chd.edu.cn

Received 12 September 2018; Revised 15 January 2019; Accepted 1 February 2019; Published 27 February 2019

Guest Editor: Yasser Hassan

Copyright © 2019 Xianglong Luo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The traffic flow prediction is becoming increasingly crucial in Intelligent Transportation Systems. Accurate prediction result is the precondition of traffic guidance, management, and control. To improve the prediction accuracy, a spatiotemporal traffic flow prediction method is proposed combined with k-nearest neighbor (KNN) and long short-term memory network (LSTM), which is called KNN-LSTM model in this paper. KNN is used to select mostly related neighboring stations with the test station and capture spatial features of traffic flow. LSTM is utilized to mine temporal variability of traffic flow, and a two-layer LSTM network is applied to predict traffic flow respectively in selected stations. The final prediction results are obtained by result-level fusion with rank-exponent weighting method. The prediction performance is evaluated with real-time traffic flow data provided by the Transportation Research Data Lab (TDRL) at the University of Minnesota Duluth (UMD) Data Center. Experimental results indicate that the proposed model can achieve a better performance compared with well-known prediction models including autoregressive integrated moving average (ARIMA), support vector regression (SVR), wavelet neural network (WNN), deep belief networks combined with support vector regression (DBN-SVR), and LSTM models, and the proposed model can achieve on average 12.59% accuracy improvement.

1. Introduction

The accurate prediction of future traffic conditions (e.g., traffic flow, travel speed and travel time) is crucial requirement for Intelligent Transportation Systems (ITS), which can help administrators take adequate preventive measures against congestion and travelers take better-informed decisions. Among different applications in ITS, traffic flow prediction has attracted significant attention over the past few decades. It is still a challenging topic for transportation researchers.

Due to the stochastic characteristics of traffic flow, accurate traffic prediction is not a straightforward task. In order to deal with this issue, many techniques are deployed for modeling the evolution of the traffic circulation. These existing prediction schemes are classified roughly into three categories: parametric methods, nonparametric methods, and hybrid methods. The parametric methods include Autoregressive Integrated Moving Average method (ARIMA) [1], Seasonal Autoregressive Integrated Moving Average method (SARIMA) [2, 3], and Kalman filter [4, 5]. The parametric methods are widely used in traffic flow

prediction, but these methods are sensitive to the traffic data for different situations. The nonparametric methods include artificial neural networks (ANNs) [6–9], k-nearest neighbor (KNN) [10–14], support vector regression (SVR) [15, 16], and Bayesian model [17, 18]. Compared to the parametric methods, nonparametric methods are more effective in prediction performance. Even so, nonparametric methods require large amount of historical data and training process. The hybrid methods are mainly combining the parametric approach with nonparametric approach [19–29]. Although the prediction accuracy of nonparametric methods and hybrid methods is superior to parametric methods, all these methods mainly considered the data closed to the prediction station, which could not fully reveal the spatiotemporal characteristics of traffic flow data. Vlahogianni et al. [30] summarized existing traffic flow prediction algorithms from 2004 to 2013. Suhas et al. [31] followed a systematic study to aggregate previous works on traffic prediction, highlight marked changes in trends, and provide research direction for future work. Lana et al. [32] summarized the latest technical achievements in traffic prediction field, along with an insightful update of

the main technical challenges that remain unsolved. The readers interested in details of models that applied in traffic prediction field could refer to review reference paper.

With the widespread traditional traffic sensors and new emerging traffic sensor technologies, tremendous traffic sensors have been deployed on the existing road network, and a large volume of historical traffic data at very high spatial and temporal resolutions has become available. It is a challenge to deal with these big traffic data with conventional parametric methods. But for nonparametric methods, most are shallow in architecture, which cannot penetrate the deep correlation and implicit traffic information. Recently, deep learning, an emerging machine learning method, has drawn a lot of attention from both academic and industrial filed. Traffic flow prediction based on deep learning methods has become a new trend.

Huang et al. [33] proposed a deep architecture for traffic flow prediction with deep belief networks (DBN) and multitask learning. Lv et al. [34] used a stacked autoencoder (SAE) model to learn generic traffic flow features. Duan et al. [35] evaluated the performance of the SAE model for traffic flow prediction at daytime and nighttime. Soua et al. [36] proposed a DBN based approach to predict traffic flow with historical traffic flow, weather data, and event-based data. An extension of Dempster-Shafer evidence theory was used to fuse traffic prediction beliefs coming from streams of data and event-based data models. Koesdwiady et al. [37] predicted the traffic flow and weather data separately using DBN. The result of each prediction was merged using data fusing techniques. Yang et al. [38] proposed a stacked autoencoder Levenberg-Marquardt model to improve prediction accuracy. The Taguchi method was developed to optimize the model structure. Zhou et al. [39] introduced an adaptive boosting scheme for the stacked autoencoder network. Polson and Sokolov [40] developed a deep learning model to predict traffic flows. An architecture was proposed combined with a linear model that was fitted using regularization and a sequence of tanh layers. Zhang and Huang [41] employed the genetic algorithm to find the optimal hyperparameters of DBN models. In recent years, recurrent neural network (RNN) was more practical in comparison with other deep learning structures for processing sequential data. Ma et al. [42] utilized a deep Restricted Boltzmann Machine and RNN architecture to model and predict traffic congestion. However, the traditional RNNs face problems of vanishing gradients and exploding gradients. To solve this problem, a long short-term memory network (LSTM) was proposed. Because LSTM can automatically calculate the optimal time lags and capture the features of time series with longer time span, a better performance can be achieved with LSTM model in traffic flow prediction. LSTM was developed to capture the long-term temporal dependency for traffic sequences by Ma et al. [43]. Shao and Soong [44] utilized LSTM to learn more abstract representations in the nonlinear traffic flow data. In recent years, LSTM was very successful in traffic flow prediction, but the spatiotemporal characteristics of traffic flow were hardly considered. Zhao et al. [45] proposed an origin destination correlation matrix to represent the correlations of different links within the road network, and

a cascade connected LSTM was used to predict traffic flow. However, the architecture of proposed LSTM model was overly complicated, making comprehension difficult. The prediction results were not very stable and reliable in different observation points.

In this paper, inspired by the successful application of LSTM in traffic flow prediction, the high spatiotemporal correlation characteristics of traffic flow data are considered in order to improve prediction performance. A hybrid traffic flow prediction methodology is proposed based on KNN and LSTM. KNN is used to choose mostly related neighboring stations with the test station. A multilayer LSTM is applied to predict traffic flow in all selected stations. The final prediction results are obtained by weighting the prediction values in all selected stations. The weights are assigned by adjusting the weight dispersion measure with rank-exponent method. The experiment results show that proposed method has better performance on accuracy compared with most existing traffic prediction methods.

The main contributions of this paper are summarized as follows.

(1) A hybrid traffic flow prediction methodology is proposed combined KNN with LSTM, which utilizes the spatiotemporal characteristics of traffic flow data. Experimental results demonstrate that proposed approach can achieve on average 12.59% accuracy improvement compared to ARIMA, SVR, WNN, DBN-SVR, and LSTM models.

(2) The prediction results are obtained by weighting the prediction values in all selected stations by adjusting the weight dispersion measure with rank-exponent method. Different from the traditional weighting method, the proposed method highlights the importance of the highly relevant stations to the prediction result.

(3) From classical understanding, closer stations from the prediction station have more correlation than those further stations. In fact, some further stations have also correlation with the prediction station. However, it is consistent with the general fact that the traffic flow in the upstream and downstream has great influence on the prediction result in the traffic flow prediction.

The rest of this paper is organized as follows. Section 2 gives details on a hybrid traffic prediction method based on KNN and LSTM. In Section 3, the dataset used is introduced for the numerical experiments. The results and performance evaluation are also presented. Finally, the conclusions and the future research are given in Section 4.

2. Methodology

2.1. LSTM Network. RNN is a neural network that is specialized for processing time sequences. Different from conventional networks, RNN allows a “memory” of previous inputs to persist in the network internal state, which can then be used to influence the network output. Traditional RNN exhibits a superior capability of modeling nonlinear time sequence problems, such as speech recognition, language modeling, and image captioning. However, traditional RNN is not able to train the time sequence with long time lags. To overcome the disadvantages of traditional RNN, LSTM is

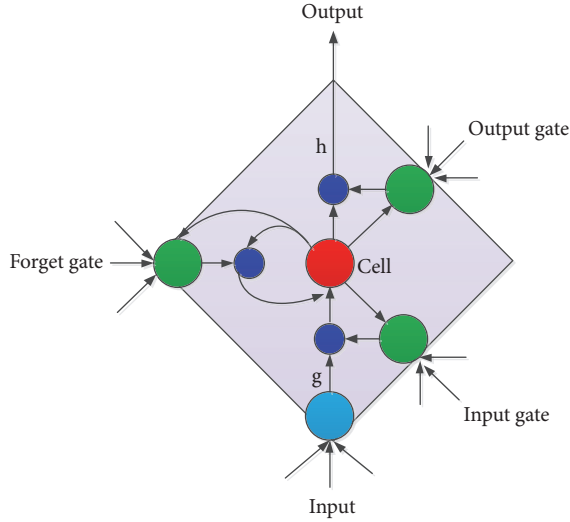


FIGURE 1: LSTM memory block with one cell.

proposed. LSTM is a special kind of RNN, designed to learn long term dependencies. The LSTM architecture consists of a set of memory blocks. Each block contains one or more self-connected memory cells and three gates, namely, input gate, forget gate, and output gate. The typical structure of LSTM memory block with one cell is in Figure 1. Input gate takes a new input from outside and process newly coming data. Forget gate decides when to forget the previous state and thus selects the optimal time lag for the input sequence. Output gate takes all results calculated and generates output for LSTM cell.

Let us denote the input time series as $X = [x_1, x_2, \dots, x_T]$, and T is input sequence length. N is the number of inputs, H is the number of cells in the hidden layer, and C is the number of memory cells. The subscripts i , f , and o refer to the input gate, forget gate, and output gate, respectively. ω_{ij} is the weight of the connection from i unit to unit j . a_j^t is the network input to some unit j at time t , and b_j^t is the value after activation function in the same unit. s_c^t is the state of cell at time t . σ is the activation function of the gates, and g and h are, respectively, the cell input and output activation functions. The LSTM model can be conducted by the following equations.

Input Gates

$$a_i^t = \sum_{n=1}^N \omega_{ni} x_n^t + \sum_{h=1}^H \omega_{hi} b_h^{t-1} + \sum_{c=1}^C \omega_{ci} s_c^{t-1} \quad (1)$$

$$b_i^t = \sigma(a_i^t) \quad (2)$$

Forget Gates

$$a_f^t = \sum_{n=1}^N \omega_{nf} x_n^t + \sum_{h=1}^H \omega_{hf} b_h^{t-1} + \sum_{c=1}^C \omega_{cf} s_c^{t-1} \quad (3)$$

$$b_f^t = \sigma(a_f^t) \quad (4)$$

Cells

$$a_c^t = \sum_{n=1}^N \omega_{nc} x_n^t + \sum_{h=1}^H \omega_{hc} b_h^{t-1} \quad (5)$$

$$s_c^t = b_f^t s_c^{t-1} + b_i^t g(a_c^t) \quad (6)$$

Output Gates

$$a_o^t = \sum_{n=1}^N \omega_{no} x_n^t + \sum_{h=1}^H \omega_{ho} b_h^{t-1} + \sum_{c=1}^C \omega_{co} s_c^{t-1} \quad (7)$$

$$b_o^t = \sigma(a_o^t) \quad (8)$$

Cell Outputs

$$b_c^t = b_o^t h(s_c^t) \quad (9)$$

By the function of the different gates, LSTM network has the capability of processing arbitrary time lags for time sequence with long dependency.

2.2. KNN Algorithm. KNN algorithm is a nonparametric method used for classification and regression. The KNN method makes use of a database to search for data that are similar to the current data. These found data are called the nearest neighbors of the current data. In this paper, KNN is used to select mostly related neighboring stations with the test station. Suppose there are M stations in the road network. $X_o(t) = [x_o(t), x_o(t-1), \dots, x_o(t-T)]$ is the historical traffic flow data in test station, and T is the sample data length. $X_m(t) = [x_m(t), x_m(t-1), \dots, x_m(t-T)]$ ($m = 1, 2, \dots, M-1$) is the historical traffic flow data in the m^{th} station, which is different from the test station. The Euclidean distance [see (10)] is used to measure the correlation between the test station with others.

$$d_m = \|X_o(t) - X_m(t)\|_2 = \sqrt{\sum_j (x_o(j) - x_m(j))^2} \quad (10)$$

According to the calculated distance, a total of K -nearest neighbors are found, and K stations are selected as mostly related stations with the test station.

2.3. Proposed Method. Different from the conventional LSTM network, KNN algorithm is used to select spatiotemporal correlation stations with the test station at first. A two-layer LSTM network is applied to predict traffic flow, respectively, in selected stations. The final prediction results in test station are obtained by weighting with rank-exponent method. At time t , the traffic flow data in the test station is denoted as $X_o(t) = [x_o(t), x_o(t-1), \dots, x_o(t-T)]$. The traffic flow data for $M-1$ stations near the test station is denoted as

$$X_{M-1}(t) = \begin{bmatrix} x_1(t), x_1(t-1), \dots, x_1(t-T) \\ x_2(t), x_2(t-1), \dots, x_2(t-T) \\ \vdots \quad \vdots \quad \dots \quad \vdots \\ x_{M-1}(t), x_{M-1}(t-1), \dots, x_{M-1}(t-T) \end{bmatrix} \quad (11)$$

$X_{ci}(t)$ ($i = 1, 2, \dots, K$) is the station selected by KNN. The prediction traffic flow in the selected stations and test station can be calculated as

$$\widehat{X}_{ci}(t+1) = W_{ho} b_c + b \quad (i = 1, 2, \dots, K) \quad (12)$$

where W_{ho} is the weight matrix between the hidden layer and output layer and b is bias term. The final prediction results in test station are obtained by weighting according to (12).

$$\widehat{y}_o(t+1) = \sum_{i=1}^K W_i \widehat{X}_{ci} \quad (13)$$

where W_i is the weight coefficient. The Rank-Exponent method of weights is used in this paper. Rank-Exponent method can provide some degree of flexibility by adjusting the weight dispersion measure as shown in (13). The value of z is set to 2 as indicated by the authors [46].

$$W_i = \frac{(K - r_i + 1)^z}{\sum_{i=1}^K (K - r_i + 1)^z} \quad (14)$$

where r_i is the rank of the i^{th} selected station, K is the total number of selected stations, and z is weight dispersion measure.

The flowchart of the proposed method is shown in Figure 2, and the detailed calculation process is shown as follows.

Step 1. Calculate the Euclidean distance between adjacent $M - 1$ stations with the test station according to (10).

Step 2. Select mostly related K stations with the test station.

Step 3. Predict traffic flow with LSTM network, respectively, in selected stations according to (13).

Step 4. Weigh prediction value in selected stations according to (14).

Step 5. Calculate the RMSE for the predicted traffic flow.

Step 6. Repeat Steps 2–5 with the different K ($K \leq M$).

Step 7. Find the smallest RMSE in all the different K .

Step 8. Obtain the predicted traffic flow in the test station when RMSE is the smallest.

3. Experiments

3.1. Data Description. The data used to evaluate the performance of the proposed model was collected in mainline detectors provided by the Transportation Research Data Lab (TDRL) at the University of Minnesota Duluth (UMD) Data Center from March 1st, 2015, to April 30th, 2015. The sampling period of the testing dataset was 5 min. In our experiment, we selected the road network in Figure 3 as the experiment area. The area mainly contains four expressways numbered I394, I494, US169, and TH100. There are 36 stations in the experiment area. The station locations and ID that are used are shown in Figure 3. Stations S339 and S448 are located near a transportation hub in road networks in the experiments. Therefore, they were selected as the test stations for the traffic flow prediction. Due to the similarity of traffic flow on the same workday in different weeks, we used the data in the one workday as train and test data in order to ensure the prediction stability. In our experiment, we chose the traffic flow data on Tuesday. Of course, we can choose any one workday from Monday to Friday. There was a total of 9-day traffic flow data on Tuesday in our test dataset. The dataset was divided into two datasets. The data in first 8 days was used as training sample, while the remaining data was employed as the testing sample for measuring prediction performance. The most commonly used prediction interval is 5 min, and we also select the prediction time interval as 5 min, and it is verified to be reasonable by the real experimental results.

Traffic flows for 5 consecutive Tuesdays are shown in Figure 4 in the station S339, and typical traffic flows are shown in Figure 5 in the station S339 and four neighboring stations. From Figure 4, we can see that there is a little difference in the rush hours; however, the profiles of the traffic flows are basically consistent. From Figure 5, it can be seen that there are some differences in different stations, but the data distribution is similar to the station S339. Because traffic flow data has high spatiotemporal correlation characteristics, it is effective to improve traffic prediction accuracy with the spatiotemporal correlations.

3.2. Performance Indexes. In order to evaluate the prediction performance, Root Mean Square Error (RMSE), which was the most frequently used metrics of prediction performance in previous work, and predicting accuracy (ACC) were chosen to evaluate the difference between the actual values with predicted values.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\widehat{y}_i - y_i)^2} \quad (15)$$

$$ACC = \left(1 - \frac{1}{N} \sum_{i=1}^N \left| \frac{\widehat{y}_i - y_i}{y_i} \right| \right) \times 100\% \quad (16)$$

where N is the length of prediction data and y_i and \widehat{y}_i are the measured value and predicted for i^{th} validation sample, respectively.

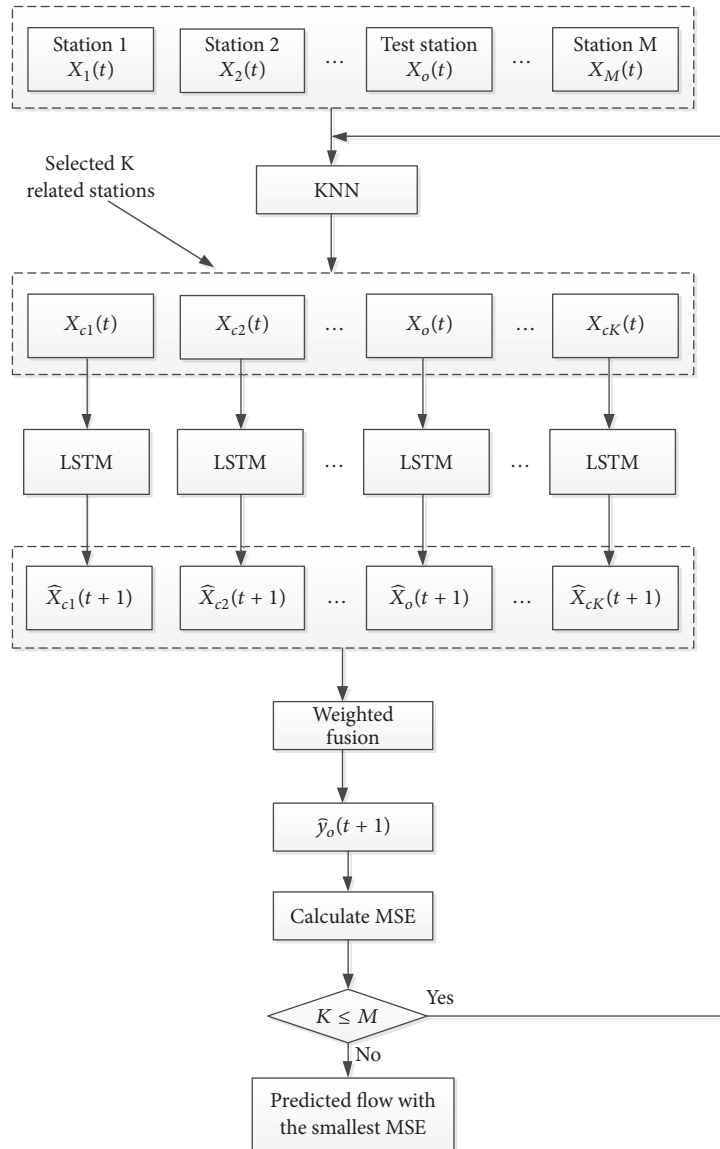


FIGURE 2: The flowchart of the proposed method.

4. Results and Discussions

4.1. Results Analysis. In our experiment, stations S339 and S448 are chosen as the test stations, which are located in the two directions of the road network. The timesteps are an important hyperparameter, which are the input size to the model and determines number of LSTM blocks in each level. Through experiment, when timesteps are set as 6, the prediction performance can achieve the optimal value. To validate the efficiency of the proposed method, the performance is compared with some representative approaches, including ARIMA model, SVR, wavelet neural network (WNN), DBN, and LSTM. In SARIMA model, AR and MA order are set as 5 and 4, and normal and seasonal differencing order are set as 1 and 2. In SVR model, kernel function is set as Radial Basis Function (RBF), the penalty parameter of the error term as 300, and the iteration number as 1000. In WNN model, the

number of hidden nodes is set as 6, the learning rate as 0.001, and the iteration number as 500. For DBN model, 3-layer architecture is used, and the number of nodes in each layer is set to 128 for simplicity.

The predicted results of different models and real traffic flow are shown within one day in Figures 6 and 7. It is observed that the predicted traffic flow has similar traffic patterns with the real traffic flow and the prediction value of the proposed KNN-LSTM model is almost coincided with the measured data, especially in morning and evening peak hours. The RMSE and ACC for different models are shown for stations S339 and S448 in Table 1. It can be seen that the proposed method has the minimum RMSE. The average ACC for the proposed method is 95.75%, which improve by 28.92%, 8.31%, 14.44%, 6.95%, and 4.32% compared with other models. The traditional ARIMA model has the worst prediction performance, which assumes the traffic flow data



FIGURE 3: The ID and locations of stations in our experiment.

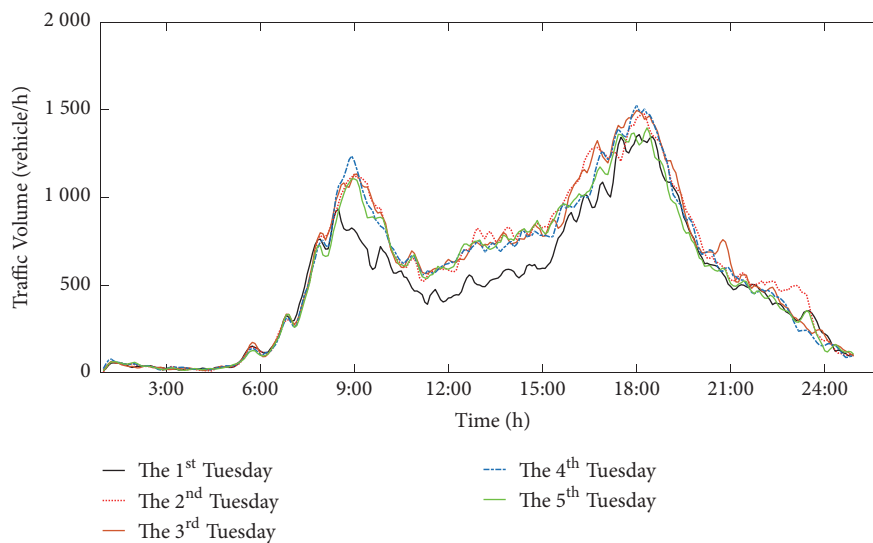


FIGURE 4: Traffic flows for 5 consecutive Tuesdays in the station S339.

is a stationary process but this is not always true in reality. The SVR and WNN method receive better RMSE and ACC than the ARIMA model, while they show weakness when compared with the deep learning methods. The DBN model has also no obvious advantage over SVR.

4.2. Discussions. In this paper, KNN is used to select mostly related K stations with the test station. The different K values have different prediction performance. We search for all possible values for K , the corresponding K is the optimal value when the RMSE is minimum. The optimal K is set as 10 for the station S339 in our experiment, and the ID numbers of selected stations are S339, S340, S341, S321, S337, S342, S338,

S344, S336, and S293. The optimal K is set as 6 for station S448, and the ID numbers of selected stations are S448, S447, S446, S450, S737, and S452. As shown in Figure 3, it can be seen that almost all of the selected stations are located in upstream and downstream in the test stations. From classical understanding, closer stations from the prediction station have more correlation than those further stations. In fact, some further stations have also correlation with the prediction station. For the test station S339, the closer station S343 is not selected, and closer station S451 is not selected for the test station S448. However, it is consistent with the general fact that the traffic flow in the upstream and downstream has great influence on the prediction result in the traffic

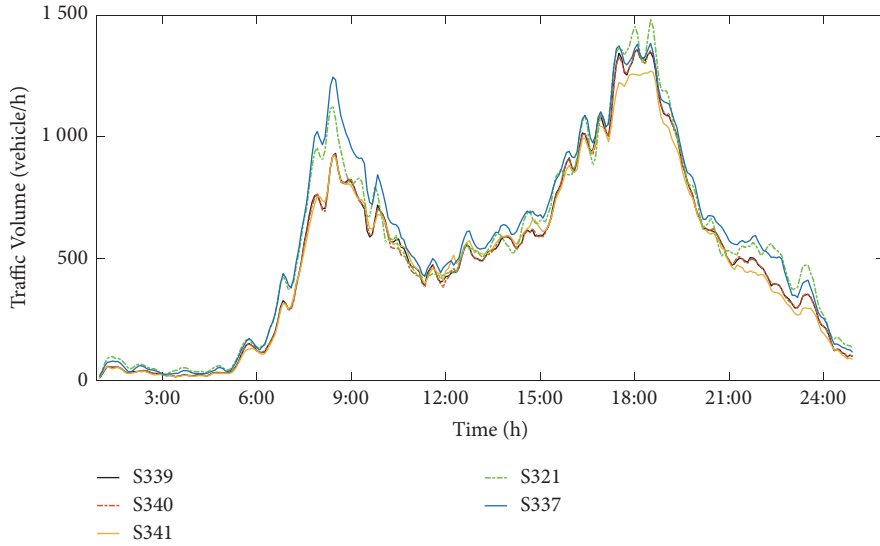


FIGURE 5: Traffic flows in the station S339 and 4 neighboring stations.

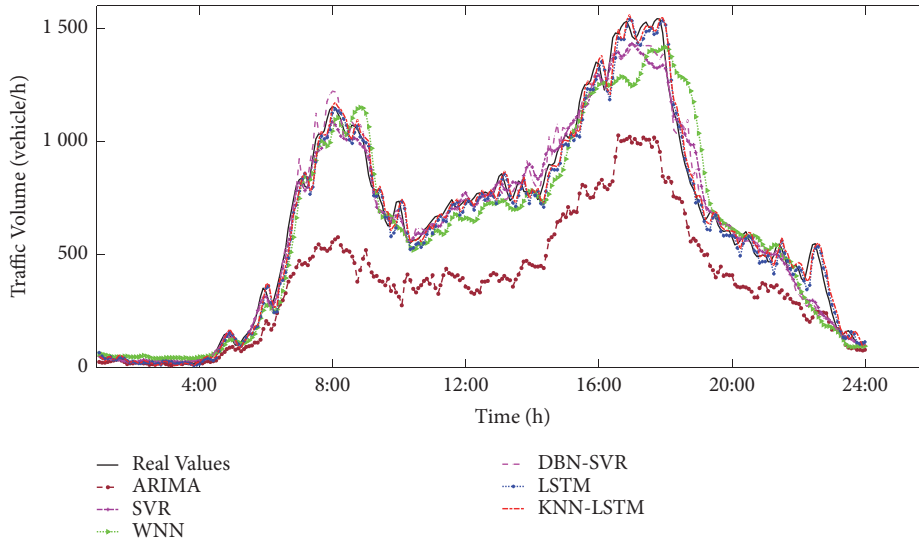


FIGURE 6: The real and predicted traffic flow in S339.

flow prediction. When $K = 1$, the temporal correlation is only considered, the average ACC is 91.43% which is decreased by 4.32% compared with the proposed method. It indicates that spatiotemporal features have important roles in the traffic prediction. These results verify the superiority and feasibility of the KNN-LSTM, which employ KNN to capture the spatial features and mine temporal regularity with the LSTM networks.

5. Conclusions

In this paper, we proposed a spatiotemporal traffic flow prediction method combined with KNN and LSTM. KNN is used to select mostly related neighboring stations that indicated the spatiotemporal correlation with the test station. A LSTM network was applied to predict traffic flow,

TABLE 1: Prediction performances of different models.

Models	S339		S448	
	RMSE	ACC(%)	RMSE	ACC(%)
ARIMA	36.3223	61.09	44.6856	72.57
SVR	7.7424	88.17	18.4911	86.71
WNN	8.5240	74.69	12.4526	87.93
DBN-SVR	7.3277	89.60	15.3746	88.01
LSTM	1.8185	90.39	2.7499	92.47
KNN-LSTM	1.7403	94.59	2.5465	96.91

respectively, in selected stations. LSTM is able to exploit the long-term dependency in the traffic flow data and discover the latent feature representations hidden in the traffic flow, which yields better prediction performance. The final

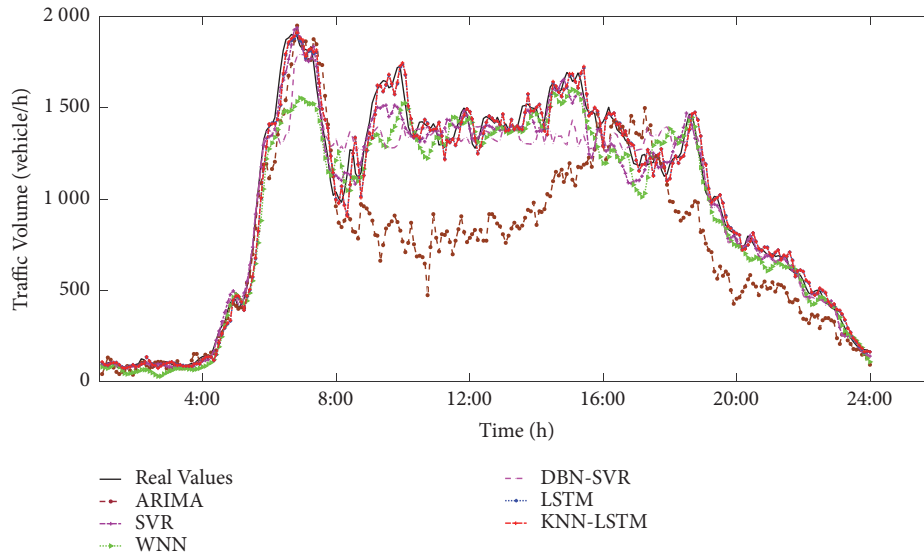


FIGURE 7: The real and predicted traffic flow in S448.

prediction results in test station are obtained by weighting with rank-exponent method. We evaluated the performance of our model with real traffic data provided by TDRL and compared with ARIMA, SVR, WNN, DBN, and LSTM model. The results show that proposed model is superior to other methods. Since the traffic flow data is affected by weather, incident, and other factors, the impact of these factors on traffic flow data will be further studied so as to improve the prediction accuracy.

Data Availability

The data used in this paper are collected in mainline detectors provided by the Transportation Research Data Lab (TDRL) at the University of Minnesota Duluth (UMD) Data Center. (<http://www.d.umn.edu/~tkwon/TMCdata/TMCarchive.html>) If any researcher requests for these data, they can download from the website.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was partly supported by the National Key R&D Program of China (2018YFC0808706) and the National Natural Science Foundation of China (Grant no. 5157081053). The authors are also grateful to the UMD Data Center (TDRL) for providing the data.

References

- [1] M. van der Voort, M. Dougherty, and S. Watson, "Combining Kohonen maps with ARIMA time series models to forecast traffic flow," *Transportation Research Part C: Emerging Technologies*, vol. 4, no. 5, pp. 307–318, 1996.
- [2] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: theoretical basis and empirical results," *Journal of Transportation Engineering*, vol. 129, no. 6, pp. 664–672, 2003.
- [3] G. Shi, J. Guo, W. Huang, and B. M. Williams, "Modeling seasonal heteroscedasticity in vehicular traffic condition series using a seasonal adjustment approach," *Journal of Transportation Engineering*, vol. 140, no. 5, pp. 1053–1058, 2014.
- [4] L. L. Ojeda, A. Y. Kibangou, and C. C. De Wit, "Adaptive Kalman filtering for multi-step ahead traffic flow prediction," in *Proceedings of the IEEE American Control Conference*, pp. 4724–4729, Washington, DC, USA, 2013.
- [5] J. Guo, W. Huang, and B. M. Williams, "Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 50–64, 2014.
- [6] B. L. Smith and M. J. Demetsky, "Short-term traffic flow prediction: neural network approach," *Transportation Research Record*, vol. 1453, pp. 98–104, 1994.
- [7] Y. W. Zhang, Z. P. Song, X. L. Weng, and Y. L. Xie, "A new soil-water characteristic curve model for unsaturated loess based on wetting-induced pore deformation," *Geofluids*, vol. 2019, Article ID 5261985, 13 pages, 2019.
- [8] Q. P. Wang and H. Sun, "Traffic structure optimization in historic districts based on green transportation and sustainable development concept," *Advances in Civil Engineering*, vol. 2019, Article ID 9196263, 18 pages, 2019.
- [9] D. Chen, "Research on traffic flow prediction in the big data environment based on the improved RBF neural network," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2000–2008, 2017.
- [10] M. Bernaś, B. Płaczek, P. Porwik, and T. Pamuła, "Segmentation of vehicle detector data for improved k-nearest neighbours-based traffic flow prediction," *IET Intelligent Transport Systems*, vol. 9, no. 3, pp. 264–274, 2015.
- [11] S. Wu, Z. Yang, X. Zhu, and B. Yu, "Improved KNN for short-term traffic forecasting using temporal and spatial information," *Journal of Transportation Engineering*, vol. 140, no. 7, Article ID 04014026, 2014.

- [12] P. Dell'acqua, F. Bellotti, R. Berta, and A. De Gloria, "Time-aware multivariate nearest neighbor regression methods for traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3393–3402, 2015.
- [13] P. Cai, Y. Wang, G. Lu, P. Chen, C. Ding, and J. Sun, "A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 62, pp. 21–34, 2016.
- [14] B. Sun, W. Cheng, P. Goswami, and G. Bai, "Short-term traffic forecasting using self-adjusting k-nearest neighbours," *IET Intelligent Transport Systems*, vol. 12, no. 1, pp. 41–48, 2018.
- [15] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han, "Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6164–6173, 2009.
- [16] Y. Sun, B. Leng, and W. Guan, "A novel wavelet-SVM short-time passenger flow prediction in Beijing subway system," *Neurocomputing*, vol. 166, pp. 109–121, 2015.
- [17] J. Wang, W. Deng, and Y. Guo, "New Bayesian combination method for short-term traffic flow forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 79–94, 2014.
- [18] Y. Xu, Q.-J. Kong, R. Klette, and Y. Liu, "Accurate and interpretable bayesian MARS for traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2457–2469, 2014.
- [19] J. Lai, K. Wang, J. Qiu, F. Niu, J. Wang, and J. Chen, "Vibration response characteristics of the cross tunnel structure," *Shock and Vibration*, vol. 2016, Article ID 9524206, 16 pages, 2016.
- [20] D. Xia, B. Wang, H. Li, Y. Li, and Z. Zhang, "A distributed spatial-temporal weighted model on MapReduce for short-term traffic flow forecasting," *Neurocomputing*, vol. 179, pp. 246–263, 2016.
- [21] F. Moretti, S. Pizzuti, S. Panzieri, and M. Annunziato, "Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling," *Neurocomputing*, vol. 167, pp. 3–7, 2015.
- [22] Z. J. Zhou, C. N. Ren, G. J. Xu, H. C. Zhan, and T. Liu, "Dynamic failure mode and dynamic response of high slope using shaking table test," *Shock and vibration*, vol. 2019, Article ID 4802740, 15 pages, 2019.
- [23] Y. W. Zhang, X. L. Weng, Z. P. Song, and Y. F. Sun, "Modeling of loess soaking induced impacts on metro tunnel using water soaking system in centrifuge," *Geofluids*, vol. 2019, Article ID 5487952, 13 pages, 2019.
- [24] S.-D. Oh, Y.-J. Kim, and J.-S. Hong, "Urban traffic flow prediction system using a multifactor pattern recognition model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2744–2755, 2015.
- [25] Z. J. Zhou, S. S. Zhu, X. Kong, J. T. Lei, and T. Liu, "Optimization analysis of settlement parameters for post-grouting piles in loess area of Shaanxi, China," *Advances in Civil Engineering*, vol. 2019, Article ID 7085104, 11 pages, 2019.
- [26] M.-L. Huang, "Intersection traffic flow forecasting based on ν -GSVR with a new hybrid evolutionary algorithm," *Neurocomputing*, vol. 147, no. 1, pp. 342–349, 2015.
- [27] X. Luo, L. Niu, and S. Zhang, "An algorithm for traffic flow prediction based on improved SARIMA and GA," *KSCE Journal of Civil Engineering*, vol. 22, no. 10, pp. 4107–4115, 2018.
- [28] X. Luo, D. Li, and S. Zhang, "Traffic flow prediction during the holidays based on DFT and SVR," *Journal of Sensors*, vol. 2019, Article ID 6461450, 10 pages, 2019.
- [29] J. Lai, J. Qiu, H. Fan, Q. Zhang, J. Wang, and J. Chen, "Fiber Bragg grating sensors-based in situ monitoring and safety assessment of loess tunnel," *Journal of Sensors*, vol. 2016, Article ID 8658290, 10 pages, 2016.
- [30] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 3–19, 2014.
- [31] S. Suhas, V. V. Kalyan, M. Katti, B. V. Prakash, and C. Naveena, "A comprehensive review on traffic prediction for intelligent transport system," in *Proceedings of the 2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)*, pp. 138–143, Bangalore, India, March 2017.
- [32] I. Lana, J. Del Ser, M. Velez, and E. I. Vlahogianni, "Road traffic forecasting: recent advances and new challenges," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 2, pp. 93–109, 2018.
- [33] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191–2201, 2014.
- [34] Y. Lv, Y. Duan, W. Kang et al., "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.
- [35] Y. Duan, Y. Lv, and F. Y. Wang, "Performance evaluation of the deep learning approach for traffic flow prediction at different times," in *Proceedings of the IEEE International Conference on Service Operations and Logistics, and Informatics*, pp. 223–227, Beijing, China, 2016.
- [36] R. Souza, A. Koesdwiady, and F. Karray, "Big-data-generated traffic flow prediction using deep learning and dempster-shafer theory," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 3195–3202, Vancouver, Canada, July 2016.
- [37] A. Koesdwiady, R. Souza, and F. Karray, "Improving traffic flow prediction with weather information in connected cars: a deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9508–9517, 2016.
- [38] H. F. Yang, T. S. Dillon, and Y. P. Chen, "Optimized structure of the traffic flow forecasting model with a deep learning approach," *IEEE Transactions on Neural Networks & Learning Systems*, vol. 28, no. 10, pp. 2371–2381, 2016.
- [39] T. Zhou, G. Han, X. Xu et al., " δ -agree AdaBoost stacked autoencoder for short-term traffic flow forecasting," *Neurocomputing*, vol. 247, pp. 31–38, 2017.
- [40] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1–17, 2017.
- [41] Y. Zhang and G. Huang, "Traffic flow prediction model based on deep belief network and genetic algorithm," *IET Intelligent Transport Systems*, vol. 12, no. 6, 2018.
- [42] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PLoS ONE*, vol. 10, no. 3, Article ID e0119044, 2015.
- [43] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015.

- [44] H. Shao and B.-H. Soong, "Traffic flow prediction with Long Short-Term Memory Networks (LSTMs)," in *Proceedings of the IEEE Region 10 Conference*, pp. 2986–2989, Singapore, 2016.
- [45] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.
- [46] F. G. Habtemichael, M. Cetin, and K. A. Anuar, "Methodology for quantifying incident-induced delays on freeways by grouping similar traffic patterns," in *Proceedings of the Transportation Research Record 94th Annual Meeting*, Washington, DC, USA, 2015.

Research Article

Driver Distraction Identification with an Ensemble of Convolutional Neural Networks

Hesham M. Eraqi ¹, Yehya Abouelnaga ²,
Mohamed H. Saad ³ and Mohamed N. Moustafa ¹

¹Department of Computer Science & Engineering, The American University in Cairo, Egypt

²Department of Informatics, Technical University of Munich, Germany

³Department of Computer and Systems Engineering, Ain Shams University, Egypt

Correspondence should be addressed to Hesham M. Eraqi; heraqi@aucegypt.edu
and Yehya Abouelnaga; yehya.abouelnaga@tum.de

Received 6 October 2018; Revised 27 December 2018; Accepted 14 January 2019; Published 13 February 2019

Guest Editor: Yasser Hassan

Copyright © 2019 Hesham M. Eraqi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The World Health Organization (WHO) reported 1.25 million deaths yearly due to road traffic accidents worldwide and the number has been continuously increasing over the last few years. Nearly fifth of these accidents are caused by distracted drivers. Existing work of distracted driver detection is concerned with a small set of distractions (mostly, cell phone usage). Unreliable *ad hoc* methods are often used. In this paper, we present the first publicly available dataset for driver distraction identification with more distraction postures than existing alternatives. In addition, we propose a reliable deep learning-based solution that achieves a 90% accuracy. The system consists of a genetically weighted ensemble of convolutional neural networks; we show that a weighted ensemble of classifiers using a genetic algorithm yields a better classification confidence. We also study the effect of different visual elements in distraction detection by means of face and hand localizations, and skin segmentation. Finally, we present a thinned version of our ensemble that could achieve 84.64% classification accuracy and operate in a real-time environment.

1. Introduction

Over the 20 years from 1980 to 2000, the number of licensed drivers in the United States increased by 23.7%, to reach 190.6 million licenses [1]. From 1990 to 2000, the urban vehicles miles traveled increased by 80%, while roads building rose by only 37% [2]; this makes driving a common activity for many people and makes the driving safety an important issue in everyday life. Construction of new roads did not keep up with the noticeable increase in vehicles, leading to more traffic congestion [2]. In addition, In-Vehicle Information Systems (IVISs) such as media players and navigation devices introduce more distraction to the driving experience and lead to more accidents.

Despite safety improvements in road and vehicle design, the total number of fatal crashes still increases [1]. The 2017 Global Status Report of the World Health Organization (WHO) reported an estimated 1.25 million yearly deaths due

to road traffic accidents worldwide, with up to 50 million people sustaining non-fatal injuries as a result of road traffic accidents [3]. To put that mortality rate into perspective, it is the same as having around seven Boeing airplanes crashing or disappearing every day. Car accidents mortality rate compares to those of serious diseases (i.e., Hepatitis & HIV). At the same year, the numbers of global deaths attributable to Hepatitis and HIV are estimated to be in the order of 1.3 million [4] and 1.1 million [3], respectively, which is almost the same as the number of the people dying yearly due to road traffic accidents. Moreover, road traffic accidents cause a huge property damage, and the number of road accidents due to distracted driving is steadily increasing.

Nearly fifth of traffic accidents are caused by a distracted driver according to the National Highway Traffic Safety Administration (NHTSA), with approximately 90% of road accidents being due to human errors in the United States [5]. Despite the fact that vehicle crashes are attributed to multiple

causes, driver error represents a dominant factor [6]. In 2015, 3,477 people were killed, and 391,000 were injured in motor vehicle crashes involving distracted drivers [7]. The major cause of these accidents was the use of mobile phones [7].

The NHTSA defines distracted driving as “any activity that diverts attention from driving”, including (a) talking or texting on one’s phone, (b) eating and drinking, (c) talking to passengers, or (d) fiddling with the stereo, entertainment, or navigation system [7]. The Center for Disease Control and Prevention (CDC) provides a broader definition of distracted driving by taking into account visual (i.e., taking one’s eyes off the road), manual (i.e., taking one’s hands off the driving wheel), and cognitive (i.e., taking one’s mind off driving) causes [8].

One way to address the problem of distracted driver is to develop distraction mitigation systems, which adapt IVIS functions according to driver state. In such a mitigation system, correctly identifying driver distraction is critical, which is the focus of this work. We envision a future where smart vehicles could detect and identify such distraction and warn the driver against it or take preventive measures. On the other hand, such detection systems can help law enforcement to identify distraction on highway using radar cameras and penalize certain forms of distraction. Moreover, the recent commercial semiautonomous cars require drivers to pay attention to the road [9]. Autonomous steering control [10] systems require drivers to be ready to take back control of the wheel [9]. This is what makes distracted drivers detection an important system component in these cars. Distraction detection can also be used to enable Advanced Driver Assistance Systems (ADAS) features [11] like Collision Avoidance Systems (CAS) that have to plan evasive maneuvers [12].

Research in the field of distracted driving detection follows the definitions presented in [7, 8]. It detects manual, visual, or cognitive types of distractions. Cognitive distractions deal with tasks of listening, conversing, daydreaming, or just becoming lost in thought. In this form of distraction, the driver is “mentally” distracted from safe driving even though they are in a safe driving posture. Visual distractions often refer to situations where the driver takes their eyes off the road due to either “the presence of salient visual information away from the road causing spontaneous off-road eye glances and momentary rotation of the head” or the use of multimedia devices (i.e., cell phones, navigation, or entertainment systems) [13]. Visual distractions are coined in the following terms: “sleepiness”, “drowsiness”, “fatigue”, and “inattention”. And, they usually depend on facial landmarks detection and tracking. Manual distractions are mainly concerned with driver’s activities other than safe driving (i.e., reaching behind, adjusting hair and makeup, or eating and drinking). In this kind of distraction, authors often tend to depend heavily on hand tracking and driving posture estimation. In this paper, we focus only on “manual” distractions where a driver is distracted by texting or using cell phone, calling, eating or drinking, reaching behind, fiddling with the radio, adjusting hair and makeup, or talking to a passenger.

Recently, the field witnessed a dramatic increase in computational power (thanks to massive parallelization in modern GPUs) and available big data for deep learning

training. That enabled more research on advanced/deep network architectures. One example is the proliferation of deep learning-based solutions utilizing convolutional neural networks to solve computer vision problems, which achieved unprecedented performance [15–17]. That improvement is a byproduct of learning feature maps (as in [15, 18]) rather than hand-crafting them in traditional computer vision practices [19].

In this paper, we detect and identify distraction using deep learning. RGB images are obtained from a camera mounted above the dashboard. We train and benchmark multiple convolutional neural network architectures. We use pretrained networks in a “transfer learning”-style where networks are pretrained on the ImageNet dataset [17] (that is, 1.2 million images and 1000 classes). This initializes the network with a generic set of features and allows it to recognize a variety of objects with high precision. Then, we retrain the latent fully connected layers to recognize the distraction postures (instead of ImageNet classes). During training, the convolutional layers are usually either kept frozen or updated with a miniscule learning rate. In such way, we transfer the networks knowledge of about 1,000 different objects (i.e., the broad domain) to our more specific domain of 10 driving distraction postures.

We present a real-time system for driver distraction identification that uses a learnable weighted ensemble of convolutional neural networks (CNNs), a new method for skin segmentation, a challenging distracted driver’s dataset (see Figure 1) on which we evaluate our proposed solution, and an annotation tool [20] for action labeling that can be used to extend our dataset.

2. Literature Review

The work in the distracted driver detection field over the past seven years could be clustered into four groups: multiple independent cell phone usage detection publications, Laboratory of Intelligent and Safe Automobiles in University of California San Diego (UCSD) datasets and publications, Southeast University Distracted Driver dataset and affiliated publications, and recently StateFarm’s Distracted Driver Kaggle competition.

2.1. Cell Phone Usage Detection. Reference [21] presents an SVM-based model that detects the use of mobile phone while driving (i.e., distracted driving). Their dataset consists of frontal image view of a driver’s face. They also make premade assumptions about hand and face locations in the picture. Reference [22] presents another SVM-based classification method to detect cell phone usage. However, their dataset is collected from transportation imaging cameras that are deployed in highways and traffic lights, which is, indeed, more competitive. Reference [23] uses AdaBoost classifier and Hidden Markov Models to classify a Kinect’s RGB-D data. Their solution depends on indoor-produced data. They sit on a chair and a mimic a certain distraction (i.e., talking on the phone). This setup misses two essential points: the lighting conditions and the distance between a Kinect and



FIGURE 1: Ten classes of driver postures from our dataset.

the driver. In real-life applications, a driver is exposed to a variety of lighting conditions (i.e., sunlight and shadow). Reference [24] suggests using a Hidden Conditional Random Fields (HCRF) model to detect cell phone usage. Their model operates face, mouth, and hand features of images obtained from a camera mounted above the dashboard. Reference [25] devised a Faster-RCNN model to detect driver's cell phone usage and "hands on the wheel". Their model is mainly geared towards face/hand segmentation. They train their Faster-RCNN on the dataset proposed in [26] (that we also use in this paper). Their proposed solution runs at a 0.06 and 0.09 frames per second for cell phone usage and "hands on the wheel" detection. Reference [27] tackles the problem of cell phone usage detection. Their approach does not hold any static assumptions though (i.e., in which region of the image a face is expected to be found). They use a Supervised Descent Method (SDM) to localize the face landmarks and, then, extract two bounding boxes to the left and the right side of the face. They train a classifier on each of the two regions to detect cell phone usage: right hand, left hand, or no usage. Using a histogram of gradients (HOG) and an AdaBoost classifier, they achieve a 93.9% classification accuracy and operate in a near real-time speed (7.5 frames per second).

2.2. UCSD's Laboratory of Intelligent and Safe Automobiles Work. Reference [28] presents a vision-based analysis framework that recognizes in-vehicle activities using two Kinect cameras that provide frontal and back views of the driver. Their approach provides "hands on the wheel" information (i.e., left hand only, both hands, no hands) and uses this information to detect three types of distractions: adjusting the radio, operating the gear, and adjusting the mirrors. Reference [29] presents a fusion of classifiers where the image is to be segmented into three regions: wheel, gear, and instrument panel (i.e., radio). It proposes a classifier for each segment to detect existence of hands in those regions. The hand information (i.e., output of the classifiers) is passed to an "activity classifier" that infers the actual activity (i.e., adjusting the radio, operating the gear). Reference [30] extends existing research to include eye cues to previously existing head and hands cues. However, it still considers three types of distractions: "wheel region interaction with two hands on the wheel, gear region activity, and instrument cluster region activity". Reference [31] presents a region-based classification

approach. It detects hands presence in certain predefined regions in an image. A model is learned for each region separately. All regions are later joined using a second-stage classifier.

2.3. Southeast University Distracted Driver Dataset. Reference [32] designs a more inclusive distracted driving dataset with a side view of the driver and more activities: grasping the steering wheel, operating the shift lever, eating a cake, and talking on a cellular phone. It introduces a contourlet transform for feature extraction and, then, evaluates the performance of different classifiers: Random Forests (RF), k -Nearest Neighbors (KNN), and Multilayer Perceptron (MLP). The random forests achieved the highest classification accuracy of 90.5%. Reference [33] showed that using a multiwavelet transform improves the accuracy of Multilayer Perceptron classifier to 90.61% (previously 37.06% in [32]). Reference [34] showed that using a Support Vector Machine (SVM) with an intersection kernel, followed by Radial Basis Function (RBF) kernel, achieved the highest accuracies of 92.81% and 94.25%, respectively (in comparison with [32, 33]). After testing against other classification methods, they concluded that an SVM with intersection kernel offers the best real-time quality (67 frames per second) and better classification performance. Reference [35] improves the Multilayer Perceptron classifier using combined features of Pyramid Histogram of Oriented Gradients (PHOG) and spatial scale feature extractors. Their Multilayer Perceptron achieves a 94.75% classification accuracy. Reference [36] utilizes motion history images (HMI) to make use of the data's temporality. Pyramid Histogram of Gradients (PHOG) is applied to the motion history images. A Random Forrest trains on the extracted features and yields a 96.56% accuracy. Reference [37] presents a convolutional neural network solution that achieves a 99.78% classification accuracy. They train their network in a 2-step process. First, they use pretrained sparse filters as the parameters of the first convolutional layer. Second, they fine-tune the network on the actual dataset. Their accuracy is measured against the 4 classes of the Southeast dataset.

2.4. StateFarm's Dataset. StateFarm's Distracted Driver Detection competition on Kaggle was the first publicly available dataset for posture classification. In the competition,

StateFarm defined ten postures to be detected: safe driving, texting using right hand, talking on the phone using right hand, texting using left hand, talking on the phone using left hand, operating the radio, drinking, reaching behind, doing hair and makeup, and talking to passenger. Our work, in this paper, is mainly inspired by StateFarm’s Distracted Driver’s competition. While the usage of StateFarm’s dataset is limited to the purposes of the competition [38], we designed a similar dataset that follows the same postures.

3. Dataset Design

Creating a new dataset was essential to the completion of this work. The available alternatives to our dataset are StateFarm and Southeast University (SEU) datasets. StateFarm’s dataset is to be used for their Kaggle competition purposes only (as per their regulations) [38]. As for Southeast University (SEU) dataset, it presents only four distraction postures. And, after multiple attempts to obtain it, we figured out that the authors do not make it publicly available. All the papers ([32–37, 39]) that benchmarked against the dataset are affiliated with either Southeast University, Xi’an Jiaotong-Liverpool University, or Liverpool University, and they have at least one shared author. With that being said, the collected “distracted driver” dataset is the first publicly available (obtainable after signing a license agreement) for driving posture estimation research. Our dataset is publicly available, subject to signing our agreement form from [40]. The dataset introduced in this work is an extended and cleaned-up version of our dataset presented in [41].

3.1. Camera Setup. Our dataset collection setup has a single camera with a fixed perspective, and the data collection was conducted on two phases. In each phase, a different camera is used. In one phase we use the ASUS ZenFone smartphone (Model ZD551KL) rear camera [42], and in the other phase we used the DS325 Sony DepthSense camera [43]. The latter camera provides depth information, but we only record the RGB images. Collecting data from different cameras adds an extra dimension of diversity to our dataset, and we demonstrate the feasibility of effective distraction detection by relying on RGB cameras which are widely available and of low cost.

The data was collected in a video format and, then, cut into individual images, 1080×1920 or 640×480 each. The cameras are fixed using an arm strap to the car roof handle on top of the front passenger’s seat. In our use case, this setup proved to be very flexible as we needed to collect data in different vehicles.

3.2. Labeling. In order to label the collected videos, we designed a simple multiplatform action annotation tool using modern web technologies: Electron, AngularJS, and JavaScript. The annotation tool is open-source and publicly available at [20].

3.3. Statistics. We had 44 participants from 7 different countries: Egypt (37), Germany (2), USA (1), Canada (1), Uganda

(1), Palestine (1), and Morocco (1). Out of all participants, 29 were males and 15 were females. Some drivers participated in more than one recording session with different time of day, driving conditions, and wearing different clothes. Videos were shot in 5 different cars: Proton Gen 2, Mitsubishi Lancer, Nissan Sunny, KIA Carens, and a prototyping car. We extracted 14,478 frames distributed over the following classes: safe driving (2,986), phone right (1,256), phone left (1,320), text right (1,718), text left (1,124), adjusting radio (1,123), drinking (1,076), hair or makeup (1,044), reaching behind (1,034), and talking to passenger (1,797). The sampling is done manually by inspecting the video files with eye and giving a distraction label for each frame. The transitional actions between each consecutive distraction types are manually removed. Figure 1 shows samples for the ten classes in our dataset.

4. Proposed Method

Our proposed solution consists of a genetically weighted ensemble of convolutional neural networks. The convolutional neural networks are trained on raw images, skin-segmented images, face images, hands images, and “face+hands” images. On those five images sources, we train and benchmark an AlexNet network [15], an InceptionV3 network [44], a ResNet network having 50 layers [45], and a VGG-16 network [46]. We fine-tune a pretrained ImageNet model (i.e., transfer learning) for these networks. Then, we evaluate a weighted sum of all networks’ outputs yielding the final class distribution using a genetic algorithm. The system overview is shown in Figure 2.

4.1. Skin Segmentation. Skin segmentation is a challenging problem to solve, mainly due to the different lighting conditions happening during driving. We use a Multivariate Gaussian Naive Bayes classifier to develop a pixel-wise skin segmentation model. Our model is similar to [47] except that we do not use a histogram as a Likelihood function. Instead, we fit the training data into Gaussian distributions to formulate the Likelihood functions. The posterior probability is evaluated as in (1).

$$\mathbb{P}(\text{skin} | x) = \frac{\mathbb{P}(x | \text{skin}) \cdot \mathbb{P}(\text{skin})}{\mathbb{P}(x)}, \quad (1)$$

$$\mathbb{P}(x | \text{skin}) \sim \mathcal{N}(\mu_{\text{skin}}, \Sigma_{\text{skin}})$$

Note that $\mathbb{P}(\text{skin}) = \mathbb{P}(\text{non-skin}) = 0.5$ (i.e., we do not make any assumptions about existence of skin pixels in the image).

We trained our model using the UCI Skin Segmentation dataset [48]. The database contains RGB colors that are labeled for the skin and non-skin classes. It is generated using skin textures from face images of people from diverse ages, genders, and races. It contains a total of 245,057 color samples, out of which 50,859 are the skin samples and 194,198 are non-skin samples. Two Gaussian distributions (Likelihoods) are constructed for the skin and the non-skin classes by estimating μ_{skin} , Σ_{skin} , $\mu_{\text{non-skin}}$, and $\Sigma_{\text{non-skin}}$ from the training data. For deployment phase, each pixel x in

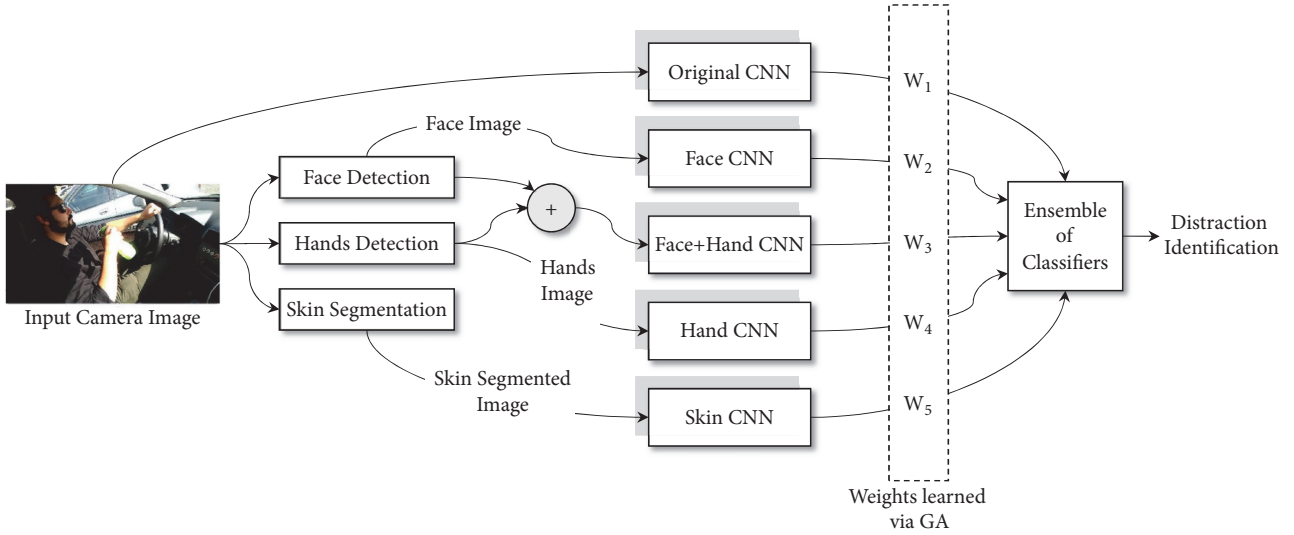


FIGURE 2: An overview of our proposed solution. A face detector, a hand detector, and a skin segmenter are run against each frame. For each output image (i.e., Skin, Face, Hands), an AlexNet and an InceptionV3 network are trained (i.e., resulting in 10 neural networks: 5 AlexNet and 5 InceptionV3). The overall class distribution is determined by the weighted sum of all softmax layers. The weights are learned using a genetic algorithm.



FIGURE 3: Left and right images show a sample for skin segmentation result without and with training the pixels spatial information, respectively.

the input image is fed to the model as in (2). And then, a probability heatmap of skin in the image can be constructed. We classify a pixel to a “skin” if $\text{Model}(x) > 0.5$. Then, we cluster the skin pixels into objects and remove those with a small number of pixels, because neither faces nor hands skin blobs are expected to have small number of pixels.

$$\begin{aligned} \text{Model}(x) &= \frac{\mathbb{P}(\text{skin} | x)}{\mathbb{P}(\text{skin} | x) + \mathbb{P}(\text{non-skin} | x)} \\ &= \frac{\mathbb{P}(x | \text{skin})}{\mathbb{P}(x | \text{skin}) + \mathbb{P}(x | \text{non-skin})} \end{aligned} \quad (2)$$

One key disadvantage of such method is that it is very sensitive to image illumination conditions. Hence, incorporating pixel location can improve the skin classification accuracy. One way is to pass the pixel location to the input feature vector. However, to the best of our knowledge, there is no available dataset to train and evaluate such method. Besides, annotating a new dataset is costly. Therefore, we adopt an active learning-based approach to supervise the training. The above classifier (without pixel spatial information) is run

against all training images to generate skin masks. Generated masks are manually inspected to cherry-pick samples with high skin segmentation accuracy. Those pixels are used as new training data of the proposed skin segmentation classifier, such that the feature vector includes pixels spatial information (X- and Y-coordinates within the images) in addition to the color information (Red, Green, and Blue color components). Figure 3 shows a sample skin-segmented image with (right) and without (left) pixel spatial information. We notice an accuracy improvement after considering the pixels spatial information. More test data results are presented in Section 5.

4.2. Face and Hands Detection. We trained the model presented in [49] on the Annotated Facial Landmarks in the Wild (AFLW) face dataset [50]. It was sensitive to distance from the camera; faces that were close to the camera were not easily detected. We found that the pretrained model (presented in [14]) produced better results on our dataset. Given that we did not have any hand labeled face bounding boxes, we could not formally compare the two models. But it was obvious that [14] gives a better detection accuracy based on inspecting the results manually. However, face misdetections are noticed in

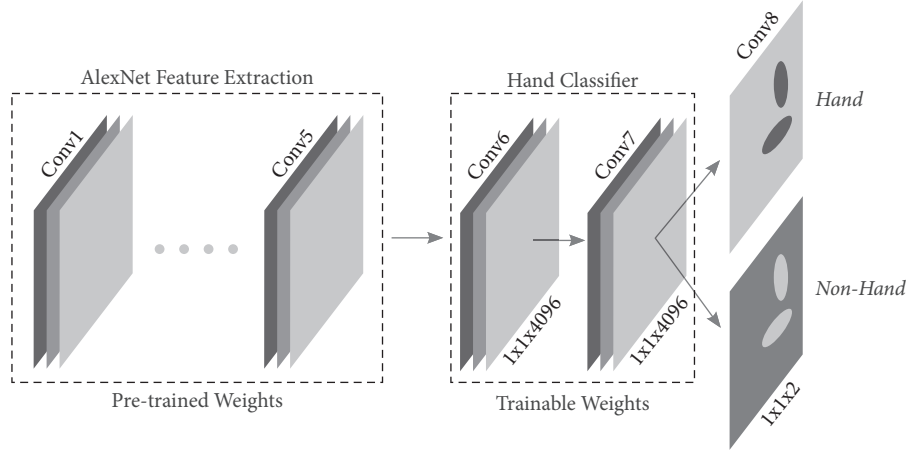


FIGURE 4: A modified version of AlexNet for hands localization (“Conv” stands for a convolutional layer). An AlexNet is trained on “hands” and “non-hands” images. Fully connected layers are replaced with convolutional layers with the same weights of the fully connected layers.

several examples, mainly because the detector is not trained to handle non-frontal faces.

As for hands detection, we used the pretrained model presented in [51] with modifications. Their trained model was a binary class AlexNet that classifies hands/non-hands for different proposal windows. We transferred the weights of the fully connected layers (i.e., fc6, fc7, and fc8) into convolutional layers such that each neuron in the fully connected layer was transferred into a feature map layer with a 1-pixel kernel size. Our proposed architecture, shown in Figure 4, accepts variant-size inputs and produces variant-size outputs. The last convolutional layer has a depth of 2 (i.e., the binary classes), and for each pixel the summation of the two depths is one as in (3), where W and H are the output’s width and height, respectively.

$$\text{Conv8}_{x,y,0} + \text{Conv8}_{x,y,1} = 1, \quad 0 \leq x \leq W \quad 0 \leq y \leq H \quad (3)$$

4.3. Convolutional Neural Network. For distracted driver posture classification, we trained and benchmarked different neural networks architectures: an AlexNet [15], an InceptionV3 [44], a ResNet network having 50 layers [45], and a VGG-16 network [46]. Each network is trained on 5 different image sources (i.e., raw, skin, face, hands, and face+hands images).

We trained our AlexNet models from scratch. We did not use a pretrained model. As for InceptionV3, we performed transfer learning. We fine-tuned a pretrained model [52] on the distraction postures. We removed the “logits” fully connected layer and replaced it with 10-neuron fully connected layer (i.e., corresponding to 10 driving postures). For all of our models, we used a gradient descent optimizer with an initial learning rate of 10^{-2} . The learning rate decays linearly in each epoch with a step of $(10^{-2} - 10^{-4})/\text{Epochs}$. We trained the networks for 30 epochs. In each epoch, we divide the training dataset into minibatches of 50 images each.

4.4. GA-Based Ensemble of Classifiers. Each classifier produces a class probability vector (i.e., output of the “softmax”

layer), $C_1 \dots C_N$, such that $C_i \in \mathbb{R}^{10}$ is a vector having 10 probabilities (for 10 distraction classes) and N is the number of classifiers. In a majority voting system as in (4), it is assumed that all experts (i.e., classifiers) can equally contribute to a better decision by taking the unweighted sum of all classifier outputs.

$$C_{\text{Majority}} = \frac{1}{N} \sum_i^N C_i \quad (4)$$

However, that is not usually a valid assumption. In a weighted voting system as in (5), we assume that classifiers do not contribute equally to the ensemble and that some classifiers might yield higher accuracy than others. Therefore, there is a need to estimate the weights of each classifier’s contribution to the ensemble. Reference [53] presents a variety of methods to estimate the weights. We opted to use a genetic algorithm (i.e., a search-based method).

$$C_{\text{Weighted}} = \frac{1}{\sum_i^N w_i} \sum_i^N w_i \cdot C_i \quad (5)$$

In our genetic algorithm, a chromosome consists of N genes that correspond to the weights $w_1 \dots w_N$. Our fitness function evaluates the Negative Log Likelihood (NLL) loss over a 50% random sample of the population. This helps prevent overfitting. Our population consists of 50 individuals. In each iteration, we retain the top 20% of the population and use them as parents. Then, we randomly select 10% of the remaining 80% of the population as parents. In other words, we have 30% of the population as parents. Now, we randomly mutate 5% of the selected parents. Finally, we cross-over random pairs of the parents to produce children until we have a full population (i.e., with 50 individuals). We ran the above procedure for only 5 iterations in order to avoid overfitting. We selected the chromosome with the highest fitness score (test against all data points, not 50%).

TABLE 1: Distracted driver posture classification results on randomly selected test data.

Model	Source	Loss (NLL)	Accuracy (%)
AlexNet	<i>Original</i>	0.3909	93.65
	Skin Segmented	0.3468	93.62
	Face	1.0516	84.28
	Hands	0.6186	89.52
	Face + Hands	0.8298	86.68
InceptionV3	<i>Original</i>	0.2654	95.17
	Skin Segmented	0.2903	94.66
	Face	0.6096	88.82
	Hands	0.4546	91.62
	Face + Hands	0.4495	90.88
AlexNet		0.2727	94.29
<i>Majority Voting Ensemble</i>		0.1661	95.77
<i>GA-Weighted Ensemble</i>		0.1575	95.98

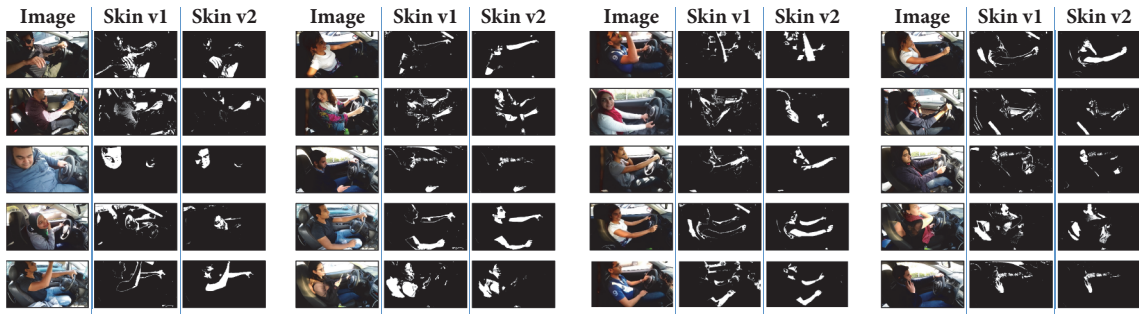


FIGURE 5: Our skin segmentation algorithm (v1 & v2) is run against sample test images. “Skin v2” is trained on pixel spatial information, while “Skin v1” is only trained on pixel color information.

5. Experiments

We divided our dataset into 75% for training and 25% held out as test data using random sampling. Then, we ran the skin segmentation, face and hand detectors on the entire dataset. We tested different deep networks against our test dataset and obtained the results reported in Table 1. We notice that both AlexNet and InceptionV3 achieve best accuracies when trained on the original images. However, the accuracy does not majorly change in both architectures when switching from the original images to skin-segmented images. Hands seem to have more weight in posture recognition than the face. “Face+hands” images produce slightly lower accuracy than the hands images alone, yet still higher than the face images alone. That happens due to face/hand detector failures. For example, if a hand is not found, we pass a face image to a “face+hands” classifier. This does not happen in individual cases of hand-only or face-only classifier because if the hand/face detection fails, we pass the original image to the hand/face classifier as a fallback mechanism. With better hand/face detectors, the “face+hands” networks are expected to produce higher accuracies than the “hands” networks. The confusion matrix of our genetically weighted ensemble of

classifiers on this randomly selected test data is shown in Table 2, and the results are shown in Table 1.

In Figure 5, we show sample results of our skin segmentation algorithm. Test data are never seen by the skin segmentation component during training. In the “Skin v2” column, the classifier is trained on pixels’ spatial information. In the “Skin v1” column, it is only trained on pixels color information. In addition to such qualitative evaluation for our skin detection method, in Table 3 we report a quantitative comparison of distraction classification accuracy by training on skin images generated with and without using pixels’ spatial information. An AlexNet is trained and tested on skin-segmented images.

Table 3 first row result is for a model trained on images generated without spatial information. In the third row, active learning is used to supervise learning pixels’ spatial information with pixels generated from 8,662 cherry-picked skin-segmented images. 100 pixels are randomly sampled from each image to provide an annotated training dataset of 866,200 pixels. A third model is trained on only 50% of these pixels and the result is reported in the second row. The spatial information is proven to improve the skin detection accuracy, while the overall method performance remains

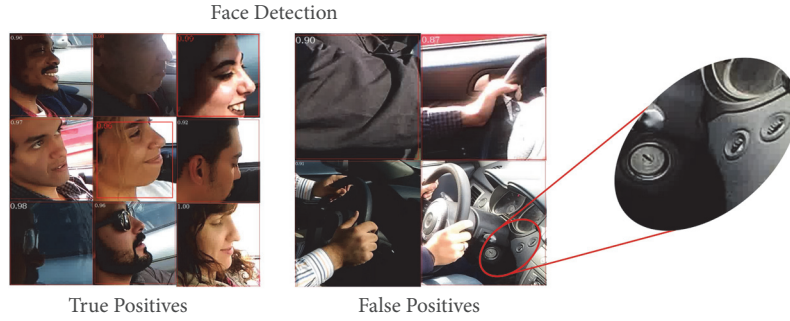


FIGURE 6: Non-frontal faces detection is challenging. A correctly detected face and a wrongly detected one by [14]. Air-conditioning openings are wrongly detected as a face.

TABLE 2: Confusion matrix of our genetically weighted ensemble of classifiers on randomly selected test data.

		Predicted									
		C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
Actual	C0	95.34	0	0.33	0.65	0.11	0.43	0.43	0.87	0.11	1.74
	C1	0.31	96.63	1.23	0.31	0.92	0	0.31	0	0.31	0
	C2	0.29	3.23	96.48	0	0	0	0	0	0	0
	C3	2.02	0.61	0	96.15	0.81	0	0.20	0	0	0.20
	C4	0	0.33	0	4.90	94.77	0	0	0	0	0
	C5	4.26	0	0	0.33	0	95.08	0	0	0	0.33
	C6	0.74	0	0	0.25	0	0.74	98.01	0.25	0	0
	C7	3.65	0	0	0	0	0	0	95.35	0	1.00
	C8	3.79	0	0	0	0	0	1.38	0.34	92.76	1.72
	C9	1.40	0	0	0	0	0	0.47	0.31	0.16	97.67

TABLE 3: Distraction classification results by training (AlexNet) and testing on skin segmented images.

Network	Accuracy [%]
No pixels' spatial info	93.14
Spatial info learned through active learning with 50% of data	93.53
Spatial info learned through active learning with 100% of data	93.62

TABLE 4: Distracted driver posture classification results on test data with unique drivers.

Network	Loss (NLL)	Accuracy [%]
VGG-16	1.246614	76.131
Resnet50	0.661483	81.695
InceptionV3	0.640018	90.068

unsatisfactory and needs better generalization against the varying lighting conditions happening during driving. Given that, in addition to several misdetections noted from the face (as in Figure 6) and hand detectors, we conclude that further work is needed to improve the accuracy of detection of these visual elements (skin, face, and hands). At the moment they

just add to the system complexity without improving the overall distraction identification accuracy.

The train/test data split presented in Table 1 causes a high correlation between training and testing data. Thus, it makes the distraction detection a much easier problem. This split is also followed in [54]. To present a more realistic setup, we adopt a more challenging strategy. We separate the data based on driving sessions rather than at random. In other words, test data split contains different drivers, cars, times of day, lighting conditions, driving conditions, and so on. Out of 44 drivers in our dataset, the training contains 38 drivers (12,555 samples), and the test data contains 6 drivers (1,923 samples).

The results of our genetically weighted ensemble of classifiers on the new split-by-driver test data is shown in Table 4. The confusion matrix is shown in Table 5. InceptionV3 achieves the best distraction identification accuracy of 90%. Figure 7 presents saliency maps generated for sample test images. The gradient of the winning distraction class is calculated and applied to input image. The gradient highlights the salient regions, i.e., the ones causing the most change and contributing to the network's decision. The drawn salient regions clearly show that our network is making its decision based on relevant features in the input camera images (i.e., hands, faces, neck, radio, driving wheel, and so on).

In a static image, a driver would appear in a "safe driving" posture, while contextually he/she might be distracted by doing some other activity. Making the distraction posture

TABLE 5: Confusion matrix of InceptionV3 network on test data with unique drivers.

		Predicted									
		C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
Actual	C0	87.57	10.98	0	0.29	0	0	0.29	0.86	0	0
	C1	0	96.71	0	1.88	0	0	0	1.41	0	0
	C2	0	0	97.42	0	0	0	0	0.52	0.52	1.55
	C3	8.33	0	0	91.67	0	0	0	0	0	0
	C4	11.76	0	0	0	87.06	0	0	0	1.18	0
	C5	0	0	0	0	0	100	0	0	0	0
	C6	0.67	0	0	0	0	0	81.82	10.49	6.99	0
	C7	0	0	0	0	0	0	0	100	0	0
	C8	2.74	0	0.68	0	3.42	0	0	7.53	84.93	0.68
	C9	1.38	2.75	0	0	0	0	0	19.27	0	76.6



FIGURE 7: Saliency map demonstrates that our network makes its decision based on relevant features/regions in the input camera images.

identification decision based only on the current camera input image restricts the system from coping with such a problem. In this experiment, to classify a camera frame F_t at timestamp t , the network gives a decision C_t . After that, a final smoothed decision is deduced as the average of all

the preceding network decisions over the M seconds before t such that $C_{smoothed_t} = \sum_{i=t-M}^t C_i$.

