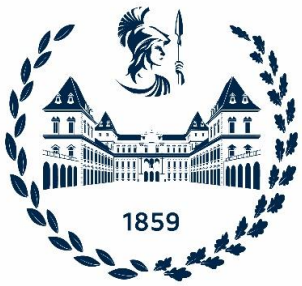


Politecnico di Torino

POLITECNICO DI TORINO

Master's Degree in Communications and Computer Networks Engineering

Master's Degree Thesis

AI/ML and Local Dynamic Map predictions

Supervisors:

Prof. CASETTI CLAUDIO ETTORE

Dr. RAVIGLIONE FRANCESCO

Dr. PULIGHEDDU CORRADO

Dr. VALLERO GRETA

Candidate:

Weiyi Qing

S265412

October 2022

Abstract

Traffic accidents cost many lives every year and can cause significant economic losses. The vast majority of accidents are caused by human drivers, especially when they are trying to change lanes or overtake. Indeed, during these maneuvers, the probability of traffic accidents is higher. Yet the vast majority of accidents can be avoided. In recent years, the buzz about self-driving cars has grown. Purely self-driving cars can help reduce traffic accidents, increase traffic efficiency and save energy. However, overtaking is one of the most difficult and complex tasks in autonomous driving technology. Overtaking and lane changing not only involves judging the surrounding environment, maintaining lanes and judging the right time to change lanes, but also requires precise control of each key component of the vehicle.

In order to realize the overtaking function, many researchers have researched vehicle-based solutions, which aim to enable cars to detect road conditions through their own sensors, and then calculate the right time to make a lane change through the on-board system. With the boom of communication technology and 5G, V2X-based solutions for lane change decision making emerged. There are localized solutions that rely on the exchange of information between cars and ultimately the car itself makes the decision to change lanes. There are also centralized solutions where a server collects data from vehicles, and leverages it to perform centralized and optimized decision, thanks to its wider view of the situation on the road. Based on the collected data, the server can predict and react to the vehicle's actions in advance with the help of appropriate artificial intelligence models. This proactive response strategy helps the server to identify potential problems on the traffic in advance, thus further enhancing road safety.

In this thesis, three neural network models for centralized maneuver management predictions are proposed and tested. These models predict the probability of each vehicle to make a lane change based on the collected vehicle operation information. The purpose

of the models is to enable the server to coordinate the operation of vehicles in the area based on these probabilities, so that vehicles that have a higher probability of making a lane change can obtain a safer lane change environment.

Acknowledgements

Firstly, I would like to thank my supervisor Professor Casetti Claudio for giving me the opportunity to work on this topic and being patient and generous to me through this project.

Then, a big thank to Dr. Raviglione Francesco, Dr. Puligheddu Corrado and Dr. Vallero Greta for giving me their valuable guidance and useful suggestions throughout this thesis work.

I also want to thank my parents who have always supported me and believed in me.

Finally, I want to thank myself for not giving up and sticking to the end.

Table of content

List of Figures

List of Tables

Acronyms

V2X	Vehicle to Everything
LCM	Localized Cooperative Maneuvering
CCM	Centralized Cooperative Maneuvering
SUMO	Simulation of Urban Mobility
TSC	Time Series Classification
ANN	Artificial Neural Network
MLP	Multi-Layer Perceptron
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
MCC	Matthew's Correlation Coefficient

Chapter 1 Introduction

1.1 Problems and Background

With the growth of economy, the traffic demand grows day by day. The problems raised by traffic growth is one of the major concerns in countries all over the world. Traffic safety is a top priority among all the issues. According to the WHO statistics, around 1.3 million people died in traffic accident around the world every year. 20 to 50 million people suffered non-fatal injuries and many disabled as result[1]. Traffic accidents also bring huge economical damage. Losses due to road traffic collisions account for 3% of GDP in most countries, while economic losses in low- and middle-income countries even reach to 5% of GDP.

Among all the causes of traffic accidents, human factors account for 94%, for example, driver fatigue, panic, lack of experience and the human response time. It is more likely to cause an accident when the driver intends to lane change or overtake. Traffic accidents have been one of the major causes of fatal and severe injury for years, yet most road traffic crashes are predictable and preventable. Humans have been the major uncertainty in road traffic system.

Compared with humans, self-driving cars can eliminate dangerous driving behaviors such as fatigue driving and drunk driving with their all-around perception systems, intelligent decision-making systems and precise execution systems without the human driver intervention. In addition, autonomous vehicles perform better in obeying traffic rules, and can prevent possible traffic violations in human drivers. Autonomous vehicles can also pre-perceive the surrounding environments through the fusion of their sensors and vehicle-road collaboration technologies, which can avoid risks in advance and reduce the incidence of accidents effectively.

Over the past few decades, Autonomous driving has raised great interests in giant companies like Google, Meta and Apple. With the continuous improvement of automation system technology in the automotive industry, SAE, the International Society of Automotive Engineers, has graded the driving automation of vehicles into six levels[2] from L0 where vehicles are manually controlled to L5 where no human intervention is needed in any circumstances, as shown in Figure 1.1. The market share of autonomous vehicles has gradually increased in recent years. Many vehicles are equipped with L2 level autonomy. However, true self-driving cars still do not exist today, despite the efforts of many research institutes and companies in this field. One of the major problems is accountability. Autonomous driving technologies still face many practical challenges, especially in complex traffic scenarios, or in adverse weather conditions.

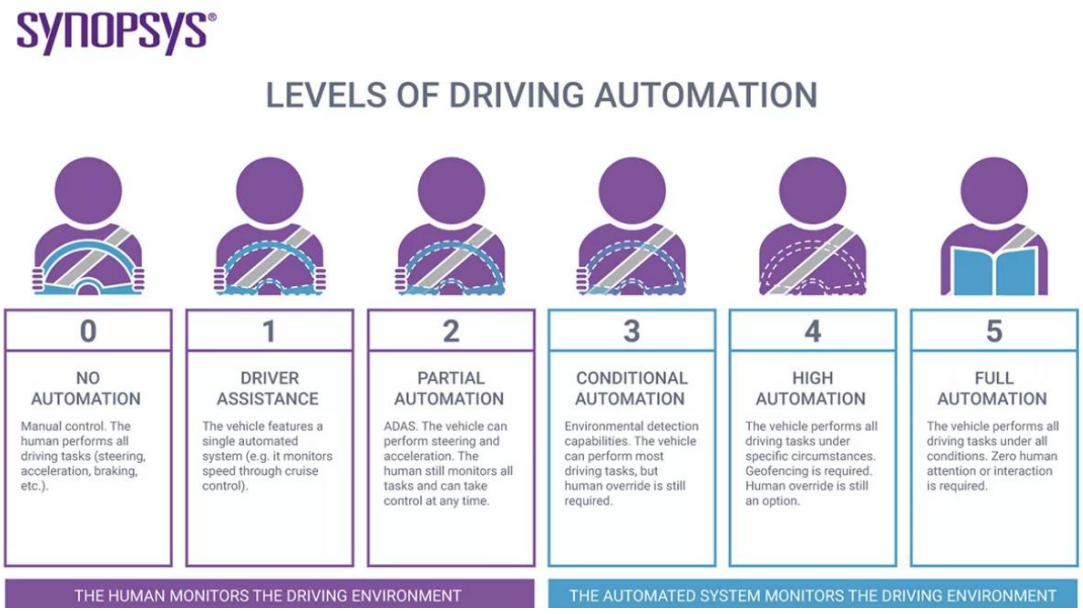


Figure 1.1 SAE Vehicle Autonomy levels[3]

Nearly 75% of traffic accidents are raised by driver's failure to recognize a dangerous lateral maneuver. Therefore, developing and exploiting automated overtaking system is necessary to reduce human error and has the potential to save lives and reduce injuries. Overtaking is one of the hardest and most complicated tasks in autonomous driving technology. Control strategies and electronic control systems such as steering and brake control involves dynamic and complicated tasks. Autonomous overtaking requires not

only high-accuracy environment perception technology to detect surrounding traffic conditions, but also predictions of potentially time-critical decisions and action implementations. Autonomous and semi-autonomous vehicles need to be able to handle complicated driving actions, such as keeping lanes, changing lanes and overtaking which are necessary in typical driving conditions. Key factors for a safe overtake maneuvering including the timing to overtake, maintaining the safe distance with the overtaken vehicle and safely merging to original lane or keeping safe distance with leader vehicle when failing to overtake. Therefore, an automated overtaking system needs to be capable of evaluating the current traffic condition in order to make decision on whether it is possible to perform a safe overtaking and manage the vehicle speed and steering.

1.2 Relevant Works

There are many researches focusing on overtaking systems for self-driving vehicles. Liu et al.[4] analyzed the features that affect the autonomous lane change. wang et al.[5] modeled an approach on generating the optimal land change times and accelerations based on game theory which can be applied to both autonomous and connected vehicle systems.

The problem of autonomous lane changing can be broken down into three sub-problems: decision making, trajectory planning and trajectory tracking. Many studies have been carried out with different considerations in all three areas. Lin et al.[6] proposed a safe overtaking maneuver procedure based on time to lane crossing estimation and the Model Predictive Control (MPC) scheme. Mehmood et al.[7] addressed that trajectory planning in MPC approach is complex and developed an overtaking trajectory planning with quintic polynomial and trajectory tracking with the help of PID controller. But their model only considered kinematic model of car, no dynamic models are included. Some researchers have tried to make cars mimic the behavior of human drivers in some ways. Zhu et al.[8] proposed an overtaking model based on Bezier curve. They tried to mimic the trajectory tracking approach of human drivers where strict trajectory curve tracking

was not required and considers only the angle between horizontal plane and vertical plane and the horizontal distance. Naranjo et al.[9] proposed a two-layer fuzzy controller architecture a contiguous reference of lane GPS for overtaking maneuver. Safety is one of the basic criteria for self-driving cars, therefore, many scholars have designed their automatic lane change models from the perspective of safety. Cui et al.[10] proposed a automatic lane change model with the decision model based on minimum safety distance, the lane change trajectory based on fifth order polynomial lane change trajectory function and the trajectory tracking based on sliding mode control strategy. Wang et al.[11] proposed their model based on the estimation of the conflict probability. The overtaking process is carried out in tracking a safe conflict probability use MPC.

The overtaking systems listed mainly focus on standalone autonomous vehicles which use the data collected by its own cameras and sensors. Although they have been proven in field tests, there still are some flaws. Petrov et al.[12] considers them as backup on-board solutions when connected vehicles are not available such as when the vehicles are in rural areas. The sensing range of on-board sensors is limited, and can only detect adjacent vehicles, but cannot detect objects blocked by adjacent vehicles. In addition, there is no cooperation between vehicles which makes it less efficient to perform a complicated maneuver.

