國立臺灣大學電機資訊學院資訊網路與多媒體研究所

碩士論文

Graduate Institute of Networking and Multimedia

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

利用遞歸神經網路和注意力機制來預測不同場景下車輛的速度

Vehicle speed prediction with RNN and attention model under different semantics

黃柏瑋

Po-wei Huang

指導教授：施吉昇 博士

Advisor: Chi-Sheng Shih, Ph.D.

中華民國 107 年 7 月

July, 2018

doi:10.6342/NTU201803784

國立臺灣大學碩士學位論文
口試委員會審定書

利用遞歸神經網路和注意力機制來預測不同場景下車輛的速度

Vehicle Speed Prediction with RNN and Attention Model under Different Semantics

本論文係黃柏瑋君（學號 R05944024）在國立臺灣大學資訊網路與多媒體研究所完成之碩士學位論文，於民國一百零七年七月廿七日承下列考試委員審查通過及口試及格，特此證明

口試委員： [illegible] （簽名）

（指導教授）

林風 [illegible]

所　長： 楊佳玲

doi:10.6342/NTU201803784

Acknowledgments

I would like to express my gratitude to Dr.Shi for his great effort of supervising my master thesis. Without his help, I probably couldn’t finish my master thesis. Thanks for his effort of collecting the dataset and advising my project. He gave me so many ideas and he used great patience to let me explore.

Moreover, I want to express my gratitude toward committee members. Thanks the committee member for reviewing my thesis and spend their time in oral exam.

Then, I want to thank my friends in lab and schools. Special thanks to Olivier, 君哲, 雅元 for spending their time reviewing my thesis. Without your help, I probably couldn’t pass my oral.

Lastly, I appreciate my parents and sister for their continuous support.

Thanks you!

Taipei, July, 2018

Po-wei Huang

doi:10.6342/NTU201803784

摘要

本論文旨在利用預測周遭車輛的速度來達成防衛性駕駛。要達成防衛性駕駛，首先要知道周遭車的意圖和預測其速度和軌跡。但是，既有的預測機制有幾個問題。第一，使用了很多資訊，像是前前車的速度，來預測前車的速度。這樣的資訊，實際上是沒有的。第二，既有的模型無法從資料中抓出子序列的規律。第三，既有的模型通常只鎖定單一模型，而沒有分析在不同路段或駕駛類型的影響。面對這些問題，本文提出了這些改進。第一、使用單台車的資料，來做預測，並探索其適用範圍。第二、利用注意力機制來抓出子序列的規律來提升預測準度。第三、分出不同的場景，如轉彎、直行、換車道，和道路的型態，並找尋模型最適合的範圍。以結果來說，相比於既有卡爾曼濾波的作法，利用遞歸神經網路和注意力機制，本文達到了11%的提升。此外，透過和多台車模型的比較，驗證單台車模型能適用於直行的預測。此外，在接近自由駕駛時，預測會更佳。

關鍵字- 車輛、軌跡、速度、場景、遞歸神經網路、注意力機制 -

doi:10.6342/NTU201803784

Abstract

In this work, we focus on vehicle velocity prediction for defensive driving. In order to achieve defensive driving, we need to understand and predict the motion of surrounding cars. However, the existing velocity prediction mechanism has three drawbacks. Firstly, they need too much information for prediction and some information are not accessible actually. Secondly, they failed to extract sub-patterns from the training data and thus may cause inaccuracy. Thirdly, only very few scenarios are considered. To deal with these problems, we start with single car model for prediction and explore its limitation by comparing it with multi-car model. Then, we use attention model to increase the accuracy. Finally, we examine the prediction under different road types and maneuver to broaden our scenario. Overall, with attention mechanism, we achieve a 11% improvement over existing solution. Moreover, we found that single car model could be used in go straight behavior, and it has best result in free driving periods.

Keywords - vehicle, trajectory prediction, velocity prediction, recurrent neural network, attention, semantics.

doi:10.6342/NTU201803784

Contents

doi:10.6342/NTU201803784

doi:10.6342/NTU201803784

doi:10.6342/NTU201803784

List of Figures

doi:10.6342/NTU201803784

doi:10.6342/NTU201803784

List of Tables

doi:10.6342/NTU201803784

Chapter 1

Introduction

1.1 Motivation

1.1.1 Motion prediction for defensive driving

Motion prediction is an essential element for defensive.Below is an scenario of aggressive cut-in which shows the importance of motion prediction. As the figure shows, the car on the right want to change lane and cut in. The ego vehicle has to slow down accordingly.

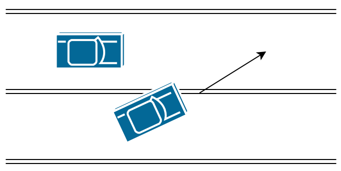


Figure 1.1: Aggressive cut in

According to [11], the ego vehicle need to do these steps in order to drive defensively. The step is also shown in the following figure.

- Detect the intention of right car to cut in.
- Predict the cut in trajectory and velocity in longitudinal and latitude direction.
- Generate a trajectory for ego vehicle according to front car.

doi:10.6342/NTU201803784

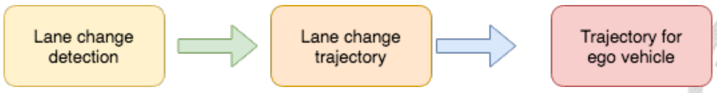


Figure 1.2: Lane change detection and defense

From this example, we could understand the importance of motion prediction. Figure 1.3 describe high-level theory about how to predict motion. Basically, we have to know his situation, and use our driving models to infer what would he do. With this in mind, figure 1.4 lays out the overall scope of motion prediction. To understand the situation, we need real-time sensing and information collection. Then, we need a driving model to make prediction. However, previous mechanism has some problems when applied to defensive driving.

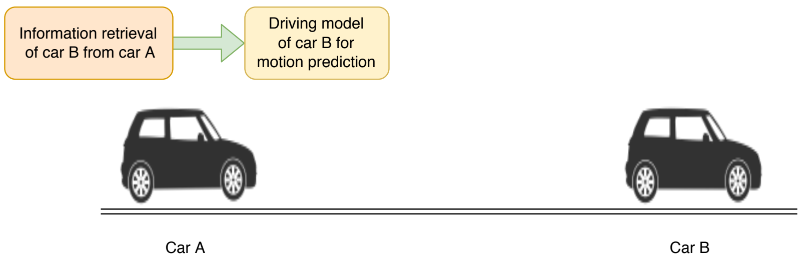

Information retrieval of car B from car A
Driving model of car B for motion prediction
Car A
Car B

Figure 1.3: Element of motion prediction

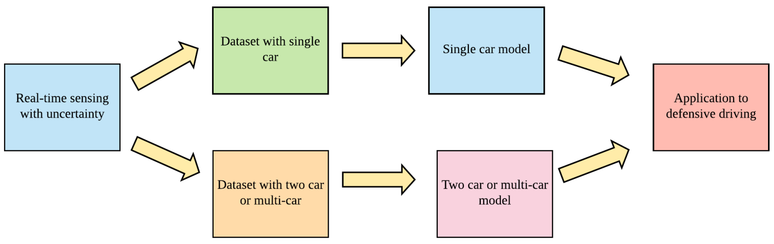

Real-time sensing with uncertainty
Dataset with single car
Dataset with two car or multi-car
Single car model
Two car or multi-car model
Application to defensive driving

Figure 1.4: Overall scope of motion prediction

1.1.2 Problems of existing prediction mechanism for defensive driving

While there exists many motion prediction methods, we found several problems when it's applied to defensive driving.

doi:10.6342/NTU201803784

- Some model used many information that's actually not available. For instance, [1] use the velocity of front front car to predict the motion of front car. While in defensive driving, we don't have that.