In Figure 8, we evaluate the system accuracy at different values of M . $M = 1$ means that distraction posture identification decision is based only on the current camera input

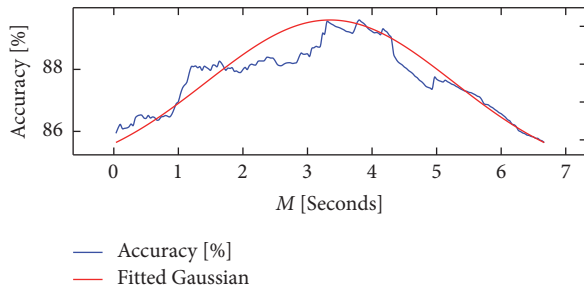


FIGURE 8: System accuracy evaluated at different values of history intervals M . At $M = 1$, current camera input image is solely used for distraction classification; no context is used. For $M > 1$, we average predictions of M frames to obtain distraction in the current frame. Up to a certain threshold, the more context incorporated in the prediction, the more accurate and robust the system. After that threshold, contextual information in the past is no longer relevant. Best accuracy is achieved at $M = 3.35$ seconds, which is the average of the fitted Gaussian curve.

image. A bell-curve-looking graph is obtained. It indicates that accuracy increases when more images from the past are considered. However, this happens until a threshold after which the past becomes irrelevant, thus, decreasing accuracy. The optimal value for M is found to be 3.35 seconds, which is the mean of a fitted Gaussian distribution to the results.

5.1. Analysis. In Table 2, we notice that the most confusing posture is the “safe driving”. This is due to the lack of temporal context in static images. In a static image, a driver would appear in a “safe driving” posture. However, contextually, he/she was distracted by doing some other activity. “Text Left” is mostly confused for “Talk Left” and vice versa. The same applies to “Text Right” and “Talk Right”. “Adjust Radio” is mainly confused for a “safe driving” posture. That is due to lack of the previously mentioned temporal context. Apart from safe driving, “Hair & Makeup” is confused for talking to passenger. That is because, in most cases, when drivers did their hair/makeup on the left side of their face, they needed to tilt their face slightly right (while looking at the frontal mirror). Thus, the network thought the person was talking to passenger. “Reach Behind” was confused for both talking to passenger and drinking. That makes sense as people tend to naturally look towards the camera while reaching behind. As for the drinking confusion, it is due to right-arm movement from the steering wheel to the back seat. A still image in the middle of that move could be easily mistaken for a drinking posture. “Drink” and “Talk to Passenger” postures were not easily confused with other postures as 98% and 97.67% of their images were correctly classified.

5.2. Real-Time System. The number of parameters in the deep models we adopted and benchmarked in 1 is huge for a real-time system. Our version of VGG16 [46], AlexNet [15], ResNet50 [45], and InceptionV3 [44] has 134.3M, 58.3M, 23.6M, and 21.8M parameters, respectively. Reference [54] adopts a real-time model that reduces the number of parameters in VGG16 to 15M, which is a marked improvement,

but is still a large number for a real-time system (in a self-driving car, for example). Hence we propose a version of NasNet Mobile [55] that reduces the number of parameters further to only 4.3M. An ensemble of two NasNet Mobile models [55] (original and skin-segmented networks) produce a satisfactory classification accuracy of 84.64%. Meanwhile, it still maintains a real-time performance on a CPU-based system.

For NasNet Mobile models fusion, we use a Multilayer Perceptron (MLP) to learn a class-based weighted ensemble from data. The advantage of our MLP-based fusion over the weighted voting system is that it does not assume that some classifier is better in absolute terms. An MLP learns more sophisticated dependencies; one classifier can be more accurate in discriminating a set of distraction classes, while another can be more accurate on another set of distraction classes. This also can happen under specific status of the ensemble resultant softmax class probability distributions. In our adopted MLP-based fusion, the vectors $C_i \in \mathbf{R}^{10}$, where $i = 1, \dots, N$, are concatenated and passed to an MLP. Note that, each C_i is a vector having 10 probabilities (corresponding to 10 distraction classes). Our MLP loss function evaluates the Negative Log Likelihood (NLL) loss over the training data in addition to a regularization term to prevent overfitting.

6. Conclusion

Distracted driving is a major problem leading to a striking number of accidents worldwide. Its detection is an important system component in semiautonomous cars. In this paper, we presented a robust vision-based system that recognizes distracted driving postures. We collected a novel publicly available distracted driver dataset that we used to develop and test our system. Our best model utilizes a genetically weighted ensemble of convolutional neural networks to achieve a 90% classification accuracy. We aim to provide a baseline performance for future research to benchmark against. We also showed that a simpler model (only using AlexNet) could operate in real-time and still maintain a satisfactory classification accuracy. Face, hands, and skin detection proved to improve classification accuracy in our ensemble. However, in a real-time setting, their performance overhead is much higher than their contribution.

In a future work, we need to devise a better face, hands, and skin detector. We would need to manually label hand and face proposals and use them to train a Fast-RCNN (or, any other object detector) to localize both faces and hands in one shot and evaluate it against our existing CNN-based localization method.

Data Availability

The data used to support the findings of this study are available from the corresponding authors upon request. Our Distracted Driver dataset is publicly available at the project website link: https://heshameraqi.github.io/distracted_driver_detection.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the German Research Foundation (DFG) and the Technical University of Munich within the Open Access Publishing Funding Programme.

Supplementary Materials

A video showing our proposed method accuracy in runtime. The probabilities for the top 3 predicted classes are displayed along with the system input camera video. (*Supplementary Materials*)

References

- [1] Y. Liang, *Detecting Driver Distraction [Theses and Dissertations]*, 2009.
- [2] A. Downs, *Why Traffic Congestion Is Here to Stay... And Will Get Worse*, ACCESS Magazine, 2004.
- [3] W. H. Organization, "World health statistics 2017: monitoring health for the sdgs, sustainable development goals," 2017.
- [4] G. W. H. Organization, "Global health estimates 2015: Deaths by cause, age, sex, by country and by region, 2000-2015," 2016, http://www.who.int/healthinfo/global_burden_disease/estimates/en/index1.html.
- [5] D. Strickland, "How autonomous vehicles will shape the future of surface transportation," *Testimony of the Honorable David L. Strickland Administrator National Highway Traffic Safety Administration House Committee on Transportation and Infrastructure Subcommittee on Highways and Transit*, 2013.
- [6] J. D. Lee, "Driving safety," *Reviews of Human Factors and Ergonomics*, vol. 1, no. 1, pp. 172–218, 2016.
- [7] T. M. Pickrell, H. R. Li, and S. KC, "Traffic safety facts," 2016, <https://www.nhtsa.gov/risky-driving/distracted-driving>.
- [8] U. D. o. H. & H. Services, "Distracted Driving," 2016, <https://www.cdc.gov/motorvehiclesafety/distracted-driving/>.
- [9] A. Eriksson and N. A. Stanton, "Takeover time in highly automated vehicles: Noncritical transitions to and from manual control," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 59, no. 4, pp. 689–705, 2017.
- [10] H. M. Eraqi, M. N. Moustafa, and J. Honer, "End-to-end deep learning for steering autonomous vehicles considering temporal dependencies," in *Proceeding of the Machine Learning for Intelligent Transportation Systems Workshop in the 31st Conference on Neural Information Processing Systems (NIPS)*, Montreal, Canada, December 2017.
- [11] H. M. Eraqi, J. Honer, and S. Zuther, "Static free space detection with laser scanner using occupancy grid maps authors," in *Proceeding of the 20th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Yokohama, Japan, October 2017.
- [12] H. M. Eraqi, Y. Emad Eldin, and M. N. Moustafa, "Reactive collision avoidance using evolutionary neural networks," in *Proceedings of the 8th International Conference on Evolutionary Computation Theory and Applications*, vol. 1, pp. 251–257, SciTePress, Porto, Portugal, November 2016.
- [13] A. Fernández, R. Usamentiaga, J. L. Carús, and R. Casado, "Driver distraction using visual-based sensors and algorithms," *Sensors*, vol. 16, no. 11, article no 1805, 2016.
- [14] S. S. Farfadi, M. J. Saberian, and L. Li, "Multi-view face detection using deep convolutional neural networks," Cornell University Library, 2015, <https://arxiv.org/abs/1502.02766>.
- [15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.
- [16] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1734–1747, 2016.
- [17] O. Russakovsky, J. Deng, H. Su et al., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [18] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014*, pp. 2155–2162, USA, June 2014.
- [19] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.
- [20] Y. Abouelnaga, "Action annotation tool," 2017, <https://github.com/devyhia/action-annotation>.
- [21] R. A. Berri, A. G. Silva, R. S. Parpinelli, E. Girardi, and R. Arthur, "A pattern recognition system for detecting use of mobile phones while driving," in *Proceedings of the 9th International Conference on Computer Vision Theory and Applications, VISAPP 2014*, vol. 2, pp. 411–418, IEEE, Portugal, January 2014.
- [22] Y. Artan, O. Bulan, R. P. Loce, and P. Paul, "Driver cell phone usage detection from HOV/HOT NIR images," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, (CVPRW '14)*, pp. 225–230, USA, June 2014.
- [23] C. Craye and F. Karray, "Driver distraction detection and recognition using RGB-D sensor," 2015, <https://arxiv.org/abs/1502.00250>.
- [24] X. Zhang, N. Zheng, F. Wang, and Y. He, "Visual recognition of driver hand-held cell phone use based on hidden CRF," in *Proceedings of the 2011 IEEE International Conference on Vehicular Electronics and Safety, ICVES 2011*, pp. 248–251, China, July 2011.
- [25] T. H. Le, Y. Zheng, C. Zhu, K. Luu, and M. Savvides, "Multiple scale faster-rcnn approach to driver's cell-phone usage and hands on steering wheel detection," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 46–53, IEEE, Las Vegas, Nev, USA, June 2016.
- [26] N. Das, E. Ohn-Bar, and M. M. Trivedi, "On performance evaluation of driver hand detection algorithms: challenges, dataset, and metrics," in *Proceedings of the 18th IEEE International Conference on Intelligent Transportation Systems (ITSC '15)*, pp. 2953–2958, Spain, September 2015.
- [27] K. Seshadri, F. Juefei-Xu, D. K. Pal, M. Savvides, and C. P. Thor, "Driver cell phone usage detection on strategic highway research program (SHRP2) face view videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, (CVPRW '15)*, pp. 35–43, USA, June 2015.

- [28] S. Martin, E. Ohn-Bar, A. Tawari, and M. M. Trivedi, "Understanding head and hand activities and coordination in naturalistic driving videos," in *Proceedings of the 25th IEEE Intelligent Vehicles Symposium, IV 2014*, pp. 884–889, IEEE, USA, June 2014.
- [29] E. Ohn-Bar, S. Martin, and M. M. Trivedi, "Driver hand activity analysis in naturalistic driving studies: challenges, algorithms, and experimental studies," *Journal of Electronic Imaging*, vol. 22, no. 4, Article ID 041119, 2013.
- [30] E. Ohn-Bar, S. Martin, A. Tawari, and M. M. Trivedi, "Head, eye, and hand patterns for driver activity recognition," in *Proceedings of the 22nd International Conference on Pattern Recognition (ICPR)*, pp. 660–665, Stockholm, Sweden, August 2014.
- [31] E. Ohn-Bar and M. Trivedi, "In-vehicle hand activity recognition using integration of regions," in *Proceedings of the 2013 IEEE Intelligent Vehicles Symposium, IEEE IV 2013*, pp. 1034–1039, Australia, June 2013.
- [32] C. H. Zhao, B. L. Zhang, J. He, and J. Lian, "Recognition of driving postures by contourlet transform and random forests," *IET Intelligent Transport Systems*, vol. 6, no. 2, pp. 161–168, 2012.
- [33] C. Zhao, Y. Gao, J. He, and J. Lian, "Recognition of driving postures by multiwavelet transform and multilayer perceptron classifier," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 8, pp. 1677–1686, 2012.
- [34] C. Zhao, B. Zhang, J. Lian, J. He, T. Lin, and X. Zhang, "Classification of driving postures by support vector machines," in *Proceedings of the Sixth International Conference on Image and Graphics (ICIG '11)*, pp. 926–930, Hefei, China, August 2011.
- [35] C. H. Zhao, B. L. Zhang, X. Z. Zhang, S. Q. Zhao, and H. X. Li, "Recognition of driving postures by combined features and random subspace ensemble of multilayer perceptron classifiers," *Neural Computing and Applications*, vol. 22, no. 1, pp. 175–184, 2013.
- [36] C. Yan, F. Coenen, and B. Zhang, "Driving posture recognition by joint application of motion history image and pyramid histogram of oriented gradients," *International Journal of Vehicular Technology*, vol. 2014, Article ID 719413, 11 pages, 2014.
- [37] F. Coenen, B. Zhang, and C. Yan, "Driving posture recognition by convolutional neural networks," *IET Computer Vision*, vol. 10, no. 2, pp. 103–114, 2016.
- [38] I. Sultan, "Academic purposes?" 2016, <https://www.kaggle.com/c/state-farm-distracted-driver-detection/discussion/20043#117982>.
- [39] S. Yan, Y. Teng, J. S. Smith, and B. Zhang, "Driver behavior recognition based on deep convolutional neural networks," in *Proceedings of the 12th International Conference on Natural Computation and 13th Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pp. 636–641, Changsha, China, August 2016.
- [40] T. A. U. in Cairo (AUC), "Auc distracted driver dataset," 2019, https://heshameraqi.github.io/distraction_detection.
- [41] Y. Abouelnaga, H. M. Eraqi, and M. N. Moustafa, "Real-time distracted driver posture classification," in *Proceedings of the Machine Learning for Intelligent Transportation Systems Workshop in the 32nd Conference on Neural Information Processing Systems (NeuroIPS)*, Long Beach, Calif, USA, December 2018.
- [42] ASUS, "Zenfone selfie smartphone (model zd551kl)," 2015, https://www.asus.com/eg-en/Phone/ZenFone_Selfie-ZD551KL/.
- [43] Sony, "Ds325 camera datasheet," 2015, https://www.sony-depthsensing.com/Portals/0/Download/WEB_20120907_SK_DS325_Datasheet_V2.1.pdf.
- [44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 2818–2826, July 2016.
- [45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 770–778, July 2016.
- [46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, <https://arxiv.org/abs/1409.1556>.
- [47] S. Phung, A. Bouzerdoum, and D. Chai, "Skin segmentation using color and edge information," in *Proceedings of the Seventh International Symposium on Signal Processing and Its Applications, 2003. Proceedings.*, pp. 525–528 vol.1, Paris, France, July 2003.
- [48] R. Bhatt and A. Dhall, "Skin segmentation dataset," *UCI Machine Learning Repository*, 2010.
- [49] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network approach for face identification," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5325–5334, Boston, Mass, USA, June 2015, http://users.eecs.northwestern.edu/~xsh835/assets/cvpr2015_cascnn.pdf.
- [50] M. Kostinger, P. Wohlhart, P. M. Roth, and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," in *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 2144–2151, IEEE, Barcelona, Spain, November 2011.
- [51] S. Bambach, S. Lee, D. J. Crandall, and C. Yu, "Lending a hand: detecting hands and recognizing activities in complex egocentric interactions," in *Proceedings of the 15th IEEE International Conference on Computer Vision, (ICCV '15)*, pp. 1949–1957, Chile, December 2015.
- [52] Tensorflow, "TensorFlow-Slim image classification library," 2018, <https://github.com/tensorflow/models/tree/master/slim>.
- [53] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.
- [54] B. Baheti, S. Gajre, and S. Talbar, "Detection of distracted driver using convolutional neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1032–1038, IEEE, Salt Lake City, UT, USA, June 2018.
- [55] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *Computer Vision and Pattern Recognition*, vol. 2, no. 6, 2017, <https://arxiv.org/abs/1707.07012>.

Research Article

Public Transport Driver Identification System Using Histogram of Acceleration Data

Nuttun Virojboonkiate , **Adsadawut Chanakitkarnchok**,
Peerapon Vateekul, and **Kultida Rojviboonchai** 

Chulalongkorn University Big Data Analytics and IoT Center (CUBIC), Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand

Correspondence should be addressed to Kultida Rojviboonchai; kultida.r@chula.ac.th

Received 1 October 2018; Revised 19 December 2018; Accepted 15 January 2019; Published 5 February 2019

Guest Editor: Mohammad H. Y. Moghaddam

Copyright © 2019 Nuttun Virojboonkiate et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper introduces a driver identification system architecture for public transport which utilizes only acceleration sensor data. The system architecture consists of three main modules which are the data collection, data preprocessing, and driver identification module. Data were collected from real operation of campus shuttle buses. In the data preprocessing module, a filtering module is proposed to remove the inactive period of the public transport data. To extract the unique behavior of the driver, a histogram of acceleration sensor data is proposed as a main feature of driver identification. The performance of our system is evaluated in many important aspects, considering axis of acceleration, sliding window size, number of drivers, classifier algorithms, and driving period. Additionally, the case study of impostor detection is implemented by modifying the driver identification module to identify a car thief or carjacking. Our driver identification system can achieve up to 99% accuracy and the impostor detection system can achieve the F1 score of 0.87. As a result, our system architecture can be used as a guideline for implementing the real driver identification system and further driver identification researches.

1. Introduction

Twenty billion units is the number of the Internet of Things (IoT) that is predicted to be installed in 2020 [1–3]. Technology of sensors and communication devices dramatically increases this number of connected things. One of the key connected things is the connected vehicle. It is reported that there will be a quarter billion connected vehicles on the road in 2020 [4]. Nowadays, consumer vehicles are equipped with a lot of sensors in order to provide driver-assistant services, such as adaptive cruise control, automatic parking, and automatic emergency breaking. These sensors in vehicles lead to an advancement in transportation research. One of the popular research topics is the study of the driving style to detect the driver's behavior. The survey of these researches has already been discussed thoroughly in [5]. Apart from detecting driving style, if a driver has his own driving style, the driver himself could also be identified. This leads to the driver identification research that can automatically identify an individual driver using sensors in the vehicle.

There are a lot of situations where a vehicle is shared by more than one person. These situations can occur when the vehicle is a city bus, a company van, or even a consumer car. The driver identification system has four significant points of benefit. First, it helps companies or governments know the actual driver of their vehicles. Until now, to recognize the driver, one could use some simple approaches such as recorded notes or an identification card. One could use even more sophisticated approaches such as a fingerprint scanner or face detection. However, by using sensors in a vehicle, the identification system can achieve the goal easier because it can automatically identify the drivers while they are driving the vehicle. Moreover, an identification card or a fingerprint system can be cheated easily by replacing the identifying item. Real-time face detection using a video camera can solve the cheating problem. However, the drivers might not feel happy and comfortable to be observed by the camera all the time. By using the driver identification detected by sensors, the driver will not feel being watched by the camera. More importantly, the drivers cannot allow other persons

to drive on their behalf. This is because it is difficult to copy the driving behavior. The second benefit of the driver identification system is that it can be applied to a car thief or carjacking detection. According to an Interpol statistic [6], more than seven million vehicles have been stolen almost every year since 2012. With the hidden sensors and the driver identification system, a thief will not even know that he is being detected. The system can immediately send a carjacking notification to the owner of the vehicle or the police. In this situation, a GPS module can also be used to detect the vehicle location. The third benefit of the driver identification system is that it can help in adjusting the environment based on the driver's preferences. For instance, once the system detects the specific driver, it can control the air temperature, songs, and seat position based on the preferences of this driver. The fourth benefit is that it can be used in the insurance business. As drivers have different behaviors causing different risk levels, insurance companies can adjust charges of insurance according to the drivers and their driving styles.

There have recently been several proposed researches for the driver identification system, which can be classified into two categories. The former is the research using multiple sensors [7–17]. The latter is the research using a single sensor [18–20].

For the first category, in [7–9], multiple external sensors were installed inside the vehicle. These sensors included GPS, gas and brake pedal force, velocity, steering angle, and engine speed. The vehicle data was analyzed by a spectral and Gaussian mixture model [7] and a neural network [8, 9]. These works achieved 75–95% of accuracy. As a Controller Area Network (CAN) bus became a signaling standard, many researchers attempted to exploit the sensor data from the CAN bus to perform driver identification [10–15]. In [10], the data of a single car driven by 15 volunteer participants was used. The variables such as time of day and radio station were controlled. An overlapped sliding window of the data was considered and optimized. There were many features evaluated including statistical features, descriptive features, and frequency features. The authors showed the best algorithm, the top sensor, and the top feature that can differentiate the drivers in the experiments. In [11], the driver identification method using only data obtained at a single turning event has been proposed. The authors classified turning locations into 12 types and conducted experiment for each of them. The model used 12 signals of the CAN bus in an Audi car driven by multiple drivers on real roads. The model could achieve 76.9% for two-driver classification and 50.1% for five-driver classification. In [12], driver identification with impostor detection has been proposed. The authors used an Extreme Learning Machine (ELM) to detect the impostor. The ELM had 8 input variables including data obtained from the CAN bus and other sensors. For driver identification, the accuracy was greater than 80% in all cases and still above 90% for 2–3-driver classification. For impostor detection, the accuracy was above 80% when a single genuine driver was considered. However, the accuracy decreased to about 50% when the number of authorized drivers increased. In [13], driver identification using vehicle acceleration and deceleration events has been proposed. The

data including duration, speed, acceleration, jerk, and path of defined acceleration and deceleration events were selected to extract statistical features. Linear Discriminant Analysis (LDA) was used as an algorithm to identify the drivers. Accuracy of up to 80% can be achieved. In [15], driver identification was conducted using vehicle data without a feature extraction process and accuracy of up to 99% was achieved. In [14], three open data were evaluated in the proposed model. The minimum training and testing time required to achieve a desired accuracy were also investigated. Apart from using data from the CAN bus, a different method has been proposed in [16]. Instead of using real-time data like other previous work, the trip-based data obtained from an In-Vehicle Data Recorder (IVDR) was exploited to identify the driver. With the proposed machine learning pipeline, the average accuracy of 88% can be achieved.

Although using data obtained from multiple sensors via the CAN bus can help achieve high accuracy, there are still a large number of vehicles that are not compatible with the CAN bus standard. Moreover, using multiple external sensors suffers from installation, data processing, implementation, and computation. Therefore, a number of recent approaches have been proposed to use only data obtained from a single sensor [18–20]. In [20], GPS data from a smartphone is used to identify the driver by deriving other data from the location such as acceleration, angular speed, and jerk. A Random Forest was used to identify the driver from statistical features extracting from those derived data. The experiment was done in nine groups of drivers separately. Each group consisted of 4–5 drivers, who drove in a similar location and time. The average accuracy for all groups of 82.3% was obtained. Many aspects were mentioned such as number of drivers, classifier algorithms, and sliding window. Another approach using only a single sensor was proposed in [18]. In this work, only acceleration data was used by collecting from smartphones located in five campus buses for three months. The PCA (Principle Component Analysis) was done on statistical variables derived from the acceleration data, which is collected only for six days. The PCA showed some unique patterns for each driver. However, some drivers still had the same pattern of variables. Moreover, the analysis was done on the whole-day data, which is not practical for the real-time driver identification system. Nevertheless, this work reveals that the acceleration data has potential to identify the driver. Furthermore, the acceleration sensor is low cost and low energy consumption and can directly represent the vehicle's accelerating behavior of the driver. Thus, in our previous work [19], we proposed driver identification using a histogram of acceleration. We selected the histogram as our main feature. This is because the histogram can tolerate noise of data much more than other statistical features. The data was collected from 13 campus buses for 10 months. A neural network was used to identify the driver from the histogram. The identification was done separately for four periods of time of a day to reduce traffic variations. Using this method, 77–88% of accuracy was achieved. Although all of the abovementioned works use only data obtained from a single sensor and provide promising results, some important issues still need to be addressed for public transport driver

identification. First, the data preprocessing method can be improved because of the characteristic of the public vehicle, which frequently stops along the trip. Second, practical issues need to be addressed such as the traffic environment, number of drivers, detection time, and training period. Third, a case study such as impostor detection should be implemented and evaluated to show the possibility and the importance of driver identification.

Therefore, in this paper, we extend our previous work in many aspects. First, we introduce the system architecture for driver identification. This system is simple to implement and apply to any type of vehicles. Second, the data preprocessing method is improved. Specifically, we observe that there exists a large amount of data recorded when the vehicles are not moving. Eliminating such data before the processing can significantly increase the accuracy. This is because public vehicles normally stop at every service station and such data does not represent the drivers' behaviors. With our new data preprocessing method, the accuracy of our system can greatly improve from 77-88% to 94-99%. Third, the experiment has been conducted by using a random time of the day to identify the driver regardless of the traffic environment. Fourth, we extensively investigate variables that could affect the accuracy. These variables include axis of data, sliding window, overlapping technique, classifiers, number of drivers, and driving period. Fifth, we show that our system can be applied to detect an impostor. Thus, it can be beneficial for detecting a car thief or carjacking.

The rest of this paper is organized as follows: the details of related work are provided in Section 2. In Section 3, we propose our system architecture, which consists of the data collection module, data preprocessing module, and driver identification module. In Section 4, the case study of impostor detection is explained by modifying the identification model. In Section 5, we evaluate the performance of our system in various aspects including the inactive-period filtering module, axis of acceleration, sliding window and overlapping technique, classifiers, number of drivers, and impostor detection. The conclusion of the work is provided in Section 6.

2. Related Work

In this section, we thoroughly discuss the researches that utilize data obtained from a single sensor to perform driver identification. Chowdhury et al. proposed a driver identification method using only GPS data [20]. Other data such as speed, acceleration, jerk, and jerk energy were computed from the GPS data. Then, statistical features of those data were extracted such as mean, median, skewness, kurtosis, standard deviation, min, max, and some percentile values. After analysis of variance (ANOVA), 137 features were then selected to be input of the Random Forest. The experiment involved 38 drivers, but they were grouped by location and timing to reduce external variables such as traffic. Sliding window and the overlapping technique were also considered. Average accuracy of 82.3% was achieved for five-driver identification. Moreover, by analyzing driving skill and

aggression, Chowdhury et al. concluded that skilled-driver identification tends to provide higher accuracy.

Phumphuang et al. proposed a driver identification model that utilized only acceleration data [18]. The data had been collected for 3 months and involved 5 drivers. Descriptive statistics such as average, standard deviation, differential, and correlation were computed from the lateral and longitudinal axis of the acceleration data. Then, the PCA (Principle Component Analysis) method was used to select proper variables. With some criteria and a set of variables, this method can correctly identify all 5 drivers. This result shows a possibility to identify an individual driver using only acceleration data.

Enev et al. [10] who used multisensors from the CAN bus to identify the driver also mentioned the result using only the single top sensor. The utilization of the brake position data that is the top sensor can identify the driver with the accuracy of 87.33% from the first part of the data and 100% for the second part of the data.

In our previous work [19], a driver identification method using only acceleration data has been proposed. The data involved 13 drivers who drove their own campus buses for 10 months. The histogram of acceleration illustrated that it can represent the unique driving behavior of an individual driver. The experiment was conducted on four periods of time of a day to reduce the traffic variable. By using the histogram of acceleration as the feature of the model, up to 88% of accuracy was achieved. However, our previous work has not considered the unique characteristic of public transport. In particular, public transport drivers normally stop their buses at every bus station. This should be taken into consideration for the data preprocessing module. Moreover, in our previous work, a number of aspects were overlooked including the sliding window size, overlapping technique, and amount of data. In this paper, all of the abovementioned issues have been considered and accuracy of up to 99% has been achieved. The system architecture has been proposed especially for public transport. Table 1 shows comparison between our proposed system and other existing work.

3. System Architecture

Our system architecture consists of three main modules. The first module is the data collection module, which is packed in one box as shown in Figure 1. We collaborated with the campus buses' company to attach each of our boxes in 13 buses. Each driver is assigned to drive a single bus. The buses are electrical vehicles as shown in Figure 2, supplying power directly to the box. Figure 3 depicts our system architecture overall. The data collection module sends the data obtained from the sensor to our server through 3G. The data is stored in the database system before being processed to the second module. The second module is the data preprocessing module, which is responsible for cleaning raw data and extracting features from it. Then, the third module, the driver identification module, identifies the driver from those processed data. The details of each module are explained in the following subsections.

TABLE 1: Comparison between our proposed system and other existing work.

Author	Sensor used for driver identification	Data collection period	Number of drivers	Features	Inactive-driving filtering	Classifier	Accuracy of driver identification (%)	Practicability for impostor detection
Our proposed system with the new data preprocessing	Acceleration sensor	10 months	3-13	Histogram of acceleration	✓	Neural network	94-99	F1 score 0.87 using KNN classifier ✓
Our previous work [19]	Acceleration sensor	10 months	3-13	Histogram of acceleration	✗	Neural network	74-88	✗
Phumphuang et al. [18]	Acceleration sensor	3 months	5	Statistical features with PCA	✗	✗	60-100	✗
Chowdhury et al. [20]	GPS	2 months (1223 hours)	4-5	Statistical features	✗	Random Forest	~ 82.3	✗
Enev et al. [10]	Brake paddle position	~ 3 hours/driver	15	Statistical, descriptive, and frequency features	✗	Random Forest	87-100	✗

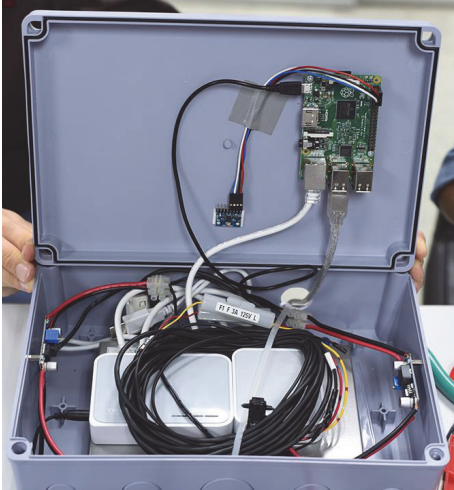


FIGURE 1: Equipment box attached in each vehicle.



FIGURE 2: Campus buses used in the experiment.

3.1. Data Collection Module. Before processing and analyzing data, the data collection module is considered for ensuring integrity. The overall data collection module is shown in Figure 4. The data collection module can be divided into 2 main processes: data producing and data processing:

- (i) *Data producing* consists of 2 sensors, which are the MPU-6050 module [21] and GPS module. The MPU-6050 sensor module contains a 3-axis accelerometer. The accelerometer allows 1 kHz sampling rates from 2g to -2g with the sensitivity of 16,384 LBS/g. This sensor module is horizontally placed in the box at the middle front panel of the bus. As illustrated in Figure 5, the longitudinal, lateral, and vertical movement are recorded on the x-, y-, and z-axis, respectively. Next, the GPS module provides the location of the shuttle bus. The GPS module is placed in the center of the metal roof of the shuttle bus, which is away from radio antennas to avoid interference. Note that GPS data is not used for driver identification. It is only used

for determining the stopping period of the bus in the data preprocessing module

- (ii) *Data processing* is done by Raspberry Pi, which is a low-cost and high-performance computer. It is equipped with an acceleration sensor module and GPS module for processing the raw data. The sampling rate of the acceleration sensor data and that of geolocation data are 90 Hz and 1 Hz, respectively. The sampling data is sent to the server via a 3G connection using a SIM card. Raspberry Pi communicates with the server by using the HTTP protocol. The data is stored in a structure database. In the storing process, data from the acceleration sensor module will be concatenated with the data from the GPS module. As the sampling rate of the data from the acceleration sensor module is more than the sampling rate of the data from the GPS module, the concatenated data will use the latest updated geolocation data with the time stamp

3.2. Data Preprocessing Module. The data preprocessing module is used to process raw acceleration data before being used as the input to the driver identification module. This module consists of two main submodules, which are the inactive-period filtering module and feature extraction module.

3.2.1. Inactive-Period Filtering Module. The characteristics of a public transport vehicle are different from a private vehicle. Normally, a public bus stops for a period in order to pick up passengers at bus stops. In this period, the driver does not expose any driving behavior through the acceleration data. We consider this period as the inactive period and it should be removed. We visualize a histogram of the stopping period as shown in Figure 6 to understand the characteristic of the public vehicle data. This figure shows the frequency of each stopping period. As illustrated in Figure 6, the bus mostly stops for less than five seconds. This indicates normal driving behaviors in light traffic. On the other hand, the bus hardly stops for longer than five seconds. As it is the long period, it takes a significant proportion in the overall data and should be removed as mentioned. Therefore, the inactive period is defined by the period in which location data is the same for more than five seconds. Algorithm 1 shows our proposed inactive-period filtering algorithm. After removing the inactive period, the remaining data will be called active driving data. This is a crucial step since it significantly increases the accuracy comparing to our previous work. Details of the results are provided in Section 5.1.

3.2.2. Feature Extraction Module. The input of the feature extraction module is the active driving data. Each axis of the acceleration is processed to a histogram. The histograms are represented in relative frequency so that they can be compared to one another. In our previous work, a histogram was built within four durations: morning, late-morning, afternoon, and evening. Each duration was evaluated separately to reduce traffic effect. However, in this paper, a

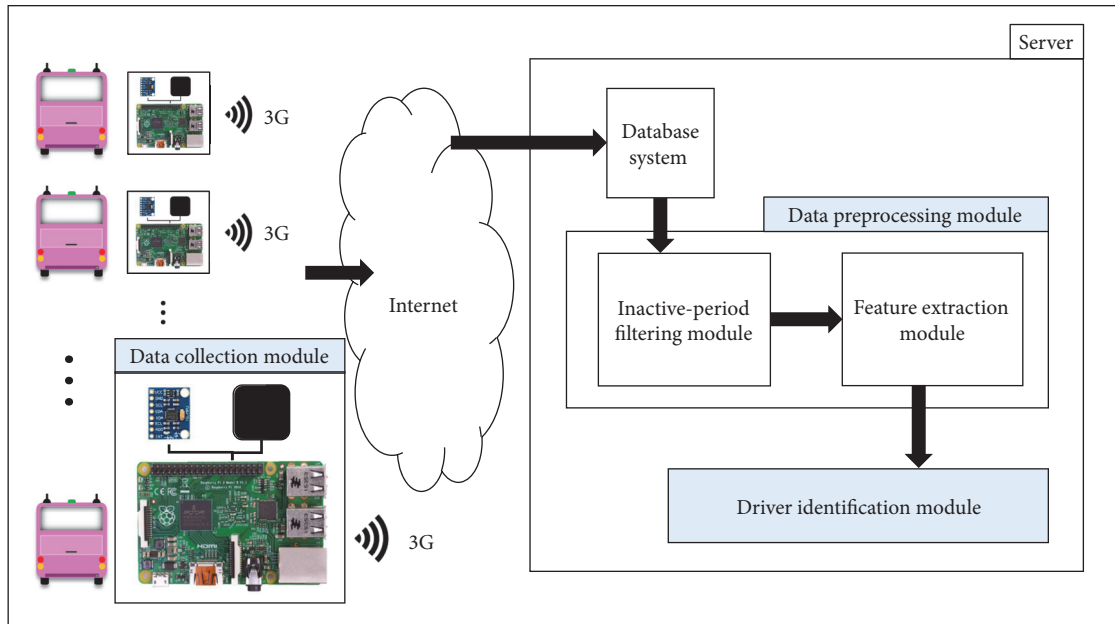


FIGURE 3: System architecture.

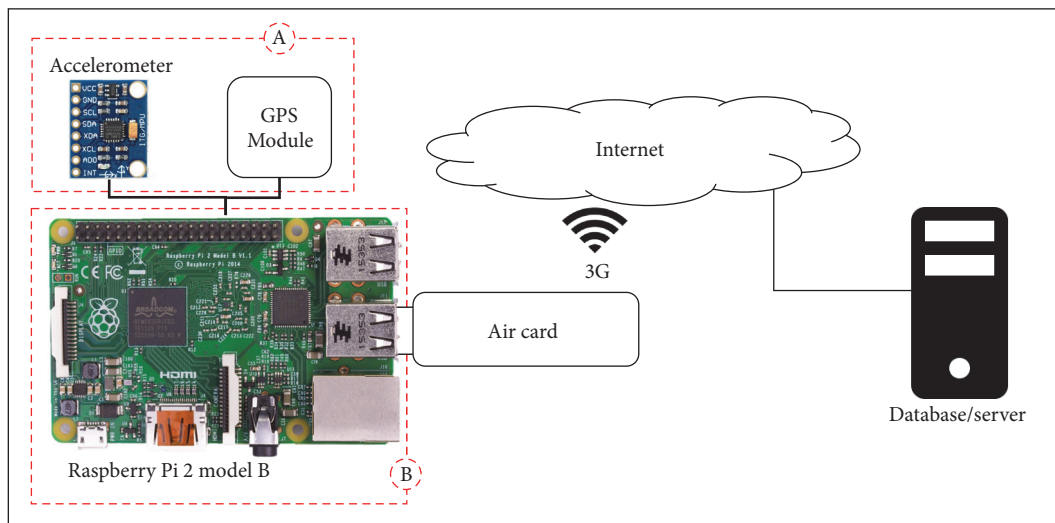


FIGURE 4: The data collection module, where A is a data producing part and B is a data processing part.

histogram is built from the acceleration data of a single driver within a sliding window. Data from a random time of the day was evaluated together so the system can be used more practically and realistically. To choose the sliding window size, the acceleration data should be collected for a proper period to allow the histogram representing the driving behavior. Moreover, the sliding window can be overlapped. This is also a crucial technique to improve the accuracy. Figure 7 depicts histogram extraction from the acceleration data. More details of the sliding window and overlapping technique will be evaluated and discussed in Section 5.3. The histogram extraction algorithm is shown in Algorithm 2. The min value, max value, and number of bins need to be set in the same way for all histograms. The min and max values should

be tested to cover most of the acceleration values. In this paper, the range of histogram value is set to -3.6 to 3.6 m/s^2 . The histogram has 100 bars as it is a proper value that can represent the curve of the histogram. Too low a value would be so rough that it cannot represent the behavior. Too high a value would cause unnecessary computation. The number of bars can also be optimized for a specific dataset.

However, after extracting the histograms, some of them provide uncommon curves shown in Figure 8. We called this the glitch of the histogram, which occurs in about 15% of all histograms. This glitch comes from a number of the same identical value of the acceleration data, which locates in a bin of the histogram. This happens because the acceleration sensor or Raspberry Pi may freeze and then record the same

```

Input: acceleration acc_raw,
latitude lat,
longitude lng
Output: acceleration (without inactive period) acc
1: for each acc_raw do:
2:   Initialize next_lat to next five-second of lat
3:   Initialize rateOfChange_lat to (next_lat - lat)/lat
4:   Initialize next_lng to next five-second of lng
5:   Initialize rateOfChange_lng to (next_lng - lng)/lng
6:   if rateOfChange_lat is zero and rateOfChange_lng is zero do:
7:     remove acc_raw
8:   end if
9: end for
10: return acc_raw as acc
    
```

ALGORITHM 1: Inactive-period filtering algorithm.

```

Input: acceleration acc,
sliding window size wind,
overlap percent ovrp
Output: histogram of acceleration hist
1: while acc remain:
2:   Initialize hist to histogram of acc for wind minutes
using fixed min, max, and bin size
3:   Change hist frequency to relative frequency
4:   Skip acc to the next (1-ovrp)*wind minutes
5: end while
6: return hist
    
```

ALGORITHM 2: Feature extraction algorithm.

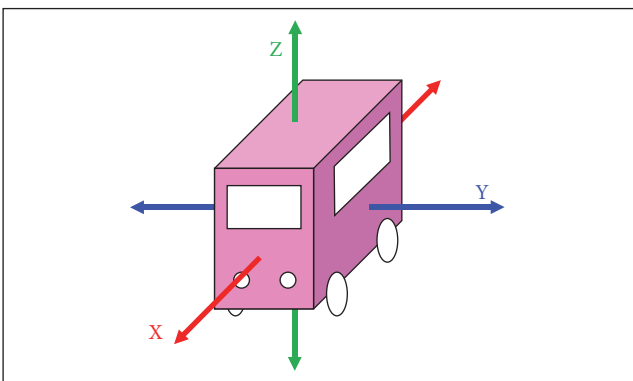


FIGURE 5: Vehicle movement represented in a coordinate system.

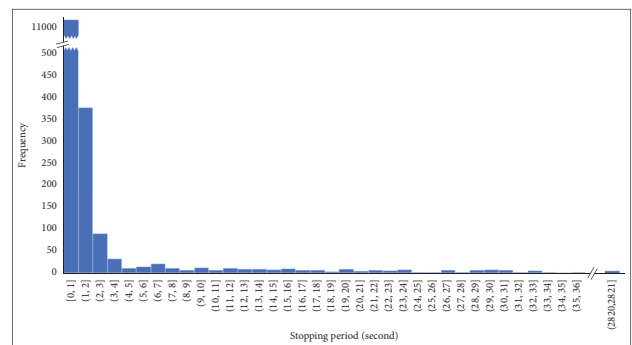


FIGURE 6: Histogram of inactive period.

acceleration value for a period of time. To solve this problem, the same identical value is removed from the histogram. The resulting histogram without the glitch is shown in Figure 9. The histogram is now ready for the next process.

3.3. Driver Identification Module. This section overviews the driver identification proposed in our previous work. After the histogram is properly built, it is used as an input of the classifier in order to identify the driver. In this paper, we

mainly focus on the neural network algorithm, but other classifier algorithms will also be discussed in Section 5.5. To construct the neural network model for training and testing, the *neuralnet* package in the Comprehensive R Archive Network (CRAN) is used. To configure the model, each layer of the neural network is set as follows:

- (i) The number of nodes in the input layer depends on the involved axis of acceleration. The histogram, which is the input of the neural network, has 100 bars for one axis. If one axis is involved, the input layer will

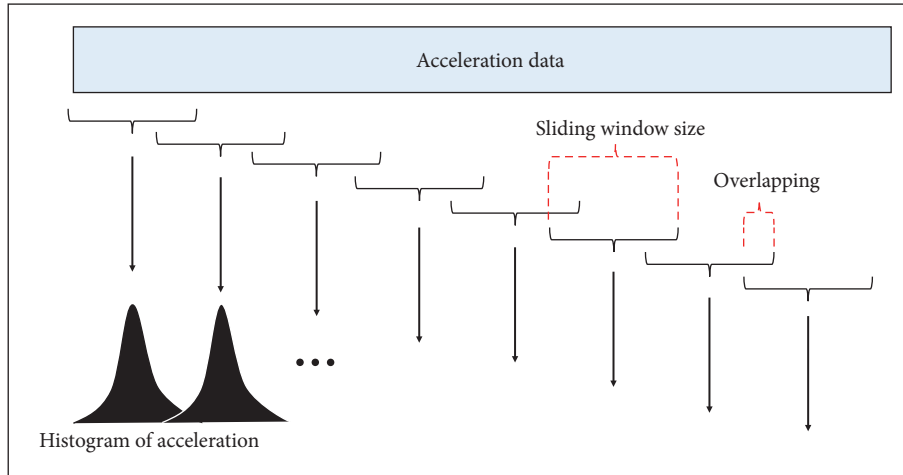


FIGURE 7: Histogram extraction from the acceleration data.

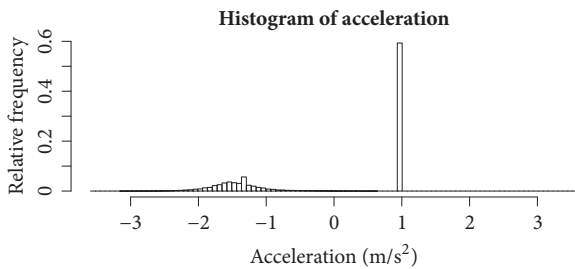


FIGURE 8: Example of a histogram that contains a glitch.

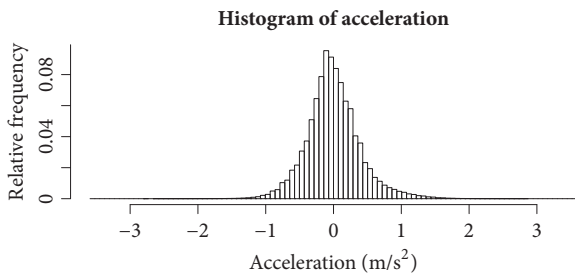


FIGURE 9: Example of a normal histogram.

have 100 nodes. The input layer will have 200 or 300 nodes if two or three axes are involved, respectively

- (ii) The number of hidden layers and number of nodes in the hidden layer need to be tuned. According to our experiment, increasing the number of hidden layers does not provide significant improvement in terms of accuracy. Moreover, it leads to more computation time. Therefore, in this paper, only one hidden layer is used. In the case of hidden nodes, according to our experiment, the optimal number of hidden nodes is one-third of the number of input nodes. To be specific, if three axes are involved, the number of hidden nodes will be 100

- (iii) The output layer represents the identified driver. Thus, the number of output nodes depends on the number of drivers. Specifically, if there are 13 drivers involved, the number of output nodes will be 13. When testing, the output node that has the highest value will be selected as the identified driver

To illustrate, we show the example of the neural network model that involves three axes and 13 drivers in Figure 10.

4. Case Study: Impostor Detection (Open Set Classification)

This section describes one use case of driver identification, which is the car thief or carjacking detection. Most driver identification researches only focus on identifying a driver in a closed set. This means an impostor driver who has never been trained in the model will be identified as one of the trained drivers. This is essential in a carjacking or car thief detection application because we have never had data of a thief to train the model beforehand. Thus, we extend the limitation of our model by adjusting only the identification module in order to identify the impostor. In a closed set of drivers, the output node of the neural network that has the maximum value will be selected as the identified driver. However, in an open set, a threshold needs to be calculated to separate an impostor from a genuine driver. The threshold is calculated for each driver by using validation data (not testing data and training data). After trying different techniques, the threshold that provides the highest accuracy is the 20th percentile of the output value in the validation data of the correctly identified drivers. Thus, in the testing step, the output value of the selected driver that is lower than the threshold will be rejected and judged to be an impostor.

However, the value from the output nodes of the neural network might not be suitable for calculating the threshold because the output value does not directly represent the

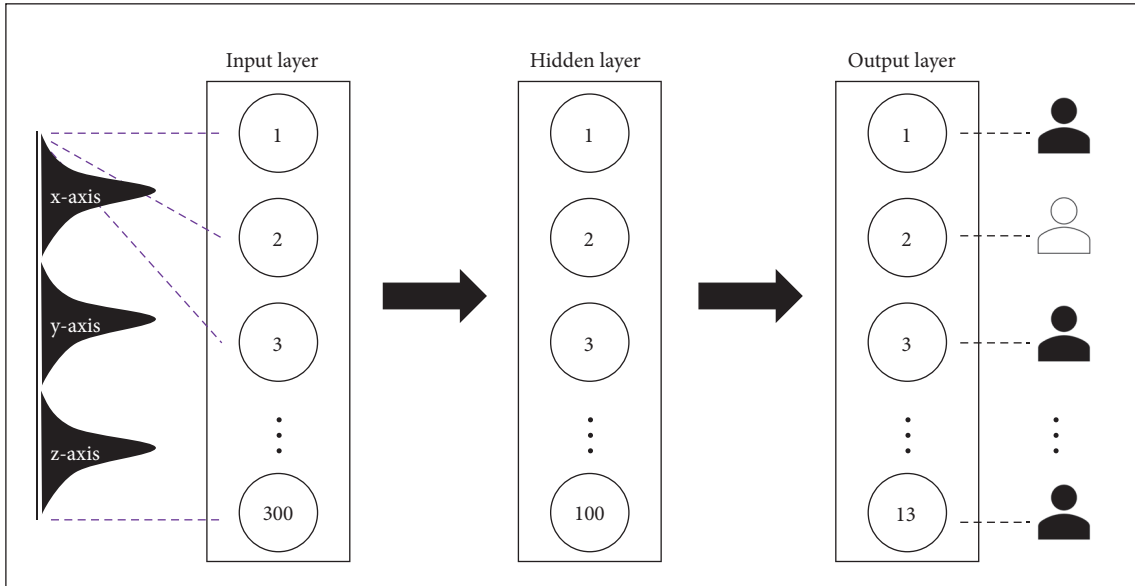


FIGURE 10: Example of the neural network model.

```

Input:    k constant in KNN
            validation data validateData
            training data trainData

Output:  threshold for each driver  $threshold_d$ 
1:          for each driver  $d$  do:
2:              $validateData_d = validateData$  that keep only driver  $d$ 
3:             for each  $validateData_d$  do:
4:                 $kList_d =$  find  $k$  nearest neighbor in  $trainData$ 
5:                 $topkList_d =$  keep maximum distance from  $kList_d$ 
6:             end for
7:              $threshold_d =$  average distance in  $topkList_d$ 
8:          end for

```

ALGORITHM 3: Threshold calculation for each driver using the KNN.

difference for each histogram. This leads to low impostor detection accuracy, the details of which will be provided in Section 5.7. Hence, we try to utilize the K-nearest neighbors (KNN) algorithm, which can specify the difference of histograms directly by calculating the distance between them. Like the utilization of the neural network, the validation data is still used to calculate the threshold for each driver. To do so, each histogram of the validation data will be used to calculate the Euclidian distance to other histograms in the training data to find the k -nearest neighbors. In this work, using $k = 17$ provides the highest accuracy. The threshold for each driver is calculated by averaging the maximum distance among the 17 nearest neighbors. Then, in the testing step, 17 new nearest neighbors will be also calculated. The majority of the drivers of those neighbors will be selected as the predicted driver if the closest distance is less than the driver's threshold. The threshold calculation is shown in Algorithm 3 and the impostor detection using the predefined threshold is shown in Algorithm 4.

5. Results and Discussion

This section aims at evaluating the driver identification system in many aspects including parameter tuning and robustness testing of the system. In Section 5.1, our proposed inactive-period filtering module is evaluated to show the improvement of the accuracy after the proposed module is applied. In Section 5.2, the importance of each axis is described together with the system evaluation. In Section 5.3, the optimal values of sliding window and overlapping percentage for the system are determined. Section 5.4 studies the effect of the number of drivers. In Section 5.5, the comparison between the neural network classifier and other classifier algorithms is shown and discussed. Section 5.6 studies the effect of the driving period. Section 5.7 shows the result when using our system to identify the impostor.

We evaluated the system by using the 10-fold cross validation, which divides all data to 10 different training and testing portions. The accuracy would be averaged across 10 different

Input:	testing histogram $testData$ training data $trainData$ k constant in KNN k threshold for driver d $threshold_d$
Output:	identification result (driver label or impostor) $result$
1:	$kList_d$ = from $testData$ find k nearest neighbor in $trainData$
2:	$nearest$ = minimum distance in $kList_d$
3:	$predictedDriver$ = driver label that is the majority in $kList_d$
4:	if $nearest$ is more than $threshold_{predictedDriver}$ do:
5:	$result$ = "impostor"
6:	else do:
7:	$result$ = $predictedDriver$

ALGORITHM 4: Impostor detection using the KNN.

TABLE 2: Confusion matrix (2x2).

Total population		True condition	
		Condition positive	Condition negative
Predicted condition	Predicted condition positive	True positive	False positive
	Predicted condition negative	False negative	True negative

testing portions. Few histograms from the training portion need to be removed as it would overlap the testing portion. Data of each driver were mixed equally. The main metric is accuracy, which is calculated as shown in (1). All of the accuracies in this section will be calculated by averaging value from the 10-fold cross validation. However, to evaluate the case study of impostor detection, false positive and false negative results are also important. To understand these values, the confusion matrix is illustrated in Table 2. According to the confusion matrix, the three following important metrics are introduced. First, the precision or positive predictive value is calculated as shown in (2). The precision can show how many predicted impostors are correct or it is actually another driver. Second, recall, true positive rate, or sensitivity is calculated as shown in (3). The recall can show how many times the impostor is predicted correctly or it is predicted as other drivers. Precision and recall measures different perspectives. Third, the F1 score or F measure is calculated from the harmonic mean of precision and recall as shown in (4). The F1 score combines information of both the precision metric and recall metric.

$$Accuracy = \frac{N_c}{N_a}. \quad (1)$$

When N_c is the number of correctly identified histograms, N_a is number of all testing histograms.

$$precision = \frac{\Sigma True\ positive}{\Sigma Predicted\ condition\ positive} \quad (2)$$

$$Recall = \frac{\Sigma True\ positive}{\Sigma Condition\ Positive} \quad (3)$$

$$F1\ score = \frac{2}{1/precision + 1/recall}. \quad (4)$$

5.1. Performance Evaluation of the Inactive-Period Filtering Module. This section aims to show the achieved accuracy after applying the inactive-period filtering module. This module is one of the most important modules, which helps the accuracy significantly increase from our previous work [19]. In our previous work, the accuracy was evaluated in three combinations of the axes shown in Figure 11 colored in blue with a diagonal strip. In this paper, as we proposed the system that applies the inactive-period filtering module, it helps increase the accuracy up to 18% as shown in Figure 11 colored in orange with a white dotted line. From the result of our previous work, using more axes obviously increases the accuracy. However, with the new filtering module, using only the longitudinal axis can achieve more than 90%, which is higher than those of the previous work's results and high enough to be used alone without other axes. The effect of involved axes will be explained in the next subsection.

5.2. Performance Evaluation on Axis Involved. As the acceleration has three axes, this section aims to show the result when using data obtained from a different axis and variation of axes involved. From our previous work, the result was shown in three combinations of the axes which are the x-axis only, the x- and y-axes, and all three axes. The result was already shown again in the previous subsection. In this section, we evaluated our new system using all possible combinations of all axes. Figure 12 shows the accuracy results when combining data from different axes. As can be seen, the x-axis (the longitudinal axis) provides the highest accuracy followed by the y-axis and z-axis, respectively. This is because each axis represents different moving directions of the vehicle. The x-axis is the longitudinal axis, representing the behavior in frontal movement. The y-axis is the lateral axis, representing the behavior when changing lanes or turning. The z-axis is the vertical axis, representing the behavior when driving

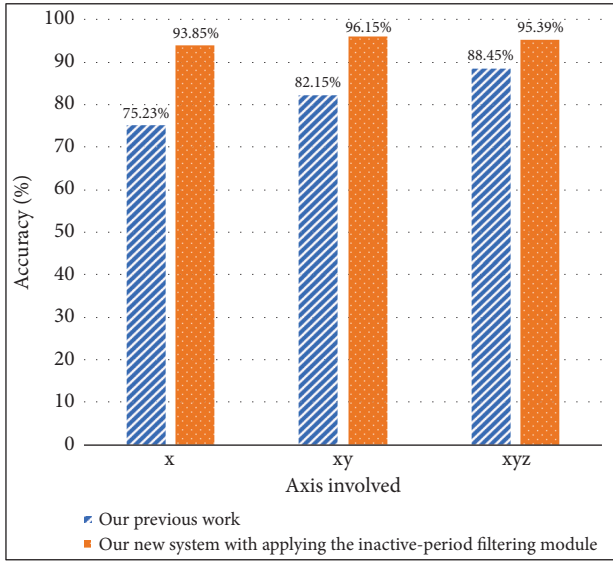


FIGURE 11: Plot of the accuracy with and without applying the inactive-period filtering module.

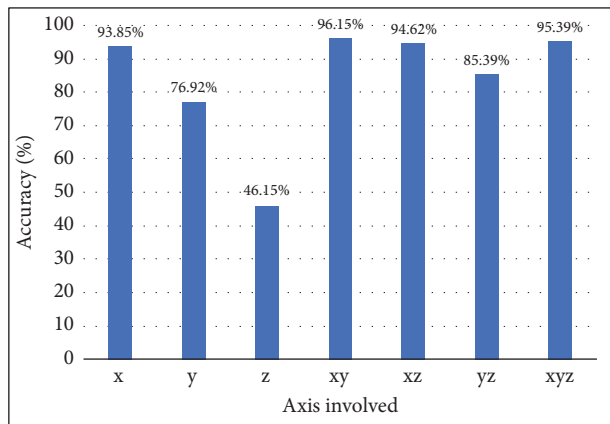


FIGURE 12: Plot of the accuracy across the axis involved.

across a slope way, such as small bridge, lamp, or speed bumper. Moreover, as can be seen in Figure 12, the results when utilizing data from the xy-axes, xz-axes, and yz-axes are 96.15%, 94.62%, and 85.39%, respectively. With the inactive-period filtering module, utilizing only data obtained from the x-axis in our system can provide accuracy of 93.85%. When combining the data obtained from the x-axis with that from the y-axis and/or z-axis, the accuracy is slightly improved. This reveals the fact that the data from the x-axis is the most important to achieve high accuracy. The idea of axis combination is not limited to the acceleration data. For other driver identification systems, this idea can be applied to other vehicular information that has more than one axis such as velocity or jerk.

5.3. Performance Evaluation of Feature Extraction Module. To represent the behavior of the driver, the acceleration data should be collected for a proper duration. The longer duration

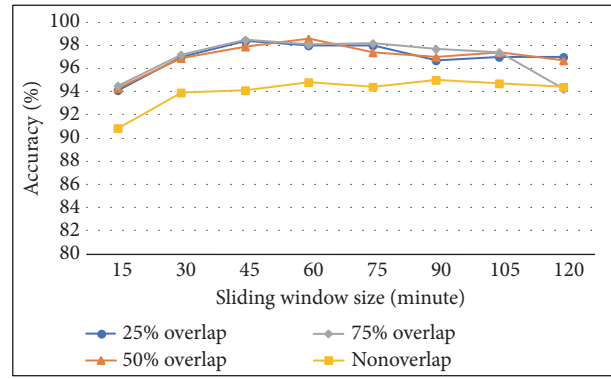


FIGURE 13: Plot of the accuracy across sliding window size and overlapping percentage.

normally provides more accuracy as the system has more behavior information. However, it leads to longer time to identify the driver, which might not be practical for some applications. Therefore, the duration that is the size of the sliding window should be tested for the optimal value. In our experiment, we varied the sliding window sizes from 15 to 120 minutes. At each sliding window size, we also tested with different overlapping percentages. Figure 13 shows accuracy results from the experiment (note that the y-axis of the figure starts at 80 percent to clarify the difference). As can be seen, the sliding window sizes that provide the highest accuracy are 45 and 60 minutes. With the sliding window size of 45 minutes, the system would identify the driver every 45 minutes, which is not very practical. However, with the overlapping technique, the system can identify the driver more frequently. To choose the overlapping percentage value, it can be seen from Figure 13 that different overlapping percentages do not provide significant difference in accuracy. However, for every tested sliding window size, our system with the overlapping technique apparently yields higher accuracy than that without the overlapping technique. This is because there is probably important driving behavior information at the edge of the nonoverlapping histogram. Using the overlapping technique could help gather such information, leading to higher accuracy. Thus, to achieve the highest accuracy, our system will be configured to use 45 minutes of the sliding window size and 75 percent of the histogram overlapping. With this configuration, the system can identify the driver every 11 minutes. Nevertheless, for some applications that need to identify the driver faster than every 11 minutes, a shorter sliding window size can be chosen. For example, according to Figure 12, if the sliding window size and the overlapping percentage are configured to 15 minutes and 75 percent, respectively, our system can identify the driver every 3-4 minutes with accuracy of 94%. The sliding window and overlapping technique can be applied to most of the driver identification systems as they normally utilize time-series data from the sensors in the vehicle.

5.4. Performance Study on the Effect of the Number of Drivers. Driver identification is a multiclass classification problem.

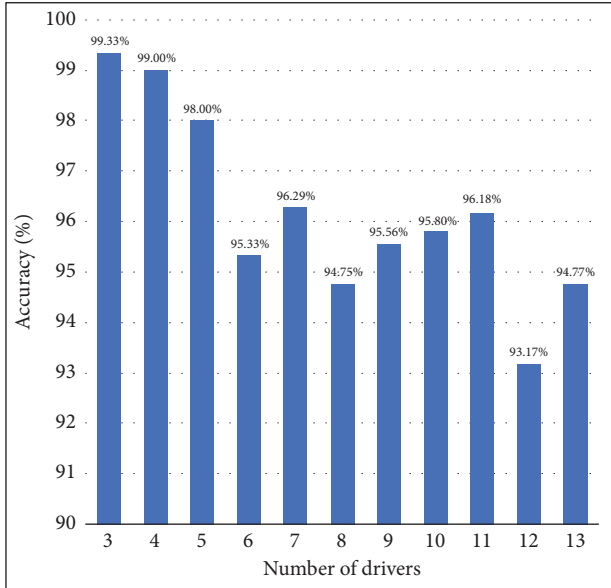


FIGURE 14: Plot of the accuracy across set of drivers.

The number of drivers in this case is the number of classes, which clearly has impact on the classification result. Thus, in our experiment, we vary the number of drivers to be identified from 3 to 13 drivers. According to Section 5.2, we will use only the data obtained from the x-axis, which is the most important axis. Additionally, the sliding window size and overlapping percentage are set to 45 minutes and 75 percent because these parameters provide the highest accuracy as revealed in Section 5.3. The accuracy result is shown in Figure 14 of which the x-axis is the total number of drivers to be identified. Note that the y-axis of the figure starts at 90. As can be seen, the lower the number of drivers to be identified, the higher the accuracy that can be achieved. Specifically, our system can achieve an accuracy of 99.33% in case of identifying 3 drivers and 94.77% in case of identifying 13 drivers.

5.5. Performance Comparison with Other Classifier Algorithms. The classifier algorithm plays an important role in the driver identification system. From our previous work, we originally used a neural network (NN) in our driver identification module. However, in this paper, we aim to show the possibility of using other classifier algorithms together with our proposed histogram feature. The decision tree, Support Vector Machine (SVM), Random Forest (RF), and k-nearest neighbors (KNN) are selected to be evaluated in this experiment. All classifier algorithms are implemented by using the library from CRAN. The decision tree, RF, and SVM are implemented by using *dtree*, *randomForest*, and *e1071*, respectively. They are configured with the default configuration. The NN is configured as explained in Section 3.3, the driver identification module. KNN is configured as explained in Section 4, the case study of impostor detection. Figure 15 illustrates the accuracy achieved by each classifier algorithm. Note that the y-axis starts at 60% to clearly show

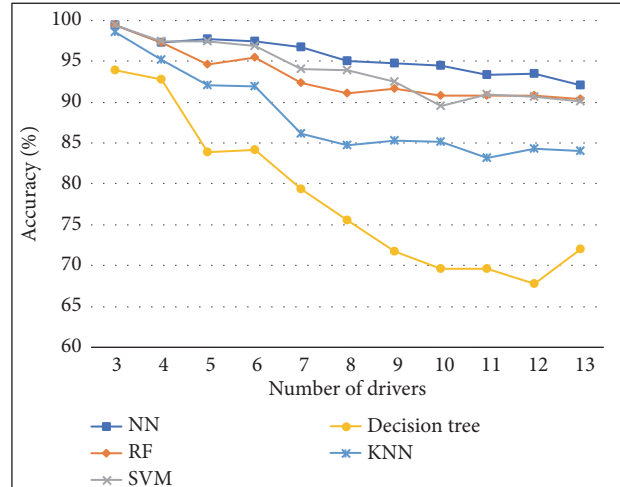


FIGURE 15: Plot of the accuracy across algorithms and number of drivers.

the difference. The x-axis of the figure denotes the number of drivers to be identified from 3 to 13 drivers. This can show the accuracy of each algorithm affected by a different number of drivers. As can be seen, the NN achieved higher accuracy than other classifier algorithms with every number of drivers. However, the complexity of the NN is the highest among these classifier algorithms. As a result, using the NN requires training time longer than other algorithms. In the aspect of prediction time, however, using the NN or other algorithms approximately requires the same period of time. Thus, for each driver identification application, the classifier algorithms should be evaluated and selected based on the requirement of complexity and accuracy.

5.6. Performance Study on the Effect of Driving Period. This section aims to show the identification accuracy affected by different driving periods. The amount of data, or the driving period, is crucial as it represents the driving hours required for the system to be operated for the first time. The training period should not be too short or too long. On one hand, a too short training period may not represent the driving behavior in the long run. On the other hand, a too long training period may not be practical because it takes time that is too long to collect data. Therefore, we evaluate our system performance with varied driving periods to ensure the integrity of our system. Figure 16 shows the accuracy when the active driving period is varied from 3 to 60 hours. We use 45 minutes of sliding window size with 75 percent overlapping as explained in Section 5.3. As can be seen, the highest accuracy is 94%, achieved when the driving period is set to 10 or 15 driving hours. The accuracy drops when having a driving period shorter than 10 hours. The accuracy also drops when having too much driving data, but it is still above 80% for all cases. In the case of using 10 hours of driving period, if the driver drives 2 hours per day, this will require 5 training days to collect the 10 hours of driving data before the system can identify the driver with 94% accuracy.

TABLE 3: Summation of the confusion matrix from 10-fold cross validation in the impostor detection experiment when utilizing the neural network.

		True driver			
		Driver A	Driver B	Driver C	Impostor
Predicted driver	Driver A	131	1	0	10
	Driver B	0	132	0	11
	Driver C	0	3	136	35
	Impostor	59	54	54	140

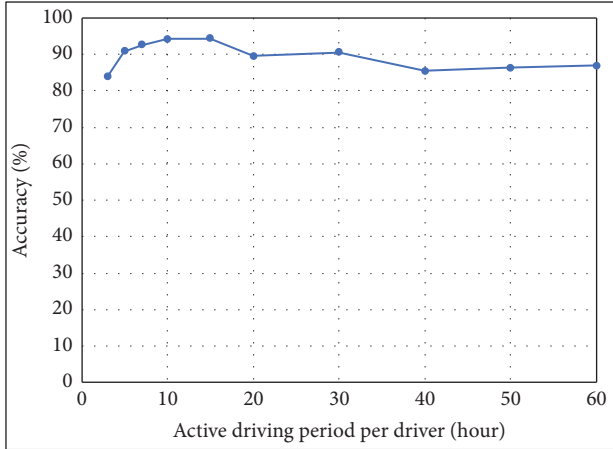


FIGURE 16: Plot of the accuracy across active driving period per driver.

5.7. Evaluation in the Case Study on Our Proposed Impostor Detection. According to Section 4, we have already explained our modified system for detecting an impostor. To evaluate our impostor detection system, we set a situation where 3 drivers share a vehicle. Therefore, from our 13 drivers, we randomly select 3 drivers to be the genuine drivers. Then, impostors are selected and mixed randomly from the remaining 10 drivers. Then, in the 10-fold cross validation process, we keep 1/10 of the data to be the testing data. We also keep another 1/10 of the data to be the validation data. This validation data is used to calculate the threshold for each driver, which has already been explained in Section 4. Thus, in each fold, the portion of training data would be 8/10. In the system training process, we only input the genuine driver's data, because, in the real situation, we do not have the impostor's data for the first place. Then, we mix the impostor's data, with the same amount of one genuine driver, in the testing data. We evaluate our impostor detection system by considering the precision, recall, and F1 score calculated as shown in (2), (3), and (4), respectively. The value of these metrics only ranges from 0 to 1, representing the performance of the system from the lowest performance to the highest performance. Drivers including impostors have their own precision, recall, and F1 score. We also define an additional metric called overall F1 score. As data across all drivers and impostor is balanced, the overall F1 score is calculated using the macro-average for simplicity as shown in (5). Figure 17 shows the result of each metric when utilizing our proposed

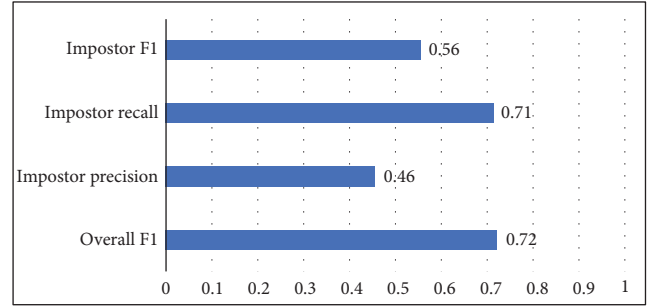


FIGURE 17: Plot of overall F1, impostor precision, impostor recall, and impostor F1 when utilizing the neural network.

neural network. As can be seen, the overall F1 score is 0.72, which is acceptable. However, if considering only the impostor, the F1 score is only 0.56, which is caused by the low impostor's precision score. The precision is only 0.46, because impostor's histograms were detected as the genuine driver many times. For more clarification of the result, the confusion matrix is also illustrated in Table 3. As can be seen in the first row, 142 histograms are identified as driver A. 131 of them are correctly identified. However, 1 of them is actually driver B and 10 of them are actually the impostor. The second row and the third row can be explained in the same direction. For the fourth row, 304 histograms are identified as the impostor. 140 of them are correctly identified. However, 59 of them are driver A, 54 of them are driver B, and 54 of them are driver C.

$$Overall\ F1 = \frac{F1_{d1} + F1_{d2} + F1_{d3} + F1_i}{4}. \quad (5)$$

When $F1_{d1}$ is the F1 score of the first driver, $F1_{d2}$ is the F1 score of the second driver, $F1_{d3}$ is the F1 score of the third driver, and $F1_i$ is the F1 score of the impostor.

As can be seen from the result, the performance of the system is not promising. This is because the output nodes of the NN are not suitable for calculating the threshold. Thus, we newly propose an algorithm using the KNN as explained in Section 4. The KNN algorithm involves the utilization of distance function. Thus, the KNN is more appropriate for threshold calculation. Figure 18 shows the precision, recall, and F1 score of the impostor detection system when utilizing the KNN. As can be seen, our system utilizing the KNN outperforms that utilizing the NN in all metrics. Specifically, the overall F1 score increases from 0.72 to 0.87. Considering only the impostor, our system with the KNN can achieve

TABLE 4: Summation of the confusion matrix from 10-fold cross validation in the impostor detection experiment when utilizing the K-nearest neighbors.

		True driver			
		Driver A	Driver B	Driver C	Impostor
Predicted driver	Driver A	169	1	0	20
	Driver B	0	175	0	0
	Driver C	0	0	170	23
	Impostor	21	14	20	153

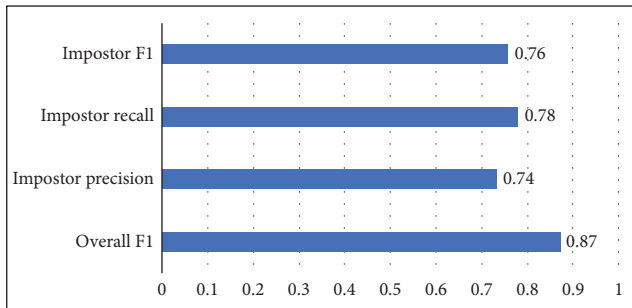


FIGURE 18: Plot of overall F1, impostor precision, impostor recall, and impostor F1 when utilizing the k-nearest neighbors.

0.76 of the F1 score. This is because the precision and recall increase to 0.74 and 0.78, respectively. For more clarification of the result, the confusion matrix is also illustrated in Table 4. As can be seen in the first row, 190 histograms are predicted as driver A. 169 of them are correctly identified. However, 1 of them is actually driver B and 20 of them are actually the impostor. The second row and the third row can be explained in the same direction. For the fourth row, 208 histograms are predicted as the impostor, 153 of which are correctly identified. However, 21 of them are predicted as driver A, 14 of them are predicted as driver B, and 20 of them are predicted as the impostor.

6. Conclusions

In this paper, we have proposed a public transport driver identification system architecture which utilized only a single acceleration sensor. The system architecture aims to improve the accuracy and the practical usage of the driver identification system for public transport. The system consists of three main modules. The first module, data collection, involved the installation of sensors and data sending. The second module, data preprocessing, involved the data cleaning, inactive-period filtering, and acceleration histogram extraction. The last module, driver identification module, involved the machine learning, which is the neural network. Our system achieved an accuracy of up to 99%. In order to show that our system is practical, the performance evaluation considered several important factors including the inactive-period filtering module, acceleration axis involved, sliding window size and the overlapping technique, classifier algorithms, number of drivers, and driving period. Moreover, by modifying the

identification module, we expanded the capability of our model to identify impostors with an overall F1 score of 0.87. Our system architecture design and consideration of aspects on the performance evaluation could be used as a guideline for real public transport driver identification system and further driver identification researches.

Data Availability

The data used to support the findings of this study are owned by a company, but it can be provided from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by Chula Computer Engineering Graduate Scholarship for CP Alumni, Chulalongkorn University. The authors would like to thank all members of ISEL, family, and friends for their help.

References

- [1] Statista, *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions)*, 2016, <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>.
- [2] Gartner, *Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016*, 2017, <https://www.gartner.com/newsroom/id/3598917>.
- [3] Ericsson, "Internet of Things forecast," 2018, <https://www.ericsson.com/en/mobility-report/internet-of-things-forecast>.
- [4] Gartner, *Gartner Says By 2020, a Quarter Billion Connected Vehicles Will Enable New In-Vehicle Services and Automated Driving Capabilities*, 2015, <https://www.gartner.com/newsroom/id/2970017>.
- [5] C. Marina Martinez, M. Heucke, F.-Y. Wang, B. Gao, and D. Cao, "Driving style recognition for intelligent vehicle control and advanced driver assistance: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 666–676, 2018.
- [6] INTERPOL, *Vehicle crime*, 2017, <https://www.interpol.int/Crime-areas/Vehicle-crime/Database-statistics>.

- [7] C. Miyajima, Y. Nishiwaki, K. Ozawa et al., "Driver modeling based on driving behavior and its evaluation in driver identification," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 427–437, 2007.
- [8] A. Wahab, C. Quek, C. K. Tan, and K. Takeda, "Driving profile modeling and recognition based on soft computing approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 20, no. 4, pp. 563–582, 2009.
- [9] I. Del Campo, R. Finker, M. V. Martinez, J. Echanobe, and F. Doctor, "A real-time driver identification system based on artificial neural networks and cepstral analysis," in *Proceedings of the 2014 International Joint Conference on Neural Networks, IJCNN 2014*, pp. 1848–1855, China, July 2014.
- [10] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, 2016.
- [11] D. Hallac, A. Sharang, R. Stahlmann et al., "Driver identification using automobile sensor data from a single turn," in *Proceedings of the 19th IEEE International Conference on Intelligent Transportation Systems, ITSC 2016*, pp. 953–958, Brazil, November 2016.
- [12] M. V. Martínez, J. Echanobe, and I. Del Campo, "Driver identification and impostor detection based on driving behavior signals," in *Proceedings of the 19th IEEE International Conference on Intelligent Transportation Systems, ITSC 2016*, pp. 372–378, Brazil, November 2016.
- [13] N. C. Fung, B. Wallace, A. D. C. Chan et al., "Driver identification using vehicle acceleration and deceleration events from naturalistic driving of older drivers," in *Proceedings of the 12th IEEE International Symposium on Medical Measurements and Applications, MeMeA 2017*, pp. 33–38, USA, May 2017.
- [14] S. Ezzini, I. Berrada, and M. Ghogho, "Who is behind the wheel? Driver identification and fingerprinting," *Journal of Big Data*, vol. 5, no. 1, p. 9, 2018.
- [15] F. Martinelli, F. Mercaldo, A. Orlando, V. Nardone, A. Santone, and A. K. Sangaiah, "Human behavior characterization for driving style recognition in vehicle system," *Computers and Electrical Engineering*, 2018.
- [16] L. Moreira-Matias and H. Farah, "On developing a driver identification methodology using in-vehicle data recorders," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2387–2396, 2017.
- [17] S. Jafarnejad, G. Castignani, and T. Engel, "Towards a real-time driver identification mechanism based on driving sensing data," in *Proceedings of the 20th IEEE International Conference on Intelligent Transportation Systems, ITSC 2017*, pp. 1–7, Japan, October 2017.
- [18] P. Phumphan, P. Wuttidittachotti, and C. Saiprasert, "Driver identification using variance of the acceleration data," in *Proceedings of the 19th International Computer Science and Engineering Conference, ICSEC 2015*, pp. 1–6, IEEE, Thailand, 2015.
- [19] N. Virojboonkiate, P. Vateekul, and K. Rojviboonchai, "Driver identification using histogram and neural network from acceleration data," in *Proceedings of the 17th IEEE International Conference on Communication Technology, ICCT 2017*, pp. 1560–1564, China, October 2017.
- [20] A. Chowdhury, T. Chakravarty, A. Ghose, T. Banerjee, and P. Balamuralidhar, "Investigations on Driver Unique Identification from Smartphone's GPS Data Alone," *Journal of Advanced Transportation*, vol. 2018, Article ID 9702730, 11 pages, 2018.
- [21] InvenSense, *MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Devices*, 2013, <https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/>.