1.3 Cooperative Maneuver

The limitations mentioned above can be tackled by V2X. V2X, or Vehicle-to-Everything, is the communication between vehicles and its surroundings through dedicated wireless networks. It is one of the supporting technologies for smart cars and smart transportation. V2X includes various communication scenarios such as V2V (Vehicle-to-Vehicle), V2I (Vehicle-to-Infrastructure), V2P (Vehicle-to-Pedestrian), V2N (Vehicle-to-Network)[13]. Figure 1.2 illustrates the application scenario of V2X. The purpose of V2X is to reduce

traffic accidents, reduce traffic congestion, improve traffic efficiency, and reduce vehicle pollutant emissions. V2X is also an important tool to realize autonomous driving. It can increase the sensing range by communicating between cooperative sensors to make up for the lack of sight distance of on-board sensors such as cameras and radars. This can also improve the vehicle ability of operation and perception at intersections and under harsh weather environments.

There are two types of V2X technologies using different wireless technologies: Dedicated Short Range Communication (DSRC)[14] based on WLAN and Cellucar-V2X (C-V2X) [15] originally based on LTE. DSRC enables V2V and V2I. The low latency of WLAN makes it well suited for V2X communications, but its coverage limitations also limit the reach of DSRC. V2X senders form ad-hoc networks when they enter each other's communication range. Their communication delay can reach milliseconds level, meet the time requirement of vehicle information interaction. The communication range is generally in the tens of meters. C-V2X is a more recent V2X communication which enables V2N in addition to V2V and V2I. It allows devices communicating directly over the PC5 interface which is called 'sidelink'. It also allows devices communicating with base station in the traditional cellular network way over the Uu interface which is referred to as V2N and is unique in C-V2X. The limitations of 4G networks in terms of latency and available channels also affect C-V2X to some extent.

With the help of 5G network, V2X can:

1. Use millimeter-wave spectrum to increase spectrum bandwidth and realize ultra-high-speed data transmission
2. Reach more than 1Gb/s throughput, with better network coverage uniformity
3. Achieve millisecond-level end-to-end latency
4. Enable penetrating augmented reality to view feedback from vehicles ahead and spot vulnerable road users

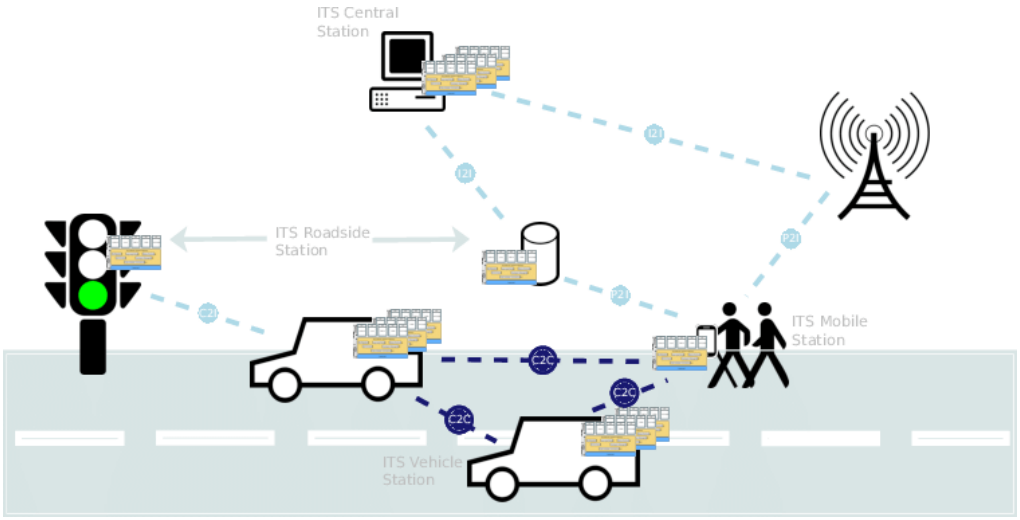


Figure 1.2 V2X application scenario[16]

Cooperative maneuvering (CM) for V2X enables a group of autonomous vehicles to coordinate their driving according to a common centralized or decentralized decision-making strategy. Cooperative maneuvering can assist vehicles to change lanes at a more appropriate speed and timing. Vehicles exchange speed, position, intended trajectory/maneuvers and other useful information with each other[17]. The on-board system can then derive an optimized driving strategy with these information.

Cooperative maneuvering can be achieved through localized or centralized strategy. With Localized Cooperative Maneuvering (LCM), every vehicle needs to be aware of neighboring vehicles. If a car wants to change its lane, it needs to judge whether there is a safe lane change condition based on the information collected from surrounding vehicles and communicate with them to obtain a safe lane change environment. Hafner et al.[18] successfully implemented a cooperative maneuvering system with the complex vehicular interaction protocol (CVIP) with a delay lower than 500ms. Due to the need for the vehicle to locally run all the communication and control algorithms, there will be higher requirements for the on-board chip. However, the reality is that the computing power of the chip that most cars are equipped with is not as powerful. Through the wireless network, the negotiation is carried out by the vehicles themselves, which also puts forward requirements on the format of the transmitted signal. The content and format of the signals transmitted by cars of different brands to the outside world need to be

consistent, otherwise the car home will not be able to communicate due to the car brand, which is not desirable in autonomous driving.

In Centralized Cooperative Maneuvering (CCM), the server will collect all vehicle information within a certain area. The Server Local Dynamic Map (S-LDM) proposed by Raviglione et al.[19] can serve as a bridge between cars and other highly automated maneuver management MEC services. It collects massive amounts of information from many cars and combine it with historical data to provide a map containing lane change related data for cars and other MEC services. Therefore, the server has an overview of the traffic flow in that area. Infrastructures can provide advices, notifications, or information that vehicles can utilize to coordinate maneuver[20]. Based on the collected traffic flow data, the server can calculate the appropriate gap from the leaders and followers for the vehicle that makes the lane change request, and coordinate with surrounding vehicles to obtain the desired gap. Because the global perspective of the server, it is easier to get a globally good solution. With its global view, the server can combine the needs of all vehicles in an area and get a reasonable solution for them, so that vehicles with lane change needs can achieve a smoother transition process. The centralized approach can also respond quickly to abnormal situations and adjust the trajectory of all cars. In the centralized scheme, the server can adjust the vehicles under its govern at any time in response to a change of traffic conditions, thus making the overall traffic flow more smoothly and improving the traffic efficiency. What's more, if the server can predict the possible lane change requests in advance based on the change of the car's driving status and the car's trace log, then it can propose a solution considering both the current and future needs and deploy it in advance for future situations. In this way, when a car does request a lane change in the future, its demand can be satisfied in a short time, which further improves the smoothness and efficiency of the traffic system.

1.4 Thesis Structure

As mentioned earlier, this thesis considers a centralized scenario. The main goal of this work is to obtain a machine learning model, so that the server can use the model to predict the lane change probability of each vehicle in the area it manages based on the collected information. According to the model output, server can coordinate well in advance the operation of other vehicles around the vehicles that are predicted to have high possibility to change lanes, so that those vehicles can safely change lanes.

The thesis is divided in five chapters:

- Chapter 1 introduces the background and main motivations of this thesis.
- Chapter 2 describes the main theoretical concepts and simulation tools used.
- Chapter 3 introduces the model setup process. Firstly, the input data format on the machine learning algorithms is described. Then, the algorithms themselves are introduced. And finally, the evaluation metrics used are described.
- Chapter 4 describes the simulation process, the trained model and the model performance evaluation.
- Chapter 5 summarizes the project and describes some future work that may exploit more model possibilities.

Chapter 2 Relevant Theory

In this chapter, the simulation tool and the relevant theory of our research are discussed. This thesis uses a server-client model to predict the probability of a vehicle taking a maneuver action while travelling on a motorway. Each vehicle acts as a client which constantly send its driving condition to the server, such as its position and speed. The server is able to collect data of vehicles driving in a local area and make a prediction on whether a lane change is likely to occur, in order to timely control the maneuver in an automated way. Concerning the data collection, SUMO is leveraged as a simulation tool and its output is considered as the data stored in the server. To make a prediction, several machine learning methods are considered.

2.1 SUMO

SUMO (Simulation of Urban Mobility) is an open-source multimodal traffic simulator. The multimodal characteristic makes it capable of combining different modes of transportation and vehicle types. SUMO is a microscopic simulator, which operates at the level of each individual object[21]. It models all vehicles, pedestrians and public transport individually. Maps for traffic simulation can be created by users or imported from reality. Users can choose to use a defined set of configurations for the simulation to run automatically or control the simulation process with the Traffic Control Interface (TraCI). SUMO allows to collect a large number of different metrics and information such as vehicle position over time and vehicle emissions. The output can be vehicle-based, simulation-based or traffic lights-based. In this thesis, the most useful output is

represented by the raw vehicle position dump[22] and the lane change output[23].

A typical sumo simulation consists of the following five parts[24]:

1. Network Building

   SUMO road networks are encoded as XML files with filename extension '.net.xml'. The network file describes all parts related to traffic in a map, including the roads, the pedestrian and the traffic lights. The main components in a SUMO network file are:

   - Edges. Edges represent the roads or streets. Each edge contains a certain number of lanes internally. The outermost lane is labeled as lane_0 and the inner most lane has the largest lane id. The position, shape and speed limit of every lane are also included. The edges in SUMO network are unidirectional.
   - Traffic Light logics.
   - Junctions. Junctions represent the area where different streams cross. For example, if there is a change in the number of lanes, or there is a crossroad, the area where different edges met is recorded in junctions.
   - Connections. Connections describe how different lanes are connected, especially within junctions. The incoming lane and the outgoing lane are recorded in a connection, as well as the traffic light logic that controls the traffic flow in the connection if the connection is controlled by a traffic light.

   SUMO provides a tool called '*netedit*'. User can choose to set up a map by hand or import a network from elsewhere. SUMO also provides a tool called '*netconvert*' if the user wants to import real-world maps. With *netconvert*, user can filter the map and import only the desired road components.

   In this thesis, only motorways are simulated, therefore no traffic light and pedestrians are involved in the simulation.

2. Demand Modelling

   Demand in a SUMO simulation represents the traffic flow and the file extension is '.rou.xml'. It includes the moving components in a network and their characteristics. The vehicle types and their characteristics need to be specified in this traffic demand

file. The amount of vehicles in a simulation and the route that each vehicle takes need to be declared too. There are two ways of defining a vehicle's route, one called 'trip', the other called 'route'. With a trip definition, only the starting edge and the destination edge are mandatory while all the edges that a vehicle will pass need to be specified within a route. SUMO enables various demand generation method. User can choose to write demand by hand, use randomized measures, or import from other sources.

The routing configuration in this thesis will be described in detail in Chapter 3.

3. Simulation configuration

   The SUMO simulation process can be managed either through a GUI (Graphical User Interface) or command line. The user thus needs to specify which XML files SUMO needs to use. Some additional configuration files can also be specified, such as the ones managing traffic light phases, bus stops and rerouters. The related XML files should be specified to SUMO as additional files. The time period of a simulation needs to be defined in the configuration part as well as the time step length. The time step length needs to be between [0.001, 1.0] which means the step length should be at least 1 ms and at most 1 second. The default value is 1 second. The final part of the configuration is related to the outputs. The output content and the location to store the desired output need to be specified in this stage.