- Failed to consider the influence of velocity prediction on safety distance. While the velocity is important for safety distance calculation, some model only predict the occupancy map for trajectory and it's hard to generate velocity as the grid size of the map is too big.

- Some models couldn't find patterns within subsequence or rely on additional information to classify the pattern in subsequences.

- Model not trained and tested on various scenario. Many works only trained and tested on the track they design or on dataset with only one or two scenario.

1.2 Our solution and result

We propose a solution for each of the problem. To tackle the challenge of limited input, we start with a single car model and compare it with the multi-car model to find out its limitation. As for the subpattern, we use attention model to extract the pattern in subsequence and thus get a 11% improvement over kalman filter and a 8% improvement over encoder-decoder. Moreover, we output speed directly to reduce the problem coming from occupancy map. Lastly, we use a dataset that contain multiple scenario and evaluate the precision under different scene. As 1.5 shows, our focus in this thesis would mostly about single car model.

1.3 Contribution

- Use attention model to deal with the patterns in subsequecne

- Consider the problem of multiple scenario in motion prediction. Specifically, this work use the gyro data , map service , and average speed to get comprehensive

doi:10.6342/NTU201803784

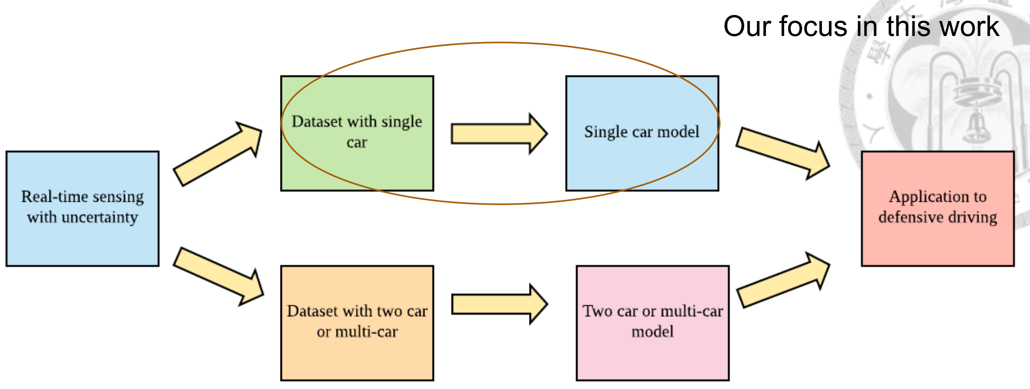

Our focus in this work
Real-time sensing with uncertainty
Dataset with single car
Single car model
Dataset with two car or multi-car
Two car or multi-car model
Application to defensive driving

Figure 1.5: Our focus in this thesis

scenario.

- Consider the problem of existing motion prediction when applied to defensive driving and explore the limitation of single car model.

1.4 Thesis Organization

The remainder of this paper is organized as follows. Chapter 2 contains background readers to understand this paper. Chapter 3 contains the software architecture and problem definition.

Chapter 4 describes our design and implementation. Lastly, chapter 5 describes our methodology and our experiment result. chapter 6 serves as a conclusion.

doi:10.6342/NTU201803784

Chapter 2

Background and Related Work

2.1 Background

2.1.1 Data preprocessing

Map service and openstreetmap

Openstreetmap[7] is a community effort for map service. The database contains many geographical information like address, road name, and road type.

2.1.2 Prediction model

encoder-decoder

Encoder-decoder[6] use LSTM to encode the sequence into a fixed vector. Then, the vector is fed into a LSTM to generate a sequence. It's suitable for next step prediction. The advantage of this model compared to kalman filter is that it could has a lot of memory and thus could accommodate various pattern.

encoder-decoder with attention

Although encoder-decoder is powerful, it has some shortage[4]. Encoder-decoder couldn't deal with the alignment problem. In machine translation , alignment is the problem in machine translation that identifies which parts of the input are relevant to the out-

doi:10.6342/NTU201803784

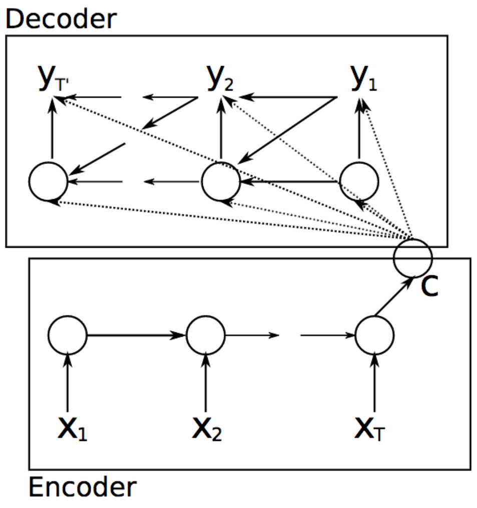


Figure 2.1: Architecture of encoder-decoder[6].

put, whereas translation is the process of using the relevant information to select the appropriate output. Instead of encode the entire sequence into a vector, attention mechanism encode each subsequence into a context vector. With the context vectors, it could generate better prediction because of the subpattern.

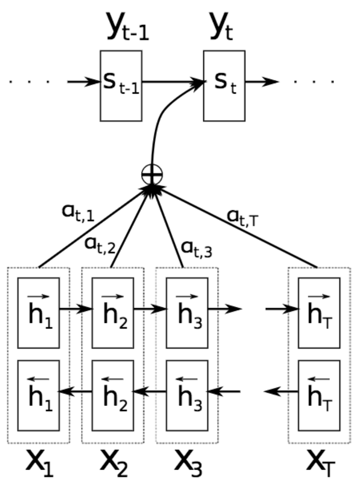


Figure 2.2: Architecture of encoder-decoder with attention model[4].

doi:10.6342/NTU201803784

2.2 Related Work

2.2.1 Previous study on motion prediction

In [1], they used a LSTM network with residual and convolution to produce the prediction. However, the input contains information of all the surrounded car, so it's different from our scenario. In addition, in [12] , they used the encoder-decoder architecture to predict the trajectory in occupancy map. However, this work didn't use attention model, and it only used beam search to enhance their recall. Besides, in [2], they used kalman filter to predict and couldn't accommodate for multiple trajectories.

