TRAFFIC STATE PREDICTION IN A CONNECTED AUTOMATED DRIVING ENVIRONMENT

BY

MOHAMMADREZA KHAJEH HOSSEINI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Civil Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2022

Urbana, Illinois

ABSTRACT

Accurate traffic state prediction is critical to implementing an effective traffic management strategy. Unfortunately, the dynamic nature of traffic flow and the limitations of conventional data sources (e.g., loop detectors and stationary sensors) have historically made it difficult to predict traffic state accurately. Furthermore, the driving environment evolves due to interactions among individual vehicles and their surrounding environment. Accordingly, capturing the interaction among the vehicles can potentially lead to more accurate traffic state prediction. Such interactions, however, cannot be captured via conventional sensors.

The introduction of connected and automated vehicles can potentially address the limitations of conventional sensors by providing accurate vehicle trajectory data. Such data can be translated into the time-space diagram, which can capture the interactions among vehicles. Utilizing such data, this study proposes: (1) traffic state prediction methodologies based on convolutional neural networks (CNN) that can directly utilize the time-space diagram in the prediction process. Convolutional layers allow capturing the features embedded in the time-space diagram and accounting for the disturbances such as shockwaves in the traffic stream; and (2) a combined microscopic and macroscopic traffic state prediction methodology to directly utilize the interactions among vehicles in the traffic state prediction process. These interactions can be defined as a combination of lateral and longitudinal maneuvers of vehicles in response to their driving environment. This approach predicts the future trajectory of individual vehicles in the traffic stream and converts that microscopic-level prediction to macroscopic-level prediction. One of the challenges with predicting the trajectory of vehicles is that more than one maneuver is feasible for every vehicle in many driving scenarios. Consequently, the accuracy of prediction at the individual level decays with the increase in the prediction horizon due to the uncertainty in the drivers' choice of maneuvers and the possibility of various configurations and outcomes. Therefore, this dissertation adopts a probabilistic approach to predict the location of individual vehicles based on different maneuvers. The key step in this combined microscopic and macroscopic traffic state prediction approach is to convert such probabilistic trajectory predictions to aggregated traffic state predictions (i.e., flow, space-mean speed, and density). Note that the traffic state prediction methodologies proposed in this dissertation are data-driven approaches, which require accurate and comprehensive training datasets. Accordingly, this dissertation utilizes both simulation-based and real-world vehicle trajectory datasets to train and evaluate the proposed models.

To my wife, for her love and support.

ACKNOWLEDGMENTS

I would like to give special thanks to my advisor, Dr. Alireza Talebpour. I am grateful for his time, ideas, and support to make this experience a success. His enthusiasm and motivation inspired me throughout my Ph.D. studies. He provided me with the guidance and determination needed to explore the depths of my research. I am also very grateful to my thesis committee members: Dr. Yanfeng Ouyang, Dr. Mani Golparvar Fard, and Dr. Saurabh Gupta, for their guidance and valuable input. I acknowledge former and present members of the Smart City Lab. Working with them has been a pleasure, and I appreciate their technical support, generous help, and constructive discussions.

Also, I would like to thank my parents, Simin and Hossein. Their many admirable qualities have inspired me and provided me with a solid foundation to meet life. Moreover, they taught me to be hardworking and persistent.

Finally, great appreciation goes to my wife, Aida. She has been an excellent friend and partner throughout my graduate studies, and it would have been impossible to complete my research without her love and support.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Delaying the onset of traffic breakdown and congestion formation are the main objectives of any congestion prevention and mitigation strategy. While a number of reactive strategies have been proposed over time (e.g., see [1] and [2]), successful congestion management strategies rely on accurate prediction of traffic flow dynamics [3]. These approaches have faced two key challenges:

1. Conventionally, stationary vehicle detectors (e.g., inductive loops, piezoelectric sensors installed in the pavements, and cameras and radars installed on supporting structures) were utilized to monitor the traffic condition and provide the necessary data for traffic prediction. Accordingly, the available data only contained measurements along a very short segment of the road and have been mostly in the form of aggregated measures (e.g., flow, occupancy/density, and average speed). Unfortunately, such data cannot accurately capture the evolution of traffic flow throughout the roadway (e.g., cannot fully capture the shockwave formation and propagation). While introducing multiple sensors along the roadway can slightly mitigate this challenge, a lot of key traffic flow features still cannot be captured due to the location-specific measurements by these sensors. For instance, Talebpour et al. [2] showed that speed harmonization can prevent flow breakdown if the system can detect shockwave formation at its onset, which cannot be achieved with point measurements from conventional sensors. In other words, by the time the shockwave reaches the sensor location, it might be impossible to prevent its propagation using common congestion management strategies.

2. Most of the existing data-driven traffic prediction models at the aggregated level are developed to work with the aggregated data provided by conventional traffic monitoring devices. The aggregated traffic data such as flow, density, and average speed describe the average behavior of the traffic stream and not the individual vehicles directly. The traffic state evolves as a result of movement and interaction among the individual traffic agents in the traffic stream and due to uncertainties such as incidents. The impact of individual vehicle behaviors and the interaction among the vehicles on the traffic

state increases with the increase in the traffic flow and density. As a result, capturing the individual behaviors and interactions among the vehicles is essential for accurate traffic state prediction. The driving environment prediction at the individual level can be performed in the form of vehicle trajectories. Most of the existing trajectory prediction models rely on physics-based motion models that are limited to a short-term prediction period (up to one second). Predicting the trajectory of the traffic agents for a more extended period is challenging due to the uncertainties in their behavior and the possibility of various configurations and outcomes. In addition, the physics-based models do not consider the change in the motion of vehicles when interacting with the traffic environment in the form of different maneuvers [4]. The vehicle's movement can be structured into different maneuvers, such as lane changing, taking over, or slowing down. Adding that structure to the vehicle's motion can help confine the trajectory prediction problem and increase the prediction accuracy [5].

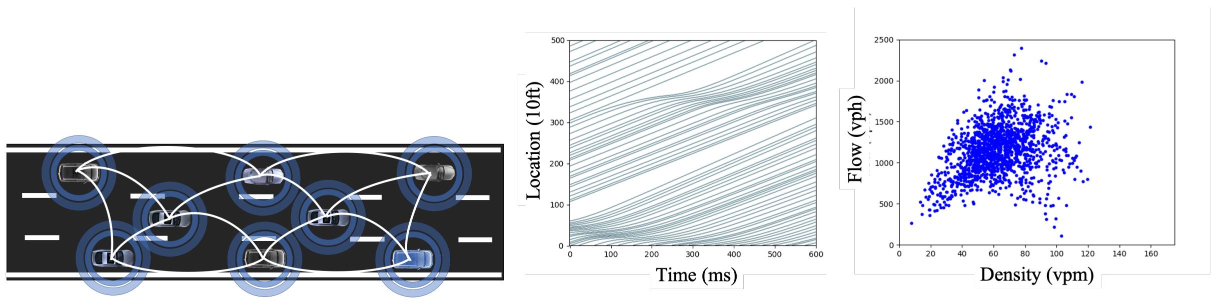
Advancement in wireless connectivity has provided exceptional communication, and data exchange opportunities between vehicles and their surroundings [6]. Connected vehicle technologies enable vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication (Figure 1.1a) that provides the opportunity to monitor the driving environment continuously over time and space. The connected vehicle data facilitates monitoring the driving environment at the individual level, such as in the form of the time-space diagram, which was not available through the conventional traffic monitoring devices. The time-space diagram (Figure 1.1b) is the plot of vehicles' position over time and space representing the traffic flow dynamics (e.g., interaction among vehicles and the formation of traffic shockwaves). The data from connected vehicles can also be aggregated into traffic flow and density to monitor the average state of the traffic stream (Figure 1.1c). Accordingly, data from connected vehicles can complement the conventional traffic detectors and provide a better picture of the traffic state and driving environment. Accordingly, the main focus of this dissertation is on developing robust traffic state prediction algorithms and approaches utilizing the newly available data from vehicular communications.

1.1 Problem Statement and Research Objectives

The driving environment is constantly evolving, and it is essential to capture its dynamic nature as part of any traffic state prediction algorithm. Predicting the upcoming driving environment can be performed at:

- *individual* vehicle level (i.e., microscopic) such as predicting a vehicle's trajectory over time; or
- at *aggregated* level considering all the vehicles in a traffic stream (i.e., macroscopic), such as predicting the average traffic flow or density.

Traffic state prediction has been historically performed at the aggregated level, while predicting the driving environment at the individual level has been studied only for automated vehicle motion planning (to plan a safe path and avoid collision with other traffic agents and obstacles). This study aims to utilize the data from connected automated vehicles to link these two approaches and increase the accuracy of traffic state prediction. Note that the accuracy of prediction usually decays with the increase in the prediction horizon. For a longer prediction horizon, predicting the driving environment at the aggregated level in the forms of traffic flow, density, and speed can be more accurate than predicting at the individual level. On the other hand, short-term prediction at the individual vehicle level can be more accurate than aggregated level predictions (due to capturing the interactions among vehicles). Consequently, it is critical to study both approaches and investigate the possibility of linking them for higher accuracy predictions.



(a) Connected Vehicles (b) Time-space Diagram (c) Flow and Density Plot

Figure 1.1: Connected vehicles data can be used to monitor traffic state over time and space.

1.1.1 Knowledge Gap and Research Contributions

While there has been several studies that have utilized connected vehicle data for more accurate traffic prediction, the majority of them still follow the same standard aggregate-based prediction approaches (although with more comprehensive and accurate data from connected vehicles). This study aims to address the existing challenges in this area and significantly improve the accuracy of traffic prediction by capturing the interactions among vehicles.

In the case of traffic prediction at the aggregated level, most of the existing data-driven prediction models utilize aggregated traffic measures (input and output) in the prediction process [7]. Unfortunately, the dynamic nature of traffic flow combined with the limitations of conventional data sources have historically prevented accurate traffic state prediction. Moreover, with the increase in traffic flow and density, the impact of unexpected individual driving behavior on the traffic state increases [8]. Consequently, capturing the interaction among individual vehicles can potentially improve the traffic state prediction.

The driving behavior and vehicle interactions can be best captured by the time-space diagram of the traffic stream (see Figure 1.1b). This comprehensive diagram contains all the information required to estimate the microscopic and macroscopic characteristics of the traffic stream. While the time-space diagram has been at the core of many traffic flow theory innovations, up until recently, no scalable approach existed to generate this diagram based on real-world data. Fortunately, the recent advancements in connected vehicles technology have provided the opportunity to construct a new data source at the individual vehicle level. Accordingly, the time-space diagram of the vehicles can be constructed from the data that connected vehicles share in this connected environment. One of the main contributions of this dissertation is to introduce traffic prediction methodologies based on convolutional neural networks (CNN) that can directly utilize the time-space diagram in the prediction process. These methodologies offer two key innovations: (1) they provide the capability of directly utilizing the time-space diagram as the input without the need for any aggregation or abstraction; (2) the convolutional layers provide the opportunity to capture and learn the features embedded in the time-space diagram required for an accurate prediction, such as the traffic dynamics, the interaction between the vehicles, and traffic shockwaves.

Traffic flow changes based on the behavior of individual vehicles. In fact, predicting the driving environment at the individual vehicle level is challenging due to the dynamic nature of the driving environment, the complex interactions among vehicles, and differences in driving behavior among different drivers. Structuring the driving task into multiple maneuvers based on the interaction among vehicles can improve vehicle trajectory prediction. These maneuvers represent the drivers' possible responses to their surrounding environment and interactions with other vehicles. Obviously, each vehicle's future trajectory changes depending on the driver's choice of maneuver. Since more than one maneuver is feasible in many driving scenarios, focusing on a single driving maneuver cannot provide a comprehensive picture of potential future trajectories. As a result, a probabilistic approach should be adopted to predict the future movement of the vehicle conditioned on different maneuvers considering the interactions among the vehicles. Despite the advantages of traffic state prediction using a time-space diagram and CNN structures, combining these probabilistic predictions into a traffic state prediction can explicitly capture the impacts of interactions on traffic state and significantly increase the accuracy of prediction. Accordingly, another key contribution of this dissertation is to introduce a methodology to convert such probabilistic trajectory predictions to aggregated traffic state predictions (i.e., flow, space-mean speed, and density). The key advantages of this approach over directly predicting traffic state based on aggregated traffic data are: (1) the ability to capture the impacts of interactions among vehicles on traffic flow dynamics to increase the accuracy of the predictions; (2) capturing and characterizing the uncertainties in individual vehicle's maneuvers and the possibility of various configurations and outcomes. Note that while aggregated level data-driven methodologies provide the means to predict for any prediction horizon, they do not have the means to explicitly capture the uncertainty in prediction as the prediction horizon changes. In other words, their underlying models (that are well-trained for a particular prediction horizon) can be incapable of accurately predicting for any horizon (other than the one that they have been trained to predict). The proposed approach of this dissertation, on the other hand, provides a probabilistic traffic state prediction for any time step between prediction time and the prediction horizon, accurately capturing the decrease in prediction accuracy as the prediction horizon increases.

Finally, there is a key advantage of using individual trajectories for traffic state prediction. Connected automated vehicles rely on the trajectory-level prediction of their surrounding traffic environment to plan a safe and efficient path. By sharing such predictions, one can estimate the future traffic state in a distributed manner. Moreover, utilizing such information means that the majority of the vehicles in the traffic stream (and their trajectories) can be captured by a limited number of connected automated vehicles. This means that an accurate time-space diagram can be generated at a low market penetration rate of these vehicles.

1.1.2 Objectives

The main objectives of this dissertation can be summarized as follows:

- Introducing novel methodologies to predict the evolution of driving environment at the aggregated level considering the vehicle interactions and driver behaviors based on data from connected and automated vehicles.
- Introducing a methodology to estimate probabilistic measures of the traffic state at the aggregated level (e.g., flow, density, space-mean speed) considering probabilistic individual-level predictions.

1.2 Organization

The remainder of this thesis is organized as follows: Chapter 2 provides a brief background on the existing traffic state and trajectory prediction models, as well as the existing vehicle trajectory datasets to support the development of those models. Chapter 3 presents the simulated and real-world trajectory datasets utilized in this dissertation for the training and evaluation of the proposed data-driven traffic state prediction models. Chapter 4 introduces a matrix representation of the time-space diagram and introduces a traffic prediction model based on a convolutional neural network (CNN) to predict the traffic flow and density using the time-space diagram as input to the model. Chapter 5 adopts the matrix representation of the time-space diagram and proposes a convolution encoder-decoder model to predict the formation of traffic shockwaves over time and space. Chapter 6 proposes a methodology to predict the traffic state at the aggregated level based on probabilistic predictions at the individual vehicle level. Finally, Chapter 7 provides a summary of the findings and concluding remarks.

CHAPTER 2

LITERATURE REVIEW

2.1 Traffic State Prediction

Traffic prediction is one of the principal components of intelligent transportation systems. The energy management system of automated and connected vehicles can adopt appropriate control actions to improve energy consumption based on the predicted traffic state. In general, traffic state prediction involves the process of predicting traffic state variables, such as flow, density, speed, and travel time, from the observed data.

The traffic state prediction approaches differ based on the type of input data, traffic flow model, and estimation methodology [9]. Van Lint and Van Hinsbergen [10] classify traffic prediction methodologies into naive, parametric, and non-parametric approaches.

Naive approaches refer to the methodologies that either assume the upcoming traffic state remains constant or close to the current state [11] or stays near the average of historical observations for the time of the prediction [12]. The naive prediction approaches are limited to traffic conditions that are almost constant for an extended period of time (note that such conditions are rare) or have nearly constant historical (e.g., daily, weekly) traffic patterns.

Parametric approaches refer to the methodologies that use a traffic flow model with parameters calibrated on historical data or in combination with new observations. The fundamental diagram is one of the well studied traffic flow models relating the macroscopic characteristic of traffic state. In traffic prediction, the fundamental diagram is mostly used in combination with first and second order traffic flow models to estimate the state of the segments of a roadway or network from a set of noisy traffic measurements collected from sensors or probe vehicles [13, 14, 15, 16, 17]. Mesoscopic traffic prediction models are another type of parametric models. The mesoscopic models predict the traffic state by tracking individual vehicles considering the macroscopic state of the traveling links [18]. One of the challenges in the use of parametric models is the trade-off between accuracy and the complexity of traffic prediction models, especially when addressing the time-variant and abnormalities in traffic dynamics.

Non-parametric approaches, on the other hand, are usually based on simple data-driven methodologies that do not explicitly rely on a traffic flow model. There is a wide range of data analysis and machine learning approaches used in the field of traffic prediction. Some of the common approaches include linear regression [19, 20], different classes of neural network [21, 22, 23, 24], support vector regression [25, 26, 27], and time series prediction [28, 29, 30]. Data-driven approaches for traffic prediction have gained more attention in the past decade with the increase in data availability. Most of the existing non-parametric approaches rely on aggregated and macroscopic traffic data such as flow, density, and speed with limited studies focused on the trajectory of vehicles [7]. With the increase in flow and density (especially around the breakdown point), the impact of unexpected driving behavior on the state of the traffic stream increases, and more complex dynamics can be observed in the traffic flow [8]. Consequently, capturing the interactions among vehicles can lead to a better traffic prediction.

2.2 Trajectory Prediction

Predicting the movement of traffic agents in the surrounding driving environment is essential for the safe path planning of automated vehicles. Lefevre et al. [4] classifies the motion prediction methodologies based on their modeling hypotheses into three groups of Physics-based, maneuver-based, and interaction aware.

The physics-based model refers to the simple methodologies that consider the movements of vehicles with respect to dynamic or kinematic motion models without considering the vehicle's maneuvers or interaction among traffic agents. Dynamic modeling is usually adopted in control applications for an ego vehicle. However, measuring a target vehicle's position, speed, and acceleration is easier than measuring the forces resulting in the vehicle's motion. Consequently, kinematic models are simpler and more popular than dynamic models to predict a target vehicle's future state [4]. Some studies such as [31] assume that the data on the vehicle's current state and the motion prediction model is adequate to estimate the future state of the target vehicle without consideration of any uncertainties. Some studies such as [32, 33] consider uncertainties in measurements and adopt Kalman Filter in motion modeling. Monte Carlo methods can also be used to sample from input data for the motion model to simulate different possibilities of the target vehicle's future state [34]. Unrealistic predictions can be removed in the post-processing of trajectories or by eliminating unrealistic input to the motion models. The physics-based models do not consider the vehicle's maneuver or the interaction among vehicles, making them limited to short-term trajectory prediction [4].

Maneuver-based models recognize the different vehicle maneuvers and predict the vehicle's movement considering the intended maneuver. In this approach, the intended maneuver is first detected by comparing the observed partial trajectory with a set of prototype trajectories for different maneuvers. Then the trajectory of the vehicle is predicted based on the detected maneuver. The prototype trajectory approach assumes that the vehicles' trajectories can be grouped into different motion patterns represented by prototype trajectories learned from previous observations (i.e., from a training set). Gaussian Process has proven as a good modeling approach for trajectory prototyping [35, 36, 37]. The partially observed trajectory is compared to the prototype trajectories, and the rest of the trajectory is predicted based on the closest prototype trajectory. Some other studies identify the vehicle's maneuver by classifying the partially observed trajectory using machine learning approaches such as support vector machine [38], multilayer perceptron (MLP) [39], logistic regression [40] or even recurrent neural networks [41]. Kinematic motion models predict the rest of the vehicle's trajectory based on the detected maneuver. Maneuver based trajectory prediction models are more accurate than simple physics-based models; however, the maneuver-based models also do not consider the entire interaction among the traffic agents that could impact the vehicle's trajectory [4].

Interaction-aware methodologies consider the interaction among traffic agents when predicting the maneuver and trajectory of a target vehicle. A simple consideration of vehicles' interaction is using trajectory prototyping and dropping the conflicting trajectories between the pair of vehicles [42]. Another approach is to consider pairwise interaction among all the vehicles in the driving environment; however, the number of pairwise combinations increases quadratically with the number of vehicles in the scene. Some interaction-aware studies [5, 43] simplify the problem by assuming an asymmetric interaction between the target vehicle and environment such that the environment impacts the target vehicle, but the target vehicle does not impact the environment. The interaction-aware prediction methodologies are more accurate than the physics-based, and maneuver-based methodologies as they consider the interdependency among the movement of traffic agents and consequently can be used for a longer prediction horizon [4].

Recent trajectory prediction studies are mostly based on deep learning approaches considering the interaction among the vehicles. The interaction among vehicles is captured in the input representation of the driving environment to the deep learning prediction model [44]. The driving environment representation to these models could be in the form of trajectory history of the target vehicle and surrounding vehicles [45, 46], or in the form of bird's eye view representation of the processed surrounding environment [5, 47] or in the form of raw sensory data [48]. The deep learning-based trajectory prediction models are mostly based on Recurrent neural networks (RNN) [45, 46] or convolutional neural networks (CNN) [47, 48] or a combination of the two types of network [5]. RNN architecture has a high capability to capture temporal dependencies in sequence data, and CNN architecture has a high capability to capture spatial dependencies in feature maps.

Deep learning-based approaches have indicated great performance to capture the interaction among the vehicles in predicting a target vehicle's trajectory. Most of the existing deep learning models rely on the trajectory history of the target and surrounding vehicles; however, other variables such as traffic conditions, traffic rules, and road geometry also impact the interaction among the vehicles and their future trajectory[44]. Consequently, it is expected inclusion of additional traffic-related data into the model could improve the prediction of these models.

2.3 Vehicle Trajectory Dataset

Vehicle trajectory is a concise yet comprehensive way to store data of an individual or collective group of vehicles for both micro- and macro-level traffic analyses. With the advancements in sensing and imaging technologies, the trajectories can be generated using cameras, infrared sensors, RADAR, and LiDAR. However, video-based imaging has been the most popular method of extracting vehicle trajectories. The early studies collected vehicle trajectories using pole-mounted cameras at intersections [49, 50]. Aerial imagery for trajectory extraction overcomes issues related to occlusion and cluttering that are associated with using pole-mounted cameras. Satellites, helicopters, Unmanned Aerial Vehicles (UAVs), and airplanes are the primary means of obtaining aerial videos and images.

Some of the existing vehicle trajectory datasets are FHWA Next Generation Simulation Models (NGSIM)[51], Strategic Highway Research Program (SHRP2)[52], and TrafficNet [53]. NGSIM is a well-known open-source trajectory dataset collected in 2006 using digital cameras at different locations, including US Highway 101 and Interstate 80 freeway. The vehicle trajectories are extracted from the images of multiple cameras combined to create a single image that looks like an aerial shot. The NGSIM trajectory data contains the location of each vehicle at a frequency of 10 Hz over a 1600 to 3200 feet (488 to 975 meters) stretch of roadway. However, the NGSIM data suffers from noise and inaccurate detection due to the low-resolution cameras at a considerable distance. [54] analyzed the NGSIM dataset and confirmed inaccuracies in the speed and positioning of vehicles. The SHRP2, in collaboration with Virginia Tech Transportation Institute (VTTI), had collected naturalistic driving data in 2012 [52]. The dataset includes more than 5 million trips that include sensory data such as speed, location, acceleration, and also vehicle and driver characteristics. The SHRP2 dataset is not freely available to public access and has a limited preview. This dataset is collected using probe vehicles, and the collected data is limited to the field of view of the onboard sensors and does not entirely define the surrounding vehicles and traffic dynamics. The TrafficNet provides processed naturalistic data with libraries for researchers to perform data analytics [53]. TrafficNet separated driving into six scenarios, such as free flow, car-following, cut-in, and the like, and classified the entire dataset into these scenarios curated to research. It is a web-based platform with a MYSQL database used to store the information. The HighD dataset [55] is another dataset that offers vehicle trajectories on German Autobahn. HighD accounts for variability in traffic composition by collecting data at six different locations. It has a truck ratio varying from 0 - 50 % and trajectories collected at different times of the day. The trajectories are analyzed and classified into specific maneuver types, such as lane changes and critical maneuvers. More recently, the pNEUMA dataset [56] used a swarm of drones to collect arterial traffic data in sequential sessions with blind gaps in between sessions. Their objective was to study Origin-Destination information, travel time and congestion propagation, and lane-changing behavior.

While the datasets mentioned earlier provide the means to analyze driver behavior, they fail to provide any information on the utilization of the ADAS by drivers. This is unfortunate, as ADAS technologies are becoming an integral part of our roadways, and capturing their impacts on traffic flow dynamics and congestion is critical. There is a key challenge associated with utilizing the existing datasets: considering that ADAS technologies have a compound annual growth rate of 12% [57], there is a very high chance that some of the vehicles in the existing datasets already utilized ADAS technologies. Assuming that ADAS technologies can potentially change the interactions among drivers on the road and lead to new traffic flow dynamics and possibly new types of high-risk driving instances, utilizing these datasets with the assumption that human drivers control all vehicles can lead to unrealistic assessments and bias. One of the more recent studies by Makridis et al. [58] has also recognized these shortcomings and developed the OpenACC database. It is an open-access database with two public and two private test campaigns conducted in Sweden and Hungary. The data collection process involves using onboard sensors such as accelerometers and global navigation satellite systems (GNSS) to record the trajectories of the probe vehicles. This limits the microscopic and macroscopic analysis to only probe vehicles, and the interaction between the ACC and conventional vehicles in the traffic is not captured.

CHAPTER 3

DATA

3.1 Introduction

Vehicle trajectory is a concise way to store data of an individual or collective group of vehicles for both micro- and macro-level traffic analyses. Some of the existing vehicle trajectory datasets are FHWA Next Generation Simulation Models (NGSIM)[51], Strategic Highway Research Program (SHRP2)[52], and TrafficNet [53]. Depending on the data collection objective, most of the existing vehicle trajectory datasets are collected at a specific location and time (e.g., NGSIM) or limited to the field of view of onboard sensors of the probe vehicles in the data collection and does not entirely define the surrounding vehicles and traffic dynamics (e.g., SHRP2 and TrafficNet).

This dissertation adopts data-driven methodologies for predicting upcoming driving environments. These methodologies require accurate and reliable data for the training process. As a result, two types of vehicle trajectory datasets are utilized to help develop the traffic prediction models in this dissertation.

- Simulation-based data collection to create an extensive dataset that includes different levels of traffic state (from free-flow to fully congested) at different posted speeds for the training of the models that can generalize well.
- Field-based vehicle trajectory data (real-world) for both training and evaluation of the proposed models of this dissertation.

3.2 Simulated Trajectory Data

To construct a comprehensive and large dataset, this dissertation implements a microscopic traffic simulator written in the Python programming language to collect the trajectory of the vehicles. The microscopic simulator adopts the Intelligent Driver Model (IDM) [59] as its car-following[1] logic and MOBIL [60] as its lane-changing[2] logic.

The simulation collects the trajectory of the vehicles traversing a one-lane or three-lane roadway segment with a length of 40000 feet over 15 minutes. At every simulation run, unique and random IDM and MOBIL parameters are assigned to every vehicle to make the simulation more realistic. In addition, two types of disturbances are used in the simulation to create a dataset with different traffic states from free-flow to fully congested. Sudden deceleration of a random vehicle for a small period (e.g., 15 seconds) creates a speed drop perturbation and disturbs the traffic stream. Another type of disturbance used in the simulation is forcing a random vehicle to move slower than the desired speed for a more extended period, such as 5 minutes, to create congestion and traffic breakdown. Both disturbances result in the formation of shockwaves in the traffic stream. Moreover, for each simulation run, the desired speed can be set or randomly selected from different speed limits including 30, 45, 50, 55, 65, 70, 75 miles per hour (mph) to create a comprehensive dataset.

3.3 Field-Based Vehicle Trajectory Data

3.3.1 Trajectory Data Collection Using Aerial Videography from Both Conventional Vehicles and Vehicles using ADAS Technologies

Existing trajectory datasets fail to provide any information on utilizing new technologies such as ADAS. Utilizing these features by drivers can potentially change the interactions among drivers on the road and can lead to new traffic flow dynamics and possibly new types of high-risk driving instances. Considering that ADAS has a compound annual growth rate of 12% [57], it is valuable to collect new trajectory data from both conventional vehicles and vehicles using ADAS technologies.

Video-based imaging has been the most popular method of extracting vehicle trajectories. Satellites, helicopters, and airplanes are conventional means of obtaining aerial videos and images. Moreover, high-resolution cameras provide the opportunity to capture a longer stretch of roadway. The key idea to extract the vehicle trajectories from aerial images is to detect vehicles in a sequence of images using computer-vision based techniques. The early works in vehicle detection from aerial images [61, 62] used a combination of edge-detection techniques with morphological operations to detect vehicles on low-resolution images. Feature-based extraction and classification techniques were used to detect and classify

[1]A car-following model determines how vehicles follow each other on a roadway

[2]A lane-changing model determines how vehicles change lanes to improve their state safely.