Research Article

Rapid Driving Style Recognition in Car-Following Using Machine Learning and Vehicle Trajectory Data

Qingwen Xue , Ke Wang , Jian John Lu , and Yujie Liu

College of Transportation Engineering, Tongji University, 4800 Cao'an Road, Shanghai 201804, China

Correspondence should be addressed to Ke Wang; kew@tongji.edu.cn

Received 4 October 2018; Revised 27 December 2018; Accepted 6 January 2019; Published 23 January 2019

Guest Editor: Mohammad H. Y. Moghaddam

Copyright © 2019 Qingwen Xue et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Rear-end collision crash is one of the most common accidents on the road. Accurate driving style recognition considering rear-end collision risk is crucial to design useful driver assistance systems and vehicle control systems. The purpose of this study is to develop a driving style recognition method based on vehicle trajectory data extracted from the surveillance video. First, three rear-end collision surrogates, Inversed Time to Collision (ITTC), Time-Headway (THW), and Modified Margin to Collision (MMTC), are selected to evaluate the collision risk level of vehicle trajectory for each driver. The driving style of each driver in training data is labelled based on their collision risk level using K-mean algorithm. Then, the driving style recognition model's inputs are extracted from vehicle trajectory features, including acceleration, relative speed, and relative distance, using Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), and statistical method to facilitate the driving style recognition. Finally, Supporting Vector Machine (SVM) is applied to recognize driving style based on the labelled data. The performance of Random Forest (RF), K-Nearest Neighbor (KNN), and Multi-Layer Perceptron (MLP) is also compared with SVM. The results show that SVM overperforms others with 91.7% accuracy with DWT feature extraction method.

1. Introduction

Driving style refers to the ways that drivers choose to habitually drive and the driver states that represent the common parts of varied driving behavior [1]. Driving style of drivers plays an important role in driving safety as well as vehicle energy consumption. Different driving styles may lead to different possibilities for traffic incidents. Recognition of a driver's driving style based on rear-end collision risk is of great significance to improve the safety of driving. With the development of connected autonomous vehicles and Advanced Driver Assistance System (ADAS), there is an urgent demand for enhancing recognition of driving style. It is not only important to guarantee the safety and adequate performance of drivers, but also essential to meet drivers' need, adjust to the drivers' preference, and ultimately improve the safety of the driving environment. Driving style recognition also has potential value to help traffic agencies design control strategies effectively [2, 3].

The availability of high-definition surveillance camera makes it possible to collect numerous vehicle motions from

real world traffic flow. The advanced video extraction software can extract vehicle trajectory data accurately and efficiently from the surveillance video. The technologies provide a good opportunity to recognize driving style using the video-extracted vehicle trajectory data. Moreover, the machine learning technique is playing a crucial role in driving behavior recognition. A growing amount of studies on machine learning algorithms have been conducted in recent years [4–7]. This paper builds a driving style recognition model based on vehicle trajectory data. Four supervised machine learning algorithms, including Supporting Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbor (KNN), and Multi-Layer Perceptron (MLP), are used in model training. A new method based on rear-end collision risk is proposed to label the driving style of each driver in the sample data. Three feature extraction methods, including Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), and statistical method, are also adopted to extract the most effective features of driving style recognition.

To the best knowledge of the authors, there are three main contributions in this paper: (1) This paper proposes

TABLE 1: Comparison of different driving behavior data collection approaches.

Collection Approaches	Advantages	Disadvantages
In-vehicle camera [29–33], sensors, hardware [10, 24]	Real-world driving data; high-accuracy data; Access to driver's personal data and vehicle control data	Expensive and time consuming; Lack of data in extreme and dangerous driving condition
Driving simulator [24–26]	Collect drivers' behavior in designed and controlled driving scenarios	Driving behavior observed in the simulator may not always correspond to real-world driving
Traffic video [35–37]	Low expense; Easy to collect enormous vehicle data in a short time; Observe vehicle interaction in real traffic flow	Video extraction is challenging; No access to driver's personal data and vehicle control data
Smartphone-based method [52–55]	Real-world driving data; Low expense in the smartphone	Data accuracy is critical

a new method based on rear-end collision risk to evaluate driving style. The trajectory of each driver is divided into segments with different risk level by the threshold of rear-end collision surrogates. (2) The DFT, DWT, and statistical feature extraction methods are all applied on vehicle trajectory data, and their performance is compared. (3) This paper builds a driving style recognition model based on vehicle trajectory data with 92.7% accuracy rate. The recognition results of SVM and other popular classification algorithms including RF, MLP, and KNN are compared.

This paper is organized as follows. Section 2 presents the related work on driving behavior data analysis and machine learning algorithms. Section 3 introduces the data analyzed in this paper. Section 4 details the driving style recognition method implemented in this paper. Section 5 shows the results and discussion. Section 6 concludes this paper and raises the possible future work.

2. Literature Review

In recent years, the machine learning algorithms applied to the driving behavior recognition have been studied in many previous works. Different types of neural network (NN) algorithms have been used. Molchanov et al. [8] proposed a convolutional deep neural network (CDNN) to recognize the risky driving. Other types such an artificial neural network (ANN) [9] and pulse coupled neural network (PCNN) [10] were adopted to classify the driving behaviors. In the study by Srinivasan [11], the effectiveness of three types of NN methods was compared. The results show that the Multi-Layer Perceptron (MLP) model can achieve excellent classification results. However, the learning rate of NN is difficult to be determined, resulting in higher possibility to be trapped in local minima. A larger size of the network could lead to a long training time [12]. The tree-like structures including decision tree algorithm [13] and Random Forest algorithm [14] are also adopted to detect the driving behaviors according to the extracted features. Some researchers proposed Hidden Markov Model (HMM) to effectively detect dangerous driving behaviors. Berndt et al. [15] established the HMM to identify lane change, steering, and follow-up intention. The recognition accuracy of left-change and right-change is, respectively, 76% and 74%. Meng et al. [16] trained the HMM by collecting driver's operation

data on acceleration pedal, brake pedal, and steering wheel to recognize the driver's profiles online. Some researchers also combined the HMM with dynamic Bayesian networks or ANN to predict the driving behavior by learning the driving data [17, 18]. While HMM requires long training time, especially for a high number of states, the recognition time also increases with the number of states [19]. Therefore, a more suitable and effective method should be found to identify the driving style. SVM has been widely applied to various kinds of pattern recognition problems, including voice identification, text categorization, and face detection [6, 20, 21]. In addition, SVM performs well with a limited number of training samples, and SVM has fewer parameters to be determined [22, 23]. Therefore, many studies employed SVM to build driving style recognition models [24–28].

Along with machine learning algorithms, driving behavior data collection is crucial to the success of driving style recognition. Table 1 summarizes the advantage and the disadvantages of different driving data collection approaches. Researchers used instrumented vehicles to conduct naturalist driving experiments to identify behaviors [29–31]. Some instrumented vehicles were equipped with in-vehicle mounted cameras to capture video images of drivers [32, 33], while others got help from specialized hardware and sensors to acquire throttle opening, pedal brake, wheel steering, vehicle speed, acceleration rate, and yaw rate [10, 24]. Although the driver controlling data and vehicle kinematic data can be collected on the instrumented vehicles, the requirement of expensive devices and sensors is a major obstacle to large scale naturalist driving experiments. In addition, extreme driving conditions, like extreme weather and driving under the influence, could be unobservable in naturalist driving studies. Some research adopted driving simulators to collect driving behavior data [24–26] in the designed and controlled driving environment. However, the results heavily relied on the fidelity and validity of the driving simulator used in research, because the driving behavior observed in the simulator may not always correspond to real-world driving.

Besides Naturalist Driving Studies (NDS) and driving simulator, another important data source is traffic video, because surveillance cameras deployed on the roadside can provide a large amount of traffic environment data and vehicle trajectory data [34]. Traffic video contains all vehicle

trajectory data on the road and can offer a full view of vehicle's interactions with other during car-following and lane-change, etc. However, extracting vehicle trajectory from video could be challenging, which depends on video quality and algorithms used [35–37].

Except for unsupervised machine learning algorithms, for example, clustering, other machine learning algorithms require labelled or partially labelled driving behavior data. In the field of driving style recognition, the method of driving style labelling for each driver in the sample is of great importance to the reliability of the recognition model. There are several methods to label driving style. One is the behavior-based or accident-based method. The driver's driving style depends on risky behavior or accident happened during observation. Chen et al. [20] defined the dangerous driving behaviors according to criteria as frequent lane changes, abrupt double lane change, and illegal lane occupation. The accidents data are also adopted to determine the risk level of driving behavior [38]. However, risky behavior or accident is hardly observable in daily traffic. Therefore, driver self-reported questionnaire [39] and expert scoring [13] are also adopted to evaluate driving style. However, these two methods rely on subjective judgments of drivers or experts and can be very time-consuming when the number of drivers in the sample is hundreds or even thousands. Some research used the facial movement or driving duration to label driver's drowsiness or fatigue driving [9, 10]. The unsupervised clustering methods including the K-means [40] and fuzzy clustering [41] are also used to label drivers in each clustering group.

This paper proposes a new driving data labelling method based on collision surrogates. There are many effective surrogates to evaluate the collision risk [42, 43]. Mahmud et al. [44] compared the advantages and disadvantages between temporal proximity indicators, i.e., Time to Collision (TTC), Time to Accident (TA), Time-Headway (THW), and distance based proximal indicators, i.e., Margin to Collision (MTC), Proportion of Stopping Distance (PSD). Many automobile collision avoidance systems or driver assistance systems used TTC as an important warning criterion for its theoretical and reliable reasons [45–47]. Since TTC can not handle zero relative speed in car-following, the Inversed TTC (ITTC) was adopted to measure the collision risk [41]. THW is another surrogate used to estimate the criticality of a follow-up situation, which is applicable in all traffic environments [44]. MTC provides the possibility of conflict when the preceding and following vehicle at the same time decelerate abruptly [48]. Modified MTC (MMTC) considers the reaction time for drivers when preceding vehicle abruptly decelerates. These three surrogates can be adopted to label the driving style corresponding to different rear-end collision effectively.

In this paper, the vehicle trajectory data extracted from traffic video is analyzed to study the driving style. Three surrogates, i.e., ITTC, THW, and MMTC, are used to effectively measure the rear-end collision risk and label the driving style. This labeling method is more efficient and objective compared with questionnaires [10] and expert scoring [20]. Then the SVM is applied to build a driving style recognition model. The vehicle trajectory features are extracted using

the Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), and statistical methods. The performance of SVM is also compared with RF, KNN, and MLP. This paper provides an efficient method to identify driving style based on the trajectory data.

3. Data

A high-fidelity vehicle trajectory dataset, Next Generation Simulation (NGSIM), was collected by U.S. Federal Highway Administration (FHWA) in 2005. This dataset is still widely used in transportation research, especially in traffic flow analysis and modelling, traffic-related estimation and prediction, and vehicular ad hoc network-related studies [49]. It has rarely been applied to driving style recognition. Since this dataset was collected more than a decade ago, the accuracy of NGSIM dataset was questioned in recent years [50]. The measurement errors in NGSIM dataset were found to be far beyond negligible, partially due to low-resolution cameras and mis-tracking of vehicles from video images. Montanino et al. [51] removed outliers and noise and reconstructed the I-80 dataset 1 (from 4:00 p.m. to 4:15 p.m.), which showed significant improvement over the original NGSIM dataset.

In this paper, the I-80 trajectory dataset is adopted to study driving style. The trajectory data was collected on a segment of I-80 freeway in Emeryville, California. The segment contains 6 lanes, where lane 1 is a high occupancy vehicle (HOV) lane. The frequency of data collection is 10 Hz, and each leader-follower pair of dataset contains detailed information including the vehicle ID, position, length, and width of the vehicle, velocity, acceleration, lane ID, and following and preceding vehicles. About 206,000 records of vehicle trajectory for 370 Leader-follower Vehicle Pairs (LVP) on HOV lane are chosen to study the driving style in this paper since there are fewer interrupting vehicles from other lanes.

4. Methodology

The flow of driving style recognition in this paper is depicted in Figure 1. Three collision risk surrogates are used to determine the risk level of every moment in the car-following process for each LVP. K-means algorithm is applied to group the drivers as normal or aggressive driving style based on their trajectory risk levels. Given the labeled driving data, driving style recognition model is built using machine learning algorithms. The input features of machine learning algorithms are extracted by DFT, DWT, and statistical methods from trajectory features, without using surrogates and risk levels. The recognition results recognized by SVM are compared with other machine learning algorithms.

4.1. Collision Risk Surrogates. For each driver, it is essential to find the most effective surrogates to describe the collision risk when driving on the road. Vehicle trajectory data such as velocity and acceleration of the vehicle usually are not good enough to estimate the rear-end collision risk. Three collision surrogates are considered to measure the collision risk, including Time to Collision (TTC), Time-Headway (THW),

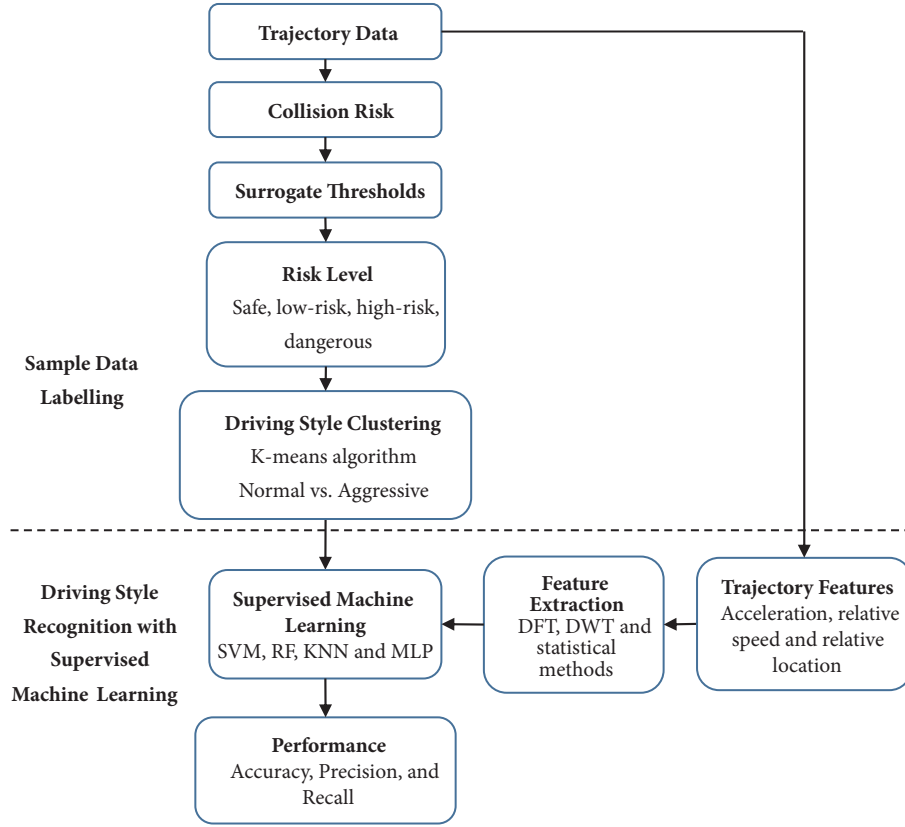


FIGURE 1: Research methodology framework.

and Margin to Collision (MTC). These three collision risk surrogates are defined and modified as follows.

Inversed Time to Collision (ITTC). TTC is the predicted time to collision between the preceding vehicle (PV) and following vehicle (FV) when two vehicles remain the current relative velocity.

$$TTC = -\frac{x_r}{v_r} = -\frac{(x_f - x_p)}{(v_f - v_p)} \quad (1)$$

where x_r and v_r denote relative distance and velocity between two following vehicles, respectively. x_f and x_p denote the front position of FV and rear position of PV, respectively. v_f and v_p , respectively, denote the velocity of FV and PV, respectively. However, TTC can be very large with lower relative velocity for two following vehicles, which happened a lot in the real driving environment. To reduce the scope of TTC, the ITTC is adopted to measure the collision risk in the paper. The risk of rear-end collision is higher with larger ITTC value.

$$ITTC = \frac{1}{TTC} \quad (2)$$

Time-Headway (THW). THW indicates the time for FV to reach the present position of PV with the current velocity. The potential collision risk of drivers is determined by THW in the steady vehicle following situation.

$$THW = -\frac{x_r}{v_f} = -\frac{(x_f - x_p)}{v_f} \quad (3)$$

The potential collision risk can be evaluated by THW when FV approaches PV with constant v_f . Lower THW indicates a higher potential collision risk.

The Modified Margin to Collision (MMTC). MTC indicates the final relative position of PV and FV if two vehicles decelerate abruptly.

$$MTC = \frac{(-x_r + d_p)}{d_f} = \frac{(-x_r - v_p^2/2a_p)}{(-v_f^2/2a_f)} \quad (4)$$

where a_f and a_p denote the deceleration for FV and PV, respectively. Usually, both are defined as $0.7G$. A modified MTC (MMTC) is used in the paper to include the reaction time of the following vehicle when the PV abruptly decelerates. The equation is modified as follows.

$$\begin{aligned} MMTC &= \frac{(-x_r + d_p - d_f)}{v_f} \\ &= \frac{(-x_r - v_p^2/2a_p + v_f^2/2a_f)}{v_f} \end{aligned} \quad (5)$$

MMTC evaluates the minimum reaction time needed for FV to avoid a collision when PV abruptly decelerates at $0.7G$.

The collision risk is higher with lower MMTC value since there is little time for drivers to react. MMTC can evaluate potential collision risk with abrupt deceleration of PV.

4.2. Driving Style Clustering. The threshold values of surrogates are adopted to divide the trajectory of each driver into several collision risk levels. Then the K-means method is used to group the drivers into normal or aggressive driving style based on their components of collision risk levels. The purpose of the method is to provide an objective and stable label of driving style for each driver in the sample data and then make it ready to use in supervised machine learning.

Assume that there are n sets of driving data, and each set consists of v dimensional features denoting λ_i , which belongs to a class y_i . Therefore, the driving data of each driver can be described as $\{\lambda_i, y_i\}$. The K-means method finds the best class y_i for each driving data. The objective function of the K-means algorithm is to minimize the total in-class error squares shown as follows.

$$\min \sum_i^k \sum_{y \in y_i} \left\| (\lambda, y_i) - (\hat{\lambda}_i, y_i) \right\|^2 \quad (6)$$

where k is the number of classes. $(\hat{\lambda}_i, y_i)$ is the mean vector of all points in class y_i .

4.3. Trajectory Feature Extraction. In this paper, the vehicle acceleration a_f , relative distance x_r , and relative velocity v_r are adopted to recognize the driving style. The Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), and statistical method are used, respectively, to extract the effective features from the vehicle acceleration a_f , relative distance x_r , and relative velocity v_r .

4.3.1. Discrete Fourier Transform. DFT has been applied to convert time series of trajectory data to signal amplitude in the frequency domain [7]. The DFT of a given time series (x_1, x_2, \dots, x_N) is defined as a sequence of N complex numbers $(DFT_0, DFT_1, \dots, DFT_{N-1})$:

$$DFT_k = \sum_{n=0}^{N-1} x_n e^{-(2\pi i/N)kn} \quad (7)$$

where i is the imaginary unit. The first 10 DFT coefficients of trajectory data are used to recognize the driving style.

4.3.2. Discrete Wavelet Transform. DWT is shown to be more suitable to analyze and decompose a given signal in some studies [54]. This paper follows the DWT method described in [54] and uses the energy of approximation sub-time series and detail sub-time series, which are decomposed from vehicle acceleration a_f , relative distance x_r , and relative velocity v_r , to recognize the driving style.

4.3.3. Statistical Method. The key statistical parameters that can capture most of the distribution information of vehicle acceleration a_f , relative distance x_r , and relative velocity v_r ,

TABLE 2: Pearson correlation analysis of ITTC, THW, and MMTC.

Surrogate	THW	MMTC	ITTC
THW	1	.980**	-.240**
MMTC	.980**	1	-.385**
ITTC	-.240**	-.385**	1

** : significant correlation at 0.01 level (bilateral).

are also selected for recognition. The statistical parameters are the maximum, minimum, mean, standard deviation, and 85% percentiles, which were proved useful in previous driving behavior study [20].

4.3.4. Feature Combinations. For each driver, during car-following process, there are three time series: acceleration a_f , relative distance x_r , and relative velocity v_r . This paper tries 7 different feature combinations as the input of driving style recognition model:

Single-source features: use only one time series out of acceleration a_f , relative distance x_r , and relative velocity v_r , and extract features from this time series.

Two-source features: use two time series out of acceleration a_f , relative distance x_r , and relative velocity v_r . Therefore, there are three combinations: $a_f + x_r$, $x_r + v_r$, and $v_r + a_f$. Features are extracted from two time series separately.

Three-source features: use all three time series and extract features from three time series separately.

5. Results and Discussion

5.1. The Sample Data Labelling

5.1.1. Threshold Value of Collision Risk Surrogates. The correlation analysis among three surrogates is shown in Table 2.

Table 2 shows that the Pearson coefficient between THW and MMTC is 0.980, indicating a strong positive correlation. ITTC and THW have a weak negative correlation. Therefore, ITTC and THW are selected to measure driving behavior risk. The classification result will not be influenced by the adopting of THW instead of MMTC because of the strong correlation between the two surrogates.

To make a reasonable adjustment on collision risk along the car-following process, each surrogate has a risk threshold that can be obtained through the probability density distribution and fitting results of ITTC, THW shown in Figure 2.

Figure 2(a) shows the fitting results of ITTC, THW by adopting three distributions, i.e., normal distribution, logistic distribution, and t distribution. The t distribution achieves a better fitting performance than other two distributions on probability density distribution of ITTC and THW. Therefore, the t distribution is adopted to determine the threshold value of features. The percentile values of ITTC are shown in Figure 2(b). The 25%, 45%, 65%, 85%, and 95% percentile values of ITTC are 0.02, 0.08, 0.12, 0.19, and 0.28 s^{-1} , respectively. The 25%, 45%, 65%, and 85% percentile values of THW are 1.26, 1.71, 2.13, and 2.73 s, respectively.

ITTC. The upper threshold of ITTC is 0.28 s^{-1} , which is equivalent to 3.5 s for TTC. Previous studies show that

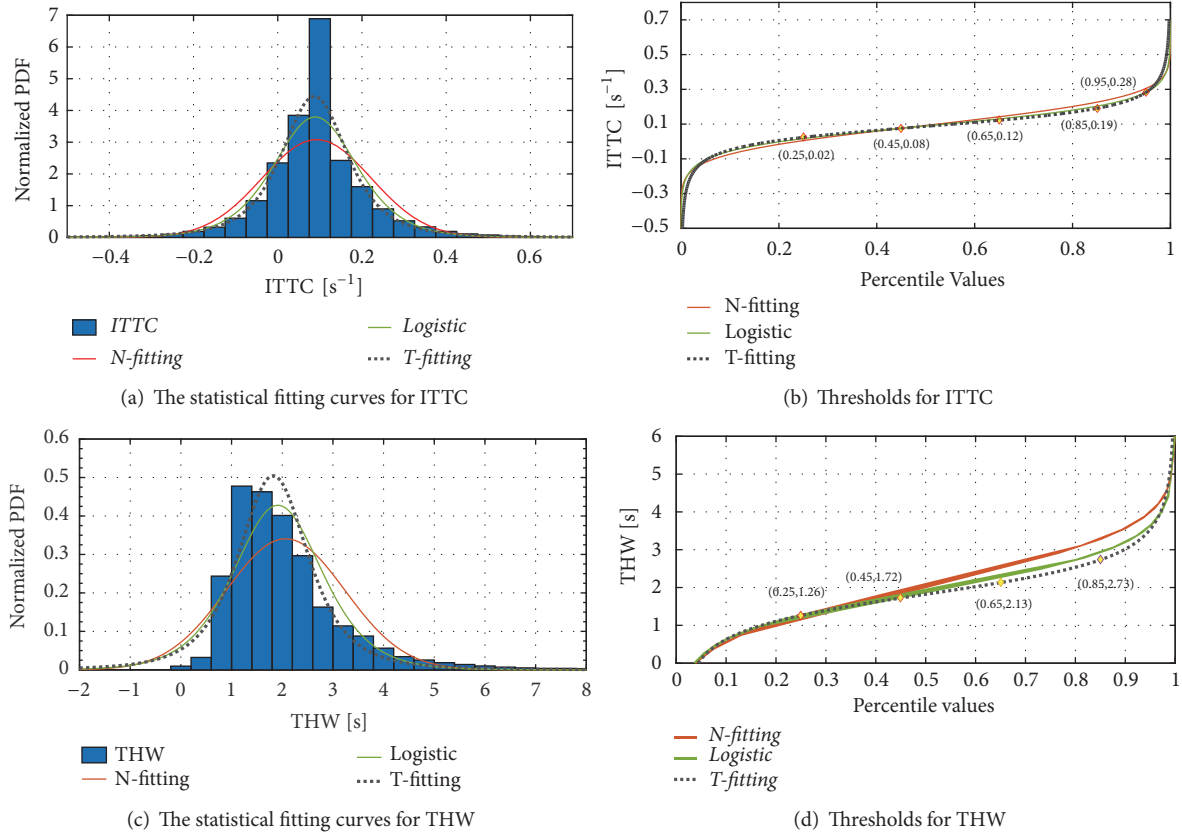


FIGURE 2: Fitting results and thresholds of surrogates.

the desirable TTC is 4 s for urban road [46] and 3.5 s for nonsupported drivers [45]. The desirable TTC for signalized intersection and two-lane rural roads is 3 s [47]. Therefore, 3.5 s is adopted in this paper as the rear-end collision risk threshold. When TTC is lower than 3.5 s, the FV is labeled as having a higher collision risk.

THW. Since a lower THW indicates a higher collision risk, the author first chose the 25% percentile, which is 1.26 s. However, many road administrations in European countries recommend a safe THW of 2 s [48]. The THW below 2 s may cause uncomfortable driving feelings and potential risk for drivers. Finally, 2 s is used as the threshold value for THW in this study.

5.1.2. Trajectory Risk Level. The threshold values of ITTC and THW, i.e., 0.28 s^{-1} and 2 s, are used to divide the driving trajectory into different risk levels. To be more specific, the different values of ITTC and THW are corresponding to different driving risk level. The driving trajectory for each driver can be divided into four risk levels: safe, low-risky, high-risky, and dangerous driving behavior, shown in Figure 3.

Safe Driving Behavior. The FV has THW above 2 s and ITTC below 0.28 s^{-1} , which indicates that the FV keeps low velocity and a large gap with the PV at car-following state.

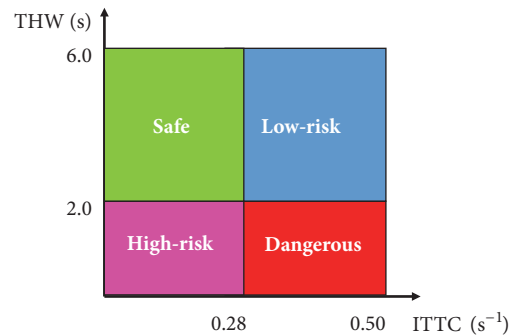


FIGURE 3: The threshold values of surrogates indicating different driving risk.

Low-Risky Driving Behavior. The FV has THW above 2 s and ITTC above 0.28 s^{-1} , which indicates that the FV keeps low velocity and a small gap with the PV at car-following state.

High-Risky Driving Behavior. The FV has THW below 2 s and ITTC below 0.28 s^{-1} , which indicates that the FV remains high velocity and a large gap with the PV at car-following state.

Dangerous Driving Behavior. The FV has THW below 2 s and ITTC above 0.28 s^{-1} , which indicates that the FV remains

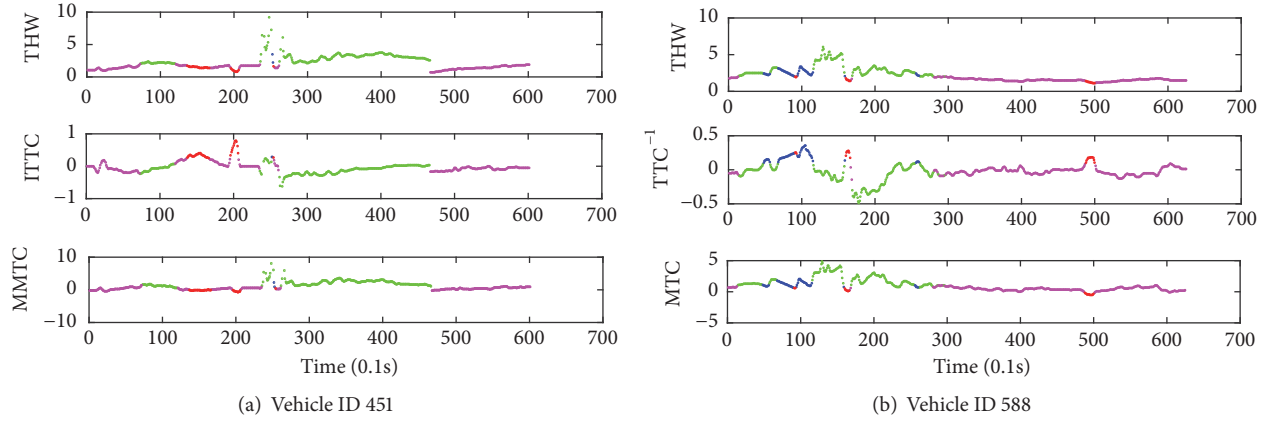


FIGURE 4: Trajectory segments for four drivers based on threshold values of ITTC and THW.

high velocity and a small gap with the PV at car-following state.

The driving trajectory of each driver can be divided into several segments, which belongs to different driving risk levels. Two drivers are selected to show the trajectory segments according to the threshold values of ITTC and THW, shown in Figure 4.

As Figure 4 shows, for most drivers, the safe and high-risk driving behaviors account for over 80% of driving trajectory. The proportion of dangerous driving and low-risk driving behaviors is limited to 10% and 5%, respectively. The driving style of each driver can be determined by the proportions of trajectory segments with different risk levels. The 370 drivers are clustered into two groups in Section 5.1.3.

5.1.3. Driving Style Clustering. Based on the proportions of trajectory segments determined by the threshold values of ITTC and THW, the drivers can be grouped into two classes using the K-means algorithm. The results show one class has 246 drivers and the other has 124 drivers. On average, drivers in the first class have 45.5% safe driving behavior, 37.5% high-risk driving behavior, and 11.4% dangerous driving behavior, and drivers in the second class have 7.4% safe driving behavior, 77.8% high-risk driving behavior, and 13.5% dangerous driving behavior. Therefore, drivers in the first class are labelled as normal drivers, while drivers in the second class are labelled as aggressive drivers. The driving style labels provided by K-means are used to train SVM in Section 5.2.

5.2. Driving Style Recognition. The SVM method is adopted to recognize the driving style for 370 drivers. In this paper, the trajectory data including the vehicle acceleration a_f , relative distance x_r , and relative velocity v_r are adopted to recognize the driving style, respectively. The DFT, DWT, and statistical methods are both applied to extract effective features from trajectory data. Every single feature can also be combined with other features as multisource features to recognize the driving style. The recognition accuracy rates are compared to find the best feature extraction method and the most important trajectory features. The z-score method is adopted to standardize features before model training.

In the study, the accuracy, precision, and recall rates are assessed to evaluate the model's ability to recognize aggressive drivers among all vehicles on the road. The performance of the recognition model is evaluated using the "leave-one-out" cross-validation method. Driving style recognition results based on different feature extraction methods and SVM are shown in Tables 3–7. Except mentioned, the SVM algorithm uses linear kernel function.

5.2.1. Discrete Fourier Transform. Shown in Table 3, the recognition accuracy rate is 83.2% based on v_r and 88.9% based on x_r . The recognition accuracy rate is 88.9% based on x_r and a_f , and 87.8% based on x_r and v_r . In general, the features x_r and v_r are better than a_f in recognizing the driving style. A possible reason is that the driving style label is determined by the rear-end collision risk, the feature a_f can not accurately describe the relative motivation between two following vehicles. The accuracy rate based on all three features can achieve 87.6%. Surprisingly, using DFT coefficients of x_r along has the highest accuracy rate.

5.2.2. Discrete Wavelet Transform. For DWT, there are two parameters to be determined, which could affect the performance of the recognition model. One is an appropriate wavelet mother function; the other is the number of decomposition levels. This paper tried 15 different wavelet mother functions (listed in Table 4) and 5 decomposition levels (listed in Table 5). The results show that Daubechies 4 mother function can generate the highest accuracy rate: 91.7%. The best decomposition level is 1, while decomposing time series further does not help to improve the accuracy rate.

With Daubechies 4 mother function and 1 decomposition level, SVM performance is assessed with different combinations of features. Shown in Table 6, the recognition accuracy rate is 83.8% based on v_r and 86.8% based on x_r . Therefore, when using x_r along in SVM, DFT extraction method works better than DWT. The recognition accuracy rate is 88.7% based on x_r and a_f and 90.2% based on x_r and v_r . The accuracy rate based on all three features can achieve 91.7%. Compared with DFT coefficients, DWT methods also get higher precision rate 92.8% and higher recall rate 81.8%.

TABLE 3: The evaluation results of driving style based on SVM using DFT.

Features	Accuracy	Precision	Recall
a_f	67.0%	55.0%	17.5%
v_r	83.2%	80.2%	67.5%
x_r	88.9%	86.3%	80.2%
$a_f + v_r$	83.2%	80.8%	66.7%
$v_r + x_r$	87.8%	86.5%	76.2%
$a_f + x_r$	88.9%	84.6%	82.5%
$a_f + v_r + x_r$	87.6%	85.1%	77.0%

TABLE 4: The accuracy of driving style recognition using DWT-SVM with different wavelet mother functions.

Wavelet Mother	Accuracy	Wavelet mother	Accuracy	Wavelet mother	Accuracy
Daubechies 1	90.3%	Symlet 1	90.3%	Coiflet 1	90.8%
Daubechies 2	90.0%	Symlet 2	90.0%	Coiflet 2	90.3%
Daubechies 3	91.1%	Symlet 3	91.1%	Coiflet 3	89.7%
Daubechies 4	91.7%	Symlet 4	89.7%	Coiflet 4	90.0%
Daubechies 5	91.1%	Symlet 5	90.5%	Coiflet 5	88.4%

TABLE 5: The performance of DWT-SVM with different decomposition levels.

Decomposition Level	Accuracy	Precision	Recall
1	91.7%	92.8%	81.8%
2	90.5%	91.0%	80.2%
3	86.5%	83.3%	75.4%
4	87.0%	83.1%	77.8%
5	85.1%	80.8%	73.8%

5.2.3. *Statistical Method.* Driving style recognition results based on the features extracted by statistical method and SVM are shown in Table 7. With any combinations of features, the accuracy rate of the statistical method is lower than that based on DFT and DWT. The highest accuracy rate in Table 7 is 85.7% when adopting three features.

5.2.4. *Machine Learning Algorithms.* This section tests the performance of four machine learning algorithms: RF, MLP, KNN, and SVM using all three features and DWT method. The accuracy, precision, and recall rates are listed in Table 8. SVM outperforms other machine learning algorithms. Random Forest is the second best algorithm. MLP gives the highest recall rate among all candidates. KNN, as the simplest classification method, unsurprisingly obtains the worst performance.

6. Conclusion

In this study, a novel driving style labelling method is proposed to assign normal and aggressive labels based on collision risk, which is critical to sample data needed in supervised machine learning. The method is based on the vehicle trajectory extracted from traffic video. The rear-end collision risk surrogates are adopted to evaluate the

risk during the car-following process. The study also applies the SVM algorithm to recognize the driving style based on the trajectory features. Three feature extraction methods are tested. Other machine learning algorithms including RF, MLP, and KNN are also adopted to compare with the SVM. Several conclusions can be obtained from this study.

(1) Three effective rear-end collision risk surrogates, namely, ITTC, THW, and MMTC, are selected to evaluate the collision risk in the car-following process. Since THW and MMTC show a strong positive correlation, only ITTC and THW are kept to evaluate driving risk level. This paper gives threshold values of ITTC and THW based on their distribution and previous studies. Each driver's trajectory can be divided into four risk levels, and all drivers can be grouped into two classes using the K-means algorithm. Using NGSIM dataset, this method labels 246 normal drivers and 124 aggressive drivers. On average, normal drivers have 45.5% safe driving behavior, 37.5% high-risk driving behavior, and 11.4% dangerous driving behavior, and aggressive drivers have 7.4% safe driving behavior, 77.8% high-risk driving behavior, and 13.5% dangerous driving behavior.

(2) DFT, DWT, and statistical methods are adopted to extract the effective features from trajectory data to facilitate the driving style recognition. Using relative distance along DFT method can convert relative distance time series into coefficients in the frequency domain and help SVM reach the accuracy rate of 88.9%, the precision rate of 86.3%, and the recall rate of 80.2%. However, when using multiple features, including acceleration, relative distance, and relative speed, DWT method can improve the accuracy rate to 91.7%, precision rate to 92.8%, and recall rate to 81.8%. Among 15 wavelet mother functions tested, Daubechies 4 mother function provides the best results.

(3) The driving style can be accurately recognized by the proposed SVM model based on the trajectory features with 91.7% accuracy rate. The recognition accuracy is superior to other famous and frequently used classifiers: RF, MLP, and

TABLE 6: The evaluation results of driving style based on SVM using DWT.

Features	Accuracy	Precision	Recall
a_f^*	69.5%	74.1%	15.9%
v_r	83.8%	83.0%	65.9%
x_r	86.8%	88.1%	70.6%
$a_f + v_r$	82.4%	81.4%	62.7%
$v_r + x_r$	90.2%	90.2%	80.2%
$a_f + x_r$	88.7%	89.6%	75.4%
$a_f + v_r + x_r$	91.7%	92.8%	81.8%

*: using polynomial kernel function in SVM to produce better results.

TABLE 7: The evaluation results of driving style based on SVM using the statistical method.

Features	Accuracy	Precision	Recall
a_f^*	66.2%	51.6%	12.7%
v_r	79.7%	72.6%	65.1%
x_r	83.5%	80.4%	68.3%
$a_f + v_r$	78.7%	71.2%	62.7%
$v_r + x_r$	83.2%	79.6%	68.3%
$a_f + x_r$	84.6%	79.5%	73.8%
$a_f + v_r + x_r$	85.7%	79.2%	78.6%

*: using polynomial kernel function, in SVM to produce better results.

TABLE 8: The performance comparison of multiple algorithms.

Features	Algorithm	Accuracy	Precision	Recall
$a_f + v_r + x_r$	RF	91.6%	89.2%	82.0%
	MLP	88.1%	73.0%	87.3%
	KNN	87.6%	85.7%	76.2%
	SVM	91.7%	92.8%	81.8%

KNN. This result indicates that the SVM method is a more appropriate method for driving style recognition based on the trajectory features.

(4) The proposed method can be effectively used to label and recognize the driving style based on the traffic video surveillance systems. The development of network connected vehicles can help to collect the data more precisely. The model with machine learning algorithm can be trained to better recognize driving style. It can help to evaluate the collision risk on the road network and also provide real-time decision support to drivers.

This study offers the possibility of developing more sophisticated driving style recognition methods. For further work, the proposed method can be extended by selecting other features that can reflect the driving style more accurately. As we know, the driving style is also influenced by the road conditions and traffic flow level. Such results can also be used to improve the driving style recognition. It is possible to use some semi-supervised and unsupervised methods to save the label time in the future.

Data Availability

The reconstructed NGSIM dataset can be accessed at <http://www.multitude-project.eu/reconstructed-ngsim.html>. The

original NGSIM data is open to download at <https://data.transportation.gov/>.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study has been funded by the National Key Research and Development Program of China (No. 2017YFC0803902).

References

- [1] J. Elander, R. West, and D. French, "Behavioral correlates of individual differences in road-traffic crash risk: an examination method and findings," *Psychological Bulletin*, vol. 113, no. 2, pp. 279–294, 1993.
- [2] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: review and future perspectives," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 6–22, 2014.
- [3] Y. Zheng, S. E. Li, J. Wang, D. Cao, and K. Li, "Stability and scalability of homogeneous vehicular platoon: Study on the

- influence of information flow topologies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 14–26, 2015.
- [4] B. Yu, H. Wang, W. Shan et al., "Prediction of bus travel time using random forests based on near neighbors," *Computer-aided Civil and Infrastructure Engineering*, vol. 33, no. 2, 2017.
- [5] B. Yu, X. Song, F. Guan et al., "k-nearest neighbor model for multiple-time-step prediction of short-term traffic condition," *Journal of Transportation Engineering*, vol. 142, no. 6, Article ID 04016018, 2016.
- [6] J. Yao, H. Xia, B. Yu et al., "Prediction on building vibration induced by moving train based on support vector machine and wavelet analysis," *Journal of Mechanical Science and Technology*, vol. 28, no. 6, pp. 2065–2074, 2014.
- [7] Y. Zhang, W. C. Lin, and Y.-K. S. Chin, "A pattern-recognition approach for driving skill characterization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 4, pp. 905–916, 2010.
- [8] P. Molchanov, S. Gupta, K. Kim et al., "Multi-sensor system for driver's hand-gesture recognition," in *Proceedings of the IEEE International Conference and Workshops on Automatic Face and Gesture Recognition*, pp. 1–8, IEEE, 2015.
- [9] A. Hashemi, V. Saba, and S. N. Resalat, "Real time driver's drowsiness detection by processing the EEG signals stimulated with external flickering light," *Basic and Clinical Neuroscience*, vol. 5, no. 1, pp. 22–27, 2014.
- [10] H. Wang, C. Zhang, T. Shi et al., "Real-time EEG-based detection of fatigue driving danger for accident prediction," *International Journal of Neural Systems*, vol. 25, no. 2, pp. 498–369, 2015.
- [11] D. Srinivasan, X. Jin, and R. L. Cheu, "Adaptive neural network models for automatic incident detection on freeways," *Neurocomputing*, vol. 64, no. 1–4, pp. 473–496, 2005.
- [12] N. K. Singh, A. K. Singh, and M. Tripathy, "A comparative study of BPNN, RBFNN and ELMAN neural network for short-term electric load forecasting: a case study of Delhi region," in *Proceedings of the International Conference on Industrial and Information Systems*, IEEE, 2015.
- [13] M. M. Bejani and M. Ghatee, "A context aware system for driving style evaluation by an ensemble learning on smartphone sensors data," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 303–320, 2018.
- [14] G. Li, S. E. Li, B. Cheng, and P. Green, "Estimation of driving style in naturalistic highway traffic using maneuver transition probabilities," *Transportation Research Part C: Emerging Technologies*, vol. 74, pp. 113–125, 2017.
- [15] H. Berndt and K. Dietmayer, "Driver intention inference with vehicle onboard sensors," in *Proceedings of the International Conference on Vehicular Electronics and Safety, ICVES '09*, pp. 102–107, IEEE, 2009.
- [16] X. Meng, K. K. Lee, and Y. Xu, "Human driving behavior recognition based on hidden markov models," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, 2007.
- [17] T. Kumagai, Y. Sakaguchi, M. Okuwa et al., "Prediction of driving behavior through probabilistic inference," in *Proceedings of the 8th International Conference on Engineering Applications of Neural Networks*, 2003.
- [18] C. F. Zong, X. Yang, C. Wang et al., "Driver's driving intention identification and behavior prediction during vehicle steering," *Journal of Jilin University*, vol. s1, pp. 27–32, 2009.
- [19] G. S. Aoude, V. R. Desaraju, L. H. Stephens et al., "Driver behavior classification at intersections and validation on large naturalistic data set," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 724–736, 2012.
- [20] Z. Chen, C. Wu, Z. Huang et al., "Dangerous driving behavior detection using video-extracted vehicle trajectory histograms," *Journal of Intelligent Transportation Systems*, vol. 21, no. 11, 2017.
- [21] W. Sun, X. Zhang, S. Peeta et al., "A real-time fatigue driving recognition method incorporating contextual features and two fusion levels," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–13, 2017.
- [22] L. Feng, Y. Yao, and B. Jin, "Research on credit scoring model with svm for network management," *Journal of Computational Information Systems*, vol. 6, no. 11, pp. 1032–1040, 2010.
- [23] C. M. Martinez, M. Heucke, F. Y. Wang et al., "Driving style recognition for intelligent vehicle control and advanced driver assistance: a survey," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–11, 2017.
- [24] M. Wu, S. Zhang, and Y. Dong, "A novel model-based driving behavior recognition system using motion sensors," *Sensors*, vol. 16, no. 10, p. 1746, 2016.
- [25] W. Wang and J. Xi, "A rapid style-recognition method for driving styles using clustering-based support vector machines," in *Proceedings of the American Control Conference*, IEEE, 2016.
- [26] W. Wang, J. Xi, A. Chong et al., "Driving style classification using a semi-supervised support vector machine," *IEEE Transactions on Human-Machine Systems*, 2017.
- [27] C. Zhang, M. Patel, S. Buthpitiya et al., "Driver classification based on driving behaviors," in *Proceedings of the International Conference on Intelligent User Interfaces*, pp. 80–84, ACM, 2016.
- [28] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation and augmented reality tracking: An integrated system and evaluation for monitoring driver awareness," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 300–311, 2010.
- [29] B. Coifman, "Using LIDAR to validate the performance of vehicle classification stations," *Journal of Intelligent Transportation Systems*, vol. 19, no. 4, pp. 355–369, 2015.
- [30] J. Schorr, S. H. Hamdar, and C. Silverstein, "Measuring the safety impact of road infrastructure systems on driver behavior: vehicle instrumentation and real world driving experiment," *Journal of Intelligent Transportation Systems*, 2017.
- [31] R. Sun, K. Han, J. Hu, Y. Wang, M. Hu, and W. Y. Ochieng, "Integrated solution for anomalous driving detection based on BeiDou/GPS/IMU measurements," *Transportation Research Part C: Emerging Technologies*, vol. 69, pp. 193–207, 2016.
- [32] T. Liu, Y. Yang, G.-B. Huang et al., "Driver distraction detection using semi-supervised machine learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1108–1120, 2016.
- [33] N. P. Chandrasiri, K. Nawa, and A. Ishii, "Driving skill classification in curve driving scenes using machine learning," *Journal of Modern Transportation*, vol. 24, no. 3, pp. 1–11, 2016.
- [34] C. Yan, F. Coenen, Y. Yue, X. Yang, and B. Zhang, "Video-based classification of driving behavior using a hierarchical classification system with multiple features," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 30, no. 5, 2016.
- [35] M. H. Zaki, T. Sayed, and K. Shaaban, "Use of drivers' jerk profiles in computer vision-based traffic safety evaluations," *Transportation Research Record*, 2014.

- [36] C. Oh, E. Jung, H. Rim et al., "Intervehicle safety warning information system for unsafe driving events," *Transportation Research Record Journal of the Transportation Research Board*, vol. 2324, pp. 1–10, 2012.
- [37] R. Chhabra, S. Verma, and C. R. Krishna, "A survey on driver behavior detection techniques for intelligent transportation systems," in *Proceedings of the International Conference on Cloud Computing, Data Science and Engineering - Confluence*, pp. 36–41, IEEE, 2017.
- [38] F. Teimouri and M. Ghatee, "A real-time warning system for rear-end collision based on random forest classifier," 2018, <https://arxiv.org/abs/1803.10988>.
- [39] M. Ishibashi, M. Okuwa, S. Doi et al., "Indices for characterizing driving style and their relevance to car following behavior," in *Proceedings of the SICE*, 2007.
- [40] N. Li, T. Misu, and A. Miranda, "Driver behavior event detection for manual annotation by clustering of the driver physiological signals," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, 2016.
- [41] B. Zhu, W. Li, N. Bian et al., "Identification of driver individualities using random forest model," *SAE Technical Papers*, 2017.
- [42] J. Ambros, R. Turek, and J. Paukert, "Road safety evaluation using traffic conflicts: Pilot comparison of micro-simulation and observation," in *Proceedings of the International Conference on Traffic and Transport Engineering*, 2014.
- [43] S. Kitajima, Y. Marumo, T. Hiraoka et al., "Comparison of evaluation features concerning estimation of driver's risk perception," *Transactions of the Society of Automotive Engineers of Japan*, vol. 40, no. 2, pp. 191–198, 2009.
- [44] S. M. S. Mahmud, L. Ferreira, M. S. Hoque et al., "Application of proximal surrogate indicators for safety evaluation: a review of recent developments and research needs," *IATSS Research*, vol. 41, no. 4, 2017.
- [45] H. J. Hogema and W. H. Janssen, "EFFECTs of intelligent cruise control on driving behavior: a simulator study," in *Proceedings of the Intelligent Transportation: Realizing the Future Abstracts of the 3rd World Congress on Intelligent Transport Systems*, 1996.
- [46] Q. Meng and X. Qu, "Estimation of vehicle crash frequencies in road tunnels," *Accident Analysis & Prevention*, vol. 48, no. 5, pp. 254–263, 2012.
- [47] T. Sayed, M. H. Zaki, and J. Autey, "Automated safety diagnosis of vehicle-bicycle interactions using computer vision analysis," *Safety Science*, vol. 59, pp. 163–172, 2013.
- [48] L. Yang, C. F. Xing, and H. B. Zhao, "Study on driver's reaction time (DRT) during car following," *Computing Technology and Automation*, vol. 34, no. 3, pp. 33–37, 2015.
- [49] V. G. Kovvali, V. Alexoadis, and P. E. Zhang, "Video-based vehicle trajectory data collection," in *Proceedings of the Transportation Research Board 86th Annual Meeting*, 2007.
- [50] Z. He, "Research based on high-fidelity NGSIM vehicle trajectory datasets: a review," *Research Gate*, pp. 1–33, 2017.
- [51] M. Montanino and V. Punzo, "Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns," *Transportation Research Part B: Methodological*, vol. 80, pp. 82–106, 2015.
- [52] J. Wahlström, I. Skog, and P. Handel, "Smartphone-based vehicle telematics: a ten-year anniversary," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2802–2825, 2017.
- [53] R. B. Zadeh, M. Ghatee, and H. R. Eftekhari, "Three-phases smartphone-based warning system to protect vulnerable road users under fuzzy conditions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2086–2098, 2018.
- [54] H. R. Eftekhari and M. Ghatee, "Hybrid of discrete wavelet transform and adaptive neuro fuzzy inference system for overall driving behavior recognition," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 58, pp. 782–796, 2018.
- [55] H. R. Eftekhari and M. Ghatee, "An inference engine for smartphones to preprocess data and detect stationary and transportation modes," *Transportation Research Part C: Emerging Technologies*, vol. 69, pp. 313–327, 2016.

Research Article

Optimizing Location of Car-Sharing Stations Based on Potential Travel Demand and Present Operation Characteristics: The Case of Chengdu

Yu Cheng ¹, Xu Chen,¹ Xiaohua Ding,¹ and Linting Zeng²

¹Shanghai Electric Vehicle Public Data Collecting Monitoring and Research Centre, 5F, No. 888, South Moyu Road, Jiading, Shanghai 201805, China

²School of Environmental and Geographical Sciences, Shanghai Normal University, Shanghai, China

Correspondence should be addressed to Yu Cheng; chengyu@shevdc.org

Received 11 September 2018; Accepted 18 December 2018; Published 20 January 2019

Guest Editor: Ali Tizghadam

Copyright © 2019 Yu Cheng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Car-sharing is becoming an increasingly popular travel mode in China and many companies invest plenty of money on that including vehicle enterprises and Internet companies. But most of them build car-sharing stations by their experience or randomly as long as there is parking space in the early development of their business. This results in many stations with low operational efficiency and causes capital loss. This study aims to use different data source with statistical models and machine learning algorithm to help car-sharing operator to choose the optimal location of new stations and adjust the location of existing stations. We select Chengdu where there are huge amounts of car-sharing travel demand and several large car-sharing operators as the research area and two main operators as the research objects. Chengdu is divided into 58724 squared grids each of which is 0.5km*0.5km instead of focusing on the buffers generated by stations. We try to find a model to estimate a potential travel demand value for each small grid with three data sources: order data, population data, and Point of Interest (POI) data. This problem is transformed into a binary form and five different methods, Logistic Regression, Logistic Regression with LASSO, Naive Bayes, Linear Discriminant Analysis, and Quadratic Discriminant Analysis, are implemented. The optimal model, Logistic Regression with LASSO, is chosen to estimate the probability of existence of demand in all grids. With car-sharing order data from different operators, an existing order heat value is also computed for each grid. Then we analyze and classify all the grids into four groups. For different groups of grids, we give different suggestions on the optimal location of stations. This study focuses on a more competitive market and finds the influential factors on order number. Suggestions on the optimal location of stations are given in consideration of competitors. We hope that our research can help operators improve their business and make rational plans.

1. Introduction

Car-sharing is a mode that allows members to access a fleet of vehicles for short-term use without actual ownership. Members just need to reserve a vehicle online or by mobile app and then move to the parking lots and drive the car. They usually pay for this after travelling according to the travelling distance or/and time [1, 2]. This kind of new service allows people to avoid buying a car and spending time finding parking lots [3]. Car-sharing is becoming an increasingly important travelling mode in recent 5 years especially with rapid growth of electric vehicles since 2016. The advantages of car-sharing include reducing vehicle ownerships, reducing

vehicle kilometres travelled, and reducing greenhouse gas emission [4–6].

So far, there are three main types of car-sharing mode: station-based, free-floating, and peer-to-peer car-sharing [7]. Station-based system requires the operators to hire parking space for stations and the vehicles are parcelled out among these stations. According to the operation mode, there are two types of station-based form: one-way car-sharing and round-trip car-sharing. One-way car-sharing allows the customers to return cars at any designated station wherever the trip started. In contrast, with round-trip car-sharing, cars should be returned to the station where the trip starts. The first form provides more flexibility but will cause the

TABLE I: Summary of average daily orders of operator F and operator H in grids.

Operator	Mean	Std.Error	Min	Max
operator F	1.2	2.4	0	37.4
operator H	2.8	4.2	0	49.7
Total	4.0	4.8	0	53.7

problem of spatial imbalance over the stations. The operators have to consider relocation problem that leads to much cost. Free-floating system operates without station. Instead, the operators define an area where the customers can park cars at any parking lots [7, 8]. Free-floating car-sharing could also be treated as a special case of one-way car-sharing [9].

Our research will mainly focus on one-way station-based car-sharing system for which station siting is a big challenge and need to take quite a lot of factors into consideration. A good choice can bring high efficiency, large profit, low operation cost, and competitive advantages when competing with other operators [10, 11]. However, at the beginning of business, operators usually locate stations by their experience. For example, trading area, universities, or airports always bring great profit if there are car-sharing stations. But different cities have different features and attitudes towards car-sharing. Therefore, experience sometimes may be unreliable and relocations are necessarily introduced when the business is mature. What is more, some operators just choose locations randomly as long as there are parking spaces depending on their huge funds in order to seize the market ahead of other competitors, where relocations are even more necessary.

Researchers have done a lot of work on relocation or site selection problem by using various methods. Analytic hierarchy process (AHP) is the most popular method among multicriteria decision making methods [12]. One study uses AHP to solve the site selection problem for EVCARD which is a car-sharing operator in Shanghai. The researchers consider the potential users, potential travel demand, potential travel purposes, and distances from existing stations totally 15 factors as the decision criteria. But this method is based on candidate stations and expert scoring method that is subjective [11]. Mathematical and statistical models are also applied. Several studies use multilinear model and mixed-integer programming model to find the optimal location of stations [13–15]. Another paper introduces an intensity model that estimates the demand and an imbalance model that describes the difference between pick-up and drop-off to find the optimal location of stations of EVCARD in Shanghai, which involves usage intensity, usage imbalance, transportation information, and built environment. These two models utilize a combination of Elastic Net, the adaptive Least Absolute Shrinkage and LASSO [3].

The previous two studies about site location of EVCARD all focus on the market in Shanghai where there is no competitor. Therefore, it is much easier to make decision on the optimal site location. Our research will focus on a more complicated market: Chengdu where there are more than five car-sharing operators and each has its own advantages. In Chengdu, there are two main car-sharing operation companies that account for a majority of market shares. For privacy,

we call them operator F and operator H. They are both station-based car-sharing operators. The differences between them are the number of electric vehicles and stations, vehicle models, and charging mode. We will consider the potential demand heat combined with the existing order heat to give suggestions on site location.

2. Data

We divide Chengdu into 58724 squared grids each of which has an area of 0.5km*0.5km. All the data we use are allocated to these grids. The data sources consist of 3 parts. The first one is the order data. We collect the order data from these two car-sharing operator mobile apps and compute the average daily order numbers for each car-sharing point. The time period is between March 28, 2018, and April 17, 2018. One order is only considered once at the beginning point. For each grid, we sum up the day-average order numbers of all points according to these two car-sharing operation companies in that grid. There are 1834 grids with car-sharing stations. The summary of these grids is shown in Table 1.

The second data source is the Point of Interest (POI) information of Chengdu that is collected from AMAP in July 2018. AMAP is an e-map that is similar to Google map and is widely used in China. However, the coordinate system of AMAP is called “GCJ-02” which is different from “WGS-84” coordinate system used in Google map. So a coordinate transformation work is introduced to ensure that the POI information is based on “WGS-84” coordinate system. The total number of POIs reaches 860196 and they are categorized into fourteen classes shown in the first column of Table 2. In AMAP POI categories, auto service contains various services such as filling station, auto-mobile rental, and charging station. But here we only consider one subcategory: auto-mobile rental since car-rental service will affect the demand of car-sharing service. Additionally, we separate transportation service into five parts: Bus Station, Underground Station, Train Station, Airport, and Parking Lots. Again we allocate these POIs to these grids and the summary of them is shown in Table 2.

The last data source used in our research is the population data (POP) called Gridded Population of the World (GPW) from Socioeconomic Data and Applications Center in NASA. Now it is the fourth version that models the population counts and densities on a continuous global raster surface. The population data is collected from the population and housing censuses between 2005 and 2014, which is used to estimate the population for the year 2000, 2005, 2010, 2015, and 2020. Some adjustments of a set of estimates based on national level, historic, and future, population predictions from the United Nation’s World Population Prospects report

TABLE 2: Summary of POI information in grids.

Category	Abbreviation	Type	Mean	Std.Error	Max	Min	
Auto Service	Car-rental Service	CRS	Numerical	0.0221	0.1836	7	0
	Food & Beverages	FB	Numerical	2.4100	12.5445	373	0
	Shopping	SH	Numerical	4.9470	26.9275	900	0
	Daily Life Service	DS	Numerical	2.4800	11.7907	226	0
	Sports & Recreation	SR	Numerical	0.3965	1.9526	78	0
	Medical Service	MS	Numerical	0.6307	3.3228	148	0
	Accommodation Service	AS	Numerical	0.3428	3.6490	251	0
	Tourist Attraction	TA	Numerical	0.0609	0.4368	29	0
	Commercial House	CH	Numerical	0.4638	2.4126	63	0
	Governmental Organization & Social Group	GS	Numerical	0.4416	2.1701	78	0
	Science/Culture & Education Service	SS	Numerical	0.5213	2.7930	117	0
Transportation Service	Bus Station	BS	Numerical	0.2033	0.6535	10	0
	Underground Station	US	Numerical	0.0053	0.0727	1	0
	Train Station	TS	Numerical	0.0013	0.0359	1	0
	Airport	AP	Numerical	0.0003	0.0184	1	0
	Parking Lots	PL	Numerical	0.6332	2.9611	81	0
	Finance & Insurance Service	FS	Numerical	0.0874	0.5926	27	0
	Enterprises	EN	Numerical	1.2750	7.1559	358	0

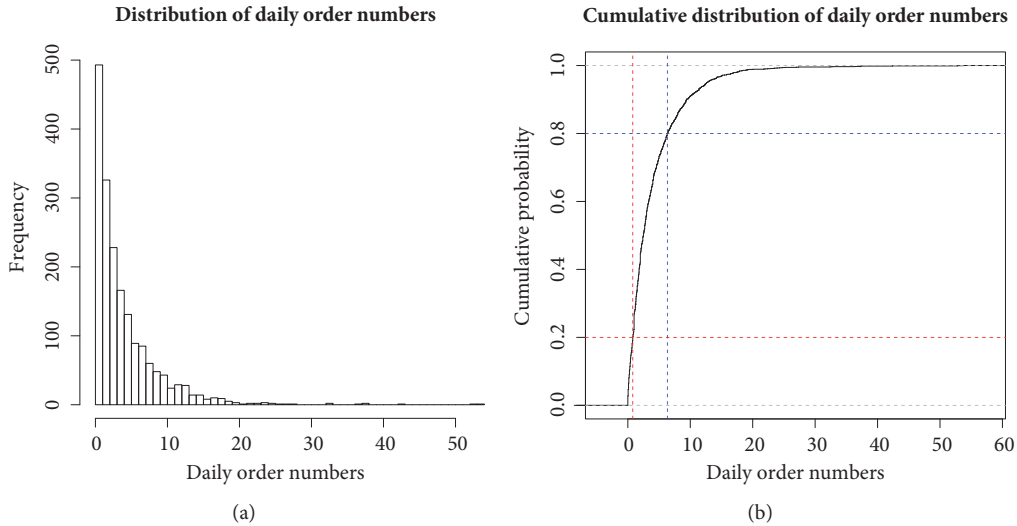


FIGURE 1: The distribution of average daily order numbers. (a) The histogram of average daily order numbers. (b) The cumulative probability distribution of average daily order numbers.

(2015 Revision) are also introduced to these sets of years. GPW is gridded with an output resolution of 30 arc-seconds which is approximately equal to 1 km at the equator. The value for each grid is not the population number in it but a scale that reflects the level of population. In our research we use the population density in 2015 and adjust the rasters to ours as well as the population density of them [16].

3. Methodology

The true value of the potential demand is unknown since it is an insubstantial concept which is difficult to measure. One possible way is to do a full sample questionnaire but this requires high human and financial resources and the results

might be biased since questionnaire contains much subjectivity. Another way is to use usage intensity as a proxy of demand [3] or use the number of bookings while this amount can only reflect the present demand that may be restricted by the number of car-sharing stations and vehicles. Therefore using a specific amount to represent the potential demand will cause deviation. An alternative way that is applied in our research is just to distinguish whether the demand of car-sharing in a grid exists or not. Therefore, the question becomes a classification problem where classification algorithms can be implemented. So far 1834 grids contain at least one car-sharing station. The distribution of order numbers in them is shown in Figure 1(a). Clearly, those with larger order numbers can be treated as high demand. Those with tiny

order numbers are reasonably defined as no demand since very small order number reflects an occasionally demand that operators do not need to satisfy with cost of renting parking space. Therefore, we choose the lower 20% and upper 20% grids as the sample set (as shown in Figure 1(b)) and let respond equal "1" for demand and "0" for no demand. What is more, we also would like to know the level of demand in a grid which is the probability that the grid belongs to the class "1". Classification algorithms such as k-nearest neighbour (KNN) and tree models cannot provide us such probability so we choose the following four methods: Logistic Regression, Naive Bayes, Linear Discriminant Analysis, and Quadratic Discriminant Analysis.

3.1. Logistic Regression with LASSO. Logistic Regression is used to find the relationship between variables $\mathbf{X} = (X_1, X_2, \dots, X_p)$ and binary response Y . It models the probability that Y belongs to a particular class. If we use a linear regression model

$$p(\mathbf{X}) = \Pr(Y = 1 | \mathbf{X}) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p \quad (1)$$

to represent the probability, we might get a result that is smaller than 0 or larger than 1, which does not make sense for probability. Therefore, in order to get an output between 0 and 1, we use the logistic function

$$p(\mathbf{X}) = \frac{e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}} \quad (2)$$

Then we have

$$\frac{p(\mathbf{X})}{1 - p(\mathbf{X})} = e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p} \quad (3)$$

By taking the logarithm of both sides, we have

$$\log\left(\frac{p(\mathbf{X})}{1 - p(\mathbf{X})}\right) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p \quad (4)$$

We can see that the Logistic Regression model has a log-odds which is linear in \mathbf{X} .

The coefficients of Logistic Regression models are usually estimated by maximum likelihood method. The likelihood function for N observations is

$$L(\boldsymbol{\beta}) = \prod_{i:y_i=1} p(\mathbf{X}_i) \prod_{i':y_{i'}=0} p(\mathbf{X}_{i'}) \quad (5)$$

where

$$\begin{aligned} \boldsymbol{\beta} &= (\beta_0, \beta_1, \dots, \beta_p) \\ \mathbf{X}_i &= (X_{i1}, X_{i2}, \dots, X_{ip}) \end{aligned} \quad (6)$$

The log-likelihood can be written as

$$\begin{aligned} l(\boldsymbol{\beta}) &= \sum_{i=1}^N \{y_i \log p(\mathbf{X}_i; \boldsymbol{\beta}) + (1 - y_i) \log (1 - p(\mathbf{X}_i; \boldsymbol{\beta}))\} \\ &= \sum_{i=1}^N \left\{ y_i \boldsymbol{\beta}^T \mathbf{X}_i - \log(1 + e^{\boldsymbol{\beta}^T \mathbf{X}_i}) \right\} \end{aligned} \quad (7)$$

Set its derivatives with respect to $\boldsymbol{\beta}$ to zero and we can get the maximum likelihood estimators for $\boldsymbol{\beta}$. [17]

The LASSO is a shrinkage method that constrains the coefficient estimates, which can significantly reduce the variance of them. A penalized term $\lambda \sum_{j=1}^p |\beta_j|$ is added to the model, where $\lambda \geq 0$ is a tuning parameter.

Therefore, for **Logistic Regression with LASSO**, we would maximize the penalized log-likelihood function:

$$\max_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^N [y_i \boldsymbol{\beta}^T \mathbf{X}_i - \log(1 + e^{\boldsymbol{\beta}^T \mathbf{X}_i})] - \lambda \sum_{j=1}^p |\beta_j| \right\} \quad (8)$$

The selecting of a good value of λ is critical since it significantly affects the coefficients. Cross-validation method is applied to choose the best λ that produces the smallest cross-validation error [18].

3.2. Naive Bayes. Assume that there are K classes and we wish to classify an observation \mathbf{X} into one of them. According to the Bayes' theorem,

$$\Pr(Y = k | \mathbf{X} = \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^K \pi_l f_l(\mathbf{x})} \quad (9)$$

where $\Pr(Y = k | \mathbf{X} = \mathbf{x})$ denotes the probability that \mathbf{X} belongs to the class k ; π_k is the prior probability that a random observation is from class k ; $f_k(\mathbf{x}) = \Pr(\mathbf{X} = \mathbf{x} | Y = k)$ is the probability density function of \mathbf{X} which belongs to class k .

For an observation $\mathbf{X} = (x_1, x_2, \dots, x_p)$, **Naive Bayes** assumes that each feature is independent given a class k . So we have

$$f_k(\mathbf{x}) = \prod_{j=1}^p f_{kj}(x_j) \quad (10)$$

where $f_{kj}(x_j)$ is the probability density function of the j -th feature given class k . Then the probability of \mathbf{X} coming from the class k is

$$\begin{aligned} \Pr(Y = k | \mathbf{X} = \mathbf{x}) &= \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^K \pi_l f_l(\mathbf{x})} \\ &= \frac{\pi_k \prod_{j=1}^p f_{kj}(x_j)}{\sum_{l=1}^K \pi_l \prod_{j=1}^p f_{lj}(x_j)} \end{aligned} \quad (11)$$

See [18].

3.3. Linear Discriminant Analysis. Linear Discriminant Analysis is based on Bayes' theorem and assumes that the observation $\mathbf{X} = (x_1, x_2, \dots, x_p)$ is generated from a multivariate Gaussian distribution with a unique class mean vector and a common covariance matrix. We write $X \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu}_k$ is the mean vector for the class k and $\boldsymbol{\Sigma}$ is the covariance matrix that is the same for all classes. Then the probability density function (pdf) of \mathbf{X} that is from class k is

$$\begin{aligned} f_k(\mathbf{x}) &= \Pr(\mathbf{X} = \mathbf{x} | Y = k) \\ &= \frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right) \end{aligned} \quad (12)$$

TABLE 3: Mean and standard error of all predictors.

	CRS	FB	SH	DS	SR	MS	AS	TA	CH	GS
Mean	0.3	32.35	57.49	29.31	4.93	6.32	5.83	0.37	5.12	4.14
Std.error	0.67	43.51	90.96	34.67	6.83	8.32	14.78	1.24	7.51	6.77
	SS	BS	US	TS	AP	PL	FS	EN	POP	
Mean	7.34	1.51	0.072	0.008	0.004	8.89	1.23	16.24	1882	
Std.error	11.35	1.4	0.26	0.09	0.06	10.09	2.23	33.81	5621.53	

TABLE 4: The results of Logistic Regression.

	Coefficient	Std.error	P value
Intercept	0.2569	0.1211	0.0338
FB	0.9960	0.2404	<0.001
MS	-0.3401	0.1581	0.0315
GS	-0.5480	0.1383	<0.001
BS	0.2814	0.1307	0.0313
PL	1.3981	0.2182	<0.001

The parameters of the multivariate Gaussian distributions are unknown and we need to estimate them from training data by

$$\begin{aligned}
\hat{\pi}_k &= \frac{N_k}{N}; \\
\hat{\boldsymbol{\mu}}_k &= \sum_{g_i=k} \frac{\mathbf{x}_i}{N_k}; \\
\hat{\boldsymbol{\Sigma}} &= \sum_{k=1}^K \sum_{g_i=k} \frac{(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T}{(N - K)}
\end{aligned} \quad (13)$$

where N_k is the number of observations in class k and g_i is the label of i -th observation [17].

3.4. Quadratic Discriminant Analysis. Quadratic Discriminant Analysis is a bit different from Linear Discriminant Analysis, which assumes that the covariance matrices for each class are different. Then the observation \mathbf{X} from class k has the distribution of $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. So the pdf becomes

$$\begin{aligned}
f_k(\mathbf{x}) &= \frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right) \quad (14)
\end{aligned}$$

See [17].

4. Results and Discussion

As mentioned in the Section 3, we take the lower 20% and upper 20% grids as the sample set. The quantiles are 0.79 and 6.33, respectively, which means that the grids with average daily order numbers that are less than 0.79 and larger than 6.33 are chosen as the samples. The sample size is 737 of which 370 samples are of no demand and 367 are of high demand. We randomly choose 500 samples (approximately 65% of total samples) as the training set and the remaining as the test set. Since population density and POI information are

in different scale, all variables are normalized by their mean and standard error (shown in Table 3).

4.1. Logistic Regression. The results of Logistic Regression are shown in Table 4. Only five variables, Food& Beverages, Medical Service, Governmental Organization & Social Group, Bus Station, and Parking Lots, are significant under 5% significance level. The population factor is not significant. Food& Beverages positively affects the probability of existence of demand, while Medical Service and Governmental Organization& Social Group has negative effects. According to our search and investigation, the car-sharing stations in Governmental Organization are always of limited access which means only staffs that work in them can get access to the stations. Once a car is returned at these stations, it always takes a long time to receive a next order. The negative effect of Medical Service could be attributed to the limited parking space and huge parking demand. The stations might be occupied by fuel vehicles. The coefficient of Bus Station is also positive, which is the same as the results in literatures [3, 19]. Parking Lots has a positive impact on the probability of existence of demand, which implies that more space provide more opportunities to build car-sharing stations. However, Chen's paper [3] concludes that more parking space leads to more private vehicle trips instead of car-sharing service.

4.2. Logistic Regression with LASSO. Logistic Regression with LASSO introduces a penalized term to the model. The results are shown in Table 5. Since LASSO is a shrinkage method that the coefficients of nonsignificant variables will shrink to zero, there is no p-value for this method and p-value also does not make sense for biased regressions such as LASSO. λ equals 0.01973 that minimizes the model deviance. Seven variables, Food & Beverages, Sports & Recreation, Governmental Organization & Social Group, Bus Station, Train Station, Airport, and Parking Lots, give nonzero coefficients. Comparing to the Logistic Regression results, Food& Beverages, Governmental

TABLE 5: The results of Logistic Regression with LASSO.

	Intercept	FB	SR	GS	BS	TS	AP	PL
Coefficient	0.1318	0.4652	0.1587	-0.2717	0.1641	0.01721	0.0336	1.0114

TABLE 6: Mean vectors for classes: demand=0 and demand=1.

	CRS	FB	SH	DS	SR	MS	AS	TA	CH	GS
Class 0	-0.2109	-0.4108	-0.2850	-0.3978	-0.3692	-0.2867	-0.2561	-0.1173	-0.3205	-0.0866
Class 1	0.1552	0.3963	0.3203	0.4089	0.3401	0.2754	0.2615	0.0390	0.3431	0.1573
	SS	BS	US	TS	AP	PL	FS	EN	POP	
Class 0	-0.3451	-0.2901	-0.0778	-0.0905	-0.0014	-0.4602	-0.2755	-0.2301	-0.1000	
Class 1	0.3720	0.2864	0.0947	0.0435	0.0622	0.5125	0.3065	0.3012	0.1076	

TABLE 7: Mean and standard deviation for the predictors grouped by class.

Variable	Class 0		Class 1	
	Mean	Std.error	Mean	Std.error
CRS	-0.2109	0.7259	0.1552	1.1169
FB	-0.4108	0.6177	0.3963	1.1262
SH	-0.2850	0.6372	0.3203	1.2895
DS	-0.3978	0.6668	0.4089	1.1068
SR	-0.3692	0.5560	0.3401	1.0806
MS	-0.2867	0.8778	0.2754	1.0337
AS	-0.2561	0.3561	0.2615	1.3175
TA	-0.1173	0.5521	0.0390	0.6665
CH	-0.3205	0.7296	0.3431	1.1695
GS	-0.0866	0.9128	0.1573	1.1696
SS	-0.3451	0.4995	0.3720	1.2910
BS	-0.2901	0.8360	0.2864	1.0089
US	-0.0778	0.8589	0.0947	1.1439
TS	-0.0905	0.0000	0.0435	1.2157
AP	-0.0014	0.9907	0.0622	1.4038
PL	-0.4602	0.6131	0.5125	1.1715
FS	-0.2754	0.6450	0.3065	1.2305
EN	-0.2301	0.5698	0.3012	1.3990
POP	-0.1000	0.7764	0.1076	1.1253

Organization & Social Group, Bus Station, and Parking Lots have the same effect. Medical Service is no longer significant in this model; instead Sports & Recreation positively affects the probability of existence of demand. The reason could be that Sports & Recreation is mostly visited by young people who have high acceptance and usage intensity of car-sharing mode [20]. In this model Train Station and Airport are also significant which could be attributed to their functions of traffic hub that brings high exposure rate and large mobile population. The population factor is again not significant.

4.3. Linear Discriminant Analysis. As mentioned in Section 3.3, Linear Discriminant Analysis (LDA) assumes that the sample x_i is from a multivariate Gaussian distribution where the mean vector is unique for different class and the

covariance matrix is the same. Table 6 shows the mean vectors for classes: demand = 0 and demand = 1. The covariance matrix is shown in Appendix A. The prior for these two classes are $\pi_0 = 0.502$ and $\pi_1 = 0.498$.