2.2.2 Previous study on time series prediction

Trajectory prediction is related to time series prediction. In [13], they used a dual-stage attention model to improve their prediction on financial data. While it's attractive to import this result directly to trajectory prediction, there are some difference in financial data and trajectory. For trajectory, maneuver and driver's decision plays a heavy role, but for financial data, factors are more complicated and thus require more powerful and deep model to learn the representation directly.

2.2.3 Previous study using road semantics

Using semantics is a native step in many geographical problems. In [14], they also used openstreetmap to retrieve the road, but the target is different. They wanted to do destination prediction, while our focus is on short-term trajectory prediction. Moreover, they use the road name as features directly, while we use speed limit as feature.

doi:10.6342/NTU201803784

Chapter 3

System architecture and problem definition

In this chapter, we describe our target problem, approach , problem definition, and system architecture.

3.1 Target problem

In the following section ,we describe the problems of existing prediction mechanism in detail.

3.1.1 Challenge I : Some model used many information that's actually not available

Many methods use information that's actually not available in defensive driving for prediction. For instance, [1] used information from surrounding cars to predict the motion of ego vehicle. In real-world scenario, we couldn't see the car in front of front car, and thus couldn't utilize this kind of model. Actually, all interaction-based models suffer from this problem and thus not applicable to our application. In other words, we could only utilize the speed and acceleration of front car to do prediction. This is a very limited input for prediction and thus incur us to use more semantics or better prediction model.

doi:10.6342/NTU201803784

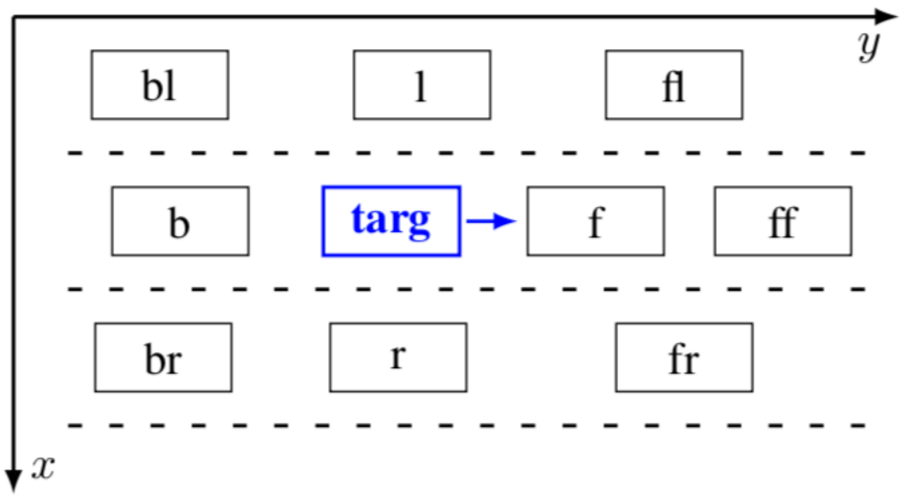


Figure 3.1: multi car model[1]

3.1.2 Challenge II : Fail to find patterns within subsequence

Sometimes, the driver breaks at the last second. So, only the last part of training sequence might be helpful. Moreover, subpatterns may exist inside sequence, as shown in Figure 3.2 . Some model [12] use encoder-decoder, which fails to deal with this. Other model[2] use kalman filter and also fail to track this due to insuffciency of latent variable.

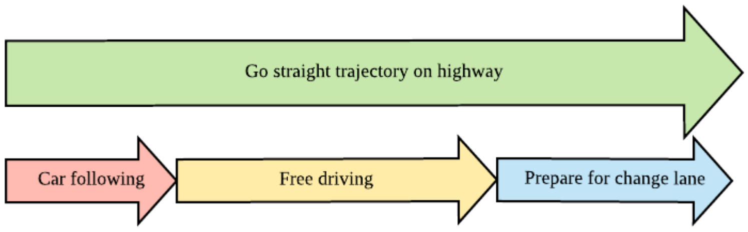


Figure 3.2: Subpattern inside a trajectory.

3.1.3 Challenge III : Models not trained and tested on various scenario

Driver will display different behavior on different scenario. For example, in highway and intersection, driver will drive very different due to factors like traffic jam and speed

doi:10.6342/NTU201803784

limit. Moreover, in different maneuver such as go straight, change lane, free driving, car following, driver will also have different behavior. Previous studies [2][12] only tested it in one or two scenarios and didn't explore its difference.

3.2 Our approach for these problems

3.2.1 Our approach for challenge I and II

To deal with the problem, we have to limit the input of our model. So, we formally define the information that's available for ego vehicle in defensive driving.(single car model) Then, we examine the difference between single car model and multi-car model.

The single car model

When a car follows another car, we want to predict the velocity of front car. We assume that the sensors could accurately know the velocity and trajectory of front car. Besides, from the camera, it could know the maneuver of front car. In summary, the following car could know the information of front car and need to make prediction by this:

- velocity and trajectory of front car and self car.
- maneuver of front car.
- other semantic information like road information and traffic information.

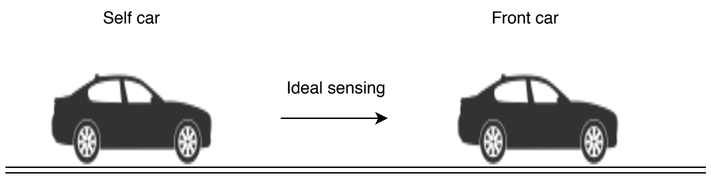


Figure 3.3: Single car model

doi:10.6342/NTU201803784

This model is simple and useful for defensive driving, but it also has its limitation. So, we need to compare it with multi-car model to validate its applicability.

Moreover, for problem II, we abandon occupancy method and predict the velocity directly. As for problem III, we need to use a model to extract the subpattern.

3.2.2 Our approach for challenge III

For this problem, we use a dataset that contains many scenario like maneuver class, road type, and average speed. Then, we cut the data by scenario and explore the accuracy for each class.

3.3 Problem Definition

3.3.1 Motion prediction

Given the velocity of the target car X1,X2,X3,....,Xn in the history, we want to predict the future velocity Z1,Z2,Z3,....,Zn. In this problem, accuracy would be main concern. The mean square error would be our metric for comparison. Moreover, we focus on short-term prediction, like prediction for next T seconds. We want to find model that minimize the prediction error. Moreover, due to the application, we focus on model that only uses the information of ego vehicle. Moreover, we need a model that could find the pattern in subsequences.