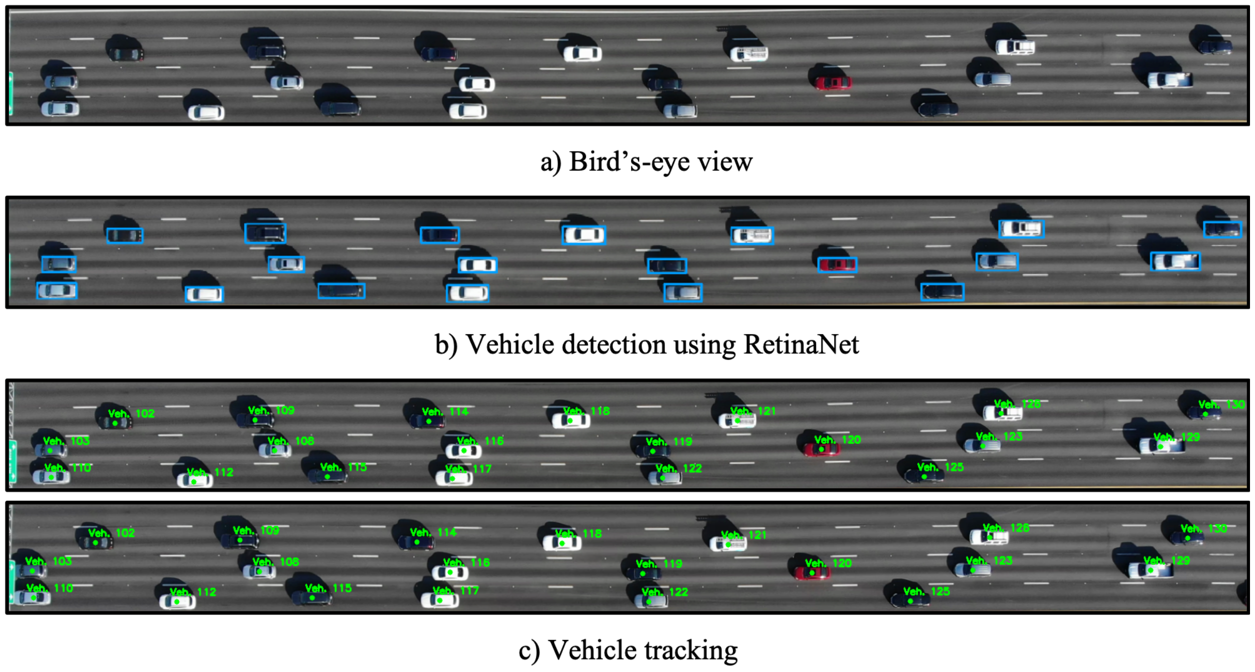


Figure 3.1: Vehicle detection and tracking in aerial images.

vehicles [63, 64] before the introduction of deep neural networks. Deep convolutional neural networks (CNN) significantly improved the task of object detection due to their capacity to learn complex features compared to simple or engineered features. [65] used the CNN framework YOLO (You Only Look Once) [66] to detect and track vehicles in real-time with high accuracy and computational efficiency.

Recent years have seen a significant increase in using aerial vehicles for remote sensing applications such as photogrammetry imaging. The trajectory of the vehicles can be extracted from the video frames recorded in the bird's-eye view from a segment of the roadway (Figure 3.1a). In every video frame, the location of the vehicles can be estimated for a fixed coordinate system and reference point on the ground. Every video recording is converted to a sequence of images (i.e., frames) separated at a constant rate over time (e.g., 25 frames per second). Tracking the location of any vehicle over the sequence of images enables extracting the vehicle's trajectory over time.

The vehicle trajectory extraction is performed in four steps: *image stabilization*, *vehicle detection* (Figure 3.1b), *vehicle tracking* (Figure 3.1c), and *trajectory construction*. In the image stabilization step, all the images are transformed to match a reference field of view. Then the vehicles are detected in every frame and tracked over the sequence of images. Finally, the vehicle's location and trajectories are constructed by converting the image coordinates to the adopted reference coordinates on the ground.

The key contribution of this chapter is to present a pipeline to convert video files to vehicle trajectories. Although the utilized models and algorithms mostly exist outside of this pipeline, to the best of our knowledge, this dissertation is one of the very few research studies that outline the pipeline and steps needed to extract the trajectory of vehicles using aerial videography. It should be noted that the proposed trajectory extraction methodology is modular, and there exist different candidate models and algorithms that can be adopted for each of the four steps. Therefore, this dissertation does not focus on comparing all the possible candidate models for each of the trajectory extraction steps. In return, this dissertation focuses on keeping the methodology simple and introducing some practical and popular models that could be utilized in each step. These steps are further elaborated in the following sections.

Image Stabilization

The location of every vehicle in an image frame is estimated by converting its position on the image map to the fixed coordinate system picked on the ground. Consequently, it is essential to find the mapping function between the image coordinate to the adopted ground coordinate. Image stabilization is the process of converting the field of view of all the image frames to a reference image for which the mapping function to the ground coordinate is known. Figure 3.2 presents an example of a reference image and the input and output of the image stabilization. Ensuring that the input image covers all (or majority) of the study segment is vital. Otherwise, as depicted in Figure 3.2, the missing areas will be black in the stabilized image.

The image stabilization is performed in three steps; first, detecting the key features in both reference and input images; second, finding the matching features between the two images; and third, estimating transformation between them. The process searches for specific unique patterns, such as corners, which are good features that can be tracked from one image to another image. There exists different algorithms for good features detection in images such as Harris corner detector [67], Scale-Invariant Feature Transform (SIFT) [68], Speeded up Robust Feature (SURF) [69], and Oriented FAST and Rotated BRIEF (ORB) [70]. ORB and Harris corner detectors are among the fast feature detectors. However, SIFT and SURF are the top-performing feature detectors in terms of scale and transformation. The result of the comparison between the SIFT, SURF, and ORB by Karami et al. [71] suggested that SIFT has the best performance in most image distortion scenarios. As a result, this dissertation employs the SIFT algorithm to detect good features. The SIFT algorithm improves the image stabilization by identifying features in both reference and input images

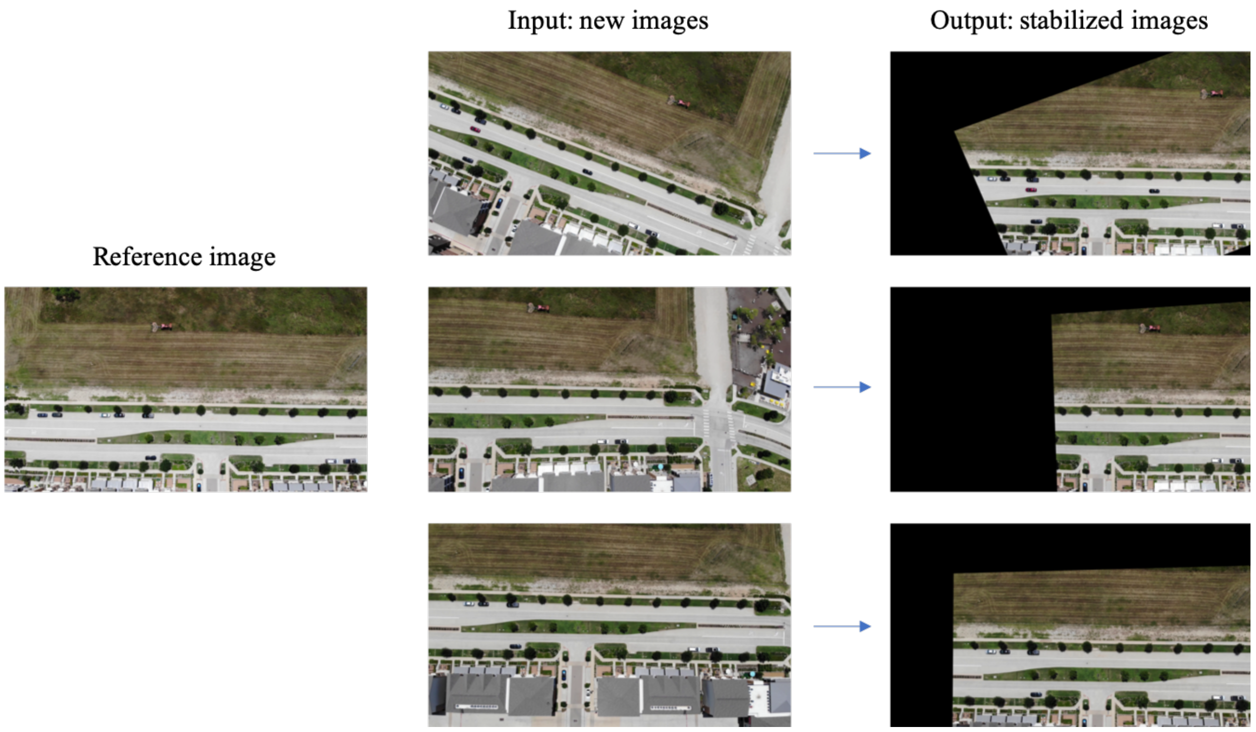


Figure 3.2: Image stabilization.

that are invariant to distortions such as rotation, scale, and point of view (see Figure 3.2).

The second step in image stabilization is matching the features between the reference image and the input image. One naive approach is to compare every feature in the reference image with every feature in the input image to find the best matching pairs. This approach is known as brute-force matching. The deficiency with this approach is that it takes a lot of computation time that is impractical for a video data collection at a high frame rate (e.g., 25 frames per second). Instead, to improve the computation speed, the Fast Library for Approximate Nearest Neighbors (FLANN) matcher [72] is utilized to match features between the images. The FLANN algorithm adopts a distance measure (e.g., L2 norm) to compare the level of matching between the descriptors of two features from two images. A lower distance between the descriptors indicates a better match between the two features. For every feature in the first image, the FLANN algorithm finds the approximate K best (i.e., nearest) matching features in the second image.

The algorithm may identify many matching features; however, the top matching pairs are picked considering Lowe's ratio test [68]. In this test, features m (in reference image) and n (in input image) are a top matching pair if n is the best match for m. Also, the distance between the descriptors of m and n should be less than a threshold multiplied by the distance between m and its second-best match. Accordingly, $K = 2$ for the FLANN algorithm and the threshold of 0.7, as recommended by Lowe [68], are adopted to find the top pairs of matching points.

The final step of the image stabilization is finding the perspective transformation between the reference and input images considering the best matching features among them. Homography, H, is a 3×3 transformation matrix that maps the points (x_2, y_2) from one image to the points (x_1, y_1) in another image in accordance with the following equation:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \tag{3.1}$$

Four correct matching points are enough to estimate the homography transformation between the two images. However, there are chances that some of the matches are incorrect. The Random Sample Consensus (RANSAC) algorithm [73] is a technique to find the model parameters from a dataset with many outliers. The RANSAC is an iterative process that searches for the model parameters that agree with most data points. As a result, the RANSAC technique is utilized to find the homography parameters from the best matches between the images. Then, every pixel in the input image can be transformed to the perspective of the reference image using the homography matrix to create a stabilized image.

Vehicle Detection

Object detection is the task of providing both the class and location of the objects in an input image. The classical approach for object detection is to identify the informative regions in the image that contain objects of interest, then extract semantic and representative features from them, and finally, classify the objects in those regions. Deep neural networks (DNN) made a great performance breakthrough in the task of object detection due to the capacity of the convolutional neural networks (CNN) to learn more complex features compared to shallower models [74]. There are multiple popular CNN based object detectors such as R-CNN [75], Fast R-CNN [76], Faster R-CNN [77], SSD [78], YOLO [66] and RetinaNet [79].

Some of the DNN object detectors, such as the R-CNN family [77, 75] are two-stage object detectors. In the first stage, regions of interest are identified, and these regions are classified in the second stage. Other networks such as the SSD [78], YOLO [66] and RetinaNet [79] are one stage object detectors. One-stage approaches perform the detection in one shot by classifying fixed and dense regions of the image and adjusting the location and size of each region to enclose the object inside it. One-stage object detectors are faster than the two-stage detectors but usually have lower accuracy. The lower accuracy of the one-stage approaches is due to the extreme imbalance between the number of foregrounds (objects of interest) and background regions evaluated in every image that affect the network's training. Conventionally, the cross-entropy loss function is used to train the classification networks. The following equation estimates the cross-entropy loss for an evaluated region for K different classes.

$$Loss_{(cross-entropy)} = -\sum_{i=1}^{K} Y_i \log(p_i) + (1 - Y_i)\log(1 - p_i) \tag{3.2}$$

where p_i is the probability of class i estimated by the network, and Y_i is equal to 1 for the true class and 0 for the rest of the classes. The background regions are considered easy examples (i.e., when p_i is relatively high) to be detected by the classification network. The high number of backgrounds in the training set dominates the loss function and prevents training. RetinaNet [79] proposes the use of focal loss for the classification loss function. The focal loss uses the focusing parameter γ that puts less emphasis on the easily classified examples (i.e., when p_i is relatively high) such as backgrounds and puts more emphasis on hard examples (i.e., when p_i is relatively low). In addition, the balancing parameter α_i is used to mitigate the effect of the imbalance among different classes. The following equation presents the focal loss estimation for one region and K classes of objects.

$$Loss_{(focal-loss)} = -\sum_{i=1}^{K} \alpha_i Y_i (1 - p_i)^{\gamma} \log(p_i) + (1 - \alpha_i)(1 - Y_i) p_i^{\gamma} \log(1 - p_i) \tag{3.3}$$

Adopting the focal loss for the classification significantly improves the accuracy of the RetinaNet compared to other one-stage networks and makes the RetinaNet an ideal object detector in terms of speed and accuracy.

RetinaNet

This dissertation utilizes the RetinaNet to detect vehicles in the aerial images; however, it should be noted that any of the modern CNN-based object detection models have the potential to serve as the vehicle detection module in the proposed model. Figure 3.3 presents a high level network architecture of RetinaNet. This network comprises four types of subnetworks, including ResNet, feature pyramid network, and multiple boxes and classification subnets. The resNet component of the network is the fully convolutional part of the ResNet,

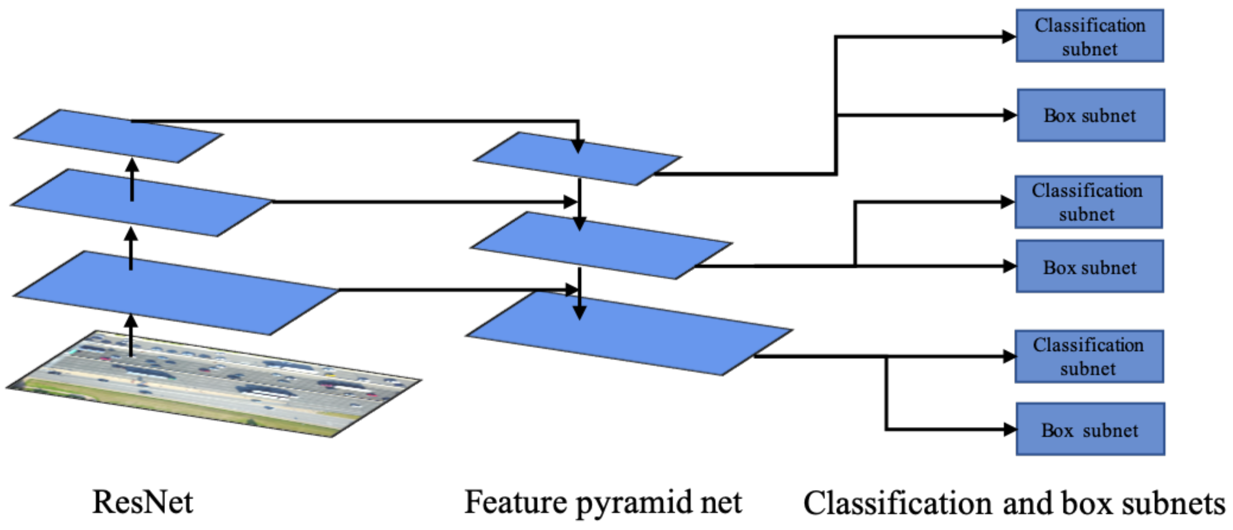


Figure 3.3: RetinaNet architecture.

which enables the RetinaNet to take input images with any size [80]. At every layer of the ResNet, the feature map from the previous layer is transformed into a new feature map with a stronger semantic level. The feature map transforms from the bottom (i.e., input layer) of the ResNet to the top of it, while the spatial resolution of the feature map decreases, which is concerning for small objects. The classification and box subnetworks evaluate regions with fixed sizes and aspect ratios centered at every location in the feature map for potential objects. Consequently, both a strong semantic level and proper spatial resolution are the requirements to detect objects in every feature map. The ResNet is complemented with the feature pyramid network to address these requirements. The feature maps are upscaled using the nearest neighbor upsampling, moving from the top of the feature pyramid network to its bottom. Each feature map in the feature pyramid network has a corresponding feature map with the same scale in the ResNet connected to it with lateral connections to improve its spatial resolution. The bottom-up and top-down transformation of the feature maps, as well as the lateral connections, are presented in Figure 3.3. ResNet and feature pyramid network construct the backbone of the RetinaNet that provides five levels of semantically and spatially strong feature maps (only three levels shown in Figure 3.3).

Each feature map level is used as the input to separate classification and box subnetworks. At every location on the feature map, A rectangular regions with different aspect ratios and sizes centered at that location are evaluated for the presence of any of the K object classes inside it. These rectangular regions are also known as anchor boxes. These anchor boxes are the initial estimate of the areas enclosing a single object. The classification subnetworks are fully convolutional networks that output a tensor with size $(W, H, K \times A)$. W and H are

Table 3.1: Training dataset for aerial vehicle detection.

Complete Training Dataset	
Small vehicles	11379
Large vehicles	751
Total	12130

the height and width of the feature map that is proportional to the input image size. The box subnetwork estimates four offset variables for each anchor box. These offsets are the differences between the center, height, and width of the anchor box and the actual bounding box enclosing the object. The box subnetwork is also a fully convolutional network that outputs a tensor with the size of $(W, H, 4 \times A)$. The fully convolutional structure of all the subnetworks of the RetinaNet enables it to take any image size as the input for inference.

Training RetinaNet

The open-source Keras library for RetinaNet, developed by Gaiser et al. [81], is used to build the network for vehicle detection in aerial images. The vehicles can be grouped into small and large vehicles based on their size and appearance. Accordingly, the classification subnet is adjusted to accommodate two classes of objects. In addition, the ResNet-50 [80] pretrained on Microsoft coco dataset [82] is adopted as the backbone for the RetinaNet. The resulting network contains more than 37.2 million parameters; however, the parameters of the backbone are fixed, which reduces the trainable parameters to nearly 13.7 million.

More than two hundred images are sampled from the collected aerial images to create a training dataset. The sampled images are manually annotated using the annotating software developed by Dutta and Zisserman [83]. This training dataset includes annotated aerial images of large and small vehicles. For this dissertation, car, van, and pick-up truck instances are combined into the small vehicle class, and trucks, busses, and recreational vehicles (RVs) are combined into the large vehicle class. The training dataset consists of 220 aerial images with 4K (4096×2160 pixels) resolution, and the number of instances for each class is presented in Table 3.1.

The dataset is divided into training (training and validation) and testing sets with the proportions of 0.9 and 0.1, respectively. In addition, data augmentation in the form of random image transformation and scaling is used to further improve the training dataset's size and, ultimately, the generalization of the network. The loss function of the RetinaNet considers both the focal loss for classification task (Equation 3.3) and the standard smooth L_1 loss [76] for box regression. Training is the iterative process of adjusting the trainable parameters of the model to gradually minimize the loss function. The focusing parameter,

Table 3.2: Detection model performance on testing dataset.

	Instances	Average Precision
Small vehicles	1316	0.9938
Large vehicles	98	0.9302
mAP using weighted average of precision		0.9894
mAP		0.9620

$\gamma = 2$, and balancing parameter, $\alpha = 0.25$ are adopted for the focal loss based on the findings of [79]. The model is trained with a batch size of four images due to the large image size (4096×2160 pixels). The batch size could be increased by reducing the image size by cropping out the study area's background or resizing the images. The model trained up to 45 epochs (45 complete iterations over the entire dataset) is selected as the vehicle detector in the aerial images. Table 3.2 presents the performance of the trained model on the test dataset in the form of average precision for each class and the mean Average Precision (mAP) for Intersection over Union threshold (IoU) of 0.5. The trained RetinaNet is used for the vehicle detection step of the trajectory extraction. The input to the vehicle detection is a stabilized image, and the output is the coordinates of the bounding boxes enclosing the vehicles in the image. Figure 3.1.b presents the detected vehicles and their visualized bounding boxes.

Vehicle Tracking

Tracking is the process of linking the new detections to previous observations. The tracking methodology adopted for this dissertation includes data association and track maintenance. Data association is the process of associating the detected vehicles in the current frame to the vehicles identified in the previous ones. The track maintenance is in charge of initiating new tracks, maintaining them, and deleting them.

The track maintenance initiates tracks with unique ids to all the vehicles detected in the first frame. After that, for every frame, all the newly detected vehicles are compared with the existing tracks using the data association. The tracks are updated as a new detection is associated with them. A new track is constructed for any new observation that is not associated with the current tracks. Moreover, if a track is not updated in the last n previous frames, the track maintenance deletes that track. A track object maintained by the track maintenance contains both the unique id of the track and the coordinates of the center of the bounding box of its last observation. $n = 5$ is found to be appropriate for the images collected at 25 frames per second when using the trained vehicle detector (i.e., RetinaNet) of this dissertation.

The simple data association considers the euclidean distance between the center of the bounding boxes of the newly detected vehicles and the ones for the tracks using a greedy policy. Every new observation is associated with the closest track, considering the Euclidean distance between the center of the bounding box of the new detected vehicle and the ones for the other tracks. This nearest center association is based on the assumption that the movement of the center of a vehicle between frames is less than the distance between the center of different vehicles. This assumption can be validated by estimating the maximum movement of a vehicle at maximum speed between two frames compared with the center to center lateral and longitudinal spacing between the vehicles. Moreover, the tightness of the bounding boxes from the vehicle detection plays a key role here in reducing the noise in the measured euclidean distance between the bounding boxes. The input to the vehicle tracking component is the set of bounding boxes detected for each frame, and the output is in the same format but with a vehicle id added to each bounding box. The simple data association component of the tracking step works very well for this dissertation due to: (1) images are stabilized and share the same field of view, (2) the high frequency of images ensures the movement of the vehicles is much less than the lateral and longitudinal spacing between the vehicles, and (3) the data is collected in bird's eye view with no occlusion. For the cases that the trajectory data collection is performed using cameras mounted on a structure with the possibility of occlusion in images and also lower frame rate, more advanced association measures such as Mahalanobis distance with Kalman Filter, or even more advanced tracking algorithms such as Deep SORT [84] can be adopted. It should be noted that if the vehicle detection component fails to detect a vehicle in more than five frames, a new tracking id will be initiated, which is not desired. Figure 3.1.c presents the vehicle ids tracked for two images, five frames apart for a video with 25 frames per second rate.

Trajectory Construction

The aerial images are transformed and stabilized, considering a reference image before extracting the vehicle trajectories. The trained RetinaNet detects the vehicles in the stabilized images, and the resulting bounding boxes are used to track the vehicles from one frame to another. In the case that the vehicle detector (i.e., RetinaNet) fails to detect a previously seen vehicle in a frame, the track maintenance keeps its track active up to n frames. If the previously observed vehicle gets detected again within the n frames, its track is updated, and the same vehicle id is assigned to its bounding box. Also, the bounding boxes for the missing frames are interpolated between the frames before and after the missing frames.

The bounding boxes represent the location of vehicles in image coordinates (i.e., row and column of pixels). For trajectory extraction, these coordinates need to be converted to a fixed ground coordinate system (e.g., feet or meters). A digital image is a 3D tensor, and every pixel is located by its row and column number in the image tensor. The location of every vehicle in an image frame is estimated by converting its position on the image map to the fixed coordinate system picked on the ground. Consequently, it is essential to find the mapping function between the image coordinate $(row, column)$ to the adopted ground coordinate (x, y):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} row \\ column \\ 1 \end{bmatrix} \tag{3.4}$$

The above transformation matrix has six parameters that can be calibrated using coordinates of three known points on the ground and image. In the presence of more known matching points with outliers, the RANSAC algorithm [73] introduced in the image stabilization section can be used to find the parameters of the transformation matrix that agree with most of the matching data points. In this dissertation, the reference image is picked such that the roadway is parallel to the row axis in the image map. In this case, the X axis is picked along the direction of traffic and the Y axis perpendicular to the direction of travel. In other words, the X axis represents the vehicle's location on the roadway with respect to a reference starting point (i.e., center), and the Y axis estimates the current lane of the vehicle. The pixel size, s, on the ground depends on the flight elevation and is the key to the mapping function between the two coordinate systems. Besides, keeping the axes of the two coordinate systems parallel makes the transformation between them simple. The front bumper is taken as the location of the vehicle on the roadway, and the trajectory of the vehicle is the list of its location over space and time (x, y, t).

Kalman Filter

The output of image stabilization and vehicle detection could be noisy. As a result, a Kalman filter is applied to reduce the noise in the state estimation of the vehicles. The vehicle state at each point, $x_i^{t_i}$, is characterized by its location and kinematic state. The state attributes include the position information, $p_i^{t_i}$, speed, $v_i^{t_i}$, and acceleration, $a_i^{t_i}$. The expected state of the vehicle after t seconds (i.e., rate of data generation), $\hat{x}_i^{t_i+t}$, can be estimated by multiplying the transition matrix A by the initial state vector.

$$\hat{x}_i^{t_i+t} = Ax_i^{t_i} = \begin{bmatrix} 1 & t & \frac{t^2}{2} \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix} [p_i^{t_i}, v_i^{t_i}, a_i^{t_i}]^T \tag{3.5}$$

In the state estimation process, the Kalman filter is usually applied to estimate the best guess on the current state of the vehicle considering the previous state and current measurements (i.e., from aerial images). Previous vehicle state, $x_i^{t_i-t}$ is transitioned to the expected current state, x_i^t, based on the process model and applying the transition matrix A:

$$\text{Process model: } x_i^{t_i} = A\hat{x}_i^{t_i-t} + \omega \tag{3.6}$$

where ω is the process noise. In this dissertation, the process model considers $\omega = [t^2, t, 1]^T \sigma_{ap}^2$, where σ_{ap}^2 is the acceleration variance. ω is assumed to be normally distributed with covariance matrix of Q:

$$Q = \begin{bmatrix} \frac{t^4}{4} & \frac{t^3}{2} & \frac{t^2}{2} \\ \frac{t^3}{2} & t^2 & t \\ \frac{t^2}{2} & t & 1 \end{bmatrix} \sigma_{ap}^4 \tag{3.7}$$

The expected current state is converted to the expected measurement, $z_i^{t_i}$, through the following measurement model:

$$\text{Measurement model: } \hat{z}_i^{t_i} = Hx_i^{t_i} + \nu \tag{3.8}$$

where ν is the measurement noise. Since only the position of the vehicles is directly measured from the aerial images, the resulting state to measurement conversion matrix, H, is [1,0,0]. ν is assumed to be normally distributed with covariance matrix of R:

$$R = \begin{bmatrix} \sigma_p^2 \end{bmatrix} \tag{3.9}$$

where σ_p^2 is the position variance. Note that the measurement covariance matrix considers the variance in position alone. A 2D Cartesian coordinate system is considered for the measurements, and the state of the vehicle is evaluated along the two axes, x and y, separately. Taking σ_{ap}^2 and σ_p^2, equal to $0.5(\frac{m}{s^2})^2$ and $0.5m^2$, respectively, performed well in addressing the noise in the state estimates.These values are carefully selected using a grid search method and choosing the values that result in smooth trajectories and not in losing recent measurements.

3.3.2 Adaptive Cruise Control (ACC) Operated Vehicles

One of the primary motivations of this data collection was to observe how recent advancements in vehicle technology and ADAS impacts traffic flow dynamics. This data collection focuses on the impacts of ACC as a core feature amongst all automated vehicles. Unfortunately, it is not possible to determine if a vehicle is using ACC from the birds-eye view without additional information. A potential solution to deal with this problem is to use probe vehicles during the data collection. Accordingly, a platoon of three probe vehicles, including two Toyota Prius and one Toyota Avalon, was used under ACC for data collection. Note that it is expected to see different mechanical performances among vehicles with different makes and models. However, Makridis et al. [58] studied 27 different models of vehicles and found that the ACC vehicles behave much more homogeneously even among different manufacturers.

The leader of the platoon was following an arbitrary vehicle on the roadway in front of it using ACC. The other two vehicles were also following their leaders with ACC. A total of five runs were performed along the study roadway segment. Note that all vehicles had full-range ACC with stop-and-go capability. The collected vehicle trajectories include the trajectory of the probe vehicles using ACC and the trajectory of other vehicles, some of which could be using ACC technologies.

3.3.3 Location and Procedure

The data is collected on the southbound of Interstate Highway 35 between Exit 237B and Exit 238A in Austin, Texas (see Figure 3.4). A single stretch of nearly 500 feet roadway was recorded for 2 hours between 07:30 AM and 09:30 AM on a Friday. This segment has four lanes in the southbound direction. Note that this highway has directional traffic with congested southbound in the morning and congested northbound in the afternoon. The traffic video is recorded with 4K resolution at 25 frames per second and flight elevation of 400 feet. Flying at a higher altitude could help collect data from a more extended study segment and longer trajectory data.