4. Simulation

   With all the necessary modules, the SUMO simulator will read the network first. Then the additional files are read in the order they are provided. After this step, the network with all additional details is set up. Then, SUMO will read the route file and start the simulation process. It will read the first n steps routing configuration, then perform the simulation for those n steps. The routing of next n steps will be read during the simulation process.

5. Simulation Output

   SUMO enables the generation of many types of output files and simulation results.

The results can be vehicle based, network-based and traffic lights-based. Taking advantage of the different types of simulation results, users can obtain a variety of desired data. The output files generated by SUMO are in XML-format and SUMO provides tools to convert them into other types, such as Comma Separated Values (CSV). Furthermore, some metrics related to the simulation status are printed on the command line every 100 simulation steps.

2.2 Machine Learning Methods

Machine learning, seen as a part of artificial intelligence, has become a trending subject in modern society. Machine learning algorithms build a model based on training data in order to make predictions without being explicitly programmed to do so. machine learning algorithms can find patterns in provided data which might be challenging for human to manually create. With machine learning, human programmers don't need to specify every step needed for a computer to perform a task. A properly built machine learning model will learn patterns from the training data itself.

Machine learning algorithms are classified into the following four categories in general:

1. Supervised learning

   Supervised learning builds a model from a set of data which is known as training data that contains both the inputs and the desired output. Supervised learning algorithms include classification and regression. The output of classification algorithms is restricted to a set of values. If the set is of size 2, the classification algorithm is also called binary classification. If the algorithm output has more than 2 output labels, it is called multiclass classification. Output of regression algorithm can be any real number within a range. One classical example of supervised learning is spam detection. The training data on whether an email is spam is classified by human expert. Every email will be associated with a label indicating if it is spam. Typical algorithms for supervised learning include linear regression, logistic

regression and random forest.

2. Unsupervised learning

   In unsupervised learning algorithms there is only input data and no specific, defined output. Since no reference output is provided, the algorithm needs to learn the data by itself. Learning from enormous amount of data, the unsupervised learning algorithms may provide previously unknown insights and identify hidden patterns. Therefore, unsupervised learning algorithms do not necessarily have clear results. Instead, it investigates the differences or similarities between sample data. Unsupervised transformation is an application of unsupervised learning which creates a new representation of the input data that might be easier for human or other machine learning algorithms to understand[25]. Clustering algorithm is another application of unsupervised learning. It assigns the data into subsets of similar items. Unsupervised learning can be used in malware detection, fraud detection and human error identification during data entry.

3. Semi-supervised Learning

   Semi-supervised learning is a combination of supervised learning and unsupervised learning. In a semi-supervised learning problem, the dataset contains both labeled and unlabeled data. In real world, it is easy to collect data but much harder to have labeled data which makes labeled data more costly in general. Obtaining labeled data requires human work to do the labelling. With as few people work as possible as well as relatively high accuracy, semi-supervised learning is getting more attention. Semi-supervised learning can be useful in pattern recognition.

4. Reinforcement Learning

   Reinforcement learning is a unique type of algorithm. The basic idea behind reinforcement learning is simple. Take game for example, if a certain gaming strategy can achieve high score, then this strategy will be ‘reinforced’. AlphaGo Zero is a famous example of applying reinforcement learning. By playing games against itself, AlphaGo Zero reached the level of AlphaGo Master in 21 days, and

exceeds all the older versions in 40 days[26].

2.2.1 Time Series Problem

Time Series Classification (TSC) is an important and challenging problem. Hundreds of TSC algorithms have been proposed since 2015[27] with the increase of temporal data availability[28]. Due to their natural temporal ordering, time series data are presented in almost every task that requires some sort of human cognitive process[29] such as the lane changing process considered in this thesis. Time series exist in many real-world applications ranging from human activity recognition to electronic health records.

Deep learning successfully demonstrated its capabilities in many TSC problems. A huge amount of algorithms has been proposed to solve problems in natural language process (NLP) field such as machine translation.

A typical deep learning framework for time series classification is represented in Figure 2.1.

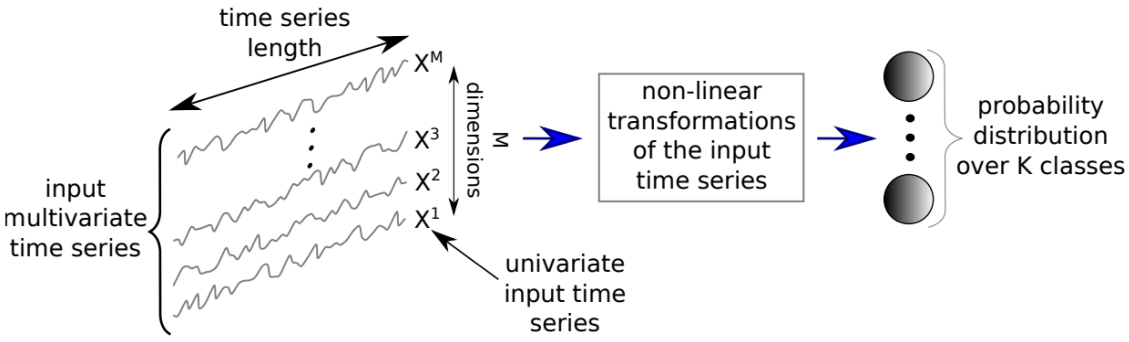


Figure 2.1 Time Series Classification framework[30]

The problem this thesis models is a time series classification problem which try to predict the lane change probability of vehicles using a time series of collected vehicle information. Several deep learning models are introduced.

2.3 Deep Learning

Deep learning, also known as deep neural networks, is part of the machine learning methods family based on artificial neural network. Deep learning architectures such as recurrent neural networks and convolutional neural networks have been widely used in fields including speech recognition, natural language processing, computer vision and autonomous driving. A well-trained neural network can produce results comparable to human experts and in some cases even surpassing. The disadvantage of deep learning is that it usually has higher requirements on hardware devices due to the complexity of the algorithm.

2.3.1 Artificial Neural Networks

Artificial neural networks were inspired by information processing and distributed communication nodes in biological neural systems. Figure 2.2 is a schematic of neuron. A neuron receives signals via the dendrites and soma and send out signals down the axon which connects to another neuron. The signal a neuron received from other neurons can be viewed as inputs; the signal it passes down to other neurons can be viewed as outputs. Human brain is composed of neurons while artificial neural network is composed of artificial neurons as shown in Figure 2.3. A neuron takes several binary inputs and produces a single binary output. Each input x_i is associated with a weight w_i. A weight is a real number representing the importance of the respective input to the output. By varying the weights and the threshold, we can get different models of decision-making. Different types of neurons are distinguished by the activation function which is how the output is calculated. For a simple type of artificial neuron called perceptron, the binary output is determined by a threshold value and the weighted sum of each input $\sum w_i x_i$. If the weighted sum exceeds the threshold value, a positive output will be produced. Otherwise, the output will be negative. Today a more commonly used artificial neuron is called sigmoid neuron. In a sigmoid neuron, the output can be calculated by the following equation, with a bias term b which controls the threshold of the neuron:

$$h\Theta(x) = sigmoid(Z) \text{ where } Z = b + \sum w_i x_i$$

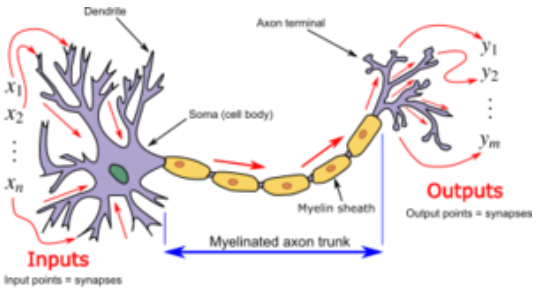


Figure 2.2 Neuron schematic[31]

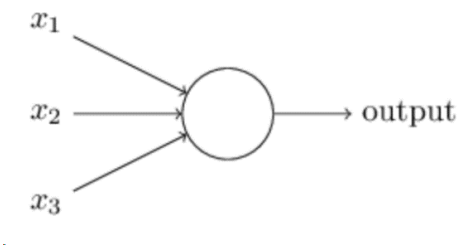


Figure 2.3 Artificial Neuron schematic[32]

An artificial neural network is a layered structure. A typical ANN is composed of an input layer, an output layer and one or more hidden layers. Each layer is composed by several artificial neurons and is connected only with its direct preceding and succeeding layer. Figure 2.4 shows an example of ANN with one hidden layer. Each node represents a neuron; each arrow represents a connection from the output of one artificial neuron to the input of another neuron.

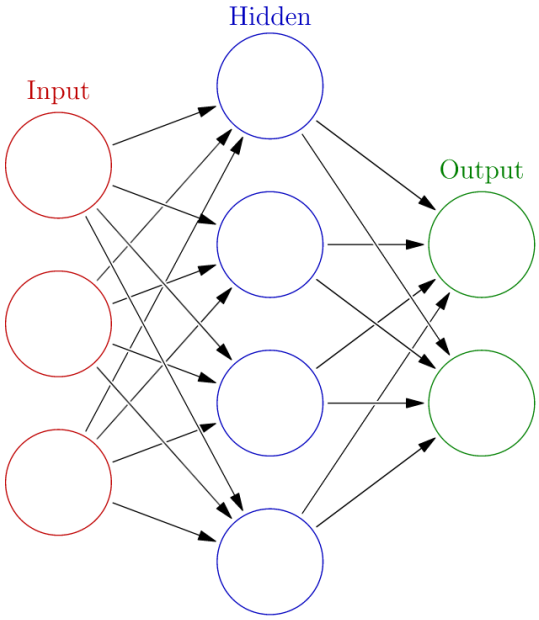


Figure 2.4 Neural Network structure[33]

The input layer is where the external data are received while the output layer is where the ultimate results are returned. The layers between input layer and output layer are called hidden layers. Hidden layers are considered as the core of a neural network. It is the

weights related to the connection between input layer and hidden layer, between hidden layers and between hidden layer and output layer that need to be learned and updated at every learning iteration.

2.3.2 Multilayer Perceptron

A multilayer perceptron (MLP) is a fully connected class of artificial neural network. A fully connected neural network means that every node in a layer is connected to every node in the next layer. A MLP consists of at least three layers. MLP usually uses a sigmoid as the activation function. Learning on MLP is based on the error in the output compared to the given results. Through backpropagation, the connection weights are updated.

2.3.3 Convolutional Neural Networks

Convolutional neural network (CNN) is a popular Deep learning architecture. Convolutional neural networks have been widely applied in daily life, such as image recognition, video analysis, natural language processing, anomaly detection and drug discovery, etc. Convolutional neural networks are a specialized type of artificial neural networks that use a mathematical operation called convolution in place of general matrix multiplication in its layer. It is inspired by human visual system and is specifically designed to process pixel data and are used in image recognition and processing. CNNs can reduce data size effectively with the features of an image retained. Images are hard to handle because the amount of data in an image is huge and the features are hard to be preserved during the image processing process.

Kunhiko Fukushima introduced the ‘neocognitron’ in 1980. The neocognition introduced two basic layers in CNN: convolutional layer and downsampling layer. Kunhiko’s work has been used in pattern recognition tasks and has been the inspiration for convolutional neural networks.