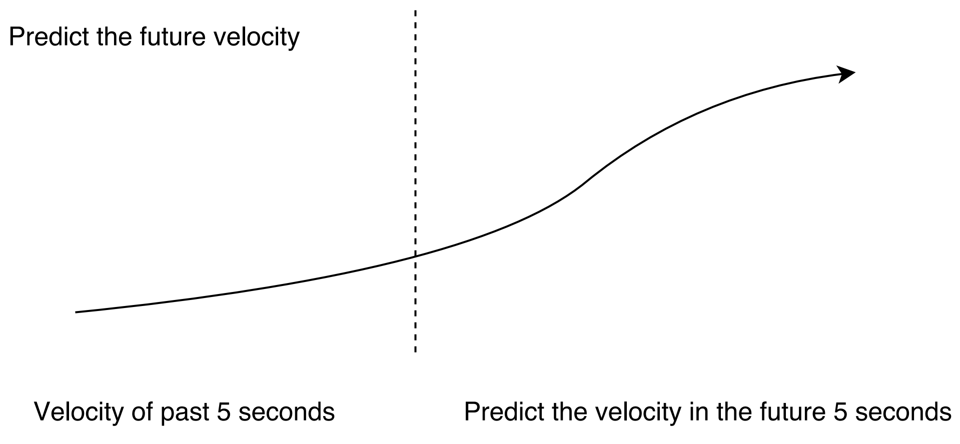

Predict the future velocity
Velocity of past 5 seconds
Predict the velocity in the future 5 seconds

Figure 3.4: Illustration for problem definition.

doi:10.6342/NTU201803784

3.3.2 Requirement of motion prediction

After the problem definition, we need to qualitatively define our requirement of T, as the prediction horizon will influence the choice of seq2seq model. A complicated model might be good for long-term prediction, but when we only need a short-term prediction, it will be an overkill. We set our prediction horizon to 5s due to following reasons:

- Driving is a series of event and our prediction should contain a full event for solid validation. According to [8], the mean lane change time of PKU and UAH are 6.8s and 5.9s respectively. So our prediction should at least be 5s.
- The short term prediction would be hard to cover two event, unless we know the road geometry and the position of all the surrounding cars in future. We couldn't know that a car will change lane or turn left or right after a go-straight event, unless we know there is an intersection. So, the prediction shouldn't be larger than 7s.
- According to [3], AEBS should issue at least one warning when TTC is equal to or less than 4.4s. So, it's important to predict for at least 4.4s.

3.3.3 Explore the limitation of single car model

As the surrounding cars might have a influence when the front car change lane or turn right, we're interested to know the limitation of single car model. So, we would like to compare the prediction made by multi-car model and single car model to know this.

3.3.4 Effectiveness of single car model under different scenario

In addition to model selection, we would like to explore the effectiveness of single car model under different scenario. For example, for each maneuver, the accuracy might be different. So, for each semantic Z, we want to explore the accuracy when the semantic has different values.

doi:10.6342/NTU201803784

3.4 System architecture

In this section, we first describe the existing system architecture for driving model prediction. Then, we describe the new system architecture after our work. As shown in figure 3.5, existing work like [1] divide the trajectory into sliding window and use seq2seq model for prediction. A natural enhancement would be use better seq2seq model for prediction. In the first part of our work use a new seq2seq model called attention model to achieve better result.

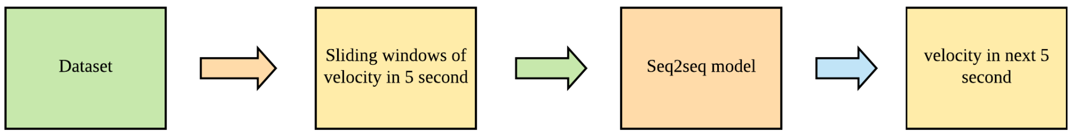

Dataset
Sliding windows of velocity in 5 second
Seq2seq model
velocity in next 5 second

Figure 3.5: Existing system architecture

Then, we add a data classifier in front of the sliding window generator. The will enable us to debug and enhance our prediction model. The classifier will divide the data based on maneuver, road type, average speed, as shown in figure 3.6. Finally, as shown in 3.7, we use NGSIM US-101 to evaluate the limitation of single model.

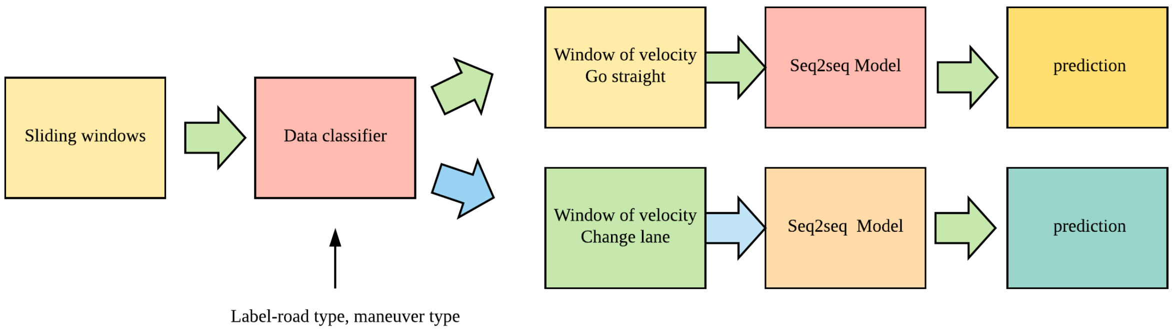

Sliding windows
Data classifier
Label-road type, maneuver type
Window of velocity Go straight
Seq2seq Model
prediction
Window of velocity Change lane
Seq2seq Model
prediction

Figure 3.6: System architecture with data classifier

doi:10.6342/NTU201803784

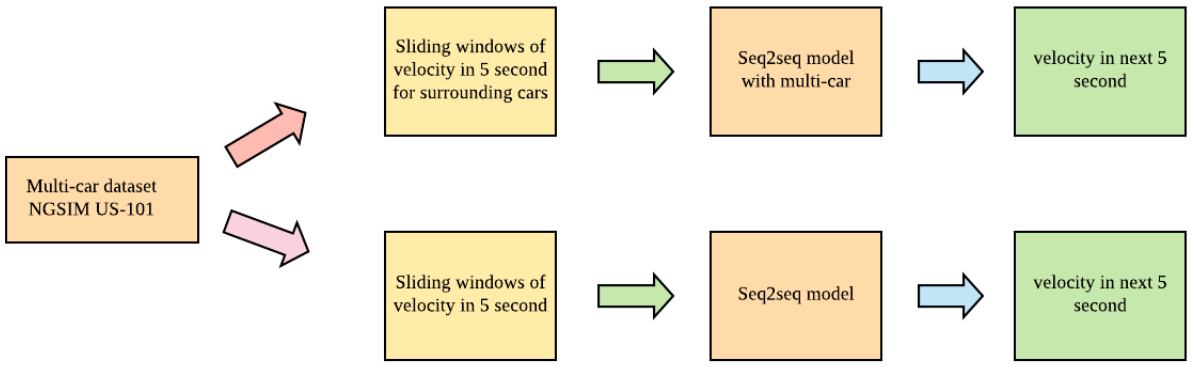

Multi-car dataset
NGSIM US-101
Sliding windows of velocity in 5 second for surrounding cars
Seq2seq model with multi-car
velocity in next 5 second
Sliding windows of velocity in 5 second
Seq2seq model
velocity in next 5 second

Figure 3.7: Comparison of single car and multi car model

doi:10.6342/NTU201803784

Chapter 4

Design and implementation

4.1 Implement existing system architecture for on-board sensing data

The first step of our design is to implement the existing system architecture from on-board sensing data. The key difference for OBD lies in the way to get sliding window from raw data. As shown in 4.1, after we get the sliding window, we could use any kind of seq2seq model for prediction. As we have many types of sensor including GPS, IMU, speedometer, we have to use sensor fusion to get the velocity. Moreover, as the amount of trajectories are so big and versatile, we need to use many data preprocessing skills to get the sliding window. Now, we would start from dataset and elaborate on how to get the sliding window.