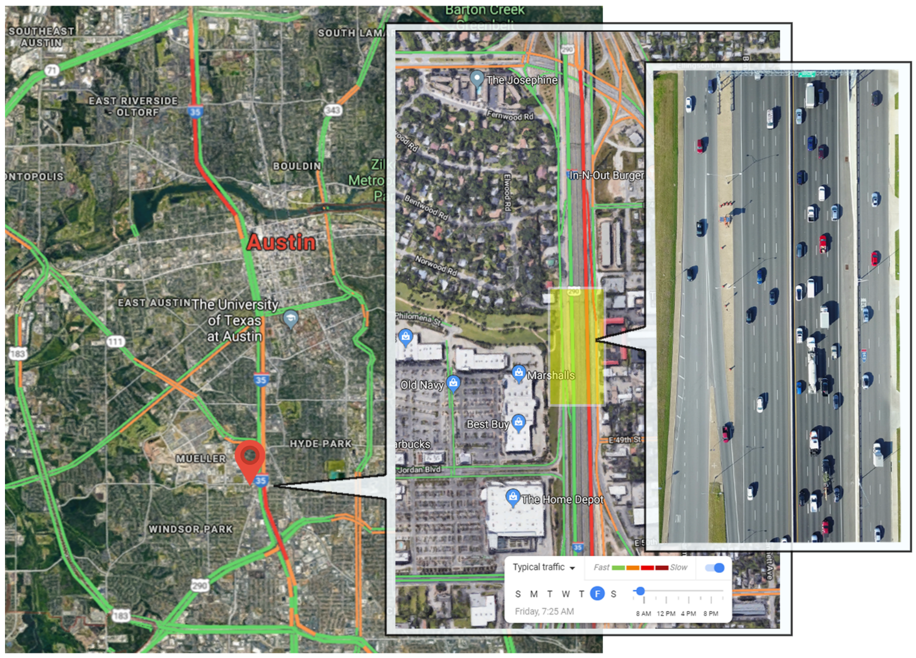


Figure 3.4: Vehicle trajectory data collection location.

3.3.4 Complete Vehicle Trajectory Data

One of the main features of this dataset is the continuity in data recording for over two hours during the morning traffic peak hours. The continuity in data collection ensures that no information or interaction among the vehicles is lost. A vehicle trajectory is recorded from the first time the vehicle was seen on the study segments and every 0.08 seconds. For most of the vehicles, the starting point of the trajectory would be at the beginning of the study segment. However, at the start of the recording, some of the vehicles were further downstream of the study segment, and the trajectory of those vehicles starts from where they were seen first. Figure 3.5 presents the examples of the time-space diagram of the trajectories extracted using the methodology of this dissertation on a roadway segment of nearly 500 feet over 10 minutes. Note that some of the trajectories are not continuous in this figure due to lane-changing maneuvers in most cases. However, there are few cases that the vehicle detector (RetinaNet) has failed to detect a vehicle in the image, causing a discontinuity in its trajectory. In this dissertation, two actions are applied to address these types of errors. First, the false-negative error in detection is mitigated in the tracking process by combining a high frame rate (i.e., 25 frames per second) and maintaining track of the vehicle for five consecutive frames after the last time it was seen. Second, the false-positive error is reduced by eliminating the trajectories with less than three data points. The trajectories with less than three data points account for less than 0.02 percent of the total

Table 3.3: Basic statistics on travel-time.

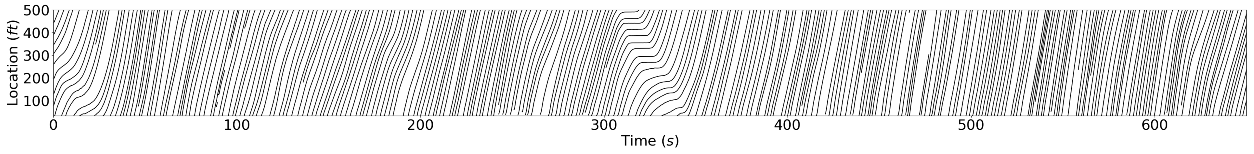
Travel-time	
Segment length (middle part of the study segment)	400 feet
Minimum	5.29 s
Maximum	54.56 s
Median	14.42 s
Mean	13.44 s

data points. In future trajectory data collection studies, it would be a good practice also to record vehicles' trajectories using onboard sensors in the probe vehicles (e.g., GPS and IMU sensors) to compare with the collected trajectories using the proposed methodology.

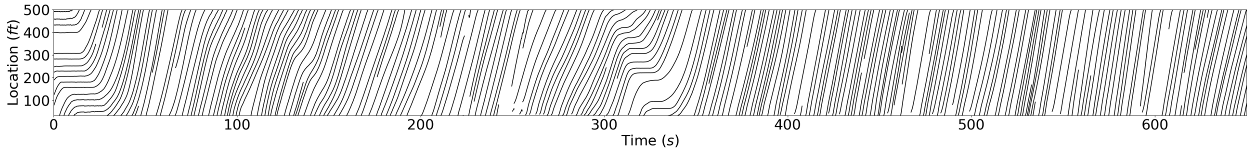
Travel-time, flow, and density can be directly extracted from the vehicle trajectory data. Figure 3.6 presents the distribution of the travel time of all the vehicles over the study segment and for the whole study duration. The travel time is estimated for the middle 400 feet part of the study segment (locations 50 feet to 450 feet). This segment selection is due to the partial observation of the large vehicles at the edges of the images, which results in inaccurate bumper location estimation at those edges. Basic statistics on the collected travel-time are presented in Table 3.3. Flow and density plots with an aggregation level of 30 seconds for individual lanes as well as their average are presented in Figure 3.7. According to these plots, the traffic dynamics of the leftmost lanes (Figures 3.7a, 3.7b) are different from the traffic dynamics of the rightmost lanes (Figures 3.7c, 3.7d). The flow and density data points for lanes one and two (leftmost lanes) are more in the congested region compared to the data points for lanes three and four (rightmost lanes).

ACC Platoon Data

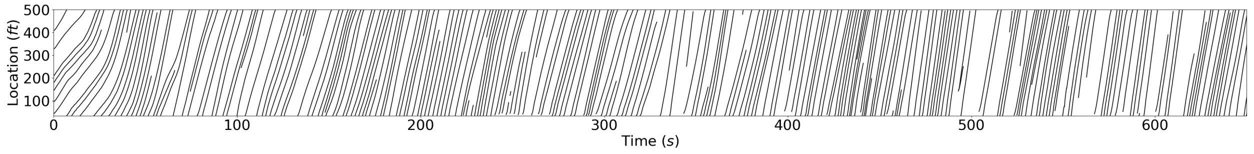
This dataset introduces one of the first collected comprehensive trajectory datasets from both ACC and human-driven vehicles. Five runs of the platoon of probe vehicles using ACC are recorded during the data collection. The first three runs are conducted in the rightmost lane (lane 4), and the last two runs are performed in the second rightmost lane (lane 3). Figures 3.8 and 3.9 illustrate the overview of the platoons and the traffic dynamics for each of the five runs. The platoon overview, Figure 3.8, presents the identification number of the ACC vehicles in the platoon, as well as the leader of the first ACC vehicle and three human-driven vehicles behind the last ACC vehicle. The identification numbers are arbitrary and unique numbers assigned to each of the vehicle trajectories in the dataset. The time-space diagrams of Figure 3.9 are generated for the period of 30 seconds before the ACC platoons



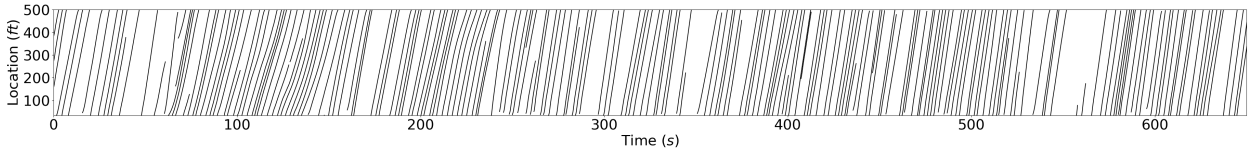
(a) Lane 1



(b) Lane 2



(c) Lane 3



(d) Lane 4

Figure 3.5: Time-space diagram of the vehicles for ten minutes on the Southbound of I-35 between Exit 237B and 238A, Austin, TX, during the morning peak time.

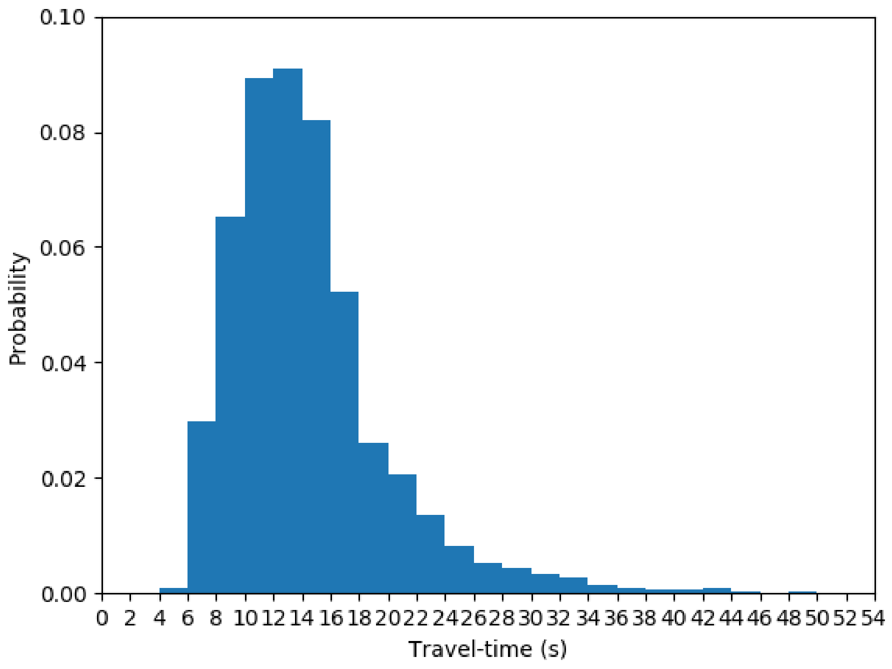
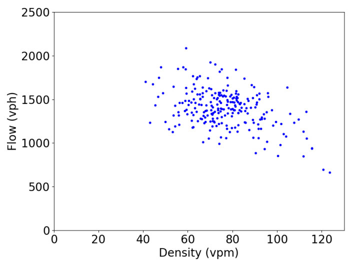
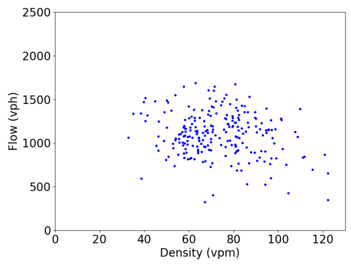


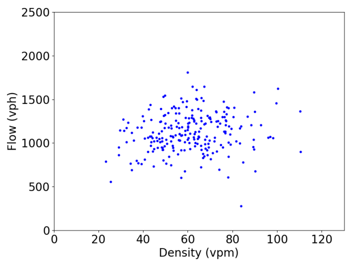
Figure 3.6: Travel time distribution.



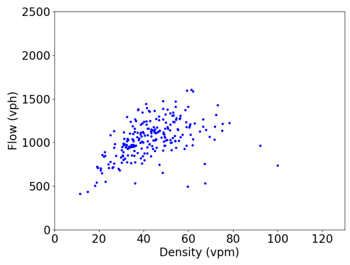
(a) Lane 1



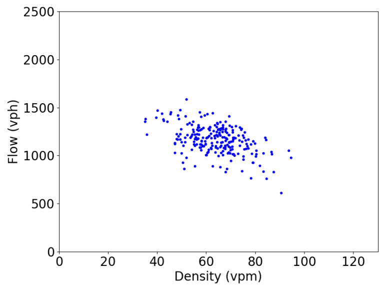
(b) Lane 2



(c) Lane 3



(d) Lane 4



(e) Average of all lanes

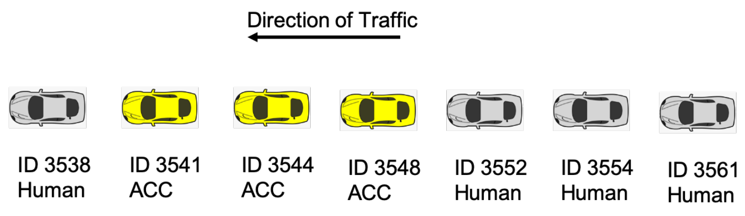
Figure 3.7: Flow and density plots for each lane with aggregation resolution of 30 seconds.

enter the segment and up to 30 seconds after exiting the study segment. The trajectories of the ACC vehicles are depicted with blue lines in the time-space diagrams.

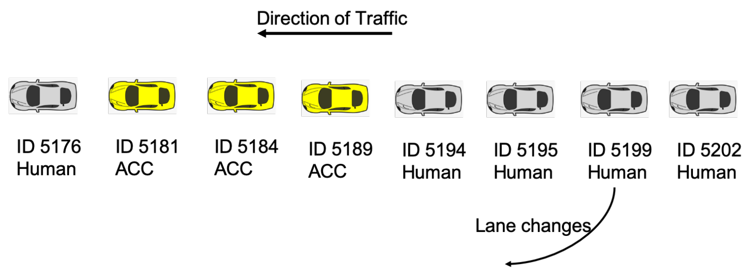
One of the constraints of this field-based vehicle trajectory data is that the length of the study segment is limited to 500 feet. This limitation is due to the relatively low flight elevation (400 feet). In addition, the small study segment results in collecting trajectories with limited time duration (Figure 3.6). As a result, this real-world trajectory dataset is only used for the training and evaluation of the traffic state prediction model proposed in Chapter 6. Accordingly, to complement this dataset, the FHWA Next Generation Simulation Models (NGSIM) dataset is utilized to evaluate all the traffic state prediction models proposed in this dissertation.

3.3.5 FHWA Next Generation Simulation Models (NGSIM) Trajectory Data

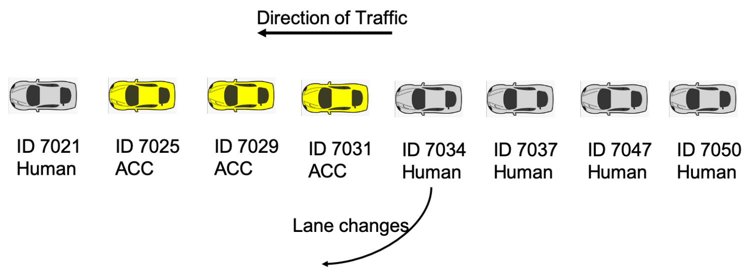
One of the commonly adopted field-based vehicle trajectory datasets is the FHWA Next generation Simulation Models (NGSIM) [51]. NGSIM is a well-known open-source trajectory dataset collected in 2006 using digital cameras at different locations, including US Highway 101 and Interstate 80 freeway. The vehicle trajectories are extracted from the images of multiple cameras combined to create a single image that looks like an aerial shot. The NGSIM trajectory data contains the location of each vehicle at a frequency of 10 Hz over a 1600 to 3200 feet stretch of roadway. The NGSIM dataset contains three sets of 15 minutes trajectory data for each of the US Highway 101 and Interstate 80 freeway. This dataset is used to evaluate all the proposed traffic prediction models developed in this dissertation.



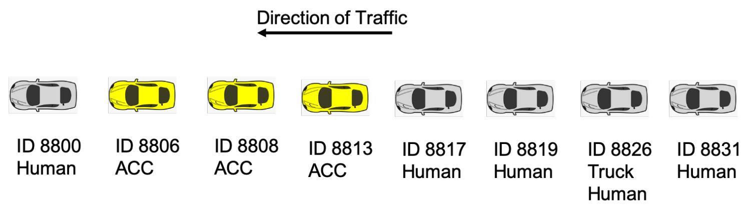
(a) First run



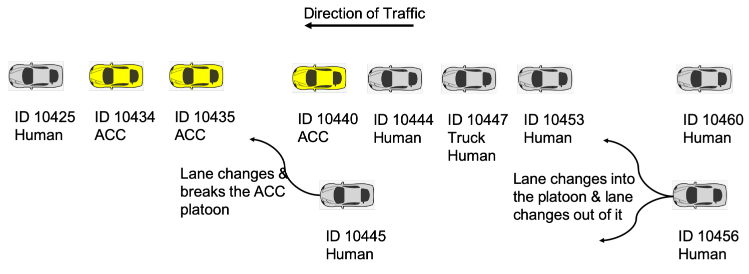
(b) Second run



(c) Third run

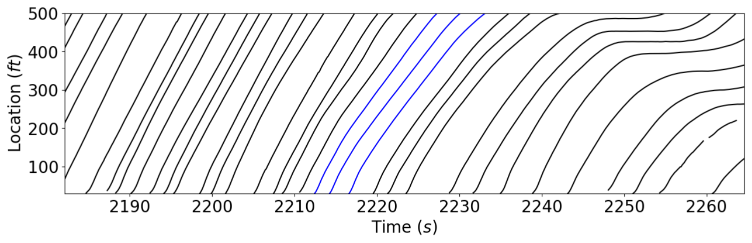


(d) Fourth run

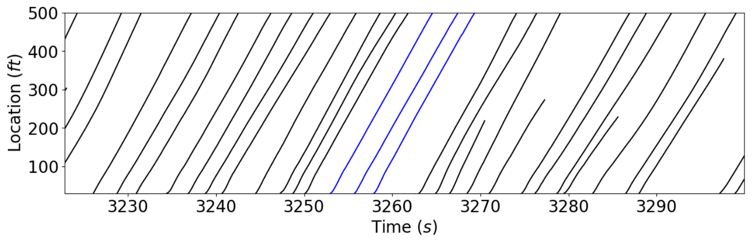


(e) Fifth run

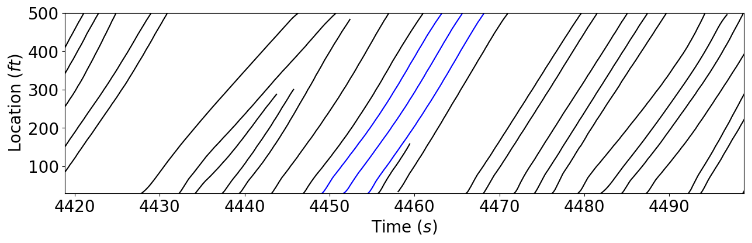
Figure 3.8: Overview of the platoon of the probe vehicles over five runs of data collection.



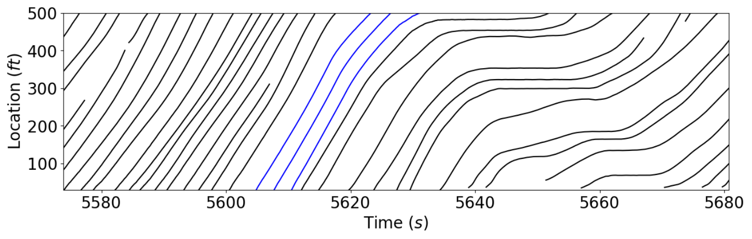
(a) First run



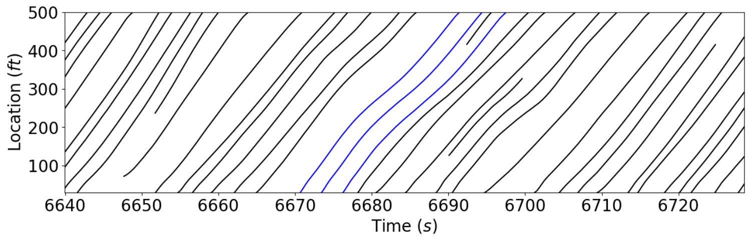
(b) Second run



(c) Third run



(d) Fourth run



(e) Fifth run

Figure 3.9: Time-space diagram of the probe vehicles over five runs of data collection.

CHAPTER 4

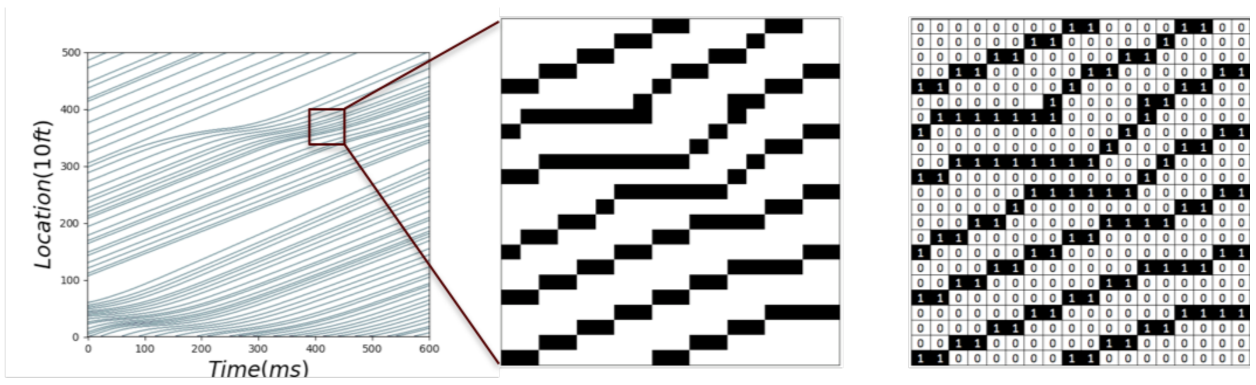
TRAFFIC STATE PREDICTION USING TIME-SPACE DIAGRAM

4.1 Introduction

Driving behavior and vehicle interactions are best captured by the time-space diagram of the traffic stream (see Figure 4.1 a). This diagram provides detailed information on both the microscopic and macroscopic characteristics of the traffic flow. Despite the time-space diagram being at the core of many traffic flow theory innovations, there was no scalable method for generating these diagrams based on real-world data until recently. As a result of recent advancements in connected vehicles technology, a new data source at the individual vehicle level has become available. As a result, the time-space diagram of the vehicles can be derived from the data shared by the connected vehicles.

A primary contribution of this chapter is to introduce a traffic prediction methodology based on convolutional neural networks (CNN), which can directly utilize the time-space diagram in the prediction. This methodology presents three major advantages: (1) it uses the time-space diagram directly as the input without the need for any aggregation or abstraction; (2) it is based on a learning process that identifies the critical features of the time-space diagram required to produce an accurate prediction. These features include traffic flow dynamics and vehicle interactions that can impact the traffic state in the future (e.g., shockwave formation and its propagation speed); and (3) this methodology offers a solution to the transferability issue of non-parametric models, i.e., these models, in general, provide location-specific solutions and they should be re-calibrated for another location.

The remainder of this chapter is organized as follows: the next section presents the details of the proposed methodology to predict the traffic state using the time-space diagram. This section is followed by a discussion on model configuration and its training process. The presented model is then evaluated against other common non-parametric methods (i.e., multilayer perceptron, support vector regression, and Autoregressive integrated moving average), and a discussion on the findings is presented. Finally, concluding remarks close the chapter.



a) Time-space diagram b) Time-space bins c) Time-space matrix

Figure 4.1: Time-space diagram.

4.2 Methodology

In general, traffic prediction models encompass a function that takes traffic-related input variables and potentially adjustable parameters to predict the traffic state in the future. This chapter adopts a CNN as the prediction function. The CNN takes the time-space diagram of segment x over the period of $(t - \Delta t, t)$ and predicts its average traffic state for the next period of $(t, t + \Delta t)$. A CNN is a representative learning approach that learns the key features in the time-space diagram without user input.

4.2.1 Time-space diagram

The time-space diagram is the plot of the trajectory of vehicles over time and space. This dissertation considers a connected vehicle environment, in which the vehicles share their state through wireless communications. It is assumed that each vehicle shares a basic safety message (BSM) transmitted every 100 milliseconds that includes the vehicle's speed, location, and heading. This chapter assumes that the vehicle's position is accurate up to 10 feet. In such a connected environment, we can divide the time and space domains into cells of 100 milliseconds and 10 feet, respectively. The trajectory of a vehicle is constructed based on the vehicle's presence in the time and space cells, similar to the analogy of image pixels (Figure 4.1 b). The time-space diagram can be stored in a matrix with rows for the space dimension and columns for the time dimension (Figure 4.1 c). In this matrix, a cell value of one indicates the presence of a vehicle at the respective time and location, and a value of zero is used for empty cells. Through this process, the time-space diagram is converted into a binary matrix that can be processed by CNN.

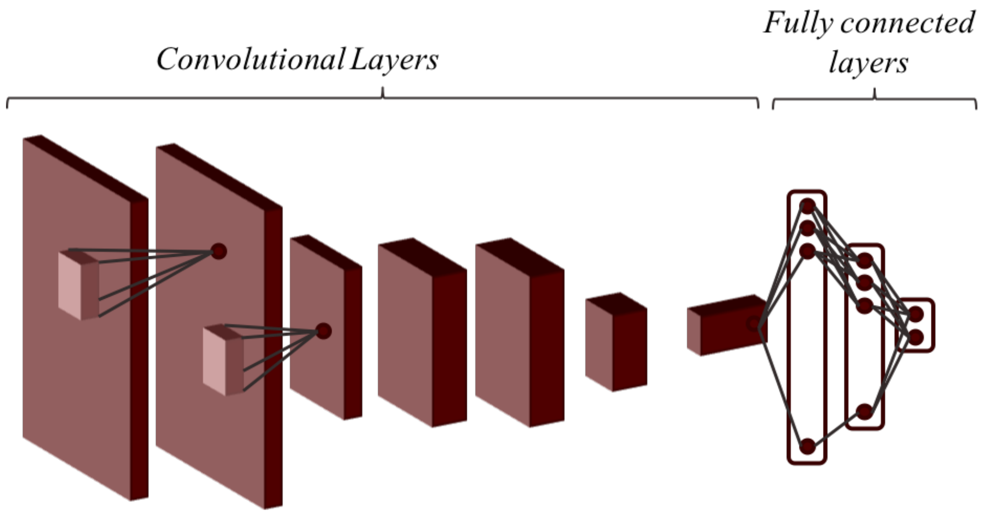


Figure 4.2: Convolutional neural network (CNN).

4.2.2 Convolutional Neural Network, CNN

CNN is a type of feed-forward neural network extensively used in image recognition. The structure of the CNN is inspired by the visual cortex [85] and encompasses multiple convolutional layers followed by fully connected layers (Figure 4.2). When processing an image, the shallower convolutional layers of the network detect simple features such as straight edges and curves, and the deeper layers identify more complex elements such as shapes and patterns. Each convolution layer uses a receptive window with a fixed size (width, height, and depth) that slides over the neurons of the previous layer. Each convolutional layer is a stack of parallel channels, and each channel is a 2D layer of neurons that share the same filter size, weights, and biases. Each neuron sums its weighted inputs and passes them through an activation function. The receptive window provides the opportunity to include the spatial correlation among the nearby units that could construct a feature of interest. Moreover, sliding the same filter over the input of the convolution layer ensures a feature identification capability independent of their location [86]. This location invariant property of the convolution layer is one of the main reasons for choosing CNNs to interpret the time-space diagram of the vehicles, as the abnormalities (e.g., shockwaves) in the traffic stream could happen at any location and time on the time-space diagram. A detailed discussion on the CNN design is provided in the next section.

4.2.3 Simulated Data

Deep learning approaches require large datasets to calibrate their significant number of parameters, including the weights and biases. The data used to train the proposed traffic prediction model is collected using the microscopic traffic simulator that was introduced in section 3.2. The simulator uses the Intelligent Driver Model (IDM) [59]. Each simulation run considers a single-lane road segment with a length of 40000 feet over 15 minutes. The simulation starts by initializing random and unique IDM parameters for the vehicles in the simulation. The IDM parameters initialization is based on the ranges and distributions indicated in Table 4.1. To create a comprehensive training dataset, we need to include different traffic states between free-flow and fully congested. The simulation uses two types of disturbances to create different traffic states. The first type of disturbance is the speed drop perturbation that forces a random vehicle to decelerate for a small period (e.g., 15 seconds). The speed drop results in a small shockwave in the traffic stream. The second type of disturbance is a slow-moving vehicle that forces a random vehicle to reduce its speed to a portion of the desired speed and to maintain it for an extended period (e.g., 5 minutes). The slow-moving vehicle creates a slow-moving shockwave resulting in high congestion and traffic breakdowns. Both of these disturbances occur at a random location, time, duration, and intensity controlled by the parameters provided in Table 4.1. Moreover, the simulation estimates the traffic state in terms of density, $K(A)$, and flow, $Q(A)$, for segments of 200 feet over 20 seconds based on Edie's definition [87].

$$Q(A) = \frac{d(A)}{|A|} \tag{4.1}$$

$$K(A) = \frac{t(A)}{|A|} \tag{4.2}$$

where $|A|$ is the area of the time-space block A (e.g., 200 feet by 20 seconds). $d(A)$ and $t(A)$ denote the total distance traveled and total time spent by all the vehicles going through block A, respectively. The simulation returns a time-space diagram, a density matrix, and a flow matrix for the whole analysis period. Each row in the flow and density matrices contains all the traffic states of a 200 feet road segment over the simulation period. And each column contains the traffic states of all the 200 feet road segments over a specific 20 seconds of simulation.

Table 4.1: Simulator parameters.