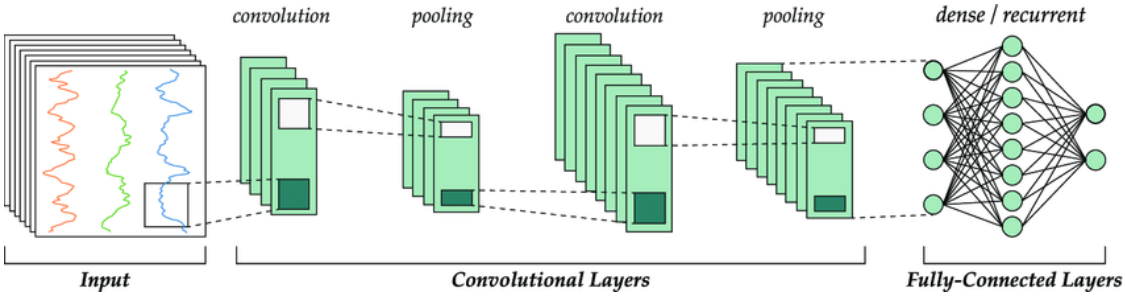


Figure 2.5 Typical CNN structure[34]

Figure 2.5 shows a typical structure of a CNN. Take image recognition as an example, a CNN is usually consisting of the following layers.

1. Input layer

   The input layer is to input the original data or data preprocessed by other algorithms into the convolutional neural network. The data type is not specifically defined. It can be a digital image or other types. Different from the input format (one-dimensional vector) of the fully connected neural network, the input format of the input layer of CNN retains the structure of the image itself. For a black and white 28x28 image, the input to the CNN is a 28x28 2D neuron. For 28×28 images in RGB format, the input of CNN is a 3×28×28 three-dimensional neuron where the third dimension is called 'channel'.

2. Convolutional Layers

   Convolutional layers are one of the most important steps in a CNN architecture which aimed to filter the specific feature of the image. The 'kernel' in convolutional layers can be seen as the features to be extracted. Therefore, choosing a suitable kernel is crucial to capture the most salient and important information contained in the input signal, which enables the model to make better inferences about the content of the signal.

   Local receptive fields and shared weights are two important concepts in a convolutional layer. Local receptive fields are basically the area covered by the kernel. It can be viewed as where the kernel capturing features from the previous layer. The next-layer neuron matrix generated by a receptive field scan with a convolution kernel is called a feature map. Shared weights means that the features in one feature map are

obtained using the same kernel. It means that each feature map is associated with a specific kernel. Those two features in convolutional layer helps a lot in reducing parameters.

3. Pooling Layers

   If the kernel size in convolutional layer is relatively small, the obtained feature map is still relatively large. An additional dimensionality reduction operation can be performed on each feature map through the pooling layer with the output depth still be the number of feature maps. There are two commonly used pooling methods: max pooling and average pooling. With the size of pooling area set, the max pooling method will select the maximum value on the pooling area of feature map while the average pooling method will give the average value.

4. Fully Connected Layers

   In the fully connected layer, the outputs of the last convolutional layer are flattened. Then every node in current layer is connected with all the nodes in the next layer. And finally connected with the output layer

2.3.4 Recurrent Neural Networks

The term "recurrent neural network" is used to refer to the class of networks with an infinite impulse response, whereas "convolutional neural network" refers to the class of finite impulse response. Recurrent Neural Networks use some form of feedback, where they return the output back to the input. It’s like a loop from the output to the input in order to pass information back to the network. Hence, Recurrent Neural Networks are capable to remember past data and use its information for prediction. From Figure 2.6, the structure of an ordinary RNN network clearly shows that it is good at solving time series related problems.

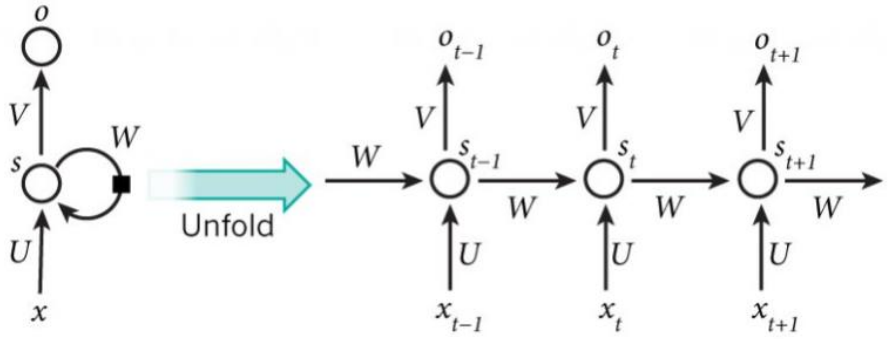


Figure 2.6 RNN structure and its unfolding schematic[35]

RNN can connect information from previous steps to the current step. The following formulas briefly describes how states are updated in RNN networks.

$$o_t = g(V_{s_t})$$

$$s_t = f(U_{x_t} + W_{s_{t-1}})$$

The first formula represents how the output is calculated. The output layer is a fully connected layer. Term V is the weight matrix of the output layer and g is the activation function. The second formula represents how the state of recurrent layer with U the weight matrix of input x and W the weight matrix of the last state which is used as part of the input in the current step.

The problem of simple Recurrent Neural Networks is Gradient Explosion and Gradient Vanishing. This happens because the errors are accumulated along each time step. The accumulated error will eventually lead to a huge error term or a nearly zero error term if the sequence is long. This results in gradients cannot be passed all the way through longer sequences during training, making RNNs unable to capture long-distance effects.

2.3.5 Long Short-Term Memory

Long short-term memory (LSTM) is a special type of RNN that is able to solve the problem of RNN lacking long term memory. However, LSTM networks can only solve the problem partially and still suffer from the exploding gradient problem.

All recurrent neural networks have the form of a chain of repeating modules of the neural network. In a standard RNN, this repeating module would have a very simple structure, such as a single tanh layer. In LSTM, however, the repeating module is more complex. The structural of LSTM is shown in Figure 2.7. The repeating module in LSTM consists of three different layers called 'gate'. Gate is a way to selectively pass information. Cell in LSTM basically means the repeating module. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

- Forget gate

  Forget gate determines which information to be aborted from cell state. It takes in the input x_t and the output from last step h_{t-1} and gives a matrix that indicating how the parameters from the cell state of the previous step should be preserved. The matrix elements are in range [0, 1] where 0 means completely discarded and 1 means completely reserved

- Input gate

  Input gate is where the cell state gets updated. The sigmoid layer determines which values to be updated and the tanh layer creates a vector of new candidate values of the cell state. Adding up the state values selected from the new input and the state values selected from the previous states, the cell state then is updated.

- Output gate

  The output is then calculated using the updated cell state and the input data.

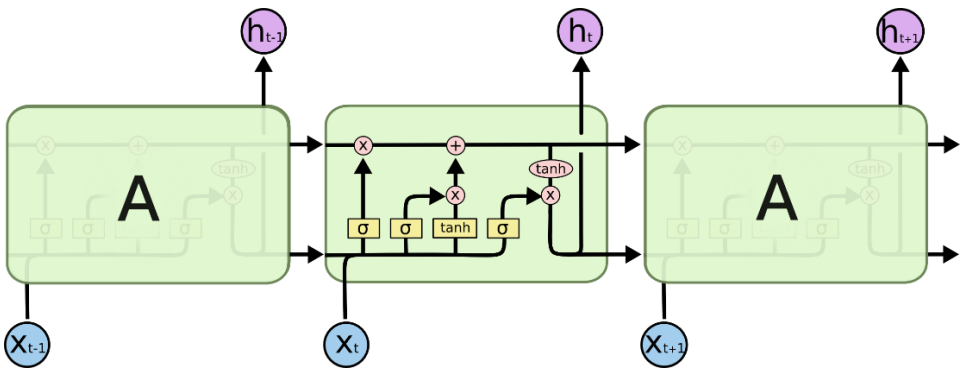


Figure 2.7 LSTM schematic[36]

The connection weights and biases in the network change once per episode of training, analogous to how physiological changes in synaptic strengths store long-term memories; the activation patterns in the network change once per time-step, analogous to how the moment-to-moment change in electric firing patterns in the brain store short-term memories. The LSTM architecture aims to provide a short-term memory for RNN that can last thousands of timesteps, thus "long short-term memory".

Chapter 3 Model Setup

In this thesis, the server which collects the vehicle information should make prediction on the probability of a vehicle changing its lane on next second. After making the prediction, the server may decide to remain silent or deliver further instructions to the vehicle and its neighboring vehicles for corporation.

3.1 Data Processing

3.1.1 Simulation

The traffic data of each vehicle are obtained with SUMO simulations. SUMO allows users to generate various types of outputs. The outputs generated by SUMO are in XML-format. The outputs used in this thesis are the raw vehicle position dump (Rawdump) and the lane change information.

Rawdump

The network dump output contains detailed information for each simulation step on each edge for each vehicle[22]. The rich details in network dump output makes it flexible. User can get various information they want with proper additional programming. As a consequence of exhaustive details, the Rawdump outputs are quite large. It is easy to get a file that is several GB.

A general Rawdump output structure is shown in Figure 3.1.

```
<netstate>
   <timestep time="<TIME_STEP>">
      <edge id="<EDGE_ID>">
         <lane id="<LANE_ID>">
            <vehicle id="<VEHICLE_ID>" pos="<VEH_POSITION>" speed="<VEH_SPEED>"/>

            ... more vehicles if any on this lane ...

         </lane>

         ... more lanes if the edge possesses more ...

         ... optional persons and containers if currently active on that edge
         <person id="<ID>" pos="<OFFSET_FROM_EDGE_BEGIN>" speed="<SPEED>"/>
         ...
         <container id="<ID>" pos="<OFFSET_FROM_EDGE_BEGIN>" speed="<SPEED>"/>
         ...

      </edge>

      ... more edges ....

   </timestep>

... the next timestep ...

</netstate>
```

Figure 3.1 Raw vehicle position dump output format[22]

The structure of the raw vehicle position dump output in this thesis after transformed into csv file using the python tool provided is shown in Table 3.1. There are no person or container in the simulations, so the Rawdump output contains only information about vehicle. The lanes that have no vehicle on it at a certain timestep are left blank.

timestep_time	edge_id	lane_id	vehicle_id	vehicle_pos	vehicle_speed
0	41575752	41575751_0	flow01	4.5	32.39
0	41575752	41575751_1			

Table 3.1 Transformed raw vehicle position dump output format

Lane change

The lane change outputs record all events where a vehicle laterally changes from one lane to another[23].