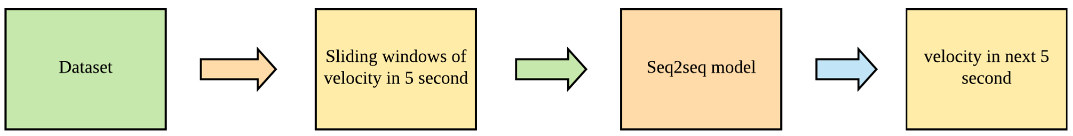

Dataset
Sliding windows of velocity in 5 second
Seq2seq model
velocity in next 5 second

Figure 4.1: Existing system architecture

doi:10.6342/NTU201803784

4.2 Dataset

Our dataset comes from the trajectory of 9 bus route. It's sampled in November and the route is in a region between Pleasanton and Livermore, CA. The trajectory includes highway, curve, turn, ramp behaviors as well as some local driving. The sensors on the bus include GPS, velocity meter, accelerometer and gyroscope.

The features includes:

- value from gyroscope in x,y, z direction in 10Hz
- acceleration in x,y,z direction in 10Hz
- latitude and longitude in 0.5Hz
- Speed from GPS in 0.5Hz
- Speed from velocity meter in 2Hz
- timestamp in unix time
- mac id for each bus

This dataset is rich in its features, and it contains several hundreds of routes for each bus. The time span of each route ranges from 2min to 20min, so the quantity is pretty large for validation and analysis. Besides, the road type is diverse, which enable us to use this for analysis. However, it also has some deficiency. For example, the diversity of bus is pretty small. Lastly, the data is stored in relational format like CSV or SQL.

4.3 From raw data to sliding window of velocity

After we get the raw data. The first step is to transform the raw data into trajectories of same length. Below, we describe our steps of transformation.

4.3.1 Group the dataset by bus id

As the raw data is in relational format, we need to get the data for each bus first.

doi:10.6342/NTU201803784

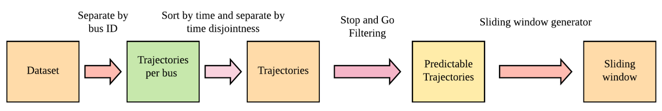


Figure 4.2: Pipeline from raw data to sliding window.

4.3.2 Group the dataset into trajectories by time disjointness

After the data is extracted, we need to transform it into trajectories. So, we sort it by time and cut it by time disjointness.

$$A\ cut\ exists\ between\ T_i\ and\ T_j,\ if\ \ \mid T_i - T_j \mid > 5s \tag{4.1}$$

4.3.3 Sensor fusion for tangential velocity

As the sampling rate of velocity from GPS is too slow, we need to use sensor fusion for tangential velocity. We have three sources of data to get tangential velocity. Firstly, velocity from GPS with 0.5Hz. Secondly, velocity from speedometer in 2Hz. Thirdly, acceleration from IMU in 10Hz. We need to combine these three data to get tangential velocity in 10Hz. To achieve this, we add the effect of acceleration on two velocity data. Then, we used weighted average to calculate the tangential velocity from two velocity. Moreover, if the ratio of two velocity is not clost to 1 or not consistent, we used heuristic validation to check which one might be wrong and only use one data directly. While we used a lot of method to ensure its correctness, the dataset doesn't have a groundtruth for tangential velocity in 10Hz. So, it's hard for us to check whether we're good enough in sensor fusion, but we have used our best effort on this.

4.3.4 Stop-and-go filtering

When we follow a car, the timing of front car to start and stop is hard to know. However, we could predict the movement after it's started. So, we need to filter our trajectories by a stop-and-go filtering. We utilize a technique called endpoint detection. When we move forward with sliding window, we calculate the root mean square of sliding window.

doi:10.6342/NTU201803784

If the value drops below a value, we add a cut at that place. The technique works better when the signal contains noise.

$$RMS\ for\ window = \sqrt{\frac{1}{n}\Sigma_{i=T_0}^{T-5} v_{predict}^2} \tag{4.2}$$

$$A\ cut\ exists\ at\ T_i\ ,\ if\ RMS_i < 1m/s \tag{4.3}$$

4.3.5 Sliding window retrieval

For the trajectory, we use sliding window of 5s (50 points) to transform it into same length. As the sliding window would duplicate data and cause heavy memory overhead, we use batch processing and generator pattern to get data from database for training.

$$Sliding\ window_i = (T_i\ , T_{i+1}\ , T_{i+2}, ..., T_{i+49}) \tag{4.4}$$

4.4 Seq2seq models

After we get sliding window, we're ready to use it in our prediction. In this part, we describe our seq2seq model and the baseline models. Moreover, we will elaborate on how we use them for prediction.

4.4.1 Baseline : constant velocity

We use the velocity of last point in history for our prediction. This serves as a lower bound to validate the powerfulness of other method.

4.4.2 Baseline : kalman filter

Kalman filter is the model used in [2]. It serves as the baseline model. The problem of it is that it didn't have too many state, so it's hard to generalize to many cars.

doi:10.6342/NTU201803784

4.4.3 Baseline: Encoder-decoder

Encoder-decoder is the model used in [12]. While it's used for occupancy map in [12], the principle is the same. Encoder-decoder[6] use LSTM to encode the sequence into a fixed vector. Then, the vector is fed into a LSTM to generate a sequence. It's suitable for next step prediction. The advantage of this model compared to kalman filter is that it could has a lot of memory and thus could accommodate various pattern.

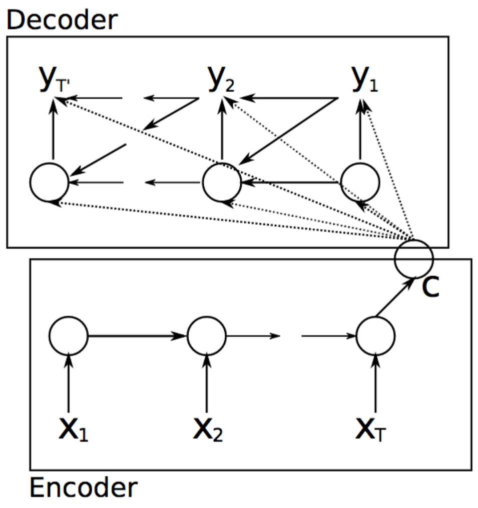


Figure 4.3: Architecture of encoder-decoder[6].