4.4. Quadratic Discriminant Analysis. The mean vector of Quadratic Discriminant Analysis (QDA) is the same of LDA. However, QDA assumes that the covariance matrix is different for these two classes. The results are shown in Appendix B.

4.5. Naive Bayes. Naive Bayes assumes that each predictor is independent and can be represented by a Gaussian distribution within each group. The mean and standard deviation for other variables grouped by class are shown in Table 7. The Train Station is a constant within class: demand = 0 since

TABLE 8: AUC value and accuracy rate for the previous five models.

	AUC value	Accuracy Rate
Logistic regression	0.8500	0.7722
Logistic regression with LASSO	0.8545	0.7637
LDA	0.8513	0.7553
QDA	0.8020	0.6835
Naive Bayes	0.8146	0.7215

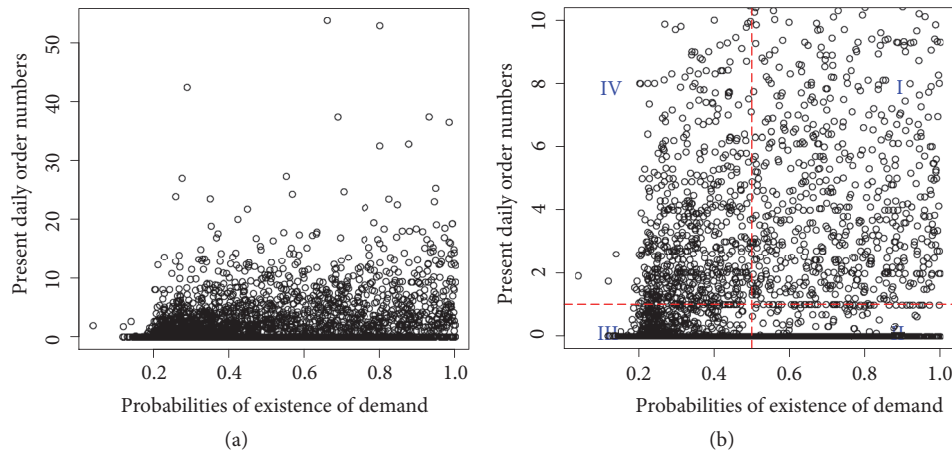


FIGURE 2: The predictive probabilities of existence of demand versus the present order numbers. (a) The whole 58725 squared grids. (b) The grids with order numbers less than 10.

the standard deviation is zero. The distributions of them are shown in Appendix C.

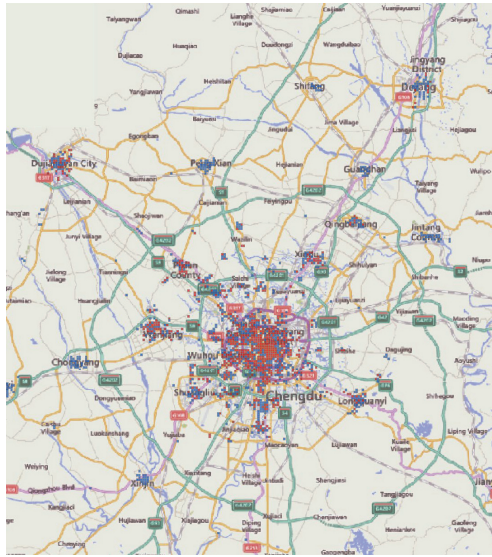
4.6. Comparing These Five Models. To judge the performance of a classifier, AUC value and accuracy rate are always applied. AUC value is the area under receiver operating characteristic (ROC) curve that varies between 0 and 1. The higher this value, the better the discrimination will be. Accuracy rate is computed by the number of correct predictions divided by the total number of predictions. We test these models with the remaining 237 observations. The values of these two measures for the previous five models are shown in Table 8. We can see that all models give an AUC value between 0.8 and 0.9 and accuracy rate between 0.65 and 0.8. QDA model produces the least AUC value and accuracy rate while the pure Logistic Regression or Logistic Regression with LASSO performs best since their AUC value and accuracy rate are the largest. LDA model and Naive Bayes model are a bit worse than Logistic Regression. Therefore, we can choose either the pure Logistic Regression or Logistic Regression with LASSO as the final model. Here we choose the Logistic Regression with LASSO model.

4.7. Optimizing Location of Car-Sharing Stations. For all of the 58724 squared grids, we normalize them with the mean and standard error in Table 3 and then run the Logistic Regression with LASSO model. The predictive probabilities of existence of demand for these grids against present order numbers are shown in Figure 2. The left figure (a) shows

the results for the whole 58724 squared grids. We can see that most grids are with order numbers less than 10 so to be clear we plot these grids in figure (b). We define predictive probability larger than 0.5 as high demand heat and otherwise low demand heat. What is more, it is reasonable to consider average daily order numbers less than 1 as low order heat and otherwise high order heat. Therefore, the grids can be sorted in to 4 groups as shown in Figure 2(b): I: high demand heat and high order heat; II: high demand heat and low order heat; III: low demand heat and low order heat; IV: low demand heat and high order heat. For stations in group III grids, operators are advised to close or remove them after investigation such as checking the time interval between two orders. For stations in group IV, further work is required to check if other influential factors that are omitted here exist since low demand heat and high order heat mutually conflict. We will mainly focus on groups I and II when optimizing the locations of car-sharing stations. Figure 3 shows the grids with high demand heat in Chengdu. The red colour refers to group I and the blue colour refers to group II. It is obvious that these grids concentrate in the city centre and town centre that is also the gathering area of crowd and business.

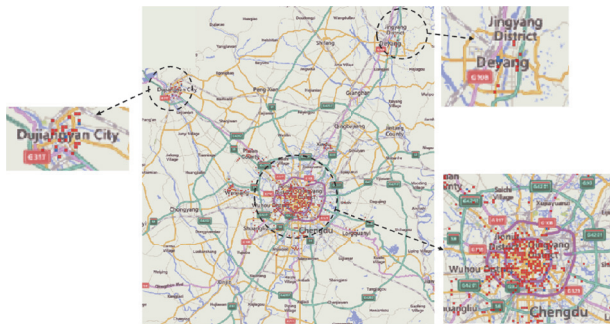
For the grids in groups I and II, we can give suggestions to the two operators: operator F and operator H on the optimal location of car-sharing stations. Three cases of grids are considered:

- (i) Case 1: grids with no operator F stations and at least one operator H stations



■ High Order Heat
■ Low Order Heat

FIGURE 3: Grids with high demand heat in Chengdu: red for grids with high order heat; blue for grids with low order heat.



■ High Order Heat
■ Low Order Heat

FIGURE 4: Case 1: grids with no operator F stations and at least one operator H stations.

- (ii) Case 2: grids with no operator H stations and at least one operator F stations
- (iii) Case 3: grids with no operator H stations and no operator F stations

Figures 4–6 display these three cases. In Case 1, we can see that operator H occupies most of the space in city centre and a majority of grids are of high order heat which means the operation effect is quite good for operator H in these grids. Only several grids are of low order heat where operator F can consider building stations. In Case 2, hundreds of grids of high demand heat surrounding the city centre are monopolized by operator F. However in the north-west of Chengdu, most grids are of low order heat, which suggests an opportunity for operator H to build stations in them.



■ High Order Heat
■ Low Order Heat

FIGURE 5: Case 2: grids with no operator H stations and at least one operator F stations.

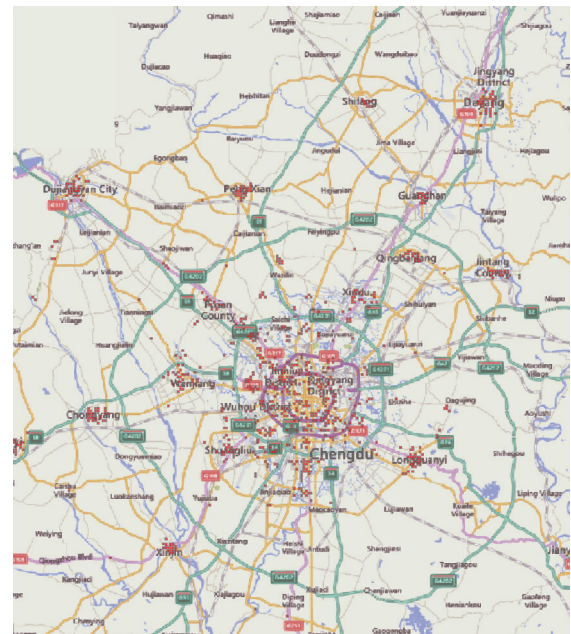


FIGURE 6: Case 3: grids with no operator H stations and no operator F stations.

Moreover, in Case 3, we find that these two operators have not entered town centres yet where most of these grids are of high demand heat. These two operators are advised to do investigation at these areas and consider building stations.

5. Conclusion

This research focuses on optimizing the car-sharing stations in Chengdu market. The main methodology is trying to estimate the potential demand combining with the present order numbers. Unlike the previous research that applies multiple linear regression to model the demand, this study transforms the question to a binary problem whether the demand exists or not. Then five classification models are

introduced to model this and estimate the probability of existence of demand. Three data sources, average daily order numbers from operator F and operator H, POI information, and population data, are used. From the five models we summarize as follows:

- (i) In the Logistic Regression, Food& Beverages, Bus Station, and Parking Lots have positive effect on the probability of existence of demand, while Medical Service and Governmental Organization & Social Group have negative effect.
- (ii) The Logistic Regression with LASSO model indicates that Food& Beverages, Sports& Recreation, Bus Station, Train Station, Airport, and Parking Lots positively affect the probability of existence of demand and Governmental Organization & Social Group have opposite effect. This model result contains the traffic hubs that could increase the demand of car-sharing service.
- (iii) Linear Discriminant Analysis (LDA) model and Quadratic Discriminant Analysis (QDA) model compute the prior probability that is 0.502 for class: demand = 0 and 0.498 for class: demand = 1. The mean vector and covariance matrix of multinomial distribution are then estimated to compute the posterior.
- (iv) In the Naive Bayes model, each variable is assumed to be normally distributed of which the mean and standard deviation are estimated with each class. The standard deviation of Train Station in class:demand = 0 is zero; thus it is treated as a constant.
- (v) Comparing the performance of these five models by AUC value and accuracy rate, we find that QDA models give the worst estimation while Logistic Regression and Logistic Regression with LASSO perform best. LDA is a bit worse than these two models and Naive Bayes works slightly better than QDA. Therefore we conclude that linear models work better in our case.

The Logistic Regression with LASSO is chosen as the final model and used to estimate the probability of existence of demand for all grids. The predictive probability larger than 0.5 is treated as high demand heat and otherwise low demand heat. These grids with high demand heat are concentrated in the city centre or town centre. Together with the present order numbers, 4 groups of grids are defined. Different suggestions on optimizing location of car-sharing stations are given for each group. Both operator F and operator H are advised to build stations in the absent grids with high demand heat and close or remove part of stations in the grids with low demand heat and low order heat. Operator H is also advised to build stations in the north-west of Chengdu where high demand exists and operator F has low operation efficiency.

However, there are still some limitations in our research.

- (i) First, Chengdu is a competitive market with more than five car-sharing operators of which only two are

considered in our research. The suggestions may be reliable without consideration of the other operators.

- (ii) Second, our research is based on 500m*500m squared grids which are suitable in city centre but are too small for other areas since the building density is much lower in these areas.
- (iii) Third, one important factor, cost of building stations, is not considered in our search. The cost always includes renting the parking space, building charging piles, and purchase electric vehicle. The definition of high order heat and low order heat in our research is simply based on the order number equalling one. However, high cost stations require high order numbers to redeem the cost. Therefore, such definition should vary with the cost of building stations.
- (iv) Fourth, samples are chosen based on the lower 20% and upper 20% grids according to average daily order numbers subjectively. Other approaches can be applied such as lower 30% and upper 30% to include more observations. The definition of 4 groups is also subjective. More objective method such as clustering may be applied.
- (v) Fifth, the models we use are all based on strong assumptions. More technical classification algorithm can be applied such as bagging, boosting, random forest, and Gaussian Process.
- (vi) Sixth, the order numbers only consider the one that is placed and omit the return behaviour. One station may have low pick-up orders while having high drop-off orders which is called usage imbalance. The decision on such station should be made carefully.
- (vii) Seventh, variable selection work can be done before we train the classification model since not all predictors are related to the response.

Furthermore, it is worth investigating the effect of adopting those suggestions by estimating the order numbers when a certain number of stations are added. One possible way is to first find the features of relationship between start and end grids and compute the intensity. Then for a grid with some new added stations, possible related grids can be found and the order number can also be estimated by intensity. Another possible way is to do transportation simulation where all of the transportation modes, the population, and the traffic operation are considered, which is a huge and challenging work.

Appendix

Covariance matrices for LDA and QDA and distributions of predictors for Naive Bayes.

A. Covariance Matrix for LDA

Covariance matrix for LDA

	CRS	FB	SH	DS	SR	MS	AS	TA	CH	GS	SS	BS	US	TS	AP	PL	FS	EN	POP
CRS	0.8858	0.1875	0.1750	0.2679	0.2222	0.2414	0.1375	0.0631	0.2409	0.1502	0.3285	0.1166	0.1639	0.0592	0.0482	0.2607	0.3043	0.2905	0.0820
FB	0.1875	0.8231	0.6025	0.6610	0.5310	0.4246	0.5320	0.1824	0.4775	0.3055	0.5137	0.1779	0.0742	0.0073	-0.0635	0.4620	0.3385	0.2653	0.0698
SH	0.1750	0.6025	1.0319	0.6486	0.4036	0.3567	0.3837	0.1518	0.4724	0.3155	0.4539	0.1889	0.0750	0.0246	-0.0471	0.4115	0.3921	0.3918	0.0058
DS	0.2679	0.6610	0.6486	0.8333	0.4571	0.5520	0.4881	0.1294	0.5501	0.3967	0.5319	0.2660	0.0577	0.0087	-0.0495	0.5160	0.4187	0.3387	0.0869
SR	0.2222	0.5310	0.4036	0.4571	0.7367	0.2411	0.4286	0.1848	0.3555	0.2912	0.5436	0.1570	0.0964	-0.0090	-0.0641	0.3969	0.3657	0.2782	0.0503
MS	0.2414	0.4246	0.3567	0.5520	0.2411	0.9189	0.2059	0.0819	0.4881	0.3090	0.3164	0.2612	0.0909	-0.0559	-0.0763	0.4362	0.3097	0.0660	0.1425
AS	0.1375	0.5320	0.3837	0.4881	0.4286	0.2059	0.9281	0.1902	0.5106	0.3763	0.5184	0.0998	-0.0049	-0.0062	-0.0436	0.4593	0.2734	0.2561	0.0507
TA	0.0631	0.1824	0.1518	0.1294	0.1848	0.0819	0.1902	0.3742	0.1875	0.1757	0.2179	0.0614	0.0557	0.0311	-0.0271	0.1373	0.1478	0.1286	0.1054
CH	0.2409	0.4775	0.4724	0.5501	0.3555	0.4881	0.5106	0.1875	0.9483	0.5832	0.5663	0.2546	0.0833	-0.0543	-0.0768	0.6676	0.4983	0.4179	0.1875
GS	0.1502	0.3055	0.3155	0.3967	0.2912	0.3090	0.3763	0.1757	0.5832	1.0996	0.5148	0.2349	0.1363	-0.0450	-0.0325	0.4994	0.4082	0.3500	0.2580
SS	0.3285	0.5137	0.4539	0.5319	0.5436	0.3164	0.5184	0.2179	0.5663	0.5148	0.9553	0.1805	0.1658	-0.0309	-0.0626	0.5865	0.5759	0.5727	0.1192
BS	0.1166	0.1779	0.1889	0.2660	0.1570	0.2612	0.0998	0.0614	0.2546	0.2349	0.1805	0.8578	0.1096	0.0839	0.0916	0.2472	0.2037	0.0969	0.1386
US	0.1639	0.0742	0.0750	0.0577	0.0964	0.0909	-0.0049	0.0557	0.0833	0.1363	0.1658	0.1096	1.0220	0.0614	0.2140	0.1473	0.2617	0.2134	0.2450
TS	0.0592	0.0073	0.0246	0.0087	-0.0090	-0.0559	-0.0062	0.0311	-0.0543	-0.0450	-0.0309	0.0839	0.0614	0.7360	0.3420	-0.0535	-0.0374	-0.0444	-0.0296
AP	0.0482	-0.0635	-0.0471	-0.0495	-0.0641	-0.0763	-0.0436	-0.0271	-0.0768	-0.0325	-0.0626	0.0916	0.2140	0.3420	1.4740	-0.0761	-0.0486	-0.0562	-0.0353
PL	0.2607	0.4620	0.4115	0.5160	0.3969	0.4362	0.4593	0.1373	0.6676	0.4994	0.5865	0.2472	0.1473	-0.0535	-0.0761	0.8722	0.6055	0.5274	0.1536
FS	0.3043	0.3385	0.3921	0.4187	0.3657	0.3097	0.2734	0.1478	0.4983	0.4082	0.5759	0.2037	0.2617	-0.0374	-0.0486	0.6055	0.9628	0.5719	0.1475
EN	0.2905	0.2653	0.3918	0.3387	0.2782	0.0660	0.2561	0.1286	0.4179	0.3500	0.5727	0.0969	0.2134	-0.0444	-0.0562	0.5274	0.5719	1.1376	0.0884
POP	0.0820	0.0698	0.0058	0.0869	0.0503	0.1425	0.0507	0.1054	0.1875	0.2580	0.1192	0.1386	0.2450	-0.0296	-0.0353	0.1536	0.1475	0.0884	0.9332

(A.1)

B. Covariance Matrix for QDA

Covariance matrix of class demand = 0 for QDA

	CRS	FB	SH	DS	SR	MS	AS	TA	CH	GS	SS	BS	US	TS	AP	PL	FS	EN	POP
CRS	0.5269	0.1022	0.1074	0.1434	0.0756	0.1750	0.0279	0.0235	0.1352	0.1095	0.0891	0.0545	0.1823	0.0000	-0.0149	0.1238	0.1372	0.1106	0.0325
FB	0.1022	0.3815	0.2864	0.3132	0.2024	0.3104	0.0934	0.0973	0.2402	0.1586	0.1735	0.1690	0.0743	0.0000	-0.0180	0.2115	0.1412	0.0747	0.0766
SH	0.1074	0.2864	0.4060	0.3553	0.1411	0.3168	0.0829	0.0665	0.2353	0.1667	0.1408	0.1992	0.0746	0.0000	-0.0170	0.1891	0.1584	0.0947	0.0394
DS	0.1434	0.3132	0.3553	0.4446	0.1626	0.3807	0.0905	0.0475	0.2773	0.2240	0.1666	0.2357	0.0978	0.0000	-0.0245	0.2703	0.2086	0.1355	0.0968
SR	0.0756	0.2024	0.1411	0.1626	0.3091	0.1459	0.0837	0.0917	0.1318	0.1063	0.1141	0.1151	0.0606	0.0000	-0.0129	0.1491	0.0958	0.0599	0.0318
MS	0.1750	0.3104	0.3168	0.3807	0.1459	0.7705	0.1273	0.1043	0.3980	0.3011	0.2095	0.2984	0.1989	0.0000	-0.0297	0.3272	0.2162	0.0707	0.2009
AS	0.0279	0.0934	0.0829	0.0905	0.0837	0.1273	0.1268	0.0540	0.1613	0.0971	0.0640	0.0328	0.0444	0.0000	-0.0087	0.1189	0.0726	0.0612	0.0250
TA	0.0235	0.0973	0.0665	0.0475	0.0917	0.1043	0.0540	0.3048	0.0851	0.0965	0.0923	0.0529	0.0255	0.0000	-0.0115	0.0704	0.1081	0.0603	0.0400
CH	0.1352	0.2402	0.2353	0.2773	0.1318	0.3980	0.1613	0.0851	0.5823	0.3564	0.2169	0.2282	0.1517	0.0000	-0.0176	0.3384	0.2249	0.1674	0.1744
GS	0.1095	0.1586	0.1667	0.2240	0.1063	0.3011	0.0971	0.0965	0.3564	0.8333	0.2058	0.2000	0.1137	0.0000	-0.0052	0.2577	0.2668	0.1507	0.2842
SS	0.0891	0.1735	0.1408	0.1666	0.1141	0.2095	0.0640	0.0923	0.2169	0.2058	0.2495	0.1331	0.1302	0.0000	-0.0023	0.2011	0.1802	0.1534	0.0675
BS	0.0545	0.1690	0.1992	0.2357	0.1151	0.2984	0.0328	0.0529	0.2282	0.2000	0.1331	0.6989	0.0183	0.0000	0.0401	0.2117	0.1587	0.0649	0.0960
US	0.1823	0.0743	0.0746	0.0978	0.0606	0.1989	0.0444	0.0444	0.0255	0.1517	0.1137	0.1302	0.0183	0.7378	0.0000	-0.0126	0.1546	0.1944	0.1424
TS	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
AP	-0.0149	-0.0180	-0.0170	-0.0245	-0.0129	-0.0297	-0.0087	-0.0115	-0.0176	-0.0052	-0.0023	0.0401	-0.0126	0.0000	0.9814	-0.0140	-0.0173	-0.0138	-0.0147
PL	0.1238	0.2115	0.1891	0.2703	0.1491	0.3272	0.1189	0.0704	0.3384	0.2577	0.2011	0.2117	0.1546	0.0000	-0.0140	0.3759	0.2298	0.1764	0.1529
FS	0.1372	0.1412	0.1584	0.2086	0.0958	0.2162	0.0726	0.1081	0.2249	0.2668	0.1802	0.1587	0.1944	0.0000	-0.0173	0.2298	0.4161	0.1689	0.1655
EN	0.1106	0.0747	0.0947	0.1355	0.0599	0.0707	0.0612	0.0603	0.1674	0.1507	0.1534	0.0649	0.1424	0.0000	-0.0138	0.1764	0.1689	0.3247	0.0348
POP	0.0325	0.0766	0.0394	0.0968	0.0318	0.2009	0.0250	0.0400	0.1744	0.2842	0.0675	0.0960	0.1383	0.0000	-0.0147	0.1529	0.1655	0.0348	0.6028

(B.1)

Covariance matrix of class demand = 1 for QDA

	CRS	FB	SH	DS	SR	MS	AS	TA	CH	GS	SS	BS	US	TS	AP	PL	FS	EN	POP
CRS	1.2475	0.2735	0.2432	0.3935	0.3700	0.3083	0.2481	0.1031	0.3474	0.1912	0.5698	0.1793	0.1452	0.1189	0.1118	0.3986	0.4727	0.4719	0.1320
FB	0.2735	1.2683	0.9212	1.0115	0.8624	0.5396	0.9742	0.2682	0.7168	0.4535	0.8567	0.1869	0.0741	0.0147	-0.1093	0.7146	0.5374	0.4574	0.0629
SH	0.2432	0.9212	1.6628	0.9443	0.6683	0.3969	0.6871	0.2378	0.7115	0.4655	0.7696	0.1786	0.0755	0.0494	-0.0774	0.6356	0.6278	0.6912	-0.0281
DS	0.3935	1.0115	0.9443	1.2251	0.7539	0.7246	0.8889	0.2120	0.8251	0.5709	0.9002	0.2965	0.0172	0.0175	-0.0748	0.7637	0.6306	0.5434	0.0770
SR	0.3700	0.8624	0.6683	0.7539	1.1676	0.3370	0.7763	0.2787	0.5811	0.4776	0.9765	0.1993	0.1324	-0.0180	-0.1158	0.6466	0.6377	0.4981	0.0690
MS	0.3083	0.5396	0.3969	0.7246	0.3370	1.0685	0.2852	0.0593	0.5789	0.3169	0.4241	0.2236	-0.0179	-0.1123	-0.1234	0.5460	0.4040	0.0614	0.0837
AS	0.2481	0.9742	0.6871	0.8889	0.7763	0.2852	1.7359	0.3274	0.8627	0.6577	0.9764	0.1674	-0.0545	-0.0124	-0.0787	0.8024	0.4759	0.4525	0.0767
TA	0.1031	0.2682	0.2378	0.2120	0.2787	0.0593	0.3274	0.4443	0.2908	0.2556	0.3444	0.0700	0.0862	0.0625	-0.0429	0.2048	0.1877	0.1974	0.1713
CH	0.3474	0.7168	0.7115	0.8251	0.5811	0.5789	0.8627	0.2908	1.3677	0.8118	0.9185	0.2813	0.0143	-0.1091	-0.1364	0.9993	0.7739	0.6705	0.2008
GS	0.1912	0.4535	0.4655	0.5709	0.4776	0.3169	0.6577	0.2556	0.8118	1.3680	0.8262	0.2701	0.1590	-0.0903	-0.0600	0.7430	0.5506	0.5508	0.2315
SS	0.5698	0.8567	0.7696	0.9002	0.9765	0.4241	0.9764	0.3444	0.9185	0.8262	1.6667	0.2283	0.2015	-0.0620	-0.1234	0.9750	0.9748	0.9953	0.1714
BS	0.1793	0.1869	0.1786	0.2965	0.1993	0.2236	0.1674	0.0700	0.2813	0.2701	0.2283	1.0180	0.2016	0.1684	0.1434	0.2830	0.2490	0.1291	0.1817
US	0.1452	0.0741	0.0755	0.0172	0.1324	-0.0179	-0.0545	0.0862	0.0143	0.1590	0.2015	0.2016	1.3084	0.1233	0.4424	0.1400	0.3295	0.2850	0.3526
TS	0.1189	0.0147	0.0494	0.0175	-0.0180	-0.1123	-0.0124	0.0625	-0.1091	-0.0903	-0.0620	0.1684	0.1233	1.4780	0.6868	-0.1075	-0.0751	-0.0892	-0.0595
AP	0.1118	-0.1093	-0.0774	-0.07															

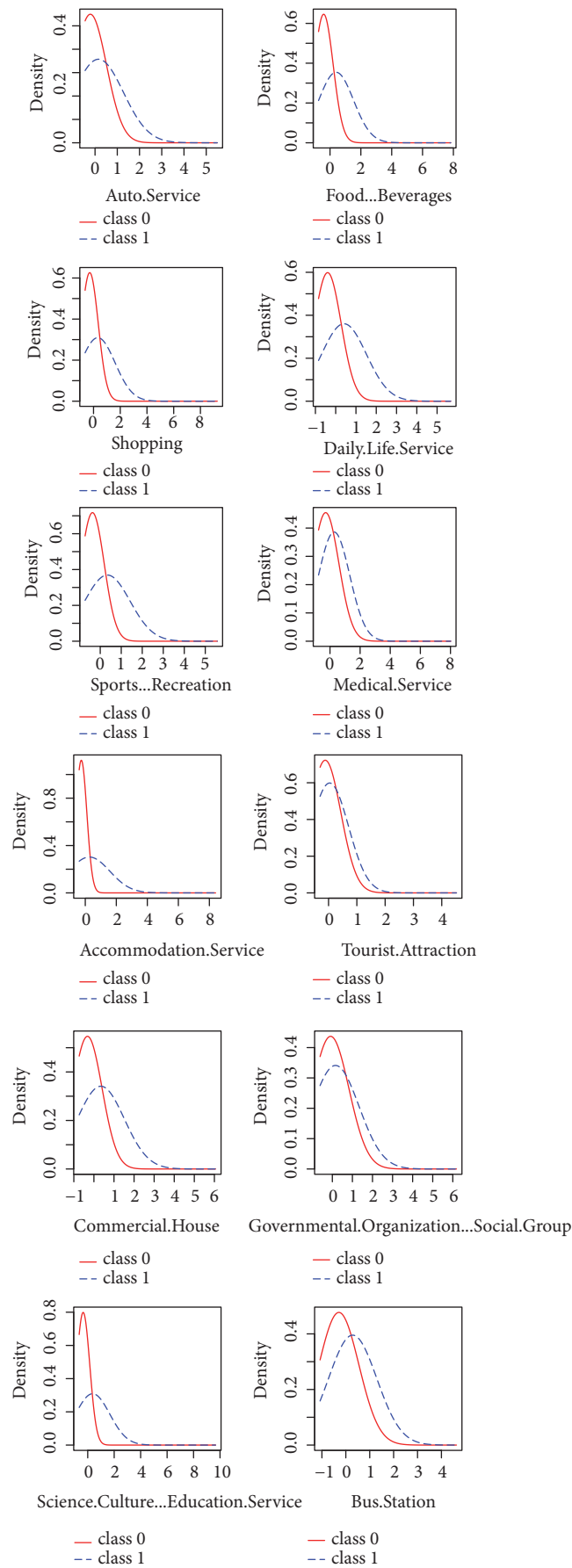


FIGURE 7: Continued.

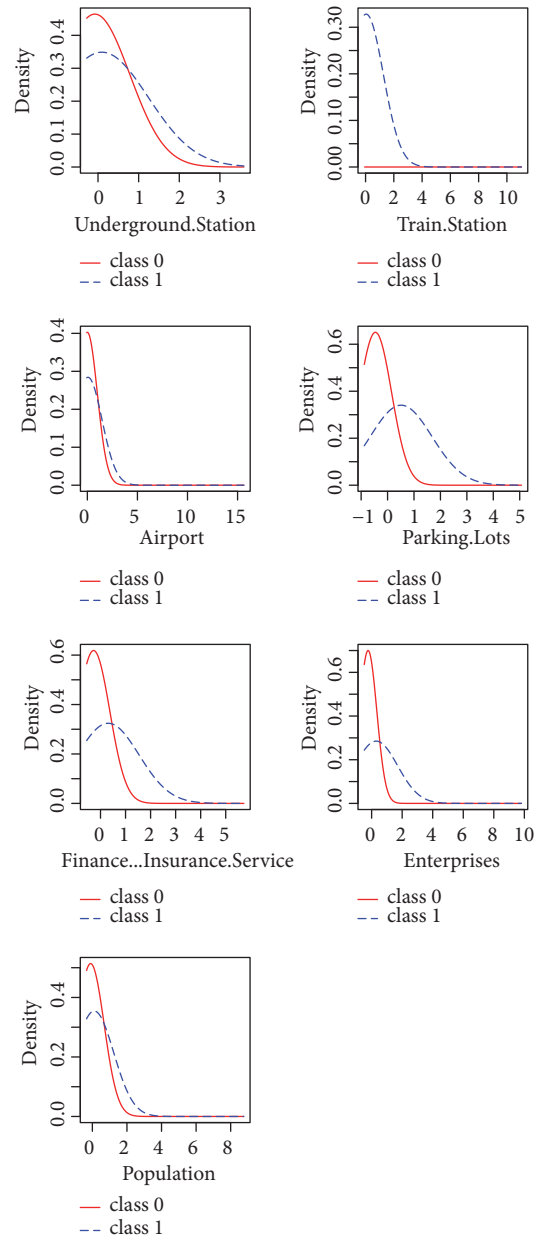


FIGURE 7: Distributions of predictors for Naive Bayes.

C. Distributions of Predictors for Naive Bayes

See Figure 7.

Data Availability

All data included in this study are available upon request by contact with the author Yu Cheng (chengyu@shevdc.org). They will all be based on the grids defined in the research since order data for a specific station is quite sensitive and private.

Disclosure

This research was presented at the conference “International Conference on Smart Mobility and Logistics in Future Cities”

and presentation slides which include brief introduction of this research are shared to that conference. This manuscript is also published in the main website of our organization for internal study and communication.

Conflicts of Interest

The authors declared that they have no conflicts of interest to this work.

Acknowledgments

This work is supported by International Science & Technology Cooperation Program of China under Contract no. 2016YFE0102200.

References

- [1] Millard-Ball, G. Murray, J. T. Schure et al., "Car-sharing: Where and how it succeeds," Tech. Rep., Transportation Research Board of the National Academies, 2005.
- [2] S. Shaheen, Innovative mobility carsharing outlook: Carsharing market overview, analysis, and trends - summer 2014, vol. 3, no. 1. 2014.
- [3] X. Chen, J. Cheng, J. Ye, Y. Jin, X. Li, and F. Zhang, "Locating Station of One-Way Carsharing Based on Spatial Demand Characteristics," *Journal of Advanced Transportation*, vol. 2018, Article ID 5493632, 16 pages, 2018.
- [4] M. G. Y. Klinevicius, C. Morency, and M. Trépanier, "Assessing impact of carsharing on household car ownership in Montreal, Quebec, Canada," *Transportation Research Record*, vol. 2416, pp. 48–55, 2014.
- [5] E. Martin, S. Shaheen, and J. Lidicker, "Impact of carsharing on household vehicle holdings: Results from North American shared-use vehicle survey," *Transportation Research Record Journal of the Transportation Research Board*, vol. 2143, pp. 150–158, 2010.
- [6] J. T. Schure, F. Napolitan, and R. Hutchinson, "Cumulative impacts of carsharing and unbundled parking on vehicle ownership and mode choice," *Transportation Research Record*, no. 2319, pp. 96–104, 2012.
- [7] S. Schmöller, S. Weikl, J. Müller, and K. Bogenberger, "Empirical analysis of free-floating carsharing usage: The munich and berlin case," *Transportation Research Part C: Emerging Technologies*, vol. 56, pp. 34–51, 2015.
- [8] D. Febbraro, N. Sacco, and M. Saeednia, *One-Way Carsharing: Solving the Relocation Problem*, 2012.
- [9] S. Weikl and K. Bogenberger, "Relocation strategies and algorithms for free-floating car sharing systems," *Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 100–111, 2013.
- [10] F. Ciari, C. Weis, and M. Balac, "Evaluating the influence of carsharing stations location on potential membership: a swiss case study," *Euro Journal on Transportation Logistics*, vol. 50, no. 3, pp. 345–369, 2016.
- [11] W. Li, Y. Li, J. Fan, and H. Deng, "Siting of carsharing stations based on spatial multi-criteria evaluation: A case study of Shanghai EVCARD," *Sustainability*, vol. 9, no. 1, p. 152, 2017.
- [12] S. D. Pohekar and M. Ramachandran, "Application of multi-criteria decision making to sustainable energy planning—a review," *Renewable & Sustainable Energy Reviews*, vol. 8, no. 4, pp. 365–381, 2004.
- [13] G. H. D. A. Correia and A. P. Antunes, "Optimization approach to depot location and trip selection in one-way carsharing systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 1, pp. 233–247, 2012.
- [14] D. Jorge, G. Correia, and C. Barnhart, "Testing the Validity of the MIP Approach for Locating Carsharing Stations in One-way Systems," *Procedia - Social and Behavioral Sciences*, vol. 54, pp. 138–148, 2012.
- [15] V. P. Kumar and M. Bierlaire, "Optimizing locations for a vehicle sharing system," in *Proceedings of the Swiss Transport Research Conference*, 2012.
- [16] SEDAC. <http://sedac.ciesin.columbia.edu/data/collection/gpw-v4>.
- [17] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, vol. 103, Springer, New York, NY, USA, 2013.
- [18] T. Hastie, R. Tibshirani, J. H. Friedman, and J. Franklin, "The elements of statistical learning, second edition: Data mining, inference, and prediction," *Mathematical Intelligencer*, vol. 270, no. 2, pp. 125, 210–211, 2009.
- [19] S. Wagner, T. Brandt, and D. Neumann, "In free float: Developing Business Analytics support for carsharing providers," *OMEGA - The International Journal of Management Science*, vol. 59, pp. 4–14, 2016.
- [20] E. Martin, S. A. Shaheen, and J. Lidicker, "Carsharings impact on household vehicle holdings: Results from a north american shared-use vehicle survey," *Institute of Transportation Studies Working Paper*, vol. 460, no. 2143, pp. 150–158, 2010.

Research Article

A Hybrid Method for Traffic Incident Duration Prediction Using BOA-Optimized Random Forest Combined with Neighborhood Components Analysis

Qiang Shang ¹, Derong Tan,¹ Song Gao,¹ and Linlin Feng²

¹School of Transportation and Vehicle Engineering, Shandong University of Technology, Zibo, Shandong 255049, China

²School of Marxism Studies, Shandong University of Technology, Zibo, Shandong 255049, China

Correspondence should be addressed to Qiang Shang; shangqiangv587@163.com

Received 8 September 2018; Revised 18 December 2018; Accepted 1 January 2019; Published 20 January 2019

Guest Editor: Hamzeh Khazaei

Copyright © 2019 Qiang Shang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Predicting traffic incident duration is important for effective and real-time traffic incident management (TIM), which helps to minimize traffic congestion, environmental pollution, and secondary incident related to this incident. Traffic incident duration prediction methods often use more input variables to obtain better prediction results. However, the problems that available variables are limited at the beginning of an incident and how to select significant variables are ignored to some extent. In this paper, a novel prediction method named NCA-BOA-RF is proposed using the Neighborhood Components Analysis (NCA) and the Bayesian Optimization Algorithm (BOA)-optimized Random Forest (RF) model. Firstly, the NCA is applied to select feature variables for traffic incident duration. Then, RF model is trained based on the training set constructed using feature variables, and the BOA is employed to optimize the RF parameters. Finally, confusion matrix is introduced to measure the optimized RF model performance and compare with other methods. In addition, the performance is also tested in the absence of some feature variables. The results demonstrate that the proposed method not only has high accuracy, but also exhibits excellent reliability and robustness.

1. Introduction

Traffic incidents such as vehicle crashes, fire, road maintenance, debris, police activities, etc. are still very common, random, and dangerous. The occurrence of traffic incidents can reduce road capacity because of lane closures that result in traffic congestion and delays [1]. The National Traffic Accident Management Association estimates that 25% of the congestion on American roads is caused by traffic incidents [2]. Traffic incidents are the main causes of nonrecurrent congestion on urban expressways and urban arterial roads [3]. In addition, traffic congestion and travel delay can increase the occurrence likelihood of secondary incident [4, 5]. For more than two decades, many cities around the world have established traffic management centers and deployed various traffic incident management systems to decrease traffic incidents and alleviate related congestion. Traffic flow management and providing travelers with timely and accurate information during traffic incident clearance periods are

two main aspects of efficient traffic incident response [6]. The response strategy during the clearance period depends to a large extent on the duration of an incident. Therefore, accurate prediction of traffic incident duration has attracted the attention of researchers because of its importance.

For decades, researchers have put forward many effective methods for predicting traffic incident duration. The data, variables, and algorithms used in these methods are usually different. From the algorithm point of view, early incident duration prediction methods include probability distribution model [7], the regression prediction method [8], and time series methods [9]. They have an advantage of being easily understood and well-established methodologies, with a long history of application, availability of software, and deep-rooted acceptance. However, these methods often depend on certain assumptions, which limit the generalization of them.

In recent years, the application of hazard/risk-based methods and machine learning methods for traffic incident duration prediction has become increasingly widespread. The

hazard/risk-based method uses hazard function to predict traffic incident duration. The risk function is essentially a conditional probability function, which can be used to analyze the probability that a traffic incident has lasted t minute and ended in the k th minute. In 2013, parametric accelerated failure time (AFT) survival models of traffic incident duration were developed by Hojati et al. [10], including log-logistic, lognormal, and Weibull—considering both fixed and random parameters, in order to better apply to different types of traffic incidents. In 2015, Li et al. [3] developed a competing risks mixture model to investigate the influence of clearance methods and various factors on traffic incident duration and predict traffic incident duration. Three candidate distributions including generalized gamma, Weibull, and log-logistic are tested to determine the most appropriate probability density function of the parametric survival analysis model. Subsequently, Li et al. [11] proposed a sequential prediction method for traffic incident duration based on competing risk mixture model and text analysis. Text analysis was used to process the textual features of the traffic incident to extract time-dependent topics. The empirical results demonstrate that the developed mixture model outperforms the non-mixture model. In 2017, accelerated failure time (AFT) hazard-based models were developed with different underlying probability distributions of the hazard function to predict traffic incident duration [12]. This study indicates that the hazard function—gamma distribution model with a time variable is the best model for the four different duration stages (including preparation, travel, and clearance as well as the total duration of the incident), and different parameters and variables were appropriate for modeling the different duration stages of traffic incidents.

As a typical machine learning method, decision trees have drawn extensive attention due to its excellent performance. Many methods have been proposed for traffic incident duration prediction based on decision trees, such as Bayesian decision trees [13], classification tree [14], M5P tree algorithm [15], and Gradient Boosting decision trees [16].

Besides decision tree-based methods, other machine learning methods for traffic incident duration have also made a number of achievements. In 2014, a model for traffic incident duration prediction was developed using an adaptive Bayesian network, which is more adaptable to the future environment than the traditional Bayesian model [17]. In 2015, Wang et al. [18] proposed a prediction method based on a nonparametric regression model whose core algorithm is K Nearest Neighbor (KNN). In the process of modeling, the distribution of incidents duration is taken into account. It is pointed out that the logarithmic transformation of duration will achieve better prediction results. In 2016, Yu et al. [19] compared the performance of artificial neural network (ANN) and support vector machine (SVM) for predicting traffic incident duration. The results show that both ANN and SVM are able to predict the incident duration within an acceptable range. When predicting the longer duration, the ANN model has better performance. On the whole, the overall performance of the SVM model is better than that of the traditional ANN model. In the same year, Park et al. [20] proposed a continuous time prediction method based

on Bayesian neural network and quantified the importance of continuous time input variables by connecting weights to improve the interpretability of the algorithm. In 2017, two nonparametric machine learning methods, including the k -nearest neighbor method and artificial neural network method, were used to develop incident duration prediction models [21]. Based on the performance comparison results, an artificial neural network model can provide good and reasonable prediction for traffic incident duration prediction with mean absolute percentage error values less than 30%, which are better than the prediction results of a k -nearest neighbor model. For more information, a review for traffic incident duration prediction can be referred to [22], mainly including the different phases of incident duration, data resources, and the various methods that are applied in the traffic incident duration influence factor analysis and duration time prediction.

In summary, both hazard/risk-based methods and machine learning methods have advantages and disadvantages. The forms of results obtained by hazard/risk-based methods often better meet the needs of traffic managers, but it is necessary to assume that traffic incident duration obeys one or more probability distributions. The machine learning methods are not limited by the hypothesis conditions, which can directly extract rules from the data set for traffic incident duration prediction with better applicability. However, the interpretability of such methods is usually poor. With the expansion of data scale and the improvement of computer performance, machine learning methods have a broader prospect for traffic incident prediction. At present, most methods use as many relevant variables as possible in order to achieve better prediction results. However, an important fact is ignored that is lack of relevant information at the beginning of the traffic incident. Therefore, it is very necessary to select relevant variables and explore more robust methods for traffic incident duration prediction using incomplete variables.

To tackle the shortcoming as mentioned above, a novel method named NCA-BOA-RF is proposed for traffic incident duration prediction. As a well-known machine learning algorithm, Random Forest (RF) is chosen as the basic algorithm because of its excellent performance of regression and classification. Firstly, based on the analysis of the influencing factors of traffic incident duration and considering the characteristics of the data sets used, 18 influencing factors were selected as the relevant variables of traffic incident duration. Then, feature weights of the relevant variables are calculated by Neighborhood Components Analysis (NCA), and feature variables of traffic incident duration are determined by the feature weights. Finally, the training set is constructed using the feature variables for training RF, and the Bayesian Optimization Algorithm (BOA) is used to optimize the parameters of RF.

The main contributions of this paper are highlighted in the following aspects. (a) NCA is used to calculate feature weights of relevant variables to determine the feature variables. (b) The cross-validation method is used to optimize the regularization parameter of the NCA to ensure its best performance. (c) BOA is used to optimize both parameters of RF at the same time, rather than a single parameter.

(d) Consider that some feature variables are unavailable when testing performance of the proposed method.

The remainder of this paper is organized as follows. Section 2 elaborates the methodology of the proposed model. Section 3 presents the experimental result and discussion. Finally, the conclusions and future study are summarized in Section 4.

2. Methodology

In this section, the basic methods relevant to the proposed model named NCA-BOA-RF are briefly introduced, including Neighborhood Components Analysis (NCA), Bayesian Optimization Algorithm (BOA), and Random Forest (RF). Then, the main steps of NCA-BOA-RF model are given. In the NCA-BOA-RF model, RF is the basic method for traffic incident duration prediction, feature variables were extracted from relevant variables of traffic incident duration, and BOA are utilized to optimize two parameters of RF at the same time.

2.1. Neighborhood Component Analysis. Feature selection is considerably important in data mining and machine learning, especially for high dimensional data. Proper feature selection not only reduces the dimensions of features, but also improves algorithms generalization performance and execution speed [23, 24]. For traffic incident duration prediction, there are many factors that are considered to be related to the traffic incident duration. Usually, these factors are transformed into relevant variables as the prediction model input, such as incident type, number of casualties, number of closed lanes, and weather conditions. However, the more variables are used as input to the model, the better prediction results are not necessarily obtained. Moreover, in the initial stage of incident occurrence, the available data are often very limited (one or more variables may be missing), which reduces the performance of the prediction model. Therefore, it is very important and necessary to select feature variables for traffic incident duration prediction. In this study, Neighborhood Component Analysis (NCA), as a feature selection method, is used to select feature variables of traffic incident duration. NCA is an algorithm that learns a Mahalanobis distance metric in the supervised k-nearest neighbor (KNN) algorithm by minimizing the leave-one-out (LOO) classification error on the training data [25]. In 2012, NCA-based feature selection is proposed, which learns a feature weighting vector by maximizing the expected LOO classification accuracy with a regularization term [26]. Principal Component Analysis (PCA) and Sequential Feature Selection (SFS) are classic and popular methods for feature selection. However, PCA may result in loss of information when mapped to lower dimensions, and SFS may not be able to remove features that become useless after adding other features [27]. In contrast, NCA is not subject to data conditions (e.g., dimension and distribution) and has the advantage that no information will be lost during the dimension reduction process.

Let $T = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_i, y_i), \dots, (\mathbf{x}_N, y_N)\}$ be a labelled training data set, where $\mathbf{x}_i \in \mathbf{R}^d$ are feature vector,

$y_i \in \{1, \dots, C\}$ are the corresponding labels, and N is the number of observations.

A Mahalanobis distance between two observations \mathbf{x}_i and \mathbf{x}_j that are denoted in terms of weighing vector \mathbf{w} is defined as follows [26]:

$$D\mathbf{w}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^d w_l^2 |x_{il} - x_{jl}| \quad (1)$$

where w_l is the l th feature weight. LOO is considered to maximize classification accuracy on training data set T . Randomly select a reference point from T to be labelled accordingly. Given a data point \mathbf{x}_i , the probability of drawing \mathbf{x}_j as a reference point of \mathbf{x}_i is defined by [26]

$$p_{ij} = \begin{cases} \frac{k(D\mathbf{w}(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{k \neq i} k(D\mathbf{w}(\mathbf{x}_i, \mathbf{x}_k))}, & i \neq j \\ 0, & i = j \end{cases} \quad (2)$$

where $k(z) = \exp(-z/\sigma)$ is a kernel function and σ is kernel width that influences the probability of each point being selected as the reference point. The average probability of LOO correct classification is the probability p_i that the randomized classifier correctly classifies observation i which can be expressed as

$$p_i = \sum_j y_{ij} p_{ij} \quad (3)$$

where

$$y_{ij} = \begin{cases} 1, & \text{if } y_i = y_j \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Then, the approximate LOO classification accuracy can be calculated as follows [26].

$$\xi(\mathbf{w}) = \sum_i p_i = \sum_i \sum_j y_{ij} p_{ij} \quad (5)$$

The goal of NCA is to maximize objective function $\xi(\mathbf{w})$ associated with \mathbf{w} using regularized term λ .

$$\xi(\mathbf{w}) = \sum_i \sum_j y_{ij} p_{ij} - \lambda \sum_{l=1}^d w_l^2 \quad (6)$$

where $\lambda > 0$ is a regularized term that can be optimized using cross-validation, which balances the first term of maximization of NCA probability and the second term of minimization of the Frobenius norm.

Because objective function $\xi(\mathbf{w})$ is differentiable, its derivative with respect to \mathbf{w} can be expressed as

$$\frac{\partial \xi(\mathbf{w})}{\partial w_l} = \sum_i \sum_j y_{ij} \left[\frac{2}{\sigma} p_{ij} \left(\sum_{k \neq i} p_{ik} |x_{il} - x_{kl}| \right) - |x_{il} - x_{kl}| w_l - 2\lambda w_l \right] = \frac{2}{\sigma}$$

$$\begin{aligned}
& \cdot \sum_i \left(p_{ij} \sum_{k \neq i} p_{ik} |x_{il} - x_{kl}| - \sum_j y_{ij} p_{ij} |x_{il} - x_{jl}| \right) w_l \\
& - 2\lambda w_l = 2 \left(\frac{1}{\sigma} \right. \\
& \cdot \sum_i \left(p_{ij} \sum_{k \neq i} p_{ik} |x_{il} - x_{kl}| - \sum_j y_{ij} p_{ij} |x_{il} - x_{jl}| \right) \\
& \left. - \lambda \right) w_l
\end{aligned} \tag{7}$$

According to the above, the corresponding gradient ascent update equation can be obtained. The problem of maximizing the objective function can be solved via the gradient descent method. More details about NCA for feature selection can be found in [26].

2.2. Bayesian Optimization Algorithm. Bayesian Optimization Algorithm (BOA) is one of the most well-known distribution algorithm estimates that combine Bayesian networks with evolutionary algorithms. In the BOA, global statistical information is extracted from optimal solutions searched currently and modeled using Bayesian networks. Therefore, BOA can overcome the disruption of building blocks in genetic algorithms. The BOA has advantages in the optimization of machine learning algorithm hyperparameters, because of its faster search speed and fewer iteration compared to traditional search algorithms [28–30]. In this study, the BOA is employed to optimize the parameters of Random Forest (which is the basic model, see Section 2.3 for details) for traffic incident duration prediction, in order to achieve better prediction results.

The algorithm parameters to be optimized are denoted as $\lambda = \{\lambda_1 \in \Lambda_1, \lambda_2 \in \Lambda_2, \dots, \lambda_m \in \Lambda_m\}$. T_{train} is training set and T_{valid} is validation set, and $L(\lambda, T_{train}, T_{valid})$ is the validation accuracy. The goal of optimization is to find a set of parameter values that can maximize L .

Firstly, initialization parameters of the machine learning algorithm are $\lambda_1, \dots, \lambda_n$. Secondly, evaluate the accuracy of the machine learning algorithm with initial parameters using the validation set and record the accuracy. Thirdly, the Gaussian Process (GP) model V is introduced to fit the recorded accuracy iteratively. Then, update the machine learning algorithm parameters according to the recommendations of the GP model. In this process, select the next operating point by the maximum of the acquisition function. The acquisition function guides the optimization by determining the next point to evaluate. Several acquisition functions have been proposed, such as probability of improvement, expected improvement, and information gain. Here, expected improvement is used as the acquisition function [30], and the best validation accuracy so far is L^* .

$$\alpha(\lambda, V) = \int_{-\infty}^{\infty} \max(L - L^*, 0) p_V(L | \lambda) d_L \tag{8}$$

where $p_V(L|\lambda)$ is the probability of L with λ , which is encoded by the GP model V . For more details about acquisition functions, see [31]. More details about BOA for hyperparameters of machine learning can be referred to [28].

2.3. Random Forest. Random Forest (RF) [32] is a machine learning algorithm that integrates multiple decision trees by ensemble learning methods. More specifically, RF is an ensemble learning method that can be used for both classification and regression. Ensemble learning is different from traditional statistic reasoning and machine learning that are trying to build an accurate single model. The goal of ensemble learning is to aggregate the results from multiple trained “weak learners” in order to obtain a “strong learner”. In general, the “weak learners” are simple and fast models with a poor performance. In the case of RF, “weak learners” are decision trees. Therefore, RF has good resistance to noise and not easy to fall into overfitting. Moreover, there are not any assumptions in the resulting RF model. As a result, it is expected that the RF model has wider applicability and better robustness compared with traditional statistic reasoning and machine learning techniques [33, 34]. Therefore, RF is used as a basic model for traffic incident duration prediction. The feature variables selected by the NCA are used as input to the model, and traffic incident duration is used as the output of the model. For more information on model training, see Section 3.

The detailed steps of RF are shown as follows:

(1) Bootstrap samples are randomly formed from the original data with replacement, which will be the training set for growing the trees. The number of the created samples is equal to the number of the trees. Around one-third of the original data are left out which are called “out-of-bag” (OOB) data, while the remaining data are called in-bag data. RF performs a cross-validation in parallel with the training step by using the OOB samples to measure the prediction error [35].

(2) Each node of decision tree selects m_{try} ($m_{try} < P$) features from the P features as a subset rather than comparing all the input variables (features) and the best split is calculated only within this subset. It is worth noting that m_{try} may affect the stability of the RF model. The sensitivity of other parameters such as the number of trees (denoted as n_{tree}) in RF, as well as the size of each tree (i.e., the number of splits in each tree called number of leaves per tree, denoted as n_{leaf}), has also been studied [36, 37].

(3) Each tree splits to its maximum size without pruning throughout the growth of the forest [32].

(4) Each decision tree gives a prediction result. For regression problems, the average results of all decision trees are calculated to find the final prediction value; and for classification problems, the majority voting result is taken as the final output prediction value.

2.4. NCA-BOA-RF Method. The flowchart of NCA-BOA-RF method for traffic incident duration prediction is shown in Figure 1. As can be seen from Figure 1, the application of the NCA-BOA-RF method includes two stages, and the main steps of the NCA-BOA-RF method are as follows.

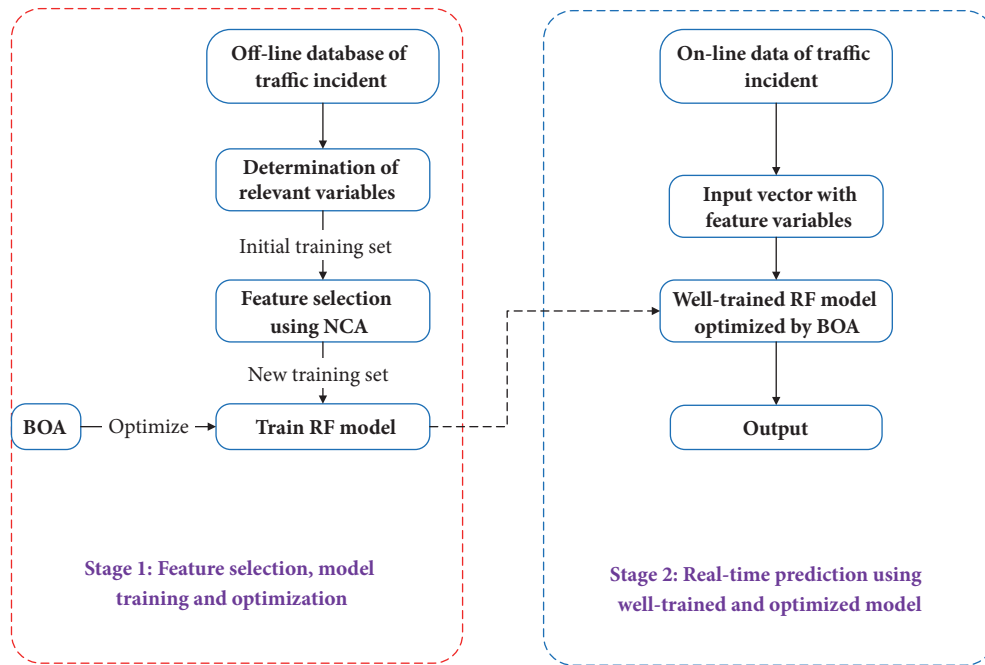


FIGURE 1: The flowchart of NCA-BOA-RF method.

Step 1. Determine relevant variables of traffic incident duration according to the available traffic incident dataset.

Step 2. The NCA method is used to select feature variables from the relevant variables for traffic incident duration prediction.

Step 3. Construct a training set using the feature variables.

Step 4. The RF model is used to learn the training set and the RF parameters are optimized by BOA.

Step 5. Input real-time collected feature variables data into the well-trained RF model and then the model will output prediction result of the traffic incident duration.

It is worth noting that more useful information may be obtained during the duration of a traffic incident. Therefore, if new useful information is collected and the traffic incident has not ended, the NCA-BOA-RF method can be used to predict the traffic incident duration again, but it is necessary to consider the time the incident has passed. In addition, with the increase of traffic incident data, offline database should be continuously updated and used to retrain and optimize the RF model.

3. Empirical Analysis

3.1. Data Description. Data used in this work were obtained from traffic incident dataset on Interstate 880 (well known as I-880 dataset), which were collected in the Freeway Service Patrol (FSP) project. In the two periods of the FSP project, the number of recorded traffic incidents was 1210 and 971, respectively. Some incidents whose start or end time was not

within the observation period cannot be used for modeling and testing due to their unknown duration. In addition, some traffic incidents were planned and predictable, such as road maintenance and traffic restrictions, which were excluded in this study.

The I-880 dataset not only records the time, type, location, and duration of the incident, but also the relative location and distance between the incident location and the road exit, the number of vehicles involved, the type and color of involved vehicles, the location and number of lanes affected, the weather during the incident, and casualties. In addition, whether or not rescue vehicles such as trailers, ambulances, and fire engines are required, as well as the arrival and departure times of rescue vehicles are also recorded. In summary, the basic information of all available traffic incidents is given in Table 1.

A total of 440 traffic incidents (235 in the before period and 205 in the after period) were used in this study. 308 traffic incidents (70%) data were randomly selected as the training set, and the remaining 132 traffic incidents data were used as test set. According to the actual situation of I-880 dataset, the relevant variables of traffic incident duration selected in this study are given in Table 2. The first 18 variables are the input, and the 19th variable is output.

3.2. Feature Selection Using NCA. The NCA algorithm is used to select feature variables for traffic incident duration. Regularization parameter λ is an important parameter of NCA. First of all, it is necessary to determine whether the value of λ is reasonable. In general, the value of λ is $1/n$, where n is the number of input variables of training set ($n = 18$ in this study). In the process of selecting feature variables using NCA, it is necessary to add a set of irrelevant variables as a

TABLE 1: The basic information of all available traffic incidents.

Incident type/Location	Before period		After period	
	Number	Mean (min)	Number	mean (min)
Accidents	45	27.3	38	28.9
Breakdowns	184	25.3	166	23.6
Debris/Pedestrian	6	10.8	1	32.0
Lane	21	24.7	16	30.6
Right Shoulder	195	25.0	169	23.8
Central Divide	19	28.4	20	26.9
All	235	25.6	205	24.7

TABLE 2: The relevant variables of traffic incident duration selected in this study.

No.	Variables	Descriptions
1	Incident type	1: Accidents, 2: Breakdowns, 3: Debris/Pedestrian
2	Incident time	1: A.M., 2: P.M.
3	Direction	1: Northbound, 2: Southbound
4	Incident location	1: Lane, 2: Right Shoulder, 3: Central Divide
5	Number of lanes closed	1: No; 2: One, 3: More than one
6	Distance from the exit	1: < 0.5 miles, 2: 0.5~1 miles, 3: >1 miles
7	Relative position to exit	1: At the exit, 2: Front of the exit, 3: Behind the exit
8	Detection type	1: Call report, 2: Operator detected
9	Need Police	1: Yes, 2: No
10	Need Ambulance	1: Yes, 2: No
11	Need Truck Wrecker	1: Yes, 2: No
12	Need firefighters	1: Yes, 2: No
13	Weather information	1: Sunny, 2: Cloudy, 3: Light Rain, 4: Heavy Rain
14	Automobile Count	1: None; 2: One, 3: Two; 4: More than two
15	Motorcycle Count	1: None; 2: One, 3: More than one
16	Heavy Truck Count	1: None; 2: One, 3: More than one
17	Light Truck Count	1: None; 2: One, 3: More than one
18	Tractor Trailer Count	1: None; 2: One, 3: More than one
19	Duration	1: <10min, 2: 10~30min, 3: 30~60min, 4: 60~90min; 5: >90min

control to highlight the importance of the feature variables. In this study, 100 irrelevant variables are randomly generated from a Normal distribution with a mean of 0 and a variance of 20. In the process of selecting feature variables using NCA, it is necessary to add a set of irrelevant variables as a control to highlight the importance of the feature variables. In this study, 100 irrelevant variables are randomly generated from a Normal distribution with a mean of 0 and a variance of 20. The generated 100 irrelevant variables are added to 18 relevant variables (which are considered as the variables relevant to traffic incident duration). The feature weights of all variables are calculated by NCA. If the value of λ is appropriate, the feature weight of the relevant variables is large, and the feature weight of the irrelevant variables is small and close to 0. If the value of λ is too large, the feature weights of all variables are close to 0; if the value of λ is too small, the irrelevant variables also have a large feature weight.

According to the recommendation in [28, 37], the cross-validation method is used to optimize the regularization parameter λ . In this study, 5-fold cross-validation is employed. That is, the training set is randomly divided

into 5 subsets, 4 subsets reconstruct a training set, and the remaining 1 subset is used as the test set, then the process is repeated five times until each subset is used as a test set, and the average value of 5 test results was adopted as the final result. The best λ value produces the minimum classification loss. Figure 2 shows the average loss values versus λ values, and the best $\lambda = 0.0021$ was obtained that corresponds to the minimum average loss of 0.1250.

The feature weights of all variables are calculated using NCA with the best λ value, and the results are shown in Figure 3. As can be seen from Figure 3(a), the feature weights of the irrelevant variables are all close to 0. As can be seen from Figure 3(b), the serial numbers of the variables with a large feature weight are no. 1, no. 5, no. 11, no. 12, no. 14, and no. 16, respectively. And their corresponding variable names are “incident type”, “number of lanes closed”, “need truck wrecker”, “need firefighters”, “automobile count”, and “heavy truck count”.

Under normal circumstances, traffic incidents involve more vehicles, larger vehicles, and more departments involved in rescue. The duration of the incidents tends to

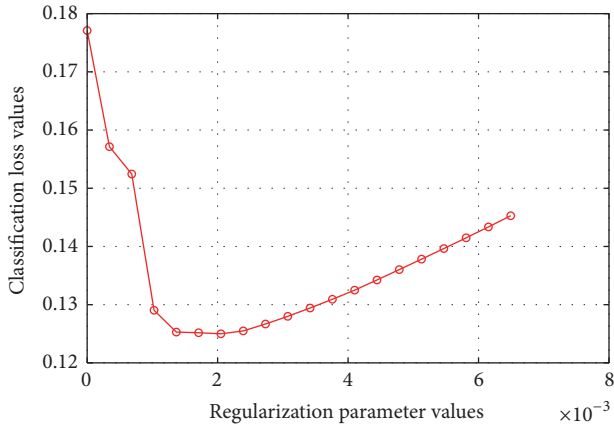


FIGURE 2: The classification loss values versus regularization parameter values.

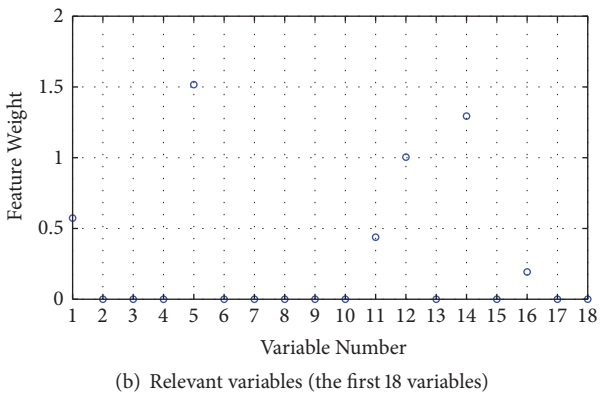
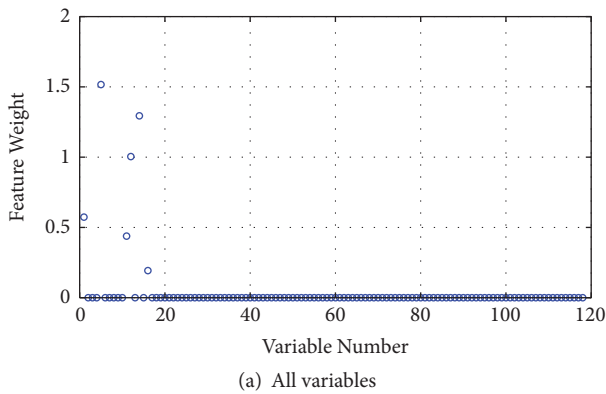


FIGURE 3: The feature weights of variables (after parameter optimization).

be longer, and the calculation results are consistent with the actual situation. Generally, the more vehicles involved, the larger the type of vehicles, and the more the departments involved in rescue, the longer the duration of the incident, and the calculation results are in agreement with the actual situation.

3.3. *RF Parameters Optimized Using BOA.* Training set is constructed using 6 feature variables selected by NCA. Before RF is trained, the RF parameters need to be determined,

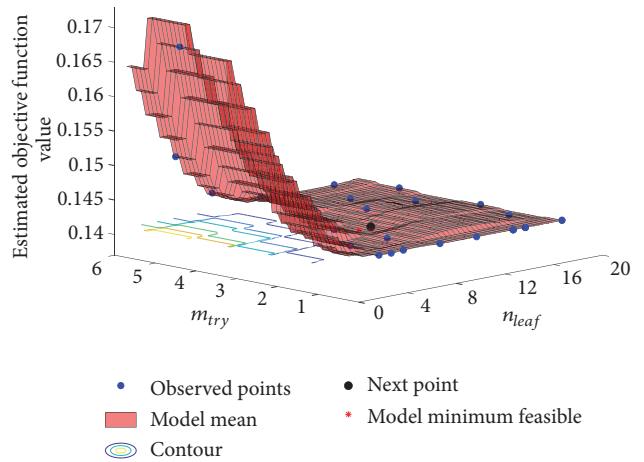


FIGURE 4: The objective function model.

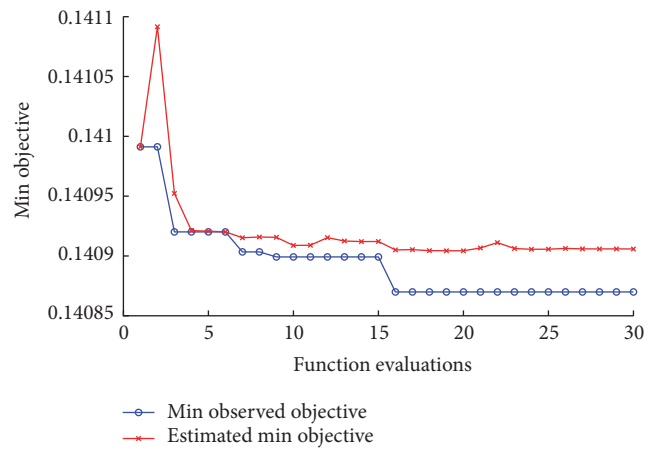


FIGURE 5: The relationship between function evaluations and the minimum objective.

including the number of trees n_{tree} , the number of leaves per tree n_{leaf} , and the number of random variables used for each node split m_{try} . Increasing the number of decision trees can improve the classification accuracy to a certain extent, but will reduce the efficiency of the algorithm. If the minimum classification loss is taken as the goal, the number of decision trees will increase dramatically. Therefore, the number of trees is not optimized in this study, but the other two parameters are optimized for improving the classification accuracy. If n_{leaf} value is too large, it will easily lead to overfitting; if n_{leaf} value is too small, it will easily lead to underfitting. The RF parameters n_{leaf} and m_{try} are tuned using BOA. The RF parameters are set as follows: $n_{tree} = 300$, $n_{leaf} \in [1, 20]$, and $m_{try} \in [1, 5]$. The objective function of BOA is the classification loss of OOB data. Figure 4 shows the objective function model. Figure 5 shows the relationship between function evaluations and the minimum objective. The optimized RF parameters are calculated as $n_{leaf} = 6$ and $m_{try} = 2$, and the observed minimum of the objective function is 0.1409.

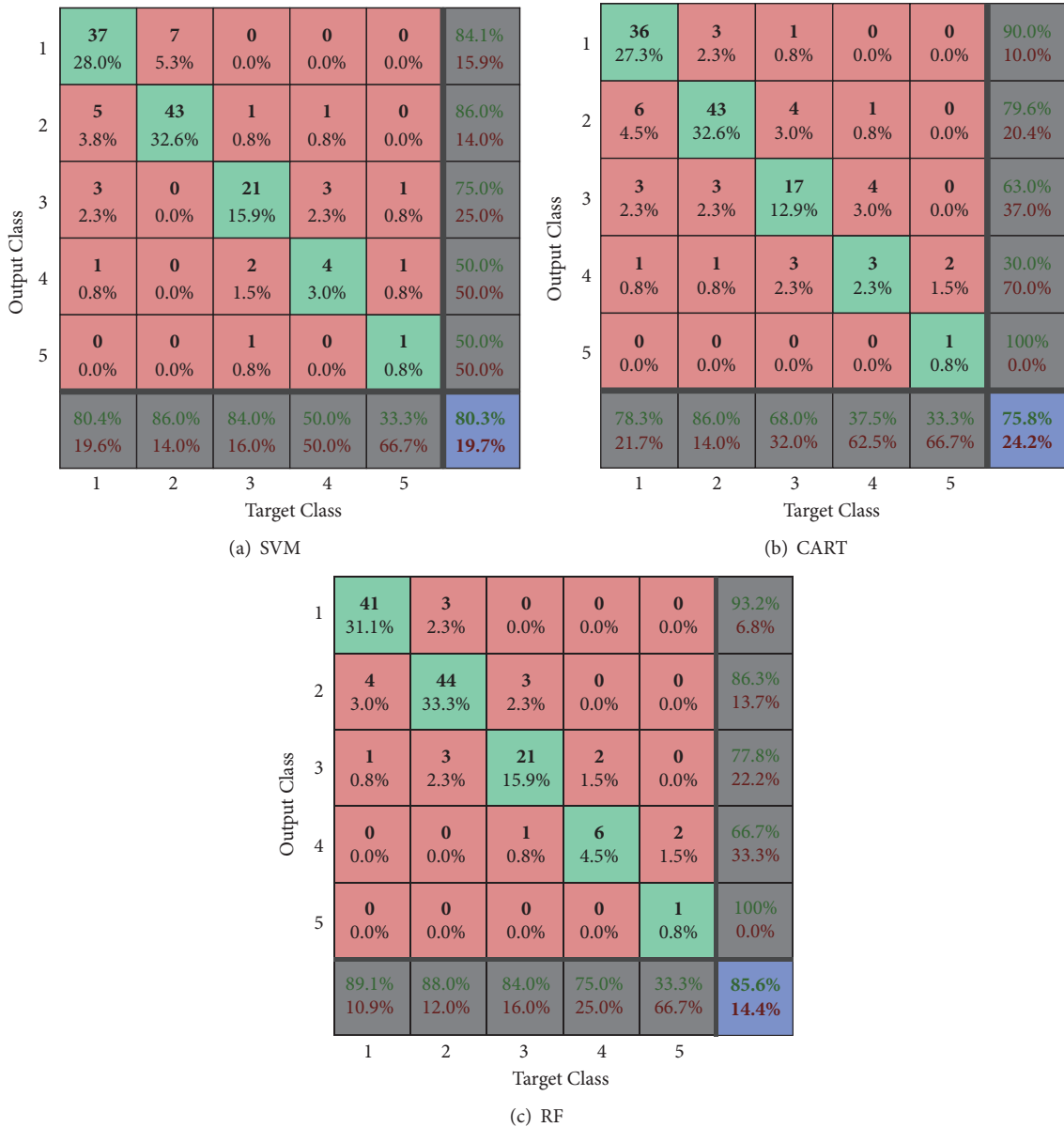


FIGURE 6: The confusion matrices for the results of these three methods.

3.4. Results and Discussion. In this section, not only is the test set used to analyze NCA-BOA-RF performance, but also classification and regression tree (CART) [15] and support vector machine (SVM) [20] are introduced for comparison. The parameters of SVM that need to be determined include kernel function parameter and penalty coefficients. The parameters of CART that need to be determined include learning rate and the maximum number of node splits. In order to ensure comparison methods have good performance, BOA is also used to optimize the parameters of SVM and CART.

Figure 6 shows the confusion matrices for the results of these three methods. The confusion matrix can provide detailed classification results of the algorithm [38]. The rows represent the output class (predicted class) and the columns represent the target class (true class). The diagonal cells

(green) represent observations that are classified correctly. The off-diagonal cells (red) represent observations which are classified incorrectly. The column on the far right of the plot shows the percentages of all the test samples predicted to belong to each class that are correctly and incorrectly classified. These metrics are often called the precision (or positive predictive value) and false discovery rate, respectively. The row at the bottom of the plot shows the percentages of all the test samples belonging to each class that are correctly and incorrectly classified. These metrics are often called the recall (or true positive rate) and false negative rate, respectively. The cell (blue) in the bottom right of the plot shows the overall classification accuracy.

Therefore, according to confusion matrices obtained as a result of the three methods, the overall classification accuracy

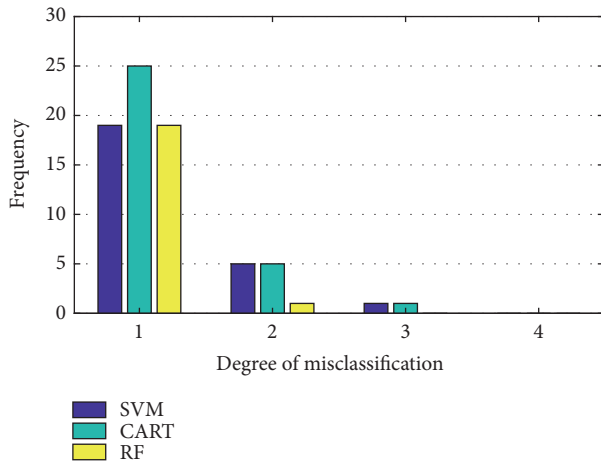


FIGURE 7: The frequency distribution of the misclassification degree of these three methods.

of RF is the highest one. For the “5” class samples, the classification accuracy of the three methods is equal (=33.3%), indicating that the three methods have a poor classification performance on the “5” class samples. This is due to the fact that the number of the “5” class samples is too small. For the “3” class samples, the classification accuracy of RF and SVM is equal (=84.0%). In addition, the classification accuracy of RF is higher than that of SVM and CART on the samples of the remaining classes.

From the confusion matrix, the degree of misclassification can be achieved. For example, if the “1” class is classified into the “2” class, the degree of misclassification is 1; if the “1” class is classified into the “3” class, the degree of misclassification is 2. Figure 7 shows the frequency distribution of the misclassification degree for these three methods. It can be seen from Figure 7 that the misclassification degree of RF is mainly 1, a small amount is 2, and there are no 3 and 4, while the other two methods have a misclassification degree of 3, and the number of misclassification degree of 2 is more than that of RF significantly.

At the beginning of the traffic incident, the available information for incident duration prediction is often limited. Therefore, we need to analyze the performance of the method in the absence of some variables. The six feature variables are 1 “incident type”, 2 “number of lanes closed”, 3 “need truck wrecker”, 4 “need firefighters”, 5 “automobile count”, and 6 “heavy truck count”. When an incident is first identified, 1 “incident type”, 4 “need firefighters”, and 5 “automobile count” are often the first to be known, so the absence of these three variables is not considered.