The general structure of a lane change output file is shown in Figure 3.2.

```
<lanechanges>
  <change id="<VEHICLE_ID>" time="<TIME_STAMP>" from="<SOURCE_LANE>" to="<DESTINATION_LANE>" reason="<CHANGE_REASON>" ... />
  ...

</lanechanges>
```

Figure 3.2 Lane change output format[23]

The information contained in a lane change output are listed in Table 3.2.

change_id	change_dir	change_time	change_pos
change_reason	change_speed	change_from	change_to
change_type	change_followerGap	change_followerSecureGap	change_followerSpeed
	change_leaderGap	change_leaderSecureGap	change_leaderSpeed
	change_origLeaderGap	change_origLeaderSecureGap	change_origLeaderSpeed

Table 3.2 Lane change output content

3.1.2 Feature Selection

The thesis aims at predicting a lane change event with machine learning aided. The features selected and processed should be suitable for a machine learning input and serve the end goal of this thesis.

When a human driver wants to change to another lane, he will look at the vehicles around to decide whether it is safe to change lane. Human driver will look at the leader vehicle on his current lane and the leader vehicle on the lane that he wants to change to. The leader vehicles determine whether there is enough space to change. Human driver will also look at the followers on these two lanes to check if they are catching up. Both distance between the vehicle and its neighbors and their relative speeds are important. We need to ensure that there is enough safe distance throughout the lane changing process. The lane changing direction which is not known in data collection state should be part of the algorithm output. So, it would more propriate to provide the vehicle information on both the lane on the left side of current driving lane and the lane on the right side, instead of providing just the current driving lane and the lane that the vehicle is changing to.

For each lane change event from SUMO lane change output, the parameters subtracted from network dump output that are considered in the machine learning part are the following ones:

Lane: the lane number that the vehicle is on, mostly 0 or 1. It indicates on which lane of an edge the vehicle is travelling.

Speed: the speed of the vehicle.

And the speed and relative distance of the vehicles around the vehicle in each lane as listed in Table 3.3:

Left Lane	Current Lane	Right Lane	Explanation
l_l	C_l	R_l	Existence of leader on each lane
L_l_speed	C_l_speed	R_l_speed	The leader's relative speed if exist
L_l_dist	C_l_dist	R_l_dist	The leader distance in range [0, 1500]
L_f	C_f	R_f	Existence of follower on each lane, 1 for exist, 0 for not
L_f_speed	C_f_speed	R_f_speed	The follower speed if exist, set to 0 if follower doesn't exist
L_f_dist	C_f_dist	R_f_dist	The follower distance in range [-1500, 0]

Table 3.3 Machine learning model input data content

Current lane refers to the lane the vehicle is driving on and the left lane and right lane refers to the relative left and right lane of the current lane. It is possible that left lane or right lane doesn't exist. For example, if the vehicle is driving on the left most lane of an edge, then there is no more left lane cause the current lane is already the left most. In this case, the left lane leader and left follower will be marked as don't exist, e.g., l_l and l_f will be labeled with 0. The same applies to the right lane.

The existence of a leader or follower on each lane are binary terms with label 1 indicating the existence of the neighbor on the specific direction and label 0 indicating the absent of that neighbor. The speed term in this dataset is the relative speed of the neighbor vehicle with respect to the vehicle whose maneuver action will be predicted. The distance refers to the distance along the edge from the vehicle considered to the specific neighbor. Horizontal distance is not considered. Only the leaders and followers within 1500 meters distance from the vehicle are considered. For most of the normal driving vehicles, it takes more half a minute to travel 1500 meters. A maneuver action takes place in quite a short period which is often several seconds. A neighboring vehicle which is too far away

doesn't have much influence on the decision of maneuvering or not. If the leader/follower is marked as absent, the distance and speed term are also set to 0.

Lane change is considered a time series action. The driver will not take action as soon as he intends to change to another lane. A normal lane change event follows the process of road traffic condition observation, evaluation and decision making. A consecutive 10 seconds of the selected feature above are feed into the time series classification algorithms. The data structure of the input data for the machine learning algorithms is represented in Table 3.4. The algorithms use data collected in the last ten seconds to predict the vehicle action in the next time instant.

	lane	speed	l_l	l_l_speed	…	r_f_speed	r_f_dist
-10							
-9							
…							
-1							

Table 3.4 Machine learning model input data structure

3.2 Classification Model

Lane change is a continuous series of actions from making the decision, observing the traffic condition to finally changing to the destination lane. It is natural to use algorithms that are capable of dealing with time series data. Standard machine learning algorithms are designed for structured data which are not always suitable for unstructured data like time series.

CNN

Motivated by the success of CNN architecture in image recognition problems, researchers has started trying to adopt them in time series problems[37]. Unlike images, convolution in time series problems is one-directional. The filters slide only over the time series while filters in image problems slide along the width and the height. The result of a convolution

on an input time series can be considered as another time series that underwent a filtering process. Time series data in CNN can be seen as viewed as a black and white photo which contains only one channel, and the convolution is performed only on one dimension. With the support of modern high-performance and high-memory graphics cards, the CNN model can be trained in a very fast speed.

A simple CNN model is introduced in this thesis and is described in Chapter 4.

RNN

RNNs are good at dealing with unstructured data like text and audio. The recurrent operation in RNN brings back information from the past to the current prediction process. In CNN, though the convolution operation can take past information into account, the scope of what it can include is limited. Convolution can only consider a range as large as the size of the kernel. However, in RNN, all the past information are considered through the output of each step. Despite the advantages RNN have in time series problems, it also suffers from a lack of long-term memory. Since the input data in this thesis consists of only ten timesteps which seems not long, a deep RNN model is introduced in this thesis to test if the longer-term memory brought by it can be more effective than that brought by the kernel in CNN. The lack of long-term memory in RNN can be interpreted as the data that are closer in time bring more effect on the prediction while the data from a longer period have little effects on the result. However, this effect is intuitively coincident in a lane change scenario: the vehicle movements closer to the moment of decision have a greater impact on the maneuvering decision.

LSTM

The main aspect in LSTM is the cells' ability to learn the important parts of the sequence and forget the less important ones. Although the properties of the RNN model are intuitively compatible with the properties of the lane-changing behavior, it still need to be tested. In this thesis, the LSTM model is used as another reference. By building an

LSTM model, this thesis wants to verify how the long-term memory forgotten by the RNN affects the accuracy of the model.

3.3 Performance Metrics

When evaluating a classification machine learning classification algorithm, accuracy is always the first consideration. But accuracy can't represent how the algorithm all the time. Other evaluation metrics should also be considered. In this thesis, the confusion matrix is included in the evaluation process.

Confusion Matrix

Confusion matrix is a specific table layout that displays the performance of a classification algorithm. Each row in the confusion matrix represents instances of an actual class while each column represents instances of a predicted class, or vice versa. The template for any binary confusion matrix is shown in Figure 3.3. With confusion matrix, many useful metrics can be obtained. This allows more detailed analysis than simply the accuracy.

		Predicted Condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual Condition	Positive (P)	True Positive (TP)	False Negative (FN)
	Negative (N)	False Positive (FP)	True Negative (TN)

Figure 3.3 Confusion Matrix format of binary classification

For a multiclass classification problem, the confusion matrix can be used too. The number of rows and columns equals to the number of different classes. For a multiclass classification problem, the metrices of each class are obtained with take one specific class

as positive class and all other classes as negative class.

F1 Score

With the confusion matrix, precision and recall are two commonly used matrices. Precision (P) is ratio of true positive among all predicted positive cases while recall (R) is the ratio of correctly predicted positive cases among all actual positive cases. Since precision and recall are usually a pair of contradictory metrices, F1-score is generally used as the evaluation metrics in order to better evaluate the comprehensive performance of the classifier. F1 score is the harmonic mean of precision and recall:

$$F_1 = 2\frac{P * R}{P + R}$$

$$\text{with } P = \frac{TP}{TP+FP},\ R = \frac{TP}{TP+FN}$$

Since F1 score considers both precision and recall, it is able to relay the true model performance even if the dataset is imbalanced.

F1-score is defined on each class. With different class viewed as the 'positive' class, the F1-scores are different. The formula of F1-score above is defined on binary classification problems. For a multi-class classification problem, the F1-score is more complicated. There are macro-average F1-score and micro-average F1-score. The macro F1 is obtained by averaging the F1 score calculated for each class. The micro F1 score is calculated with the P and R value obtained by adding up the TP, FP and FN of each class.

For imbalanced dataset, macro F1 score is a better measure of model performance, since it gives equal importance to each class while micro F1 score gives equal importance to each data which will lead to major class shadowing the minor class.

Since the dataset in this thesis is unbalanced, the macro F1 score is used for model evaluation.

Matthew's Correlation Coefficient

Accuracy is always the first metric comes in mind when talking about evaluating a machine learning algorithm. Accuracy is the proportion of correct classifications and can

be calculated with the confusion matrix as the following equation:

$$ACC = \frac{TP + TN}{TP + FN + FP + TN}$$

The problem with accuracy is that it will yield misleading results if the dataset is unbalanced. Unbalanced means that there are many more instances in some class and much less instances on the others. In this condition, the algorithm may seem work well looking at only accuracy.

Matthew's correlation coefficient (MCC) can be an alternative of accuracy for unbalanced datasets. It represents the correlation between true values and the predicted ones. MCC takes into consideration all four aspects from the confusion matrix and can eliminate the side effects on accuracy brought by the unbalanced dataset. Some scientists believe that MCC is the most informative single score to evaluate binary classifiers' prediction quality. One of MCC's benefit is that it is symmetric. Swapping positive class and negative classes don't affect the score. It ranges from -1 to 1. A score of 1.0 means a perfect classifier while a value close to 0 means the classifier performs close to random guessing. The Matthew's correlation coefficient of a binary classifier can be calculated with:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

For multiclass classification problems, MCC is defined over the $K * K$ confusion matrix of K classes and is called the R_k statistic.

$$MCC = \frac{\sum_k \sum_l \sum_m C_{kk} C_{lm} - C_{kl} C_{lmmk}}{\sqrt{\sum_k (\sum_l C_{kl})(\sum_{k'|k' \neq k} \sum_{l'} C_{k'l'})} \sqrt{\sum_k (\sum_l C_{lk})(\sum_{k'|k' \neq k} \sum_{l'} C_{l'k'})}}$$

The formula is complex but can be calculated easily with sklearn module in python with the following code:

```
from sklearn.metrics import matthews_corrcoef
mcc = matthews_corrcoef(y_true, y_predict)
```

Chapter 4 Experiment and Result

This chapter presents the experiment performed for traffic data collection and the corresponding data processing process. The data then are feed into the machine learning algorithms described in the previous chapter. The machine learning algorithms performance are evaluated and compared with the previously discussed metrices.

4.1 Data Collection

4.1.1 SUMO Simulation

Network

In this thesis, 6 simulations on A22 highway on the Italy to Austria border are run with three different types of vehicles. The map used for the simulation is represented in Figure 4.1. This thesis focuses on the vehicle performance when travelling across the national border. The vehicle routes are highlighted. The route is 39 km long. Each simulation lasts for one hour. To better emulate the actual road traffic, both normal vehicles and trucks are included in the simulation.

Traffic Demand

The characteristics of each vehicle are listed in Table 4.1. Marco et al.[38] identified the capacity of A22 highway overtaking lane is 1916pcu/h. Different traffic density scenarios are to be provided by the six simulations. The traffic flows for each simulation are listed in Table 4.2 ranging from a relatively small flow to full utilization of the overtaking lane capacity.