Encoder-decoder with attention

Although encoder-decoder is powerful, it has some shortage[4]. Encoder-decoder couldn't deal with the alignment problem. In machine translation , alignment is the problem in machine translation that identifies which parts of the input are relevant to the output, whereas translation is the process of using the relevant information to select the appropriate output. Instead of encode the entire sequence into a vector, attention mechanism encode each subsequence into a context vector. Moreover, it used the weighted average from each subsequence to make the prediction and thus solve the alignment problem. This is the reason that attention mechanism could deal with the subpattern and generate better prediction.

doi:10.6342/NTU201803784

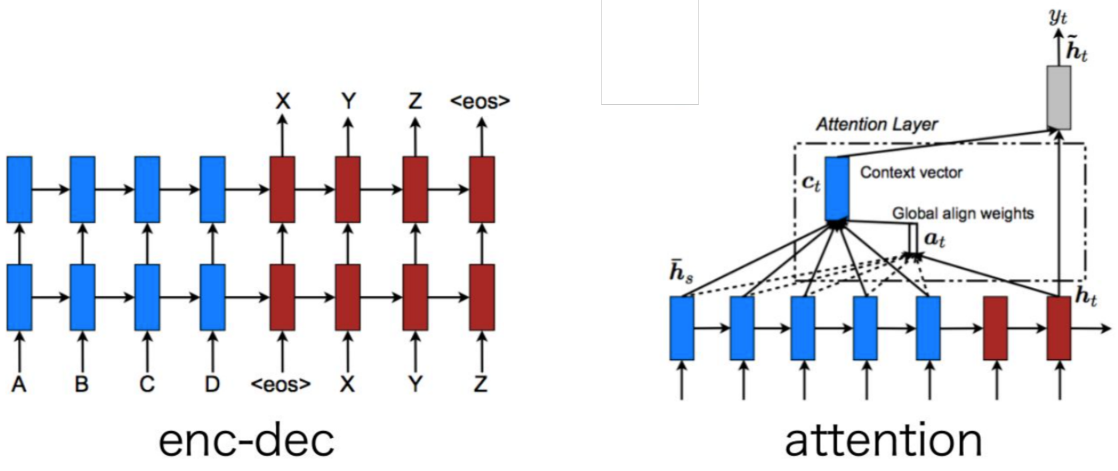


Figure 4.4: Comparison of attention mechanism and original encoder-decoder[10].

Why would it be helpful for our prediction?

To understand why attention model would be helpful, we need to look at the alignment problem in velocity prediction. The go-straight maneuver still contain many patterns. Car following, free driving, action before change lane and after change lane are all classified as go-straight maneuver. Thus, when we look at a trajectory, the pattern might start at the middle of it and cause the problem of alignment. This is the reason why attention model would be helpful.

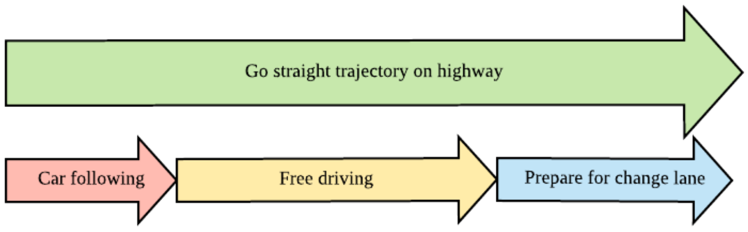


Figure 4.5: Subpattern inside a trajectory.

How we use the model

After high-level description, we would describe the model how we use it in detail in this section.

doi:10.6342/NTU201803784

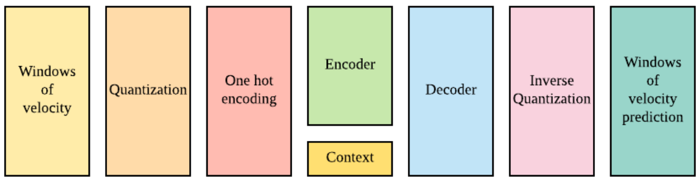


Figure 4.6: Overview of the model and how we use it for prediction.

Quantization and one hot encoding The model takes a sequence of velocity. Each velocity, V,is fed into a uniform quantization with 256 classes. So, the length of each class, Q, would be:

$$V_{max} = \max_{all\ trajectory} V \tag{4.5}$$

$$V_{min} = \min_{all\ trajectory} V \tag{4.6}$$

$$Q = \frac{V_{max} - V_{min}}{256} \tag{4.7}$$

Then, the conversion from velocity would be:

$$Q(V) = \lfloor \frac{V - V_{min}}{Q} \rfloor Q + \frac{Q}{2} + V_{min} \tag{4.8}$$

After the quantization, we do one hot encoding. So, now, our model takes a sequence of velocity and generate a prediction of 5s. As the sampling rate is 10Hz, we have 50points in each sequence.

$$V_{input} = (V_{x1}, V_{x2}, ..., V_{x50}), V_i \in \mathcal{R}^{256} \tag{4.9}$$

$$V_{output} = (V_{y1}, V_{y2}, ..., V_{y50}), V_i \in \mathcal{R}^{256} \tag{4.10}$$

At the end of the model, we need to do inverse quantization for calculation of MSE.

doi:10.6342/NTU201803784

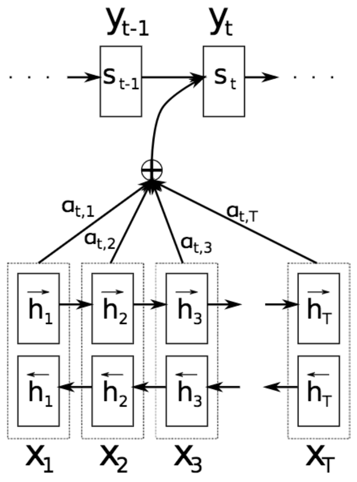


Figure 4.7: Architecture of encoder-decoder with attention model[4].

Encoder Encoder reading source sentence while updating hidden state. In the following formula, f could be LSTM or GRU. In our work, we use LSTM.

$$h_t^{enc} = f(h_{t-1}^{enc}, v_t) \tag{4.11}$$

Decoder Decoder producing target sentence while updating hidden state. In attention model, the context vector c is also used to produce the next step.

$$h_t^{dec} = f(h_{t-1}^{dec}, v_t, c_t) \tag{4.12}$$

Context Context vector is calculated by a combination of encoder hidden states weighted by attention weights:

$$c_t = \sum_{i=1}^{50} a_{t,i} * h_i^{enc} \tag{4.13}$$

$$a_{t,1....T} = softmax(align(h_{t-1}^{dec}, y_{t-1}, h_{1...50}^{enc})) \tag{4.14}$$

In our model, align is a feed forward network.

doi:10.6342/NTU201803784

4.5 Design of data classifier

Our data classifier aims to seperate the data for different driving scenarios. From our dataset, the following characterisctice are used for classification. The implementation of each characteristic will be described in the following sections.