In the absence of variables, the classification accuracy of the three methods is shown in Figure 8. As can be seen from Figure 8, when a single variable is missing, the classification accuracy of the three methods reduces to a certain extent; when two variables are missing, the classification accuracy of the three methods continues to decrease; when three variables are missing, the classification accuracy of the three methods is greatly reduced to about 50%. However, from the comparison of the three methods without some variables, the

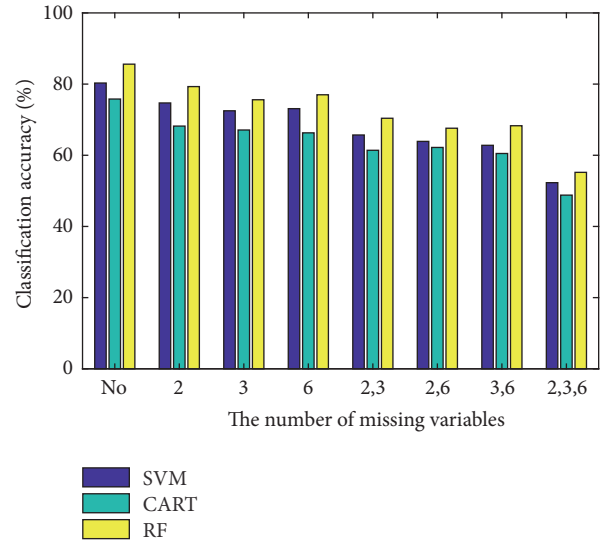


FIGURE 8: The classification accuracy of the three methods in the absence of variables.

classification accuracy of RF is still higher than that of SVM and CART.

4. Conclusions

In this study, a hybrid method named NCA-BOA-RF is proposed to integrate the NCA and the BOA-optimized RF model for traffic incident duration prediction. Firstly, 18 influencing factors are selected as the relevant variables of traffic incident duration, considering influencing factors of traffic incident duration and the data set that we used. Secondly, feature weights of the relevant variables are calculated by Neighborhood Components Analysis (NCA), and feature variables of traffic incident duration are determined by the feature weights. Secondly, the NCA is applied to select the most powerful features (called feature variables) for traffic incident duration prediction. In NCA, regularization parameter is optimized using cross-validation to ensure better classification accuracy (less classification loss). Then, the training set is constructed using the feature variables to train RF model, and the BOA is used to optimize the RF parameters. Finally, we conduct experiments to test performance of the proposed method and two comparison methods. Confusion matrix was introduced to better illustrate the experimental results. Not only has the proposed method the highest classification accuracy on the whole data, but also the classification accuracy of the proposed method is greater than or equal to that of the other two methods on each class of the data. In addition, the performance of the proposed method is tested with some feature variables unavailable. The results demonstrate that the performance is still better than comparison methods, although the performance of all methods is reduced. Based on the comparison and analysis of the experimental results, the conclusions can be drawn that NCA-BOA-RF is a better method for traffic incident

duration prediction, because of its high accuracy rate and good generalization ability.

For future research, more and more comprehensive data sets should be used to further test the method performance for drawing a more general conclusion. From an incident being detected to the incident being cleared up, more useful information is expected to be available. Multistage updates of information should be considered in future study. In addition, predicting different stages of event duration (such as response time) separately is also a direction for future research.

Data Availability

The traffic incident data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported by the National Natural Science Foundation of Shandong Province (Grant Nos. ZR2018BF024 and ZR2016EL19), MOE (Ministry of Education in China) Project of Humanities and Social Sciences (Grant No. 18YJC190003), the National Natural Science Foundation of China (Grant No. 61573009), and the Dr. Scientific Research Start Funding Projects of Shandong University of Technology (Grant Nos. 4041/417006, 4033/718003).

References

- [1] Y. Chung, H. Kim, and M. Park, "Quantifying non-recurrent traffic congestion caused by freeway work zones using archived work zone and ITS traffic data," *Transportmetrica*, vol. 8, no. 4, pp. 307–320, 2012.
- [2] W. Kim and G.-L. Chang, "Development of a hybrid prediction model for freeway incident duration: a case study in Maryland," *International Journal of Intelligent Transportation Systems Research*, vol. 10, no. 1, pp. 22–33, 2012.
- [3] R. Li, F. C. Pereira, and M. E. Ben-Akiva, "Competing risks mixture model for traffic incident duration prediction," *Accident Analysis & Prevention*, vol. 75, pp. 192–201, 2015.
- [4] A. Khattak, X. Wang, and H. Zhang, "Incident management integration tool: Dynamically predicting incident durations, secondary incident occurrence and incident delays," *IET Intelligent Transport Systems*, vol. 6, no. 2, pp. 204–214, 2012.
- [5] H. Zhang and A. Khattak, "What is the role of multiple secondary incidents in traffic operations?" *Journal of Transportation Engineering*, vol. 136, no. 11, pp. 986–997, 2010.
- [6] Y. Lou, Y. Yin, and S. Lawphongpanich, "Freeway service patrol deployment planning for incident management and congestion mitigation," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 2, pp. 283–295, 2011.
- [7] D. Nam and F. Mannering, "An exploratory hazard-based analysis of highway incident duration," *Transportation Research Part A: Policy and Practice*, vol. 34, no. 2, pp. 85–102, 2000.
- [8] W. M. Liu, L. P. Guan, and X. Y. Yin, "Prediction of incident duration based on multiple regression analysis," *Journal of Highway & Transportation Research & Development*, vol. 22, no. 11, pp. 126–129, 2005.
- [9] A. J. Khattak, J. L. Schofer, and M. Wang, "A simple time sequential procedure for predicting freeway incident duration," *Journal of Intelligent Transportation Systems*, vol. 2, no. 2, pp. 113–138, 1995.
- [10] A. Tavassoli Hojati, L. Ferreira, S. Washington, and P. Charles, "Hazard based models for freeway traffic incident duration," *Accident Analysis & Prevention*, vol. 52, pp. 171–181, 2013.
- [11] R. Li, F. C. Pereira, and M. E. Ben-Akiva, "Competing risk mixture model and text analysis for sequential incident duration prediction," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 74–85, 2015.
- [12] R. Li, M. Guo, and H. Lu, "Analysis of the Different Duration Stages of Accidents with Hazard-Based Model," *International Journal of Intelligent Transportation Systems Research*, vol. 15, no. 1, pp. 7–16, 2017.
- [13] B. Jiyang, X. Zhang, and L. Sun, "Traffic incident duration prediction grounded on Bayesian decision method-based tree algorithm," *Tongji Daxue Xuebao/Journal of Tongji University*, vol. 36, no. 3, pp. 319–324, 2008.
- [14] H. L. Chang and T. P. Chang, "Prediction of freeway incident duration based on classification tree analysis," *Journal of the Eastern Asia Society for Transportation Studies*, vol. 10, pp. 1964–1977, 2013.
- [15] C. Zhan, A. Gan, and M. Hadi, "Prediction of lane clearance time of freeway incidents using the M5P tree algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1549–1557, 2011.
- [16] X. Ma, C. Ding, S. Luan, Y. Wang, and Y. Wang, "Prioritizing Influential Factors for Freeway Incident Clearance Time Prediction Using the Gradient Boosting Decision Trees Method," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2303–2310, 2017.
- [17] S. Demirolok and K. Ozbay, "Adaptive learning in bayesian networks for incident duration prediction," *Transportation Research Record*, vol. 2460, no. 1, pp. 77–85, 2014.
- [18] S. Wang, R. Li, and M. Guo, "Application of nonparametric regression in predicting traffic incident duration," *Transport*, vol. 2015, pp. 1–10, 2015.
- [19] B. Yu, Y. T. Wang, J. B. Yao, and J. Y. Wang, "A comparison of the performance of ANN and SVM for the prediction of traffic accident duration," *Neural Network World*, vol. 26, no. 3, pp. 271–287, 2016.
- [20] H. Park, A. Haghani, and X. Zhang, "Interpretation of Bayesian neural networks for predicting the duration of detected incidents," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 20, no. 4, pp. 385–400, 2016.
- [21] Y. Lee, C.-H. Wei, and K.-C. Chao, "Non-parametric machine learning methods for evaluating the effects of traffic accident duration on freeways," *Archives of Transport*, vol. 43, no. 3, pp. 91–104, 2017.
- [22] R. Li, F. C. Pereira, and M. E. Ben-Akiva, "Overview of traffic incident duration analysis and prediction," *European Transport Research Review*, vol. 10, no. 2, pp. 22, 2018.
- [23] Y. Lee and C.-H. Wei, "A computerized feature selection method using genetic algorithms to forecast freeway accident duration times," *Computer-Aided Civil and Infrastructure Engineering*, vol. 25, no. 2, pp. 132–148, 2010.

- [24] Q. Shang, Z. Yang, S. Gao, and D. Tan, "An Imputation Method for Missing Traffic Data Based on FCM Optimized by PSO-SVR," *Journal of Advanced Transportation*, vol. 2018, Article ID 2935248, 21 pages, 2018.
- [25] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in neural information processing systems*, vol. 17, pp. 513–520, 2005.
- [26] W. Yang, K. Wang, and W. Zuo, "Neighborhood component feature selection for high-dimensional data," *Journal of Computers*, vol. 7, no. 1, pp. 162–168, 2012.
- [27] M. Jin and W. Deng, "Predication of different stages of Alzheimer's disease using neighborhood component analysis and ensemble decision tree," *Journal of Neuroscience Methods*, vol. 302, pp. 35–41, 2018.
- [28] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems 2012, NIPS 2012*, pp. 2951–2959, USA, December 2012.
- [29] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast Bayesian hyperparameter optimization on large datasets," *Electronic Journal of Statistics*, vol. 11, no. 2, pp. 4945–4968, 2017.
- [30] L. Wang, M. Feng, B. Zhou, B. Xiang, and S. Mahadevan, "Efficient hyper-parameter optimization for NLP applications," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pp. 2112–2117, Portugal, September 2015.
- [31] E. Brochu, V. M. Cora, and N. De Freitas, *A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning*, 2010, <https://arxiv.org/abs/1012.2599>.
- [32] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [33] Z. Bei, Z. Yu, N. Luo, C. Jiang, C. Xu, and S. Feng, "Configuring in-memory cluster computing using random forest," *Future Generation Computer Systems*, vol. 79, pp. 1–15, 2018.
- [34] F. Ouallouche, M. Lazri, and S. Ameer, "Improvement of rainfall estimation from MSG data using Random Forests classification and regression," *Atmospheric Research*, vol. 211, pp. 62–72, 2018.
- [35] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern Recognition Letters*, vol. 31, no. 14, pp. 2225–2236, 2010.
- [36] C. Strobl, J. Malley, and G. Tutz, "An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests," *Psychological Methods*, vol. 14, no. 4, pp. 323–348, 2009.
- [37] U. Grömping, "Variable importance assessment in regression: linear regression versus random forest," *The American Statistician*, vol. 63, no. 4, pp. 308–319, 2009.
- [38] E. Vigneau, P. Courcoux, R. Symoneaux, L. Guérin, and A. Villière, "Random forests: A machine learning methodology to highlight the volatile organic compounds involved in olfactory perception," *Food Quality and Preference*, vol. 68, pp. 135–145, 2018.

Research Article

A Machine Learning Method for Predicting Driving Range of Battery Electric Vehicles

Shuai Sun ^{1,2}, Jun Zhang,³ Jun Bi ^{1,2} and Yongxing Wang ^{1,2}

¹School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, China

²MOE Key Laboratory for Urban Transportation Complex Systems Theory and Technology, Beijing Jiaotong University, Beijing 100044, China

³Yunnan Travelsky Airport Technology Co. Ltd., Kunming 650200, China

Correspondence should be addressed to Shuai Sun; 17114261@bjtu.edu.cn and Jun Bi; bilinghc@163.com

Received 24 September 2018; Revised 30 November 2018; Accepted 26 December 2018; Published 9 January 2019

Guest Editor: Mohammad H. Y. Moghaddam

Copyright © 2019 Shuai Sun et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

It is of great significance to improve the driving range prediction accuracy to provide battery electric vehicle users with reliable information. A model built by the conventional multiple linear regression method is feasible to predict the driving range, but the residual errors between -3.6975 km and 3.3865 km are relatively unfaithful for real-world driving. The study is innovative in its application of machine learning method, the gradient boosting decision tree algorithm, on the driving range prediction which includes a very large number of factors that cannot be considered by conventional regression methods. The result of the machine learning method shows that the maximum prediction error is 1.58 km, the minimum prediction error is -1.41 km, and the average prediction error is about 0.7 km. The predictive accuracy of the gradient boosting decision tree is compared against that of the conventional approaches.

1. Introduction

With the rapid development of automobile industry and the continuous improvement of people's living standard, car ownership and sales continue to rise, which brings a series of energy and environment problems. In the face of increasing energy and environmental problems, the development of new energy vehicles has become a new trend in the automobile industry [1], and the battery electric vehicle (BEV) is the main force of new energy vehicles. However, BEVs have many disadvantages compared with conventional fuel vehicles; for example, the charging station has a sparse distribution, the charging time is too long, and the energy stored per unit of mass is lower in electrochemical batteries with respect to fossil fuels [2]. Besides, users of BEVs have a range anxiety problem that the residual power will be worried about not ensuring to reach the destination. All of these restrict the promotion and development of BEVs. The range anxiety is an easier case to be figured out than other issues in real-world application of BEVs [3]. Therefore, it is of great significance to increase the practicability and reliability of

BEVs by improving the driving range prediction accuracy to provide users with reliable information [4].

Therefore, various mathematic methods have been used in the driving range prediction to improve the accuracy and the credibility of it. The driving mode was incorporated into the study of driving range; it indicates that the stable driving habit plays an important role in saving the battery power and extending the driving range [5]. Fuzzy Transform, a model-free method, was adapted to online use for the prediction of remaining range of an electric vehicle [6]. A simple feature-based linear regression framework modeling the distribution parameters was proved to be an efficient approach to compute probabilistic attainability maps and model a driver's route preferences for electric vehicles [4]. A multiobjective problem, the driving range prediction, with maximized electric motor efficiency and minimized energy consumption, was solved to get the optimal speeds, along with the total trip time corresponding to a predicted driving range [7]. In another study, the energy consumption was analyzed and it was found that the electric vehicle has the lower energy consumption in the lower speed and more frequent

TABLE 1: Statistical comparison of SOC before and after deletion operations.

Interval	Original frequency number	Frequency number after deletion	Original frequency	Frequency after deletion
(0,10]	40	10	0.0003	0.0001
(10,20]	134	54	0.0009	0.0004
(20,30]	1920	1438	0.0127	0.0113
(30,40]	7168	6130	0.0475	0.0481
(40,50]	14914	12638	0.0988	0.0992
(50,60]	18916	15854	0.1253	0.1244
(60,70]	23028	18954	0.1526	0.1487
(70,80]	27148	22468	0.1799	0.1763
(80,90]	24066	21354	0.1595	0.1676
(90,100]	33584	28528	0.2225	0.2239
Total	150918	127428	1	1

stops [1]. In addition, basing on the LR (linear regression) and SVR (support vector regression) and the neural network, genetic algorithm and fuzzy logic intelligent optimization methods were fused into driving range prediction model of energy consumption to improve the prediction accuracy [8]. A real-time method was proposed to estimate the continuous driving range, considering both the driving behavior and the steepness of the driving route [9]. In another research, the relationship between the energy consumption and the load (air conditioning, heating, etc.) was studied to put forward the prediction model in different load and different driving mode [10, 11].

Many studies used to take less factors into account when establishing the prediction model of driving range, which might lead to the poor applicability and prediction accuracy of the model. A series of energy equations based on linear models and Dijkstra's graph search algorithm were derived to calculate the driving range and the route minimizing energy consumption available to EVs based on the real-world traffic condition and topology of the road. However, weight, temperature, and many other parameters were not be included in this work [12]. The battery remaining discharge energy prediction technique was studied by an energy prediction method based on the coupled prediction of future energy-related variables, but the future temperature variation was not be considered [13].

In a word, there are many methods that can predict the driving range and many factors that affect the driving range prediction, but the current studies cannot take both the accuracy and comprehensiveness of them into account [14].

For the problem existing in the previous studies, basing on the gradient boosting decision tree (GBDT), a new driving range prediction method, the machine learning method novel in including a large number of feature variables [15], has been presented to improve both the applicability and the accuracy, considering the real-world working condition, the battery status, and the traffic environment. The organization of the study is as follows. The collecting and processing of data is simply introduced in Section 2. The conventional multiple linear regression model of driving range is established and verified in Section 3. The machine learning method for the prediction problem is presented in Section 4. To investigate

the prediction performance of the proposed method, a comparative study is conducted in Section 5. Finally, conclusions are drawn in Section 6. The nomenclature of symbols and abbreviations in this study is shown in Nomenclature.

2. Data Collecting and Processing

2.1. Data Collecting. To begin with we will provide a brief introduction on the data collection. The real-time data of travel status and battery status, collected by vehicle-mounted information collection equipment, was sent to the remote data monitoring center every 5~10 seconds by GPRS wireless transmission network, and for storage. The research in this paper is based on the historical operation data of BEV, of which the type is E150EV produced by Baic New Energy Automobile Co., Ltd., rented and managed by a car-sharing company in Beijing. Of all the rented BEVs, No. 25 BEV, which has the longest running time and the largest data volume, is selected as the main research object. The discharge data, including 596 discharge process and 523,678 original data from March 1, 2015, to March 1, 2016, is extracted from the database, filtered, and processed.

2.2. Data Processing. The information about vehicle state and battery status transmits through the wireless network. In the process, the transmission can be affected by many factors, such as weather, building density, channel conflict, data stability, and so on. Therefore, there will be data losses and errors in the collected data.

For subsequent analysis and modeling, deletion has been operated on the attributes (SOC, current, voltage, speed, etc.) of repeated and error data. Table 1, for example, shows the results of frequency number and frequency before and after the deletion operation of SOC.

It can be seen from Table 1 that frequency number after the deletion operation is reduced, while frequency after the deletion operation is basically the same as the original frequency. The result proves that the error is generated with the random influence of the driving environment and the data acquisition device, rather than deliberately. As shown above, the frequency number of SOC between 0 and 20 is relatively scarce. Besides, previous studies find that the battery

TABLE 2: Analysis of the performance index of interpolation.

Index	SOC	<i>V</i> Total	Max <i>V</i>	Min <i>V</i>	Max <i>T</i>	Min <i>T</i>	Speed	TotalMile
RMSE	0.178	3.2	0.0262	0.0251	3.2E-15	0.2331	15.21	0.381
RMSRE	0.0021	0.0098	0.0079	0.0067	1.12E-16	0.012	54.23	0.0001

TABLE 3: Correlation test result between parameters and driving range.

	No. 25 BEV		No. 15 BEV		No. 12 BEV	
	R	Sig.	R	Sig.	R	Sig.
<i>SOC</i>	-0.998	.000	-0.999	.000	-0.998	.000
<i>Speed</i>	0.029	.775	-0.172	.089	0.021	.791
<i>V</i> Total	-0.946	.000	-0.826	.000	-0.861	.000
<i>I</i> Total	0.145	.149	-0.241	.153	0.137	.159
Max <i>V</i>	-0.827	.000	-0.789	.000	-0.830	.000
Min <i>V</i>	-0.850	.000	-0.808	.000	-0.858	.000
Max <i>T</i>	0.816	.000	0.970	.000	0.826	.000
Min <i>T</i>	0.889	.000	0.963	.000	0.899	.000
<i>EVD</i>	0.063	.534	0.092	.377	0.081	.517
<i>ETD</i>	-0.163	.104	-0.159	.115	-0.173	.109

R represents Pearson's simple correlation coefficient between parameters and the distance range; Sig. represents the probability value p of t test statistics, and $\alpha = 0.05$.

performance is unstable when the *SOC* of the battery is less than 10%, which is easy to cause irreversible damage to the physical properties of the battery [11]. Battery performance is relatively stable only when *SOC* is above 15%. Therefore, the *SOC* should be greater than or equal to 20% in the calculation of the driving range prediction in this study.

To facilitate subsequent analysis and modeling, Lagrange interpolation method has been used to make up the data gaps, making sure each discharge process complete. To accurately determine the interpolation effect, the root mean square error and the relative error of root mean square are calculated, as shown in Table 2.

In addition, the processed data has been averaged, which conforms to the requirements of modeling for accuracy and standard.

3. Multiple Linear Regression Modeling

3.1. Correlation Analysis. Generally, there are many factors affecting the driving range of BEV under the actual working conditions, including the driver's own characteristics, the vehicle's own parameters, and the road environment, etc. However, only several items of data can be collected and used. Therefore, the performance parameters of battery (*SOC*, voltage, current, and temperature) and state parameters of the vehicle (speed) are chosen to be researched [5].

Considering that the data used in this paper is distance-dependent, Pearson's simple correlation coefficient is used to measure the strength of the correlation degree between the driving range and another variable. The definition of Pearson's simple correlation coefficient is shown in the following:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (1)$$

where n is the number of samples, x_i and y_i are the variable values of two variables, respectively, and \bar{x} and \bar{y} are the corresponding mean values, respectively. When $|r| > 0.8$, there is a strong linear correlation between the two variables; when $|r| < 0.3$, the linear correlation between the two variables is weak. Corresponding to Pearson's simple correlation coefficient are t test statistics, and its mathematical definition is as follows:

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}} \quad (2)$$

when the probability value p of t -test statistics is less than the significance level α , the two variables are generally considered to have significant linear correlation. Otherwise, there is no significant linear correlation between the two variables.

No. 25 BEV discharge process data, from 09:45 to 14:30 on September 1, 2015, No. 15 BEV discharge process data, from 10:05 to 15:30 on August 11, 2015, and No. 12 BEV discharge process data, from 09:23 to 13:45 on July 20, 2015, are selected to calculate Pearson's simple correlation coefficient and the probability value p , reflecting the correlation between parameters and driving range in a numerical way, as shown in Table 3.

From Table 3, the driving range has a strongly linear relationship with *SOC*, total voltage, maximum cell voltage, minimum cell voltage, maximum cell temperature, and minimum cell temperature ($|R|_{SOC,VTotal,MaxV,MinV,MaxT,MinT}^{25,15,12} > 0.8$, $P_{SOC,VTotal,MaxV,MinV,MaxT,MinT}^{25,15,12} \ll 0.01$), respectively. There is no significant linear relationship between speed, total current, extreme voltage difference, extreme temperature difference, and driving range ($|R|_{Speed,ITotal,EVD,ETD}^{25,15,12} < 0.8$, $P_{Speed,ITotal,EVD,ETD}^{25,15,12} > 0.05$), respectively.

TABLE 4: Partial correlation test results under SOC is controlled.

	No. 25 BEV		No. 15 BEV		No. 12 BEV	
	R	Sig.	R	Sig.	R	Sig.
<i>VTot</i>	-0.172	.088	-0.251	.067	-0.209	.045
<i>MaxV</i>	-0.100	.322	-0.214	.388	-0.301	.276
<i>MinV</i>	-0.267	.007	-0.321	.012	-0.491	.034
<i>MaxT</i>	0.408	.000	0.517	.000	0.559	.000
<i>MinT</i>	0.487	.000	0.629	.000	0.537	.000

R represents partial correlation coefficient between parameters and the distance range; Sig. represents the probability value p of t test statistics, and $\alpha = 0.05$.

3.2. Partial Correlation Analysis. In multivariate correlation analysis, Pearson's simple correlation coefficient, however, generally cannot truly reflect the correlation between variables. Because the relationship between variables is more complex at this time, it may be affected by more than one variable respectively. Currently, partial correlation coefficient is a better choice. Partial correlation coefficient reflects the degree of net correlation between variables.

When analyzing the partial correlation between variables x_1 and y , under the condition of controlling the linear action of x_2 , the first-order partial correlation coefficient between x_1 and y is defined as follows:

$$r_{y1,2} = \frac{r_{y1} - r_{y2}r_{12}}{\sqrt{(1 - r_{y2}^2)(1 - r_{12}^2)}} \quad (3)$$

where r_{y1} , r_{y2} , and r_{12} is the correlation coefficient of y and x_1 , is the correlation coefficient of y and x_2 , and is the simple correlation coefficient of x_1 and x_2 . The basic steps of partial correlation analysis are as follows: firstly, the null hypothesis is proposed; that is, the partial correlation coefficient between two populations is not significantly different from zero. Secondly, the test statistic of partial correlation analysis is t statistic, whose mathematical definition is shown in the following:

$$t = r \sqrt{\frac{n - q - 2}{1 - r^2}} \quad (4)$$

where r is the partial correlation coefficient, n is the sample number, q is the order number, and $n - q - 2$ is the degree of freedom. Thirdly, calculate the observation value of the t -test statistic and the corresponding probability value p . Lastly, if the probability value p of the t -test statistic is less than the given significance level α , the null hypothesis should be rejected and the partial correlation coefficient of the two populations is significantly different from zero. Otherwise, it is considered that there is no significant difference between the partial correlation coefficient and zero of the two populations.

No. 25 BEV discharge process data, from 09:45 to 14:30 on September 1, 2015, No. 15 BEV discharge process data, from 10:05 to 15:30 on August 11, 2015, and No. 12 BEV discharge process data, from 09:23 to 13:45 on July 20, 2015, are selected to calculate the partial correlation coefficient and the probability value p , determining whether the correlation between each parameter and the driving distance is affected by other parameters.

From Table 3, SOC has the highest absolute value of the simple correlation coefficient, while total current, speed, extremum voltage difference, and extreme temperature difference have no significant linear relationship with the driving range, respectively. Therefore, SOC is selected as control variable, and partial correlation coefficients of total voltage, maximum cell voltage, minimum cell voltage, maximum cell temperature, and minimum cell temperature are calculated.

As can be seen from Table 4, the linear relationship between total voltage, maximum cell voltage and minimum cell voltage, and the driving distance is affected by SOC. Therefore, after controlling the variable SOC, total voltage, maximum cell voltage, and minimum cell voltage have no significant linear effect on the driving range ($|R|_{VTot,MaxV,MinV}^{25,15,12} < 0.5$, $P_{VTot,MaxV,MinV}^{25,15,12} > 0.05$). Correspondingly, there is a significant linear correlation between maximum cell temperature, minimum cell temperature, and the driving distance ($|R|_{MaxT,MinT}^{25,15,12} > 0.4$, $P_{MaxT,MinT}^{25,15,12} \ll 0.01$).

According to the above correlation analysis and partial correlation analysis, minimum cell temperature has the second highest correlation with the driving range, so it is selected as the control variable for the partial correlation test of maximum cell temperature and the driving range. From the partial correlation test results, the relationship between the driving distance and maximum cell temperature is affected by minimum cell temperature, and there is no significant linear correlation between them ($R_{MaxT}^{25} = -0.066$, $P_{MaxT}^{25} = 0.519 > 0.05$; $R_{MaxT}^{15} = -0.129$, $P_{MaxT}^{15} = 0.351 > 0.05$; $R_{MaxT}^{12} = -0.218$, $P_{MaxT}^{12} = 0.229 > 0.05$).

3.3. Variable Selection and Modeling. In multivariate linear regression analysis, it is very important to choose the right independent variables to enter the regression model to make it have better generalization ability and higher prediction accuracy. It is necessary that only independent variables that play a major role are retained and the average variation of the dependent variable is described with fewer independent variables. It can avoid the problem of overfitting and generalization ability reducing caused by the entry of all relevant variables into the model. Therefore, based on the correlation analysis and partial correlation analysis results, some parameters that have greater impact on the dependent variable can be considered and selected as the independent variables. On the contrary, other parameters that have little influence on the dependent variable can be ignored.

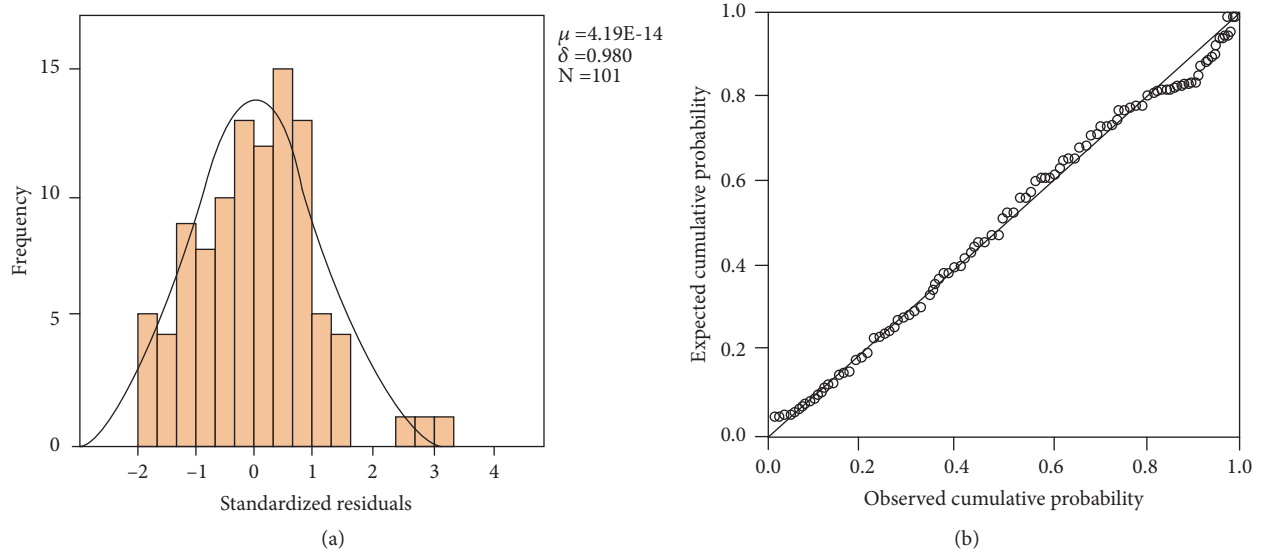


FIGURE 1: The statistical tests of standardized residuals.

In view of the above result of the correlation analysis and partial correlation analysis, SOC and minimum cell temperature have been selected into the variables of the model. The multiple linear regression model is as follows:

$$M = \beta_0 + \beta_1 s + \beta_2 t + \varepsilon \quad (5)$$

where M represents the driving range, the unit being km; s represents SOC, the value ranges from 20 to 100; t represents minimum cell temperature; β_0 , β_1 , β_2 are parameters to be measured; ε is the residual error.

3.4. Parameter Identification and Statistical Test. When the regression model is determined, it is necessary to use the collected data to identify unknown parameters in the model according to certain estimation criteria. The least square method is widely used to identify parameters because of its excellent properties. No. 25 BEV discharge process data, from 09:45 to 14:30 on September 1, 2015, are used as input, and the least square parameter identification has been performed. The parameter identification results ($\beta_0 = 126.960$, $\beta_1 = -1.719$, $\beta_2 = 1.627$) are introduced into (3), and the driving range prediction model is as follows:

$$M = 126.960 - 1.719s + 1.627t + \varepsilon \quad (6)$$

A variety of statistical tests are conducted to ensure that the model has good stability and generalization ability, and the results are obtained as shown in Figure 1.

As can be seen from Figure 1(a), the residual sequence of the model is basically normal distribution, with the mean value of $4.19E-14$, which approximates 0, and the standard deviation is 0.98. Figure 1(b) shows that the residual distribution of the observed value is compared with the normal distribution, standardized residual distribution scatter is very close to the straight line, so that standardized residuals obey normal distribution with mean zero. According to the statistical test results, the goodness of fit is high ($\bar{R}^2 = 0.996$);

the linear relationship between the driving range and the explained variables is significant ($F = 14095.605$, $p < 0.01$); the linear relationship between the driving range and each of the explained variables (SOC, $MinT$) is significant ($t_1 = 12.328$, $t_2 = -76.532$, $t_3 = 5.525$, $p_{1,2,3} < 0.01$); there is no autocorrelation between residuals; the residual sequence is independent ($DW = 2.23$).

To sum up, the multiple regression model satisfies a series of requirements of statistical test, and the model can be used to predict and analyze.

3.5. Model Establishment and Verification. No. 25 BEV total 10 discharge process data, from September 2 to September 22, 2015, have been chosen to conduct the pretreatment and the least squares parameter identification, making the model have higher prediction precision and applicability. Then, the final model parameters were obtained in order: $\beta_0 = 126.527$, $\beta_1 = -1.579$, $\beta_2 = 1.564$. The final driving range prediction model is as follows:

$$M = 126.527 - 1.579s + 1.564t \quad (7)$$

where the value range of s is [20, 100].

No. 15 BEV discharge process data, on September 11, September 19, and September 28, 2015, have been selected to further verify the reliability and practicability of the model. The results of the residual error sequence are shown in Figure 2, and the statistical residual errors are shown in Table 6.

It can be seen from Table 5 that the residual error is between -3.6975 km and 3.3865 km, the mean absolute error is about 1.5 km, the root-mean-square error is less than 2 km, and the root-mean-square relative error is less than 0.5 km. Although it is feasible to predict the driving range by the multiple linear regression model, the residual errors are relatively large for real-world driving condition.

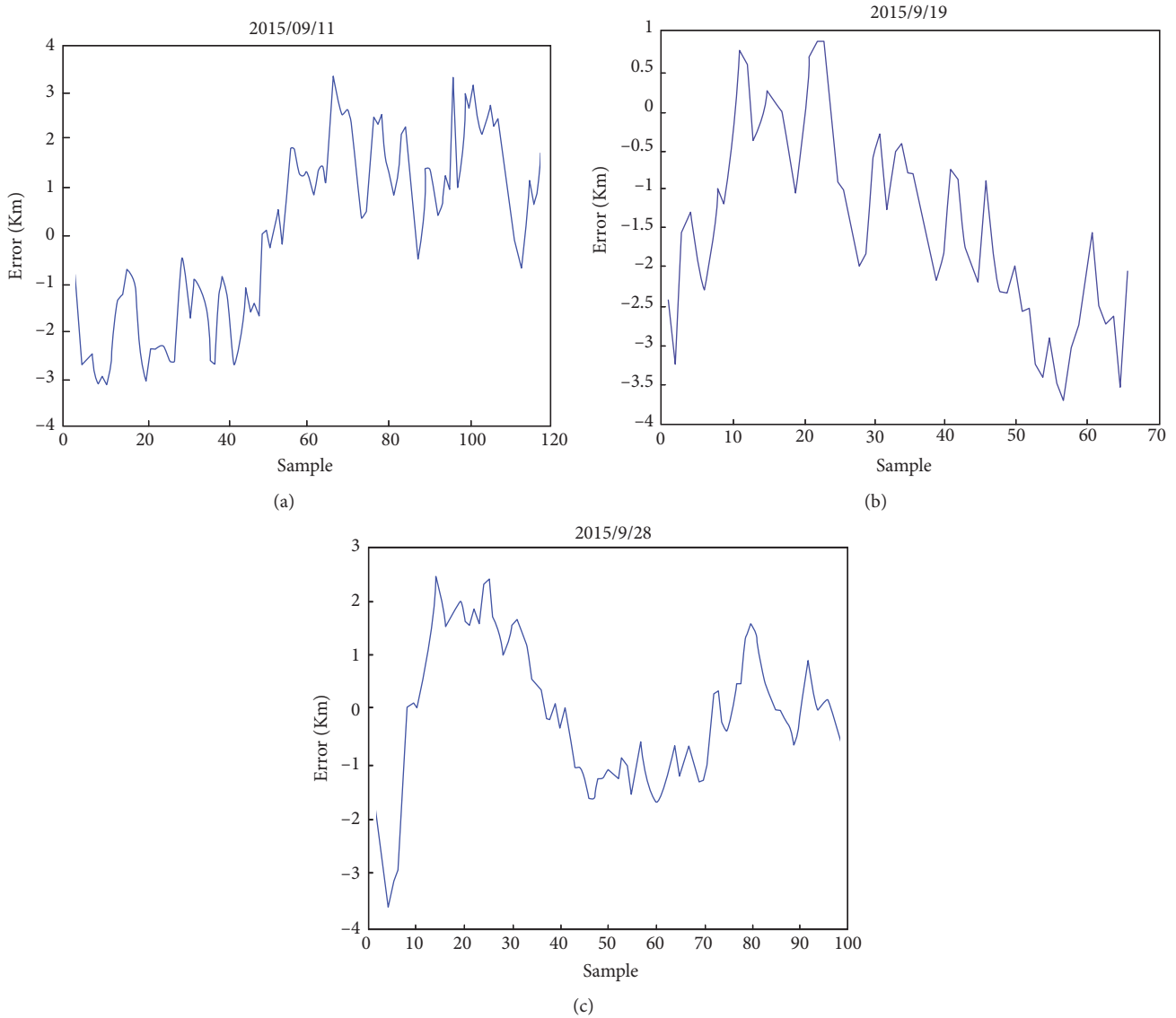


FIGURE 2: The residual error sequence chart of No. 15 BEV.

TABLE 5: Statistical error results of prediction.

Date	Maximum error	Minimum error	MAE	RMSE	RMSPE
2015/9/11	3.3865	-3.1664	1.6260	1.8575	0.229
2015/9/19	0.8749	-3.6975	1.5701	1.8523	0.418
2015/9/28	2.5925	-3.6078	1.4612	1.3026	0.377

4. Machine Learning Methods

4.1. Classification and Regression Tree. Decision tree is a kind of classification and regression method. Decision tree method generally includes three processes: the feature selection, the tree creation, and the tree pruning (remove fitting). It can summarize some good performance classification rules from training set, which not only can well fit the training data, but also can make well predictions to the unknown data.

TABLE 6: Prediction error of the GBDT model.

Date	Minimum error	Maximum error	MAE
2015/3/10	-0.72	1.49	0.61
2015/8/21	-0.59	1.58	0.82
2016/1/9	-1.41	1.52	0.76

Classification and Regression Tree (CART) was put forward by Breiman et al. in 1984, different with ID3 and

C4.5 classification tree whose none-leaf nodes have multiple branches, CART's none-leaf nodes only have two branches, and its output values of a leaf node are the mean of the sample label [16]. Therefore, the generation process is to construct the binary decision tree based on the training set recursively, and to prune the generated trees by using the loss function and validation set.

4.1.1. CART Generation. First, a training set D is given as follows:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \quad (8)$$

where y is a continuous variable. CART model generated under D is defined as follows:

$$f(x) = \sum_{i=1}^M c_i I(x \in R_i) \quad (9)$$

From (9), CART divides its eigenspace into M units R_1, R_2, \dots, R_M , and each unit corresponds to a fixed output value c_i . The generation process of CART can be expressed as follows.

Algorithm Framework 1: CART Generation

Input: A training set D ;

Output: CART $f(x)$;

Begin

In the characteristic space of D , each region is divided into two subregions recursively, and the optimal output value of each subregion is calculated, and the binary decision tree is constructed.

(1) Solve (10); select the optimal cut variable j and the optimal cut point s .

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right] \quad (10)$$

Equation (11) is the space value of R_1 and R_2 :

$$\begin{aligned} R_1(j, s) &= \{x \mid x^{(j)} \leq s\}, \\ R_2(j, s) &= \{x \mid x^{(j)} > s\} \end{aligned} \quad (11)$$

Iterate through the variables j , and then scan the cut point s orderly in the specified cut variable j , and select the value pair (j, s) to make sure that (8) is minimum.

(2) Figure out the corresponding optimal output value:

$$c_i = \frac{1}{N_i} \sum_{x_k \in R_i(j,s)} y_i, \quad x \in R_i, \quad i = 1, 2 \quad (12)$$

(3) Continue to call steps (1) and (2) of the two subregions until the stop condition is satisfied.

(4) The input space of D is divided into M regions R_1, R_2, \dots, R_M and CART is generated (9).

End

By the CART generation algorithm, each time the recursive calculation, the optimal output value is generated from each division unit using the least square error criterion; that is, the optimal output value is the mean of all labels on the unit; a heuristic algorithm is used to solve the optimal cut variables and optimal cut points. The decision tree constructed from the above generation algorithm is called the least square CART.

4.1.2. CART Pruning. In view of the problem of overfitting in the CART generated above, the pruning operation is necessary. The CART pruning is cut from the bottom end of the decision tree to make it simple, so that the unknown data has better generalization ability and higher prediction accuracy.

In the pruning process, the loss function of subtree is calculated by the following:

$$\begin{aligned} C_\alpha(T) &= C(T) + \alpha |T|, \\ C(T) &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \end{aligned} \quad (13)$$

where T represents any subtree, $C(T)$ represents the square error of training data, $|T|$ is the number of leaf nodes of T , $\alpha (> 0)$ represents the fitting degree and the complexity of the model, $C_\alpha(T)$ represents the overall loss in the subtree under α , and the only optimal subtree for fixed α exists. The CART pruning algorithm is given as follows.

Algorithm Framework 2: CART Pruning

Input: T_0 constructed from CART generation;

Output: the optimal CART T_α ;

Begin

(1) Suppose $k = 0, T = T_0$;

(2) Suppose $\alpha = +\infty$;

(3) Calculate from the top down on $C(T_t), |T|$, and

$$g(t) = \frac{C(t) - C(T_t)}{|T_t| - 1} \quad (14)$$

$$\alpha = \min(\alpha, g(t)) \quad (15)$$

(4) $g(t) = \alpha$ is pruned by the internal node t , the output value of the leaf node t is calculated by average method, and the tree T is obtained;

(5) $k = k + 1, \alpha_k = \alpha, T_k = T$;

(6) Determine whether T_k is composed of the root node and two leaf nodes, if it is, $T_k = T_n$; if not, go back to step (3);

(7) Based on the independent verification data set, the cross-validation method is used to select the optimal subtree

T_α in the subtree sequence $\{T_k\}$ ($k = 1, 2, \dots, n$) according to the square error.

End

In the above algorithm, $g(t)$ represents the decrease degree of the total loss function after pruning. It is indicated that (1) the size of the optimal subtree T_α is positively correlated with the size of α ; (2) the subtrees in the corresponding subtree sequences $\{T_k\}$ ($k = 1, 2, \dots, n$) are nested by small increments α ; (3) in the optimal subtree sequence, each subtree T_k corresponds to one α , so when the optimal subtree T_k is determined, the corresponding α is determined. When the pruning operation is completed, it is possible to integrate the new base learner into the existing GBDT model.

4.2. Gradient Boosting Decision Tree. The CART is used as the base learner in the gradient boosting decision tree (GBDT) [17]. For its excellent performance, GBDT is widely used in various fields of real life.

4.2.1. Estimation Function. The purpose of GBDT algorithm is to estimate the unknown function [18]. Since it is a kind of supervised learning, the prerequisite for learning is to have enough data sets with labels $(x_i, y_i)_{i=1}^N$, where N is the size of the sample set, $x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)})^T$, y_i is the sample label. The purpose of supervised learning is to give an estimation function $\hat{f}(x)$ to the real function $f: x \rightarrow y$ and to minimize the loss function $L(y, \hat{f}(x))$ to improve the accuracy of the prediction, as shown in the following:

$$\hat{f}(x) = \arg \min_{f(x)} L(y, \hat{f}(x)) \quad (16)$$

Equation (16) can also be written to the minimized expected loss form, as shown in the following:

$$\hat{f}(x) = \arg \min_{f(x)} E_x [E_y [L(y, \hat{f}(x))] | x] \quad (17)$$

To materialize the target problem, the parameters θ of the search space are limited, as shown in the following:

$$\hat{\theta} = \arg \min_{\theta} E_x [E_y [L(y, \hat{f}(x, \theta))] | x] \quad (18)$$

So far, no specific formal assumptions have been made on estimation functions and real functions. Moreover, in most cases, the problem described above does not have a closed form solution, so the recursive numerical process is usually optimized.

4.2.2. Optimization Method. At normal circumstances, the loss function adopted in optimizing is square loss function and index loss function; the general Boosting algorithm (such as AdaBoost) can achieve the goal of optimization. However, for general loss function, it is difficult to adopt common optimization methods. In response to this problem, Friedman proposed GBDT algorithm, using the value of the loss function in the negative gradient direction, as shown

in (19), to approximate residuals and fit regression trees, improving the performance of the prediction model.

$$- \left[\frac{\partial L(y, f(x_i))}{\partial f(x_i)} \right] \quad (19)$$

GBDT is an algorithm to recursively solve prediction model. In the beginning of each stage of solving, unperfect model, a very weak model, can be used only to predict the average of the training set; and then a better model can be got by adding an estimator $h(x)$ to $F_m(x)$, as shown in the following:

$$F_{m+1}(x) = F_m(x) + h(x) \quad (20)$$

According to the empirical risk minimization principle,

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma) \quad (21)$$

$$F_m(x) = F_{m-1}(x) + \arg \min_h \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + h_m(x_i)) \quad (22)$$

Then, the gradient descent method is used to minimize the loss function, and the model is updated according to the following:

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^n \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)) \quad (23)$$

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) - \gamma \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))) \quad (24)$$

To sum up, the algorithm framework of GBDT is as follows:

Algorithm Framework 3: GBDT

Input:

- (i) A labeled training set D
- (ii) Iterations M
- (iii) The loss function $L(y, f)$
- (iv) The base learner $h(x)$

Output: A prediction model $F_m(x)$

Begin

- (1) Initialization model:

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma) \quad (25)$$

- (2) **For i = 1 to M, do**

Calculated pseudo residuals:

$$\gamma_{im} = - \left[\frac{\partial L(y, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \quad i = 1, 2, \dots, n \quad (26)$$

Obtain $h_m(x)$ using CART to fit pseudo residual, and calculate the weighted coefficient γ_m :

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)) \quad (27)$$

Update model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \quad (28)$$

(3) Get the final prediction model $F_m(x)$

End

In some cases, overfitting and prediction error bias may occur in the above algorithm. In general, the regularization technique can be used to reduce overfitting effect by controlling the fitting process, so the updating rules of the above algorithm are modified as follows:

$$F_m(x) = F_{m-1}(x) + v \cdot \gamma_m h_m(x), \quad 0 < v < 1 \quad (29)$$

where v is called the “learning rate”, which is the weight reduction coefficient of the base learner.

It has been found that a small learning rate ($v < 0.1$) can significantly improve the generalization ability of the model, but the disadvantage is that the number of iterations is increased. Overall, a regularized GBDT algorithm framework is adopted in the following modeling process.

5. GBDT Modeling

5.1. Data Integration and Feature Extraction. Above all, No. 25 BEV discharge data from March 1, 2015, to March 1, 2016, is selected as training set, and No. 15 BEV discharge process data on January, March, and August 2015, is selected as test set.

Considering the influence of the external environment on BEVs [19], weather information of Beijing urban area needs to be integrated in the training and test set, which comes from the national meteorological science data sharing service platform. The speed variable in the data is processed to average for its frequent change and nonlinear effect on the driving range, namely, the speed of the driving range for k corresponding to the average speed of driving range from 0 to k . In this way, the effect of average speed on the driving range is incorporated into the future prediction model.

The purpose of GBDT algorithm is to extract the structure and essence of the target problem from the original data set. To make the selected features well explain the current problem, the selection of features should meet the following requirements that can construct the prediction model with high efficiency and low consumption, improving prediction accuracy. In fact, the extraction of features is to select the optimal feature set for model training from the original

feature set. Good features often improve the prediction accuracy of GBDT algorithm. According to the previous analysis and research, *SOC*, *MaxT*, *MinT*, *MaxV*, *MinV*, *TotalV*, *EDT*, *EDV*, *AveSpeed*, *TotalMile*, *Temper*, *Visibility*, and *Precip* are extracted to train and test the model.

5.2. Parameter Setting and Relative Importance Calculation. GBDT algorithm needs to set some key parameters, including each iteration step length, v ; loss function, L ; maximum depth of tree, *MaxDepth*; number of iterations, N .

The specific steps of the parameter adjustment of GBDT algorithm are as follows.

(1) According to experience, the maximum depth of the tree is set to 10 (reference range for 6 to 20). Considering the accuracy requirement, the step length is set to 0.1; the loss function is set as the mean square error. Search for appropriate number of iterations within a range of 100 to 400.

(2) Then, the maximum depth of the tree and step length v are detected and adjusted until the optimal parameters are found.

In practice, the input features rarely have the same correlation. In order to understand the size of contribution of each characteristic in driving range prediction, the relative importance of input variables need to be calculated. The calculation of global relative importance of features is as shown in the following:

$$F_j = \frac{1}{M} \sum_{m=1}^M \hat{F}_j(T_m) \quad (30)$$

where M is the number of base learners. The importance of feature j in a single tree is as shown in the following:

$$\hat{F}_j(T_m) = \sum_{t=1}^{N-1} i_t(v_t = j) \quad (31)$$

where N is the number of leaf nodes, $N-1$ is the number of non-leaf nodes, v_t is the characteristic associated with node t , and i_t is the reduction value of square loss after node t division. In short, the importance of a feature is the mean of its importance in all the basic learners.

5.3. Result

5.3.1. Model Establishment. According to the parameter setting method above, the statistical error results of the initial iteration are shown in Figure 3.

As shown in Figure 3, when the number of iterations is [100, 300], the mean absolute error is rising, and the root mean square error is decreasing; when the number of iterations is greater than 300, the mean absolute error shows a downward trend, and the root mean square error is decreased after increasing trend. Since the maximum value of the mean absolute error is only 0.00466 from the minimum, considering the stability of the prediction model, the optimal iteration number is the number of iterations with the minimum root mean square error, 300. Then find the optimal maximum depth of the tree, and its statistical error is shown in Figure 4.

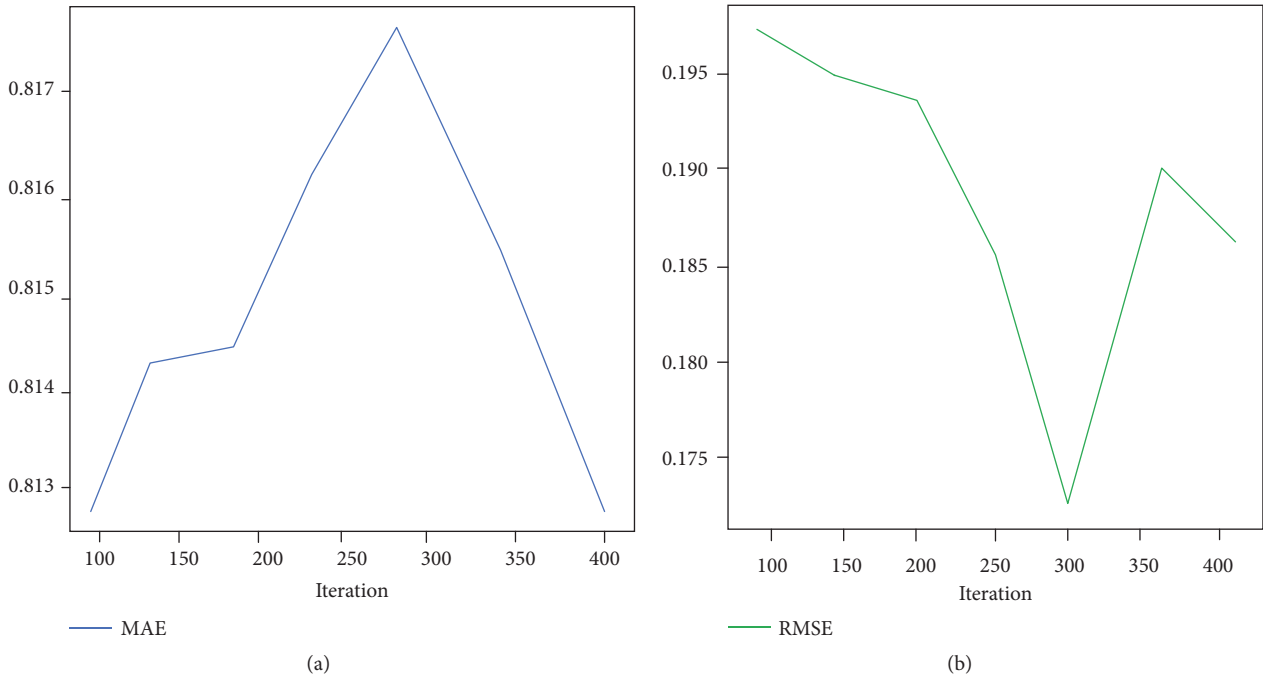


FIGURE 3: Statistical error results of iterations.

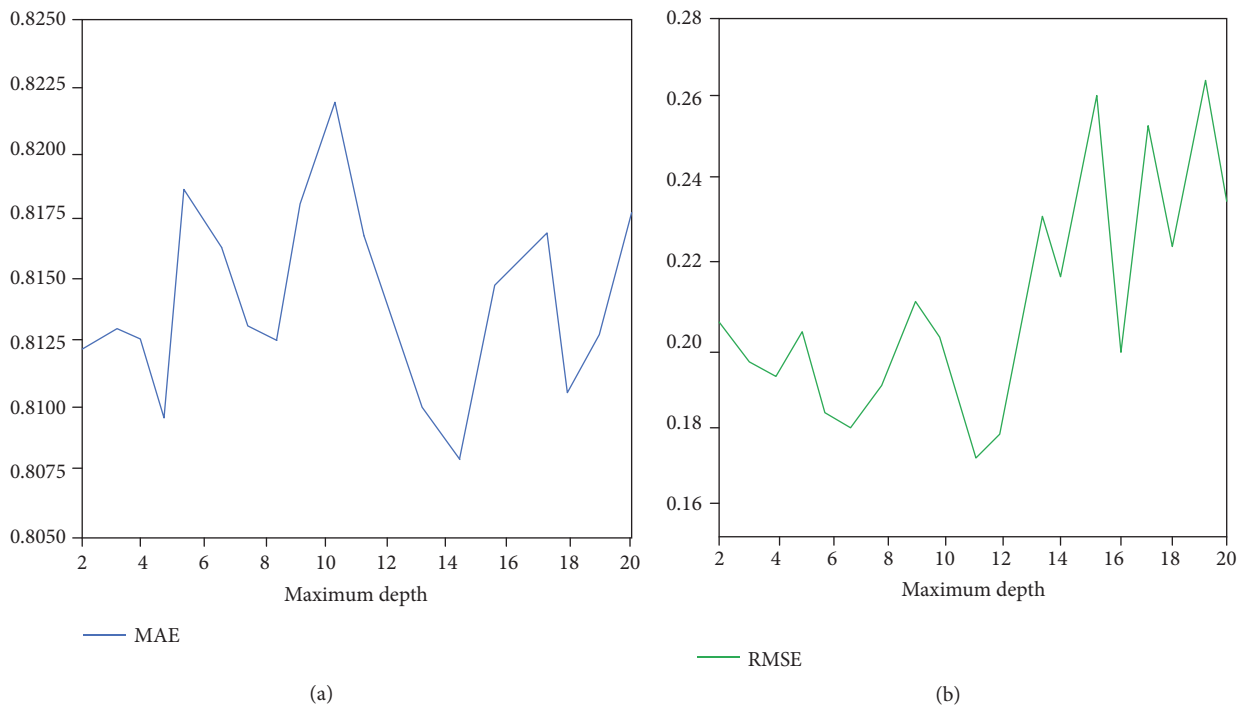


FIGURE 4: Statistical error results of maximum depth.

From Figure 4, as the maximum depth of the tree increases, the mean absolute error fluctuates. However, the difference between the maximum and minimum values of the mean absolute error is only 0.01212. To make the model have better robustness, the optimal maximum depth should be chosen according to the root mean square error. The

minimum mean square error of 0.1733 corresponds to the maximum depth of 11, which is the optimal maximum depth of tree. Then, other optimal parameters are detected as $\nu = 0.05$ and $N = 300$.

Training the GBDT model with the optimal parameters, the results are shown in Figure 5.

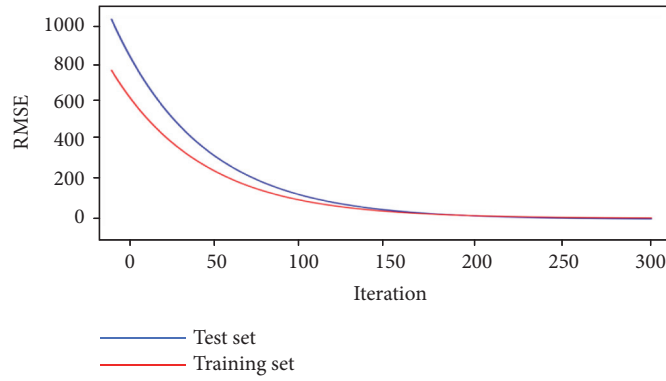


FIGURE 5: Results of model training.

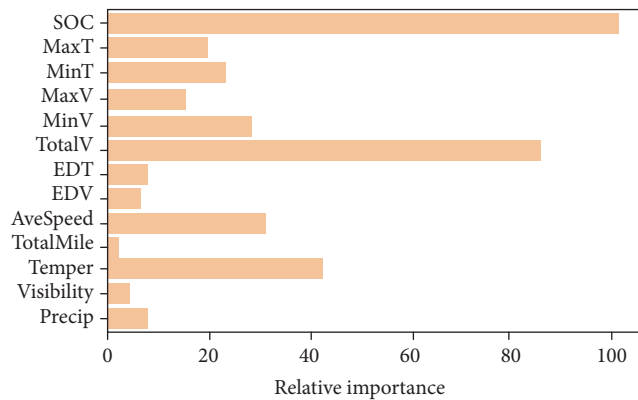


FIGURE 6: Relative importance of the feature.

Figure 5 shows that, in the beginning of the iteration, both the training set error and the test set error are large; the two errors decrease with the increase of the number of iterations; when the number of iterations reaches about 300, the two error curves basically coincide and stop changing. The error statistical results of the GBDT model are given as RMSE = 0.278, MAE = 0.813, maximum error = 1.61 and minimum error = -1.58.

According to (28), the relative importance of each feature is shown in Figure 6. It can be seen that SOC and TotalV are the key feature of GBDT model for driving range.

5.3.2. *Model Verification.* To verify the reliability of the GBDT prediction model, No. 12 BEV discharge process data on March 10, 2015, August 21, 2015, and January 9, 2016, are used for verification; the result is shown in Figure 7.

The results of the minimum error, the maximum error, and the mean absolute error obtained from the verification are shown in Table 6.

Table 6 shows that the maximum prediction error is 1.58 km, the minimum prediction error is -1.41 km, and the average prediction error is about 0.7 km.

5.4. *Discussion.* No. 15 BEV discharge process data on September 11, 2015, September 19, 2015, and September 28, 2015, have been selected as a data sample. To conduct

comparison analysis, three methods, that is, GBDT, CART, and the multiple linear regression (MLR), are performed on the same data sample. The comparison results are shown in Table 7.

When data has many features and the relationships between them are complex, the idea of building a global model is difficult. One approach is to use conventional linear regression analysis to model; some variables will be excluded from the model for the multicollinearity between them. However, that does not mean the model ignores the global impact of other variables. The excluded variables still affect the model because of the existence of the remaining variables. The model established by traditional regression method can be used to predict and the results are reliable. Another approach is to use the decision tree to model; CART is a widely used decision tree. In the regression with CART, each node has a predicted value, which is equal to the average value of all samples belonging to the node. When branching, the best segmentation point of each threshold value of each attribute is selected, and the criterion to be measured is to minimize the mean variance. The value of this node is set as the average value of the training sample that falls on this node until it is indivisible or reaches a certain height or the attribute is used up or the mean square error does not decrease. The test samples are dropped according to the segmentation points during the training and fall to the leaf

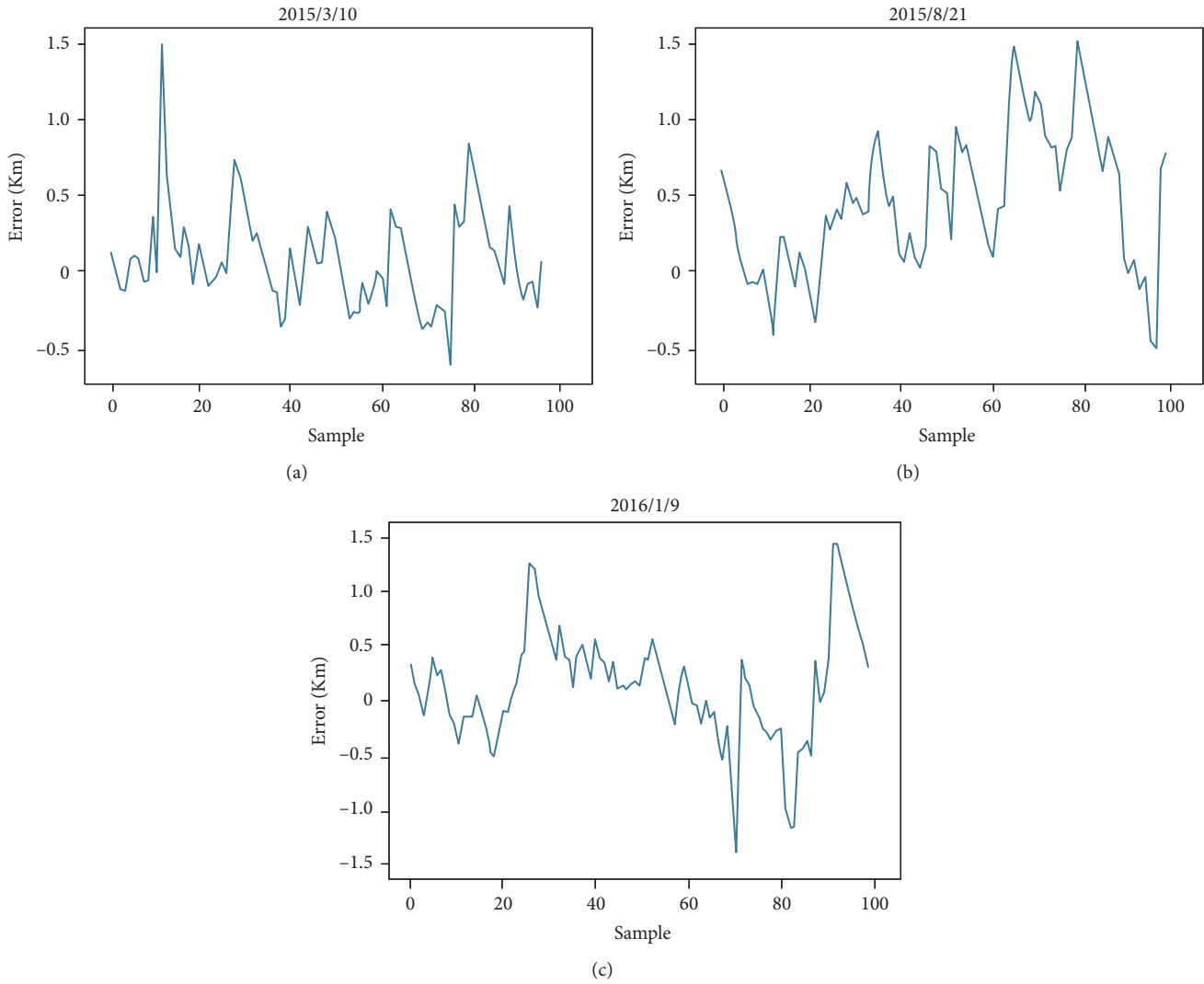


FIGURE 7: Residual error sequence chart of No. 12 BEV.

TABLE 7: Comparison results of predictive performance.

Data	Maximum error			Minimum error			MAE		
	GBDT	CART	MLR	GBDT	CART	MLR	GBDT	CART	MLR
2015/9/11	1.234	2.034	3.3865	-0.98	-1.684	-3.1664	0.678	1.237	1.6260
2015/9/19	1.459	1.497	0.8749	-1.023	-1.767	-3.6975	0.719	1.116	1.5701
2015/9/28	1.501	1.972	2.5925	-1.237	-1.791	-3.6078	0.821	1.084	1.4612

nodes. The average value of the leaf nodes is the predicted value. Moreover, GBDT is an algorithm based on CART and is an iterative tree. Obviously, all variables are considered by GBDT and CART in the process of modeling. Although conventional linear regression only introduces some variables in the final model, it is also established based on all variables. Therefore, it is no problem to compare their prediction results.

It can be seen from Table 7 that the predictive performance (maximum error, minimum error, and mean absolute error) of GBDT is overall better than that of CART. GBDT

produces a weak classifier through multiple iterations, each iteration produces a weak classifier. Each classifier trains based on the residual of the classifier in the previous round and continuously improves the accuracy of the final classifier by reducing the deviation. Furthermore, even if GBDT and CART takes as many as 13 parameters into consideration, the predictive performance (maximum error, minimum error, and mean absolute error) of MLR model is the worst of them. Therefore, it is apparent that the GBDT model has higher prediction accuracy and reliability. It indicates that the GBDT model of driving range has better predictive performance

and can better meet the requirements of real-world driving conditions.

6. Conclusions

In recent years, the number of BEVs is increasing gradually, but the problem of inaccurate residual power display has been restricting the promotion and the use of BEVs. The purpose of this study is to solve the problem of “range anxiety” caused by battery performance and other factors by predicting the BEV driving range. Many studies usually take less factors into account when establishing the prediction model of driving range, which may lead to the poor applicability and prediction accuracy of the model. In this study, a prediction model for BEV driving range based on machine learning has been established. The study is innovative in its application of machine learning method, GBDT algorithm, which includes a very large number of feature variables that cannot be considered by conventional regression methods. Moreover, the study is novel in its accuracy and reliability of a prediction model for BEV driving range.

As the GBDT model belongs to the black box algorithm, it can only give the importance distribution of the feature variables but cannot specify the interconnection and interaction between the feature variables. In future studies, there is a lot of research space for the correlation of variables within the model. The prediction model proposed in this study can meet the requirements of actual working conditions, but it needs to be further optimized to improve the prediction accuracy in the future. For instance, the cloud computing can be applied in the task of modeling, which is responsible for irregularly training, to obtain more accurate prediction model.

Nomenclature

BEV:	Battery electric vehicle
GBDT:	Gradient boosting decision tree
CART:	Classification and regression tree
MLR:	Multiple linear regression
MAE:	Mean absolute error
RMSE:	Root mean square error
RMSPE:	Root mean square percent error
SOC:	State-of-charge
MaxT:	Maximum cell temperature
MinT:	Minimum cell temperature
MaxV:	Maximum cell voltage
MinV:	Minimum cell voltage
TotalV:	Battery set total voltage
EDT:	Extreme temperature difference
EDV:	Extreme voltage difference
AveSpeed:	Average speed of BEV
TotalMile:	Total driving range of BEV
Temper:	Environment temperature
Visibility:	Horizontal visibility
Precip:	Amount of precipitation.

Data Availability

Real-world operation BEV data used to support the findings of this study have not been made available because we

signed a confidential agreement with Yi Weixing (Beijing) Technology Co. Ltd., and all operation data related to commercial secrets is not suitable for disclosure. However, the authenticity and validity of all the data can be guaranteed.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported by National Key R&D Program of China under Grants no. 2018YFC0706005 and no. 2018YFC0706000.

References

- [1] X. Yuan, L. Li, H. Gou, and T. Dong, “Energy and environmental impact of battery electric vehicle range in China,” *Applied Energy*, vol. 157, pp. 75–84, 2015.
- [2] M. Ceraolo and G. Pede, “Techniques for estimating the residual range of an electric vehicle,” *IEEE Transactions on Vehicular Technology*, vol. 50, no. 1, pp. 109–115, 2001.
- [3] A. Shekhar, V. Prasanth, P. Bauer, and M. Bolech, “Generic methodology for driving range estimation of electric vehicle with on-road charging,” in *Proceedings of the IEEE Transportation Electrification Conference and Expo, ITEC '15*, June 2015.
- [4] P. Ondruska and I. Posner, “Probabilistic attainability maps: Efficiently predicting driver-specific electric vehicle range,” in *Proceedings of the 25th IEEE Intelligent Vehicles Symposium, IV '14*, pp. 1169–1174, June 2014.
- [5] C. Bingham, C. Walsh, and S. Carroll, “Impact of driving characteristics on electric vehicle energy consumption and range,” *IET Intelligent Transport Systems*, vol. 6, no. 6, pp. 29–35, 2012.
- [6] I. Cunningham and K. Burnham, “Online use of the fuzzy transform in the estimation of electric vehicle range,” *Measurement and Control*, vol. 46, no. 9, pp. 277–282, 2013.
- [7] W. Vaz, A. K. R. Nandi, R. G. Landers, and U. O. Koylu, “Electric vehicle range prediction for constant speed trip using multi-objective optimization,” *Journal of Power Sources*, vol. 275, pp. 435–446, 2015.
- [8] A. Bolovinou, I. Bakas, A. Amditis, F. Mastrandrea, and W. Vinciotti, “Online prediction of an electric vehicle remaining range based on regression analysis,” in *Proceedings of the 2014 IEEE International Electric Vehicle Conference, IEVC '14*, December 2014.
- [9] C. K. Wai, Y. Y. Rong, and S. Morris, “Simulation of a distance estimator for battery electric vehicle,” *Alexandria Engineering Journal*, vol. 54, no. 3, pp. 359–371, 2015.
- [10] E. Kim, J. Lee, and G. S. Kang, “Real-time prediction of battery power requirements for electric vehicles,” in *Proceedings of the ACM/IEEE International Conference on Cyber-Physical Systems*, IEEE, 2013.
- [11] G. Wager, J. Whale, and T. Braunl, “Driving electric vehicles at highway speeds: the effect of higher driving speeds on energy consumption and driving range for electric vehicles in australia,” *Renewable and Sustainable Energy Reviews*, vol. 63, pp. 158–165, 2016.
- [12] M. Neaimeh, G. A. Hill, Y. Hübner, and P. T. Blythe, “Routing systems to extend the driving range of electric vehicles,” *IET Intelligent Transport Systems*, vol. 7, no. 3, pp. 327–336, 2013.

- [13] G. Liu, M. Ouyang, L. Lu, J. Li, and J. Hua, "A highly accurate predictive-adaptive method for lithium-ion battery remaining discharge energy prediction in electric vehicle applications," *Applied Energy*, vol. 149, no. 1, pp. 297–314, 2015.
- [14] X. Yuan, C. Zhang, G. Hong, X. Huang, and L. Li, "Method for evaluating the real-world driving energy consumptions of electric vehicles," *Energy*, vol. 141, pp. 1955–1968, 2017.
- [15] Y. Zhang and A. Haghani, "A gradient boosting method to improve travel time prediction," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 308–324, 2015.
- [16] B. Trawiński, M. Smetek, Z. Telec, and T. Lasota, "Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms," *International Journal of Applied Mathematics and Computer Science*, vol. 22, no. 4, pp. 867–881, 2012.
- [17] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [18] Q. M. J. L. Ying Cao, "Advance and prospects of adaBoost algorithm," *Acta Automatica Sinica*, vol. 39, no. 6, pp. 745–758, 2013.
- [19] M. K. M. A. Mathias Petzl, "Lithium plating in a commercial lithium-ion battery - A low-temperature aging study," *Journal of Power Sources*, vol. 275, pp. 799–807, 2015.

Research Article

Developing a Travel Time Estimation Method of Freeway Based on Floating Car Using Random Forests

Juan Cheng , Gen Li , and Xianhua Chen 

School of Transportation, Southeast University, Nanjing 211189, China

Correspondence should be addressed to Xianhua Chen; chenxh@seu.edu.cn

Received 28 May 2018; Revised 2 November 2018; Accepted 18 December 2018; Published 3 January 2019

Guest Editor: Ali Tizghadam

Copyright © 2019 Juan Cheng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Travel time of traffic flow is the basis of traffic guidance. To improve the estimation accuracy, a travel time estimation model based on Random Forests is proposed. 7 influence variables are viewed as candidates in this paper. Data obtained from VISSIM simulation are used to verify the model. Different from other machine learning algorithm as black boxes, Random Forests can provide interpretable results through variable importance. The result of variable importance shows that mean travel time of floating car \bar{t}_f , traffic state parameter X , density of vehicle K_{all} , and median travel time of floating car t_{medf} are important variables affecting travel time of traffic flow; meanwhile other variables also have a certain influence on travel time. Compared with the BP (Back Propagation) neural network model and the quadratic polynomial regression model, the proposed Random Forests model is more accurate, and the variables contained in the model are more abundant.

1. Introduction

Along with economic and populations grow, the number of cars has increased dramatically, causing a series of problems such as traffic congestion, traffic accidents, and environmental pollution [1–3]. To tackle these issues, Intelligent Transportation System (ITS) is applied to the road system. Through the harmonious and close cooperation of people, vehicles, and roads, ITS can improve the efficiency of transportation, ease traffic congestion, improve road network capacity, reduce traffic accidents, lower energy consumption, and decrease environmental pollution. Travel time is the most intuitionistic index to reflect the running condition, which is an important foundation for constructing ITS [4]. Obtaining accurate travel time information, on the one hand, traffic departments improve traffic management decisions; on the other hand, travelers can make better travel choices [5]. Therefore, the accurate travel time of traffic flow is paid more attention by travelers, traffic managers, and scholars. As the basis of ITS, some researchers have conducted special studies on travel time.

Travel time can be achieved directly or indirectly. Direct methods measure travel time using probe vehicle, records at toll stations, tracking of cell phones, and many other

technologies [6, 7]. Indirect methods infer travel time using measured traffic volume, speed, and occupancy in point sensors (e.g., loop detector and video camera) along the vehicle trajectory [8].

Recently, GPS on the vehicles and smartphones carried by occupants of motor vehicles can provide data support for travel time [9]. Therefore, travel time estimation using GPS data has been carried out [10–12]. Over the past few decades, a great number of models for travel time estimation have been developed, including models based on mathematical statistics and models based on artificial intelligence technology.

(1) Models based on mathematical statistics. These models provide interpretable parameters and a simple model structure [7, 13]; e.g., a piecewise truncated quadratic speed trajectory to estimate travel time was proposed by Sun [14]. The speed value can be selected between the highest and lowest, which was selected as the basis of the running condition of the vehicle. The method was more accurate when the vehicle is in the state of transition and congestion. Nevertheless, in the state of free-flow, the advantages of the proposed model were not obvious. Taken the number of single lanes, the speed limitation, and the instantaneous speed as independent variables, a multiple linear regression model based on the floating car was raised by Bobba [15].

The model was applicable during peak and off-peak periods. Yet the road section studied was the section between two signalized intersections (exclude signalized intersections), which was different from most current researches. Choosing different parameters, a linear regression model and a multiple linear regression model were developed by Faria [16]. The multiple linear regression model was more accurate, but the accuracy of the model is limited, only 60%. Using the GPS data with lower frequency, two mathematical models were proposed by Sanaullah [17]. The two models were based on the number of map matched points, connectivity of links, and spatial and temporal travel time components of the link, respectively. The experimental results indicate that vehicle penetration rates, data sampling frequencies, vehicle coverage on the links, and time window lengths all influence the accuracy of link travel time estimation. Zhan [18] used GPS-OD data from New York taxis to estimate travel time of road network segments. The impact of the single lane of the road on the driving vehicle was taken into account. When the road section was wider, the number of lanes may be more, and the single lane may not portray the fineness of the road network. Using the same data, a Bayesian model to estimate short-term travel time was presented by Zhan [19]. However, to reduce the modeling complexity, several assumptions were posed. Because of easy to implementation and low computational effort, models based on mathematical statistics are widely used. However, the accuracy is generally low.

(2) Models based on artificial intelligence technology. These models do not assume any particular model structure of the data but treat it as unknown. Some successful models include Fuzzy reasoning [20], machine learning [21], and the hybrid model [22, 23]. Such as, a three-layer Artificial Neural Network (ANN) model was presented to estimate link travel time by Zheng [24]. In the proposed model, individual probe vehicle's positions, link IDs, timestamps, and speed were used as input information. Compared with Hellinga's model, the ANN model performed quite well under different traffic conditions. However, the ANN model was applied to estimate travel time based on one car with GPS. Using the sparse and large-scale GPS trajectories, Tang [25] presented a tensor-based context-aware approach to estimate personalized travel time. The model was comprised of map matching, travel time tensor construction, context-aware feature extraction, and travel time tensor factorization. The proposed model considers the spatial correlation between different road segments, the deviation between different drivers, the fine-grain temporal correlation between different time slots, and the coarse-grain temporal correlation between recent and historical traffic conditions. A bus travel time prediction model based on SVM was proposed by Reddy [26]. The model used V-Support vector regression as a linear kernel function and used the data collected by public bus equipped with a GPS system to validate. The result showed that accuracy of the model was significantly improved under the condition of high variance. Although these models need large amounts of computation, the high accuracy drives scholars to shift their research focus on artificial intelligence technology method.

In summary, a wide range of models has been developed for travel time estimation. Although these models have

their own advantages, the number of independent variables selected is limited, and the influence of traffic flow parameters on travel time has not been thoroughly considered.