IDM Parameters				
Parameter	Minimum	Maximum	Distribution	Unit
free acceleration exponent δ	4		constant	
jam distance s_0	6.56		constant	ft
maximum acceleration a	2.5	3.5	uniform	ft/s^2
desired deceleration b	4.0	8.0	uniform	ft/s^2
desired time gap T	1.0	2.0	uniform	s

Constant Parameters		
Parameter	Value	Unit
lanes	1	#
segment length	40000	ft
simulation time	900	s
vehicles	850	#

Random Parameters				
Parameter	Mean	Standard Deviation	Distribution	Unit
initial gap	100	30	half normal	ft
deceleration wave duration	15	5.0	normal	s
Parameter	Minimum	Maximum	Distribution	Unit
number of deceleration waves	60	100	discrete uniform	#
intensity of deceleration waves	1.5	2.5	uniform	ft/s^2
number of slow-moving vehicles	3	7	discrete uniform	#
speed of slow-moving vehicles	$0.05\times V_0$	$0.70\times V_0$	uniform	ft/s
slow-moving duration	300	500	uniform	s

4.3 Model Configuration and Training

This chapter configures and trains a CNN to predict the traffic state of a roadway segment with a length of 2000 feet over the next 20 seconds by taking the time-space diagram of the segment in the previous 20 seconds. Deep learning methods are subject to overfitting due to their large number of parameters and model complexity. Using a large and comprehensive dataset is one of the regularization [88] approaches to prevent overfitting. Accordingly, the data simulator discussed in the previous section is used to collect more than 1.4 million data points for training and evaluation of the CNN model.

4.3.1 Training Data

The traffic is simulated for a single-lane roadway with a length of 40000 feet and a simulation period of 15 minutes with the desired speed of 30 miles per hour. The CNN model in this chapter predicts the traffic state of a segment for the next 20 seconds based on its time-space diagram in the past 20 seconds. Each run of the traffic simulator can be divided into multiple data points for smaller segments. Each data point consists of the time-space diagram of a 2000 feet long segment over 20 seconds as the input to the model and its average flow and density in the next 20 seconds as the expected outputs of the traffic prediction model. Each simulation run can be divided into $40000/2000 \times (900-20)/20 = 880$ data points. The data is collected from 1600 runs of simulation resulting in $1600 \times 880 = 1,408,000$ data points. The simulation runs are divided into sets of 1000, 300, and 300 for training, validation, and testing purposes.

4.3.2 Loss Function

Predicting traffic state in terms of flow and density using a deep learning approach can be treated as a regression problem. As a result, the mean squared error (MSE) is an appropriate loss function to train the proposed CNN-based traffic prediction model (Figure 4.3). Moreover, MSE, mean absolute error (MAE) and mean absolute percentage error (MAPE) are used as the performance measures to compare different models.

$$Loss = MSE = \frac{1}{n}\sum_{i=1}^{n}[(\hat{Q}_i - Q_i)^2 + (\hat{K}_i - K_i)^2] \tag{4.3}$$

$$MSE_Q = \frac{1}{n}\sum_{i=1}^{n}(\hat{Q}_i - Q_i)^2, MSE_K = \frac{1}{n}\sum_{i=1}^{n}(\hat{K}_i - K_i)^2 \tag{4.4}$$

$$MAE_Q = \frac{1}{n}\sum_{i=1}^{n}|\hat{Q}_i - Q_i|, MAE_K = \frac{1}{n}\sum_{i=1}^{n}|\hat{K}_i - K_i| \tag{4.5}$$

$$MAPE_Q = \frac{100}{n}\sum_{i=1}^{n}\frac{|\hat{Q}_i - Q_i|}{Q_i}, MAPE_k = \frac{100}{n}\sum_{i=1}^{n}\frac{|\hat{K}_i - K_i|}{K_i} \tag{4.6}$$

It should be noted that flow and density have different ranges, and it is required to normalize them before training. Normalization is the process of subtracting each point by mean value and dividing it by the standard deviation. The flow (Q_i) and density (K_i) in Equation 4.3 should be normalized using the mean and standard deviation of flow and density in the training dataset. Normalization of the outputs prevents optimization toward the larger scale also leads to faster convergence in training [89].

4.3.3 Training

Training is the process of minimizing the loss function through an iterative process of changing model parameters. The training is performed in multiple steps, and at each step, a batch (i.e., subset) of the training data points is used to estimate the loss. At each step of training, the model parameters are updated based on their loss gradients times a learning rate to gradually move toward the local minimum of the loss function. We use Adam optimizer developed by Kingma et al. [90] to train our CNN model. This optimizer is among the top choices in the training of deep learning models [91]. Adam optimizer is a type of stochastic gradient-based optimizer that maintains individual adaptive learning rate for the model parameters [90].

4.3.4 CNN Design

Figure 4.3 presents the CNN structure designed in this chapter for interpreting the time-space diagram of a 2000 feet segment of roadway over 20 seconds to predict its traffic state in the next 20 seconds. The network input is in the form of a time-space matrix of trajectories. The proposed structure is motivated by the work of [92] that uses deep convolutional layers with small filter (i.e., receptive window) sizes such as 3×3 and 5×5. A stack of smaller filter size convolutional layers can provide the same overall receptive window as one layer with a larger filter size. The advantage of using a stack of smaller filter sizes (e.g., 3×3) is that we introduce more nonlinearity to the model and also reduce the number of parameters in the model. The final CNN structure (Figure 4.3) is the result of the search process by gradually increasing the depth of the network and number of channels and choosing the

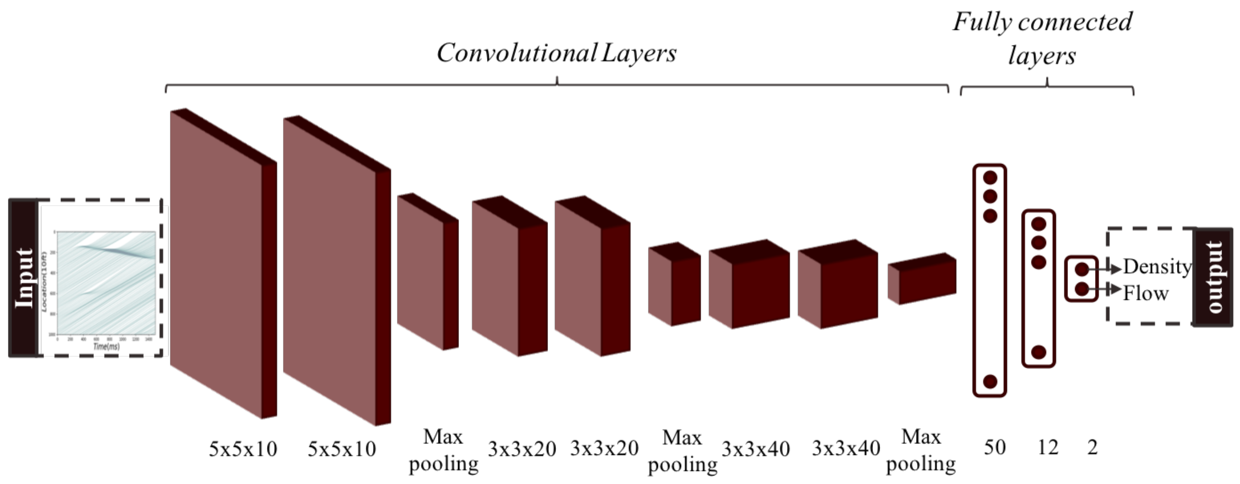


Figure 4.3: CNN for traffic prediction.

structure with the best performance on the validation dataset after five training epochs (a complete pass through the entire training set). Referring to Figure 4.3, the first stack of the convolutional layers uses a filter size of 5×5 and ten channels ($5 \times 5 \times 10$) with a stride number of 3 to reduce the size of the feature map and ultimately the number of parameters in the model. Stride implies the number of units of movements when sliding the receptive window on the previous layer of the network. A small filter size of 3×3 and a stride of 1 is used in the remaining convolutional layers. Moreover, the max pooling with a filter size of 2×2 and stride number of 2 is used after each pair of convolutional layers. Pooling is the process of summarizing the features in the filter size regions by taking their maximum or average to reduce the size of the feature space through the network. This chapter adopts the max pooling due to its fast convergence performance [93] . The last pooling layer is followed by two fully connected layers and the output layer. Rectified linear unit (Relu) activation function is used for all the neurons in the convolutional layers and the first two fully connected layers. Relu function returns the maximum of zero or its input value that performs well in the training process for deep neural networks [94]. The output layer uses a linear activation function and returns the predicted flow and density values.

4.3.5 Performance on Test Dataset

The training process is performed over multiple passes through the training set. A batch size of 880 is used in combination with early stopping to prevent overfitting the training dataset. At the end of every epoch, the loss function for the validation dataset is estimated. The training is terminated after five epochs from the epoch with the minimum validation loss. Nineteen epochs of training resulted in the minimum loss (normalized MSE = 0.087) on the validation set. The trained model is used to predict the traffic flow and the density of the test set. Since the model is trained to predict normalized values of flow and density, the predictions are transformed using the mean and standard deviation of flow and density of the training dataset. Table 4.2 shows the performance of the model in predicting flow and density in terms of MSE, MAE, and MAPE.

4.4 Comparison and Evaluation

Three non-parametric models, i.e., support vector regression (SVR), multilayer perceptron (MLP), and Autoregressive Integrated Moving Average (ARIMA), are used to assess the effectiveness of the proposed CNN approach. The first two models fall in the regression-based prediction models, while ARIMA is a time series based prediction model. Different variations of these models have been used in the literature for short-term traffic prediction. For comparison, these models are also trained to predict the traffic state of the roadway segment with the length of 2000 feet over 20 seconds using the same dataset used for the CNN. The SVR and MLP take the average traffic flow and density of the segment for the past 20 seconds as the input and predict the average traffic state of the segment in the next 20 seconds as the output. The ARIMA model takes the time series of the past traffic state of the segment as input and predicts its traffic state for the next time step.

4.4.1 Multilayer Perceptron (MLP)

MLP is a class of fully-connected neural networks with few layers between input and output layers. For the comparison, two separate models are trained for flow and density prediction. The structure used for the MLP models is a three-layer network which includes the input layer, one hidden layer, and the output layer. The input layer consists of two nodes for the segment's current average flow and density, and the output layer is a single neuron for flow or density. The Relu activation function is used for the neurons of the hidden layer. The number of neurons on the hidden layer is optimized by searching between two and twenty and selecting the number of neurons that produces the best performance on the validation dataset. Normalized flow and density values are used for training the MLP models. Fifteen and fourteen neurons on the hidden layer resulted in the minimum loss (MSE) for the flow and density models, respectively. The result of the performance of the MLP-based flow and density models are presented in Table 4.2.

4.4.2 Support Vector Regression (SVR)

SVR is a branch of support vector machine (SVM) algorithm in which the feature map is transformed to a higher dimensional feature space to find a hyperplane that fits the data linearly. We use a kernel-based SVR with radial basis function (RBF) to predict traffic flow and density. The support vector regression can easily overfit the training data, and ϵ, C, and γ are the three regularization parameters for RBF-based SVR to control the overfitting. ϵ is the size of the error margin for which no penalty is considered, and a lower ϵ results in a more complex fit. C controls the penalty for the points that fall beyond the ϵ margin and, consequently, the function's complexity. γ is a parameter of the RBF kernel that controls the radius of influence of each training point.

Similar to the MLP approach, two separate SVR models for flow and density prediction are trained using the normalized flow and density. The differential evolution optimization algorithm [95] is used to search for the regulatory parameters that result in a minimum loss (MSE) on the validation dataset. The search boundaries of $(0.01, 0.5)$, $(0.1, 10)$ and $(0.1, 5)$ are selected for the ϵ, C and γ, respectively. Since the computational complexity of the SVM is n^3 [96], n being the number of training points, the search for the regulatory parameters is performed on a subset of 10000 randomly selected data points from the training dataset. The selected values for the [ϵ, C, γ] are [0.20, 2.34, 0.11] for the SVR flow prediction model, and [0.18, 0.99, 0.15] for the SVR density prediction model. After selecting the regulatory parameters, ten subsets of 10000 training data points are randomly selected. A flow and a density model are fitted to each training subset. The performances of the models are evaluated using the complete test dataset, and the 95 percent confidence interval of the prediction performance of the flow and density models are reported in Table 4.2.

4.4.3 Autoregressive Integrated Moving Average (ARIMA)

Autoregressive integrated moving average (ARIMA) is a prediction model for stationary time series. A stationary time series has a constant mean, variance, and autocorrelation of the series over time. Differencing is an approach to generate stationary series from non-stationary ones and refers to recreating a new time series from the differences of each observation with the one in the previous d time steps. The general ARIMA prediction equation is a linear combination of the previous (lags) observations and the previous errors in prediction. The ARIMA model is defined by three parameters of p, d, and q. p is the number of lags of observation, q is the number of lags of previous errors used in the prediction equation, and d denotes the number of differences needed to make the time series stationery.

Two separate ARIMA models are considered to predict the traffic flow and density of a segment of 2000 feet long for the next 20 seconds. The ARIMA model for flow (density) prediction takes the time series of the average flow (density) of the segment with time step resolutions of 20 seconds as input and predicts the flow (density) for the next 20 seconds. The data collected from the simulation is used to create 15 minutes long time series at 20 seconds resolution. The first 10 minutes of each series is used to train the model, and the last 5 minutes is used to evaluate its accuracy. Since different combinations of (p, d, q) can create different classes of the ARIMA model, a grid search is performed to find the best combination of ($p \in [0, 5]$, $d \in [0, 3]$, $q \in [0, 3]$) that results in the minimum MSE. A sample of 1000 time series of flow and density randomly selected from the training dataset is used in the grid search. The best performing ARIMA model concerning MSE turned out to be ($p = 1, d = 1, q = 0$) for both the flow and density prediction models. The following equation is the general form of ARIMA$(1, 1, 0)$ prediction equation,

$$\Delta x_{t+1} = \alpha \Delta x_t + \beta \tag{4.7}$$

$$\hat{x_{t+1}} = x_t + \alpha(x_t - x_{t-1}) + \beta \tag{4.8}$$

where x represents either flow or density. Since d is equal to 1, the prediction equation is for the time series of consecutive differences. Equation 4.7 indicates that the future change of state Δx_{t+1} is proportional to the change from the previous state Δx_t plus a constant β. This equation is extended to Equation 4.8 for prediction of the future state $\hat{x_{t+1}}$ based on the current state x_t. For every 15 minutes, long time series in the test dataset, an ARIMA$(1, 1, 0)$ is trained on its first 10 minutes, and its performance for prediction of the next 5 minutes is estimated. The performance of ARIMA$(1, 1, 0)$ models for flow and density on the test dataset are presented in Table 4.2.

4.4.4 Performance Comparison

Test Dataset

According to Table 4.2, the proposed CNN model performed better than the other three non-parametric models in the prediction of flow and density on the test dataset. The performance of the MLP and SVR are very similar, which is interesting as they are both provided with the same input features, including the flow and density of the segment. The average errors of the ARIMA model are slightly higher than the MLP and SVR for both flow and density prediction of the test dataset. Figure 4.4 presents the prediction performance of the CNN model and the other models over different traffic states on the fundamental diagram. In this figure, the color of the points is correlated with the error of the model in the prediction of the future traffic state. The color changes from light yellow to dark blue proportional to the summation of the squared error of normalized flow and density predictions. A darker color (e.g., blue) indicates a higher prediction error, while a lighter color (e.g., yellow) shows a lower prediction error. As demonstrated in Figure 4.4a, the error in predictions of the CNN model is consistently low over the fundamental diagram even at points with low flow and moderate density caused by random shockwaves. On the other hand, Figures 4.4b and 4.4c display good performance for the MLP and SVR at some part of the free flow portion of the fundamental diagram but poor performance at high and low flow levels. Figure 4.4d presents the performance of the ARIMA model in the last 5 minutes of every time series in the test dataset that has a lower number of test data points compared to the other three models. Even though the average performance of the ARIMA model is slightly worse than the MPR and SVR but considering Figure 4.4d, the ARIMA model performed more consistently than the MPR and SVR on the free flow part of the fundamental diagram. However, ARIMA performed poorly in other low flow levels with moderate density. This observation suggests that the ARIMA$(1, 1, 0)$ model is inadequate for transient abnormalities in traffic caused by random shockwaves.

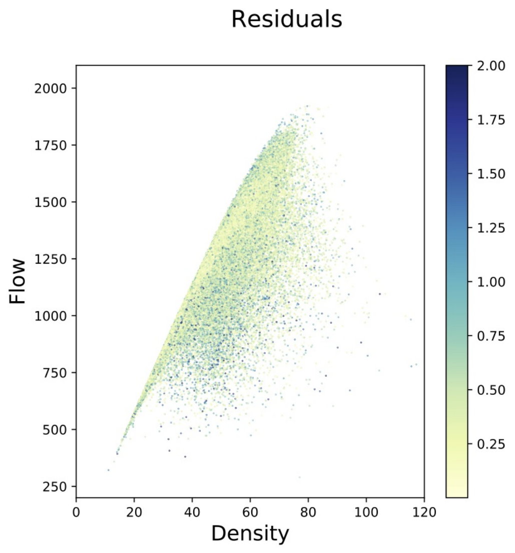
NGSIM dataset

The FHWA Next Generation Simulation (NGSIM) US-101 dataset [97] is also used to evaluate the proposed CNN model. This dataset covers three 15 minutes periods during the morning peak. The time-space diagram for each lane is recreated from the trajectory data for the middle 2000 feet of the segment, and the average flow and density are estimated for every 20 seconds using Equations (4.1) and (4.2). The NGSIM data sums up to $3 \times 5 \times (900 - 20)/20 = 660$ data points and is directly used to test the previously trained CNN, MLP and SVR models. The flow and density for every 20 seconds and every lane in the NGSIM dataset sums up to $3 \times 5 = 15$ time series for both flow and density. For each time series, an ARIMA$(1, 1, 0)$ is trained using its first 10 minutes, and the performance of the ARIMA model is evaluated over the last 5 minutes of the time series. The performance of the CNN, MLP, SVR, and ARIMA models are reported in Table 4.2. According to Table 4.2, the ARIMA model performed better than the other three models on the NGSIM dataset. Part of this performance is due to training the ARIMA models on the first 10 minutes of

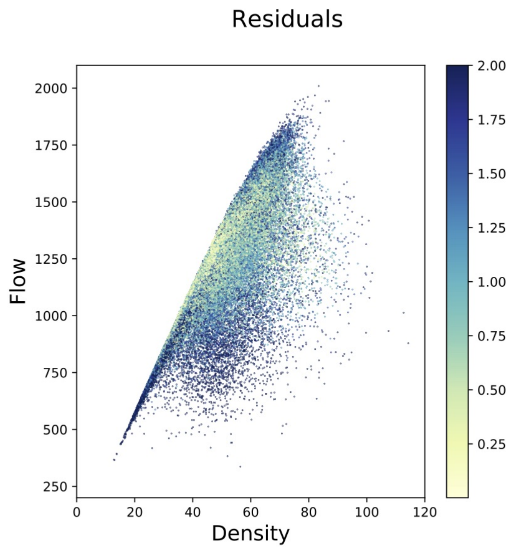
Table 4.2: Performance of the different models on the test dataset with desired speed of 30 mph and NGSIM dataset (flow vph, density vpm).

		MSE	MAE	MAPE
Test Dataset	Flow			
		MSE	MAE	MAPE
	CNN	3868.78	46.21	3.84
	MLP	17467.24	100.57	8.57
	SVR	(17497.20, 17641.33)	(100.11, 100.26)	(8.56, 8.62)
	ARIMA	21882.32	112.27	9.56
	Density			
		MSE	MAE	MAPE
	CNN	5.04	1.69	3.67
	MLP	26.58	3.97	8.84
	SVR	(26.49, 26.65)	(3.96, 3.97)	(8.75, 8.79)
	Arima	29.44	4.24	9.34
NGSIM Dataset	Flow			
		MSE	MAE	MAPE
	CNN	43667.47	167.43	22.23
	MLP	43756.79	157.43	35.29
	SVR	53901.08	167.98	41.26
	ARIMA	16726.27	105.69	10.55
	Density			
		MSE	MAE	MAPE
	CNN	97.13	8.44	22.50
	MLP	116.86	8.88	32.45
	SVR	259.84	11.39	60.26
	Arima	47.92	5.53	10.04

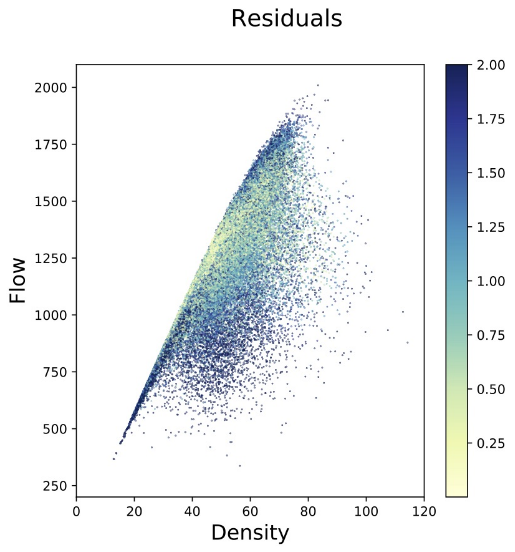
the NGSIM dataset and then estimating its performance. However, the CNN, MLP, and SVR trained on the simulated dataset are directly tested on the NGSIM dataset to evaluate their transferability. The CNN model performed better than the MLP and SVR, especially concerning the MAPE. Note that unlike the training dataset used for the CNN, MLP, and SVR models, the NGSIM data contains many data points in the congested traffic state. To address this issue, we create a more comprehensive dataset containing these instances to improve the model's generalization capability.



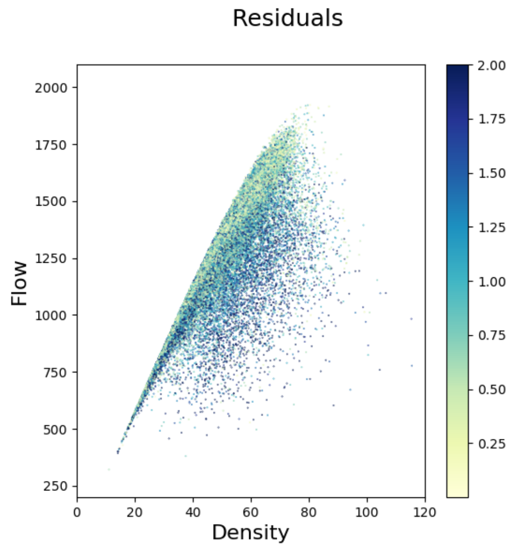
(a) CNN



(b) MLP



(c) SVR



(d) Arima

Figure 4.4: Normalized errors (residuals) for the test dataset with desired speed of 30 mph (flow vph, and density vpm).

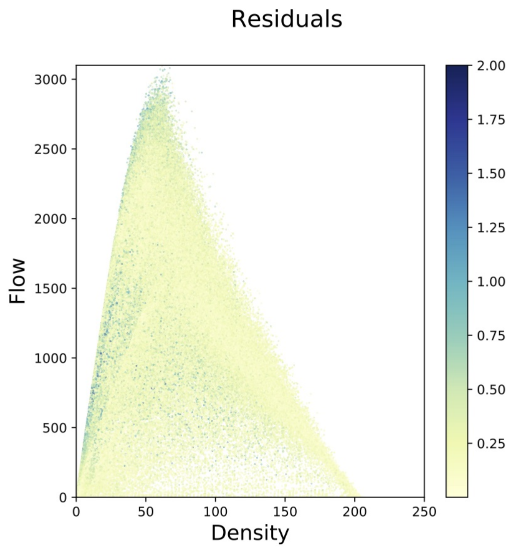
4.4.5 *Training with a Comprehensive Dataset*

To improve the generalization of the CNN model, a more comprehensive dataset is collected using the data simulator introduced previously. The random presence of slow-moving vehicles is added to the simulation to increase the development of congested traffic states and traffic breakdowns. Also, seven different desired speeds, including 30, 45, 50, 55, 65, 70, and 75 miles per hour, are simulated to create a more comprehensive dataset. The data is collected from 4088 runs of simulation, resulting in a total of 3,597,440 data points for segments of 2000 feet long and a prediction period of 20 seconds. The simulation runs are divided into sets of 2100, 994, 994 for training, validation, and testing, respectively.

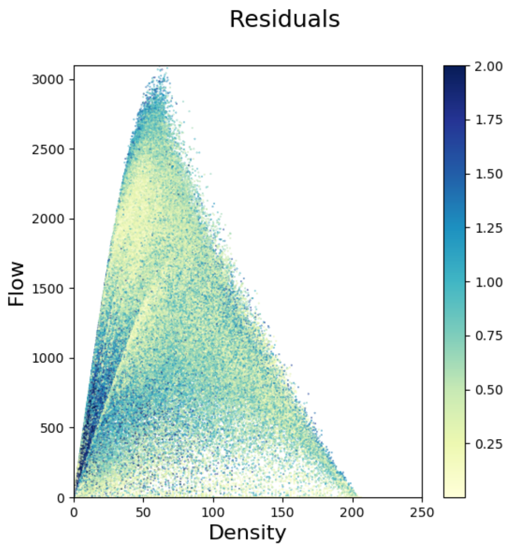
The comprehensive training dataset is used to train the CNN model as well as the other models, and the parameters of models are adjusted based on their performance on the validation dataset. Table 4.3 presents the performance of the CNN model as well as the other three models on the test-set of the comprehensive dataset. The prediction error for all the models is high, and this is due to the use of slow-moving vehicles to create congestion in the simulation. A sudden slow-moving vehicle results in a sudden flow drop and consequently high absolute percentage error when predicting for the time duration with the start of slow-moving from the previous time. Table 4.3 demonstrates a considerably better performance for the CNN model on the comprehensive dataset compared to the other three models concerning MSE, MAE, and MAPE in both flow and density predictions. The better performance of the CNN model is also evident in the plots of normalized errors in Figure 4.5. Figure 4.5a illustrates the normalized errors of the CNN model and shows a consistent distribution of errors over the fundamental diagram. The plots of normalized residuals for the MLP, SVR, and ARIMA are very similar to each other and indicate low levels of normalized error only limited to flow levels between 1500 to 2500 vehicles per hour and density levels lower than 100 vehicles per mile. Moreover, the CNN trained on the comprehensive dataset is also evaluated with the NGSIM dataset that resulted in improved performance of MAPE = 15.43 for flow and MAPE = 13.81 for density that are closer to the ARIMA model performance in Table 4.2.

Table 4.3: Performance of the different models on the comprehensive test dataset (flow vph, density vpm).

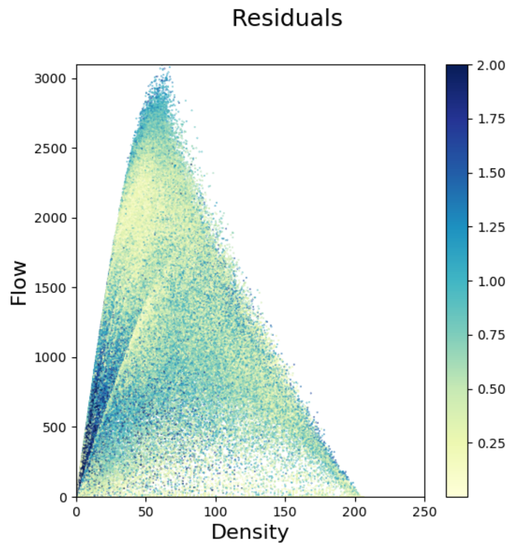
	Flow		
	MSE	MAE	MAPE
CNN	16939.98	91.83	177.62
MLP	70702.12	188.13	562.55
SVR	(69860.27, 71077.77)	(183.91, 184.50)	(525.35, 712.30)
ARIMA	88193.74	207.65	874.42
	Density		
	MSE	MAE	MAPE
CNN	9.70	2.30	10.57
MLP	63.99	5.65	37.28
SVR	(63.43, 63.96)	(5.57, 5.59)	(37.77, 41.46)
Arima	61.88	5.67	38.06



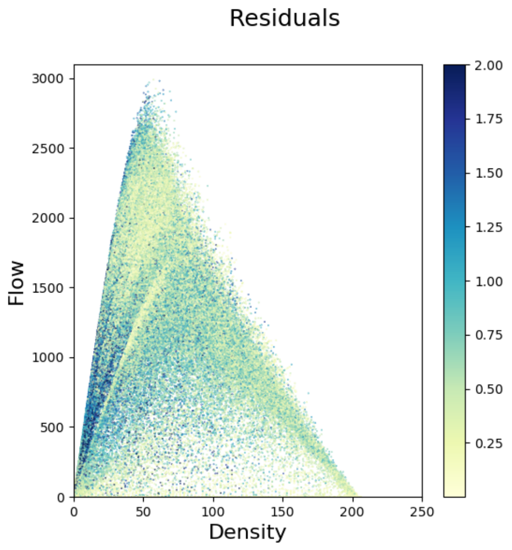
(a) CNN



(b) MLP



(c) SVR



(d) Arima

Figure 4.5: Normalized errors (residuals) for testing on the comprehensive dataset (flow vph, and density vpm).