Vehicle type	Normal	Truck	Heavy truck
Length (m)	4.4	6	10.2
Max Speed (m/s)	50	35	28

Table 4.1 Vehicle types of SUMO simulations

Simulation ID	1	2	3	4	5	6
Traffic flow (veh/h)	801	999	1200	1500	1701	1899

Table 4.2 Traffic flows in SUMO simulations

Configuration

The input files contain only the network of motorways in Italy border as shown in Figure 4.1 and the traffic demand file. Each of the three types of vehicles accounted for one-third of the total traffic flow in each simulation. Each simulation lasts for one hour with the simulation step size set to 1s. The seed is set to 22 to ensure the reproducibility of the experimental results.

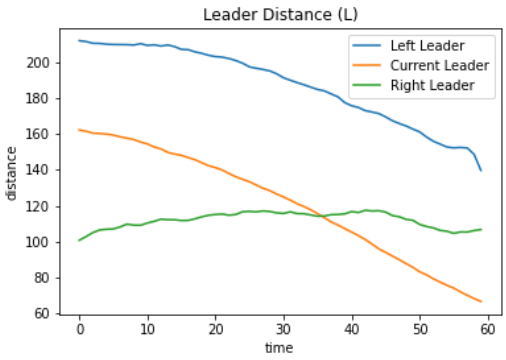
Figure 4.1 Simulation trajectory

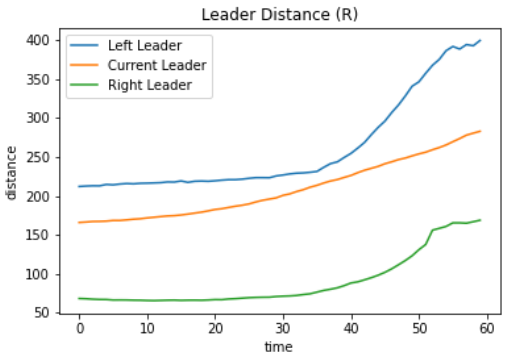
4.1.2 Data Processing

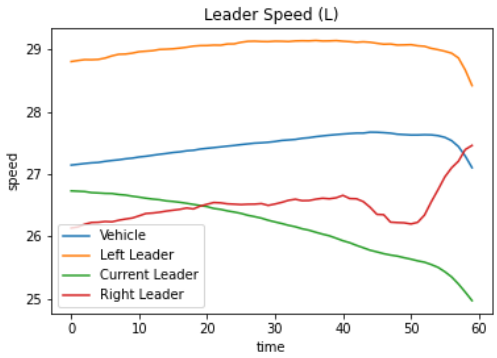
For each simulation output, every lane change event happens later than 30 seconds after

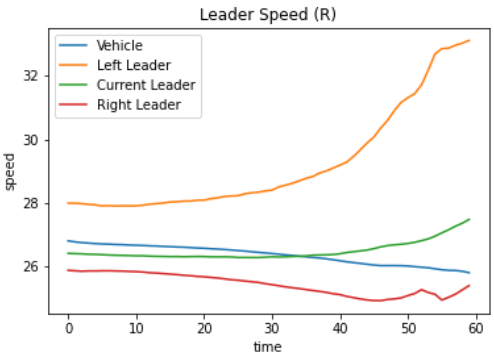
the vehicle inserted into the simulation was accounted. The vehicle id of every lane change event obtained is used to find the vehicle route that the vehicle traveled during the simulation. 30 seconds of neighboring vehicle data before the lane change time are collected. The lane change output provides information about the lane change event while the rawdump output provides all the other information needed to calculate the data for the machine learning model input.

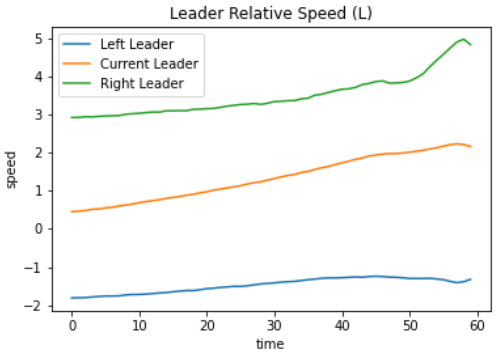
The figures below show the average value of each of the machine learning algorithm input data and the derived acceleration from 60 seconds before the lane change event to the second before the vehicle took action to change its lane. Time 0 in those figures refers to one minute before the vehicle change its lane and time 60 refers to the second before lane changing. Figures on the left column shows the distances and speed of the vehicle itself and its leaders and followers of vehicles that turned to the left at time 61, similarly on the right column where vehicles that merge in the right lane are considered. For figures on the left column, the relationship between data labeled ‘current’ and ‘left’ is more important while the relationship between ‘current’ and ‘right’ is more important on the right column.

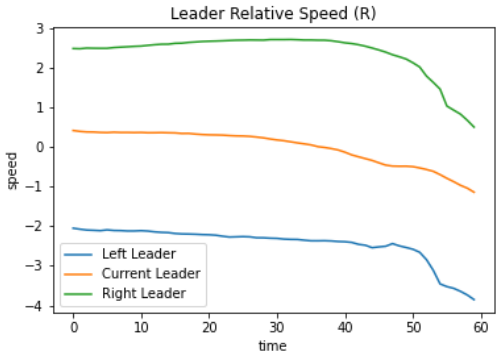


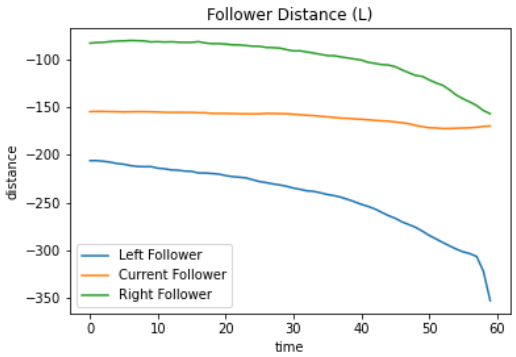


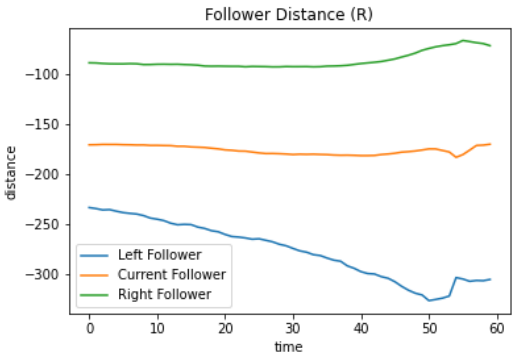

Leader Speed (L)
speed
time
Vehicle
Left Leader
Current Leader
Right Leader

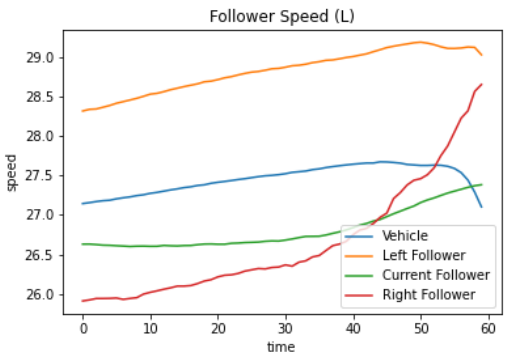

Leader Speed (R)
speed
time
Vehicle
Left Leader
Current Leader
Right Leader

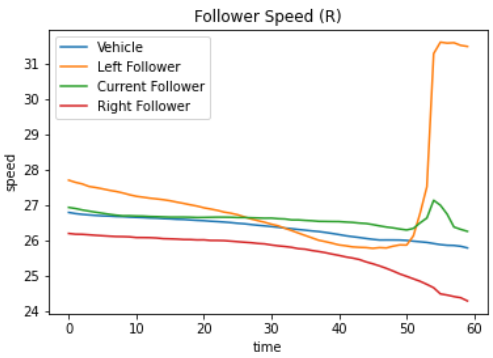

Leader Relative Speed (L)
speed
time
Left Leader
Current Leader
Right Leader


Leader Relative Speed (R)
speed
time
Left Leader
Current Leader
Right Leader


Follower Distance (L)
distance
time
Left Follower
Current Follower
Right Follower


Follower Distance (R)
distance
time
Left Follower
Current Follower
Right Follower


Follower Speed (L)
speed
time
Vehicle
Left Follower
Current Follower
Right Follower


Follower Speed (R)
speed
time
Vehicle
Left Follower
Current Follower
Right Follower

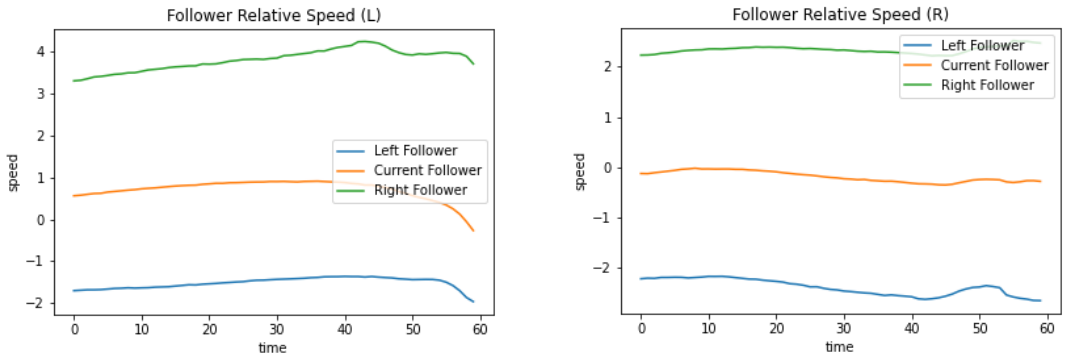


Figure 4.2 Average neighboring vehicle speed and distance

From an empirical point of view, the reason for turning left is usually overtaking or lane reduction. As can be seen from the picture above, the speed of the car will slowly increase before changing lanes, while the speed of the car blocking it does not increase significantly and may even be slowing down. This leads to the closer and closer distance between the car and the leader. Before the car changed lanes to the left, the car had to slow down to maintain a safe distance because it was too close to the car in front. From the collected data, it can also be seen that the relative distance between the follower in the same lane and the car remains basically constant. Due to the acceleration of the car, its distance from the leader on the left gradually decreases, and the distance from the follower on the left lane gradually increases. The above observations are consistent with what the driver would encounter before overtaking in actual road conditions.

However, from the figures on the right, it can be seen that the relationship between the leader and the follower is relatively stable until the car wants to change lanes to the right. This also corresponds to the situation in actual road conditions. In actual roads, vehicles change lanes to the right generally because they need to return to the original lane or leave the current road after overtaking. The above reasons have little to do with the operation of the surrounding vehicles. This can lead to difficulties in making predictions to the right when using machine learning to make predictions.