- Maneuver
- Speed limit
- Average speed

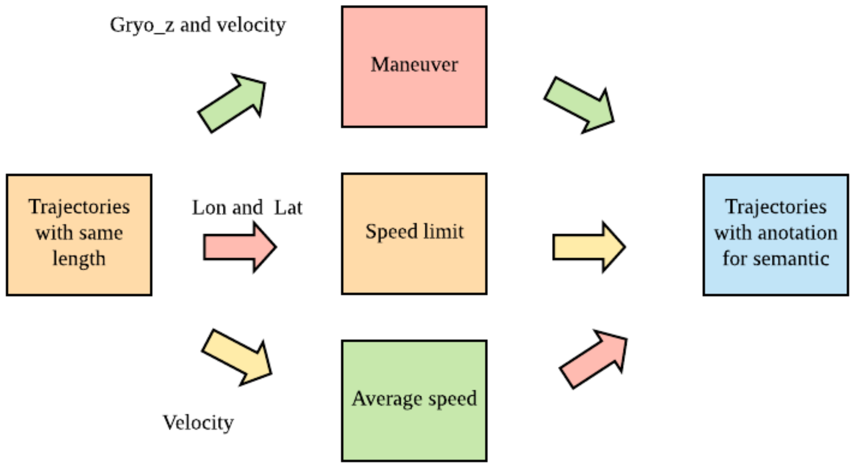


Figure 4.8: Overview of data classifier

4.5.1 Classifier for speed limit

In order to get geographical information, we utilize openstreetmap to get the road map and road type. From its database, we query the road name and its speed limit from longitude and latitude.The query API that we used, Overpass, doesn't support k nearest neighbor query, so we have to find the nearest road based on results of bounding box query.

doi:10.6342/NTU201803784

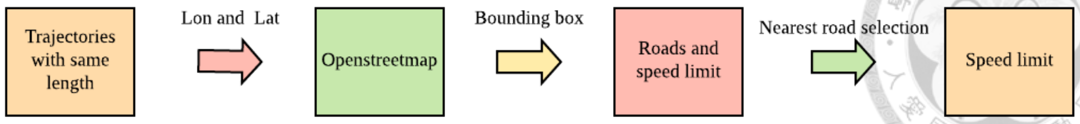
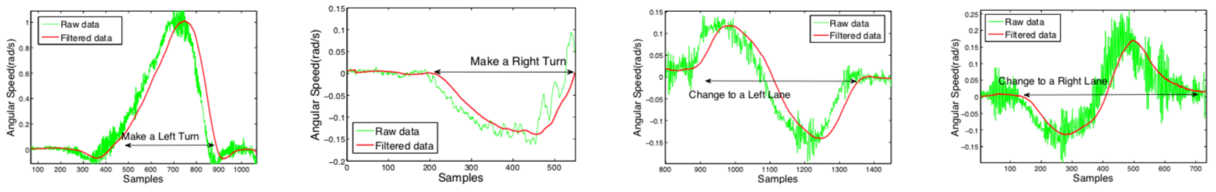


Figure 4.9: Query and select the speed limit

4.5.2 Classifier for maneuver

From trajectories, now we have to define our maneuver class and describe how we get it. The maneuver class has three types: go straight, change lane, and turn. We use the gyro data for classification. According to [5], gyro on the z axis will have one or two bumps when the vehicle change lane and turn. So, by detecting the number of bump, we could classify the maneuvers.



(a) Make a left turn (b) Make a right turn (c) Change to a left lane (d) Change to a right lane

Figure 4.10: Maneuver classification[5].

4.5.3 Classifier for average speed

For each road in the trajectory, we calculate the average speed for self-car. We divide all the data into 5 classes. The average speed could serve as a inference for driver's behavior and traffic condition.

$$Average\ speed\ for\ window = \frac{1}{50}\Sigma_{i=T_0}^{T_{50}} v_{predict} \tag{4.15}$$

doi:10.6342/NTU201803784

4.5.4 Explore the prediction under different scenarios

After the classification above, we turn our raw data into better structure. The sequences of sensor data now turn into a sequence of steps and action on road. For example, go straight at road A, turn to road B, go straight at road B, and turn to road C. This makes the raw data more meaningful. After this step, we train a model for each driving scenario and compare prediction, as shown in figure 4.11.

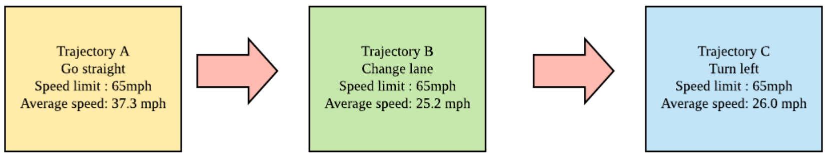


Figure 4.11: Better interpretation of raw data

4.6 Comparison of single car model and multi-car model

To explore the limitation of single car model, we use different dataset and model. Our new dataset should represent the interaction between multiple cars. As shown in figure 4.13, we will extract the data of ego vehicle and surrounding vehicle respectively. Then, we use the single car model and multi-car model on our data for prediction. When we compare the result of prediction, we could realize the limitation of single car model.

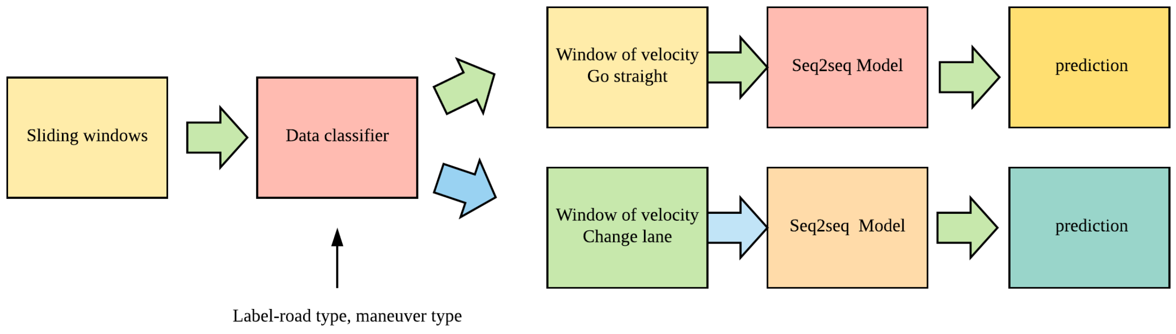


Figure 4.12: Prediction under different scenario

doi:10.6342/NTU201803784

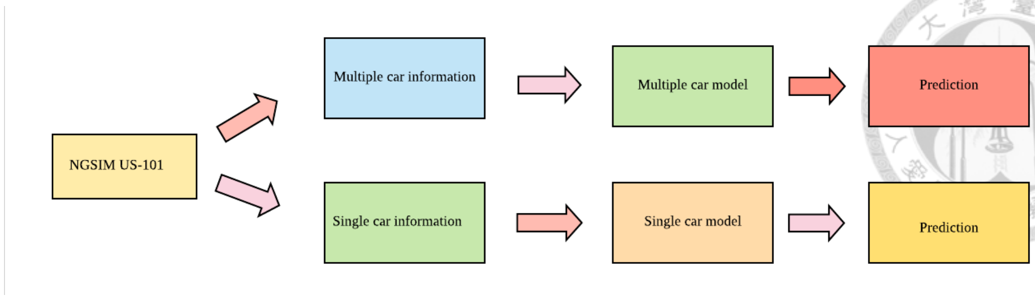


Figure 4.13: comparison of multiple car model and single car model

4.6.1 NGSIM US-101

NGSIM US-101 [9] is a public dataset that's widely used in traffic research. It contains 45 minutes of trajectories for vehicles on the US101 highway between rush hour of 7:50am and 8:35am.