In recent years, data mining and machine learning have gradually come into sight. The development of traffic information acquisition technology (such as data of GPS trajectories) has provided us with a large amount of traffic data, which offer an opportunity to develop a more accurate travel time estimation based on data mining. Compared with traditional parametric models, data mining algorithm can be deeply explored implicit relationships between variables. In view of this, the paper introduces a new data mining technique called Random Forests for travel time estimation. The influence of variables on travel time can be deeply excavated through Random Forests.

The rest of the paper is structured as followed. The next section will give the methodology of Random Forests to build a travel time estimation model followed by Section 3, which describes the data used in this paper. Results and discussions are presented in Section 4. Finally, the conclusions are outlined in Section 5.

2. Methodology of Random Forests

Random Forests is an integrated learning algorithm based on decision tree proposed by Breiman in 2001 [27]. Random Forests is a high-precision algorithm in machine learning, which can overcome the shortcomings of a single prediction or classification model.

2.1. Theory. Random Forests is a combination model consisting of a set of regression decision trees. Equation (1) shows the definition of Random Forests [28].

$$\{h(x, \theta_t), t = 1, 2, \dots, T\} \quad (1)$$

where $h(x, \theta_t)$ is a tree-structured classifier and $\{\theta_t\}$ is independent identically distributed random vectors. x is the independent variable. θ_t is the independent distributed random variable. T represents the number of decision trees.

Use the idea of ensemble learning to take the average of each decision tree as a regression prediction result, which is shown in

$$\bar{h} = \frac{1}{T} \sum_{i=1}^T \{h(x, \theta_t)\} \quad (2)$$

where $h(x, \theta_t)$ is output based on x and θ .

In order to overcome the problem that the decision tree model is not high in accuracy and is prone to overfitting, the idea of bagging and stochastic subspace was introduced in Random Forests [28, 29].

(1) *Bagging.* Bagging is a Bootstrap sampling technique proposed in 1996 [28]. Assuming that S is the original sample and N is the number of samples in S . The probability that each sample in S is not extracted is $(1 - 1/N)^N$.

If $N \rightarrow \infty$, then

$$\lim_{N \rightarrow \infty} \left(1 - \frac{1}{N}\right)^N = \frac{1}{e} \approx 0.368 \quad (3)$$

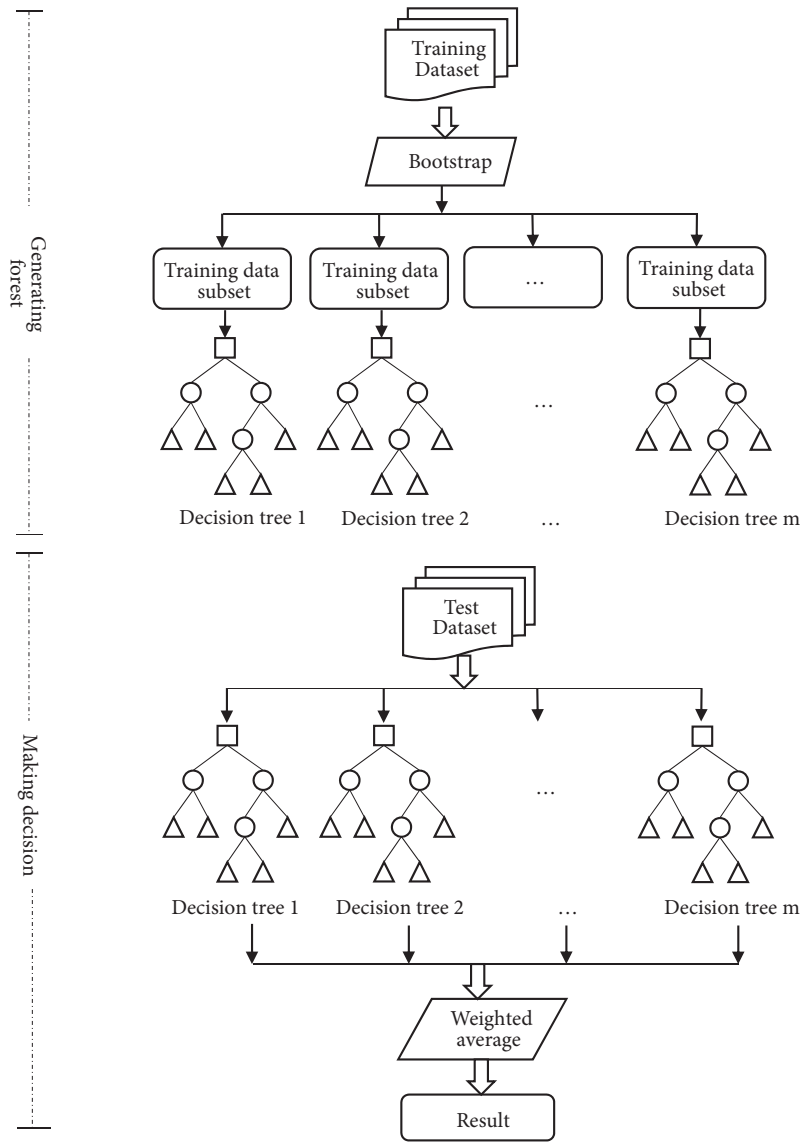


FIGURE 1: The process of establishing a Random Forests.

Equation (3) indicates that about 36.8% of the samples are not extracted each time, which is called OOB (Out-of-Bag) data.

(2) *Stochastic Subspace*. In the process of constructing the regression decision tree, each split node randomly extracts the feature subspace from the total feature space as the candidate feature set of the node and selects the optimal feature for splitting. The method ensures that the feature subsets are not only different among trees, but also the independence and diversity of the tree, and further improve the randomness in node splitting of Random Forests. Determining the stochastic subspace is to choose the number of explanatory variables to be checked for the splitting process.

In Random Forests, the final predictive performance of the model is determined by the number of trees in the forest

(T) and the number of explanatory variables to be checked for the splitting process (m).

Figure 1 is the flowchart of the classifier and the flow of training and testing phases, which shows the process of establishing a Random Forests.

2.2. *Generalization Error*. Generalization error reflects the ability of the model to predict data outside the training set and is an important indicator for judging the quality of the model.

Definition 1. It is assumed that the training sets are extracted from the independent and identically distributed random vectors (X, Y) , and the formed training sets are independent of each other. Then the mean squared error of the output $h(X)$ is $E_{X,Y}(Y - h(X))^2$.

In Random Forests, when there are enough regression decision trees and $h_t(X) = h(X, \theta_t)$, according to the large number theorem, Theorem 2 can be obtained.

Theorem 2. When $t \rightarrow \infty$, mean square generalization error converges on

$$E_{X,Y} (Y - \bar{h}(X, \theta_t))^2 \rightarrow E_{X,Y} (Y - E_{\theta}(X, \theta))^2 = PE^* \quad (4)$$

where θ_t is a random variable of the t -th regression decision subtree. E_{θ} is a mathematical expectation. PE^* is generalization error of Random Forests.

Theorem 2 shows that, with the increase of the regression decision subtree t , Random Forests gradually converges, and generalization error will eventually tend to a limit value. Although Random Forests has been proven not prone to overfitting in mathematics [27], in the actual application process, the parameters of the Random Forests are optimized by experiments to further avoid overfitting.

3. Data

3.1. Traffic Simulation Software. To collect enough data for training and testing, traffic simulation software is used. Traffic simulation software is widely used in the study of traffic planning and traffic flow. The microscopic traffic simulation software can describe the road network and simulate the traffic flow through different models. Many types of traffic simulation software can be used to collect travel time data [30, 31].

(1) *VISSIM*. VISSIM is a microscopic traffic simulation software developed by PTV of Germany, which is a simulation system based on traffic behavior model. It uses a discrete, random, microscopic model with a time step of 0.1s. The longitudinal movement of the vehicle adopts the psychophysical car-following model proposed by Professor Wiedemann, and the lane-changing behavior of the vehicle adopts a rule-based algorithm. After an open COM interface, VISSIM has a good secondary development capability.

(2) *CORSIM*. CORSIM is developed by the US Federal Highway Administration (FHWA) and consists of two models, FRESIM and NETSIM. FRESIM is mainly used for the simulation of highways and expressways, while NETSIM is used for the simulation of urban road networks. It has lane change and car-following model simulation module and simulates the state of traffic flow in the road network with 1s simulation step. The software has functions such as analog timing, dynamic filter control, and cooperative filtering control. However, CORSIM lacks an allocation algorithm and it is difficult to evaluate the traffic volume transfer caused by ramp control, accidents, and travel information.

(3) *PARAMICS*. Developed by British Quadstone, PARAMICS can be applied to traffic simulation at different levels, from a single road network to a large-scale urban road network. PARAMICS supports multiuser parallel computing

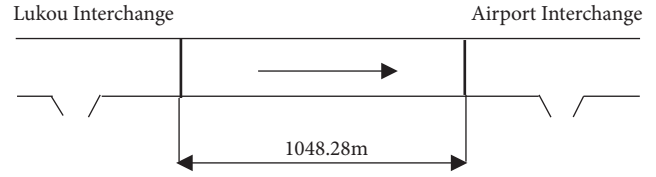


FIGURE 2: The study area.

with a powerful application interface. However, PARAMICS lacks a model of mixed traffic and complex traffic flow.

(4) *SimTraffic*. SimTraffic is originally developed as transportation software for signal optimization timing and traffic model building. With the development of traffic simulation technology, SimTraffic has gradually developed into mature and fully functional microscopic traffic simulation software. It adds ramps, roundabouts, and highway modeling tools based on the original functions. Nevertheless, SimTraffic does not have a dedicated lane, as well as bus and car parking spots.

Through the above description, it can be found that the traffic simulation software has its own advantages and disadvantages. However, VISSIM has the ability to propose separate file output parameters, high traffic description accuracy, and simulated traffic has diverse characteristics. Therefore, data produced by VISSIM simulation software are used to verify the proposed travel time estimation model in this paper.

3.2. The Source of Data

3.2.1. Selection of the Simulation Section. Nanjing Airport freeway between the Airport Interchange and Lukou Interchange with the length of 1048.28m and 4 lanes in one direction is selected as the research area. Time detectors are set at both ends of the selected freeway section. The route diagram is presented in Figure 2.

3.2.2. Determination of the Simulation Parameters

(1) *Simulation Traffic*. The VISSIM simulation software is calibrated according to actual hourly traffic flow in Nanjing Airport freeway from Nanjing to Airport investigated by airport toll station at 9:00-15:00 on August 22, 2017. Since the real traffic flow does not include congestion, in order to cover the state of free-flow, transition, and congestion in the freeway, the traffic flow increased 600Veh/h from the real measured value of the previous period during 15:00-17:00, which reflect the state of congestion. Only increasing the number of vehicles does not necessarily result in congestion. However, based on the state of transition, the authors guarantee that all variables are constant and continue to increase the traffic flow to characterize the state of congestion. The input traffic flow is shown in Table 1.

(2) *Vehicle Type*. The user-defined taxi type is 1, and the vehicle color is blue; the truck type is 2, and the vehicle color is yellow; the bus type is 3, and the vehicle color is blue; the

TABLE 1: The input traffic flow.

time segments (s)	9:00-9:30	9:30-10:00	10:00-10:30	10:30-11:00
Simulation time segments (s)	0-1800	1800-3600	3600-5400	5400-7200
traffic flow (veh/h)	800	1200	1600	2000
time segments (s)	11:00-11:30	11:30-12:00	12:00-12:30	12:30-13:00
Simulation time segments (s)	7200-9000	9000-10800	10800-12600	12600-14400
traffic flow (veh/h)	2400	2600	2800	3000
time segments (s)	13:00-13:30	13:30-14:00	14:00-14:30	14:30-15:00
Simulation time segments (s)	1400-16200	16200-18000	18000-19800	19800-21600
traffic flow (veh/h)	3600	4200	4800	5400
time segments (s)	15:00-15:30	15:30-16:00	16:00-16:30	16:30-17:00
Simulation time segments (s)	21600-23400	23400-25200	25200-27000	27000-28800
traffic flow (veh/h)	6000	6600	7200	7800

car type is 4, and the vehicle color is red. The user-defined taxi is chosen as the floating car in this paper.

(3) *Speed Distribution.* On the freeway, the expected speed of car, truck, and bus is 120,100 and 100 km/h. The speed distribution of cars, trucks, buses, and taxis is shown in Figure 3.

(4) *Vehicle Proportion.* Through investigation, the vehicle proportion on the airport freeway section is car: truck: bus: taxi = 0.42:0.12:0.26:0.2.

(5) *Time Detector.* In the freeway section, time detectors are set up to collect travel time of the individual floating car and travel time of the traffic flow. The mean and median values of travel time are calculated by the collection travel time of an individual floating car.

3.3. Design of Experimental Scheme. In the process of experiment, the dynamic changing process of the freeway traffic flow was simulated by changing the input traffic flow, including the state of free-flow, transition, and congestion.

Using different random seed number, the experiment simulated 133 times and the simulation time was 28800s. At last, 133 sets of data were obtained, representing 133 days' data of 9:00-17:00.

Travel time was obtained at the sampling interval of 300 seconds. At the same time, travel time of the floating cars was acquired at the sampling interval of 1 second.

3.4. Variables of the Model

3.4.1. Traffic State Parameter. In the Highway Capacity Manual [32], traffic state of the freeway was divided into six levels (namely A to F) according to the average speed and density. As we all know, speed, density, and traffic flow are three basic parameters, which are interrelated. If values for two of these parameters are known, the third can be computed. The standard of traffic state classification of a freeway is shown in Table 2.

In this paper, traffic state parameter refer to the standard of traffic state classification of a freeway, let $x = 1$ to 6 for representing the traffic state A to F of the freeway respectively. The paper combined existing traffic state levels and described the freeway at a lower level. Therefore, traffic state of the freeway was divided into three categories. The state of free-flow includes level A and B, namely $x_f = 1, 2$; the state of transition includes level C and D, namely $x_t = 3, 4$; the state of congestion includes level E and F, namely $x_c = 5, 6$, which is presented in Table 3. The traffic parameter is $X = \{x_f, x_t, x_c\}$.

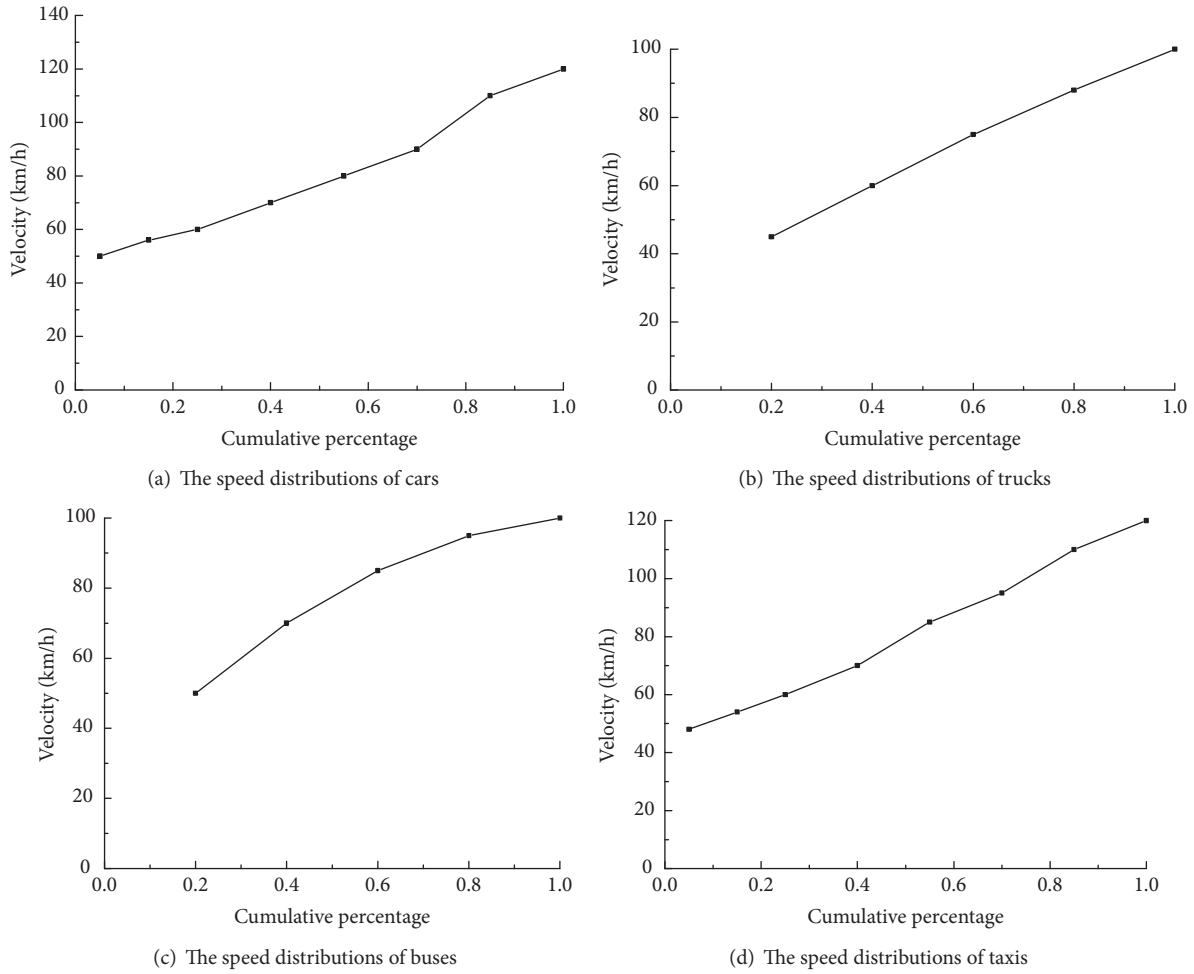


FIGURE 3: The speed distributions.

TABLE 2: The standard of traffic state classification of a freeway [32].

Traffic state	Density Range		Design speed (mi/h)		
	(pc/mi/ln)	Speed (mi/h)	75	65	traffic flow (pc/h/ln)
A	11	75	820	65	710
B	18	74.8	1350	65	1170
C	26	70.6	1830	64.6	1680
D	35	62.2	2170	59.7	2090
E	45	53.3	2400	52.2	2350
F	>45	<53.3	>2400	<52.2	>2350

Note: in order to keep the data neat, the unit used pc/mi/ln and mi/h in Table 2. Table 2 can change into pc/km/ln and km/h by 1mi = 1.609km. The number in Table 2 is the maximum value of each level.

3.4.2. *Travel Time Calculation of the Floating Car.* On any freeway section Δx_n , there are s floating cars within the time interval t_i to t_{i+1} , which is shown in Figure 4.

Assuming that travel time of each floating car on the freeway section is t_g ($g = 1, 2, 3 \dots s$), since the mean value expressed as \bar{t} and the median value denoted by t_{men} can represent the general level of the whole data. The travel time

of the floating car is being calculated by the mean value and median value respectively.

3.4.3. *Variables of the Model.* In the process of VISSIM simulation, 14 traffic variables can be obtained, that is, number of floating car N_f , occupancy of floating car R_f , number of vehicle N_{all} , occupancy of vehicle R_{all} , density of

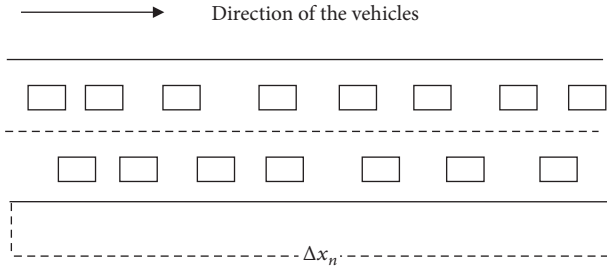


FIGURE 4: The distribution of floating cars on the freeway.

TABLE 3: The table of traffic state parameter.

traffic state	traffic state parameter	traffic state of the paper
A	1	free-flow
B	2	
C	3	transition
D	4	
E	5	congestion
F	6	

floating car K_f , speed of floating car V_f , traffic flow of floating car Q_f , density of vehicle K_{all} , speed of vehicle V_{all} , traffic flow of vehicle Q_{all} , ratio of floating car $Ratio_f$, travel time of floating car (mean value \bar{t}_f and median value t_{men}), and traffic state parameter X .

4. Results and Discussions

Using the data obtained from VISSIM and the variables discussed above, the Random Forests model for travel time estimation was established.

SPM 8.2 data mining software developed by Salford Systems was used to establish the Random Forests model [33], although Random Forests can use the OOB error to evaluate the model. However, in order to compare with other models, in this paper 133 sets of data were trained and validated different models in two scenarios. Total data of 132 simulations were selected as training data, and the 5th simulation data were selected as test data, which used to compare with the mathematical statistics model. Meanwhile, in order to contrast with machine learning model, data of 133 days were divided into two data sets, in which 27-133 days of data were used as training data sets and 1-26 days of data were used as test data sets.

Mean Square Error (MSE), Mean Absolute Deviation (MAD), and Relative Error (RE) were selected as evaluation criteria.

$$MSE = \frac{1}{n} \sum_{i=1}^n \left(y_i - \hat{y}_i \right)^2 \quad (5)$$

$$MAD = \frac{1}{n} \sum_{i=1}^n \left| y_i - \hat{y}_i \right| \quad (6)$$

$$RE = \frac{y_i - \hat{y}_i}{y_i} \quad (7)$$

where in (5)-(7) n is the total number of samples, y_i is the real value of travel time, and \hat{y}_i is the estimation value of travel time.

4.1. Parameter Determination. There are 2 parameters to be ascertained in Random Forests, namely, the number of trees in the forest (T) and the number of explanatory variables to be checked for the splitting process (m).

(1) *The Number of Trees in the Forest (T).* In Random Forests, decision trees are not pruned. It is demonstrated that increasing the number of trees would not increase the precision but brings the computational burden. However insufficient trees are generated, the calculated variables importance may not be accurate enough. The number of trees in the forest was determined by 10-fold cross-validation. Table 4 shows the 10-fold cross-validation errors with a different number of trees in the forest.

It can be seen from Table 4 that the number of trees was 500 and 600 with the same minimum test error. The fewer the trees are, the smaller the computational burden is; therefore, the number of trees was 500 in the Random Forests model.

(2) *The Number of Explanatory Variables to Be Checked for the Splitting Process (m).* In Random Forests, only a subset of independent variables is checked to find the best splits, which makes the forest development more efficient. Beriman [27] has shown that randomly selecting a subset of independent variables to find the best split makes the process faster and leads to accurate results. Ghasri [34] adopted the square root of the number of independent variables in each split. There are other methods, such as twice the square root and half the square root of the number of independent variables in each split. When the number of explanatory variables to be checked for the splitting process is determined by 10-fold cross-validation error, the effect is not obvious. Random Forests can use OOB (Out-of-Bag) error estimates as unbiased estimates of generalization error without running a cross-validation procedure to measure the Random Forests model [28, 34–36]. Therefore, the OOB error was used to determine the number of explanatory variables to be checked for the splitting process. In this paper, the OOB errors are obtained using a different number of explanatory variables to be checked for the splitting process, respectively, and finally, choose the number of independent variables with the smallest value of OOB error. Table 5 shows the OOB errors with a different number of independent variables in each split.

Finally, the number of explanatory variables to be checked for the splitting process was 3 in the Random Forests model.

4.2. Variable Importance. Using the training data to train the model, the order of variables importance can be gained.

TABLE 4: 10-fold cross-validation errors with a different number of trees in the forest.

trees	100		200		300	
Data sets	Learn	Test	Learn	Test	Learn	Test
RMSE	2.2149	2.6889	2.2153	2.6857	2.2179	2.6789
MAD	0.9228	1.1243	0.9189	1.2112	0.9163	1.1165
trees	400		500		600	
Data sets	Learn	Test	Learn	Test	Learn	Test
RMSE	2.2126	2.6752	2.2144	2.6703	2.2144	2.6703
MAD	0.9141	1.1163	0.9152	1.1151	0.9152	1.1151

TABLE 5: The OOB errors with a different number of independent variables in each split.

m	1	2	3	4	5	6	7
OOB errors	10.283	7.219	6.342	6.936	6.591	6.474	7.230
m	8	9	10	11	12	13	14
OOB errors	7.306	7.235	7.300	7.260	7.282	7.395	7.302

TABLE 6: Variable importance.

Variable	Variable importance
mean travel time of floating car \bar{t}_f	58.98
density of vehicle K_{all}	35.58
traffic state parameter X	29.95
median travel time of floating car t_{menf}	15.72
density of floating car K_f	5.39
speed of vehicle V_{all}	4.74
occupancy of vehicle R_{all}	2.75
speed of floating car V_f	1.68
traffic flow of vehicle Q_{all}	0.22
number of vehicle N_{all}	0.22
occupancy of floating car R_f	0.21
traffic flow of floating car Q_f	0.15
number of floating car N_f	0.07
ratio of floating car $Ratio_f$	0.05

Variable importance explains the influence of independent variables on the dependent variable. The higher the value of the variable importance is, the stronger the influence on the model is. Variable importance is shown in Table 6.

It can be seen from Table 6 that mean travel time of floating car \bar{t}_f , density of vehicle K_{all} , traffic state parameter X , and median travel time of floating car t_{menf} are the important factors, which are much greater than other variables. It is indicated that travel time of traffic flow is closely related to travel time of floating car, density of vehicle and traffic state parameter.

4.3. Filtering Feature Variables. As can be seen from Table 6, in the established Random Forests model, much variable importance has low values, such as the ratio of floating car, indicating that there are some redundant variables in the model and the variables need to be screened. The literature

[37] uses the variable importance obtained by the model to filter the feature variables, which is presented as follows:

- (1) Create a Random Forests model using a set of feature variables containing n variables and rank the variable importance of the n feature variables in descending order.
- (2) Delete the variable with the lowest variable importance among the n feature variables, and get the feature variable set containing $n-1$ variables.
- (3) Create a Random Forests model using a set of feature variables containing $n-1$ variables and rank the variable importance of the $n-1$ feature variables in descending order.
- (4) Delete the variable with the lowest variable importance among the $n-1$ feature variables, and get the feature variable set containing $n-2$ variables.
- (5) Repeat steps (3) and (4) until there is one remaining feature variable.
- (6) The Random Forests models are established containing $n, n-1, n-2 \dots 1$ variables. OOB errors are ranked in order, and the Random Forests model with the smallest OOB error and the feature variable set is selected.

According to the method described above, a set of feature variables and a Random Forests model containing 7 feature variables are obtained. The training result of the model is shown in Figure 5 and the variable importance is shown in Table 7.

From Figure 5, reducing the number of feature variables does not reduce the performance of the model; however, the OOB error decreases from 6.342 to 5.586. It can be observed from Table 7 that mean travel time of floating car \bar{t}_f , traffic state parameter X , density of vehicle K_{all} , and median travel time of floating car t_{menf} are still the most important factors, which is much greater than the other three variables.

In all the 7 variables, \bar{t}_f and t_{menf} enter the model at the same time, but the variable importance of \bar{t}_f is much larger

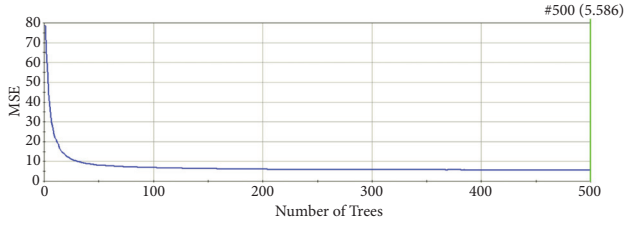


FIGURE 5: The result of Random Forests.

TABLE 7: Variable importance after filtering feature variables.

Variable	Variable importance
mean travel time of floating car \bar{t}_f	113.69
traffic state parameter X	45.07
density of vehicle K_{all}	41.38
median travel time of floating car t_{menf}	13.46
speed of vehicle V_{all}	4.88
density of floating car K_f	1.17
speed of floating car V_f	0.94

than t_{menf} . In the previous research, due to the correlation of variables, the two parameters were generally not included in the model at the same time. However, the mean value and median value can represent the general level of the whole data with different statistical significance. Therefore, the Random Forests model uses both \bar{t}_f and t_{menf} as variables to take full advantage of different variables.

Density of vehicle K_{all} and density of floating car K_f have different effects on the Random Forests model. Density is the most important parameter of traffic flow, and it is an evaluation index of traffic demand. The higher the density is, the slower the speed is and the longer the travel time is. Density is an important indicator that affects travel time. K_{all} is much more important than K_f , which is simple to understand. The paper uses travel time of the floating car to calculate travel time of traffic flow but K_{all} represents the condition of all vehicles, which is a more intuitive reflection on travel time of traffic flow.

Speed of vehicle V_{all} and speed of floating car V_f have an influence on the estimated travel time because speed is the most intuitive reflection of travel time. The variable importance value of V_f is the lowest (0.94), but it is also an important factor affecting travel time of traffic flow. The reason is that the travel time estimation model is based on travel time of floating car and speed of floating car is closely related to travel time of floating car.

Traffic state parameter X is the second most important influence variable in the Random Forests model. As a newly introduced parameter in this paper, X is an intuitive indicator that directly reflects traffic states.

To sum up, the paper uses travel time of floating car to reflect travel time of traffic flow. Travel time of floating car (both \bar{t}_f and t_{menf}) is an indispensable factor in the Random Forests model. Speed and density are the most intuitive reflection of travel time; therefore K_{all} , K_f , V_{all} , and V_f are

the selected influence variable of the Random Forests model. The variable of X is selected in the Random Forests model because it directly reflects the traffic states.

4.4. Accuracy of the Established Model. To test the accuracy of the model presented in this paper, a quadratic polynomial regression model with different states was established according to the method of [38]. Consistent with the Random Forests model, the total data of 132 simulations were selected for quadratic polynomial regression and the 5th simulation data were selected as validation data. The quadratic polynomial regression model is provided in

$$T_{flow} = \begin{cases} 0.0275\bar{T}_f^2 - 1.2203\bar{T}_f + 45.5727 & \text{free-flow} \\ 0.0058T_{ment}^2 + 0.0902T_{ment} + 37.1768 & \text{transition} \\ -0.0034\bar{T}_c^2 + 2.0308\bar{T}_c - 65.0968 & \text{congestion} \end{cases} \quad (8)$$

where T_{flow} is travel time of traffic flow. \bar{T}_f is mean travel time of floating car in the state of free-flow. T_{ment} is median travel time of floating car in the state of transition. \bar{T}_c is mean travel time of floating car in the state of congestion.

Equation (8) is a regression model with different states. Although there are no separate states to establish the Random Forests model, the introduced traffic state parameter X can distinguish different traffic states. The errors of different models are presented in Table 8.

Meanwhile, BP (Back Propagation) neural network model was also established by using the data of 27-133 days of data as training data sets. BP neural network [39] is a multilayer feedforward network trained by error inverse propagation algorithm, which was proposed by a team of scientists led by Rumelhart and McClland in 1986.

The topology of BP neural network includes input layer, hidden layer, and output layer, which are divided into information forward propagation and error back propagation [40]. BP neural network model with a three-layer feedforward Perceptron algorithm is used to estimate travel time. Figure 6 is the network structure of the three-layer BP neural network model designed in this paper.

In order to a fair comparison with the Random Forests model, the input variable in the BP neural network model is the selected variables in Section 4.3, and the output variable is the estimated travel time. The function of the hidden layer is logistic in the BP neural network. The network structure is 7-6-1, that is, the number of input layer nodes is 7, the number of hidden layer nodes is 6, and the number of output layer nodes is 1. Then the model was tested using 1-26 days of data sets. The training and test errors of different models are shown in Table 9.

Figure 7 is travel time obtained by different models. Figure 8 shows the comparison between travel time of the 5th day in the test data sets (real travel time) and travel time obtained with various models.

Several conclusions can be drawn based on Tables 8-9 and Figures 7-8.

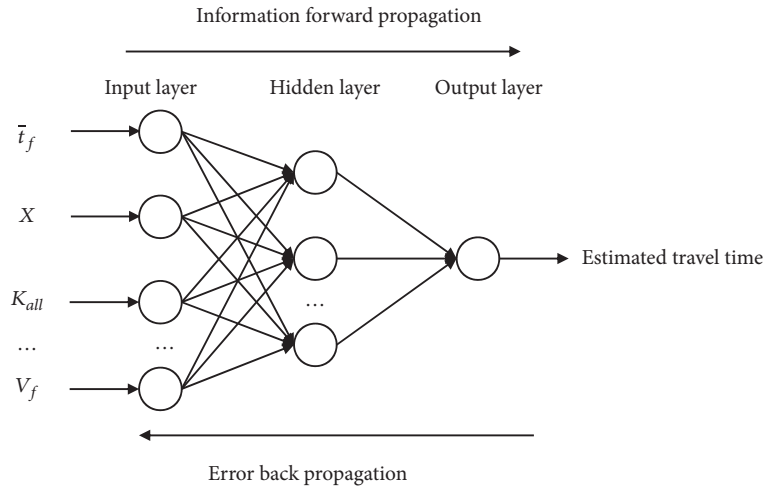


FIGURE 6: The network structure of the three-layer BP neural network model.

TABLE 8: The MAD of Random Forests model and reference [38] model.

MAD	Random Forests model	model of reference [38]
free-flow	0.5295	0.6556
transition	0.6640	1.2326
congestion	1.7573	6.8407
average	0.8047	5.5113

TABLE 9: The MAD of Random Forests model and BP neural network model.

Data set	Traffic state	Random Forests model	BP Neural Network model
Training Data	free-flow	0.5264	0.5414
	transition	0.5768	0.6003
	congestion	1.2856	1.8827
	average	0.8043	0.8987
Test Data	free-flow	0.5405	0.5898
	transition	0.6640	0.8375
	congestion	1.3024	1.9009
	average	0.8139	0.9076

Firstly, it can be seen from Table 8 that the accuracy of the Random Forests model is much greater than that of the quadratic polynomial regression model. In addition to travel time (mean and median) of floating car, the proposed model has selected another six variables, which indicate that Random Forests are not sensitive to the interaction between variables. Therefore, the Random Forests model can choose a richer impact variable.

Secondly, Table 9 shows the comparison between the Random Forests model and the BP neural network model; it is found that the error of the Random Forests model is generally less than the BP neural network model in both training data sets and test data sets. The reason may be different from the machine learning algorithm as black boxes (such as BP neural network and SVM); Random Forests has capabilities of data

mining. The relationship between variables can be deeply exploited through Random Forests.

Thirdly, it revealed that, in Table 9, when traffic flow is operating in the state of free-flow with high speed, travel time obtained by the two models is close to the real value in both the two data sets. While in the state of transition and congestion with the lower speed, error of the proposed model is obviously less than the BP neural network model in both two data sets. It is shown that the model proposed in this paper has more advantages in the state of transition and congestion.

Fourthly, as indicated in Figures 7-8 that travel time obtained in this paper is consistent with the real travel time, which indicates that the proposed Random Forests model is effective.

TABLE 10: The variables entering the model after eliminating multicollinearity.

traffic state	variables
free-flow	mean travel time of floating car \bar{T}_f , traffic state parameter X_f , occupancy of vehicle R_{allf} , number of floating car N_f
transition	median travel time of floating car T_{ment} , traffic state parameter X_t , occupancy of vehicle R_{allt} , number of floating car N_t
congestion	mean travel time of floating car \bar{T}_c , traffic state parameter X_c , occupancy of vehicle R_{allc} , number of floating car N_c

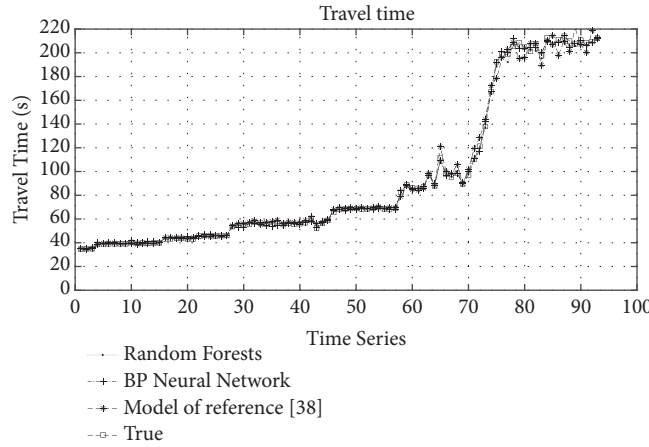


FIGURE 7: The estimated travel time.

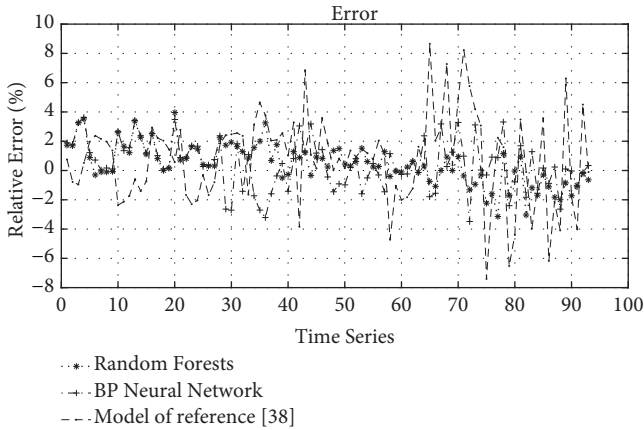


FIGURE 8: The relative error of the models.

In addition, the paper established a multiple linear regression model for different states using the 14 variables mentioned in Section 3.4.3. Before establishing a regression model, the multicollinearity between independent variables is tested firstly. Multicollinearity (collinearity for short) proposed by Freund [41] refers to a precise correlation or a high degree of correlation between the variables in the linear regression model, which makes the model difficult to estimate accurately. After eliminating multicollinearity, the variables entering the model are shown in Table 10.

It can be seen from Table 10 that, due to the multicollinearity of variables, when the regression model is

established, only 4 variables are selected, and some of the variables that affect travel time are ignored, such as speed, density, etc. However, Random Forests is not affected by multicollinearity of variables. The relationship between travel time and variables can be deeply excavated through Random Forests.

5. Conclusion

In this paper, Random Forests is proposed for travel time estimation, which is a hotspot algorithm in machine learning and can deeply excavate the complex relationships between variables. The proposed model is established with 7 variables, namely, mean travel time of floating car \bar{t}_f , traffic state parameter X , density of vehicle K_{all} , median travel time of floating car t_{menf} , speed of vehicle V_{all} , density of floating car K_f , and speed of floating car V_f . Using different random seed number, the experiment simulates 133 times with VISSIM simulation software. Total data of 132 simulations are selected as training data, and the 5th simulation data are selected as test data, which used to compare with the quadratic polynomial regression model. Meanwhile, data of 133 days are divided into two data sets, in which 27-133 days of data are used as training data sets and 1-26 days of data are used as test data sets in order to contrast with the BP neural network model. Comparison results show that the Random Forests model is more accurate than the quadratic polynomial regression model and the BP neural network model. The included variables are more abundant in the Random Forests model.

However, data are obtained by VISSIM, which limited the diversity of data. In future research, the variables of weather, characters of drivers, and other variables which affect travel time will be considered in the Random Forests model.

Data Availability

The data of the study was simulated by VISSIM and can be obtained up request.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study is supported by the National Natural Science Foundation of China under Grants nos. 51478114 and 51778136.

References

- [1] E. Jenelius and H. N. Koutsopoulos, "Travel time estimation for urban road networks using low frequency probe vehicle data," *Transportation Research Part B: Methodological*, vol. 53, pp. 64–81, 2013.
- [2] J. F. Xi, Z. H. Zhao, W. Li, and Q. Wang, "A traffic accident causation analysis method based on AHP-apriori," *Procedia Engineering*, vol. 137, pp. 680–687, 2016.
- [3] T. Yi and B. M. Williams, "Dynamic traffic flow model for travel time estimation," *Transportation Research Record*, vol. 2526, pp. 70–78, 2015.
- [4] W.-H. Lee, S.-S. Tseng, and S.-H. Tsai, "A knowledge based real-time travel time prediction system for urban network," *Expert Systems with Applications*, vol. 36, no. 3, pp. 4239–4247, 2009.
- [5] D. M. Miranda and S. V. Conceição, "The vehicle routing problem with hard time windows and stochastic travel and service time," *Expert Systems with Applications*, vol. 64, pp. 104–116, 2016.
- [6] D. Woodard, G. Nogin, P. Koch, D. Racz, M. Goldszmidt, and E. Horvitz, "Predicting travel time reliability using mobile phone GPS data," *Transportation Research Part C: Emerging Technologies*, vol. 75, pp. 30–44, 2017.
- [7] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos, "Non-parametric estimation of route travel time distributions from low-frequency floating car data," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 343–362, 2015.
- [8] L. Lu, J. Wang, Z. He, and C.-Y. Chan, "Real-time estimation of freeway travel time with recurrent congestion based on sparse detector data," *IET Intelligent Transport Systems*, vol. 12, no. 1, pp. 2–11, 2018.
- [9] M. Rahmani, H. N. Koutsopoulos, and E. Jenelius, "Travel time estimation from sparse floating car data with consistent path inference: A fixed point approach," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 628–643, 2017.
- [10] C.-H. Wu, J.-M. Ho, and D. T. Lee, "Travel-time prediction with support vector regression," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 276–281, 2004.
- [11] C. A. Quiroga and D. Bullock, "Travel time studies with global positioning and geographic information systems: an integrated methodology," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 1-2, pp. 101–127, 1998.
- [12] J. Tang, F. Liu, Y. Wang, and H. Wang, "Uncovering urban human mobility from large scale taxi GPS data," *Physica A: Statistical Mechanics and Its Applications*, vol. 438, pp. 140–153, 2015.
- [13] Z. Ma, H. N. Koutsopoulos, L. Ferreira, and M. Mesbah, "Estimation of trip travel time distribution using a generalized Markov chain approach," *Transportation Research Part C: Emerging Technologies*, vol. 74, pp. 1–21, 2017.
- [14] L. Sun, J. Yang, and H. Mahmassani, "Travel time estimation based on piecewise truncated quadratic speed trajectory," *Transportation Research Part A: Policy and Practice*, vol. 42, no. 1, pp. 173–186, 2008.
- [15] R. Bobba, *Predicting Speeds on Urban Streets Using Real Time GPS Data*, University of Texas at Arlington, Arlington, Va, USA, 2002.
- [16] D. Faria, "A framework to transform real-time GPS derived from transit vehicles to determine speed-flow," 2003.
- [17] I. Sanaullah, M. Quddus, and M. Enoch, "Developing travel time estimation methods using sparse GPS data," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 20, no. 6, pp. 532–544, 2016.
- [18] X. Zhan, S. Hasan, S. V. Ukkusuri, and C. Kamga, "Urban link travel time estimation using large-scale taxi data with partial information," *Transportation Research Part C: Emerging Technologies*, vol. 33, pp. 37–49, 2013.
- [19] X. Zhan, S. V. Ukkusuri, and C. Yang, "A Bayesian mixture model for short-term average link travel time estimation using large-scale limited information trip-based data," *Automation in Construction*, vol. 72, pp. 237–246, 2016.
- [20] Y. Li and M. McDonald, "Link travel time estimation using single GPS equipped probe vehicle," in *Proceedings of the 5th IEEE International Conference on Intelligent Transportation Systems, ITSC 2002*, pp. 932–937, Singapore, September 2002.
- [21] A. Hofleitner, R. Herring, and A. Bayen, "Arterial travel time forecast with streaming data: A hybrid approach of flow modeling and machine learning," *Transportation Research Part B: Methodological*, vol. 46, no. 9, pp. 1097–1122, 2012.
- [22] S. Lee, B. Lee, and Y. Yang, "Estimation of link speed using pattern classification of GPS probe car data," in *Proceedings of the International Conference on Computational Science and Its Applications*, vol. 3981, pp. 495–504, 2006.
- [23] B. A. Kumar, L. Vanajakshi, and S. C. Subramanian, "A hybrid model based method for bus travel time estimation," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 22, no. 5, pp. 390–406, 2017.
- [24] F. Zheng and H. van Zuylen, "Urban link travel time estimation based on sparse probe vehicle data," *Transportation Research Part C: Emerging Technologies*, vol. 31, pp. 145–157, 2013.
- [25] K. Tang, S. Chen, and A. J. Khattak, "Personalized travel time estimation for urban road networks: A tensor-based context-aware approach," *Expert Systems with Applications*, vol. 103, pp. 118–132, 2018.
- [26] K. K. Reddy, B. A. Kumar, and L. Vanajakshi, "Bus travel time prediction under high variability conditions," *Current Science*, vol. 111, no. 4, pp. 700–711, 2016.
- [27] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [28] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

- [29] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.
- [30] A. Louati, S. Darmoul, S. Elkosantini, and L. ben Said, "An artificial immune network to control interrupted flow at a signalized intersection," *Information Sciences*, vol. 433/434, pp. 70–95, 2018.
- [31] J. Wu, M. Brackstone, and M. McDonald, "The validation of a microscopic simulation model: a methodological case study," *Transportation Research Part C: Emerging Technologies*, vol. 11, no. 6, pp. 463–479, 2003.
- [32] National Research Council, *HCM2010: Highway Capacity Manual*, Transportation Research Board, 5th edition, 2010.
- [33] M. Gualtieri, C. A. Rowan, and K. TaKeaways, "The Forrester Wave™: Big Data Predictive Analytics Solutions, Q1," *Forrester Research*, 2013.
- [34] M. Ghasri, T. Hossein Rashidi, and S. T. Waller, "Developing a disaggregate travel demand system of models using data mining techniques," *Transportation Research Part A: Policy and Practice*, vol. 105, pp. 138–153, 2017.
- [35] L. Cheng, X. Chen, J. De Vos, X. Lai, and F. Witlox, "Applying a random forest method approach to model travel mode choice behavior," *Travel Behaviour and Society*, vol. 14, pp. 1–10, 2019.
- [36] J. Bao, P. Liu, X. Qin, and H. Zhou, "Understanding the effects of trip patterns on spatially aggregated crashes with large-scale taxi GPS data," *Accident Analysis & Prevention*, vol. 120, pp. 281–294, 2018.
- [37] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern Recognition Letters*, vol. 31, no. 14, pp. 2225–2236, 2010.
- [38] J. W. Li, *Estimation and Prediction of Link Travel Time for Urban Trunk and Secondary Stress*, Jilin University, 2012.
- [39] Y.-K. Liu, F. Xie, C.-L. Xie, M.-J. Peng, G.-H. Wu, and H. Xia, "Prediction of time series of NPP operating parameters using dynamic model based on BP neural network," *Annals of Nuclear Energy*, vol. 85, pp. 566–575, 2015.
- [40] X. Yu, J. Han, L. Shi, Y. Wang, and Y. Zhao, "Application of a BP neural network in predicting destroyed floor depth caused by underground pressure," *Environmental Earth Sciences*, vol. 76, no. 15, Article ID 535, 2017.
- [41] R. J. Freund and R. C. Littell, *SAS System for Regression*, SAS Publishing, 3rd edition, 2000.

Research Article

Evaluation and Application of Urban Traffic Signal Optimizing Control Strategy Based on Reinforcement Learning

Yizhe Wang ^{1,2}, Xiaoguang Yang ^{1,2}, Yangdong Liu ^{1,3} and Hailun Liang ^{1,2}

¹Key Laboratory of Road and Traffic Engineering of the Ministry of Education, Tongji University,
4800 Cao'an Road, Shanghai 201804, China

²Intelligent Transportation System Research Center of Tongji University, 4801 Cao'an Road, Shanghai 201804, China

³Hangzhou Hikvision Digital Technology Co., Ltd., No. 555 Qianmo Road, Binjiang District, Hangzhou 310052, China

Correspondence should be addressed to Xiaoguang Yang; yangxg@tongji.edu.cn

Received 6 August 2018; Revised 3 November 2018; Accepted 9 December 2018; Published 26 December 2018

Guest Editor: Hamzeh Khazaei

Copyright © 2018 Yizhe Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Reinforcement learning method has a self-learning ability in complex multidimensional space because it does not need accurate mathematical model and due to the low requirement for prior knowledge of the environment. The single intersection, arterial lines, and regional road network of a group of multiple intersections are taken as the research object on the paper. Based on the three key parameters of cycle, arterial coordination offset, and green split, a set of hierarchical control algorithms based on reinforcement learning is constructed to optimize and improve the current signal timing scheme. However, the traffic signal optimization strategy based on reinforcement learning is suitable for complex traffic environments (high flows and multiple intersections), and the effects of which are better than the current optimization methods in the conditions of high flows in single intersections, arteries, and regional multi-intersection. In a word, the problem of insufficient traffic signal control capability is studied, and the hierarchical control algorithm based on reinforcement learning is applied to traffic signal control, so as to provide new ideas and methods for traffic signal control theory.

1. Introduction

Traffic congestion has become a world-concerned problem all over the world. With the increasing number of vehicles, traffic congestion has deeply affected people's daily life and the development of social economy. Traffic control is one of the most important technological means of regulating traffic flow, improving obstruction, and improving its safety and even energy conservation and emission reduction. At present, traffic signal control problem not only has a long-time congestion phenomenon at peak time, but also has obvious ability of grooming in peak time. In order to ease the traffic pressure, rational analysis and control are considered as an important tool. Its progress and development are always keeping pace with the times, accompanied by information technology, computer technology, and system science.

According to the system's ability to adapt to the environment and the level of intelligent decision-making, Gartner proposed the evolution of urban transport control system

development level in 1996 [1]. The first-generation self-adaptive control system adopts the multi-time timing control of fine division of period, or completely isolated self-adaptive control, to realize the simple regulation of traffic flow. The second-generation traffic signal control system dynamically adjusts the parameters of the signal timing scheme (cycle length, split, offset). Typical second-generation control systems include SCATS [2] and SCOOT [3]. The UK Transport Research Laboratory had a worldwide reputation for contributions to the field of traffic signal control, especially as originators of the TRANSYT and SCOOT signal coordination methods [4]. The third-generation control system uses similar idea to the second-generation to dynamically adjust the signal timing parameter in response to the fluctuation of the time-varying traffic flow at the intersection. HK Lo and HF Chow investigated the relationship of finer resolutions and larger errors in adaptive traffic control system through an extensive simulation of scenarios in Hong Kong with a recently developed dynamic traffic control model, DISCO

[5]. Aboudolas K and Papageorgiou M tested a preliminary simulation-based investigation of the signal control problem for a large-scale urban road network using store-and-forward modeling demonstrating the comparative efficiency and real-time feasibility of the developed signal control methods [6]. The fourth-generation traffic signal control system is an integrated traffic management and control system. Meneguzzer C presented two alternative deterministic, discrete-time DP models of the interaction between signal control and route choice, which are proposed and compared with the conventional iterative optimization and assignment (IOA) method for network traffic signal setting [7]. The fifth-generation traffic signal control system is based on the abilities of artificial intelligence and self-learning.

2. Literature Review

2.1. Traffic Big Data Environment. Digitized and informational infrastructure of urban road traffic and constructions of related systems have developed rapidly in the past ten years, and urban traffic control is developing from the “data poverty” times to the “data rich” times. Meanwhile, the appearance of ICV (intelligent connected vehicle) and autonomous vehicles will construct the future traffic environment jointly, which significantly differs from conventional manual driving vehicles in terms of individual information acquisition, perception ability, reaction time, interactive behavior, etc. New requirements of traffic control have formed a high-level demand for the next generation of traffic control [8]. The research on the next generation of traffic signal control for regional transportation under the “data rich” environment is on the agenda. Ma D et al. proposed the lane-based saturation degree estimation for signalized intersections and maximum queue length estimation for traffic lane groups, which enriched the way to obtain traffic parameters and increased the precision of estimation method. For example, the results show that the new method of maximum queue length estimation has a higher precision compared to the existing method based on a similar concept, with maximum and average deviations of 39.36% and 12.25%, respectively, over twenty cycles [9, 10].

Under the conditions of limited cross-section traffic flow data, many existing adaptive traffic control systems have adopted traffic models to actively predict the evolution of network traffic flows and then adopted the aggregative indicator method to optimize and solve timing parameters. However, the real-time detection of the spatiotemporal data based on urban road network traffic status can provide rich and high-quality basic data and fine-grained assessment of control effects for traffic control. In the face of the main defects encountered in the existing self-adaptive traffic control system, a closed-loop feedback self-adaptive control system with better uncertainty response capability and higher intelligent decision-making level is an inevitable result of the objective needs of the development and application technologies [11]. Ma D proposed a calculation method for the occupancy per cycle under different traffic conditions presented, based on the relationship between the three basic

traffic flow parameters, speed, traffic flow, and density [12]. The results show that the precision of this method was affected by the detector location and bus ratio insignificantly [13].

2.2. Reinforcement Learning Traffic Control. According to real-time collection of states, rewards, and punishments, the single intersection's signal control of reinforcement learning can find an optimization strategy of traffic signal control suitable for traffic flow characteristics through the interaction. In recent years, more and more domestic scholars have studied principles of reinforcement learning and discussed the applications of reinforcement learning algorithms in traffic control. Reinforcement learning has developed rapidly in the optimizing control [14, 15].

Scholars have done a lot of research on reinforcement learning theory, algorithms, and applications and have obtained many famous research results. Ma D proposed a new control method that brings significant and positive effects to the bottleneck link itself and to the entire test area [16]. Yang W concluded that critical issues in developing agent-based traffic control systems for integrated network were addressed as interoperability, adaptability, and extendibility [17]. Zhang L drew a conclusion that extensive simulation results for the designed Shanghai simulation scenarios indicate that most of the observed counts match quite well with the traffic simulation volumes and demonstrate the potential of MATSIM for large-scale dynamic transport simulation [18]. Aslani M developed adaptive traffic signal controllers based on continuous residual reinforcement learning (CRL-TSC) that was more stable, and the best setup of the CRL-TSC leads to saving average travel time by 15% in comparison to an optimized fixed-time controller [19].

Reinforcement learning control has the advantages of real-time online and feedback control, which especially accords with the control thoughts of signal adaptive control in urban intersections. However, there is a question as to whether the traffic signal optimization strategy based on the reinforcement learning is applicable to all the traffic environments.

3. Traffic Signal Control Strategy Based on Reinforcement Learning

Reinforcement learning is a typical data-driven control method. In this paper, the method of signal control scheme improvement is proposed. According to the different traffic flow characteristics, the subregions are divided. Based on the three key parameters of cycle length, arterial coordination signal offset, and green split, a set of hierarchical control algorithms based on reinforcement learning is constructed to optimize and improve the current signal timing scheme.

3.1. Control Subregions Division and Cycle Optimization. As for the regional coordination control, the primary content is the division of the coordination subregions. In the signal control road network, each intersection has its influence range, and the intersection and section within this range

are greatly affected by it. To quantify the impact and define the scope of influence, literature defines direct relevance to describe the relationship between adjacent intersections, finding that when the upstream node traffic flows into the downstream node, it is close to or greater than the downstream node's import capacity. It is found that the path correlation is mainly affected by the traffic network topology and OD distribution between the two intersections. The more the OD paths through two nodes at the same time, the stronger the correlation between nodes. The higher the flow rate of OD path passing through both nodes at the same time, the stronger the correlation between nodes. The more the OD paths that pass through both nodes at the same time are unique, the stronger the correlation between nodes will be.

The optimization range is region-level road network optimization. The control subregions are divided by characteristic parameters such as average travel time; vehicle OD amount between intersections and traffic coordination control subregions are finally determined.

The signal cycle refers to the time required for the signal color to display one cycle in the set phase order, that is, the sum of the steps of each control step in one cycle. The signal cycle is the key control parameter that determines the effectiveness of traffic signal control. If the signal cycle is too short, it is difficult to ensure that the vehicles in all directions can pass through the intersection smoothly, resulting in frequent stops at the intersection and a decline in the utilization rate of the intersection. If the signal cycle is too long, it will cause the driver to wait for too long, greatly increasing the delay time of the vehicle. The cycle in the green wave control is taken as the common cycle by the maximum signal cycle of the key intersection of the arterial, and the signal cycle of the remaining intersections is reallocated to each phase according to the traffic flow ratio.

According to different evaluation indexes, the optimal cycle is obtained by using model-based algorithm. Regarding the evaluation indicators of traffic efficiency at intersections, traffic capacity, saturation, service level, travel time, number of stops, and queue length are commonly used at home and abroad. The delay is mainly due to the travel time loss caused by traffic friction and traffic control. It is closely related to the cycle duration, green split, and saturation. It is an important indicator for evaluating the traffic service level and operational efficiency of signalized intersections, including queue delay, parking delay, control delay, and lane approach delay.

3.2. Offset Optimization Based on Bayesian Optimization Algorithm. The phase offset is also called the time offset or the green time offset. The phase offset includes the absolute phase offset and the relative phase offset. Absolute phase offset refers to the offset between the starting or ending point of the signal green light (red light) in the coordinated direction of the arterial at each intersection and the starting or ending point of the signal green light (red light) in the coordinated direction of the arterial at a certain intersection (generally a key intersection). Relative phase refers to the time offset between the starting or ending points of the green light

(red light) signal in the coordinated direction of the arterial at adjacent intersections. The relative phase offset is equal to the difference value between the absolute phase offset of two intersections, which is determined by the actual vehicle speed.

According to the coordination effect between the intersections, it is divided into several control subregions, and internal coordination control is implemented for its traffic characteristics. The basic principles of control subregions division are as follows:

(1) The distance between adjacent intersections is less than 600 meters and control subregion contains no more than 10 intersections.

(2) The optimal period length of each intersection is an integer multiple relationship.

The following lines with inconsistent coordination effects should not be included in a subregional coordination:

(1) An excessively long connection, and the traffic flow along the connection is highly discrete.

(2) There are traffic production sources or attraction sources (such as large parking lots and shopping malls) and very frequent pedestrian activities along both sides of certain lines, which seriously interfere with traffic flow.

The Bayesian optimization algorithm belongs to the sequential model-based optimization (SMBO) algorithm. This algorithm determines the value of the next (optimal) sample set by analyzing historical observations of a loss function f . Since the Bayesian optimization algorithm was proposed around 2010, it has been used to optimize the hyperparameters of machine learning models in the field of machine learning in recent years. The so-called superparameter is the model parameter that needs to be set artificially. In this competition, due to the large number of timing parameters that need to be optimized, which includes the signal split and phase offset of multiple different intersections, the solution space dimension is relatively high and the optimization is quite difficult. The overall idea of the Bayesian optimization algorithm is as follows:

Calculate the posterior expectation of the loss function f using the observed sample set $\mathbf{X}_{1:n}$.

Generate a new set of samples \mathbf{X}_{new} to sample the loss function f , which can maximize the expectation of f in the value range of independent variables.

Repeat the above steps until the preset convergence condition is reached. End the optimization process.

The algorithm will be described in detail below and the process will be summarized.

To calculate the posterior expectation of the loss function f , the likelihood model of the sample and the prior probability model of f should be obtained in advance. In the Bayesian optimization process, we can assume that the sample obeys the multivariate Gaussian distribution and obtain the Gaussian likelihood function:

$$\mathbf{y} = f(\mathbf{X}) + \epsilon, \quad \epsilon \sim N(0, \sigma_\epsilon^2) \quad (1)$$

For the prior distribution, we assume that the loss function f can be described by a Gaussian process (GP). The essence of the Gaussian process is the generalization

of the multivariate Gaussian distribution to the function distribution. Therefore, just as the Gaussian distribution is determined by its expectation and variance, the Gaussian process is completely determined by its expectation function $m(\mathbf{X})$ and the covariance function $k(\mathbf{X}, \mathbf{X}')$. The Gaussian process is widely used in the application of all probabilistic models because its description of the posterior distribution of the loss function is easier for us to analyze and calculate.

One of the most widely used acquisition functions is the expected improvement (EI) function. The EI function is defined as

$$EI(\mathbf{X}) = \mathbb{E} \left[\max \{0, f(\mathbf{X}) - f(\widehat{\mathbf{X}})\} \right] \quad (2)$$

where $\widehat{\mathbf{X}}$ is the current optimal sample set, and this function gives a new sample set that can best enhance the expectation of the loss function. Moreover, the expected lifting function can be calculated based on the Gaussian process model, namely,

$$EI(\mathbf{X}) = \begin{cases} (\mu(\mathbf{X}) - f(\widehat{\mathbf{X}})) \Phi(Z) + \sigma(\mathbf{X}) \vartheta(Z) & \text{if } \sigma(\mathbf{X}) > 0 \\ 0 & \text{if } \sigma(\mathbf{X}) = 0 \end{cases} \quad (3)$$

$$Z = \frac{\mu(\mathbf{X}) - f(\widehat{\mathbf{X}})}{\sigma(\mathbf{X})} \quad (4)$$

where $\Phi(Z)$ and $\vartheta(Z)$ are the cumulative distribution function and probability density distribution function of the multivariate standard Gaussian distribution, respectively. When the posterior expectation $\mu(\mathbf{X})$ is higher than the current loss function optimal value $f(\widehat{\mathbf{X}})$, EI will get a larger value. When the uncertainty $\sigma(\mathbf{X})$ of \mathbf{X} is high, EI will get a larger value.

After the above analysis and introduction, the whole principle and process of Bayesian optimization can be summarized to form a Bayesian optimization algorithm:

Given the observed value $f(\mathbf{X})$ of the loss function, the posterior expectation of the loss function f is updated based on the Gaussian model.

Solve the expected lifting function (EI function) to find the new best sample set: $\mathbf{X}_{new} = \arg \max EI(\mathbf{X})$.

Calculate the value of the loss function at \mathbf{X}_{new} .

Repeat the above steps until the preset number of repetitions (i.e., the number of iterations) is reached or the convergence condition is met.

In (2) of the above steps, we can use the gradient-based solution method to optimize the EI function to get \mathbf{X}_{new} .

On the basis that the parameters such as the optimal cycle length are determined, the phase offset of the intersections after deduplication $\mathbf{X} = \{x_1, \dots, x_n\}$ can be regarded as input loss function sample set. The sample \mathbf{X} of the function, which is returned by the online feedback, can be iterated multiple times based on the Bayesian optimization algorithm.

3.3. Split Optimization Based on Q-Learning Algorithm. In the urban transportation system, the traffic flow, vehicle

speed, and traffic density are the most intuitive reflections of traffic conditions. They are the three characteristic parameters of traffic flow and the research focus and foundation of traffic flow theory. Among them, the traffic flow refers to the number of vehicles passing through per unit time; the vehicle speed refers to the distance that the vehicle passes per unit time; and the traffic density refers to the number of vehicles on the section per unit length. The traffic flow theory is the basis for the establishment of urban traffic signal control system.

The traffic model uses a discrete-time difference equation or a continuous time subdivision tool to introduce a dynamic relationship between the concepts of traffic volume Q , vehicle speed V , and traffic density K , which summarizes the physical quantities of the traffic network and is used to describe the collective average behavior of a large number of vehicles. In the free flow, the interaction between vehicles can be neglected, and the traffic flow increases linearly with the vehicle density. The wide moving jam flow is usually characterized by stop-go-stop traffic, that is, a series of jams. The density of vehicles in the region is high and the average speed and flow of vehicles are small. The average velocity of the synchronized flow is significantly lower than that of the free flow.

At present, Q-learning algorithm is one of the most frequently used methods in the fields of reinforcement learning, proposed by Watkins in 1989 [20]. Q-learning algorithm is widely used in the fields of control, depending on the update mode of its special value function.

In Q-learning, the solution formula of the mainstream value function is as follows.

$$\begin{aligned} Q(s_t, a_t) & \leftarrow Q(s_t, a_t) \\ & + \alpha \left[\left(r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right) \right] \end{aligned} \quad (5)$$

According to the formula, at the moment of t , the state of Q-learning is s_t . If the taken action is a_t , the corresponding value function will be $Q(s_t, a_t)$. The update of the value function is determined by three factors. The first is the current value of the action state value function, $Q(s_t, a_t)$, that needs to be updated. The second is to control the corresponding maximum value of all Q-values of actions in the postexecution state of $s(t+1)$, and the third is the immediate return, $r(t+1)$, after the action. Besides, there are also two model parameters, learning rate $\alpha \in [0,1]$ and discount factor $\gamma \in (0,1]$. The former is used to balance the relationship between the learning and utilization of the algorithm. When $\alpha \rightarrow 1$, the controller tends to explore new knowledge; otherwise it will use the existing knowledge. The latter is used to coordinate the present relationship with the future. When $\gamma \rightarrow 1$, the controller tends to consider the future return, and when $\gamma \rightarrow 0$, the controller mainly considers immediate return [21].

Whether in theoretical research or in engineering practice analysis, road traffic density is an effective indicator for measuring the degree of traffic congestion. The operation of

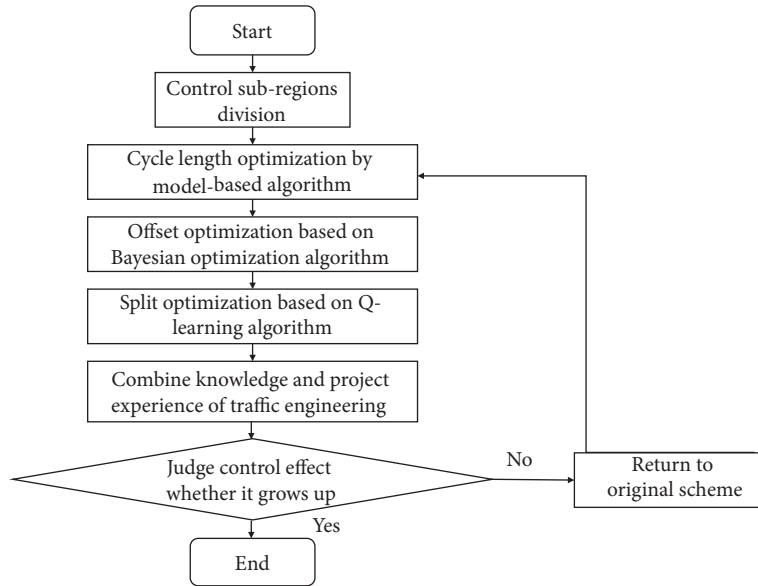


FIGURE 1: The flow chart of the hierarchical control algorithm based on reinforcement learning.

the traffic on the section is affected by the signal control of the upstream and downstream intersections. The release signal at the upstream intersection directly changes the density of the section, which indirectly affects the traffic capacity and saturation at the section of the stop line and indirectly affects the density of the queue section. The mutual influence of the two is especially noticeable in the supersaturated state. Since the penetration rate of the connected vehicles in different sections is unknown, it is impossible to visually reflect the actual flow of the road through the number of discrete connected vehicles. Even by expanding the sample, it is difficult to guarantee accuracy, but it can clearly reflect the speed of the overall traffic flow. Therefore, this paper uses traffic density as the core parameter to provide a basis for green split optimization.

3.4. The Flow of the Control Algorithm. Firstly, according to different evaluation indexes, the optimal cycle is obtained by using model-based algorithm. Using the combination of the average travel time of vehicles and the Bayesian optimization method based on the Gaussian process, which is commonly used in the optimization of machine learning algorithms, the arterial coordination control is set. The phase offset is optimized by the two-way flow ratio of the upstream and downstream roads and the reasonable setting of the pedestrian crossing phase. Then set different green wave bandwidths to match the upstream and downstream traffic of the morning rush hour and the tidal phenomenon with uneven travel speed. The intelligent algorithm such as Q-learning is used to optimize the green split of each intersection by using key traffic flow parameters at each intersection.

In conclusion, the flow of the traffic signal control strategy based on reinforcement learning is as shown in Figure 1.

4. Verification of Traffic Signal Control Strategy Based on Reinforcement Learning

4.1. Verification of Single Intersection Signal Control Strategy Based on Reinforcement Learning. The intelligent algorithm such as Q-learning is used to optimize the green split of single intersection by using key traffic flow parameters at the intersection. We have compared the Q-learning control method adopted in this paper with the traditional timing signal control and the adaptive control method in second-generation traffic signal control system. The delay of intersection means the average delay for all vehicles passing through all of lane groups at the intersection in the same cycle. The results are shown in Figure 2.

Having compared the traditional timing control with the traffic signal control method based on Q-learning algorithm applied in this paper, the study has shown that the application of Q-learning control method has achieved good performance. In terms of effectiveness, compared with the traditional timing control, the optimization effects of traffic signal control based on Q-learning have, respectively, reached 31.68%, 30.10%, 37.59%, 38.07%, 40.69%, and 43.89%, which has shown that compared with traditional timing control, the traffic signal control based on Q-learning can achieve better optimization effects. However, compared with the existing traffic control strategy, the optimization effects of traffic signal control based on Q-learning have, respectively, reached -4.21%, -5.28%, 3.14%, 6.23%, 13.11%, and 9.72%. The optimization effects of traffic signal control based on reinforcement learning are inferior under low flow conditions, while they are better under medium and high flow conditions.

4.2. Verification of Arterial Intersection Signal Control Strategy Based on Reinforcement Learning. The green wave coordinated control has three important parameter conditions: the

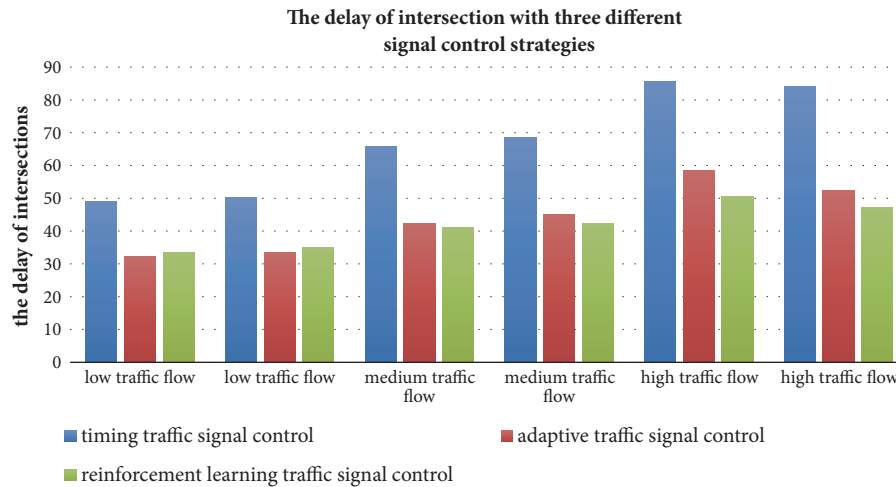


FIGURE 2: Comparisons of traffic control methods of single intersections.

signal clocks of each intersection should be synchronized; the signal cycle should be the same and have phase offset (the travel time calculated by the adjacent intersection based on the actual average speed). Only with these three conditions can the validity of green wave be guaranteed.

Discrete connected vehicle trajectory data cannot directly obtain the data required for signal distribution and intersection channelization scheme under traditional conditions but can obtain more detailed and complete trajectory-level data. The complete physical trajectory of the vehicle during driving can not only reflect the driving path of the vehicle on the road network, but also reflect the changing characteristics of the vehicle speed with time and space. It is the most comprehensive and complete expression form of the traffic flow operating state, containing a wealth of traffic flow information which is key parameter for offset optimization (for example, travel speed, queue length, delay, and stop times).

For the coordinated control problem of the supersaturation state during morning rush hour, the above research basically follows the idea based on the strong mathematical hypothesis model, but the control system deviates from the original trajectory due to the inaccuracy based on the strong mathematical hypothesis model and the interference from the outside of the control system. In response to these shortcomings, some scholars have further proposed the idea of predictive control, which enables the system to correct the trajectory deviation in real time and achieve the purpose of optimal control. However, the establishment of the optimal control model is still a centralized processing idea. In the application of intersection control problems, it focuses on the control problem of single-point intersections. From the perspective of the structure of the control algorithm, the hierarchical control structure can integrate more control personnel's design ideas, which is of great help to solve the problem of complex control state of the road network. With the development of intelligent control technology, Bayesian optimization methods based on Gaussian process, fuzzy control, reinforcement learning calculation, and neural network

have been also widely used in traffic control. However, these applications present a similar feature that is loosely integrated with the actual traffic condition. The computational speed of the online control system is still a big obstacle, and it is more difficult to push it to practical applications. Therefore, traffic control at the network level for rush hours should be based on offline large-scale optimization calculation based on traffic model (based on travel time, then obtaining the relative phase offset between adjacent intersections) and intelligent algorithm (Bayesian optimization methods based on Gaussian process), seeking to achieve a system-optimized phase offset timing scheme.

On the other hand, in the traditional arterial coordinated control scheme, the green wave velocity, the forward green wave bandwidth, and the reverse green wave bandwidth between the starting point and the ending point are always the same or almost the same, and no or less consideration is given to the individualization of the velocity distribution between the sections and the tidal of the traffic flow. Therefore, we adopted different green wave speed optimization methods for different road sections, combined with the unique tidal phenomenon. In the direction of large traffic flow, based on the calculated green wave bandwidth, the bandwidth of the reverse green wave is appropriately increased to match the traffic demand of rush hour. At the same time, the vehicle traffic phase is, respectively, covered to the forward and reverse two-way green wave at the pedestrian crossing, which minimizes the probability that the vehicle stops at the signal control pedestrian crossing.

As is shown in Figure 3, the comparison of original signal control scheme and hierarchical traffic signal control method in different traffic flows has been given. Compared with the original signal control scheme, the arterial traffic signal optimization control method based on hierarchical traffic signal control performs better in medium and high traffic flow. The average delay per vehicle of the original signal control scheme is, respectively, 28.47 seconds, 40.34 seconds, and 61.38 seconds. While with regard to the arterial traffic signal optimization control method based on hierarchical traffic

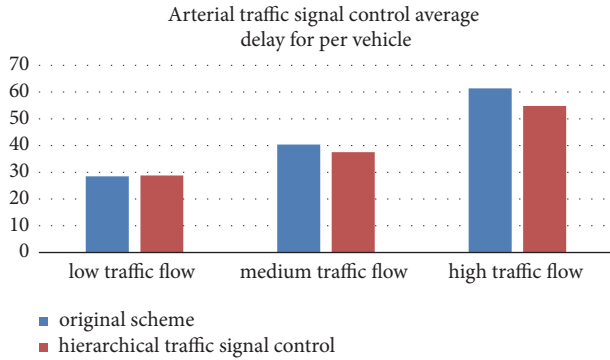


FIGURE 3: Evaluation of the traffic control method of arterial multiple intersections.

signal control, the average delay per vehicle is, respectively, 28.77 seconds, 37.56 seconds, and 54.79 seconds. Then, the optimization percentage is, respectively, -1.05%, 6.89%, and 10.74%. However, due to the randomness of traffic flows, although there is some increasing average delay in low traffic flow, the general trend has similar conclusions.

4.3. Verification of Regional Traffic Signal Control Strategy Based on Reinforcement Learning. The concept of regional signal control can be divided into a narrow sense and a broad sense. In a narrow sense, regional signal control is a signal control method that unifies several intersections with strong correlation and carries out mutual coordination, namely, the so-called regional signal coordination control. In a broad sense, regional signal control refers to the monitoring of all intersections within the region under the management of a command and control center. It is a comprehensive signal control for single isolated intersection, multiple intersections of the arterial and the highly connected intersection group. It can be classified according to control strategy (timed offline control system, adaptive online control system), control mode (scheme selection, solution generation), and control structure (centralized, distributed).

The purpose of vehicle path feature extraction is to obtain the information of the nodes (i.e., intersections and OD points) that each vehicle passes through, so as to be able to calculate other dynamic features of the sections and road networks (such as traffic, average speed, and road network OD matrix). However, the trajectory data does not contain information such as when the vehicle passed through which node, and we only know the coordinate points of the vehicle trajectory. Then, can we extract the vehicle trajectory by using the trajectory coordinate points and the node information?

In the beginning, we tried clustering based on clustering methods, trying to cluster the coordinate points according to the separated road segments. However, after experimenting with various mainstream clustering methods, it is found that clustering cannot solve the tagging problem of coordinate points. How do we mark the vehicle coordinate points with the tag of the node? After further experimentation, we thought that the vehicle path extraction can be carried out

by using the function `inpolygon` that comes with MATLAB. The core idea is as follows:

(1) Taking each node as the center, the regular polygons that can cover a certain section range are constructed, respectively. These regular polygons represent the in-out range of the node, which is referred to as the node regular polygon. If a coordinate point is within a node regular polygon, it is considered to belong to the node; otherwise, it does not belong to the node.