4.1.3 Machine Learning Input

With the simulation results processed, the data should be transformed into a structure that is suitable for the machine learning algorithm. The algorithms take a series of vehicle information as input. The input series lasts for 10 seconds with one sample per second that is 200 parameters in total is inserted to the machine learning algorithms. The output is whether the vehicle will change its lane in the next second. The possible output are the following three labels: label 0, label 1 and label 2. Label 0 means the vehicle will travel along the original lane; label 1 means the vehicle is going to change to its left lane for the next second; label 2 means the vehicle will change to the right lane in one second.

From the available data, the most straightforward is the label 1 and label 2 data. Those two types of data are obtained with the information collected from 10 seconds before the lane change event to the second before changing. For the label 0 data, they are sampled each 5 second. The collected data are separated into 4 label 0 input data and 1 label 1 or 1 label 2 input data that is 5 machine learning input data are extracted from each lane change event. In the end, there are 313908 label 0, 32092 label 1 and 46385 label 2 input data that are used for the machine learning process.

The dataset is separated into training set and test set. 30% of the data are randomly put into the test set. The training set is further split into two parts. One with 70% is used for the training process and the other, called validation set, is used for model selection when tuning hyperparameters.

4.2 Machine Learning Models

A common problem when working with deep learning projects is optimizing the hype-parameters. In deep learning, there are many parameters that need to be tuned to find a relatively good model. Some of the parameters are shared among the three models. The batch size is set to 10 for each model to get a model with better accuracy. The optimizer used is Adam. Adam stands for adaptive moment estimation which is one of the most

popular optimizers in the field of deep learning. It has fast convergence speed and is easy to adjust parameters. The learning rate is set to 0.001 after a few trials and adjustments. A higher learning rate leads to the machine learning model not learning and a lower learning rate slows down the learning process greatly without improving model performance.

CNN

One principle used for determining the network structure is to keep the network as small as possible on the premise of keeping the model performance high. The final CNN model has two one-directional convolutional layers with 40 filters of size 5 on the first layer and 20 filters of size 3 on the second layer. No pooling layer is added after the second convolutional layer with the consideration that the input data are not that bulky, using a second pooling layer may result in the model drop too much information and hard to make a correct prediction. The network structure is shown in Figure 4.3. The CNN model is trained with 100 epochs. The training of each epoch takes about 77 seconds.

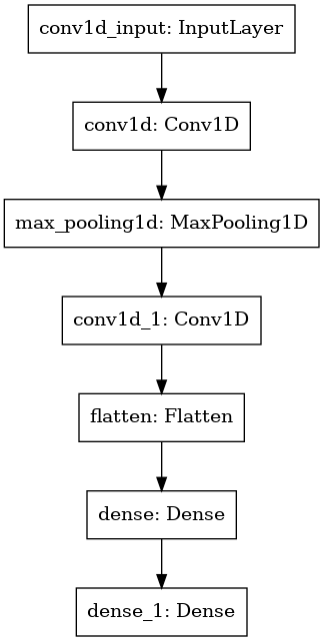


Figure 4.3 CNN model structure

RNN

The model consists of two SimpleRNN layers with 64 units in the first layer and 32 units in the second layer. RNNs with single SimpleRNN layer are also tested. But the performance is not stable and the accuracy is not as good. Therefore, a deep RNN is used instead of a single-layer RNN. A dropout of 0.3 is added before the output layer to prevent overfitting. Dropout in neural network training discards a part of neurons randomly with a certain probability to simplify the network and increase the robustness of the network. The structure of the RNN model is shown in Figure 4.4. The RNN model is trained for 100 epochs while each epoch takes around 155 seconds to train.

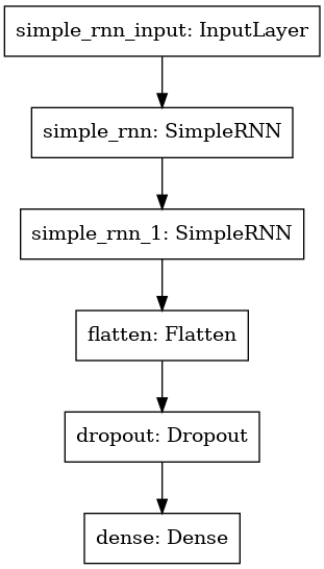


Figure 4.4 RNN model structure

LSTM

After tuning of hyperparameters, the LSTM model is composed of two LSTM layers followed by the output dense layer at the end. The model is trained with dropout rate 0.3. The two LSTM layers contains 128 and 64 units each. Figure 4.5 shows the LSTM network structure. The LSTM model is obtained after 40 epochs of training. Training of each epoch takes 440 seconds when no acceleration is enabled and reduced to 115 seconds with GPU acceleration.

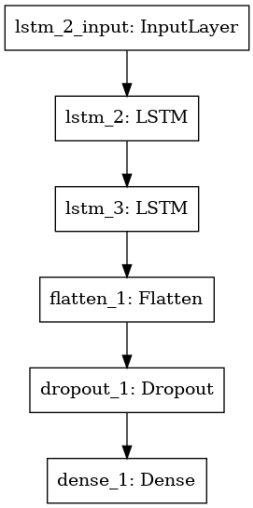


Figure 4.5 LSTM model structure

4.3 Model Evaluation

The three deep learning models are trained with training set, selected with validation set and tested with test set. Their performances in terms of the metrices mentioned in chapter 3 are listed in Table 4.3.

Model	Train_acc	Val_acc	Test_acc	Macro F1-score	MCC
CNN	0.8817	0.8725	0.8730	0.74	0.59
RNN	0.8805	0.8712	0.8693	0.75	0.59
LSTM	0.9005	0.8874	0.8864	0.78	0.65

Table 4.3 Model performace

The confusion matrix for each model in test set are shown in the following tables. Figure 4.6 shows the classification report of those three models.

CNN		Predicted Class		
		0	1	2
Actual Class	0	89713	2068	2381
	1	2685	6977	0
	2	7819	1	6072

Table 4.4 Confusion Matrix of CNN model

RNN		Predicted Class		
		0	1	2
Actual Class	0	87937	1725	4500
	1	2911	6749	2
	2	6242	1	7649

Table 4.5 Confusion Matrix of RNN model

LSTM		Predicted Class		
		0	1	2
Actual Class	0	88636	2013	3513
	1	1951	7708	3
	2	5860	1	8031

Table 4.6 Confusion Matrix of LSTM model

CNN Model:

	precision	recall	f1-score	support
Unchange	0.90	0.95	0.92	94162
Turn Left	0.77	0.72	0.75	9662
Turn Right	0.72	0.44	0.54	13892
accuracy			0.87	117716
macro avg	0.79	0.70	0.74	117716
weighted avg	0.86	0.87	0.86	117716

RNN Model:

	precision	recall	f1-score	support
Unchange	0.91	0.93	0.92	94162
Turn Left	0.80	0.70	0.74	9662
Turn Right	0.63	0.55	0.59	13892
accuracy			0.87	117716
macro avg	0.78	0.73	0.75	117716
weighted avg	0.86	0.87	0.87	117716

LSTM Model:

	precision	recall	f1-score	support
Unchange	0.92	0.94	0.93	94162
Turn Left	0.79	0.80	0.80	9662
Turn Right	0.70	0.58	0.63	13892
accuracy			0.89	117716
macro avg	0.80	0.77	0.79	117716
weighted avg	0.88	0.89	0.88	117716

Figure 4.6 Classification reports of the models on test data

If we focus only on accuracy, all three models seem to make better predictions, but since the dataset is not balanced, we need to use other metrics to evaluate the performance of the models. From macro F1, we can learn that CNN and RNN get about the same model accuracy, and the LSTM model is more accurate than the other two.

From the confusion matrix, it can be learned that most of the maintained original lanes can be accurately identified. For the case of lane change to the left, these models can also accurately predict about three-quarters of the cases. However, for the right lane change, the models are not able to make accurate predictions. This is consistent with the previous analysis of the speed-distance relationship between the leader and the follower when changing lanes to the right.

The LSTM model has the best performance, but the LSTM model is much larger than the other two models. the size of the LSTM model is 1540kB, while the other models are only about 150kB. Due to the size of the model, it also causes the LSTM model to be slower than the other two models when computing.

4.4 Prediction Results

Figure 4.7 shows an example of the input data to the machine learning models. The first row refers to the vehicle states 10 seconds before the prediction moment. From the data, it can be told that the right lane neighbors don’t exist which may be due to the vehicle is traveling on the right most lane. Another observation is that the left lane neighbors are far away. Concentrating on the current lane the vehicle is traveling on, it can be concluded that the leader vehicle is blocking and the distance between the vehicle and its leader is getting smaller. Figure 4.8 and Figure 4.9 show the trend graphically. From the analysis, it is possible that the vehicle will overtake its leader therefore it may turn to the left lane in the next second. The predictions made by the three models are reported in Table 4.7. Each element in Table 4.7 refers to the probability of a model prediction on one certain

label. All the three models make strong prediction on label 1 which is turn left.

lane	speed	l_l	l_l_speed	l_l_dist	l_f	l_f_speed	l_f_dist	c_l	c_l_speed	c_l_dist	c_f	c_f_speed	c_f_dist	r_l	r_l_speed	r_l_dist	r_f	r_f_speed	r_f_dist
0	31.68	1	0.8	480.25	1	-3.72	-700.41	1	-3.85	84.56	1	-3.74	-194.65	0	0	0	0	0	0
0	30.77	1	2.11	482.37	1	-2.95	-703.35	1	-2.8	81.77	1	-2.73	-197.37	0	0	0	0	0	0
0	31.03	1	1.86	484.23	1	-3.16	-706.51	1	-3.1	78.67	1	-3.01	-200.38	0	0	0	0	0	0
0	31.79	1	0.66	484.88	1	-3.91	-710.42	1	-3.94	74.73	1	-3.21	-203.59	0	0	0	0	0	0
0	31.2	1	1.55	486.43	1	-3.71	-714.13	1	-3.43	71.3	1	-2.66	-206.25	0	0	0	0	0	0
0	31.32	1	1.76	488.2	1	-3.74	-717.87	1	-3.53	67.77	1	-3.2	-209.45	0	0	0	0	0	0
0	30.78	1	1.99	490.18	1	-2.79	-720.66	1	-3.1	64.67	1	-2.77	-212.22	0	0	0	0	0	0
0	29.94	1	2.47	492.66	1	-2.18	-722.83	1	-2.44	62.23	1	-1.4	-213.61	0	0	0	0	0	0
0	29.47	1	3.5	496.17	1	-1.74	-724.56	1	-1.87	60.37	1	-0.69	-214.29	0	0	0	0	0	0
0	30.41	1	1.66	497.83	1	-2.87	-727.43	1	-2.47	57.9	1	-1.89	-216.18	0	0	0	0	0	0

Figure 4.7 Machine Learning model input data example

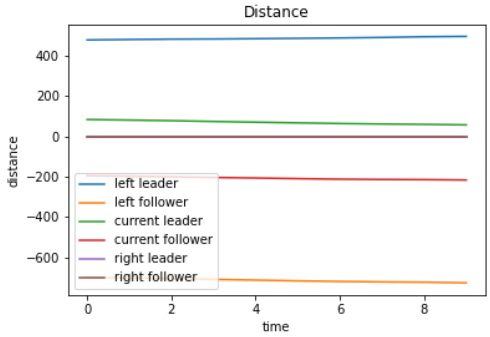


Figure 4.8 Relative distance between vehicle and its neighbors

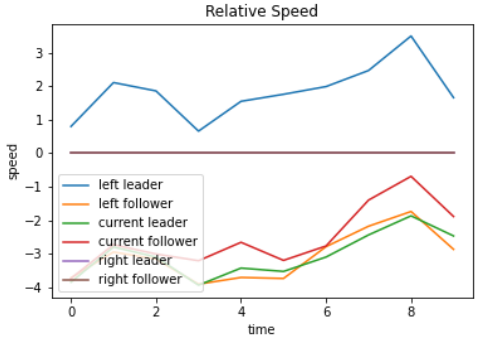


Figure 4.9 Relative Speed between vehicle and its neighbors

Class	0	1	2
CNN	6.60e-2	9.34e-1	3.85e-7
RNN	6.57e-2	9.34e-1	1.18e-5
LSTM	1.16e-2	9.88e-1	2.81e-6

Table 4.7 Model prediction results

Chapter 5 Conclusion and Future Work

Under the idea of centralized cooperative maneuver management, three lane change prediction models based on vehicle speed and relative distance are proposed in this thesis. We proposed three different ANN models which have been trained with a dataset built from SUMO to predict the probability of vehicles changing lanes. Then, the model performance is evaluated and analyzed. The specific conclusions of this thesis are summarized as follows.