4.5 Discussions

Considering the plot of normalized errors in Figure 4.5, the CNN model has learned a better generalization and prediction capability over the different regions of the fundamental diagram compared to the other three models. The CNN model is directly provided with the time-space diagram of the trajectories of the vehicles on the segment, which contains more information than the aggregated macroscopic features. Using the time-space diagram as the input to the CNN model provides the opportunity for the model to learn from the changes in traffic flow dynamics that cannot be easily captured in aggregated traffic features. Figure 4.6 illustrates the advantage of using the time-space diagram for traffic prediction over the use of flow and density in non-parametric approaches. All the three segments shown in Figure 4.6 have a similar flow level of 1500 ± 50 vehicles per hour and density of 75 ± 5 vehicles per mile, but with different traffic flow dynamics at the time step t and traffic state at time $t + 1$. Figure 4.6 also presents the predicted flow and density based on the CNN, MLP, and SVR models. Since the segments have a similar flow and density level at time t, the MLP and SVR predict the same traffic state for the next time step for all three of them. However, the CNN model predicts a different traffic state for each segment that is closer to its actual value. The CNN model in this chapter is trained to predict the traffic state based on the trajectory of the vehicles. This has enabled the CNN model to predict traffic state with the consideration of traffic flow perturbations.

4.6 Chapter Summary and Conclusions

This chapter proposed using the convolutional neural network (CNN) to predict the traffic state using the time-space diagram as the input. The interaction among the individual vehicles and their impact on traffic stream are not easily captured in aggregated traffic features such as flow and density. Using a CNN model provides the opportunity to utilize the features embedded in the time-space diagram and to account for the disturbances in the traffic stream, such as the formation of shockwaves, for a better traffic prediction. The CNN model in caparison with other non-parametric models, i.e., MLP, SVR, and ARIMA, shows a better generalization in predicting traffic state in different regions of the fundamental diagram. Evaluating the CNN model, which is trained based on the simulated data, against a real-world dataset (NGSIM US-101) also revealed its generalization capability (the performance is far better than MLP and SVR and almost as good as the ARIMA model, which is trained on the US-101 dataset).

This chapter does not capture the effects of lane-changing on traffic flow dynamics and prediction accuracy. Utilizing a 3D tensor input with the added dimension to address this challenge has been left for future research.

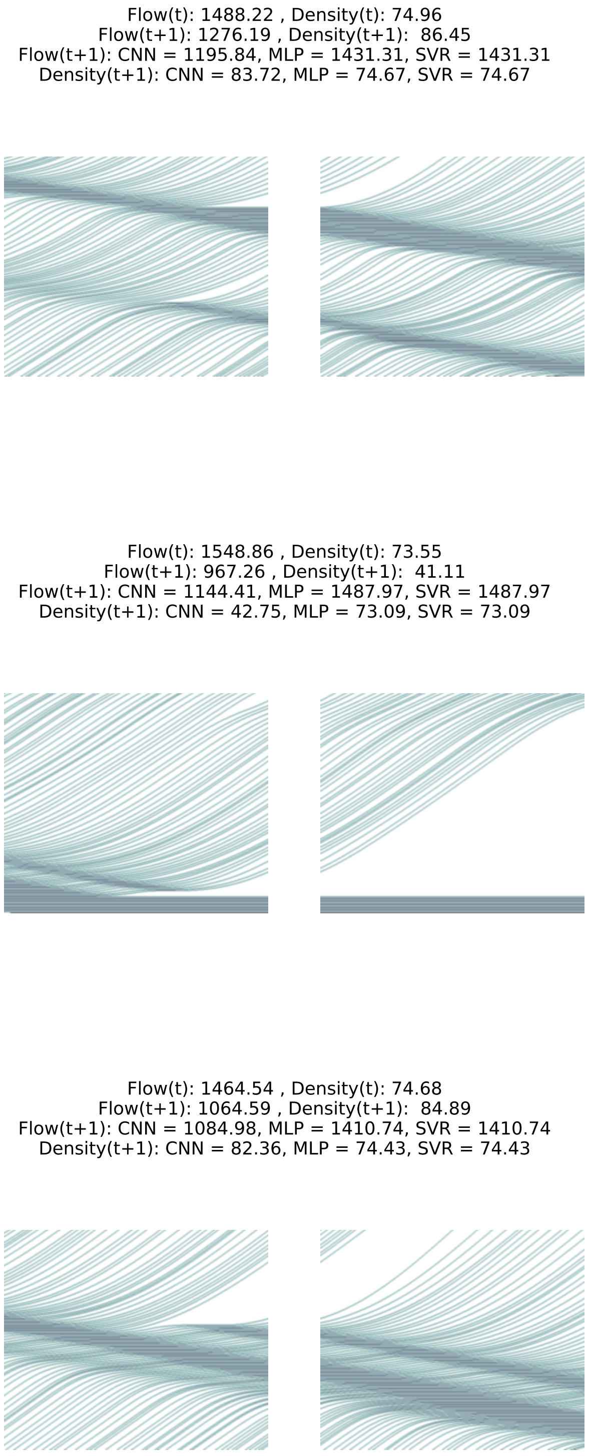


Figure 4.6: Traffic shockwaves and breakdown examples.

CHAPTER 5

PREDICTING TRAFFIC SHOCKWAVE FORMATION AND PROPAGATION FROM TIME-SPACE DIAGRAM

5.1 Introduction

The boundary between two different traffic states is known as a traffic shockwave. The driving dynamics change from one state to another as the speed of the vehicles and their spacing change. The differences between the two traffic states can be mild such as a high-speed traffic stream reaching a traffic stream with moderate speed and density, or it can be significant when reaching a high density and low-speed traffic stream (e.g., congested area). In general, when the traffic state changes, the vehicles need to respond by adjusting their speed and acceleration. Currently, the approaches adopted for guidance (e.g., lane changing) of the autonomous vehicles involves the consideration of the current state of the surrounding vehicles in terms of their location and speed with limited attention to the response of the other vehicles to their surrounding environment and how the traffic state could evolve (e.g., formation of shockwaves). As a result, predicting the propagation of traffic shockwaves in time and space can help in improving both the safety and performance of autonomous vehicles. Considering the valuable information that the connected vehicles could provide, we are proposing a methodology to predict the propagation of traffic shockwaves that accounts for both the individual behaviors and the collective change in the state of the traffic.

The connectivity provides the opportunity to monitor the traffic stream and how the traffic evolves over time and space more extensively than the location-specific traffic monitoring devices. The individual level location data transmitted by connected vehicles help to construct the time-space diagram. The time-space diagram is a comprehensive representation of the traffic stream without any abstraction or aggregation. The traffic flow dynamics, as well as vehicles' interactions and shockwave formation and propagation, are embedded in the time-space diagram. Chapter 4 used a matrix representation of the time-space diagram to predict the future traffic state of the roadway segment in the form of average flow and density. The objective of this chapter is to introduce a methodology to predict the propagation of traffic shockwaves (i.e., the change of flow and density over time and space) using a similar matrix representation of the time-space diagram. The main innovation of the proposed methodology is introducing the model that can learn the features embedded in the time-space diagram to predict the propagation of the traffic shockwaves.

The remainder of this chapter is organized as follows: the next section presents the details of the proposed methodology to predict the propagation of traffic shockwaves. This section is followed by a discussion on model configuration and its training process. The presented model is then evaluated on simulated and real-world trajectory data, and a discussion on the findings is presented. Finally, concluding remarks close the chapter.

5.2 Methodology

The traffic state and the interaction among the vehicles are best captured in the time-space diagram. The time-space diagram is the plot of the trajectory of the vehicles traversing the roadway (i.e., space domain) over time. This plot is comprehensive and provides the location and speed (slope of each trajectory) of the vehicles, as well as the interaction among the vehicles and the traffic state. Besides, the traffic shockwaves, which are the boundary between the different traffic states, are evident in the time-space diagram. As a result, this chapter adopts the time-space diagram as valuable input and predicts the propagation of shockwave in a time-space diagram format.

5.2.1 Time-Space Diagram

In order to construct the time-space diagram as the input for the prediction, this dissertation assumes a connected vehicles environment. In this environment, the connected vehicles share their speed and location every 0.1 seconds comparable to the basic safety message (BSM) [98]) transmitted through wireless communication. The time-space diagram can be generated in the form of a time-space matrix as proposed in Chapter 4. The time-space matrix (Figure 4.1) approximates the time-space diagram by dividing the time and space domains into cells of 10 feet by 100 milliseconds (discretization). The time-space matrix is a binary matrix, and the rows represent the discrete space domain, and the columns represent the discrete time domain. In this matrix, the cell value of one indicates the presence of a vehicle in that space and time cell, and the value of zero indicates an empty cell. The time-space matrix is a 2D tensor that can be used in the convolution process.

5.2.2 Convolutional Encoder-Decoder

This chapter proposes the use of a deep neural network to predict the propagation of the traffic shockwave from the current time-space diagram of the vehicles as shown in Figure 5.1. The convolutional encoder-decoder structure is an appealing type of mapping function for this chapter as the input and output of the networks are 2D tensors with similar properties. The convolution is the process of sliding a fixed size filter (e.g., three-dimensional receptive window) over the input tensor. Each convolutional layer applies the convolution process to the output of the previous layer and provides an output tensor. The convolution process accounts for the spatial correlation among the units that fit in the receptive window of the filter. Also, sliding the same filter over the input space ensures feature detection independent of its location. This location independence aspect of the convolutional layers makes them a practical choice to encode the time-space matrix since the traffic shockwaves can occur at any point in time and space.

There are different convolutional encoder-decoder network architectures depending on the use of convolutional, deconvolutional, pooling, and upsampling layers. Some networks only use convolutional layers in both encoder and decoder components, such as the Fully Convolutional Network (FCN) [99] and Seg-Net [100]. While other networks such as DeconvNet [101] and RED-Net [102] use deconvolutional layers in the decoder component. Some of the challenges in the convolutional encoder-decoder network are the vanishing gradient and reconstructing lost features from the max-pooling and convolution process. The use of skip connections [102], and memorizing the maximum features [101, 100] of the pooling process to use for the upsampling are the solutions. The skip connections inspired by the residual network (ResNet)[80] allow the signal to be propagated to the bottom layers and address the vanishing gradient.

The proposed encoder-decoder architecture in Figure 5.1 is inspired by the RED-Net [102] developed for image restoration and consists of symmetric layers of convolution and deconvolution with skip-layer connections. The encoder component of the network contains three pairs of convolutional layers with a small receptive window of 3×3 and increasing channels from 16 to 64. The decoder component of the network contains symmetrical deconvolutional layers. The deconvolution process, unlike the convolution process, associates a single input with multiple outputs. The encoder component encodes the features embedded in the time-space diagram. The decoder component predicts the propagation of the traffic shockwaves in the form of a new time-space diagram.

The skip-layers connect the symmetric convolution and deconvolution layers every two layers. The skip-layer connection sums the convolutional feature maps with the deconvolutional

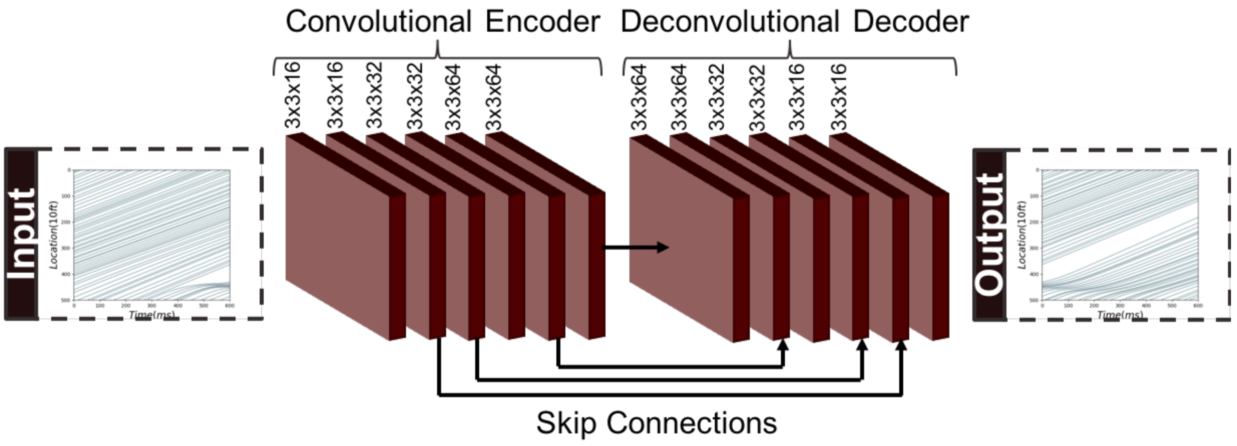


Figure 5.1: Shockwave prediction model: a convolutional encoder-decoder network.

feature map element-wise. The encoding convolutional layers extract the main features and abstract the input, while the deconvolution layers decode the abstract input and predict the shockwave propagation. The proposed network is deep, and the skip-layer connections allow the propagation of the gradient to the beginning layers of the network. The skip-layer connections address the vanishing gradient problem in very deep networks. Moreover, the proposed architecture is capable of taking any size of the time-space matrix as input since the network only utilizes the convolutional and deconvolutional layers.

5.2.3 Simulated Data

There are limited available data on the trajectory of the individual vehicles to create an extensive data set for the training of the proposed deep neural network that can generalize well. As a result, to construct a comprehensive and large data set, this chapter uses the microscopic simulator that was introduced in section 3.2 to collect the trajectory of the vehicles. The microscopic simulator adopts the Intelligent Driver Model (IDM) [59] as its car-following logic, and the MOBIL [60] as its lane-changing logic.

The simulation collects the trajectory of the vehicles traversing a three-lane roadway segment with a length of 40000 feet over 15 minutes. At every simulation run, unique and random IDM and MOBIL parameters are assigned to every vehicle to make the simulation more realistic. The simulation starts by initializing random and unique IDM and MOBIL parameters for the vehicles in the simulation. The IDM and MOBIL parameters initialization are based on the ranges and distributions indicated in Table 5.1. In order to create a data set with different traffic states from free-flow to fully congested, two types of disturbances are used in the simulation. Sudden deceleration of a random vehicle for a short period (e.g., 15 seconds) creates a speed drop perturbation and disturbs the traffic stream. Another type of disturbance used in the simulation is forcing a random slow-moving vehicle for a more extended period, such as 5 minutes, to create congestion and traffic breakdown. Both of the disturbances result in the formation of shockwaves in the traffic stream.

The start time of these two disturbances is limited during periods of $(20i, 20i+20)$ seconds, and i being even numbers between 0 and 45. When constructing pairs of input and output data for the training of the model, this limitation becomes useful. Both of these disturbances occur at a random location, duration, and intensity controlled by the parameters provided in Table 5.1. The model cannot predict random occurrences of these disturbances, and this limitation helps to exclude the start of these disturbances from the output data. Also, for each simulation run, the desired speed is randomly (uniform distribution) selected from different speed limits, including 30, 45, 50, 55, 65, 70, 75 miles per hour to create a more comprehensive data set.

5.3 Model Configuration and Training

5.3.1 Input and output data

The proposed encoder-decoder (Figure 5.1) of this chapter takes the time-space matrix as input and predicts the propagation of the traffic shockwaves in the same time-space matrix form. The binary time-space matrix (Figure 4.1) not only presents the traffic shockwaves but also depicts the crisp location of the individual vehicles in time and space domains. Training the network to output a binary tensor of shape $(200, 200)$ is challenging, and breaking the binary constraint improves the training. Averaging the cells of the time-space matrix with their neighbors, Figure 5.2a, blurs the exact location of the vehicles on time and space domains; however, averaging over a small window maintains the propagation of traffic shockwaves (Figure 5.2b). The colors of points on the averaged time-space diagram presented in Figure 5.2b change from light yellow to red proportional to the value of cell ranging from 0 to 1. Taking the averaged time-space matrix as the type of output improves the training of the network. The encoder-decoder network approximates the mapping function from the averaged time-space matrix of segment x over the period of $(t-20, t)$ to the averaged time-space matrix of the segment x over the period of $(t, t+20)$. The averaged time-space matrix is derived by replacing every cell in the time-space matrix with the average of itself and its neighbors up to 50 feet and 0.5 seconds on each side (i.e., averaging window of 100 feet by

Table 5.1: Simulator parameters.

IDM Parameters				
Parameter	Minimum	Maximum	Distribution	Unit
free acceleration exponent δ	4		constant	
jam distance s_0	6.56		constant	ft
maximum acceleration a	2.5	3.5	uniform	ft/s^2
desired deceleration b	4.0	8.0	uniform	ft/s^2
desired time gap T	1.0	2.0	uniform	s

MOBIL Parameters				
acceleration threshold	0.33		constant	ft/s^2
maximum safe deceleration	13		constant	ft/s^2
politeness factor	0.25	1.0	uniform	

Constant Parameters		
Parameter	Value	Unit
lanes	3	#
segment length	40000	ft
simulation time	900	s
vehicles	2000	#

Random Parameters				
Parameter	Mean	Standard Deviation	Distribution	Unit
initial gap	100	30	half normal	ft
deceleration wave duration	15	5.0	normal	s
Parameter	Minimum	Maximum	Distribution	Unit
number of deceleration waves	60	100	discrete uniform	#
intensity of deceleration waves	1.5	2.5	uniform	ft/s^2
number of slow-moving vehicles	3	7	discrete uniform	#
speed of slow-moving vehicles	$0.05\times V_0$	$0.70\times V_0$	uniform	ft/s
slow-moving duration	300	500	uniform	s

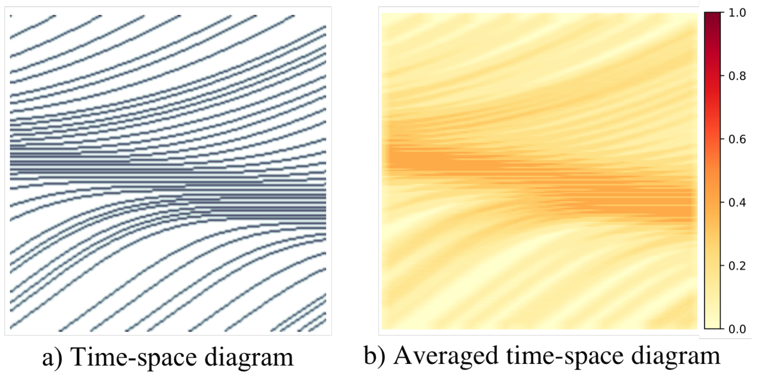


Figure 5.2: Averaged time-space diagram

1 second).

5.3.2 Training data

The microscopic traffic simulator provides a 40000 feet by 900 seconds time-space diagram for each lane and simulation run. This diagram can be divided into 900 smaller time-space diagrams for segments of 2000 feet and a shorter period of 20 seconds. The 900 smaller time-space diagrams are divided into 450 input and output data pairs. One could extract more pairs of input and output if they relax the limitation on keeping the start of the artificial disturbances in the input data. The data is collected from more than 2000 simulation runs resulting in more than 0.9 million data points. The collected data is divided into training, validation, and testing sets with ratios of 80%, 10%, and 10%, respectively.

5.3.3 Training

Training is the iterative process of adjusting the trainable parameters of the model to gradually minimize the loss function. The convolutional encoder-decoder of this chapter contains 180,449 trainable parameters. Adopting the small receptive window of 3×3, and fully convolutional and deconvolutional layers kept the number of network parameters small. The model parameters are updated in multiple iterations (steps). At each iteration, the loss function is estimated for a batch of data points, and the parameters are adjusted based on their loss gradient times the learning rate (a small constant). The Adam optimizer [90] is a stochastic gradient-based optimizer that is adopted for the training of the network of this chapter.

5.3.4 Loss Function

The prediction model of this chapter is a regression model that maps the current averaged time-space diagram to the future averaged time-space diagram. The mean squared error (MSE), Equation (4.3), is a standard performance measure used as the loss function for the training of regression type neural networks. The output of network (model) F with parameters Θ for input X^i is $F(X^i;\Theta)$, and the true value of output is Y^i. Mean absolute error (MAE), Equation (5.2), is another performance measure for regression problems. However, the MAE is not useful as the loss function and estimation of gradients in neural networks.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} ||F(X^i;\Theta) - Y^i||^2 \tag{5.1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} ||F(X^i;\Theta) - Y^i|| \tag{5.2}$$

The input and output of the model of this chapter are 2D tensors of size $(200, 200)$. A smoothed version of the output can be constructed by replacing each cell of the output tensor with the average of itself and its neighboring cells. A well-trained neural network model is expected to predict outputs comparable to the true outputs. Besides, it is expected that the smoothed versions of the predicted and true outputs are also comparable. In order to speed up the training (convergence) of the model and to guide the gradients, this chapter proposes the use of the custom loss function of Equation 5.3. The MSE_{10}, MSE_5, and MSE_3 are the estimated MSE between the true and predicted outputs when smoothed with sliding average windows of size 10×10, 3×3, and 5×5 respectively. The sliding window size indicates the extent of neighboring cells considered in the estimation of the average for that cell. Adopting this custom loss function significantly improved the convergence of the training process.

$$loss = MSE + 1000(MSE_{10} + MSE_5 + MSE_3) \tag{5.3}$$

The training process of the model is conducted in two steps. In the first step, the model is trained using the loss function of Equation 5.3 until the loss value on the validation set starts increasing. Then, in the second step, the model is retrained using the MSE, Equation (5.1), as the loss function.

Table 5.2: Model performance on time-space matrix

	Validation Dataset	
	MSE	$MSE_{10} + MSE_5 + MSE_3$
Training Step 1	0.0037	0.0071
Training Step 2	0.0029	NA
	Testing Dataset	
	MSE	MAE
Trained Model	0.0030	0.0408

5.4 Results and Discussions

In the training process of the model, a batch size of 60 and the early stopping policy are used to prevent overfitting the training data. The loss function is estimated at the end of every epoch (a complete iteration over the entire dataset). The training is stopped after five epochs from the one with the minimum loss on the validation dataset. Table 5.2 presents the performance of the network of this chapter in prediction on the validation and testing dataset. Training the model with the custom loss function, Equation (5.3), helped the convergence in the first step of training. Also, retraining the model in the second step by adopting the original MSE as the loss function further improved the model's performance from the MSE error of 0.0037 to 0.0029. According to Table 5.2, the performance of the fully trained model in terms of MSE and MAE on the testing dataset are 0.0030 and 0.0408, respectively.

5.4.1 Traffic shockwave propagation prediction

Figure 5.3 presents some of the traffic predictions of the model in the form of the averaged time-space matrix. The prediction model of this chapter takes the averaged time-space matrix of time $(t - 20, t)$ as input (x) and predicts the future averaged time-space matrix of time $(t, t + 20)$ as the output (y). Comparing the predictions of the model (predicted y) and the actual states of the traffic (y) in Figure 5.3, the model is capable of predicting the propagation of the traffic shockwaves. According to this figure, the predicted averaged time-space diagrams present dissemination, propagation, the forward and backward movement of the traffic shockwaves over the evaluated segment of roadway.

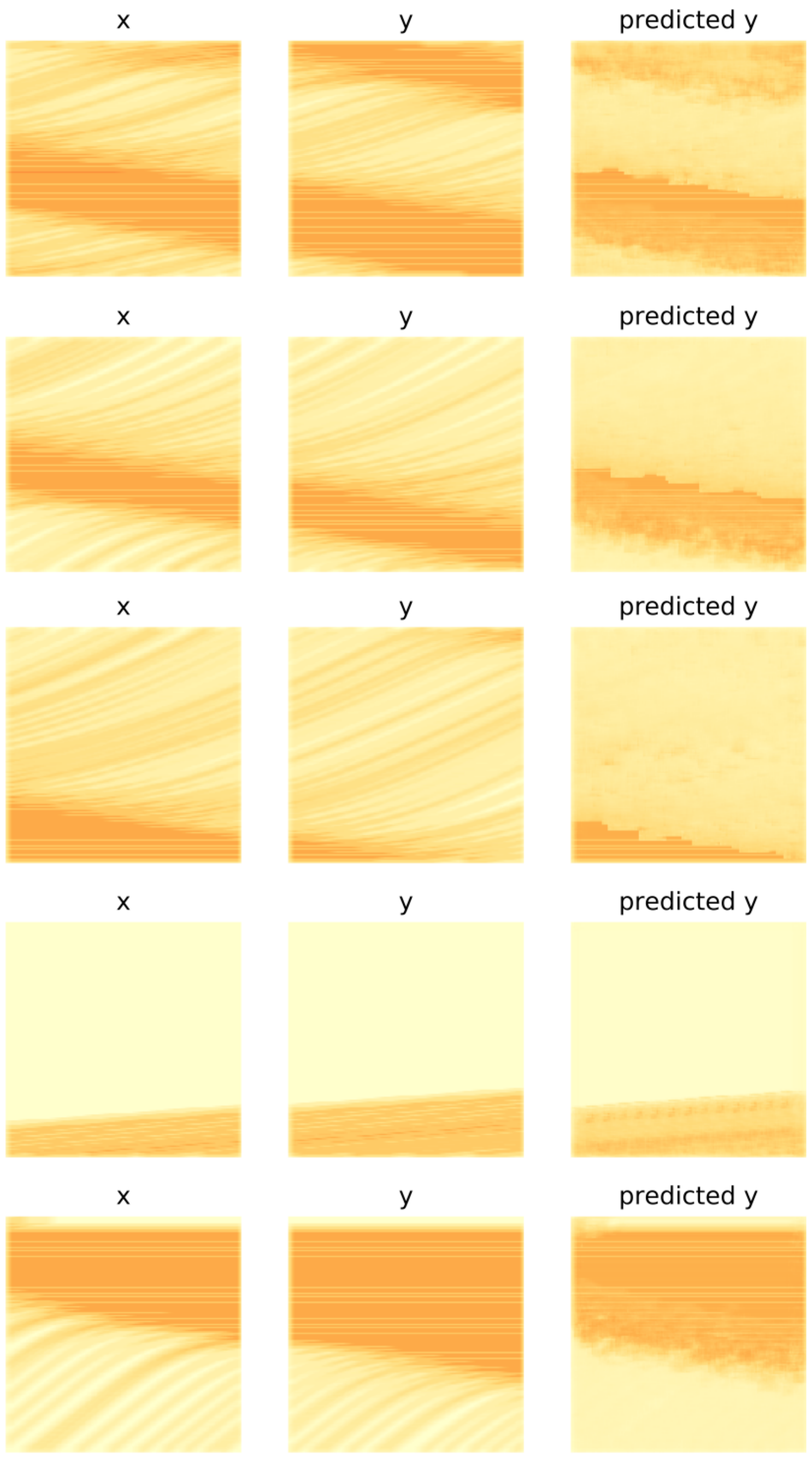


Figure 5.3: Traffic shockwave propagation prediction results

5.4.2 Density time-space matrix

The traffic shockwave propagation prediction of the network can be evaluated more quantitatively. The traffic shockwave is the boundary between two states of traffic. [87] estimates the average density $K(A)$ for a time-space block of A (e.g. 100 feet by 1 second) based on Equation (5.4). In this equation, $|A|$ is the area of the time-space block A, and $t(A)$ stands for the total time spent by all the vehicles going through block A.

$$K(A) = \frac{t(A)}{|A|} \tag{5.4}$$

As specified in the methodology section, the time-space matrix is a binary matrix constructed by dividing time and space domains into 10 feet by 100 milliseconds cells. In this matrix, one represents the presence of a vehicle in that time and space cell, and zero represents an empty cell. The number of occupied cells of the time-space block A equals the summation of all the cells of its representative binary time-space matrix (i.e., $sum(A)$). As a result, the total time spent by all the vehicles going through any arbitrary time-space block of A is equal to multiplying the number of occupied cells in that block by 0.1 seconds, Equation 5.5.

$$t(A) = 0.1 \times sum(A) \tag{5.5}$$

Considering Edie's [87] definition on the average density of a time-space block, the averaged time-space matrix ($\overline{TS}$) can be used to estimate the density time-space matrix ($\mathbf{K}$). Similar to the time-space matrix, the rows of this matrix represent the discrete space domain, and their columns represent the discrete time domain. The values of each cell in the matrix K is the average density of a time-space block (e.g., 100 feet by 1 second) centered at that location in time and space. The density time-space matrix depicts the change in traffic state over time and space and consequently the propagation of the traffic shockwaves.