(2) Traverse the trajectory data and find out which node the coordinate points belong to. If a coordinate point does not belong to all nodes, it means that the coordinate point is on the road segment and special labeling is carried out.

(3) Compress the trajectory information of each vehicle, and record the starting line number of each vehicle passing through the node and the number of coordinate points at each node.

(4) Continue to compress the trajectory information of each vehicle, record all the nodes that each vehicle passes through (in the form of a string and a vector), and derive the OD characteristics of the vehicle.

In this paper, the method of signal control scheme improvement is proposed. According to the different traffic flow characteristics, the subregions are divided. Based on the three key parameters of cycle, arterial coordination signal offset, and green split, a set of hierarchical control algorithms based on reinforcement learning is constructed to optimize and improve the current signal timing scheme. Firstly, according to different evaluation indexes, the optimal period is obtained by using model-based algorithm. Using the combination of the average travel time of vehicles and the Bayesian optimization method based on the Gaussian process, which is commonly used in the optimization of machine learning algorithms, the arterial coordination control is set. The phase offset is optimized by the two-way flow ratio of the upstream and downstream roads and the reasonable setting of the pedestrian crossing phase. Then set different green wave bandwidths to match the upstream and downstream traffic of the rush hour and the tidal phenomenon with uneven travel speed. The intelligent algorithm such as Q-learning is used to optimize the green split of each intersection by using key traffic flow parameters such as traffic flow, density, and speed at each intersection. In the end, this paper uses the hierarchical traffic signal control algorithm based on reinforcement learning and combines the relevant knowledge of traffic engineering and engineering project experience to fine-tune the phase offset and green split to solve the problem of green wave bottleneck point of the arterial and signal interference caused by the right-turning vehicle and then obtain the optimal solution.

There are four key evaluation indexes included the number of vehicles leaving the road network at the end of the simulation, and the total delay time, the total travel time, and the total stopping number have been reduced to varying degrees shown in Figure 4.

For the different traffic flow states, the optimization effect caused by the Q-learning signal control method proposed in this paper in the high flow conditions (32.73%) is better than that in the medium flow and low flow conditions (22.32%,

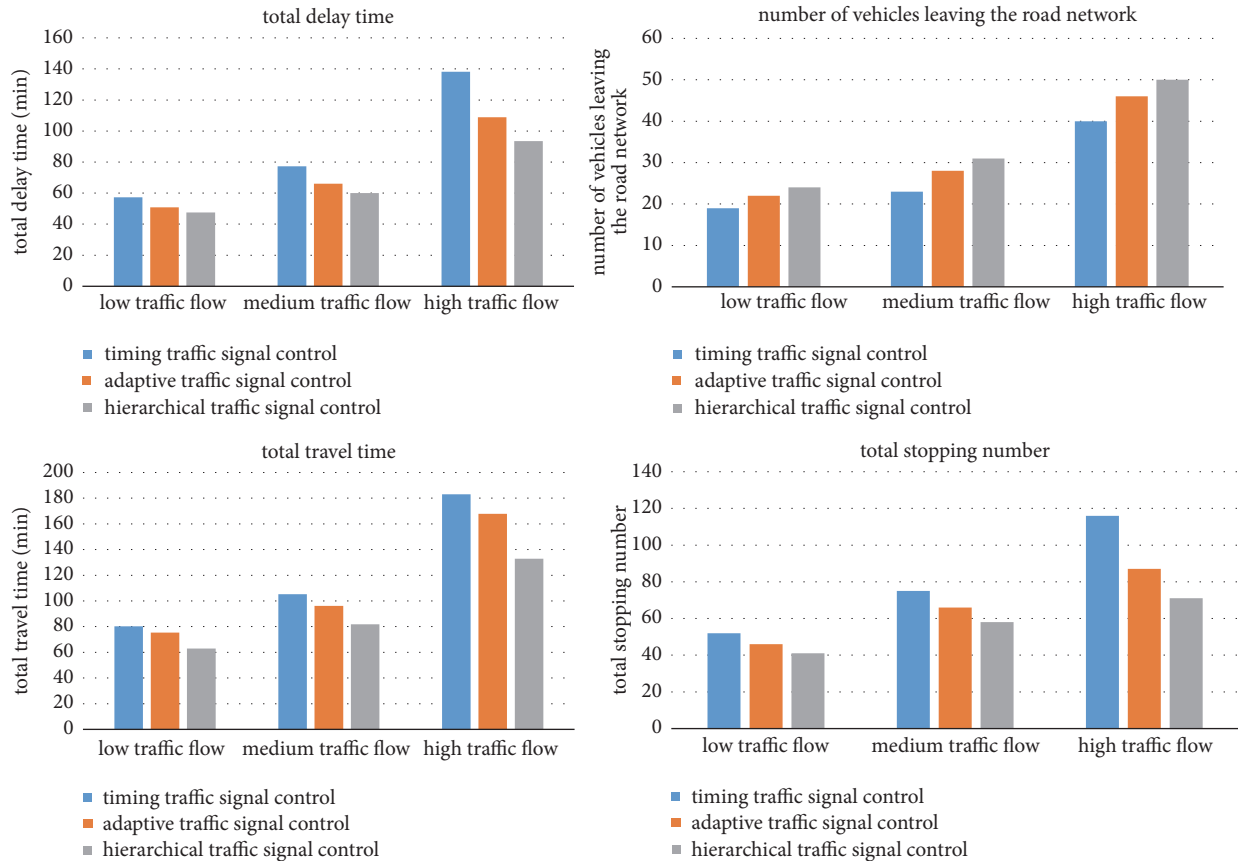


FIGURE 4: Evaluation of the traffic control method of regional road network.

17.11%). This also proves that the traffic control strategy based on reinforcement learning is more suitable for complex traffic environment (medium and high traffic flow, multi-intersection).

4.4. Analysis of Simulation Results. Through the above comparisons and analysis, it can be concluded that the traffic signal optimization strategy based on reinforcement learning is not applicable for all traffic environments. As to single intersections and arteries, their control effects are inferior to the current adaptive traffic signal control strategy in the low flow conditions. However, the traffic signal optimization strategy based on reinforcement learning is suitable for complex traffic environments (high flows and multiple intersections), and the effects of which are better than the current optimization methods in the conditions of high flows, as to both single intersections and arteries. In the future, we will focus on the network, continue to study the network traffic signal optimization method based on reinforcement learning, and then compare the effects with traditional optimization algorithms.

5. Conclusion and Discussion

This paper uses the hierarchical traffic signal control algorithm based on reinforcement learning and combines the

relevant knowledge of traffic engineering and engineering project experience to fine-tune the phase offset and green split to solve the problem of green wave bottleneck point of the arterial and signal interference caused by the right-turning vehicle and then obtain the optimal solution.

In terms of the temporal dynamics of traffic control, reinforcement learning does not have complex optimizing modules and instant decisions can be made to respond to the uncertainty of time-varying traffic flow according to the characteristics of traffic flow observed in real time, which also corresponds with the actual conditions. Therefore, this paper focuses on the application of reinforcement learning in the field of traffic control and concludes that the traffic control method based on reinforcement learning has a better applicability in the complex traffic environment (high flows and multiple intersections), but it is not applicable to all traffic conditions. Furthermore, different from single intersection signal control, facing the integrated control for mainline and networked level traffic, it is still necessary to make further analysis in the aspects of data models and samples, coordination optimization techniques and multi-agent strategies, and mechanism analysis of the interaction between heuristic guidance and higher-level optimization mechanisms such as the pure stochastic optimization and hierarchical algorithm.

Data Availability

According to the data support, the authors have obtained data in the field and simulated them by VISSIM—C# to implement secondary development with kernel algorithm.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to acknowledge the Intelligent Transportation System Research Center of Tongji University for data support. The research is supported by Project of National Natural Science Foundation of China (Project No. 61773293 and No. 61773288) and Key Project of National Natural Science Foundation of China (Project No. 51238008).

References

- [1] N. H. Gartner, C. Stamatiadis, and P. J. Tarnoff, "Development of advanced traffic signal control strategies for intelligent transportation systems: multilevel design," *Transportation Research Record*, no. 1494, pp. 98–105, 1995.
- [2] A. G. Sims, "The Sydney coordinated adaptive traffic system," in *Proceedings of the Engineering Foundation Conference on Research Directions in Computer Control of Urban Traffic Systems*, Calif, USA, 1979.
- [3] Y.-T. Wu and C.-H. Ho, "The development of Taiwan arterial traffic-adaptive signal control system and its field test: A Taiwan experience," *Journal of Advanced Transportation*, vol. 43, no. 4, pp. 455–480, 2009.
- [4] R. Vincent, "Safer signalized junction design and self-optimizing control," *Journal of Advanced Transportation*, vol. 28, no. 3, pp. 217–226, 2010.
- [5] H. K. Lo and H. F. Chow, "Adaptive traffic control system: Control strategy, prediction, resolution, and accuracy," *Journal of Advanced Transportation*, vol. 36, no. 3, pp. 323–347, 2010.
- [6] C. Meneguzzer, "Dynamic process models of combined traffic assignment and control with different signal updating strategies," *Journal of Advanced Transportation*, vol. 46, no. 4, pp. 351–365, 2012.
- [7] K. Aboudolas, M. Papageorgiou, and E. Kosmatopoulos, "Store-and-forward based methods for the signal control problem in large-scale congested urban road networks," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 2, pp. 163–174, 2009.
- [8] J. Hao, *Studies of the application of data-driven control method in traffic control*, Beijing Jiaotong University, Beijing, China, 2013.
- [9] D. Ma, X. Luo, S. Jin, W. Guo, and D. Wang, "Estimating Maximum Queue Length for Traffic Lane Groups Using Travel Times from Video-Imaging Data," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 3, pp. 123–134, 2018.
- [10] D. Ma, X. Luo, S. Jin, D. Wang, W. Guo, and F. Wang, "Lane-based saturation degree estimation for signalized intersections using travel time data," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 3, pp. 136–148, 2017.
- [11] Y. Wang, X. Yang, H. Liang, and Y. Liu, "A Review of the Self-Adaptive Traffic Signal Control System Based on Future Traffic Environment," *Journal of Advanced Transportation*, vol. 2018, Article ID 1096123, 12 pages, 2018.
- [12] D.-F. Ma, D.-H. Wang, F. Sun, Y.-M. Bie, and S. Jin, "Method of spillover identification in urban street networks using loop detector outputs," *Journal of Central South University*, vol. 20, no. 2, pp. 572–578, 2013.
- [13] D. Ma, D. Wang, Y. Bie, S. Jin, and Z. Mei, "Identification of spillovers in urban street networks based on upstream fixed traffic data," *KSCE Journal of Civil Engineering*, vol. 18, no. 5, pp. 1539–1547, 2014.
- [14] Littman M. L., "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of The 11th International Conference on Machine Learning*, pp. 157–163, 1994.
- [15] S. J. Bradtke and Michael O. D., "Reinforcement learning methods for continuous-time Markov decision problems," *Advances in Neural Information Processing Systems*, pp. 393–400, 1995.
- [16] D. Ma, F. Fu, S. Jin et al., "Gating control for a single bottleneck link based on traffic load equilibrium," *International Journal of Civil Engineering*, vol. 14, no. 5, pp. 281–293, 2016.
- [17] W. Yang, L. Zhang, Y. Shi, and M. Zhang, "Applications of Agent Technology in Urban Traffic Signal Control Systems: A Survey," *Journal of Wuhai University of Technology (Transportation Science Engineering)*, vol. 38, no. 4, pp. 709–718, 2014.
- [18] L. Zhang, W. Yang, J. Wang, and Q. Rao, "Large-scale agent-based transport simulation in shanghai, china," *Transportation Research Record*, vol. 2399, no. 1, pp. 34–43, 2013.
- [19] M. Aslani, S. Seipel, and M. Wiering, "Continuous residual reinforcement learning for traffic signal control optimization," *Canadian Journal of Civil Engineering*, vol. 45, no. 8, pp. 690–702, 2018.
- [20] C. J. Watkins, *Learning from delayed rewards*, University of Cambridge, UK, 1989.
- [21] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, 1992.

Research Article

An Improved Deep Learning Model for Traffic Crash Prediction

Chunjiao Dong ,^{1,2,3} Chunfu Shao,^{1,2} Juan Li,¹ and Zhihua Xiong ¹

¹MOE Key Laboratory for Urban Transportation Complex Systems Theory and Technology, Beijing Jiaotong University, Beijing 100044, China

²Key Laboratory of Industry of Big Data Application Technology for Comprehensive Transport, Ministry of Transport, Beijing Jiaotong University, Beijing 100044, China

³Center for Transportation Research, Tickle College of Engineering, University of Tennessee, 600 Henley Street, Knoxville, 37996, Tennessee, USA

Correspondence should be addressed to Chunjiao Dong; cjdong@bjtu.edu.cn

Received 12 June 2018; Accepted 12 November 2018; Published 10 December 2018

Guest Editor: Hamzeh Khazaei

Copyright © 2018 Chunjiao Dong et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Machine-learning technology powers many aspects of modern society. Compared to the conventional machine learning techniques that were limited in processing natural data in the raw form, deep learning allows computational models to learn representations of data with multiple levels of abstraction. In this study, an improved deep learning model is proposed to explore the complex interactions among roadways, traffic, environmental elements, and traffic crashes. The proposed model includes two modules, an unsupervised feature learning module to identify functional network between the explanatory variables and the feature representations and a supervised fine tuning module to perform traffic crash prediction. To address the unobserved heterogeneity issues in the traffic crash prediction, a multivariate negative binomial (MVNB) model is embedding into the supervised fine tuning module as a regression layer. The proposed model was applied to the dataset that was collected from Knox County in Tennessee to validate the performances. The results indicate that the feature learning module identifies relational information between the explanatory variables and the feature representations, which reduces the dimensionality of the input and preserves the original information. The proposed model that includes the MVNB regression layer in the supervised fine tuning module can better account for differential distribution patterns in traffic crashes across injury severities and provides superior traffic crash predictions. The findings suggest that the proposed model is a superior alternative for traffic crash predictions and the average accuracy of the prediction that was measured by RMSD can be improved by 84.58% and 158.27% compared to the deep learning model without the regression layer and the SVM model, respectively.

1. Introduction

Road traffic injuries are a leading cause of preventable death, especially, for the young people. In the United States, traffic crashes were the number one cause of death among people from 16 to 24 years old for each year from 2012 to 2014 [1]. In 2015, the nation lost 35,092 people in traffic crashes, a 7.2-percent increase from 32,744 in 2014, which is the largest percentage increase in nearly 50 years [2]. This is an average of approximately 96 people being killed on the nation's roadways every day of the year, and an average of more than four people per hour. In other words, one person dies on roadways every 15 minutes.

To understand the relationship between the influence factors and traffic crash outcomes, with the extracted data from police reports and state highway-asset-management databases, the analyses of traffic safety estimate and predicate the likelihood of a traffic crash. The number of crashes occurring on a defined spatial entity over a specific time period (for example, the number of crashes per year occurring at a roadway intersection, over a specified roadway segment, or in a region) would be considered as the dependent variables and some of the many factors affecting the likelihood of a traffic crash are analyzed and examined (see [3–5] for a comprehensive review). Though more and more factors that are relevant to the traffic crashes have been incorporated and

the proposed models became more and more sophisticated, there are still some factors that are not available to the researchers and the models result in bias estimations and erroneous predictions. In this study, we proposed an innovative approach for traffic crash prediction, which incorporates a multivariate regression layer into a dynamic deep learning model that contains an unsupervised feature learning module and a supervised fine tuning module governing the state dynamics to improve the performances of prediction.

2. Literature Review

The statistical methodologies, such as the Poisson, negative binomial (NB), and their variants in univariate and multivariate regression frameworks, have been successfully applied in crash count analyses [4, 6–8], which attempt to deal with the data and methodological issues associated with traffic crash estimations and predictions, and enhance our understanding on the relationship between the influence factors and traffic crash outcomes. However, current research in traffic safety indicates that the applied statistical modeling fails when dealing with complex and highly nonlinear data [9], which could suggest that the relationship between the influence factors and traffic crash outcomes is more complicated than can be captured by a single statistical approach. In addition, most of the statistical methods are based on some strong assumptions, such as specifying a priori and the error distribution. Moreover, a problematic issue is multicollinearity, i.e., the high degree of correlation between two or more independent variables. Furthermore, statistical models have difficulty when dealing with outliers, missing or noisy data [10].

To deal with the limitations of statistical methodologies, the machine learning methods, including Artificial Neural Network (ANN), Support Vector Machine (SVM) models, and deep learning models, have been applied to various traffic safety problems and used as data analytic methods because of their ability to work with massive amounts of multidimensional data. In addition, because of the modeling flexibility, learning and generalization ability, and good predictive ability, the machine learning has been considered as generic, accurate, and convenient mathematical models in the field of traffic safety.

Because the commonly used Poisson or NB regression models assume the predefined underlying relationship between dependent and independent variables and the violation of the assumption would lead to erroneous estimation, ANN and Bayesian neural network (BNN) models have been employed to analyze the traffic safety problems for many years. Although both ANN and BNN models have similar multilevel network structures, they are different in predicting the outcome variables. For ANN, the weights are assumed to fix. However, the weights of BNN follow a probability distribution and the prediction needs to be integrated over all the probability weights. Basically, the ANN can be characterized by three features: network architecture, model of a neuron, and learning algorithms. Chang [11] compared the performances of NB regression model and ANN in crash frequency analyses. The results showed that

ANN is a consistent alternative method for analyzing crash frequency. Abdelwahab and Abdel-Aty [12] employed two well-known ANN paradigms [the multilayer perceptron and radial basis functions (RBF) neural networks] to analyze the traffic safety of toll plazas and evaluate the impacts of electronic toll collection (ETC) systems on highway safety. The performance of ANN was compared with calibrated logit models. Modeling results showed that the RBF neural network was the best model for analyzing driver injury severity. Xie, Lord, and Zhang [13] evaluated the application of BNN models for predicting traffic crashes by using data collected on rural frontage roads in Texas. The results showed that back-propagation neural network (BPNN) and BNN models perform better than the NB regression model in terms of traffic crash prediction. The results also showed that BNNs could be used to address other issues in highway safety, such as the development of crash modification factors, and enhance the prediction capabilities for evaluating different highway design alternatives. Kunt, Aghayan, and Noii [14] employed a genetic algorithm (GA), pattern search, and ANN models to predict the severity of freeway traffic crashes. The results showed that the ANN provided the best predictions. Jadaan, Al-Fayyad, and Gammoh [15] developed a traffic crash prediction model using the ANN simulation with the purpose of identifying its suitability for predicting traffic crashes under Jordanian conditions. The results demonstrated that the estimated traffic crashes are close to actual traffic crashes. Akin and Akbas [16] proposed an ANN model to predict intersection crashes in Macomb County of the State of Michigan. The predictive capability of the ANN model was determined by classifying the crashes into these types: fatal, injury, and property damage only (PDO) crashes. The results were very promising and showed that ANN model is capable of providing an accurate prediction (90.9%) of the crash types. In summary, though ANN and BNN models show better linear/nonlinear approximation properties than traditional statistical approaches, these models often cannot be generalized to other data sets [3].

The SVM models have recently been introduced for traffic safety analyses [17, 18], which are a new class of models that are based on statistical learning theory and structural risk minimization [19]. These models are supposed to approximate any multivariate function to any desired degree of accuracy with a set of related supervised learning methods. Li et al. [17] evaluated the application of SVM models for predicting motor vehicle crashes. The results showed that SVM models predict crash data more effectively and accurately than traditional NB models. In addition, the findings indicated that the SVM models provide better (or at least comparable) performance than BPNN models and do not over-fit the data. To identify the relationship between severe crashes and the explanatory variables and enhance model goodness-of-fit, Yu and Abdel-Aty [20] developed three models to analyze crash injury severity, which include a fixed parameter logit model, a SVM model, and a random parameter logit model. The results showed that the SVM models and the random parameter models provide superior model fits compared to the fixed parameter logit model. Findings also demonstrate

that it is important to consider possible nonlinearity and individual heterogeneity when analyzing traffic crashes. Chen et al. [21] employed the SVM models to investigate driver injury severity patterns in rollover crashes using two-year crash data collected in New Mexico. The results showed that the SVM models produce reasonable predictions and the polynomial kernel outperforms the Gaussian RBF kernel. Dong, Huang, and Zheng [22] proposed a SVM model to handle multidimensional spatial data in crash prediction. The results showed that the SVM models outperform the nonspatial models in terms of model fitting and predictive performance. In addition, the SVM models provide better goodness-of-fit compared with Bayesian spatial model with conditional autoregressive prior when utilizing the whole dataset as the samples. Ren and Zhou [23] proposed a novel approach that combines particle swarm optimization (PSO) and SVM for traffic safety prediction. The results showed that the predictions of PSO-SVM are better than that from BP neural network. Yu and Abdel-Aty [24] proposed the SVM models with different kernel functions to evaluate real-time crash risk. The results showed that the SVM model with RBF kernel outperformed the SVM model with linear kernel and Bayesian logistic regression model. In addition, the findings showed that smaller sample size could improve the classification accuracy of the SVM models and variable selection procedure is needed prior to the SVM model development. Overall, it has been found that the SVM models showed better or comparable results to the outcomes predicted by ANN/BNN and other statistical models [19]. However, like ANN and BNN, the SVM models often cannot be generalized to other data sets and they all tend to behave as black-boxes, which cannot provide the interpretable parameters as statistical models do.

Other than the ANN/BNN and SVM models, other machine learning methods have been introduced in traffic safety analyses. Abdel-Aty and Haleem [25] introduced a recently developed machine learning technique—multivariate adaptive regression splines (MARS) to predict vehicle angle crashes using extensive data collected on unsignalized intersections in Florida. The results showed that MARS outperformed the NB models. The proposed MARS models showed promising results after screening the covariates using random forest. The findings suggested that MARS is an efficient technique for predicting crashes at unsignalized intersections.

Deep learning is a recently developed branch of machine learning method and has been successfully applied in speech recognition, visual object recognition, object detection, and many other domains such as drug discovery and genomics [26, 27]. Compared to the conventional machine learning techniques that were limited in their ability to process natural data in their raw form, deep learning constructs computational models aiming to extract inherent features in data from the lowest level to the highest level. Though the deep learning methods have shown outstanding performances in many applications [26], the applications of deep learning in the field of transportation are relatively few and only focusing on the topic of traffic flow prediction [28–30]. In this study, we proposed an improved deep learning model

for traffic crash prediction. The proposed model includes two modules: an unsupervised feature learning module and a supervised fine tuning module. To discover nonlinear relationship between the investigated variables and identify the impacts of influence factors on traffic crashes for roadway network, a DAE model is proposed in the unsupervised feature learning module to learn the features of explanatory variables. In addition, a multivariate negative binomial (MVNB) regression is embedding into the supervised fine tuning module to address the heterogeneity issues. The proposed model performances are evaluated by comparing to the deep learning model without the MVNB layer and SVM models by using five-year data from Knox County in Tennessee.

3. The Modeling Framework Formulation

A novel model is proposed for the traffic crash prediction and Figure 1 illustrates the modeling framework. The proposed model includes two modules. One is the unsupervised feature learning module and another is the supervised fine tuning module. The obtained encoded feature representations from the unsupervised feature module are used as the input for the supervised fine tuning module.

3.1. Unsupervised Feature Learning Module. Compared to the commonly used deep learning architectures including deep belief network [31], stacked autoencoder [32], and convolutional neural networks [27], the symmetrical neural networks in an unsupervised manner have shown better performances, which can automatically learn an appropriate sparse feature representation from the raw data [33]. The unsupervised feature learning module includes a denoising autoencoder (DAE) model to learn the underlying structure of the dynamic pattern among the characteristics of roadway, traffic, and environment. With the explanatory variables, such as roadway geometric design features, traffic factors, pavement factors, and environmental characteristics as the input, the designed DAE model can identify the nonlinear relationship between the investigated variables in an unsupervised and hierarchical manner and the robust feature representations can be obtained. In addition, the designed DAE model can encode the explanatory variables into an embedding low-dimensional space. The proposed DAE model contains a visible layer, a hidden layer, an output layer, and a reconstruction layer. Unlike the conventional DAE model with K hidden layers [34], the proposed model uses a reconstruction error optimizing the output layer and the noisy input layer to generate higher level representations.

Assume $\Phi = \{[\mathbf{v}_i, \mathbf{u}_i]\}_{i=1}^n$ is a training set that contains n roadway entities, where $\mathbf{v}_i \in \mathfrak{R}^D$ is the explanatory variable vector with dimension D and \mathbf{u}_i is the multivariate traffic safety outcomes for roadway entity i . Given the training set, the proposed DAE model is trained to develop a robust feature representation by reconstructing the

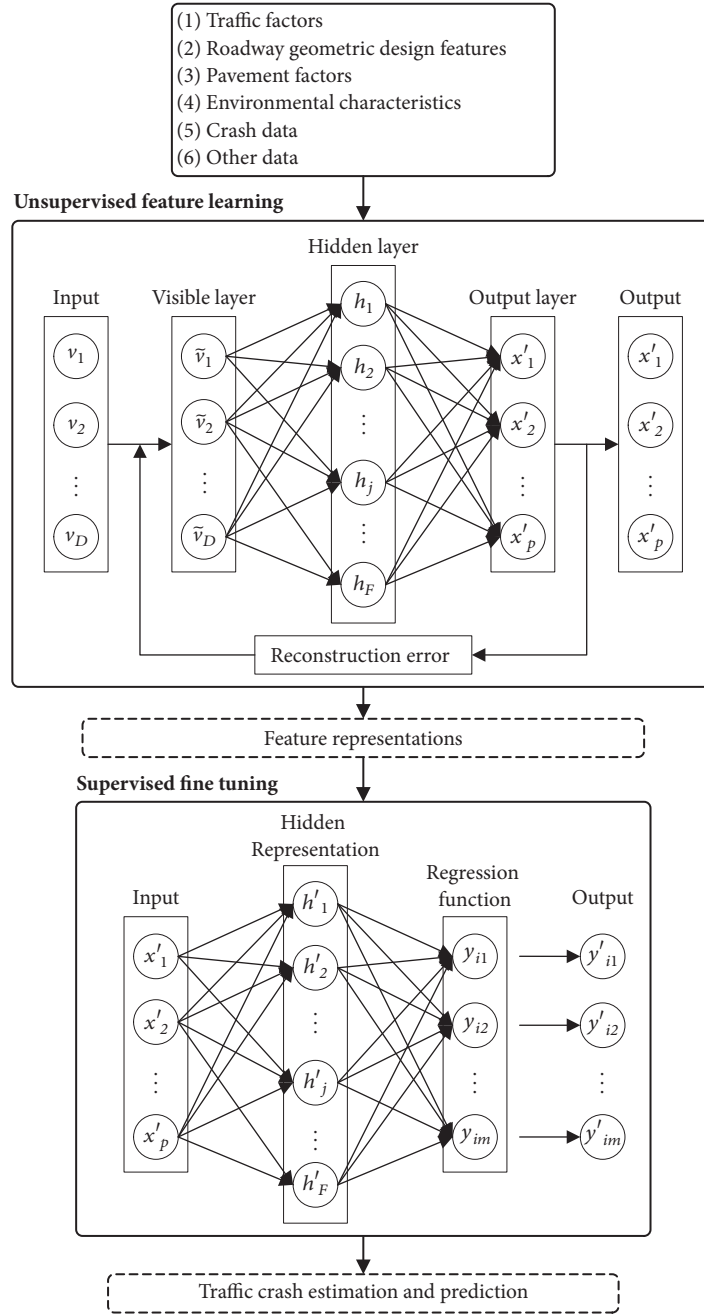


FIGURE 1: Flowchart of the proposed model for traffic crash prediction.

input v_i from its noisy corrupted version \tilde{v}_i , as shown in Figure 1.

There are D units in the visible layer and F units in the hidden layer and the proposed model can be defined by a parameter set $\Theta = \{\mathbf{W}, \mathbf{a}, \mathbf{b}, \mathbf{c}\}$, where $\mathbf{W} = [W_{ij}] \in \mathcal{R}^{D \times F}$ is the interlayer connection weights, $\mathbf{a} = [a_i] \in \mathcal{R}^D$ is the visible self-interactions or biases, $\mathbf{b} = [b_j] \in \mathcal{R}^F$ is the hidden biases, and $\mathbf{c} = [c_k] \in \mathcal{R}^K$ is the reconstruction error. The joint probability distribution between the noisy input variables and

hidden variables is defined as

$$P(\mathbf{v}, \mathbf{h}; \Theta) = \frac{1}{Z(\Theta)} \exp[-\varepsilon(\mathbf{v}, \mathbf{h}; \Theta)] \quad (1)$$

where $\varepsilon(\mathbf{v}, \mathbf{h}; \Theta)$ is an energy function defined by symmetric interactions between the noisy input variable, hidden variables, and a set of interaction parameters Θ ; $Z(\Theta)$ is a normalized factor.

For a binary variable, a Bernoulli-Bernoulli energy function [34, 35] and the conditional distribution of a single stochastic hidden variable are given by

$$\begin{aligned} \varepsilon(\mathbf{v}, \mathbf{h}; \Theta) &= \mathbf{v}^T \mathbf{W} \mathbf{h} - \mathbf{a}^T \mathbf{v} - \mathbf{b}^T \mathbf{h} \\ P(h_j = 1 | \mathbf{v}; \Theta) &= \frac{1}{1 + \exp[-(b_j + \sum_{i=1}^D W_{ij} v_i)]} \end{aligned} \quad (2)$$

For a continuous variable, a Gaussian-Bernoulli energy function [31] and the conditional distribution of a single stochastic hidden variable are given by

$$\begin{aligned} \varepsilon(\mathbf{v}, \mathbf{h}; \Theta) &= -\mathbf{v}^T \mathbf{W} \mathbf{h} + \mathbf{a}^T \mathbf{a}' - \mathbf{b}^T \mathbf{h} \\ P(h_j = 1 | \mathbf{v}; \Theta) &= \frac{1}{1 + \exp[-(b_j + \sum_{i=1}^D (W_{ij} v_i / \sigma_i))]} \end{aligned} \quad (3)$$

where $\mathbf{v}' = (v_1/\sigma_1, v_2/\sigma_2, \dots, v_i/\sigma_i, \dots, v_D/\sigma_D)^T$, $\mathbf{a}' = ((v_1 - a_1)/\sigma_1, (v_2 - a_2)/\sigma_2, \dots, (v_i - a_i)/\sigma_i, \dots, (v_D - a_D)/\sigma_D)$, and σ_i is the standard deviation of the i -th visible variable v_i centered on the bias a_i .

The conditional distribution over hidden units can be factorized and computed by

$$P(\mathbf{h} | \mathbf{v}; \Theta) = \prod_{j=1}^F P(h_j | \mathbf{v}; \Theta) \quad (4)$$

To estimate the parameters, the method proposed by Hjelm et al. [35], which maximizes the log-likelihood of the marginal distribution of the hidden units to find the gradient of the log-likelihood, is employed in this research. To simplify the estimation process, the free energy in terms of the probability at a data point \mathbf{v}_n can be used to replace the energy function and the gradient has the following form:

$$\begin{aligned} \frac{\partial}{\partial \Theta} \sum_{\mathbf{v} \in D} \log p(\mathbf{v}) &= - \sum_{\mathbf{v} \in D} \left\langle \frac{\partial}{\partial \Theta} \varphi(\mathbf{v}, \mathbf{h}; \Theta) \right\rangle_{p(\mathbf{h} | \mathbf{v}; \Theta)} \\ &+ \sum_{\mathbf{v} \in D} \left\langle \frac{\partial}{\partial \Theta} \varphi(\mathbf{v}, \mathbf{h}; \Theta) \right\rangle_{p(\mathbf{v}, \mathbf{h} | \Theta)} \end{aligned} \quad (5)$$

where $\varphi(\mathbf{v}, \mathbf{h}; \Theta) = -\log \sum_{\mathbf{h}} \exp[-\varepsilon(\mathbf{v} = \mathbf{v}_n, \mathbf{h}; \Theta)]$, and the conditional distribution over hidden units should be replaced by a loss function.

$$\varphi(\mathbf{h} | \mathbf{v}_n) = \begin{cases} -\mathbf{a}^T \mathbf{v}_n - \sum_{j=1}^F \log [1 + \exp(\mathbf{v}_n^T W_j + b_j)] & \mathbf{v}_n \text{ is binary} \\ -\|\mathbf{a} - \mathbf{v}_n\|^2 - \sum_{j=1}^F \log [1 + \exp(\mathbf{v}_n^T W_j + b_j)] & \mathbf{v}_n \text{ is continuous} \end{cases} \quad (6)$$

Considering the hidden layer as the input layer for the outcome layer and outcome layer as the hidden layer, the joint probability distribution between the hidden variables and outcome variables, energy function, the conditional distributions of the outcome variables and units, and parameter estimation process can be obtained as those for hidden layer. The model will stop training when c_k satisfies the reconstruction error requirements or the dimension of the feature representation achieves the designed goals.

3.2. Supervised Fine Tuning Module. The supervised fine tuning module is a supervised fine tuning procedure that includes a regression layer on the top of the resulting hidden representation layers to estimate the likelihood of the crash occurrences, as shown in Figure 1. The obtained encoded feature representations from the unsupervised feature module are used as the input for the supervised fine tuning module. To jointly estimate the occurrence likelihood for more than one type of crashes simultaneously and address the potential heterogeneity issues in the interdependent crash data, a multivariate negative binomial (MVNB) model is used in the supervised fine tuning module to estimate and predict the traffic crashes across injury severities. Assume $\mathbf{y}_i = (y_{i1}, y_{i2}, \dots, y_{im})'$ is a vector of crash occurrence likelihood for roadway entities i , which includes m types of crashes. The

particular NB regression model employed in this study has the following form:

$$\begin{aligned} p(y_{ij}) &= \frac{\Gamma(\lambda_{ij}/\sigma + y_{ij})}{\Gamma(\lambda_{ij}/\sigma) \Gamma(y_{ij} + 1)} \left(\frac{1}{1 + \sigma} \right)^{\lambda_{ij}/\sigma} \left(\frac{\sigma}{1 + \sigma} \right)^{y_{ij}} \end{aligned} \quad (7)$$

where $\Gamma(\cdot)$ is the gamma function, y_{ij} is the crash number of crash type j for roadway segment i and $E[y_{ij}] = \lambda_{ij} = \exp(\boldsymbol{\beta}_j \mathbf{X}_{ij} + \varepsilon_{ij})$. $\exp(\varepsilon_{ij})$ is a multivariate gamma-distributed error term with mean 1 and variance α^{-1} .

As described in Shi and Valdez [36] and Anastasopoulos et al. [37], with the expected number of crashes λ_{ij} , the MVNB model has a joint probability function:

$$\begin{aligned} p(y_{1j}, y_{2j}, \dots, y_{ij}) &= \sum_{k=0}^{\min(y_{1j}, y_{2j}, \dots, y_{ij})} p(k) \prod_{i=1}^N p(y_{ij} - k) \end{aligned} \quad (8)$$

The model parameters can be estimated by maximizing the log-likelihood function:

$$L = \sum_{i,j} \left\{ \begin{array}{l} \log [\Gamma(\sigma^{-1} + y_{ij})] - \log [\Gamma(\sigma^{-1})] - \log (y_{ij}!) \\ -\sigma^{-1} \ln [1 + \sigma \exp(\boldsymbol{\beta}_j \mathbf{X}_{ij})] - y_{ij} \log [1 + \sigma^{-1} \exp(\boldsymbol{\beta}_j \mathbf{X}_{ij})] \end{array} \right\} \quad (9)$$

The MVNB regression layer is added on the top of the resulting hidden representation layers to perform traffic crash prediction. This yields a deep learning model tailored to a task-specific supervised learning. Then we fine tune the module 2 using backpropagation by minimizing the following cost function:

$$\begin{aligned} f(\boldsymbol{\theta}) &= -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m s(y'_{ij} = y_{ij}) \log \left[\frac{\exp(h_{\theta}(x_i))}{\sum_{l=1}^F \exp(h_{\theta}(x_i))} \right] \\ &+ \frac{\alpha}{2p} \left[\|W_{regression}\|_F^2 + \sum_{l=1}^F \|W_l\|_F^2 \right] \end{aligned} \quad (10)$$

where $s(\cdot)$ is an indicator function, if $y'_{ij} = y_{ij}$, then $s(\cdot) = 1$; otherwise, $s(\cdot) = 0$, α is a regularization parameter, and $\|\cdot\|_F$ is the Frobenius norm. The first term refers to the cross entropy loss for the regression layer, the second term is the weight decay penalty, and $h_{\theta}(x_i)$ is the output of deep learning for an input x_i .

The cost function is minimized with a min-batch gradient descent algorithm [27]. The parameters in module 2 $\boldsymbol{\theta} = \{\mathbf{W}_l, \mathbf{b}, \mathbf{W}_{regression}\}$ are estimated by initializing the weights $\mathbf{W}_{regression}$ of the regression layer to small random values and the weights of the F hidden layers are initialized by the encoding weights obtained in the unsupervised feature learning module.

4. Data

The data are obtained from the Tennessee Roadway Information Management System (TRIMS) and the Pavement Management System (PMS), which are maintained by the Tennessee Department of Transportation (TDOT). The dataset includes crash data, traffic factors, geometric design features, pavement factors, and environmental characteristics. The traffic, geometric, pavement, and environmental characteristics are linked to the crash data through the common variable *id_number*. An extensive and comprehensive data screening that includes cleaning, consistency, and accuracy checks is processed and performed to ensure the data are useable, reliable, and valid for the analyses. After the initial data screening, in total 635 roadway segments in Knox County are chosen for the analyses. For the selected roadway segments, each of them has a completed dataset that links to the crash data. In other words, the dataset contains detailed information on traffic factors, geometric design features, pavement characteristics, and environment factors.

In TRIMS, the crash data have been classified into five categories, fatal crashes, incapacitating injury crashes, nonincapacitating injury crashes, possible-injury crashes, and PDO

crashes. Because the category of fatal crashes has only a few observations, the categories of fatal crashes and incapacitating injury crashes have been combined and referred to as major injury crashes. The possible-injury crashes and PDO crashes have been combined and referred to as no-injury crashes. The nonincapacitating injury crashes are referred to as minor injury crashes. A few of previous literature [38–40] has used a similar classification for injury outcomes. For those selected 635 roadway segments, from 2010 to 2014, a total of 5365 traffic crashes were reported by the police officers, which include 135 (2.51%) major injury crashes, 1312 (24.46%) minor injury crashes, and 3917 (73.02%) PDO crashes. Individual roadway segment experienced from 0 to 23 crashes per year with a mean of 1.54 and a standard deviation of 1.89. As expected, a significant amount of zeros is observed. The dependent variables and their descriptive statistics are shown in Table 1. The descriptive statistics of continuous independent variables and categorical independent variables are shown in Tables 1 and 2, respectively.

The considered traffic factors include the logarithm of annual average daily traffic (AADT) per lane, truck traffic percentage, and posted speed limits. Roadway segment AADT per lane from 2010 to 2014 varies from 851 to 32,359 vehicles with a mean of 3,388.44 and a standard deviation of 5495.41. Other than the TRIMS dataset maintained by the TDOT, traffic flow information can be obtained from <https://www.tdot.tn.gov/APPLICATIONS/traffichistory>, which is an AADT map providing traffic volumes based on a 24-hour, two-directional count at a given location. The website also provides the traffic history of any specific count station. The variable of posted speed limit has a mean of 38.65 and a standard deviation of 6.69 with a minimum value of 30 and a maximum value of 70. The truck traffic percentage varies from 1 to 33 with a mean of 6.71 and a standard deviation of 4.98.

Important measurements of geometric design features considered in this study include segment length, degree of horizontal curvature, median widths, outsider shoulder widths, number of through lanes, lane widths, number of left-turn lanes, median types, and shoulder type. Among them, the segment length, degree of horizontal curvature, median widths, and outsider shoulder widths are considered as the continuous variables and the others are considered as the categorical variables. Other than the traffic factors and geometric design features, the impacts of pavement surface characteristics are considered to better address traffic safety issues for roadway design and maintenance. The considered pavement surface characteristics include international roughness index (IRI) and rut depth (RD). The analyzed IRI varies from 25.45 to 182.58 with a mean of 65.85 and a standard deviation of 27.75, which is calculated

TABLE 1: Summary statistics of analyzed continuous variables.

Variable	Mean	Std. Dev.	Min.	Max.
Independent variable				
The number of major injury crashes per year per roadway segment	0.04	0.32	0	3
The number of minor injury crashes per year per roadway segment	0.41	0.54	0	9
The number of no injury crashes per year per roadway segment	1.23	1.74	0	22
Traffic factors				
The logarithm of AADT per lane	3.53	3.74	2.93	4.51
Truck traffic percentage	6.71	4.98	1	33
Posted speed limits	38.65	6.69	30	70
Geometric design features				
Segment length (miles)	0.81	1.03	0.02	12.31
Degree of horizontal curvature	1.51	3.23	0	14.00
Median widths	1.12	2.02	0	12
Outside shoulder widths	3.06	1.88	3.52	8
Pavement factors				
International roughness index	65.85	27.75	25.45	182.58
Rut depth (in.)	0.13	0.05	0.06	0.41

TABLE 2: Summary statistics of analyzed categorical variables.

Variable	Category	Frequency	Percent
Geometric design features			
Number of through lanes	3	585	18.43
	2	1660	52.28
	1	930	29.29
Lane widths (ft)	12	915	28.82
	11	1650	51.97
	10	610	19.21
Number of left-turn lanes	2	920	28.98
	1	650	20.47
	0	1605	50.55
Median type	1 for non-traversable median	1185	37.32
	0 for traversable median	1990	62.68
Shoulder type	2 for pavement	760	23.94
	1 for gravel	1595	50.24
	0 for dirt	820	25.83
Environmental factors			
Terrain type	1 for mountainous	1206	37.97
	0 for rolling	1969	62.03
Land use type	2 for residential	1622	51.10
	1 for commercial	775	24.41
	0 for rural	778	24.49
Indicator for lighting	1 for lighting exists on the roadway segments	1304	41.06
	0 for others	1871	58.94

using a quarter-car vehicle math model and the response is accumulated to yield a roughness index with units of slope (in/mi). Another pavement condition indicator is the RD, which is measured at roadway speeds with a laser/inertial profilograph. The analyzed RD varies from 0.06 to 0.41 with a mean of 0.13 and a standard deviation of 0.05.

The environmental factors, including terrain types, lighting condition, and land use type, are considered. Two terrain types are examined, which include rolling terrace (62.03%) and mountainous terrace (37.97%). Lighting condition was considered as a category variable, which indicated whether lighting devices are provided at the roadway segments.

Three types of land use are considered, including commercial (24.41%), rural (24.49%), and residential (51.10). These variables are considered because they might have potential significant effects on traffic safety.

5. Modelling Results

The MATLAB was employed for model development. Four-year data, from 2010 to 2013, were used as the training set and one-year data, the year of 2014, were used as the testing set. In order to obtain the model with superior performance, module 1 was developed using 9 Gaussian visible units, 13 binary visible units, and a number of hyperbolic tangent hidden units ranging from 32 to 128 in steps of 2. The number of hidden units was setting based on two rules, greater than the input data dimensionality and the powers of two. The parameters for learning rate and weight decay were selected to optimize reduction of reconstruction error over training. Module 1 was trained with a sample size of 2540 (four-year crash data) to allow for full convergence of the parameters. The input data were processed by using module 1 to capture the relationship between traffic factors, geometric design features, pavement conditions, and environmental characteristics. Module 2 was developed using 4 Gaussian visible units and a number of hyperbolic tangent hidden units ranging from 8 to 32 was tested and examined. The learning procedure will stop when the number of feature representations is achieved to four or the reconstruction error is less than 0.01.

5.1. Results of Unsupervised Feature Learning Module. For the final model, the reconstruction error is less than 0.01 and four hidden layers are included. The weights between the input layer and output layer can be calculated as $\mathbf{W}_{32 \times 16} \mathbf{W}_{16 \times 8} \mathbf{W}_{8 \times 4} = \mathbf{W}_{32 \times 4}$. The results are shown in Table 3. The negative sign represents a crash-prone condition and a positive sign represents a safe-prone condition. The valued number indicates an evaluation score. Because the feature learning module identifies relational information between input variables and output feature representations, the connection weights between visible units and hidden units can be interpreted as the functional networks [35]. The results show that each of the output units is significant associated with traffic factors, geometric factors, pavement factors, and environmental factors.

Since the proposed feature learning module has symmetric connection between visible and hidden layers and the units in both layers have the probabilistic characteristics, the proposed feature learning module also can be called as an auto-encoder. Thus, the values of output units can be interpreted as a feature representation and the results are shown in Table 4. The number of output units is smaller than the number of visible units, which indicates the feature representation with the values of output units has reduced the dimensionality of the input, but still preserving the original information. Hence the four output units are defined as traffic feature representation, geometric feature representation,

pavement feature representation, and environmental feature representation.

The results show that the signs of the means of four feature representations are negative, which indicate that the feature representations are associated with crash-prone condition. In other words, the current traffic, geometric, pavement, and environmental features are the main factors that attribute to the risk of crash occurrences. The findings indicate that the traffic, geometric, pavement, and environmental factors have a direct influence on traffic safety and need to be improved.

The traffic feature representations have a wide range with a minimum value of -4.420 and a maximum value of 6.650, which indicate the traffic factors have a significant impact on traffic safety. The pavement factors have comparative impacts on traffic safety with a minimum value of -12.076 and a maximum value of 6.753. The ranges of geometric and environmental feature representations are from -1.386 to 1.280 and from -1.805 to 0.847, respectively, which indicate that the geometric and environmental feature representations have comparative effects on traffic safety. Compared to those crash modeling techniques that use only geometric design features as the input factors, the current research reveals new insights that would benefit the development of updated guidelines.

5.2. Results of the Supervised Fine Tuning Module. For the supervised fine tuning module, the visible units of the input layer use the feature representations as the input and the crash counts across injury severities are used as the training target. The aggregate weights between the input layer and output layer are shown in Table 5. The results show that the traffic and geometric feature representations have positive effects and pavement and environmental feature representations have negative effects on traffic crashes across injury severities. The findings indicate that decreasing the values of traffic and geometric feature representations will increase the likelihood of crashes and increasing the values of geometric and environmental feature representations will increase the likelihood of crashes. The comparison results show that traffic feature representation and geometric feature representation have significant impact on PDO crashes. The pavement feature representation and environmental feature representation have significant impact on minor injury crashes. Among four feature representations, the geometric feature representation, pavement feature representation, and traffic feature representation have most direct impacts on major injury, minor injury, and PDO crashes, respectively.

Considering the data in the year of 2014 as the input variables, the developed deep learning model is used to predict the crash counts for the year of 2014. To validate the superiority of the proposed models, the predicted results are compared to the observed values. In addition, a deep learning model without the regression layer and a support vector machine (SVM) model are also developed to predict crash counts across injury severities. There are two key issues related to the development of SVR models, kernel selection, and parameters optimization. To address the nonlinear relationship between the outcomes and attributes, the commonly

TABLE 3: The functional network between the input variables and the feature representations.

Variable	Traffic feature representation	Geometric feature representation	Pavement feature representation	Environmental feature representation
Traffic factor				
The logarithm of AADT per lane	-0.529	-0.007	-0.016	-0.034
Truck traffic percentage	-0.659	-0.002	-0.024	-0.010
Posted speed limits	-0.030	-0.010	0.175	-0.047
Geometric factors				
Segment length (miles)	0.007	-0.580	-0.007	-0.003
Degree of horizontal curvature	0.002	-0.918	-0.006	-0.009
Median widths	0.002	0.826	0.068	0.015
Outside shoulder widths	0.005	0.419	0.016	0.064
Number of through lanes				
3 lanes	-0.021	-0.891	-0.006	-0.026
2 lanes	-0.015	-0.864	-0.022	-0.060
Lane widths				
12 ft	0.004	0.776	0.005	0.001
11 ft	0.009	0.592	0.076	0.004
Number of left-turn lanes				
2 left-turn lanes	-0.003	-0.212	-0.019	-0.017
1 left-turn lane	0.002	-0.783	0.028	0.016
Median type				
Non-traversable median	0.013	0.247	0.006	0.007
Shoulder type				
Pavement	-0.035	-0.144	-0.027	-0.016
Gravel	0.005	-0.743	0.011	0.060
Pavement factor				
International roughness index (in./mile)	0.101	0.007	-0.183	-0.001
Rut depth (in.)	-0.004	-0.004	-0.341	-0.038
Environmental factor				
Terrain type				
Mountainous	-0.014	-0.011	-0.010	-0.954
Land use type				
Residential	0.042	0.015	0.011	0.847
Commercial	0.058	0.001	0.006	-0.915
Indicator for lighting				
Lighting exists on roadway	0.018	0.008	0.027	0.781

TABLE 4: Summary statistics of feature representations.

Statistics	Mean	Std. Dev.	Min.	Max.
Traffic feature representation	-0.697	1.464	-4.420	6.650
Geometric feature representation	-1.620	0.521	-1.386	1.280
Pavement feature representation	-5.388	2.441	-12.076	6.753
Environmental feature representation	-2.167	0.442	-1.805	0.847

TABLE 5: The functional network between the feature representation and different crash types.

Variable	Major injury crashes	Minor injury crashes	PDO crashes
Traffic feature representation	-0.109	-0.421	-1.535
Geometric feature representation	0.129	0.261	0.539
Pavement feature representation	-0.033	-0.872	-0.315
Environmental feature representation	0.113	0.298	0.146

used RBF kernel is chosen to develop the SVM model. To precisely reflect the performance on regressing unknown data and prevent the overfitting problem, the k -fold cross-validation approach is employed for optimizing the two parameters in RBF kernels— C and ϵ [41, 42]. The optimized parameters are (42.30, 37.09) and the optimized SVM model yields to a training mean absolute error (MAE) less than 0.01 and a training root-mean-squared deviation (RMSD) less than 5%.

To evaluate the accuracy and reliability of the prediction results, the predicted means for each injury severity level by the proposed model, the deep learning model without the regression layer, and the SVM model are compared to the observed means and two evaluation measures are used, which include MAE and RMSD. The comparison and validation results are shown in Table 6. Results in Table 6 indicate that the predicted means from the proposed model (0.110, 0.573, 1.335, and 2.019 for major injury, minor injury, no-injury, and all crashes, respectively) are very close to the observed means (0.096, 0.556, 1.306, and 1.986). We believe that a very importation feature of the proposed model is that the included regression layer can provide a good estimate of the chance that the roadway segment is in the crash-free state or some crash-prone propensity states.

For all the observed samples, the proposed model results in a MAE of 0.030, 0.080, 0.071, and 0.150 and a RMSD of 17.298%, 29.961%, 27.206%, and 43.652% for major injury, minor injury, PDO, and all crashes, respectively. The deep learning mode without the regression layer results in a MAE of 0.043, 0.202, 0.405, and 0.520 and a RMSD of 20.620%, 51.741%, 65.086%, and 82.862% for major injury, minor injury, PDO, and all crashes, respectively. The SVM model results in a MAE of 0.055, 0.257, 0.471, and 0.660 and a RMSD of 26.022%, 61.350%, 72.416%, and 96.636% for major injury, minor injury, PDO, and all crashes, respectively. The results suggest that the proposed model predicts better than the deep learning model without the regression layer and the SVM model. In summary, compared to the deep learning model without the regression layer and the SVM model, the

proposed model results in the smallest prediction MAE and RMSD, no matter for the crash types across injury severities. We hypothesize that the proposed model better addresses the issue of heterogeneity and allows for excess zero counts in correlated data.

The findings indicate that the predictions from the proposed model have significant improvements over all comparison models, in both accuracy and robustness. The best-performing result of the proposed model for major injury crashes has a MAE of 0.030, which indicates an 42.105% and 84.211% improvement from the deep learning model without the regression layer and the SVM model, respectively. The proposed models perform worse for the minor injury crashes, with a 0.080 RMSD for all observed samples. However, it is still better than the evaluation measurements from the deep learning model without the regression layer and the SVM model, which are 0.202 and 0.257, respectively. This represents a MAE improvement of 150.980% and 219.608% for minor injury crash prediction. For the PDO crashes, the MAE improvements of the proposed models over the comparison models range from 471.111% to 564.444%. For all the observed samples, the MAE improvements range from 247.368% to 341.053%. Clearly, the improvement is significant for the traffic crash predictions. Therefore, the proposed model seems to be a better alternative for crash count predictions.

The proposed model has better performances in terms of small error variances than the comparison models, since the regression model is imbedding into the proposed model. The overall performances of the proposed model for all crashes show an 89.824% RMSD improvement over the deep learning mode without regression layer and an 121.378% RMSD improvement over the SVM model. It is clear that the predictions obtained from the proposed models are superior to those obtained from the comparison models. The greatest difference is demonstrated for the PDO crashes where the proposed model yields a RMSD of 27.206% compared to a 65.086% RMSD value from the deep learning model without the regression layer and a 72.416% RMSD value from the

TABLE 6: Results of roadway segment crash count prediction.

	Major injury	Minor injury	No injury	Total
Observation mean	0.096	0.556	1.306	1.986
Proposed model				
Estimated mean	0.110	0.573	1.335	2.019
MAE	0.030	0.080	0.071	0.150
RMSD (%)	17.298	29.961	27.206	43.652
Deep learning model without the regression layer				
Estimated mean	0.101	0.556	1.332	1.989
MAE	0.043	0.202	0.405	0.520
RMSD (%)	20.620	51.741	65.086	82.862
SVM model				
Estimated mean	0.101	0.513	1.329	1.943
MAE	0.055	0.257	0.471	0.660
RMSD (%)	26.022	61.350	72.416	96.636

SVM model. The differences in the proposed model and the SVM model for the minor injury crashes are also significant (29.961% versus 61.350%).

6. Conclusions

Because traffic crashes are a big concern of the public, agencies, and policy makers and result in countless fatalities and injuries, there is a need to perform a comprehensive analysis that aims to understand the relationship between the influence factors and traffic crash outcomes. In this study, we presented an innovative approach for traffic crash prediction. Methodologically, we demonstrated a novel deep learning technique embedded within a multivariate regression model can be used to identify relationship between the examined variables and the traffic crashes. Future applications of this approach have the potential to provide insights into basic questions regarding roadway spatial and temporal dynamic function and practical questions regarding countermeasures. The investigation results provide sufficient evidence for the following conclusions:

(1) The results show that the feature learning module identifies relational information between input variables and output feature representations. The findings indicate that the feature representations have reduced the dimensionality of the input, but still preserving the original information.

(2) The proposed model that includes a MVNB regression layer in the supervised fine tuning module can better account for different patterns in crash data across injury severities and provide superior traffic crash predictions. In addition, the proposed model can perform multivariate estimations simultaneously with a superior fit.

(3) The proposed model has superior performances in terms of prediction power compared to the deep learning model without a regression layer and the SVM model. The overall performances of the proposed model for all crashes show an 89.824% RMSD improvement over the deep learning model without a regression layer and an 121.378% RMSD improvement over the SVM model.

(4) The findings suggest that the proposed model is a superior alternative for traffic crash predictions. The proposed model can better account for heterogeneity issues in traffic crash prediction.

The proposed models can perform traffic crash prediction for a given facility. The proposed methodology could be applied to other roadway networks if appropriate attribute variables are available. Traffic and transportation engineering agencies can employ the proposed models with relative cases and develop them to their needs to obtain traffic crash predictions for various time periods. Further investigation of the proposed models includes the predictions of spatial-temporal dynamic pattern in crash data.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Special thanks are due to TDOT for providing the TRIMS data. This research is supported by funding provided by the Southeastern Transportation Center—a Regional UTC funded by the USDOT—Research and Innovative Technology Administration. Additional funding was provided by the National Natural Science Foundation of China (Grant Nos. 51678044, 51338008, 71621001, 71210001, and 71501011).

References

- [1] NHTSA, *Motor Vehicle Traffic Crashes as a Leading Cause of Death in the United States, 2012-2014*, Department of Transportation, National Highway Traffic Safety Administration, Traffic Safety Facts, DOT HS 812 297, 2016.

- [2] NHTSA, *2015 Motor Vehicle Crashes: Overview*, U.S. Department of Transportation, National Highway Traffic Safety Administration, Traffic Safety Facts, DOT HS 812 318, 2016.
- [3] D. Lord and F. Mannering, "The statistical analysis of crash-frequency data: a review and assessment of methodological alternatives," *Transportation Research Part A: Policy and Practice*, vol. 44, no. 5, pp. 291–305, 2010.
- [4] F. L. Mannering and C. R. Bhat, "Analytic methods in accident research: methodological frontier and future directions," *Analytic Methods in Accident Research*, vol. 1, pp. 1–22, 2014.
- [5] F. L. Mannering, V. Shankar, and C. R. Bhat, "Unobserved heterogeneity and the statistical analysis of highway accident data," *Analytic Methods in Accident Research*, vol. 11, pp. 1–16, 2016.
- [6] J. C. Milton, V. N. Shankar, and F. L. Mannering, "Highway accident severities and the mixed logit model: an exploratory empirical analysis," *Accident Analysis & Prevention*, vol. 40, no. 1, pp. 260–266, 2008.
- [7] C. Dong, D. B. Clarke, X. Yan, A. Khattak, and B. Huang, "Multivariate random-parameters zero-inflated negative binomial regression model: an application to estimate crash frequencies at intersections," *Accident Analysis & Prevention*, vol. 70, pp. 320–329, 2014a.
- [8] C. Dong, D. B. Clarke, S. S. Nambisan, and B. Huang, "Analyzing injury crashes using random-parameter bivariate regression models," *Transportmetrica A: Transport Science*, vol. 12, no. 9, pp. 794–810, 2016.
- [9] M. G. Karlaftis and E. I. Vlahogianni, "Statistical methods versus neural networks in transportation research: differences, similarities and some insights," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 3, pp. 387–399, 2011.
- [10] J. C. Principe, N. R. Euliano, and W. C. Lefebvre, *Neural and Adaptive Systems: Fundamentals through Simulations*, vol. 672, Wiley, New York, NY, USA, 2000.
- [11] L.-Y. Chang, "Analysis of freeway accident frequencies: Negative binomial regression versus artificial neural network," *Safety Science*, vol. 43, no. 8, pp. 541–557, 2005.
- [12] H. T. Abdelwahab and M. A. Abdel-Aty, "Artificial neural networks and logit models for traffic safety analysis of toll plazas," *Transportation Research Record*, no. 1784, pp. 115–125, 2002.
- [13] Y. Xie, D. Lord, and Y. Zhang, "Predicting motor vehicle collisions using Bayesian neural network models: an empirical analysis," *Accident Analysis & Prevention*, vol. 39, no. 5, pp. 922–933, 2007.
- [14] M. M. Kunt, I. Aghayan, and N. Noii, "Prediction for traffic accident severity: Comparing the artificial neural network, genetic algorithm, combined genetic algorithm and pattern search methods," *Transport*, vol. 26, no. 4, pp. 353–366, 2011.
- [15] K. S. Jadaan, M. Al-Fayyad, and H. F. Gammoh, "Prediction of Road Traffic Accidents in Jordan using Artificial Neural Network (ANN)," *Journal of Traffic and Logistics Engineering*, vol. 2, no. 2, pp. 92–94, 2014.
- [16] D. Akin and B. Akbaş, "A neural network (NN) model to predict intersection crashes based upon driver, vehicle and roadway surface characteristics," *Scientific Research and Essays*, vol. 5, no. 19, pp. 2837–2847, 2010.
- [17] X. Li, D. Lord, Y. Zhang, and Y. Xie, "Predicting motor vehicle crashes using Support Vector Machine models," *Accident Analysis & Prevention*, vol. 40, no. 4, pp. 1611–1618, 2008.
- [18] Y. Zhang and Y. Xie, "Forecasting of short-term freeway volume with v-support vector machines," *Transportation Research Record*, vol. 2024, no. 1, pp. 92–99, 2007.
- [19] V. Kecman, "Support Vector Machines – An Introduction," in *Support Vector Machines: Theory and Applications*, vol. 177 of *Studies in Fuzziness and Soft Computing*, pp. 1–47, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [20] R. Yu and M. Abdel-Aty, "Analyzing crash injury severity for a mountainous freeway incorporating real-time traffic and weather data," *Safety Science*, vol. 63, pp. 50–56, 2014.
- [21] C. Chen, G. Zhang, Z. Qian, R. A. Tarefder, and Z. Tian, "Investigating driver injury severity patterns in rollover crashes using support vector machine models," *Accident Analysis & Prevention*, vol. 90, pp. 128–139, 2016.
- [22] N. Dong, H. Huang, and L. Zheng, "Support vector machine in crash prediction at the level of traffic analysis zones: assessing the spatial proximity effects," *Accident Analysis & Prevention*, vol. 82, pp. 192–198, 2015.
- [23] G. Ren and Z. Zhou, "Traffic safety forecasting method by particle swarm optimization and support vector machine," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10420–10424, 2011.
- [24] R. Yu and M. Abdel-Aty, "Utilizing support vector machine in real-time crash risk evaluation," *Accident Analysis & Prevention*, vol. 51, pp. 252–259, 2013.
- [25] M. Abdel-Aty and K. Haleem, "Analyzing angle crashes at unsignalized intersections using machine learning techniques," *Accident Analysis & Prevention*, vol. 43, no. 1, pp. 461–470, 2011.
- [26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [27] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.
- [28] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191–2201, 2014.
- [29] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic Flow Prediction with Big Data: A Deep Learning Approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.
- [30] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1–17, 2017.
- [31] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [32] P. Swietojanski, A. Ghoshal, and S. Renals, "Convolutional neural networks for distant speech recognition," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1120–1124, 2014.
- [33] M. M. A. Rahhal, Y. Bazi, H. Alhichri, N. Alajlan, F. Melgani, and R. R. Yager, "Deep learning approach for active classification of electrocardiogram signals," *Information Sciences*, vol. 345, pp. 340–354, 2016.
- [34] H.-I. Suk, C.-Y. Wee, S.-W. Lee, and D. Shen, "State-space model with deep learning for functional dynamics estimation in resting-state fMRI," *NeuroImage*, vol. 129, pp. 292–307, 2016.
- [35] R. D. Hjelm, V. D. Calhoun, R. Salakhutdinov, E. A. Allen, T. Adali, and S. M. Plis, "Restricted Boltzmann machines for neuroimaging: An application in identifying intrinsic networks," *NeuroImage*, vol. 96, pp. 245–260, 2014.

- [36] P. Shi and E. A. Valdez, "Multivariate negative binomial models for insurance claim counts," *Insurance: Mathematics & Economics*, vol. 55, pp. 18–29, 2014.
- [37] P. C. Anastasopoulos, V. N. Shankar, J. E. Haddock, and F. L. Mannering, "A multivariate tobit analysis of highway accident-injury-severity rates," *Accident Analysis & Prevention*, vol. 45, pp. 110–119, 2012.
- [38] L. Chang and J. Chien, "Analysis of driver injury severity in truck-involved accidents using a non-parametric classification tree model," *Safety Science*, vol. 51, no. 1, pp. 17–22, 2013.
- [39] J. Pahukula, S. Hernandez, and A. Unnikrishnan, "A time of day analysis of crashes involving large trucks in urban areas," *Accident Analysis & Prevention*, vol. 75, pp. 155–163, 2015.
- [40] Q. Wu, F. Chen, and G. H. Zhang, "Mixed logit model-based driver injury severity investigations in single- and multi-vehicle crashes on rural two-lane highways," *Accident Analysis & Prevention*, vol. 72, pp. 105–115, 2014.
- [41] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, Cambridge, UK, 2000.
- [42] C. Campbell, "Kernel methods: A survey of current techniques," *Neurocomputing*, vol. 48, pp. 63–84, 2002.

Research Article

A SVM Approach of Aircraft Conflict Detection in Free Flight

Xu-rui Jiang,^{1,2} Xiang-xi Wen ,^{1,2} Ming-gong Wu,^{1,2} Ze-kun Wang,^{1,2} and Xi Qiu³

¹Air Traffic Control and Navigation College, Air Force Engineering University, Xi'an 710051, China

²National Key Laboratory of Air Traffic Collision Prevention, Xi'an 710051, China

³Nuclear Power Institute of China, Chengdu 610051, China

Correspondence should be addressed to Xiang-xi Wen; wxxajy@163.com

Received 2 May 2018; Accepted 13 November 2018; Published 6 December 2018

Guest Editor: Ali Tizghadam

Copyright © 2018 Xu-rui Jiang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Probabilistic conflict detection methods typically require high computational burden to deal with complex multi-aircraft conflict detection. In this article, aircraft conflict detection is considered as a binary classification problem; therefore, it can be solved by a pattern recognition method. A potential conflict would be identified, as long as its flight data features are extracted and fed to a classifier which has been trained by a large number of flight datasets. Based on this, a new method based on support vector machine (SVM) is employed to detect multi-aircraft conflict in "Free Flight" airspace and to estimate the conflict probability. For that purpose, the current positions, velocity vectors, and predicted look-ahead time are selected as detection factors, and the detection model is established by SVM to detect aircraft conflict within look-ahead time during short and medium terms. Moreover, conflict probabilities are determined by the sigmoid function mapping method. Nevertheless, false alarm rate is always a first and foremost problem that troubles air traffic controllers. For the purpose of reducing false alarm rates, Synthetic Minority Over-sampling Technique (SMOTE) method is used to handle imbalanced datasets. Extensive simulation results are presented to illustrate the rationality and accuracy of this method.

1. Introduction

Nowadays, the demand for air travel continues to grow at a rapid rate with air traffic becoming more and more complex. In this case, collision avoidance (CA), whose task is to maintain a separation between aircraft in air traffic management (ATM), is challenged by increased flight flow. In future, a number of automated decision support technologies, such as Conflict Detection and Resolution (CDR), will be required to enable continued provision of safe and efficient services in increasingly congested skies. Particularly in Free Flight, pilots will have the freedom to choose their trajectory and speed in real time to maximize their flight objectives while maintaining safe separation from neighboring traffic [1], therefore, seeking a fast conflict detection method in Free Flight, which is suitable in both the terminal areas and en-route areas, is extremely significant.

Conflict prediction is based on inexact trajectory prediction [2, 3]. Deterministic trajectory of an aircraft is influenced by a large number of parameters, particularly due to the

wind and also due to the tracking, navigation, and control error, making the trajectory prediction inexact, and random patterns have to be considered. That makes conflict detection become a probability problem, and a number of conflict detection methods based on stochastic process have been proposed. Russel et al. [4] presented a method to estimate mid-term conflict probability of a pair of aircrafts for level flight; in his paper, a coordinate transformation was used to derive an analytical solution. They extended this method to nonlevel flight in [5]. Hu [6] proposed a short-term conflict detection algorithm based on Brownian motion (BM) for level flight. D. Li [7] used J. Hu's [6] algorithm and improved the algorithm to reduce the rate of false alarms. Jilkov et al. [8] proposed a more accurate method for estimating the conflict probability by utilizing the information from multiple model aircraft trajectory prediction. However, previous work on multi-aircraft conflict detection in Free Flight is still rare. It is shown in H. Blom's research [9] that the interacting particle system (IPS) algorithm is efficient when more complex air traffic scenarios and more advanced conflict

zones have to be taken into account. Damien Jacquemart et al. [10] proposed an adaptive algorithm to significantly increase the convergence rate. Moreover, some hierarchical IPS algorithms proposed in research are considered for Free Flight modeling described by advanced stochastic Petri nets [11]. In 2013, Damien Jacquemart [12] also used Markov chain model to simulate aircraft trajectories and applied the important splitting method to estimate conflict probabilities to overcome the flaw of Monte Carlo method which is not efficient to estimate small probabilities. In 2015, Qiao et al. [13] proposed Hidden Markov model-based Trajectory Prediction (HMTP) method to overcome the difficulty of describing the position and behavior of moving objects in a network-constraint environment. Yang et al. [14] proposed a probabilistic reachability analysis approach. In particular, ellipsoidal probabilistic reach sets are determined by formulating a chance-constrained optimization problem and solving it via a simulation-based method called scenario approach. Conflict detection is then performed by verifying if the ellipsoidal reach sets of different aircraft intersect. Their method gives this paper a great inspiration. However, most of the above methods have a common shortcoming, which is every pair of aircraft conflict detection requires a corresponding trajectory prediction.

Actually, the essence of conflict detection is a two-class classification problem: conflict or nonconflict. In this paper, we introduce SVM to solve the problem of aircraft conflict detection. Support vector machine (SVM) is an excellent two-class classification algorithm, which has been successfully applied in many domains ranging from digit recognition [15] and face recognition [16] to network anomaly detection [17] because of solid theoretical foundation and appealing classification performance. However in air traffic management operation, air traffic controllers (ATCs) expect conflict probability to estimate collision risk and to direct the pilots to avoid the conflicts according to the conflict levels. Standard SVM is a binary classifier: for sample x , the output of the SVM, is $+1/-1$. Some researchers [18, 19] modified the SVM outputs into posterior probabilities, and this method is widely applied in many areas [20, 21]. The calibrated posterior probabilities still inherit the sparseness of the SVM; moreover, they can provide probabilistic prediction decisions. In this paper, we choose Platt's method to evaluate the conflict probability. On these bases, aircraft conflict during short-term to medium-term look-ahead time is predicted, which is suitable for the case of uncertain trajectory prediction error in Free Flight. At the same time, conflict probability can be estimated for ATCs to make a quick decision.

The remainder of the paper is organized as follows. In Section 2, some basic theories used in this paper: the SVM classifier, Platt's probabilistic output, and ellipsoidal protected zone model are introduced. The whole conflict detection process based on SVM is presented. In Section 3, trajectory prediction model based on Brownian motion is introduced to simulate different flight scenarios. Conflict detection experiments on these scenarios are worked out to verify the proposed method. Finally, we conclude in Section 4.

2. Probabilistic Conflict Detection Model Based on SVM

2.1. SVM Classifier. Support Vector Machine is a statistical learning method with a good performance [22–27]. Given a dataset D in the form of $\{\mathbf{x}_j, y_j\}_{j=1}^N$, standard SVM for the binary classification problem maps the feature vector $\mathbf{x} \in R^d$ into a high (possibly infinite) dimensional Euclidean space, H , using a nonlinear mapping function $\Phi: R^d \rightarrow H$. The goal of support vector machines is to find the optimal separating hyperplane $\mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0$, which maximizes the margin, and it can be obtained by solving the convex optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \left(\sum_{i=1}^n \xi_i \right) \\ \text{s.t.} \quad & y_i [\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b] - 1 + \xi_i \geq 0 \\ & \xi_i \geq 0, \quad i = 1, \dots, n \end{aligned} \quad (1)$$

over $\mathbf{w} \in H, b \in R$, and the nonnegative slack variable $\xi \in R^N$. In the above, C is a parameter that balances the size of \mathbf{w} and the sum of ξ_i . It is well known that the numerical computation of Problem (1) is achieved through its dual formulation. Suppose α_i is the Lagrange multiplier corresponding to the i^{th} inequality, then the dual of (1) can be shown to be

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} \quad & \sum_{i=1}^n y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, n \end{aligned} \quad (2)$$

where the kernel function $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ and

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \Phi(\mathbf{x}_i) \quad (3)$$

With (3), the expression of the hyperplane $\mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0$ can be changed into

$$f(\mathbf{x}) = \sum_{i=1}^N y_i \alpha_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) + b \quad (4)$$

and serves as the decision function. The predicted class can be defined as

$$Y = \text{sgn}[f(\mathbf{x})] \quad (5)$$

The predicted class is $+1$ if $f(\mathbf{x}) > 0$ and -1 otherwise.

2.2. Platt's Probabilistic Output. Air traffic controllers (ATCs) expect conflict probability to estimate collision risk; however, standard SVM produces an uncalibrated value that is not a probability. Platt [18] came over this shortcoming and

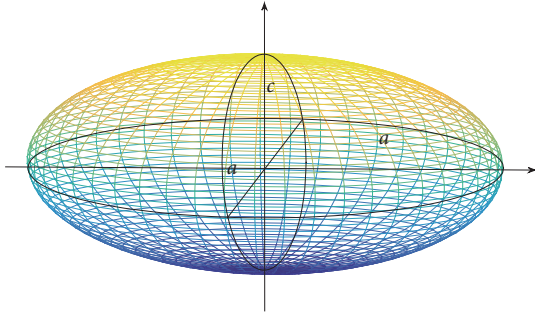


FIGURE 1: A ellipsoidal protected zone.

mapped $f(\mathbf{x})$ into $p(c | \mathbf{x})$ through sigmoid function, providing probabilistic information from standard SVM output. Suppose N_+ and N_- are the numbers of positive ($y = +1$) and negative ($y = -1$) samples, respectively, in dataset D . The Platt's probability output [28] is

$$\hat{p}(c | \mathbf{x}) = \frac{1}{1 + \exp(Af(\mathbf{x}) + B)} \quad (6)$$

where $f(\mathbf{x})$ is the SVM output given by (4) and the parameters A and B are found by minimizing the negative log likelihood of the training data, which is a cross-entropy error function, in the form of

$$\begin{aligned} & \min F(A, B) \\ & = \min \left\{ -\sum_i [t_i \log \hat{p}(c | \mathbf{x}_i) + (1 - t_i) \log (1 - \hat{p}(c | \mathbf{x}_i))] \right\} \quad (7) \end{aligned}$$

with $t_i = (N_+ + 1)/(N_+ + 2)$ if $y_i = +1$ and $t_i = 1/(N_- + 2)$ if $y_i = -1$. Hereafter, $\hat{p}(c | \mathbf{x})$ refers to the estimated posterior probability belonging to class +1 given \mathbf{x} obtained from (7), while $\tilde{p}(c | \mathbf{x})$ refers to the true but typically unknown posterior probability belonging to class c given \mathbf{x} .

2.3. Ellipsoidal Protected Zone Model. The SVM needs training samples to establish the detection model. To identify the potential conflicts, ellipsoidal protected zone is introduced. In Free Flight, aircrafts are surrounded by an imaginary cylindrical volume called the protected zone (PZ). The radius of the PZ is half (2.5 NM) the required horizontal separation standards and the height is equal to the vertical separation (1000 ft). The two aircrafts are said to be in conflict if their PZs overlap. In this paper, an imaginary cylindrical volume is replaced by an ellipsoidal-shaped conflict volume [29, 30], which can be shown in Figure 1.

In this model, we have adopted a slightly modified definition where only the target aircraft is surrounded by the PZ whose dimensions are twice than those defined in the literature [22]. In simplified terms the conflict is identified when potential conflict aircraft enters the PZ of target aircraft within the look-ahead time, and the conflict area is constructed as the following formula:

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{a^2} + \frac{(z - z_0)^2}{c^2} \leq 1 \quad (8)$$

where (x_0, y_0, z_0) are the coordinates of aircraft located in the ellipsoid center and (x, y, z) are the coordinates of potential conflict aircraft. Set ellipsoidal equatorial radius $a = 5$ NM and ellipsoidal polar radius $c = 2000$ ft.

The above models are used to judge conflict within look-ahead time: in the whole time window from current time t_0 to look-ahead time t . If there is a condition that the distance between a pair of aircraft does not satisfy the protected zone constraint, the conflict occurs. A number of researches have certified that the above model has an effect on actual scenarios during a period of 20 min because of accumulated error, and this model is applicable for conflict in short-term or medium-term [6, 31]. Thus, the datasets are selected within 20 min.

2.4. Model Establishment. According to above preliminaries, SVM is adopted to classify conflict conditions into two classes $\{-1, +1\}$, -1 referring to "conflict" and +1 to "nonconflict", respectively. A trained C-SVM model is encapsulated as an application module before flight to predict conflict and estimate probability. The flow chart is shown in Figure 2.