- A dataset was built based on the simulation results of SUMO, including three different types of lane change cases: no lane change, lane change to the left and lane change to the right. The total number of data of different labels is 392385.
- Three deep learning models that can make predictions based on the distance and speed of the reference vehicle and its surrounding cars are proposed. These three models were trained on the collected dataset. Generally speaking, the LSTM model has the best performance, followed by the CNN model, and the RNN model has no advantage in terms of training speed and model accuracy, which is seen as a result of the lack of long-term memory of the RNN.

The problem of changing lanes and overtaking is one of the most complex problems in the field of autonomous driving. Although it has been discussed by many researchers [39-42], there are still many challenges in the highly automated maneuver management. Using deep learning and V2X based centralized cooperative maneuver approaches, the lane-changing problem can be effectively predicted and managed. It can be used as an

alternative solution rather than having the vehicles detect the environment and make the lane change decision by itself.

This thesis discusses this problem, related work, and implementation options for deep learning systems. Several aspects which can be considered as future work:

- The dataset size is quite small for a complex problem like the one discussed in this thesis. Collecting more simulation data or even actual road data can improve the prediction accuracy.
- In the analysis of simulation data and test results, it is found that the models have some difficulties in predicting merge to the right maneuvers.
- The dataset does not take into account the actual road conditions.
- The data is sampled at only 1 Hz; in future work, higher frequencies should be tested.

Bibliography

[1] WHO. "Road traffic injuries." World Health Organization Web. https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries (accessed 2022-09-20).

[2] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_202104; Scope*, S. International, United States, 2021.04.30 2021.

[3] Synopsys. "LEVELS OF DRIVING AUTOMATION." https://www.synopsys.com/automotive/autonomous-driving-levels.html (accessed 2022-09-13).

[4] Y. Liu, X. Wang, L. Li, S. Cheng, and Z. Chen, "A Novel Lane Change Decision-Making Model of Autonomous Vehicle Based on Support Vector Machine," *IEEE Access,* vol. 7, pp. 26543-26550, 2019, doi: 10.1109/access.2019.2900416.

[5] M. Wang, S. Hoogendoorn, W. Daamen, B. van Arem, and R. Happee, "Optimal lane change times and accelerations of autonomous and connected vehicles," in *Transportation Research Board 95th Annual Meeting*, 2016: Transportation Research Board (TRB), pp. 16-2914.

[6] Y.-C. Lin, C.-L. Lin, S.-T. Huang, and C.-H. Kuo, "Implementation of an Autonomous Overtaking System Based on Time to Lane Crossing Estimation and Model Predictive Control," *Electronics,* vol. 10, no. 18, 2021, doi: 10.3390/electronics10182293.

[7] A. Mehmood, M. Liaquat, A. I. Bhatti, and E. Rasool, "Trajectory planning and control for lane-change of autonomous vehicle," in *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, 2019: IEEE, pp. 331-335.

[8] Y. Zhu, M. Feng, X. Wang, and X. Xu, "Research on intelligent vehicle autonomous overtaking based on single neuron PID control," in *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, 2012, vol. 3: IEEE, pp. 1192-1195.

[9] J. E. Naranjo, C. Gonzalez, R. Garcia, and T. de Pedro, "Lane-Change Fuzzy Control in Autonomous Vehicles for the Overtaking Maneuver," *IEEE Transactions on Intelligent Transportation Systems,* vol. 9, no. 3, pp. 438-450,

2008, doi: 10.1109/tits.2008.922880.
[10] Z. Wang, S. Cui, and T. Yu, "Automatic Lane Change Control for Intelligent Connected Vehicles," presented at the 2019 4th International Conference on Electromechanical Control Technology and Transportation (ICECTT), 2019.
[11] W. Fenghui, Y. Ming, and Y. Ruqing, "Conflict-Probability-Estimation-Based Overtaking for Intelligent Vehicles," *IEEE Transactions on Intelligent Transportation Systems,* vol. 10, no. 2, pp. 366-370, 2009, doi: 10.1109/tits.2009.2020200.
[12] P. Petrov and F. Nashashibi, "Modeling and nonlinear adaptive control for autonomous vehicle overtaking," *IEEE Transactions on Intelligent Transportation Systems,* vol. 15, no. 4, pp. 1643-1656, 2014.
[13] A. Alalewi, I. Dayoub, and S. Cherkaoui, "On 5G-V2X Use Cases and Enabling Technologies: A Comprehensive Survey," *IEEE Access,* vol. 9, pp. 107710-107737, 2021, doi: 10.1109/access.2021.3100472.
[14] M. Recommendation, "2084: Radio interface standards of vehicle-to-vehicle and vehicle-to-infrastructure communications for Intelligent Transport System applications," *Recommendations, ITU-R (Sep 2015),* 2015.
[15] A. Papathanassiou and A. Khoryaev, "Cellular V2X as the essential enabler of superior global connected transportation services," *IEEE 5G Tech Focus,* vol. 1, no. 2, pp. 1-2, 2017.
[16] S. Eckl, D. Krefft, and U. Baumgarten, "KIA4SM-Cooperative integration architecture for future smart mobility solutions," in *Conference on Future Automotive Technology (CoFAT)*, 2015, vol. 4, p. 04.
[17] "5G-CARMEN D2.1, 5G CARMEN Use Cases and Requirements." https://5gcarmen.eu/wp-content/uploads/2020/03/5G_CARMEN_D2.1_FINAL.pdf (accessed 2022-09-13).
[18] B. Hafner, J. Sauerhammer, G. A. Schmitt, and J. Ott, "Performance of Cooperative Maneuver Protocols in Real-World Automated Vehicles," presented at the ICC 2022 - IEEE International Conference on Communications, 2022.
[19] F. Raviglione, C. M. R. Carletti, C. Casetti, F. Stoffella, G. M. Yilma, and F. Visintainer, "S-LDM: Server Local Dynamic Map for Vehicular Enhanced Collective Perception," presented at the 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), 2022.
[20] A. Correa *et al.*, "Infrastructure support for cooperative maneuvers in connected and automated driving," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019: IEEE, pp. 20-25.
[21] P. A. Lopez *et al.*, "Microscopic traffic simulation using sumo," in *2018 21st international conference on intelligent transportation systems (ITSC)*, 2018: IEEE, pp. 2575-2582.
[22] eclipse.org. "Rawdump." https://sumo.dlr.de/docs/Simulation/Output/RawDump.html (accessed 2022-09-13).

[23] eclipse.org. "Lanechange." https://sumo.dlr.de/docs/Simulation/Output/Lanechange.html (accessed 2022-09-13).
[24] eclipse.org. "SUMO User Documentation." https://sumo.dlr.de/docs/ (accessed 2022-09-13).
[25] A. C. Müller and S. Guido, *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.
[26] D. Silver *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815,* 2017.
[27] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Min Knowl Discov,* vol. 31, no. 3, pp. 606-660, 2017, doi: 10.1007/s10618-016-0483-9.
[28] D. F. Silva, R. Giusti, E. Keogh, and G. E. A. P. A. Batista, "Speeding up similarity search under dynamic time warping by pruning unpromising alignments," *Data Mining and Knowledge Discovery,* vol. 32, no. 4, pp. 988-1016, 2018, doi: 10.1007/s10618-018-0557-y.
[29] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters,* vol. 42, pp. 11-24, 2014, doi: 10.1016/j.patrec.2014.01.008.
[30] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery,* vol. 33, no. 4, pp. 917-963, 2019.
[31] User:Dhp1080. "Neuron description." "Anatomy and Physiology" by the US National Cancer Institute's Surveillance, Epidemiology and End Results (SEER) Program. https://zh.m.wikipedia.org/wiki/File:Neuron.svg (accessed 2022-10-01).
[32] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2019.
[33] Glosser.ca. "Artificial neural network with layer coloring." https://en.wikipedia.org/wiki/File:Colored_neural_network.svg (accessed 2022-09-13).
[34] A. Baldominos, Y. Saez, and P. Isasi, "Evolutionary design of convolutional neural networks for human activity recognition in sensor-rich environments," *Sensors,* vol. 18, no. 4, p. 1288, 2018.
[35] J. Prassanna, K. K. Nawas, J. C. Jackson, R. Prabakaran, and S. Ramanathan, "Towards building a neural conversation chatbot through seq2seq model," *International Journal of Scientific and Technology Research,* vol. 9, no. 3, pp. 1219-1222, 2020.
[36] C. Olah. "Understanding LSTM Networks." http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed 2022-09-13).
[37] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887,* 2017.
[38] M. Guerrieri, R. Mauro, A. Pompigna, and N. Isaenko, "Road Design Criteria and Capacity Estimation Based on Autonomous Vehicles Performances. First Results

from the European C-Roads Platform and A22 Motorway," *Transport and Telecommunication Journal,* vol. 22, no. 2, pp. 230-243, 2021, doi: 10.2478/ttj-2021-0018.

[39] Ü. Dogan, J. Edelbrunner, and I. Iossifidis, "Autonomous driving: A comparison of machine learning techniques by means of the prediction of lane change behavior," in *2011 IEEE International Conference on Robotics and Biomimetics*, 2011: IEEE, pp. 1837-1843.

[40] M. Bahram, C. Hubmann, A. Lawitzky, M. Aeberhard, and D. Wollherr, "A combined model-and learning-based framework for interaction-aware maneuver prediction," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, no. 6, pp. 1538-1550, 2016.

[41] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013: IEEE, pp. 797-802.

[42] D. Kasper *et al.*, "Object-oriented Bayesian networks for detection of lane change maneuvers," *IEEE Intelligent Transportation Systems Magazine,* vol. 4, no. 3, pp. 19-31, 2012.