As mentioned in the methodology section, the averaged time-space matrix ($\overline{TS}$) of this chapter is estimated by replacing every cell in the time-space matrix with the average of itself and its neighbors up to 50 feet and 0.5 seconds on each side. Each cell in the time-space matrix is representative of a cell with dimensions of 10 feet in the space domain and 0.1 s in the time domain. The averaging window of 100 feet by 1 second is equivalent to a 10×10 averaging window on the time-space matrix. In other words, each cell of the averaged time-space matrix ($\overline{TS}$) is the average of 100 cells in the time-space matrix. Therefore, if the averaged time-space matrix ($\overline{TS}$) is multiplied by 100, the cells of the resulting matrix indicate the number of occupied cells in the blocks of 100 feet by 1 second centered on that

Table 5.3: Network performance on density time-space matrix

	Validation Dataset	
	MSE	$MSE_{10} + MSE_5 + MSE_3$
Training Step 1	1031.50	1979.36
Training Step 2	808.47	NA
	Testing Dataset	
	MSE	MAE
Trained Model	836.35	21.54

location on the time-space matrix. As a result, the density time-space matrix ($\mathbf{K}$) can be estimated based on Equation (5.6). In this equation, the constant 5280 is applied for the unit conversion from feet to miles. Vehicles per mile (vpm) is the unit for the values in the resulting density time-space matrix ($\mathbf{K}$).

$$\mathbf{K} = \frac{t(A)}{|A|} = \frac{100 \times \overline{TS} \times 0.1}{100 \times 1} \times 5280 = 528 \times \overline{TS} \tag{5.6}$$

According to Equation (5.6), the averaged time-space matrix ($\overline{TS}$) can be converted to the density time-space matrix ($\mathbf{K}$) by a constant scalar. Therefore, the prediction (output) of the model is proportional to the density time-space matrix. The performance of the model in Table 5.2 are updated for the density time-space matrix presented in Table 5.3. According to Table 5.3, the mean absolute error of the model in predicting the density for small blocks of 100 feet by 1 second is 21.54 vehicles per mile (vpm). Root mean squared error (RMSE) is another valuable performance measure that has the same unit as the output. Based on Table 5.3, the RMSE of the model in the prediction of the density on the testing dataset is estimated as 28.91 vpm. Considering a range of 200 vpm for the density, the MAE and RMSE of the model are between %10 to %14 of the range of density.

5.5 Evaluation on the NGSIM Dataset

The FHWA Next Generation Simulation (NGSIM) US-101 dataset [97] is also used to evaluate the proposed convolutional encoder-decoder model. This dataset covers three 15-minute periods during the morning peak. The time-space diagram for each lane is recreated from the trajectory data for the middle 2000 feet of the segment for every 20 seconds. The NGSIM data sums up to 660 data points for five lanes and is directly used to test the previously trained encoder-decoder model. The performance of the model on this dataset is reported in Table 5.4 in the form of MSE and MAE for both the time-space matrix and the

Table 5.4: Network performance on the NGSIM dataset

Time-space Matrix	
MSE	MAE
0.0044	0.0505
Density time-space matrix	
MSE	MAE
1226.64	26.66

density time-space matrix. The MSE and MAE values for the NGSIM dataset are slightly higher than the respective ones for the testing dataset. This difference is expected as the real-world collected data is different from simulated data. According to Table 5.4, the MAE and the root square of MSE (RMSE) of the model in predicting the density of small blocks of 100 feet by 1 second are 26.66 and 35.02 vpm. The MAE and RMSE values of the trained model on the NGSIM dataset are less than 15 percent of the density range. Moreover, Figure 5.4 compares the model predictions (predicted y) and the true states of the traffic (y) for a few input examples from the NGSIM dataset. According to this figure, the model can predict the propagation of traffic shockwaves on the real-world collected data. It should be noted that the model's performance can further improve by training the model with some of the collected data from NGSIM.

5.6 Chapter Summary and Conclusions

This chapter proposes a methodology to predict the propagation of traffic shockwaves in the form of the averaged time-space matrix. The averaged time-space matrix is comparable to a density time-space matrix derived from Edie's definition of average density. The traffic shockwave is the boundary between two traffic states, and the density time-space matrix depicts the state changes in the form of density. The result of the analysis indicated that the model is capable of predicting the dissemination, propagation, the forward and backward movement of the traffic shockwaves over the study segment. Moreover, the model's performance in the form of MAE and RMSE for predicting the density time-space matrix is 21.54 vpm and 28.91 vpm for the simulated testing dataset, and 26.66 and 35.02 vpm for the NGSIM dataset. Considering a range of 0 to 200 vpm for the density, the model's performance is acceptable for predicting the traffic shockwaves propagation.

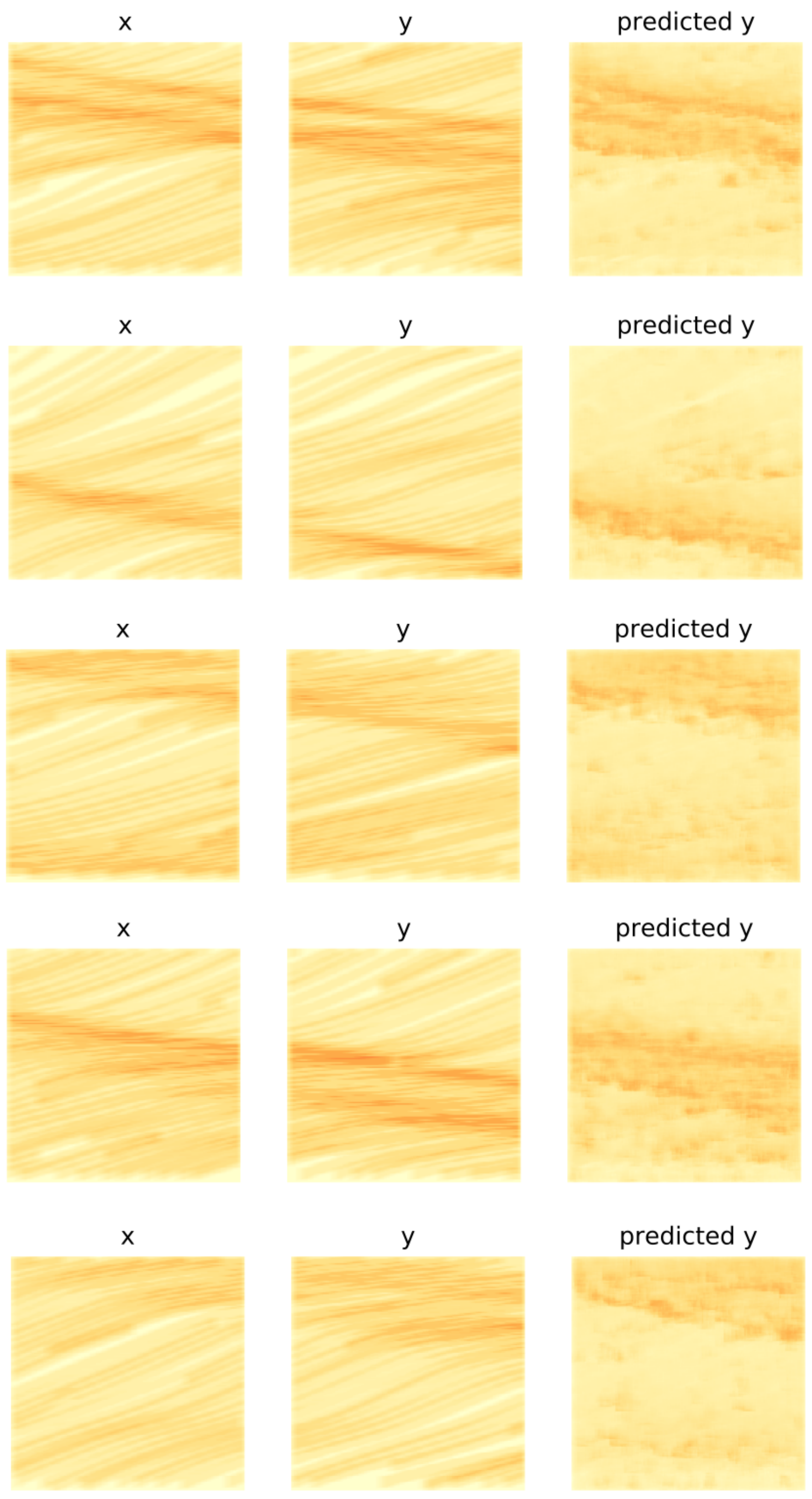


Figure 5.4: Traffic shockwave propagation prediction results on the NGSIM dataset

CHAPTER 6

PROBABILISTIC TRAFFIC STATE PREDICTION FROM INDIVIDUAL LEVEL TO AGGREGATED LEVEL

6.1 Introduction

Accurate prediction of the driving environment is crucial for safe path planning and navigation through traffic, particularly for connected and automated vehicles. The driving environment evolves as a result of interactions among the individual vehicles. These interactions can be defined as a combination of lateral and longitudinal maneuvers of vehicles in response to their driving environment. At the individual vehicle level prediction, structuring the driving task into different maneuvers and considering the vehicles' interactions can improve the vehicle trajectory prediction accuracy. As a result of a vehicle's maneuver, its future trajectory (e.g., its location and speed) changes. In many driving scenarios, more than one maneuver is feasible. Therefore, it would be more realistic to predict the location of the vehicles in a probabilistic manner based on the different maneuvers. Driving environments change as a result of the maneuvers taken by individual vehicles. With the increase in the prediction horizon, the accuracy of prediction decays due to the uncertainty in the vehicles' interactions and the increase in the possibility of different configurations and outcomes. For an extended prediction period, predicting at the aggregated level can be more appropriate. Furthermore, the aggregated level prediction can also benefit from considering the interaction among vehicles. As a result of the connectivity, the traffic state estimation at the macroscopic level can now take into account individual-level data. This chapter proposes estimating the macroscopic traffic states (e.g., flow, density, and space-mean speed) by aggregating probabilistic individual-level predictions.

The remainder of this chapter is organized as follows: the following section provides the details on the adopted probabilistic trajectory prediction model at the microscopic level and introduces the proposed probabilistic traffic estimates at the macroscopic level. Next, the presented probabilistic traffic state prediction approach is evaluated using real-world vehicle trajectory data, and the results and a discussion on the findings are presented. Finally, the concluding remarks close the chapter.

6.2 Methodology

6.2.1 Microscopic Level: Probabilistic Trajectory Prediction

The driving environment prediction at the individual level can be performed in the form of trajectory prediction. However, the dynamic nature of the driving environment, the interaction among vehicles, and the variation in driving behavior make predicting at the individual level challenging. The vehicle trajectory prediction can be improved by dividing the driving task into different maneuvers and considering the vehicle's interactions. Vehicle maneuvers alter the vehicle's future trajectory (e.g., location, speed). In many driving scenarios, multiple maneuvers are available, and focusing on a single driving maneuver may not be sufficient. Thus, it is more appropriate to adopt a probabilistic trajectory prediction model that includes the possibility of different maneuvers and considers the interaction among the vehicles. This dissertation utilizes the convolutional social pooling model proposed by Deo and Trivedi [5] as the core of the individual vehicle trajectory prediction module. This model predicts the probabilistic trajectory of a target vehicle for different longitudinal and lateral maneuvers considering the observed history of its surrounding vehicles and captures the interaction among the vehicles considering a bird's eye view of the driving environment. At the time of prediction, it is assumed that an observed history of the surrounding driving environment H is available, considering a connected vehicle environment or based on the sensory data collected by an automated vehicle. The maneuver-based probabilistic trajectory prediction can then be defined based on:

$$P(T|H) = \sum_{i=1}^{M} P_{\Theta}(T|H, m_i)P(m_i|H) \tag{6.1}$$

In Equation 6.1, the future trajectory, T, is predicted based on the observed history of the surrounding environment, H, and the combination of M possible maneuvers (i.e., m_i for $i = 1 \dots M$). Θ is the parameters of the conditional probability distribution of the vehicle trajectory. The prediction model can be trained to estimate Θ and to predict $P(m_i|H)$, the probability of individual maneuvers.

The convolutional social pooling [5] uses an LSTM encoder-decoder network to predict the probability distribution of a target vehicle's future location. The model estimates parameters of the conditional distribution of the target vehicle's location, considering the observed track histories. Moreover, this model expands the conditional distribution to different lateral and longitudinal maneuvers. The lateral maneuvers include three movements of maintaining the lane and lane-changing to the right and left. The longitudinal maneuvers include braking and not braking.

The original deep learning model of [5] predicts the probability of each of the lateral and longitudinal maneuvers separately and assumes independence between the lateral and longitudinal maneuvers. Predicting the probability of the lateral and longitudinal maneuvers separately results in the same probability of longitudinal maneuver independent of the lateral maneuver and vice versa. In other words, the probability of deceleration would be independent of the vehicle maintaining its current lane or moving to one of its neighboring lanes. However, the acceleration and deceleration due to a lateral maneuver depend on the new car following conditions (e.g., new leader) in the new lane. As a result, the lateral and longitudinal maneuvers are dependent, and this dissertation modifies the deep learning model to predict the joint probability of lateral and longitudinal maneuvers. In fact, after updating the model to predict the lateral and longitudinal maneuvers jointly, the model's performance in predicting the maneuver is increased by more than 8 percent on the NGSIM testing dataset of this chapter. More details on the training and testing dataset used for this model are provided in Section 6.3.

The location of a vehicle i over the roadway segment, can be described with longitudinal and lateral coordinates of the vehicle ($X_i = [x_i, y_i]^T$). The probabilistic trajectory prediction model is trained to estimate the probability of every possible maneuver, as well as the future trajectory of vehicle i, in the form of the mean (μ) and covariance (Σ) of a bivariate normal distribution for every time step t in the future and for every maneuver m.

$$X_i^{t,m} \sim N(\mu_i^{t,m}, \Sigma_i^{t,m}) \tag{6.2}$$

Equation 6.2 defines the probabilistic location of the vehicle considering one of the possible maneuvers. The probabilistic location of the vehicle can be defined based on a weighted summation of these bivariate Gaussian distribution models and the probability of each maneuver. Accordingly, the probability density function of the vehicle's location, $p(X^t)$ can be written as:

$$p(X^t) = \sum_m P(m)\Phi(X^{t,m}|\mu^{t,m}, \Sigma^{t,m}) \tag{6.3}$$

In this bivariate Gaussian mixture model, $P(m)$ is the probability of taking maneuver m, and Φ is the probability density function of the vehicle location for maneuver m at time t. Utilizing this approach, one can define the probabilistic macroscopic measures of traffic flow (i.e., flow, speed, density, and occupancy).

Probabilistic Occupancy Map

Occupancy map is a grid-based representation of the study area (surrounding environment). The roadway segment is divided into small cells (e.g., four by four feet) represented by a matrix with values indicating the probability of that cell being occupied. The probabilistic occupancy map of the study area can be created considering the probabilistic location of all the vehicles on the study segment. Considering the probabilistic location of individual vehicles (Equation 6.3), the occupancy probability of cell c with boundaries of $[x_1, x_2]$ and $[y_1, y_2]$ can be estimated based on:

$$P(O_c = 1) = \sum_v \int_{y_1}^{y_2} \int_{x_1}^{x_2} \sum_m P(m_v)\Phi(X_v^{t,m}|\mu_v^{t,m}, \Sigma_v^{t,m})dxdy \tag{6.4}$$

Equation 6.4 estimates the occupancy probability of each cell (e.g., c) of the occupancy matrix (O) by summing the probability of that cell being occupied by each of the vehicles on the study area (traffic scene).

Probabilistic Time-Space Diagram

The time-space diagram is the plot of the trajectory of vehicles over time and space. In order to create the probabilistic time-space diagram, this chapter utilizes the representation proposed in Chapter 4. The time-space matrix (Figure 4.1) approximates the time-space diagram by dividing the time and space domains into small cells of (e.g., 10 feet by 100 milliseconds). The time-space matrix is a binary matrix, and the rows represent the discrete space domain, and the columns represent the discrete time domain. In this matrix, the cell value of one indicates the presence of a vehicle in that space and time cell, and the value of zero indicates an empty cell. The time-space matrix is a 2D tensor that can be used in the convolution process. A probabilistic prediction of the time-space matrix can be created by breaking the binary constraint and replacing the value of each cell with the expected value of the time-space cell. The probability of vehicle v passing through a time-space cell ($[t_1, t_2]$,$[y_1, y_2]$) can be estimated by considering the probabilistic location of the vehicle at the beginning of the time period of the cell and the probabilistic location of the vehicle at the end on the time period of the cell. A vehicle can move from one time-space cell to another either by crossing the beginning of the space domain of the cell during its time period or remaining within the exact space boundaries similar to its previous cell. These two movements in time and space are depicted with red and yellow arrows in Figure 6.1 respectively. The probability of a vehicle passing through a time-space cell is the summation

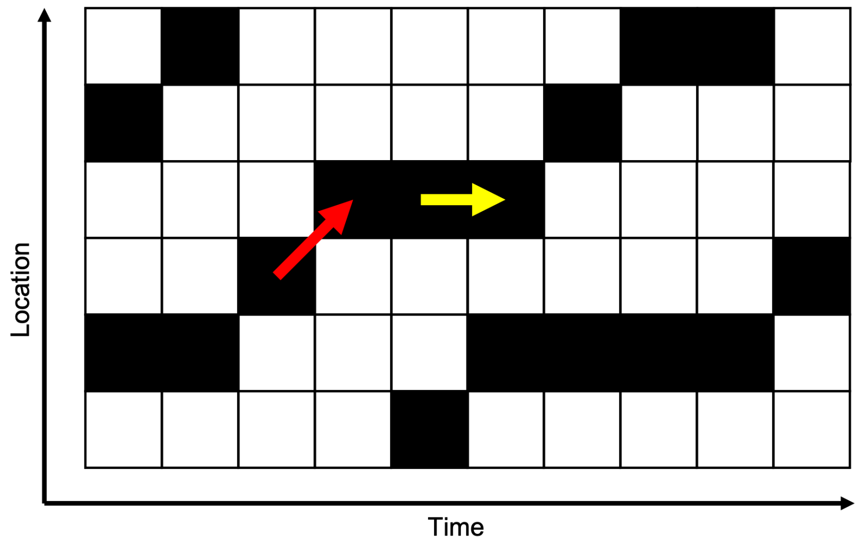


Figure 6.1: Moving from one time-space cell to another.

of the probability of both possible ways to enter the new time-space cell. The expected value of each cell in the time-space matrix (TS) is the summation of the probability of every vehicle going through that cell multiplied by its contribution to the value of the cell, which is one for all the vehicles.

$$E[TS_c] = \sum_v [P(y_v^{t_1} \leq y_1, y_v^{t_2} \geq y_1) + P(y_v^{t_1} \geq y_1, y_v^{t_2} \leq y_2)] \times 1 \tag{6.5}$$

Considering the Gaussian mixture model of the location of the vehicles (Equation 6.3), the expected value of each cell ($c = [[t_1, t_2], [y_1, y_2]]$) of the time-space matrix can be estimated based on the following equation.

$$\begin{aligned} E[TS_c] = \sum_v [&\int_{-\infty}^{y_1} \int_{-\infty}^{\infty} \sum_m P(m_v)\Phi(X_v^{t_1,m} | \mu_v^{t_1,m}, \Sigma_v^{t_1,m}) dxdy \\ &\times \int_{y_1}^{\infty} \int_{-\infty}^{\infty} \sum_m P(m_v)\Phi(X_v^{t_2,m} | \mu_v^{t_2,m}, \Sigma_v^{t_2,m}) dxdy \\ &+ \int_{y_1}^{\infty} \int_{-\infty}^{\infty} \sum_m P(m_v)\Phi(X_v^{t_1,m} | \mu_v^{t_1,m}, \Sigma_v^{t_1,m}) dxdy \\ &\times \int_{-\infty}^{y_2} \int_{-\infty}^{\infty} \sum_m P(m_v)\Phi(X_v^{t_2,m} | \mu_v^{t_2,m}, \Sigma_v^{t_2,m}) dxdy] \end{aligned} \tag{6.6}$$

It should be noted that in Equation 6.6, the probability of passing the edge of the cell, $P(y_v^{t_1} \leq y_1, y_v^{t_2} \geq y_1)$ is approximated by multiplying the probability of the location of the vehicle being behind that point at the beginning of the given period multiplied by the probability of the vehicle being beyond that point at the end of the given period. This approximation is based on the consideration that the dependence between the mean (μ) and covariance (Σ) of the distributions for every time step t on the previous time steps' predictions is captured by the LSTM decoder component of the convolutional social pooling [5]. Similar approximation is considered when estimating the probability remaining within the space boundaries ($P(y_v^{t_1} \geq y_1, y_v^{t_2} \leq y_2)$).

6.2.2 Macroscopic Level: Probabilistic Density and Flow Estimation

Probabilistic Density Estimation

Traffic density is defined by the number of vehicles occupying the unit length of the roadway. Based on this definition, the expected value of traffic density (K) can be estimated considering the expected number of vehicles on the roadway segment divided by the length of the roadway. The expected number of vehicles on a roadway segment with boundaries of $[x_1, x_2]$ and $[y_1, y_2]$ is the summation of the probability of every vehicle being on the segment multiplied by its contribution to the density, which is one for every passenger car. Note that this value can be adjusted to account for heavy vehicles and their impact on traffic flow dynamics (i.e., the concept of passenger car equivalent).

$$E[K(t)] = \frac{\sum_v P(y_1 < y_v^t < y_2, x_1 < x_v^t < x_2) \times 1}{y_2 - y_1} \tag{6.7}$$

Considering the probabilistic location of vehicles from Equation 6.3 and the general definition of expected density in Equation 6.7, the expected density can be estimated based on:

$$E[K(t)] = \frac{\sum_v \int_{y_1}^{y_2} \int_{x_1}^{x_2} \sum_m P(m_v)\Phi(X_v^{t,m} | \mu_v^{t,m}, \Sigma_v^{t,m}) dx dy}{y_2 - y_1} \tag{6.8}$$

The numerator of Equation 6.8 is comparable to the the occupancy probability in Equation 6.4. The occupancy map is a matrix representation of the study area with values indicating the probability of that cell in space being occupied. Consequently, the expected density can also be estimated by summing all the values of the submatrix of the occupancy map that covers the target area and divided by the length of that segment.

$$E[K(t)] = \frac{\overrightarrow{1}^T O[[r_{y_1}, r_{y_2}]; [c_{x_1}, c_{x_2}]] \overrightarrow{1}}{y_2 - y_1} \tag{6.9}$$

Where O is the occupancy matrix and r_{y_1} and r_{y_2} are the row number of the occupancy map corresponding to y_1 and y_2 respectively. c_{x_1} and c_{x_2} denote the column number of the occupancy map corresponding to x_1 and x_2 respectively and $\overrightarrow{1}$ is a vector of all ones. The vectors of all ones are used in the multiplication to facilitate summing all the members of the submatrix of the occupancy map.

Probabilistic Flow Estimation

The traffic flow rate is defined as the number of vehicles passing a point during a given period. Accordingly, the expected value of traffic flow (Q) at a given point can be estimated considering the expected number of vehicles passing the point over a given period. The probability of a vehicle passing a specific point of roadway over a period of time (e.g., t_1 to t_2) is the probability of the location of the vehicle being behind that point at the beginning of the given period and the vehicle being beyond that point at the end of the given period.

$$P_v^{crossing}(Y) = P(y_v^{t_1} < Y, y_v^{t_2} > Y) \tag{6.10}$$

In Equation 6.10, $P_v^{crossing}(Y)$ is the probability of vehicle v crossing point Y during the time period of t_1 to t_2. The expected number of vehicles passing a specific point on the roadway over a fixed period is the summation of the probability of every vehicle passing that point during that period multiplied by its contribution to the flow rate, which is one for all passenger cars. Considering the Gaussian mixture model of the vehicle's location (Equation 6.3), the expected flow rate, $E[Q]$, can be estimated based on the following equations:

$$E[Q(Y)] = \frac{\sum_v P(y_v^{t_1} < Y, y_v^{t_2} > Y) \times 1}{t_2 - t_1} \tag{6.11}$$

$$\begin{aligned} E[Q(Y)] = \sum_v [\int_{-\infty}^{Y} \int_{-\infty}^{\infty} \sum_m P(m_v) \Phi(X_v^{t_1,m} | \mu_v^{t_1,m}, \Sigma_v^{t_1,m}) dxdy \\ \times \int_{Y}^{\infty} \int_{-\infty}^{\infty} \sum_m P(m_v) \Phi(X_v^{t_2,m} | \mu_v^{t_2,m}, \Sigma_v^{t_2,m}) dxdy] / (t_2 - t_1) \end{aligned} \tag{6.12}$$

In Equation 6.12, the probability of a vehicle passing a specific point of roadway over a period of time (e.g., t_1 to t_2) is approximated by the probability of the location of the vehicle being behind that point at the beginning of the given period multiplied by the probability of the vehicle being beyond that point at the end of the given period. This approximation is based on the consideration that the dependence between the mean (μ) and covariance (Σ) of the distributions for every time step t on the previous time steps' predictions is captured by the LSTM decoder component of the convolutional social pooling [5].

Probabilistic Space-Mean Speed

There exists a variety of definitions of the space-mean speed ($\bar{U_s}$) in the literature that are slightly different [103]. The space-mean speed is the speed based on the average time taken by the vehicles to travel a specific segment of the roadway.

$$\bar{U_s} = \frac{D}{\frac{1}{N}\sum_i t_i} \tag{6.13}$$

In this equation, t_i is the time that took vehicle i to travel a specific roadway segment with the length of D. One of the challenges with this definition is that it only averages the travel time of the vehicles that traveled the roadway segment completely. A more general definition of the space-mean speed in line with this definition describes the space-mean speed considering all the vehicles traveling a specific segment of roadway over a given period of time [103]. In this definition, the space-mean speed is estimated by dividing the total distance traveled by all the vehicles by the total time spent by those vehicles on the specific segment of the roadway over a given period of time:

$$\bar{U_s} = \frac{d(A)}{t(A)} \tag{6.14}$$

In this equation, $d(A)$ and $t(A)$ denote the total distance traveled and total time spent by all the vehicles going through the time-space black A. This general definition of the space-mean speed is comparable to the ratio of Edie's [104] generalized average flow and density of a time-space block A. The expected distance traveled by each vehicle in the time-space block of A with boundaries $[t_1, t_2]$ and $[y_1, y_2]$ can be estimated based on the following equations:

$$E[d_v] = \int_{y_1}^{y_2} P(y_v^{t_1} \leq y, y_v^{t_2} \geq y)dy \tag{6.15}$$

Equation 6.15 estimates the expected distance traveled by vehicle v in time-space block A, considering the probability of the vehicle going through every point of the segment of roadway in block A during the period of this time-space block. Similar to the concepts of the probabilistic time-space diagram, the time and space domains can be divided into smaller time-space cells. In this representation, the expected distance traveled by each vehicle can be estimated based on the following equation:

$$E[d_v] = \sum_i P(y_v^{t_1} \le y_i, y_v^{t_2} \ge y_i)\Delta y \tag{6.16}$$

Equation 6.16 is similar to Equation 6.15 except the integration is replaced with a summation. The expected time spent by each vehicle in the time-space block of A with boundaries $[t_1, t_2]$ and $[y_1, y_2]$ can be estimated based on the following equation:

$$E[t_v] = \int_{t_1}^{t_2} P(y_v^t \le y_2, y_v^t \ge y_1)dt \tag{6.17}$$

Equation 6.17 estimates the expected time spent by vehicle v in time-space block A, considering the probability of the vehicle being within the space boundaries of block A at every point of time within the period of time-space block. Similar to the concepts of the probabilistic time-space diagram, the time and space domains can be divided into smaller time-space cells. In this representation, the expected time spent by each vehicle can be estimated using summation instead of integration:

$$E[t_v] = \sum_t P(y_v^t \le y_2, y_v^t \ge y_1)\Delta t \tag{6.18}$$

From Equation 6.14, the space-mean speed ($\bar{U_s}$) can be estimated from the total expected distance traveled divided by the total time spent by all the vehicles on the study segment during the study period:

$$\bar{U_s} = \frac{E[d(A)]}{E[t(A)]} = \frac{\sum_v \sum_i P(y_v^{t_1} \le y_i, y_v^{t_2} \ge y_i)\Delta y}{\sum_v \sum_t P(y_v^t \le y_2, y_v^t \ge y_1)\Delta t} \tag{6.19}$$

Fundamental Equation of Traffic Flow

The traffic flow rate is defined as the number of vehicles passing a roadway point during a given period. The traffic flow rate is a function of location, $Q(y)$, and its average over the definite segment of roadway, $[y_1, y_2]$ can be estimated by integration of the traffic flow function:

$$\bar{Q} = \frac{\int_{y_1}^{y_2} q(y)dy}{(y_2 - y_1)} \tag{6.20}$$

The integration in Equation 6.20 can be approximated by a summation over small roadway segments (Δy). Considering Equation 6.11, the average traffic flow can be estimated based on the following equation:

$$\bar{Q} = \frac{\sum_i \sum_v P(y_v^{t_1} < Y, y_v^{t_2} > Y)\Delta y}{(t_2 - t_1)(y_2 - y_1)} \tag{6.21}$$

Traffic density is defined by the number of vehicles occupying the unit length of the roadway at a point of time. The traffic density is a function of time, and similar to the average flow, the average density over the definite time period, $[t_1, t_2]$ can be estimated by integrating the traffic density function:

$$\bar{K} = \frac{\int_{t_1}^{t_2} K(t)dt}{(t_2 - t_1)} \tag{6.22}$$

Taking into account Equation 6.7 and replacing the integration of Equation 6.22 with summation, the average density can be defined as:

$$\bar{K} = \frac{\sum_t \sum_v P(y_1 < y_v^t < y_2)\Delta t}{(y_2 - y_1)(t_2 - t_1)} \tag{6.23}$$

From equations 6.21 and 6.23, the ratio of the average flow and average density of a time-space block A with boundaries $[t_1, t_2]$ and $[y_1, y_2]$ is:

$$\frac{\bar{Q}}{\bar{K}} = \frac{\sum_i \sum_v P(y_v^{t_1} < Y, y_v^{t_2} > Y)\Delta y}{\sum_t \sum_v P(y_1 < y_v^t < y_2)\Delta t} = \frac{\sum_v \sum_i P(y_v^{t_1} < Y, y_v^{t_2} > Y)\Delta y}{\sum_v \sum_t P(y_1 < y_v^t < y_2)\Delta t} = \bar{U_s} \tag{6.24}$$

The ratio of average flow and density of time-space block A in Equation 6.24 is comparable to the space-mean speed in Equation 6.19 when the order of summations are changed in both numerator and denominator. The fundamental relation among the average traffic flow, density, and space-mean speed is preserved under the probabilistic formulations of this chapter.

$$\bar{Q} = \bar{K}\bar{U}_s \tag{6.25}$$

6.3 Field-Based Vehicle Trajectory Data

This chapter utilizes the real-world vehicle trajectory data introduced in section 3.3 collected using aerial videography of the traffic stream. The vehicle trajectories are extracted from the video frames recorded in the bird's-eye view from a 500 feet segment of Highway 35 in Austin, Texas. The vehicle trajectory data is collected continuously for two hours during the morning peak hours (7:30 AM to 9:30 AM). A vehicle trajectory is recorded from the first time the vehicle was seen on the study segments and every 0.08 seconds. This dataset is adopted for training the probabilistic trajectory prediction model of [5] and also evaluating the proposed probabilistic macroscopic estimates of this chapter.