Overall, the main process of conflict detection can be divided into three steps:

(1) **Data Acquisition.** In conflict detection, data acquisition and feature extraction are the fundamental parts. A pair of aircraft position coordinates and velocity vectors are collected: Aircraft A and B's current position coordinates $\vec{X}_a = (x_a, y_a, z_a)$, $\vec{X}_b = (x_b, y_b, z_b)$ and Aircraft A and B's velocity vectors $\vec{v}_a = (v_x, v_y, v_z)$, $\vec{v}_b = (v'_x, v'_y, v'_z)$. The features are extracted according to the importance of representing flight condition; the features are as follows: the relative distance of the pair of aircraft $\vec{X}_r = \vec{X}_b - \vec{X}_a = (x_r, y_r, z_r)$, the relative velocity vector $\vec{V}_r = \vec{v}_b - \vec{v}_a = (v_{xr}, v_{yr}, v_{zr})$, and predicted look-ahead time t . Thus, the feature dimension is reduced to $D = 7$:

$$(x_r, y_r, z_r, v_{xr}, v_{yr}, v_{zr}, t) \quad (9)$$

(2) **Data Processing.** In actual flight scenario, aircraft conflicts are small probability events, so that the positive samples predominate in collected flight data. In SVM classification problem, when the distribution of the training data among classes is imbalanced, the learning algorithm is generally dominated by the majority classes. The minority classes are normally difficult to be fully recognized. However, conflict samples (negative samples) are quiet important in this problem. In this paper, Synthetic Minority Over-sampling Technique (SMOTE) algorithm [32, 33] is applied to handle imbalanced data. The specific approach is to insert virtual negative samples in adjacent negative samples in order to enhance the classification accuracy for the minority classes.

In addition, there are considerable magnitude differences among the 7 indicators. Min-Max Normalization performs a linear transformation on the original data x into the

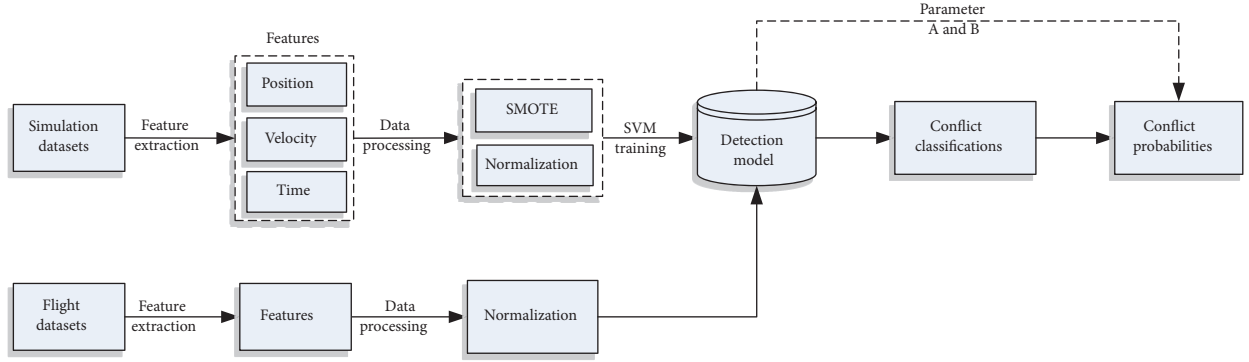


FIGURE 2: Probabilistic conflict detection by SVM flow graph.

specified interval $[-1, +1]$, and the normalization equation is as follows:

$$Y = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (10)$$

where X is feature value, X_{\min} and X_{\max} are minimum and maximum values of data, and $Y \in [-1, 1]$ is normalized feature value.

(3) *Training and Prediction.* On the bases of features and labels above, SVM is applied to establish the detection model, in which 1000 samples are trained. Afterwards, parameters A and B are obtained by formula (7). In conflict prediction process, extracted features from flight data are inputted to predict overall conflict in look-ahead time. At the same time, conflict probabilities are estimated by sigmoid function mapping method.

3. Simulations and Discussion

To establish and verify the detection model, we need training samples and testing scenarios. In this section, trajectory prediction model based on Brownian motion is introduced to simulate different flight scenarios and acquire training datasets. In training process, 1000 groups of simulation datasets (half positive samples and half negative samples) are used to train the detection model, and actual flight datasets are used for test. Conflict detection experiments in these scenarios are worked out to verify the proposed method.

3.1. Trajectory Simulation Based on Brownian Motion. Traditionally, deterministic trajectory of an aircraft is governed by flight mechanic equations. However, a large number of parameters are involved, particularly due to the wind, tracking, navigation, and control error, which makes the trajectory prediction inexact. Here we choose the three-dimensional Brownian motion [6, 31] as the trajectory prediction model.

A stochastic model is worked out for aircraft trajectories with constant speed following a straight line. Observations presented in [34] show that aircraft position error for a given time is set with Gaussian distribution. The perturbation in

position can be treated as a scaled Brownian motion (BM) [6]. One can then assume that

$$\frac{dX(t)}{dt} = u(t) + \omega(t) \quad (11)$$

where $X(t)$ is the position of the aircraft at time t , $u(t)$ is a three-dimensional velocity vector, and $\omega(t)$ is random variables following Gaussian distribution with zero mean. Now, if there is an angle $\theta(t)$ at time t between the projecting line of the velocity vector in horizontal plane and x -axis, the predicted position is given by

$$dX(t) = u(t) dt + R(\theta(t)) \Sigma dB(t) \quad (12)$$

where Σ is the variance growth rate. More precisely, $\Sigma = \text{diag}(\sigma_x, \sigma_y, \sigma_z)$ models the variance growth rate in the along-track, cross-track, and vertical direction. $B(t)$ is a standard three-dimensional Brownian motion, and $R(\theta(t))$ is a rotation matrix:

$$R(\theta(t)) = \begin{bmatrix} \cos(\theta(t)) & -\sin(\theta(t)) & 0 \\ \sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (13)$$

For collecting training and test samples, a pair of aircraft flight scenarios is simulated for 1000 times randomly. Training samples are classified to two classes $\{-1, +1\}$ in look-ahead time $t = \{t \mid t \in (0, 20)\}$ by trajectory prediction and protected zone model.

3.2. Validity Analysis. According to the airspace situation, an intersection flying scenario of 10 aircraft has been illustrated. The airspace is established as $90\text{km} \times 90\text{km}$, and the flight altitude is set from 0 to 15000 m below. The uncertain trajectories in Free Flight are simulated in Figure 3.

From Figure 3 we can find that the airspace is crowded. The features of 10 aircraft above are extracted as test set, which is inputted to trained classifier to predict conflict during look-ahead time $t = 3\text{min}$, and probability is estimated by formula (5).

To achieve the goal mentioned above, the program LibSVM 3.22 [35] is employed to establish the SVM model.

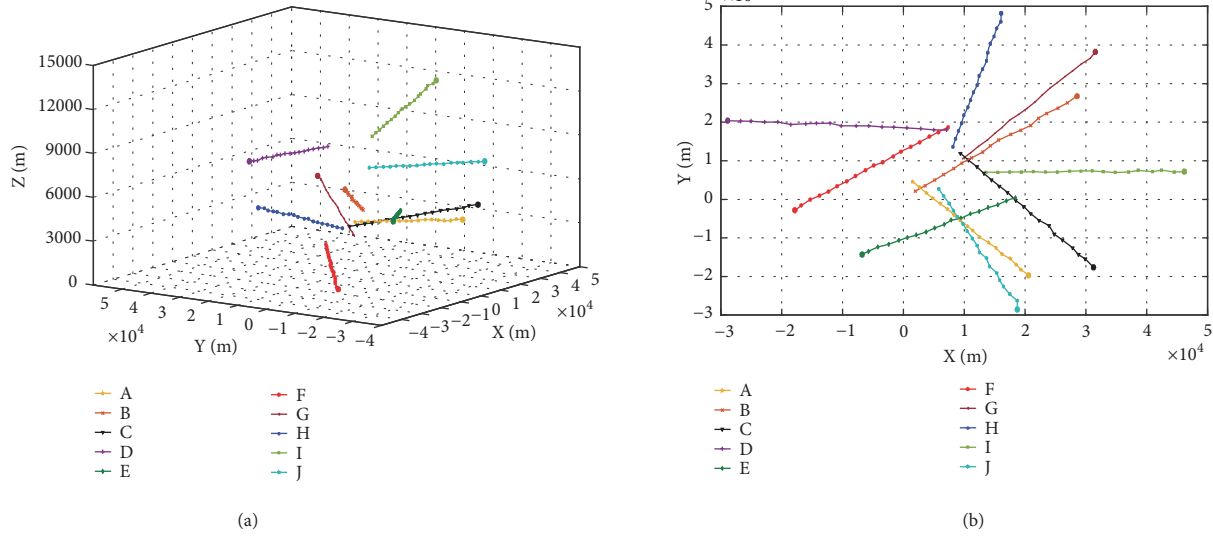


FIGURE 3: 10 aircraft's flight trajectories: (a) 3D and (b) X-Y axis.

The radial basis function (RBF) kernel was used owing to its nonlinear reflection ability. Grid searches were made on the basis of 5-fold cross-validation to choose a penalty factor C and a kernel parameter γ . After several searches, a large penalty factor $C = 21.3254$ and a kernel parameter $\gamma = 0.3515$

were found to handle the problem of overfitting. The parameters A and B are obtained from minimizing the negative log likelihood, which are -1.5881 and -0.0692 , respectively. The result is shown in following matrix $\mathbf{N}_{10 \times 10}$:

$$\begin{matrix} & A & B & C & D & E & F & G & H & I & J \\ \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \end{matrix} & \left(\begin{array}{cccccccccccc} 0 & 82.91 & 58.56 & 0.00^\# & 15.87 & 0.00^\# & 6.27 & 2.28 & 0.00^\# & 0.00^\# \\ & 0 & 10.45 & 0.00^\# & 1.32 & 0.00^\# & 15.58 & 9.41 & 0.00^\# & 0.00^\# \\ & & 0 & 0.00^\# & 28.21 & 0.07 & 84.35 & 9.41 & 0.00^\# & 0.00^\# \\ & & & 0 & 0.00^\# & 0.00^\# & 0.00^\# & 0.00^\# & 0.25 & 0.00^\# \\ & & & & 0 & 0.00^\# & 0.00^\# & 0.00^\# & 0.00^\# & 0.00^\# \\ & & & & & 0 & 0.00^\# & 0.00^\# & 0.00^\# & 0.00^\# \\ & & & & & & 0 & 65.20 & 60.62 & 0.00^\# & 0.00^\# \\ & & & & & & & 0 & 52.89 & 0.00^\# & 0.00^\# \\ & & & & & & & & 0 & 0.00^\# & 0.00^\# \\ & & & & & & & & & 0 & 0.00^\# \\ & & & & & & & & & & 0 \end{array} \right) & \end{matrix} \tag{14}$$

Annotation: "0.00[#]" means 2nd digit after the decimal point is less than 5.

In the upper triangular matrix (14), the elements on the diagonal line are zero. Other elements indicate the conflict probability value of corresponding pair of aircraft. If conflict probability is more than 50%, it is considered that a conflict exists. 10 aircraft conflict detection results in matrix (14) are as follows: there are conflicts between A and B , A and C , C and G , F and G , F and H , G and H , which are similar to actual conflict detection results in Figure 4. Most conflict probability values are 0.00[#], which evidences that aircraft conflicts are small

probability events. This case reveals the great validity of this method in multi-aircraft conflict detection.

3.3. False Alarm Rate and Missing Alarm Rate Analysis. False alarms and missing alarms are vital indicators, which are extremely concerned by air traffic controllers in conflict detection, and they are mainly influenced by the classifier performance. Particularly in this problem, imbalanced datasets have a negative impact on detection results, which have more missed alarms. Therefore, SMOTE algorithm is introduced to improve the classification performance. In this experiment,

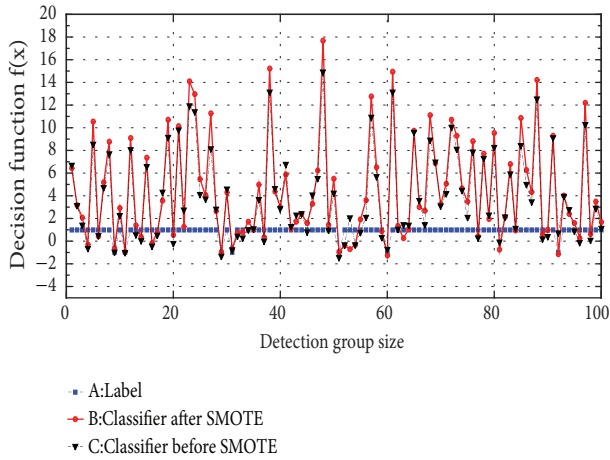


FIGURE 4: Classifier performance comparison before and after SMOTE treatment.

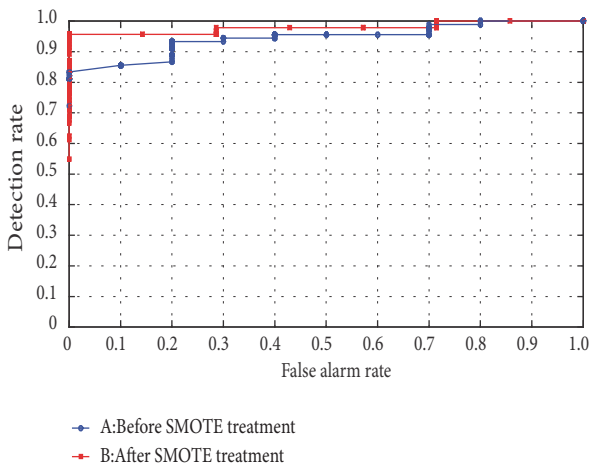


FIGURE 5: ROC curve comparison.

the classification performance whose training datasets are dealt with SMOTE algorithm is compared with that without a treatment. Different scenarios of a pair of aircraft flying are simulated for 100 times, and their samples are collected to test in differently trained classifier, and the conflicts are predicted within 6 min. The results are shown in Figure 4.

In Figure 4, Curve A is class label, -1 referring to “conflict” and +1 to “nonconflict”, respectively. Curves B and C are the decision function value of the classifiers. If $f(x) > 0$, the test set is classified as positive sample. If $f(x) < 0$, the test set is classified as negative sample. Curve C shows the classification results for samples trained by the classifier before SMOTE treatment: there are 7 conflicts in total which are detected 6 times; in addition, there are 10 time false alarms. The missing alarm rate is 14.29% and the false alarm rate is 62.50%. Curve B shows the results after SMOTE treatment: every conflict is predicted correctly and 40% false alarms are rejected effectively. From the results we can know that SMOTE algorithm improves the classification performance to a high level.

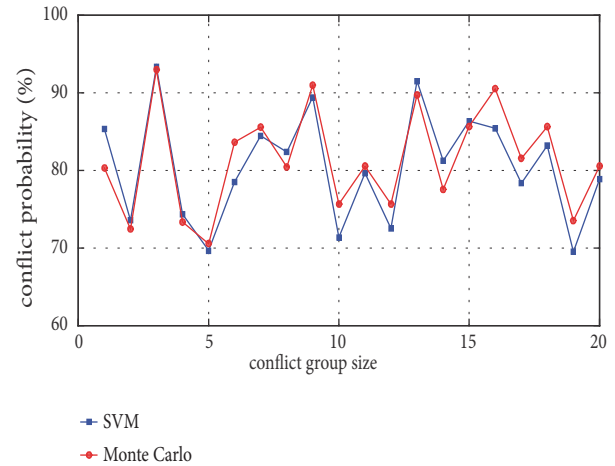


FIGURE 6: Comparison of conflict probability between SVM and Monte Carlo.

Additionally, Receiver Operating Characteristic (ROC) curve is applied to analyze the performance of SMOTE treatment. ROC [36, 37] analysis is used for graphical analysis. On the x -axis we show the false positive rate (FPR) and on the y -axis we show the true positive rate (TPR). ROC curves neatly visualise how the TPR and the FPR change for different (crisp) classifiers or evolve for the same (soft) classifier (or ranker) for a range of thresholds. The notion of threshold is the fundamental idea to adapt a soft classifier to an operating condition. ROC analysis is the tool that illustrates how classifiers and threshold choices perform. Furthermore, it reflects the relationship between the false alarm rate and detection rate. The result is shown in Figure 5.

In Figure 5, Curve A is the ROC curve before SMOTE treatment, and Curve B is plotted after SMOTE treatment. Curve B is upper than Curve A, which shows that classifier trained after SMOTE treatment has a superior performance. When detection rate values are equal, the false alarm rate value on Curve A is greater than the value on Curve B, which verifies that the SMOTE algorithm rejects some false alarms effectively.

3.4. Accuracy of Conflict Probability Analysis. The above simulations have shown the conflict detection ability of the proposed method. However, it is undefined whether the conflict probability estimation is effective. In the following research, conflict probabilities estimated by proposed method will be compared with the conflict frequencies simulated by Monte Carlo method in different conflict scenarios for 20 times, and each conflict scenario is simulated by Monte Carlo method for 10^6 times. The comparison result can be shown in Figure 6.

From Figure 6 we can find that the proposed method has a high-fitting-accuracy; conflict probability error of each group sample is less than 10%.

3.5. Stability of Conflict Probability Analysis. In this section, the scenario of a conflict between a pair of aircraft is studied, and the time-dependent change of distances between the

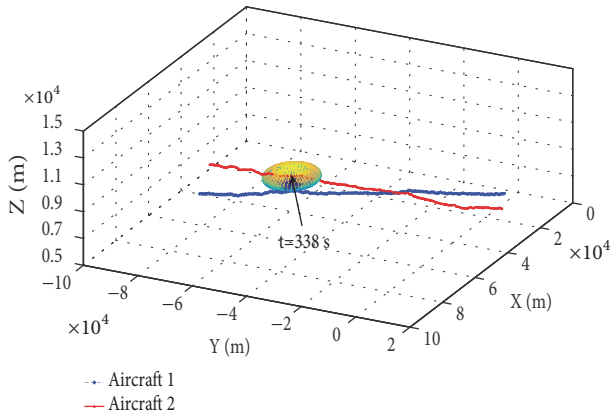


FIGURE 7: A pair of aircraft trajectories.

pair of aircraft and conflict probability within 6 min is analyzed. Unlike the former analysis, in this section a sample is classified to negative class -1 only if conflict exists at the time t . Set time step $T = 1$ s, and the conflict probability is changing with the relative distance between a pair of aircraft. The different scenarios are as follows. At first, a pair of aircraft departure scenarios are illustrated and discussed: set the heading crossing angle $|\theta| \leq 10^\circ$, time interval of departures $t = 1$ min, and the time period begins upon the last aircraft departure. The trajectories are shown in Figure 7.

The time-dependent change of distances between this pair of aircraft and conflict probability within 6 min is shown in Figure 8.

In Figure 8, the conflict probability curve exhibits the continuous and linear characteristics, which is identical to actual flight. Since the aircraft has low climb rate in departure, the conflict probabilities remain at a high level (more than 85%) in 0-210 s. With the increase of relative distances, the conflict probabilities continue to decrease. At 338 s, the distance is 11 752 m, and conflict probability $P(t=338\text{ s})=50\%$. Shortly afterwards, conflicts disappear. With Figure 7 analysis, potential conflict aircraft is at the edge of protected zone at $T=338$ s, which verifies the accuracy of the conflict probability output.

Furthermore, three scenarios are illustrated and simulated such as approach to landing phase, intersection flying, and parallel flying in adjacent flight level. In the approach to landing phase, the heading crossing angle is set as $|\theta - 90^\circ| \leq 10^\circ$. In the intersection flying phase, height difference between aircraft 1 and aircraft 2 is set as $|\Delta h| \leq 600$ m, and the heading crossing angle is set as $|\theta - 90^\circ| \leq 10^\circ$. In the parallel flying phase, set height difference as $|\Delta h| \leq 600$ m, the horizontal separation as $|\Delta s - 1000\text{ m}| \leq 4000$ m, and the heading crossing angle as $|\theta| \leq 5^\circ$. The results of time-dependent change of distances between this pair of aircraft and conflict probability within 6 min are shown in Figure 9.

In Figure 9(a), during 0-100 s, the distance between the pair of aircraft is greater than 10 km, and the conflict probability remains at a low level accordingly. After time $t=168$ s, since the relative distance is less than 7 625 m, the conflict probability is greater than 50% and a potential

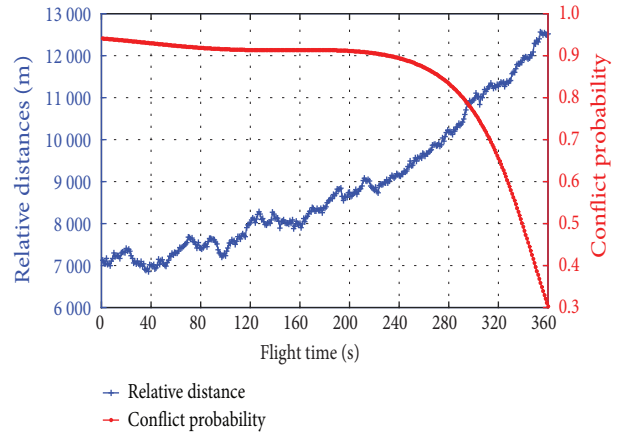


FIGURE 8: Flight time-distances/conflict probability relationship in departure phase.

conflict exists. In Figure 9(b), a conflict exists from the very beginning. Along with the decrease of relative distance, conflict probability continues to increase. With the time period from 210 s to 240 s, the pair of aircraft flies across each other, and at that time, the relative distance is minimum. The relative distance fluctuates at about 1 700 m because of external factor destabilization such as wind and navigation errors. Conflict probability is about 94%, which remains a high level. After time $t=322$ s, the conflict disappears with the increase of the distance. In Figure 9(c), conflict probability ranges from 91% to 96% during the whole flying time because of the short distances. Influenced by the wind and the navigation equipment performance, the distances of the pair of aircraft in parallel flight undulate at a narrow range, and the change of conflict probability is opposite to that of the relative distance.

4. Conclusion

In this paper, we deal with the conflict detection as a two-class classification problem. SVM is introduced to establish detection model. Compared with traditional probabilistic conflict detection algorithm, this method avoids setting alarm thresholds artificially which influences the detection accuracy. In addition, the method solves model generalization problem. As long as current positions and velocity vectors of aircraft are obtained, the conflict probabilities between aircraft can be detected quickly. These advantages make it possible that multi-aircraft conflicts are detected in real time. The simulation results have shown the high accuracy, stability, and efficiency. Moreover, the proposed algorithm provides a theoretical reference for ATM automation system.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

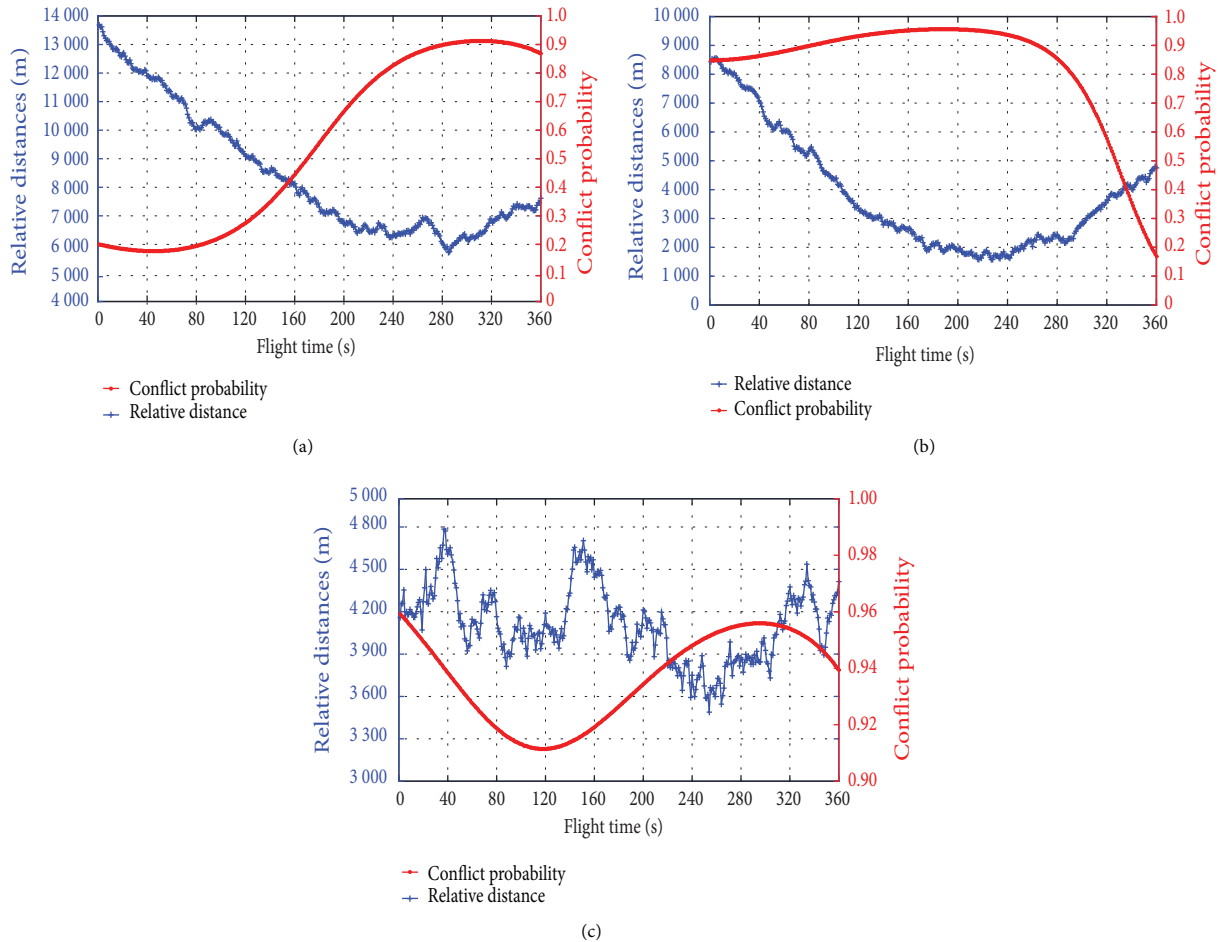


FIGURE 9: Flight time-distances/conflict probability relationship: (a) in approach to landing phase, (b) in intersection flight, and (c) in parallel flight.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant Nos. 71801221 and 61503409) and Nature Science Basic Research Plan in Shaanxi Province of China (Grant No. 2018JQ7004).

References

- [1] RTCA, "Report of the RTCA Board of Director's Select Committee on Free Flight," Tech. Rep., RTCA Inc, Washington, DC, 1995.
- [2] D. P. Thipphavong, C. A. Schultz, A. G. Lee, and S. H. Chan, "Adaptive algorithm to improve trajectory prediction accuracy of climbing aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 1, pp. 15–24, 2013.
- [3] S. Hong and K. Lee, "Trajectory prediction for vectored area navigation arrivals," *Journal of Aerospace Information Systems*, vol. 12, no. 7, pp. 490–512, 2015.
- [4] R. A. Paielli and H. Erzberger, "Conflict probability estimation for free flight," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 3, pp. 588–596, 1997.
- [5] R. A. Paielli and H. Erzberger, "Conflict probability estimation generalized to non-level flight," *Air Traffic Control Quarterly*, vol. 7, no. 3, pp. 195–222, 1999.
- [6] J. Hu, J. Lygeros, M. Prandini, and S. Sastry, "Aircraft conflict prediction and resolution using Brownian motion," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 2438–2443, Phoenix, AZ, USA, 1999.
- [7] D. Li and D. Cui, "Air traffic control conflict detection algorithm based on Brownian motion," *Journal of Tsinghua University*, vol. 48, no. 4, pp. 477–481, 2008.
- [8] V. P. Jilkov, X. R. Li, and J. H. Ledet, "Improved estimation of conflict probability for aircraft collision avoidance," in *Proceedings of the 17th International Conference on Information Fusion*, pp. 1–7, IEEE Press, Piscataway, NJ, USA, 2014.
- [9] H. Blom, G. Bakker, J. Krystul, M. Everdij, B. Klein Obbink, and M. Klompstra, "Sequential Monte Carlo simulation of collision risk in free flight air traffic," Hybridge Report 9, 2005.
- [10] D. Jacquemart and J. Morio, "Adaptive interacting particle system algorithm for aircraft conflict probability estimation," *Aerospace Science and Technology*, vol. 55, pp. 431–438, 2016.
- [11] H. A. Blom, J. Krystul, G. J. Bakker, M. B. Klompstra, and B. Klein Obbink, "Free flight collision risk estimation by sequential MC simulation," in *Stochastic Hybrid Systems*, pp. 248–282, Taylor and Francis group, CRC Press, 2007.
- [12] D. Jacquemart and J. Morio, "Conflict probability estimation between aircraft with dynamic importance splitting," *Safety Science*, vol. 51, no. 1, pp. 94–100, 2013.

- [13] S. Qiao, D. Shen, X. Wang, N. Han, and W. Zhu, "A self-adaptive parameter selection trajectory prediction approach via hidden Markov models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 284–296, 2015.
- [14] Y. Yang, J. Zhang, K.-Q. Cai, and M. Prandini, "Multi-aircraft conflict detection and resolution based on probabilistic reach sets," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 309–316, 2017.
- [15] J. M. Saavedra, "Handwritten digit recognition based on pooling SVM-classifiers using orientation and concavity based features," *Lecture Notes in Computer Science*, vol. 8827, pp. 658–665, 2014.
- [16] A. Vijayan, S. Kareem, and J. J. Kizhakkethottam, "Face Recognition Across Gender Transformation Using SVM Classifier," *Procedia Technology*, vol. 24, pp. 1366–1373, 2016.
- [17] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.
- [18] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, MIT Press, Cambridge, UK, 2000.
- [19] A. Madevska-Bogdanova, D. Nikolik, and L. Curfs, "Probabilistic SVM outputs for pattern recognition using analytical geometry," *Neurocomputing*, vol. 62, no. 1-4, pp. 293–303, 2004.
- [20] R. Seifi Majdar and H. Ghassemian, "A probabilistic SVM approach for hyperspectral image classification using spectral and texture features," *International Journal of Remote Sensing*, vol. 38, no. 15, pp. 4265–4284, 2017.
- [21] P. Beigi, R. Rohling, T. Salcudean, V. A. Lessoway, and G. C. Ng, "Detection of an invisible needle in ultrasound using a probabilistic SVM and time-domain features," *Ultrasonics*, vol. 78, pp. 18–22, 2017.
- [22] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "Training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory (COLT '92)*, pp. 144–152, July 1992.
- [23] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [24] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.
- [25] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, Cambridge, UK, 2000.
- [26] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical Review Letters*, vol. 113, no. 3, 2014.
- [27] Z.-M. Yang, H.-J. Wu, C.-N. Li, and Y.-H. Shao, "Least squares recursive projection twin support vector machine for multi-class classification," *International Journal of Machine Learning and Cybernetics*, vol. 7, no. 3, pp. 411–426, 2016.
- [28] K.-Q. Shen, C.-J. Ong, X.-P. Li, and E. P. V. Wilder-Smith, "Feature selection via sensitivity analysis of SVM probabilistic outputs," *Machine Learning*, vol. 70, no. 1, pp. 1–20, 2008.
- [29] K. Havel and J. Husarčík, "A theory of the tactical conflict prediction of a pair of aircraft," *Journal of Navigation*, vol. 42, no. 3, pp. 417–429, 1989.
- [30] R. P. Patera, "Space vehicle conflict probability for ellipsoidal conflict volumes," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 6, pp. 1818–1822, 2007.
- [31] L. Shi and R. Wu, "A probabilistic conflict detection algorithm in terminal area based on three-dimensional Brownian motion," in *Proceedings of the 2012 11th International Conference on Signal Processing, ICSP 2012*, pp. 2287–2291, China, October 2012.
- [32] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
- [33] L. Torgo, R. P. Ribeiro, B. Pfahringer, and P. Branco, "SMOTE for regression," *Lecture Notes in Computer Science*, vol. 8154, pp. 378–389, 2013.
- [34] R. Paielli and M. Field, "Empirical test of conflict probability estimation," in *Proceedings of the USA/Europe Air Traffic Management R and D Seminar*, 1998.
- [35] C. Chang and C. Lin, "LIBSVM: a Library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 75–102, 2011.
- [36] J. Hernández-Orallo, "ROC curves for regression," *Pattern Recognition*, vol. 46, no. 12, pp. 3395–3411, 2013.
- [37] J. M. Lampinen, "ROC analyses in eyewitness identification research," *Journal of Applied Research in Memory and Cognition*, vol. 5, no. 1, pp. 21–33, 2016.

Research Article

Identifying Public Transit Commuters Based on Both the Smartcard Data and Survey Data: A Case Study in Xiamen, China

Shichao Sun ¹ and Dongyuan Yang ²

¹College of Transportation Engineering, Dalian Maritime University, Dalian 116026, China

²Key Laboratory of Road and Traffic Engineering of the Ministry of Education, Tongji University, Shanghai 201804, China

Correspondence should be addressed to Shichao Sun; sunshichao1988@hotmail.com

Received 23 May 2018; Revised 3 October 2018; Accepted 17 October 2018; Published 1 November 2018

Guest Editor: Hamzeh Khazaei

Copyright © 2018 Shichao Sun and Dongyuan Yang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Understanding the travel patterns of public transit commuters was important to the efforts towards improving the service quality, promoting public transit use, and better planning the public transit system. Smartcard data, with its wide coverage and relative abundance, could provide new opportunities to study public transit riders' behaviors and travel patterns with much less cost than conventional data source. However, the major limitation of smartcard data is the absence of social attributes of the cardholders, so that it cannot clearly extract public transit commuters and explain the mechanism of their travel behaviors. This study employed a machine learning approach called Naive Bayesian Classifier (NBC) to identify public transit commuters based on both the smartcard data and survey data, demonstrated in Xiamen, China. Compared with existing methods which were plagued by the validation of the accuracy of the identification results, the adopted approach was a machine learning algorithm with functions of accuracy checking. The classifier was trained and tested by survey data obtained from 532 valid questionnaires. The accuracy rate for identification of public transit commuters was 92% in the test instances. Then, under a low calculation load, it identified the objectives in smartcard data without requiring travel regularity assumptions of public transit commuters. Nearly 290,000 cardholders were classified as public transit commuters. Statistics such as average first boarding time and travel frequency of workdays during peak hours were obtained. Finally, the smartcard data were fused with bus location data to reveal the spatial distributions of the home and work locations of these public transit commuters, which could be utilized to improve public transit planning and operations.

1. Introduction

Public transit systems have long been regarded as an effective way to mitigate the growing urban congestion, exhaust emissions, and energy consumption caused by the excessive use of private automobiles [1, 2]. The effect is very significant especially in China where the level of car ownership has developed rapidly in the past decade [3, 4]. In order to unceasingly improve the performance and promote public transit use, the authorities in metropolises of China have been working for years to obtain a better understanding of passengers' travel characteristics [5–7]. In this context, mining the travel patterns of public transit commuters has received much attention of researchers [1, 4, 8–11]. It results

from the fact that the commuters not only represent the frequent public transit users, but also represent the ones who comprise the major component of travel demand during peak hours. The accurate knowledge of their demands and travel patterns may help the agencies adjust route plans, provide appropriate policies to retain the loyal riders and enhance the attractiveness of public transit systems [10]. However, acquiring the precise travel information of public transit commuters used to be challenging. The traditional way to analyze the public transit demand largely depended on travel diary surveys, which was often plagued by its high cost, low data accuracy, low response rates, and also privacy concern. Fortunately, Automatic Fare Collection (AFC) systems have been widely implemented in China as a more efficient way of

managing fare over the manual collection method [12]. At the same time, smartcard data can provide more abundant and higher quality travel data with less cost, through which travel patterns could be analyzed based on precise observations of individuals' smartcard usage in a time period [10, 13–18].

Nevertheless, the archived smartcard data collected from AFC systems also have its limitations [19]. For instance, smartcards are usually anonymous in China. Thus, social-demographic attributes of the cardholders were absent, such as gender, age, education, and trip purpose. Therefore, identifying public transit commuters from smartcard data was not a straightforward task [5]. To this end, spatial and temporal regularities of recurring trips on the public transit networks were usually incorporated to determine the public transit commuters among the cardholders [4, 6, 9], while, even if the spatial and temporal characteristics of each cardholder could be mined from the smartcard data, how the identified spatiotemporal travel pattern tied to a real commuter was still not clear [5]. To demonstrate it, a case study in Xiamen, China was given, where the smartcard dataset harvested from 1st to 26th June, 2015 was employed. Specifically, for each cardholder, we calculated the number of workdays where he/she swiped cards during both the morning peak and the evening peak (on a maximum of 20 workdays). Then, we obtained the distribution diagram of cardholders according to the calculated number of workdays, as illustrated in Figure 1. In this context, a large number of related works usually adopted a hypothetical threshold or clustering methods to segment the commuters and noncommuters (e.g., cardholders traveling for at least a number of days are identified as transit commuters) [4, 12, 20]. However, there seemed to be no obvious fluctuation between the number of cardholders corresponding to “9 workdays” and the subsequent counterparts. Thus, it was really hard and confused to determine an exact threshold value for the identification according to the statistics in Figure 1. Moreover, even if a threshold value could be set up, it was still challenging to test the accuracy of results without any further circumstantial evidence. Therefore, in the case study of Xiamen, it was difficult to identify the commuters among cardholders by using the traditional approaches derived from the literature.

To this end, an original machine learning algorithm called Naïve Bayesian Classifier (NBC) was adopted in this paper to identify public transit commuters based on both the smartcard data and survey data. Compared with existing methods which were plagued by the validation of the accuracy of the identification results, the employed approach was a machine learning algorithm with functions of accuracy checking. The rest of the paper was organized as follows. In Section 2, a review of literature on smartcard data application was drawn, and the existing approaches of identifying commuters in smartcard data were discussed. Then, the Naïve Bayesian Classifier was introduced, and the classifier was built, trained, and tested in Section 3. Subsequently, the attribute of “commuter or not” for each cardholder was estimated based on the tested classifier, and travel characteristics and regularities of the identified commuters were analyzed in Sections 4 and 5. In the end, conclusions were drawn, and future directions were under discussion.

2. Literature Review

In recent years, the application of smartcard data mining in the planning and management of public transit systems has received much research attention. The previous works can be grouped into three categories, which can be, respectively, described as tactical-level studies, operational-level studies, and strategic-level studies [10].

In the tactical level, the trip pattern analysis and market segment were frequently addressed, contributing to the operational adjustment in public transit [14, 15]. Zhao et al. [21] developed a methodology to detect individuals' travel pattern changes by using smartcard data from London, UK, over two years. They specified one distribution for each of the three dimensions of travel behavior, and a Bayesian method was developed to identify significant change points in travel patterns. The results show that, compared to the traditional generalized likelihood ratio (GLR) approach, the Bayesian method required less predefined parameters and was more robust. Ma et al. [6] employed a Density-Based Scanning Algorithm with Noise (DBSCAN) to segment the smartcard data by analyzing different travel patterns. Then, spatial and temporal regularity of each cluster was derived through a continuous long-term observation period. The original DBSCAN method was improved by Kieu et al. [8, 9] and then applied to divide the passengers under a much lower calculation complexity. The heterogeneity of public transit riders was analyzed by Langlois et al. [22] through four-week smartcard data, and 11 travel patterns were generated. Legara and Monterola [16] developed a new classification method with promising accuracy by using the concept of eigentravel matrices which captured a commuter's characteristic travel routine, while, in operational-level studies, precise performance indicators such as schedule adherence, the number of transfers, and vehicle-kilometers on a public transit network were discussed [1, 10, 13]. Among them, the study of origin-destination (OD) and interchange inference was a hot topic and received much research attention. Trepanier et al. [23] and Ma et al. [24] combined AFC data and AVL data to infer boarding locations. For the alighting location estimation, Munizaga and Palma [7] set up a disutility function which took account of time and distance and then bus destinations were inferred by minimizing the generalized cost of the function. Sanchez-Martinez [17] extended the disutility function approach to rail networks, and a dynamic programming algorithm was designed to infer destinations. Wang et al. [25] applied the trip-chaining to infer bus passenger OD from AFC and AVL data, while Carrel et al. [26] inferred OD matrices by using smartphone and AVL data.

Regarding the applications on the strategic side, although previous researches have adopted smartcard data for the assessment and planning of public transit systems, however, the data source was somehow fragmentary for the behavior analysis [19]. For example, most of the previous literature on the analysis of cardholders' commuting characteristics oversimplified the definition of a public transit commuter [5]. In this context, Kusakabe and Asakura [19] developed a data fusion method based on the Naïve Bayesian Classifier (NBC) by integrating the smartcard data with the personal travel

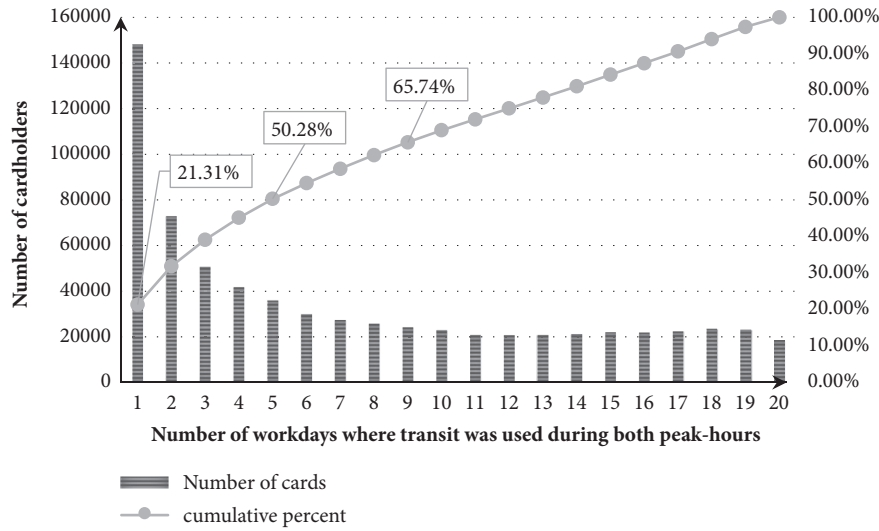


FIGURE 1: Basic statistics of smartcard data in Xiamen, China.

TABLE 1: Examples of smartcard transaction records.

Card id	Date	Transaction Time	Bus Route	Bus Plate Number
326352926	2015/6/22	09:43:47	Bus Route 815	D3789
154380354	2015/6/23	06:40:50	Bus Route 301	D3775
323446469	2015/6/23	20:40:13	Bus Route 713	D9342
051397471	2015/6/24	11:40:23	Bus Route 301	D3764
360122282	2015/6/24	08:42:30	Bus Route 512	D6453

survey data. The approach was intended to estimate absent attributes of trip purpose in smartcard data and enhance the understanding of travelers' behavior. Nevertheless, their study was implemented at only one railway station run by a private company, and the proportion of cardholders only accounted for about 10% of the total ridership. Thus, the applicability, portability, and representativeness of the proposed approach may be affected by the sample size. Moreover, other than estimating the travel purpose for each trip, it was also not clear whether the data fusion method could be applied to infer the attribute of cardholders. Since achieving the identification of public transit commuters in smartcard data was not an easy task in most metropolis of China, this paper extended the application of NBC approach and employed the method to estimate the attribute of cardholders instead of trip purpose. Besides, as an extension to the work of Kusakabe and Asakura [19], the approach was adopted and applied to the whole public transit network in Xiamen, China, with 85% of the transactions completed through smartcards. The spatial distribution of commuters' home and work locations in the case study were also revealed by fusing together smartcard and Automatic Vehicle Location (AVL) data.

3. The Smartcard Data

3.1. A Brief Introduction of Xiamen, China. Xiamen is administered as a subprovincial city of Fujian province with an area

of 1,699.39 square kilometers. The population in Xiamen was nearly 3.5 million according to the Census in 2014. Regular bus systems and the Bus Rapid Transit (BRT) are the principal public transportation modes for local residents. There are nearly 320 regular bus routes and 5 BRT lines currently in service in Xiamen.

3.2. Data Preparation. "E-Tong Card" is the only contactless smartcard used for electronic payments in the public transportation system of Xiamen, China, and it started to come into use in 2006. Through the smartcards, passengers can access all the bus routes and BRT lines. Until 2015, the circulation of E-Tong cards has exceeded 6 million, and over 85% of the transactions in the public transit network were completed through the automated fare system. Thus, it provides a good opportunity for researchers to obtain the users' travel activities through smartcard data mining.

The smartcard data analyzed in this paper were harvested from the archived data in the AFC system of "E-Tong Card". The original dataset contained whole transaction records on the public transit network, collected from 22nd to 26th June, 2015 (except elder cards and student cards; 5 consecutive workdays were covered). Since the pricing of the bus system was not distance-based in Xiamen, the passenger only swiped the card once when boarding. Each record included the information of card id, date, transaction time, and bus route as well as plate number, as shown in Table 1.

4. Methods

4.1. The Naïve Bayesian Classifier. The Naïve Bayes classifier was a machine learning technique efficiently utilized for classification/identification applications in data mining environment. The reliability in predicting and decision-making constructed the statistical nature of the NBC. In practice, NBC could estimate the absent attributes of the data by predicting probabilities of class membership. To classify a new instance, the algorithm used the Bayesian rule to calculate the conditional probability of each class value and took the class with the maximum probability as the identified class. The collected survey data were usually split into two parts: training part and test part. The algorithm used the training data to build the classifier and then estimated the required probability values in the test data to test the accuracy of the built classifier.

Let vector $\mathbf{F} = \{f_1, f_2, \dots, f_N\}$ be a set of behavioral attributes/features, as shown in Figure 2. Each element of vector \mathbf{F} represented a common attribute or travel characteristic which existed in both survey and smartcard data, such as travel frequency or boarding time. Let C be a variable of the absent classification item in the smartcard data but could be measured by survey data [19]. It was assumed that elements of vector \mathbf{F} were conditionally independent when C was given. In this study, classification item C and elements of vector \mathbf{F} were treated as discrete variables. Then, $p(C | \mathbf{F})$ could be trained through Bayes' theorem by using the survey data.

$$\begin{aligned} p(C | \mathbf{F}) &= \frac{1}{p(\mathbf{F})} p(C) p(\mathbf{F} | C) \\ &= \frac{1}{p(\mathbf{F})} p(C) \prod_{N=1}^{N'} p(f_N | C) \end{aligned} \quad (1)$$

In (1), $p(C)$, $p(\mathbf{F})$, and $p(f_N | C)$ could be estimated by survey data. Specifically, $p(C)$ and $p(\mathbf{F})$ were the probabilities derived from the proportion of interviewees having classification item C and vector \mathbf{F} , respectively. The conditional probability distribution $p(f_N | C)$ represented the composition rate of interviewees having attribute f_N corresponding to classification item C .

The attribute vector \mathbf{F} could also be measured for each cardholder in the smartcard dataset, since it was shared in both of the two data sources. Then, the trained classifier could be adopted to estimate the absent attribute C of cardholders. It could be expressed as the following equation:

$$\hat{c}(\mathbf{F}) = \arg \max_{c \in C} p(c | \mathbf{F}), \quad C = \{c_1, c_2, \dots, c_M\} \quad (2)$$

where c was one item of the classification attribute C .

4.2. Survey Design. The flow chart of the employed NBC method was illustrated in Figure 3. The core of machine learning in this study was to estimate absent elements of smartcard data by using the survey data to train and test the classifier. The survey was designed as shown in Figure 3. The classification item C could only be observed in the survey data, and it could be measured through the

interviewees' response to "whether the bus service is your first choice when commute". The attribute vector \mathbf{F} represented behavioral attributes that could be observed by both of the two data sources. In this study, the vector \mathbf{F} contained three independent attributes, and it was relatively easy to measure them by using any one of the two datasets: (f_1) "average first boarding time on the workdays in last week"; (f_2) "average last boarding time on the workdays in last week"; and (f_3) "the number of workdays where commuting by bus in the last week", which represented the number of workdays where the cardholders swiped cards during both of the two peak-hours in the last week.

5. Experimental Analysis

5.1. The Trained and Tested NBC. The NBC model was built and tested by the use of survey data harvested from Xiamen, China. The research area in this paper was not broad with about 150 square kilometers; it can be assumed that the characteristics of public transit users were homogenous all over the city. The survey was implemented in the major industrial parks, software parks, and CBDs of Xiamen on 29th, June, 2015. Since these areas nearly provided more than 70% of job positions in Xiamen [27], the samples could be used to serve as the representative of typical commuters in Xiamen. The interviewees were selected at their workplaces with random sampling by meeting the conditions: (1) permanent staff with smartcards and (2) usually commute between home and their workplaces. Thus, the definition of public transit commuters as well as its response indicator "whether the bus service is your first choice when commute" in the survey could be better understood by the selected interviewees in a standard way. The survey was designed as above mentioned, through which the three independent attributes and the classification item could be measured. A total of 900 questionnaires were issued. Eventually, 532 valid samples were collected with 62% (330) of which stated themselves as public transit commuters. The reliability of the survey data was acceptable, since the value of Cronbach's Alpha was greater than 0.8.

Regarding the descriptive statistics of the valid samples (Table 2), 54.1% of the interviewees were female corresponding with 45.9% of which were male. It was in line with the census data of Xiamen in 2014. The age of the valid interviewees mostly fell in the range between 18 and 60. Almost half of the passengers were between 21 and 30 years old (49.6%). Thus, it was also consistent with the characteristics of commuters and meet the feature of smartcard dataset excluding student cards and elder cards. Statistics related to the occupation and income of the interviewees were also calculated. It indicated that company employees (46.7%) accounted for almost half of the samples. Correspondingly, more than half of the interviewees earned less than 4500 yuan (RMB) per month, whereas there was still 38.6% of the ones had a monthly income of less than 3,000 yuan (RMB). The average monthly salary of Xiamen in 2015 was 3,508 yuan (RMB), according to the salary report of employees in 2016. Thus, it implied that there was no potential bias from the selected interviewees regarding the income.

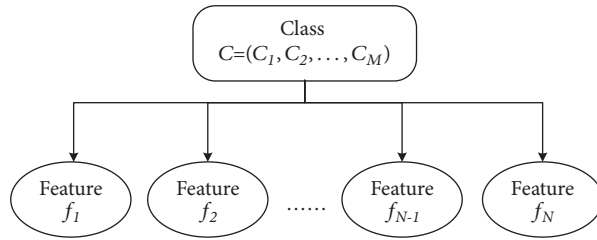


FIGURE 2: Basic structure of the NBC.

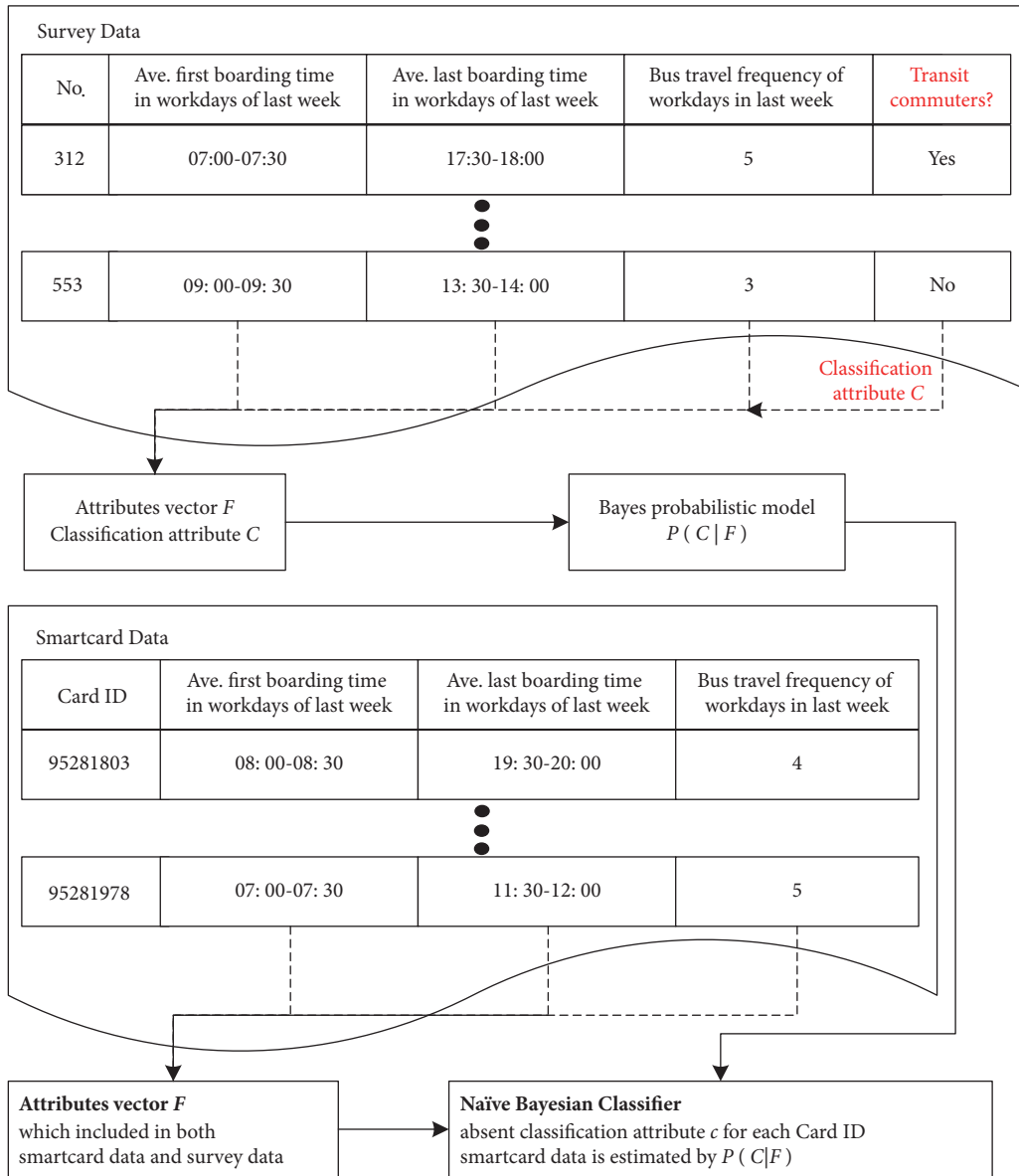


FIGURE 3: Flow chart of NBC for identifying commuters in smartcard data.

Then, the survey data were divided into two datasets: a training dataset and a testing dataset. Each dataset contained 266 valid samples, respectively, 165 of which were stated public transit commuters. By using the training dataset, the conditional probability distribution $p(f_N | C)$ could be

derived from the proportion of interviewees having attribute f_N ($N = 1, 2, 3$) corresponding to each classification C (public transit commuters or not), shown in Figures 4(a), 4(b), and 4(c). Since $P(F)$ did not depend on C , it could be regarded as a constant value when F was given. The distribution $P(C)$

TABLE 2: Descriptive statistics of valid samples.

Attributes	Statistics
Gender	Male (45.9%), female (54.1%)
Age (years)	<=20 (1.2%), 21-30 (49.7%), 31-40 (24.0%), 41-50 (11.7%), 51-60 (11.9%), ≥60 (1.5%)
Occupation	Company employees (46.7%), civil servant (6.8%), educator and researcher (8.8%), service worker (20.7%), self-employed worker (14.6%), other (2.4%)
Monthly income (Unit: RMB)	<=3000 (38.6%), 3001-4500 (20.1%), 4501-6000 (25.6%), >6000 (15.7%)

TABLE 3: Test Results of the trained model.

	Transit Commuters (165 samples)	Non-Transit Commuters (101 samples)
Correctly estimated	151	87
Failure number	14	14
Accuracy Rate	92%	86%

could also be easily derived from the training data, which was calculated here as follows: $P(C_{commuter}) = 62\%$.

Based on (1) and (2), when a new attribute vector \mathbf{F} was given, the conditional probability of each classification item could be calculated. The item with the maximum probability could be considered as the identified class. For example, the probability of an interviewee being a public transit commuter could be calculated as $p(C_{commuter} | \mathbf{F}) = 97.4\%$, if his/her attribute vector could be denoted as

$$\mathbf{F} = \{f_1 = 07 : 00 : 00 - 07 : 30 : 00, f_2 = 16 : 30 : 00 - 17 : 00 : 00, f_3 = 5\} \quad (3)$$

Then, the trained NBC model was tested by using the testing dataset. Firstly, only the attribute vector \mathbf{F} of each interviewee was utilized in the NBC model to predict their class values. Then the accuracy would be checked by comparing the estimated class membership with the actual classification of the interviewee [19]. Test results were reported in Table 3.

For a total of 266 samples in the testing dataset, the average accuracy rate could be calculated as $[1 - (14+14)/266] * 100\% = 90\%$, including 14 failures in each of classification items. Specifically, 92.0% of public transit commuters (151 samples) and 86.0% of nonpublic transit commuters (87 samples) were correctly identified. Since the Root Mean Square Error for the identification of public transit commuters was less than 0.3 (0.291), the accuracy rate of the trained NBC could be considered acceptable. Regarding a smartcard dataset which contained M cardholders, when P of them were identified as commuters by using the NBC ($P < M$), then the actual number of public transit commuters among cardholders could be calculated according to (4), taking the accuracy rate into account.

$$\text{Actual Number} = [P * 0.92 + (M - P) * 0.14] \quad (4)$$

5.2. Results and Discussions. The NBC was employed to identify the commuters in the smartcard dataset, which was trained and tested by using survey data. The identification

results and some other statistics were listed in Table 4. Eventually, nearly 290,000 cardholders (41.94% of the total) were identified as public transit commuters. According to the error analysis based on test instances in the last section, the accuracy rate for this result of commuter-identification should be around 92%, corresponding with the Root Mean Square Error of 0.291. Regarding the identified commuters, the statistics reflected that they averagely began their trips at 7:30-8:00 and swiped cards for their last-boarding at 17:30-18:00, 87% of which swiped cards twice in a typical workday. This was likely to represent a typical commuting trip chain, where public transit riders took a bus from home to their workplaces in the morning and then returned home in the evening [6].

Then, in order to better improve and test the accuracy of results, we cross-checked the bus route information of each identified commuter who swiped cards twice per workday. Due to the bus routes overlap, the demand of cardholders' first travel may be fulfilled by different bus routes in a period time. Nevertheless, during the 5-workday period, the statistical results indicated that the bus routes corresponding to commuters' first boarding were stable, more than 93% of which depended on the same bus route. In addition, we also examined whether the bus route corresponding to each commuter's last trip per workday was contained in his/her bus route dataset of first boarding. Since a typical commuting chain reflected a home-work-home journey, the bus route/metro line used in commuter's last trip on the workday should have a strong correlation with the ones of his/her first boarding. The results indicated that 96% of the assumed home-work-home journeys (2 trips per workday) satisfied the above conditions. Thus, the primary OD demand of public transit commuters can be obtained by connecting the first-boarding locations with the last-boarding locations of the identified commuters who swiped cards twice per workday. Since 87% of the identified commuters swiped cards twice per workday, it implied that the planning and the network structure of bus lines in Xiamen were really well to fulfill the demand of commuters. However, the average bus line overlap factor in Xiamen was 5.15 [27]. Thus, in

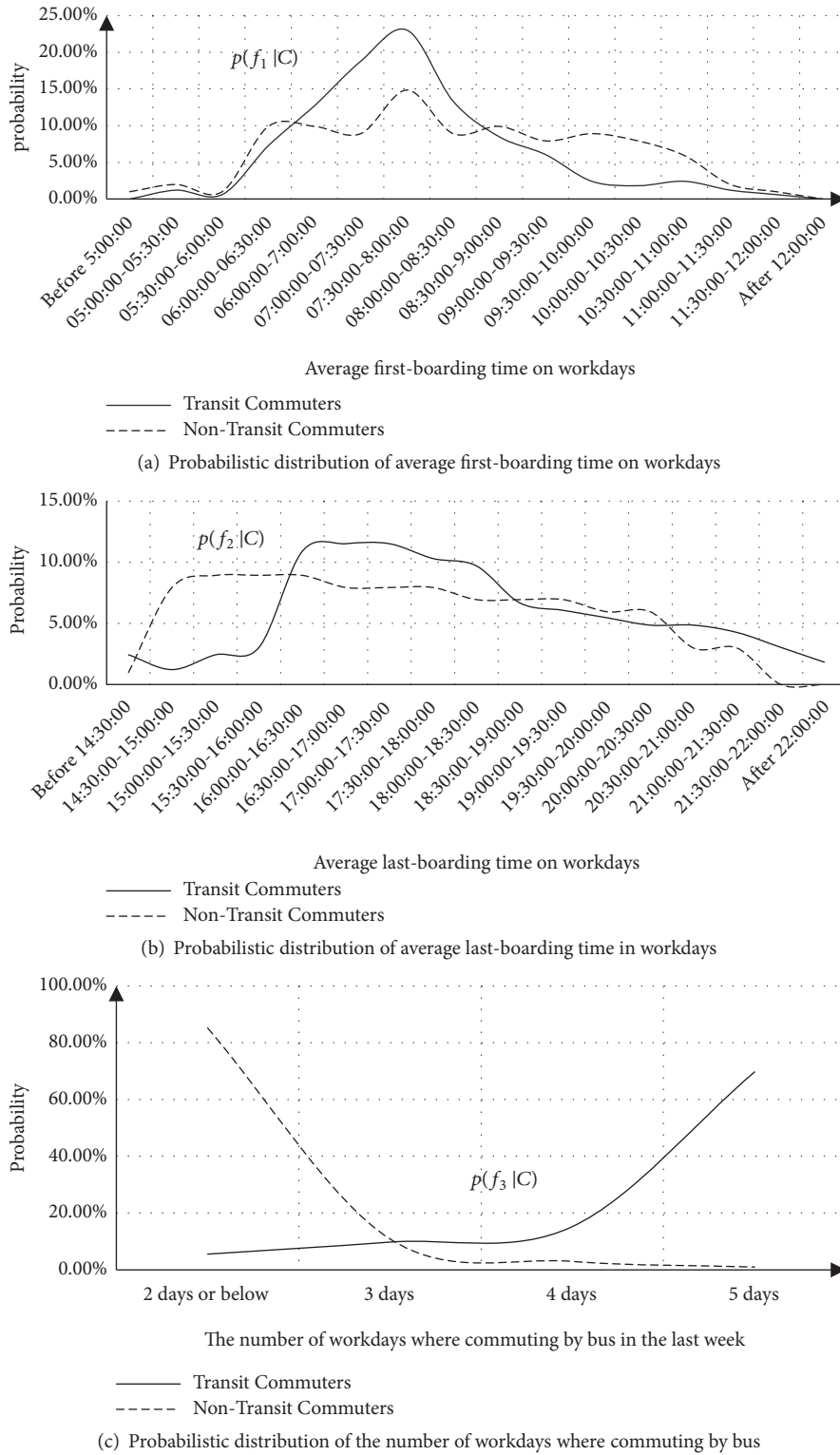
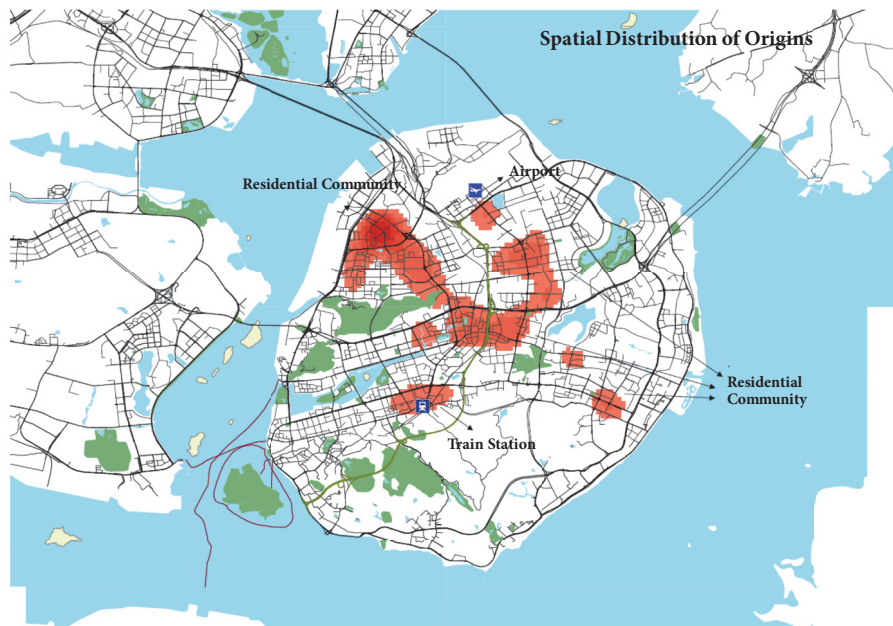


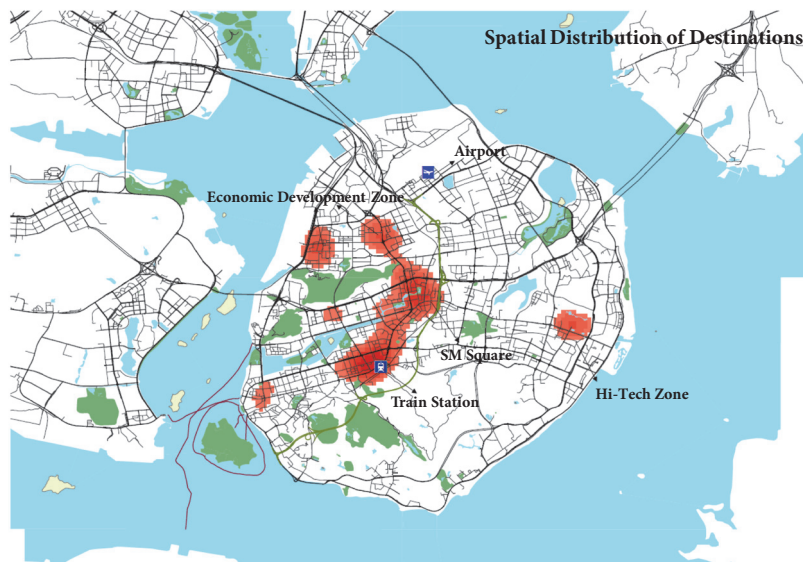
FIGURE 4

TABLE 4: Identification results of the NBC model.

Transit Commuters	Number of cards	Ave. first-boarding time	Ave. last-boarding time	Travel frequency of workdays in a week
	288153	7:30-8:00	17:30-18:00	4.2 days



(a) Spatial distribution of origins of commute activities



(b) Spatial distribution of destinations of commute activities

FIGURE 5

contrast, the above results also reflected that the existing bus overlap lines have not played an effective role in supporting the majority of commuting activities, which may lead to an extreme crowding of bus routes on main public transit corridors.

As for the analysis of primary OD demand of public transit commuters, however, the AFC system was not initially integrated with AVL system in Xiamen, China. Thus, the boarding-location should be estimated by integrating the two separate databases. In this paper, we employed the estimation method of boarding location proposed in the literature of Ma et al. [6]. It matched the transaction time of smartcard data with the time of GPS data generated from the bus with the same plate number and then inferred the location where he/she got on board. Finally, it obtained the station information by minimizing the distance between the boarding-location and bus stations on the same direction of the route. About 94% of the transaction records were finally matched with the GPS data by minimizing the time difference and then obtained the boarding-location information. Then, based on the boarding-location information, the distribution of commuters' origins and destinations of commuting trips could be obtained, shown in Figure 5. It was likely to depict a whole picture of the distributions of their residence and workplaces. Specifically, the results indicated that the spatial distribution of public transit commuters' origins was disperse when compared with the distribution of their destinations. Most of the public transit commuters lived in several core residential communities. However, workplaces of public transit commuters were more concentrated in downtown and several development zones. For instance, a large proportion of public transit commuters worked at the downtown area near the train station as well as the SM Square which was the CBD of Xiamen. Another two concentration areas of employees were located in the economic development zone in the northwest and the Hi-Tech parks in the east of Xiamen.

6. Conclusion and Future Researches

This paper employed the Naïve Bayesian Classifier to identify public transit commuters based on both the smartcard data and the survey data. A case study was given in Xiamen, China, and the classifier was trained and tested by related survey data collected from 532 valid samples. Then, it was applied to the smartcard cardholders for the identification of public transit commuters. The results indicated the following:

- (i) The success rate of the identification of public transit commuters was 92% in the test case. Nearly 290,000 cardholders were classified as public transit commuters. However, considering the error of the NBC for identification of commuters, the actual number of public transit commuters with smartcards in Xiamen should be around 323,000.
- (ii) Regarding the identified commuters, the statistics reflected that they averagely began their trips at 7:30-8:00 and swiped cards for their last-boarding at 17:30-18:00, 87% of which swiped cards twice in a typical workday.
- (iii) The travel pattern reflected by the commuters who swiped cards twice in a typical workday was likely to represent a typical commuting trip chain, where public transit riders took a bus from home to their workplaces in the morning and then returned home in the evening.
- (iv) Through few transfers, bus lines in Xiamen were really well to fulfill the demand of commuters. Conversely, it also reflected that the existing bus overlap lines have not played an effective role in supporting the majority of commuting activities, which may lead to an extreme crowding of bus routes on main public transit corridors.
- (v) The primary OD demand of public transit commuters was obtained. It was found that home locations of public transit commuters were more disperse than their work locations in Xiamen. Most of the public transit commuters lived in several core residential communities. However, workplaces of public transit commuters were more concentrated in downtown and several development zones.

Nevertheless, based on the identified public transit commuters, travel pattern analysis of commuting activities could be conducted in further studies. More and more smartcard data related challenges may need to be addressed when researching on deep analysis of public transit rider's travel behavior or fusion method with other data sources.

Data Availability

The data supporting the results in this paper was authorized by "Transport Commission of Xiamen Municipality". Thus, the database of residents' travel survey could only be accessible with the authorization and permission of the organization.

Conflicts of Interest

The authors declare that there are no conflicts of interest in this paper.

Acknowledgments

This research is supported by National Natural Science Foundation (Project: 51478350) and the Fundamental Research Funds for the Central Universities (3132018305; 3132016301).

References

- [1] J. Y. Park, D.-J. Kim, and Y. Lim, "Use of smart card data to define public transit use in Seoul, South Korea," *Transportation Research Record*, no. 2063, pp. 3-9, 2008.
- [2] Y. Shiftan, Y. Barlach, and D. Shefer, "Measuring passenger loyalty to public transport modes," *Journal of Public Transportation*, vol. 18, no. 1, pp. 1-16, 2015.
- [3] S. Sun, "Public Transit Loyalty Modeling Considering the Effect of Passengers' Emotional Value: A Case Study in Xiamen,"

- China,” *Journal of Advanced Transportation*, vol. 2018, pp. 1–12, 2018.
- [4] J. Zhou, E. Murphy, and Y. Long, “Commuting efficiency in the Beijing metropolitan area: An exploration combining smart-card and travel survey data,” *Journal of Transport Geography*, vol. 41, pp. 175–183, 2014.
- [5] X. Ma, C. Liu, H. Wen, Y. Wang, and Y. Wu, “Understanding commuting patterns using transit smart card data,” *Journal of Transport Geography*, vol. 58, pp. 135–145, 2017.
- [6] X. Ma, Y. J. Wu, Y. Wang, F. Chen, and J. Liu, “Mining smart card data for transit riders’ travel patterns,” *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 1–12, 2013.
- [7] M. A. Munizaga and C. Palma, “Estimation of a disaggregate multimodal public transport Origin-Destination matrix from passive smartcard data from Santiago, Chile,” *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 9–18, 2012.
- [8] L.-M. Kieu, A. Bhaskar, and E. Chung, “A modified density-based scanning algorithm with noise for spatial travel pattern analysis from smart card AFC data,” *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 193–207, 2015.
- [9] L. M. Kieu, A. Bhaskar, and E. Chung, “Passenger segmentation using smart card data,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1537–1548, 2015.
- [10] M.-P. Pelletier, M. Trépanier, and C. Morency, “Smart card data use in public transit: a literature review,” *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 4, pp. 557–568, 2011.
- [11] S. Sun, Z. Duan, Q. Xu, and P. Chen, “School bus routing problem in the stochastic and time-dependent transportation network,” *PLoS ONE*, vol. 13, no. 8, p. e0202618, 2018.
- [12] J. Chen, Y. K. Lv, and M. L. Cui, “Estimating alighting stops of smart card public transportation passengers based on travel patterns,” *Xi’an Univ. of Arch. & Tech I*, vol. 1, pp. 23–29, 2018.
- [13] M. Bagchi and P. R. White, “The potential of public transport smart card data,” *Transport Policy*, vol. 12, no. 5, pp. 464–474, 2005.
- [14] A.-S. Briand, E. Côme, M. Trépanier, and L. Oukhellou, “Analyzing year-to-year changes in public transport passenger behaviour using smart card data,” *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 274–289, 2017.
- [15] M. K. El Mahrsi, E. Côme, L. Oukhellou, and M. Verleysen, “Clustering Smart Card Data for Urban Mobility Analysis,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, 2017.
- [16] E. F. T. Legara and C. P. Monterola, “Inferring passenger types from commuter eigentravel matrices,” *Transportmetrica B*, vol. 6, no. 3, pp. 230–250, 2018.
- [17] G. E. Sánchez-Martínez, “Inference of public transportation trip destinations by using fare transaction and vehicle location data: Dynamic programming approach,” *Transportation Research Record*, vol. 2652, pp. 1–7, 2017.
- [18] D. Tribone, D. Block-Schachter, F. Salvucci, J. Attanucci, and N. Wilson, “Automated, data-driven performance regime for operations management, planning, and control,” *Transportation Research Record*, vol. 2415, pp. 72–79, 2014.
- [19] T. Kusakabe and Y. Asakura, “Behavioural data mining of transit smart card data: a data fusion approach,” *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 179–191, 2014.
- [20] J. Chen and D.-Y. Yang, “Identifying boarding stops of bus passengers with smart cards based on intelligent dispatching data,” *Jiaotong Yunshu Xitong Gongcheng Yu Xinxi/Journal of Transportation Systems Engineering and Information Technology*, vol. 13, no. 1, pp. 76–80, 2013.
- [21] Z. Zhao, H. N. Koutsopoulos, and J. Zhao, “Detecting pattern changes in individual travel behavior: A Bayesian approach,” *Transportation Research Part B: Methodological*, vol. 112, pp. 73–88, 2018.
- [22] G. Goulet Langlois, H. N. Koutsopoulos, and J. Zhao, “Inferring patterns in the multi-week activity sequences of public transport users,” *Transportation Research Part C: Emerging Technologies*, vol. 64, pp. 1–16, 2016.
- [23] M. Trépanier, N. Tranchant, and R. Chapleau, “Individual trip destination estimation in a transit smart card automated fare collection system,” *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 11, no. 1, pp. 1–14, 2007.
- [24] X.-L. Ma, Y.-H. Wang, F. Chen, and J.-F. Liu, “Transit smart card data mining for passenger origin information extraction,” *Journal of Zhejiang University SCIENCE C*, vol. 13, no. 10, pp. 750–760, 2012.
- [25] W. Wang, J. P. Attanucci, and N. H. M. Wilson, “Bus passenger origin-destination estimation and related analyses using automated data collection systems,” *Journal of Public Transportation*, vol. 14, no. 4, pp. 131–150, 2011.
- [26] A. Carrel, P. S. C. Lau, R. G. Mishalani, R. Sengupta, and J. L. Walker, “Quantifying transit travel experiences from the users’ perspective with high-resolution smartphone and vehicle location data: Methodologies, validation, and example analyses,” *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 224–239, 2015.
- [27] S. Zhang, M. Ding, and Z. F. Shi, “Transport Commission of Xiamen Municipality,” Xiamen Transport Annual Report in 2014, Xiamen, China, 2015.