The probabilistic trajectory prediction model of [5] takes around three seconds of the past trajectory of a target vehicle as well as three seconds of the past trajectory of all its neighboring vehicles within the distance of 90 feet. The model predicts the probabilistic trajectory of the target vehicle for a fixed prediction period (e.g., 5, 10, 15, or 20 seconds). To train the trajectory prediction model, it is required to have a large number of long enough trajectories to provide the needed input history (i.e., 3 seconds) in addition to the ground truth (e.g., 5, 10, 15, or 20 seconds) for the future prediction. One of the constraints of the trajectory dataset collected in Austin (section 3.3) is that the length of the vehicle trajectories is relatively short (Figure 3.6) due to the limited study segment (500 feet) as a result of restricted flight elevation. Accordingly, a prediction period of nearly five seconds is considered for the models trained and evaluated using the field-based trajectory dataset of section 3.3. Moreover, to complement this dataset, the FHWA Next Generation Simulation Models (NGSIM) dataset is also utilized for evaluating the proposed probabilistic estimates for extended prediction periods (e.g., 10, 15, and 20 seconds).

6.4 Results and Discussion

The vehicle trajectory dataset collected in Austin (section 3.3) is split into training (80%), validation (5%), and testing (15%) datasets. The probabilistic trajectory prediction model of [5] is trained on the training portion of the dataset to predict the trajectory of individual

Table 6.1: Accuracy of the trajectory prediction model on the testing dataset of vehicle trajcetory data collected in Austin, Texas.

Prediction Period	Location RMSE (feet)	Lateral Accuracy(%)	Longitudinal Accuracy (%)
5.12 Seconds	5.15	99.03	89.14

vehicles for a prediction period of 5.12 seconds (64 time steps of 0.08 seconds). The trajectory prediction model takes as input 3.2 seconds (40 time steps of 0.08 seconds) of the track histories (locations over time) of a target vehicle and the vehicles within ± 90 feet in the longitudinal direction and within two adjacent lanes. The spatial configuration of surrounding vehicles of the target vehicle is summarized by a 13×3 matrix/grid. In this matrix, the rows indicate cells of 15 feet in the longitudinal direction, and the columns present the current and adjacent lanes of the target vehicle. The output of the trajectory prediction model is the parameters of the conditional probability distribution of the location of the vehicle and the probability of individual maneuvers (based on Equation 6.1) for every 0.16 seconds for the next 5.12 seconds in the future.

The trained trajectory prediction model is used to predict the future probabilistic trajectory of every vehicle in every frame of a five-minute subsample of the vehicle trajectory dataset collected in Austin (section 3.3). The predicted probabilistic trajectories are used to predict the proposed probabilistic microscopic and macroscopic traffic estimates proposed in this dissertation for the middle 400 feet of the highway segment. The predicted macroscopic states are compared with the true traffic states calculated from the actual trajectories of the dataset.

6.4.1 Microscopic Level: Probabilistic Trajectory Prediction

The input to the maneuver-based model is 3.2 seconds of the past trajectory of a target vehicle as well as 3.2 seconds of the past trajectory of all its neighboring vehicles within the distance of 90 feet. As it is mentioned in the methodology section, the deep learning model of [5] is modified to predict the probability of lateral and longitudinal maneuvers jointly. Considering Equation 6.1, the model is trained to predict Θ, the parameters of the conditional probability distribution of the location of the vehicle and $P(m_i|H)$, the probability of individual maneuvers for time steps of 0.16 second for the next 5.12 seconds in the future.

The probabilistic location of the vehicle is described by a bivariate normal distribution (Equation 6.2) for every maneuver, and the probability of each maneuver, $P(m_i)$, is predicted

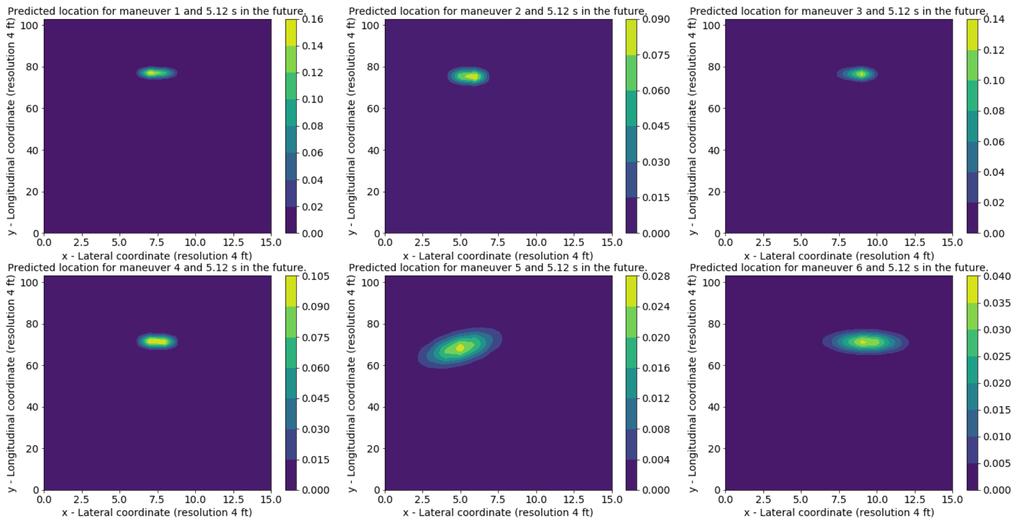


Figure 6.2: Predicted location of a single vehicle for 5.12 seconds in the future and for different maneuvers.

separately. Each maneuver is a combination of lateral and longitudinal movements. Figure 6.2 presents six plots of the predicted probabilistic density function of the location of a vehicle for five seconds in the future and for six different maneuvers. It should be noted that the color of points changes from dark blue to light yellow with the increase in the value of the probability density function of the points. According to this figure, the vehicle's location and its spread differ for various maneuvers.

Figure 6.3 presents four plots of the predicted probabilistic density function of the location of a single vehicle (Equation 6.2) for different prediction periods. The predicted location of the vehicle in these plots is depicted based on the density function of the predicted bivariate normal distribution for the vehicle's location in the future, and the color of points changes from dark blue to light yellow proportion to the density function value. The uncertainty in the vehicle's location increases with the increase in the prediction period. This increase in the uncertainty of the vehicle's location is also evident in the plots of Figure 6.3 with the increase in the spread of the density function along both axes.

Figure 6.4 presents the plots of the bivariate Gaussian mixture models (Equation 6.3) of the location of a single vehicle for different prediction periods. In most cases, the predicted probability of a single maneuver is significantly higher than the probability of other maneuvers. Consequently, the shape of the resulting Gaussian mixture model is closer to the shape of the density function of the maneuver with the highest probability.

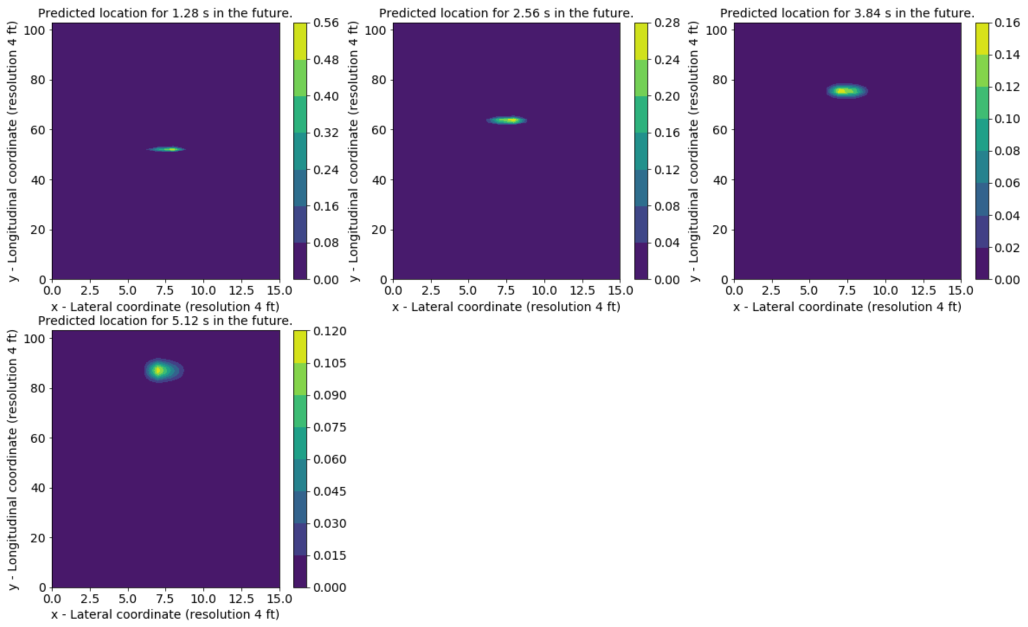


Figure 6.3: Predicted location of a single vehicle over different prediction periods up to 5.12 seconds in the future and for the first maneuver.

Probabilistic Occupancy Map

The space domain is divided into small cells of four by four feet, and the occupancy map of the roadway segment is represented by a matrix with values estimated using Equation 6.4 indicating the probability of that cell being occupied. Figure 6.5 presents the probabilistic occupancy map of the segment of roadway for up to 5.12 seconds in the future. In these plots, the color of each cell changes from dark blue to light yellow proportional to the probability of the cell being occupied. The lighter the color of the cell indicates a higher probability of being occupied. The uncertainty in the vehicle's location increases with the increase in the prediction period. The increase in the uncertainty of the location of the vehicle is also evident in the plots of Figure 6.5 with the spread of cells with low occupancy probability.

Probabilistic Time-Space Diagram

This chapter proposes using a probabilistic time-space matrix representation of the time-space diagram. In this matrix, the time and space domains are divided into smaller cells, and the value of each cell is the expected value of that cell being occupied by a vehicle.

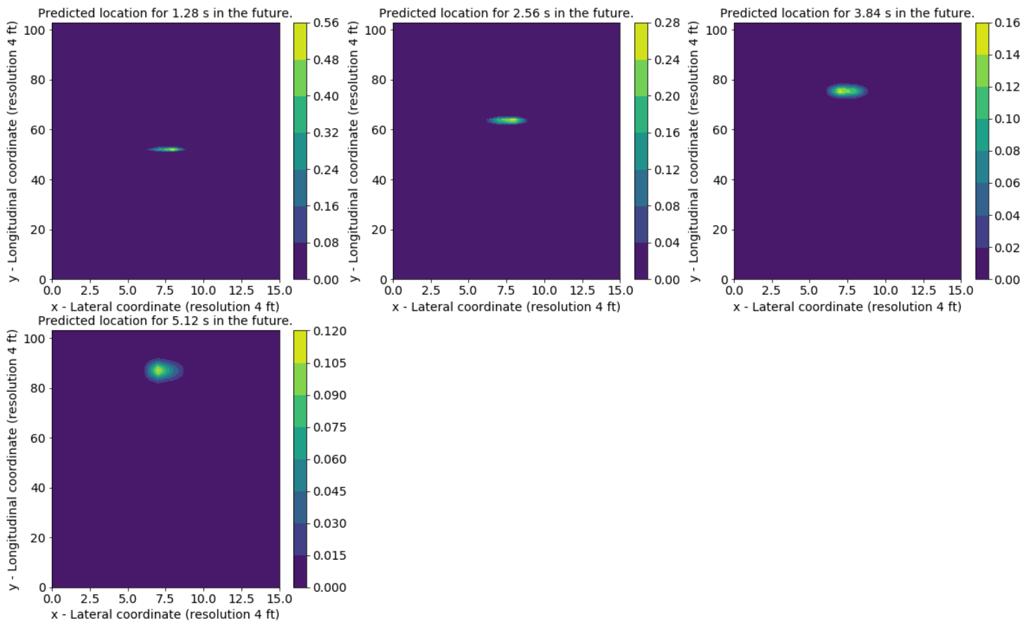


Figure 6.4: Predicted location of a single vehicle for over different prediction periods up to 5.12 seconds in the future considering all maneuvers.

Equation 6.5 estimates the probability of a vehicle passing through a time-space cell based on two probabilities either by crossing the beginning of the space domain of the cell during its time period or remaining within the exact space boundaries similar to its previous cell. The probability of each of these two movements in time and space as well as their summation are depicted in Figure 6.6 for a single vehicle. For the specific vehicle presented in this figure, the probability of the vehicle remaining within the exact space boundaries similar to its previous cell (Figure 6.6b) is relatively low compared to the probability of entering new space domains (Figure 6.6a), particularly at the beginning of the time domain. With the increase in the prediction time, the uncertainty in the location of the vehicle increases, and the probability of the two possible movements from one time-space cell to another gets slightly closer to each other. Figure 6.6c presents the summation of the probability of these two possible movements in time-space. In addition, Figure 6.7 presents predicted probabilistic time-space matrix of all the vehicles on the study area for the next 5.12 seconds, and Figure 6.8 present the corresponding actual time-space matrix of the study area. The predicted probabilistic time-space matrix is comparable to the actual time-space matrix for different lanes and vehicle trajectories.

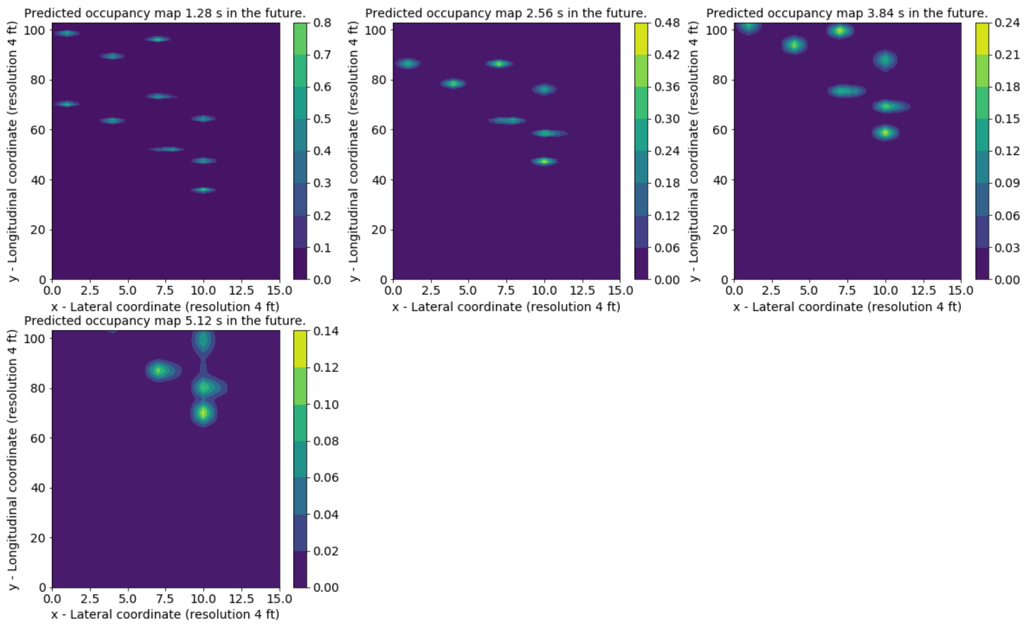


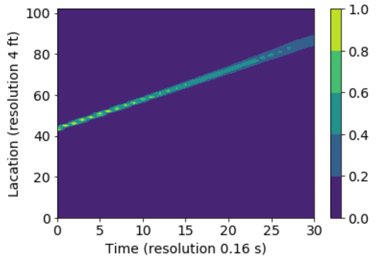
Figure 6.5: Probabilistic occupancy map for five seconds in the future.

6.4.2 Macroscopic Level

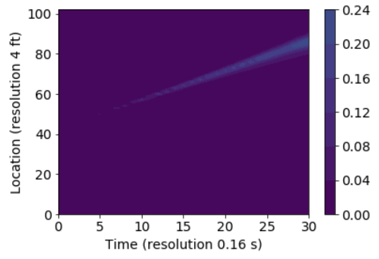
This chapter proposes probabilistic estimates of the traffic flow, density, and space-mean speed considering uncertainties in the location of the vehicles. First, the trajectory prediction model is used to predict the probabilistic trajectory of every vehicle in every 0.8 seconds of the subsample of the Austin dataset. Then, the probabilistic traffic states are estimated for the middle 400 feet of the highway segment, considering the probabilistic trajectories of the vehicles.

In this section, the expected traffic flow is estimated based on Equation 6.21. In this equation, the duration of time $(t_2 - t_1)$ is considered equal to the prediction period (e.g., 5.12 seconds), and the expected traffic flow is estimated for every ten feet of the study segment. The expected traffic density is estimated for the middle 400 feet of the highway segment and for every 0.32 second time step in the future up until the prediction period (e.g., 5.12 seconds) using Equation 6.23. Then, the space-mean speed is estimated considering the expected total distance traveled (Equation 6.16) and expected total time spent (Equation 6.18) by all the vehicles traveling the middle 400 feet during the prediction period (e.g., 5.12 seconds).

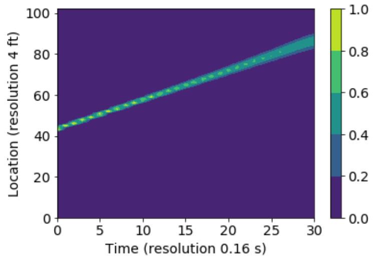
The predicted macroscopic traffic states (i.e., flow, density, and space-mean speed) are



(a) Probability of a vehicle crossing the beginning of space domain

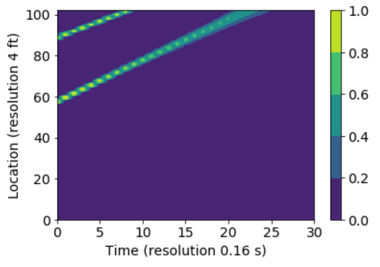


(b) Probability of a vehicle remaining within the exact space boundaries of its previous cell

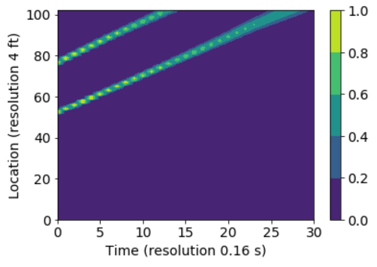


(c) Probability of a vehicle occupying cells of time-space matrix

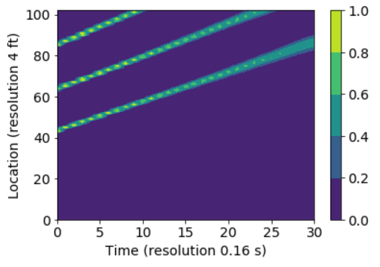
Figure 6.6: Probabilistic time-space matrix for a single vehicle.



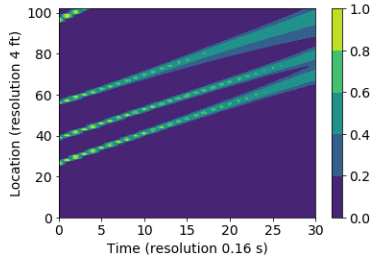
(a) Lane 1



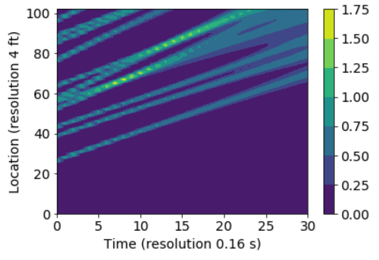
(b) Lane 2



(c) Lane 3

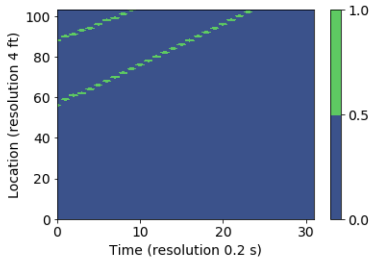


(d) Lane 4

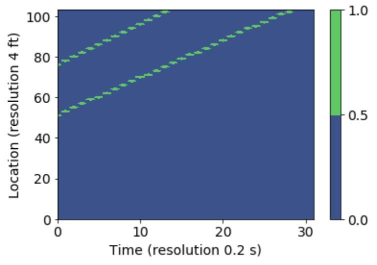


(e) All lanes

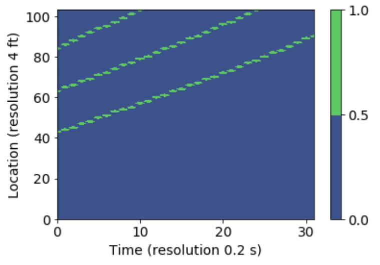
Figure 6.7: Probabilistic time-space matrix for all lanes and individual lanes.



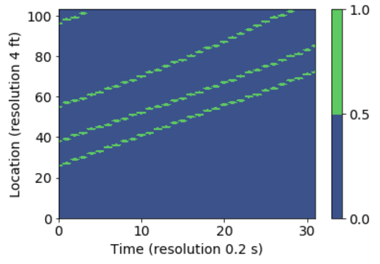
(a) Lane 1



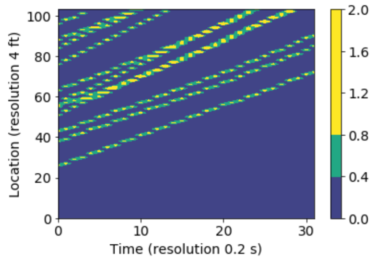
(b) Lane 2



(c) Lane 3



(d) Lane 4



(e) All lanes

Figure 6.8: True time-space matrix for all lanes and individual lanes.

Table 6.2: Mean absolute percentage error (MAPE) of the traffic estimates on vehicle trajectory collected in Austin, Texas.

	Flow	Density	Space-mean Speed
Mean Absolute Percentage Error (MAPE)	5.47	4.59	6.78

compared with the true states calculated from the actual trajectories of the Austin dataset. This chapter adopts the mean absolute percentage error (MAPE) to evaluate the performance of the proposed methodology. Table 6.2 presents the mean absolute-percentage error (MAPE) of the predicted flow, density, and space-mean speed based on the following equations:

$$MAPE_Q = \frac{100}{n} \sum_{i=1}^{n} \frac{|E[Q_i] - Q_i|}{|Q_i|} \tag{6.26}$$

$$MAPE_K = \frac{100}{m} \sum_{j=1}^{m} \frac{|E[K_j] - K_j|}{|K_j|} \tag{6.27}$$

$$MAPE_{U_s} = \frac{100}{l} \sum_{k=1}^{l} \frac{|\bar{U}_{s,k} - U_{s,k}|}{|U_{s,k}|} \tag{6.28}$$

Table 6.3: Accuracy of the trajectory prediction model on the testing dataset of the NGSIM vehicle trajectory data.

Prediction Period	Location RMSE (feet)	Lateral Accuracy(%)	Longitudinal Accuracy (%)
5 Seconds	5.94	97.99	91.72
10 Seconds	17.06	97.99	87.29
15 Seconds	34.41	97.98	84.37
20 Seconds	54.72	97.95	82.82

6.4.3 Evaluation on the NGSIM Dataset for Longer Prediction Periods

The NGSIM dataset is also utilized for evaluating the proposed probabilistic traffic estimates of this chapter for prediction periods longer than 5.12 seconds. The complete NGSIM dataset is split into training (80%), validation (5%), and testing (15%) datasets. The probabilistic trajectory prediction model of [5] is trained on the training portion of the dataset to predict the trajectory of individual vehicles for different prediction periods, including 5, 10, 15, and 20 seconds. It should be noted that the location of the vehicles in the NGSIM dataset is recorded every 0.1 seconds; as a result, the trajectory prediction model is trained to take as input 3 seconds (30 time steps) of the trajectory of the target vehicle and all its neighboring vehicles within the distance of 90 feet. The model predicts the probabilistic trajectory of the target vehicle for a fixed prediction period (e.g., 5, 10, 15, or 20 seconds). Table 6.3 presents the accuracy of four models trained for different prediction periods. According to this table, the model's accuracy in predicting the vehicle's maneuver and location decays with the increase in the prediction period.

The trajectory prediction models trained on the NGSIM dataset are used to predict the probabilistic trajectory of every vehicle in every 5 seconds of a subsample of the NGSIM dataset. The subsample is five minutes of the first set of trajectory data of highway 101. Then, the probabilistic traffic states are estimated for the middle 1000 feet of the highway segment, considering the probabilistic trajectories of the vehicles.

Figure 6.9 presents the mean absolute percentage error (MAPE) of the traffic estimates for different prediction periods on the subsample of the NGSIM dataset. According to this plot, the MAPE increases for all the three traffic state estimates with the increase in the prediction period. The decay in the accuracy of the prediction is expected since the uncertainty in the probabilistic trajectories increases. The MAPE of flow increases from 5.50 to 7.23 percent, with an increase in the prediction period from 5 to 20 seconds. Moreover, the MAPE of the space-mean speed increases from 3.87 to 10.06 percent, with an increase in the prediction period from 5 to 20 seconds. The MAPE reported for density in Figure 6.9 is the average of the MAPE over all the time-steps of the prediction period and changes from 1.10 to 4.71

percent from 5 to 20 seconds, respectively.

Chapter 4 also proposes a deep learning traffic state prediction model based on a convolutional neural network (CNN) using the observed time-space diagram of the roadway as the input to the model to capture the interaction among the vehicles when predicting the traffic state. The performance of the CNN-based model in Chapter 4 in terms of MAPE on predicting the flow and density for the next 20 seconds on the NGSIM dataset is 15.43 and 13.81, respectively. Moreover, Chapter 4 also compares the performance of the proposed CNN-based model with other non-parametric models, including multilayer perceptron (MLP), support vector regression (SVR), and autoregressive integrated moving average (ARIMA). The ARIMA model trained on the NGSIM dataset performed relatively better than the other non-parametric models with MAPE of 10.55 and 10.04 for predicting the average flow and density of the segment in the next 20 seconds. Even though the performance of the probabilistic estimates of the flow and density proposed in this chapter top the performance of the models evaluated in Chapter 4, it should be noted that the performances of the probabilistic traffic state estimates are directly dependent on the performance of the probabilistic trajectory prediction model at the individual level. Therefore, the performances reported in this chapter are just proof of concept investigating the opportunity of probabilistic estimates of the traffic state.

Figure 6.10 presents the MAPE for density estimate over time steps for different trajectory prediction models trained for different prediction periods. The general trend of the error in density estimate is also increasing with the increase in the time steps, and this is due to the decay in the accuracy of the trajectory prediction with the increase in the prediction period. The MAPE in density gets to 13.45 percent when predicting the density of the segment at 20 seconds in the future. According to Figure 6.10, the MAPE in density estimates for different trajectory prediction models (trained for different prediction periods) performs relatively similarly on their overlapping steps, specifically for models with prediction periods of 5, 10, and 15. However, the model with a prediction period of 20 seconds performs slightly better than the other models, indicating that the model trained to predict for a more extended period learns a better generalization. It should be noted that all the four trajectory prediction models are trained with five training epochs and random initialization of the parameters. The random initialization could also result in slightly different performances among the models.

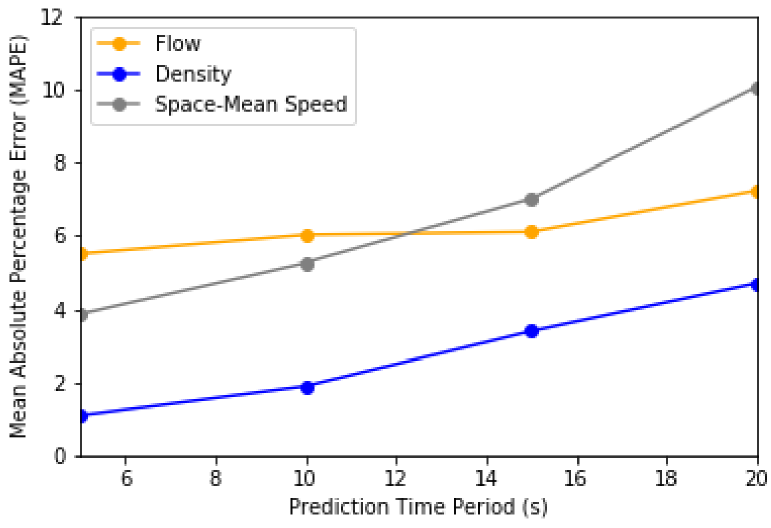


Figure 6.9: Mean absolute percentage error (MAPE) of the traffic estimates for different prediction period.

6.5 Chapter Summary and Conclusions

This chapter proposes probabilistic estimates of the traffic state from probabilistic predictions of vehicle trajectories. At the microscopic level, this chapter develops a probability-based version of the time-space diagram of the vehicles. In this representation, the time and space are divided into smaller cells, where the value of each cell is the expected value of that cell being occupied by a vehicle. With the increase in the prediction period, the uncertainty in the location of the vehicles increases. As a result, more cells are expected to have values larger than zero further along the end of the prediction horizon but smaller than the cell values at the start of the prediction. The predicted probabilistic time-space matrix is comparable to the actual time-space matrix for most lanes and vehicle trajectories. There are instances, however, where predicted and actual future trajectories diverge. In most cases, the discrepancies are due to lane-changing happening in the future. This shortcoming is due to a limitation in the trajectory prediction model used in this chapter that does not capture future interactions or lane-changing maneuvers.

This chapter proposes probabilistic estimates of flow, density, and space-mean speed at the macroscopic level. The expected flow is defined based on the expected value of the number of vehicles crossing a specific roadway point over a predetermined period. The expected number of vehicles crossing a particular roadway point is estimated depending on the probability of individual vehicles passing that point during the fixed period. The

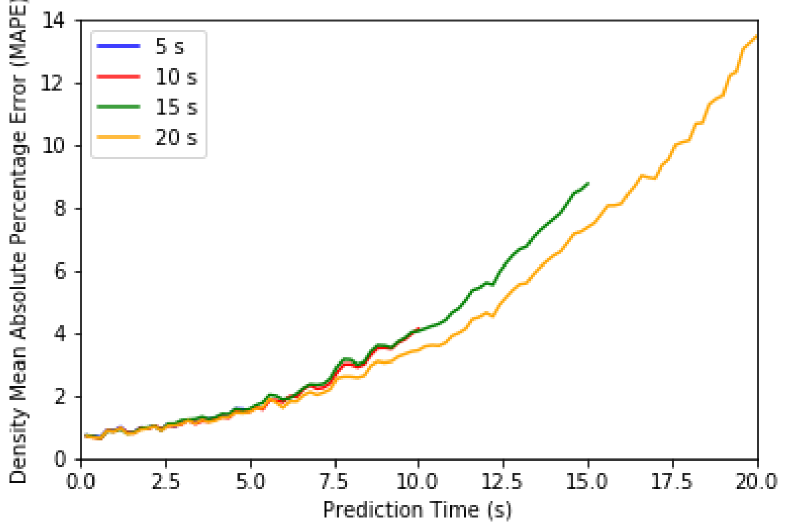


Figure 6.10: Mean absolute percentage error (MAPE) of the density estimates for different prediction period.

expected density is defined based on the expected number of vehicles occupying a specified segment of the roadway at a given time. The expected number of vehicles occupying a fixed segment of the roadway is estimated considering the probability of the individual vehicles being on the specified segment at that given time. The probabilistic estimate of the space-mean speed is defined based on the expected total distance traveled by all the vehicles going through a segment of roadway over a given period divided by the expected total time spent by all the vehicles going through that time-space block. The expected time spent by individual vehicles can be estimated considering the probability of the vehicle being within the space boundaries of the roadway segment at every point of time within the given period. The expected distance traveled by every vehicle can be estimated considering the probability of the vehicle going through every point of the segment of the roadway during the given period. Moreover, this chapter proves that the fundamental relation among the average traffic flow, density, and space-mean speed is preserved under the probabilistic formulations of this chapter. In addition, the mean absolute percentage of error (MAPE) for each of the probabilistic estimates at the macroscopic level is also estimated for a five-minute subsample of both the real-world vehicle trajectory data collected in Austin, Texas (Section 3.3) and the NGSIM dataset for different prediction periods.

With an increase in prediction period time, the MAPE increases for all three traffic state estimates. As the uncertainty in probabilistic trajectories increases, the accuracy of the prediction decays. On the NGSIM dataset, the MAPE of flow rises from 5.50 to 7.23 percent, with increasing the prediction period from 5 to 20 seconds. Moreover, the MAPE of the space-mean speed rises from 3.87 to 10.06 percent, with an increase in the prediction period from 5 to 20 seconds. The MAPE reported for density is the average of the MAPE over all the time-steps of the prediction period and changes from 1.10 to 4.71 percent from 5 to 20 seconds, respectively. It should be noted that the performances of the probabilistic traffic state estimates proposed in this chapter are directly dependent on the performance of the probabilistic trajectory prediction model at the individual level. Therefore, the performances reported in this chapter are just proof of concept investigating the opportunity of probabilistic estimates of the traffic state.

CHAPTER 7

SUMMARY

Traditional traffic monitoring and prediction methods rely on stationary vehicle detection devices (e.g., inductive loops, piezoelectric sensors, cameras, and radars) to provide the necessary information about traffic conditions. Thus, the data gathered along the road segment covered only a concise segment and was mainly composed of aggregated measures (e.g., flow, occupancy, and average speed). Due to this limitation, such data cannot accurately reflect the dynamics of traffic flows through the roadway, such as capturing the shockwave formation and propagation. Adding multiple sensors to the roadway may mitigate this challenge to some extent, but many key traffic flow features still cannot be captured due to location-specific measurements.

When predicting traffic at the aggregated level, the majority of existing data-driven prediction models use aggregated traffic measurements as input and output in the prediction process [7]. Unfortunately, the dynamic nature of traffic flow and the limitations of conventional data sources have historically made it difficult to accurately predict the state of traffic. Furthermore, with the increase in traffic flow and density, unexpected driving behaviors can have a more significant impact on the traffic state [8]. Accordingly, capturing the interaction among individual vehicles can potentially lead to more accurate traffic prediction. In recent years, advances in vehicular communications have led to exceptional communications and data exchange opportunities between vehicles and their surroundings [6]. Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications allow for continuous monitoring of the driving environment over time and space at the individual vehicle level. Furthermore, the introduction of connected automated vehicles provides an opportunity to monitor the interaction among individual vehicles.

The driving behavior and the vehicles' interactions can be best captured by the time-space diagram of the traffic stream. The time-space diagram contains all the necessary information for estimating the microscopic and macroscopic characteristics of the traffic stream. Despite time-space diagrams being at the core of many traffic flow theory innovations, there was no scalable approach to generate this diagram from real-world data until recently. With the recent advances in connected vehicle technology, it has become possible to collect a new data source at the individual vehicle level. Therefore, the time-space diagram of the vehicles can be constructed from the data that connected vehicles share in a connected environment. This dissertation assumes a connected vehicle environment where each vehicle shares its location over time. The time and space domains can be divided into smaller cells in such a connected environment. The trajectories of vehicles are constructed based on their presence in time and space cells, similar to the analogy of image pixels. The time-space diagram can be stored in a binary matrix with rows for the space dimension and columns for the time dimension. In this matrix, a cell value of one indicates the presence of a vehicle at the respective time and location, and a value of zero is used for empty cells.

One of the main contributions of this dissertation is proposing traffic prediction methodologies that can directly utilize the time-space diagram in the prediction process. Chapter 4 proposes a convolutional neural network (CNN) to predict the traffic state in the form of flow and density using the time-space diagram as input to the model. Using a CNN model allows capturing the features embedded in the time-space diagram and accounting for the disturbances such as shockwaves in the traffic stream. The traffic shockwaves are the boundary between the different traffic states, which are evident in the time-space diagram, and cannot be easily captured in aggregated features. The location independence aspect of the convolutional layers makes them a practical choice to encode the time-space matrix since the traffic disturbances or shockwaves can occur at any point in time and space. The proposed CNN model of Chapter 4 in comparison with three non-parametric models, i.e., support vector regression (SVR), multilayer perceptron (MLP), and Autoregressive Integrated Moving Average (ARIM), show a better generalization in predicting the traffic state in different regions of the fundamental diagram. Furthermore, evaluating the CNN-based model trained based on the simulated data against a real-world dataset (NGSIM US-101) revealed its generalization capability compared to the other non-parametric approaches.

Chapter 5 proposes a convolution encoder-decoder model to predict the formation and propagation of traffic shockwaves over time and space using the time-space diagram as an input to the model. The encoding convolutional layers extract the main features and abstract the input time-space diagram, while the deconvolution layers decode the abstract input and predict the shockwave propagation over time and space. Training the model to output a binary time-space matrix is challenging, and breaking the binary constraint improves the training process. The model's output is an averaged version of the time-space matrix by replacing every cell in the time-space matrix with the average of itself and its neighbors up to 50 feet and 0.5 seconds on each side. The model's output is comparable to a density time-space matrix, with every cell representing the average density of a time-space block of 100 feet and 1 second. The results of analysis on both simulated and real-world (NGSIM US-

101) datasets indicate that the proposed model can predict the dissemination, propagation, forward and backward movement of the traffic shockwaves.

The CNN-based models proposed in Chapters 4 and 5 offer two key innovations: (1) they provide the opportunity to directly utilize the time-space diagram as the input without the need for any aggregation or abstraction; (2) the convolutional layers provide the opportunity to capture and learn the features embedded in the time-space diagram required for an accurate prediction, such as traffic flow dynamics, interaction between the vehicles, and traffic shockwaves.

Chapter 6 proposes a methodology to predict the future traffic state at the aggregated level based on probabilistic predictions at the individual vehicle level. Predicting the driving environment at the individual vehicle level (i.e., vehicle trajectory) is challenging due to the dynamic nature of the driving environment, the complex interactions among vehicles, and differences in driving behavior among different drivers. Drivers respond to their surrounding environment and interact with other vehicles through different maneuvers (e.g., acceleration, deceleration, and lane changing). In many driving scenarios, more than one maneuver is feasible when predicting the trajectory of an individual vehicle. Consequently, the accuracy of prediction at the individual level decays with the increase in the prediction horizon due to the uncertainty in the drivers' choice of maneuvers and the possibility of various configurations and outcomes. As a result, a probabilistic approach can be adopted when predicting the future movement of an individual vehicle conditioned on different maneuvers. The key contribution of Chapter 6 is to introduce a methodology to convert such probabilistic predictions to aggregated traffic state prediction (i.e., flow, density, and space-mean speed).

At the microscopic level, Chapter 6 develops a probability-based version of the time-space diagram of the vehicles. In this representation, the time and space domains are divided into smaller cells, where the value of each cell is the expected value of that cell being occupied by a vehicle. At the macroscopic level, this chapter proposes probabilistic estimates of flow, density, and space-mean speed. The expected flow is estimated based on the expected number of vehicles crossing a particular roadway point during a fixed period considering the probability of individual vehicles passing that point. The expected density is estimated based on the expected number of vehicles occupying a specified segment of the roadway at a given time, considering the probability of the individual vehicles on the specified segment at the given time. The probabilistic estimate of the space-mean speed is defined based on the expected total distance traveled by all the vehicles going through a segment of roadway over a given period divided by the expected total time spent by all the vehicles going through that time-space block. Chapter 6 also shows that the fundamental relation among the average traffic flow, density, and space-mean speed is still preserved under the proposed novel traffic state prediction approach.

The proposed probabilistic estimates of the traffic state in Chapter 6 are evaluated using real-world traffic trajectory data collected in Austin, Texas, and the NGSIM dataset for different prediction periods. The performance of the probabilistic estimates of the flow and density proposed in Chapter 6 top the performance of the models evaluated in Chapter 4. It should be noted that the performances of the probabilistic traffic state estimates are directly dependent on the performance of the probabilistic trajectory prediction model at the individual level, and with an increase in the prediction period, the accuracy of traffic state estimates decay. The key advantages of the approach presented in Chapter 6 over directly predicting the traffic state based on aggregated traffic data are: (1) the ability to capture the impacts of interactions among vehicles on traffic flow dynamics to increase the accuracy of the predictions; (2) capturing the uncertainties in individual vehicle's maneuvers and the possibility of various configurations and outcomes when predicting the traffic state at the aggregated level.

This dissertation adopts data-driven methodologies for predicting upcoming driving environments. These methodologies require accurate and reliable data for the training process. As a result, this study utilizes both simulation-based and real-world trajectory datasets (Chapter 3) to train and evaluate the traffic state prediction approaches proposed in this dissertation. The microscopic traffic simulator is used to create an extensive dataset that includes different levels of traffic state (from free-flow to fully congested) at different posted speeds for the training of the models that can generalize well. The microscopic simulator adopts the Intelligent Driver Model (IDM) [59] as its car-following logic, and MOBIL [60] as its lane-changing logic. Additionally, Chapter 3 introduces a robust, scalable, and cost-effective methodology for real-world vehicle trajectory data collection through aerial imagery using aerial videography. The proposed aerial data collection is used to collect real-world trajectory data on Interstate 35 near Austin, TX, which is used to evaluate the probabilistic approach of Chapter 6.

7.1 Limitations of the Current Work and Future Research Directions

The traffic state prediction methodologies proposed in this dissertation are based on the assumption of complete knowledge of the observed trajectory of all the vehicles in the study area considering a fully connected environment or from the sensory data of connected and automated vehicles. In practice, the future traffic stream could be a mix of conventional, connected, and connected automated vehicles. While it is feasible to capture and monitor the majority of the vehicles in the traffic stream based on the sensory data from the connected and automated vehicles when their market penetration rate is above a minimum level, Talebpour et al. [105] showed that due to signal interference, many information packets would not reach their destinations, even in a fully connected driving environment. Accordingly, it is critical to investigate and improve the proposed models in this dissertation to predict the traffic state based on partial or incomplete data from the traffic stream. This dissertation introduced deep learning methodologies based on convolutional neural networks (CNN) to directly use the time-space diagram in the traffic state prediction. Following the development of the models proposed in this dissertation, some recent studies such as [106] and [107] have also proposed methodologies to reconstruct the current time-space diagram of the traffic stream based on partial or limited observation of the vehicle trajectories. Benkraouda et al. [106] adopt a convolutional encoder-decoder architecture comparable to the model proposed in Chapter 5, and Zhang et al. [107] adopt Generative Adversarial Networks (GANs) to reconstruct the existing time-space diagram of the traffic stream. Similar approaches could also be investigated to predict the future traffic state considering partial or limited observation of the vehicle trajectories for lower market penetrations of the connected and automated vehicles in the traffic stream. In addition, more complex network architectures such as a combination of convolutional and recurrent neural networks (CNN-RNN) as well as graph convolutional neural networks (GCN) can be adopted to include additional spatiotemporal dependencies of the traffic state at the network level.

Moreover, the prediction accuracy usually decays with the increase in the prediction horizon due to the uncertainty in drivers' behavior and an increase in the possibility of various configurations and outcomes. The prediction horizon is dependent on the planning horizon, and in general, a longer prediction horizon is preferred for reactive congestion mitigation methods. One of the advantages of the probabilistic traffic state prediction methodology proposed in Chapter 6 is capturing the uncertainties in individual vehicle's maneuvers and the possibility of various configurations and outcomes when predicting the traffic state over time. In future research studies, it would be valuable to investigate the trade-off between the prediction accuracy and the prediction horizon, considering the effectiveness of the congestion mitigation methodologies.

REFERENCES

[1] M. Papageorgiou, H. Hadj-Salem, and F. Middelham, "Alinea local ramp metering: Summary of field results," *Transportation research record*, vol. 1603, no. 1, pp. 90–98, 1997.

[2] A. Talebpour, H. S. Mahmassani, and S. H. Hamdar, "Speed harmonization: Evaluation of effectiveness under congested conditions," *Transportation research record*, vol. 2391, no. 1, pp. 69–79, 2013.

[3] A. Elfar, A. Talebpour, and H. S. Mahmassani, "Predictive speed harmonization in a connected environment: A machine learning approach," Tech. Rep., 2019.

[4] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH journal*, vol. 1, no. 1, pp. 1–14, 2014.

[5] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1468–1476.

[6] M. A. Rahim, M. A. Rahman, M. M. Rahman, A. T. Asyhari, M. Z. A. Bhuiyan, and D. Ramasamy, "Evolution of iot-enabled connectivity and applications in automotive industry: A review," *Vehicular Communications*, vol. 27, p. 100285, 2021.

[7] A. Elfar, C. Xavier, A. Talebpour, and H. S. Mahmassani, "Traffic shockwave detection in a connected environment using the speed distribution of individual vehicles," Tech. Rep., 2018.

[8] S. J. L. Ossen, "Longitudinal driving behavior: theory and empirics," 2008.

[9] T. Seo, A. M. Bayen, T. Kusakabe, and Y. Asakura, "Traffic state estimation on highway: A comprehensive survey," *Annual Reviews in Control*, vol. 43, pp. 128–151, 2017.

[10] J. Van Lint and C. Van Hinsbergen, "Short-term traffic and travel time prediction models," *Artificial Intelligence Applications to Critical Transportation Issues*, vol. 22, no. 1, pp. 22–41, 2012.

[11] G. Huisken and E. C. van Berkum, "A comparative analysis of short-range travel time prediction methods," in *82nd Annual Meeting of the Transportation Research Board, Washington, DC*, 2003.

[12] R. Eglese, W. Maden, and A. Slater, "A road timetabletm to aid vehicle routing and scheduling," *Computers & operations research*, vol. 33, no. 12, pp. 3508–3519, 2006.

[13] Y. Wang, M. Papageorgiou, A. Messmer, P. Coppola, A. Tzimitsi, and A. Nuzzolo, "An adaptive freeway traffic state estimator," *Automatica*, vol. 45, no. 1, pp. 10–24, 2009.

[14] Y. Sun and D. B. Work, "A distributed local kalman consensus filter for traffic estimation," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on.* IEEE, 2014, pp. 6484–6491.

[15] Z. Zheng and D. Su, "Traffic state estimation through compressed sensing and markov random field," *Transportation Research Part B: Methodological*, vol. 91, pp. 525–554, 2016.

[16] R. Wang, D. B. Work, and R. Sowers, "Multiple model particle filter for traffic estimation and incident detection." *IEEE Trans. Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3461–3470, 2016.

[17] T. Seo, T. T. Tchrakian, S. Zhuk, and A. M. Bayen, "Filter comparison for estimation on discretized pdes modeling traffic: Ensemble kalman filter and minimax filter," in *Decision and Control (CDC), 2016 IEEE 55th Conference on.* IEEE, 2016, pp. 3979–3984.

[18] H. S. Mahmassani, J. Dong, and J. Kim, "Dynasmart-p 1.5 user's guide and programmer's guide," 2009.

[19] M. R. Wilby, J. J. V. Díaz, A. B. Rodríguez Gonz´ lez, and M. Á. Sotelo, "Lightweight occupancy estimation on freeways using extended floating car data," *Journal of Intelligent Transportation Systems*, vol. 18, no. 2, pp. 149–163, 2014.

[20] H. B. Gogineni, E. L. Lydia, and N. Supriya, "An intelligent vehicular traffic flow prediction model using whale optimization with multiple linear regression," *Computational Intelligence for Sustainable Transportation and Mobility*, vol. 1, pp. 1–15, 2021.

[21] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang et al., "Traffic flow prediction with big data: A deep learning approach." *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

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

[23] 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.

[24] K. Lee, M. Eo, E. Jung, Y. Yoon, and W. Rhee, "Short-term traffic prediction with deep neural networks: A survey," *IEEE Access*, vol. 9, pp. 54 739–54 756, 2021.

[25] H. Su, L. Zhang, and S. Yu, "Short-term traffic flow prediction based on incremental support vector regression," in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, vol. 1. IEEE, 2007, pp. 640–645.

[26] 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.

[27] S. Angayarkanni, R. Sivakumar, and Y. Ramana Rao, "Hybrid grey wolf: Bald eagle search optimized support vector regression for traffic flow forecasting," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 1293–1304, 2021.

[28] C. Chen, J. Hu, Q. Meng, and Y. Zhang, "Short-time traffic flow prediction with arima-garch model," in *Intelligent vehicles symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 607–612.

[29] S. V. Kumar and L. Vanajakshi, "Short-term traffic flow prediction using seasonal arima model with limited input data," *European Transport Research Review*, vol. 7, no. 3, p. 21, 2015.

[30] X. Lin and Y. Huang, "Short-term high-speed traffic flow prediction based on arima-garch-m model," *Wireless Personal Communications*, vol. 117, no. 4, pp. 3421–3430, 2021.

[31] M. Khajeh Hosseini, A. Talebpour, and S. Shakkottai, "Privacy risk of connected vehicles in relation to vehicle tracking when transmitting basic safety message type 1 data," *Transportation research record*, vol. 2673, no. 12, pp. 636–643, 2019.

[32] T. Batz, K. Watson, and J. Beyerer, "Recognition of dangerous situations within a cooperative group of vehicles," in *2009 IEEE Intelligent Vehicles Symposium*. IEEE, 2009, pp. 907–912.

[33] M. T. Abbas, M. A. Jibran, M. Afaq, and W.-C. Song, "An adaptive approach to vehicle trajectory prediction using multimodel kalman filter," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 5, p. e3734, 2020.

[34] A. Broadhurst, S. Baker, and T. Kanade, "Monte carlo road safety reasoning," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* IEEE, 2005, pp. 319–324.

[35] S. A. Goli, B. H. Far, and A. O. Fapojuwo, "Vehicle trajectory prediction with gaussian process regression in connected vehicle environment," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 550–555.

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

[37] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, "A bayesian nonparametric approach to modeling motion patterns," *Autonomous Robots*, vol. 31, no. 4, p. 383, 2011.

[38] Y. Dou, F. Yan, and D. Feng, "Lane changing prediction at highway lane drops using support vector machine and artificial neural network classifiers," in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM).* IEEE, 2016, pp. 901–906.

[39] S. Yoon and D. Kum, "The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles," in *2016 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2016, pp. 1307–1312.

[40] S. Klingelschmitt, M. Platho, H.-M. Groß, V. Willert, and J. Eggert, "Combining behavior and situation information for reliably estimating multiple intentions," in *2014 IEEE Intelligent Vehicles Symposium Proceedings.* IEEE, 2014, pp. 388–393.

[41] A. Khosroshahi, "Learning, classification and prediction of maneuvers of surround vehicles at intersections using lstms," Ph.D. dissertation, UC San Diego, 2017.

[42] A. Lawitzky, D. Althoff, C. F. Passenberg, G. Tanzmeister, D. Wollherr, and M. Buss, "Interactive scene prediction for automotive applications," in *2013 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2013, pp. 1028–1033.

[43] M. Liebner, M. Baumann, F. Klanner, and C. Stiller, "Driver intent inference at urban intersections using the intelligent driver model," in *2012 IEEE Intelligent Vehicles Symposium.* IEEE, 2012, pp. 1162–1167.

[44] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, "Deep learning-based vehicle behavior prediction for autonomous driving applications: A review," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[45] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in *2018 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2018, pp. 1179–1184.

[46] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6120–6127.

[47] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA).* IEEE, 2019, pp. 2090–2096.

[48] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[49] P. G. Michalopoulos, "Vehicle detection video through image processing: the autoscope system," *IEEE Transactions on vehicular technology*, vol. 40, no. 1, pp. 21–29, 1991.

[50] J. Zhou, D. Gao, and D. Zhang, "Moving vehicle detection for automatic traffic monitoring," *IEEE transactions on vehicular technology*, vol. 56, no. 1, pp. 51–59, 2007.

[51] FHWA, "U.s. federal highway administration. next generation simulation (ngsim)," *https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm*, 2006.

[52] J. M. Hankey, M. A. Perez, and J. A. McClafferty, "Description of the shrp 2 naturalistic database and the crash, near-crash, and baseline data sets," Virginia Tech Transportation Institute, Tech. Rep., 2016.

[53] D. Zhao, Y. Guo, and Y. J. Jia, "Trafficnet: An open naturalistic driving scenario library," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–8.

[54] B. Coifman and L. Li, "A critical evaluation of the next generation simulation (ngsim) vehicle trajectory dataset," *Transportation Research Part B: Methodological*, vol. 105, pp. 362–377, 2017.

[55] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *2018 IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[56] E. Barmpounakis and N. Geroliminis, "On the new era of urban traffic monitoring with massive drone data: The pneuma large-scale field experiment," *Transportation Research Part C: Emerging Technologies*, vol. 111, pp. 50–71, 2020.

[57] A. Bhutani and P. Bhardwaj, "Automotive camera market share 2019-2025: Global industry report," *https://www.gminsights.com/industry-analysis/automotive-camera-market*, May 2019.

[58] M. Makridis, K. Mattas, A. Anesiadou, and B. Ciuffo, "Openacc. an open database of car-following experiments to study the properties of commercial acc systems," *Transportation research part C: emerging technologies*, vol. 125, p. 103047, 2021.

[59] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[60] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[61] M. Schreuder, S. Hoogendoorn, H. Van Zulyen, B. Gorte, and G. Vosselman, "Traffic data collection from aerial imagery," in *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, vol. 1. IEEE, 2003, pp. 779–784.

[62] T. Zhao and R. Nevatia, “Car detection in low resolution aerial images,” *Image and Vision Computing*, vol. 21, no. 8, pp. 693–703, 2003.

[63] H. Harzallah, F. Jurie, and C. Schmid, “Combining efficient object localization and image classification,” in *2009 IEEE 12th international conference on computer vision.* IEEE, 2009, pp. 237–244.

[64] 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.

[65] M. Radovic, O. Adarkwa, and Q. Wang, “Object recognition in aerial images using convolutional neural networks,” *Journal of Imaging*, vol. 3, no. 2, p. 21, 2017.

[66] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[67] K. G. Derpanis, “The harris corner detector,” *York University*, pp. 2–3, 2004.

[68] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[69] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European conference on computer vision.* Springer, 2006, pp. 404–417.

[70] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, “Orb: An efficient alternative to sift or surf.” in *ICCV*, vol. 11, no. 1. Citeseer, 2011, p. 2.

[71] E. Karami, S. Prasad, and M. Shehata, “Image matching using sift, surf, brief and orb: performance comparison for distorted images,” *arXiv preprint arXiv:1710.02726*, 2017.

[72] M. Muja and D. Lowe, “Fast library for approximate nearest neighbors (flann),”,” *git://github. com/mariusmuja/flann. git. url: http://www. cs. ubc. ca/research/flann*, 2013.

[73] K. G. Derpanis, “Overview of the ransac algorithm,” *Image Rochester NY*, vol. 4, no. 1, pp. 2–3, 2010.

[74] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, “Object detection with deep learning: A review,” *IEEE transactions on neural networks and learning systems*, 2019.

[75] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[76] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[77] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[78] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision.* Springer, 2016, pp. 21–37.

[79] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[80] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[81] H. Gaiser and et al., “fizyr/keras-retinanet 0.5.1,” June 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3250670

[82] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision.* Springer, 2014, pp. 740–755.

[83] A. Dutta and A. Zisserman, “The VIA annotation software for images, audio and video,” in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM ’19. New York, NY, USA: ACM, 2019. [Online]. Available: https://doi.org/10.1145/3343031.3350535

[84] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *2017 IEEE international conference on image processing (ICIP).* IEEE, 2017, pp. 3645–3649.

[85] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural networks*, vol. 61, pp. 85–117, 2015.

[86] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[87] L. C. Edie, “Car-following and steady-state theory for noncongested traffic,” *Operations research*, vol. 9, no. 1, pp. 66–76, 1961.

[88] S. Dabiri and K. Heaslip, “Inferring transportation modes from gps trajectories using a convolutional neural network,” *Transportation research part C: emerging technologies*, vol. 86, pp. 360–371, 2018.

[89] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[90] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[91] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[92] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[93] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks–ICANN 2010*. Springer, 2010, pp. 92–101.

[94] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[95] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[96] A. Abdiansah and R. Wardoyo, "Time complexity analysis of support vector machines (svm) in libsvm," *International Journal Computer and Application*, 2015.

[97] FHWA, "Next generation simulation: Us101 freeway dataset," 2007.

[98] SAE, "J2735 dedicated short range communications (dsrc) message set dictionary," *Society of Automotive Engineers, DSRC Tech Committee*, 2016.

[99] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[100] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[101] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.

[102] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Advances in neural information processing systems*, 2016, pp. 2802–2810.

[103] F. L. Hall, "Traffic stream characteristics," *Traffic Flow Theory. US Federal Highway Administration*, vol. 36, 1996.

[104] L. C. Edie, *Discussion of traffic stream measurements and definitions.* Port of New York Authority, 1963.

[105] A. Talebpour, H. S. Mahmassani, and F. E. Bustamante, “Modeling driver behavior in a connected environment: Integrated microscopic simulation of traffic and mobile wireless telecommunication systems,” *Transportation Research Record*, vol. 2560, no. 1, pp. 75–86, 2016.

[106] O. Benkraouda, B. T. Thodi, H. Yeo, M. Menendez, and S. E. Jabari, “Traffic data imputation using deep convolutional neural networks,” *IEEE Access*, vol. 8, pp. 104 740–104 752, 2020.

[107] K. Zhang, X. Feng, N. Jia, L. Zhao, and Z. He, “Tsr-gan: Generative adversarial networks for traffic state reconstruction with time space diagrams,” *Physica A: Statistical Mechanics and its Applications*, vol. 591, p. 126788, 2022.