The original data includes many columns, but we select local coordinate X and Y as a start. Moreover, we adopted the data preprocessing techniques used in this LSTM paper to get velocity and acceleration. Besides, the data has a record about lane. So, our features include:

- X,Y in local coordinate, with sampling rate 10Hz.
- Timestamp
- Vx, Vy by processing.
- Ax, Ay by processing.
- Lane ID

Maneuver classification for NGSIM

As the data contains lane id, a lane change happens when the lane ID changes.

4.6.2 The multi-car model

[1] used a recurrent model with residual connection for ego vehicle. This model, as shown in figure , it will use the information of ego vehicle and surrounding car to predict

doi:10.6342/NTU201803784

the trajectory of ego vehicle. We used this model for comparison.

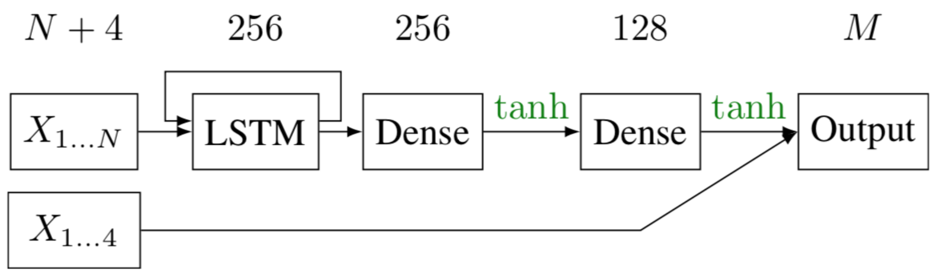


Figure 4.14: multiple car model[1]

doi:10.6342/NTU201803784

Chapter 5

Performance Evaluation

5.1 Experiment Environment

In our work, we use Keras and Tensorflow to implement our model. Moreover, the following parameters, model, and dataset are used:

- input and output sequences: 5s, 50 points in 10Hz.
- stack layer for RNN=2
- model: encoder-decoder with attention.

5.2 Performance metric

In our work, the root mean square error(RMSE) is used for evaluation. It's used in [1][12][2], so it's a legitimate target for comparison with related work.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(v_{predict} - v_{truth})^2} \tag{5.1}$$

doi:10.6342/NTU201803784

5.3 Experiment on new model

5.3.1 Experiment on private dataset

Experiment design

In the experiment, 4 models are compared. The training and the prediction will run for y direction. (tangential direction)

The parameters and model are the same as the previous one. Moreover, the go-straight maneuver class is used. Besides, the speed limit is set as 65mph and the average speed is set as V5(the largest). This combination would has the smallest influence from surrounding cars, and will best reflect the power of model.

Model	Input	Output	Metric
Constant Velocity	5s	5s	RMSE
Kalman	5s	5s	RMSE
Encoder-decoder	5s	5s	RMSE
Attention	5s	5s	RMSE

Table 5.1: Experiment for model comparison

Experiment result

In the experiment, 4 models are compared. The unit for MSE is meter per second.

Experiment result

Model	1s	2s	3s	4s	5s
Constant Velocity	1.345	1.456	1.782	1.94	2.35
Kalman	0.932	1.15	1.323	1.576	1.728
Encoder-decoder	0.881	1.1	1.293	1.544	1.674
Attention	0.807	1.03	1.243	1.502	1.642

Table 5.2: Result for model selection on private data set

doi:10.6342/NTU201803784

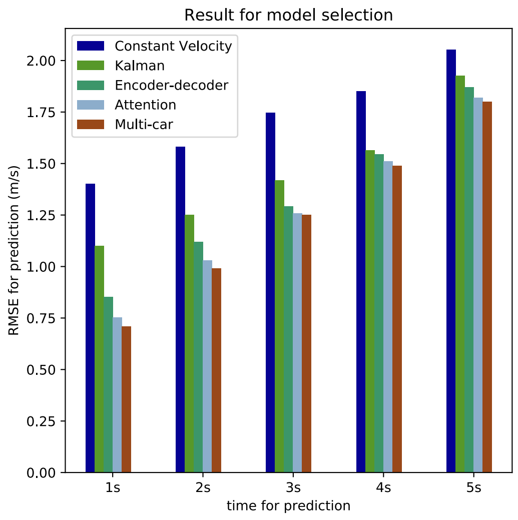


Figure 5.1: Result for model comparison on private dataset

Discussion

From the result, we found encoder-decoder with attention could achieve 11% improvement over kalman filter. Besides, encoder-decoder is also better than kalman filter. on the other hand, constant velocity predictor serves as a baseline to indicate the effect of other method. We could see that these methods achives much better result than constant velocity predictor. However, in prediction at 4s and 5s, the methods didn't achieve significant difference. The error might come from the fact tha vehicle 4s and 5s have gone to another event and thus makes it hard to predict in a non-linear way.

5.3.2 Experiment on NGSIM US-101

Experiment design

In the experiment, the multi-car model is added for comparison.

Model	Input	Output	Metric
kalman[2]	5s	5s	RMSE
Encoder-decoder	5s	5s	RMSE
Attention	5s	5s	RMSE
multi-car[1]	5s	5s	RMSE

Table 5.3: Experiment for model comparison on NGSIM

doi:10.6342/NTU201803784

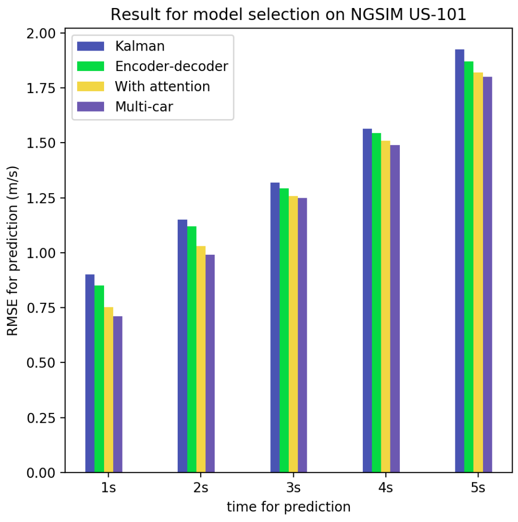


Figure 5.2: Result for model comparison on ngsim

Experiment result

Model	1s	2s	3s	4s	5s
Constant Velocity	1.402	1.581	1.747	1.85	2.053
Kalman	1.101	1.25	1.419	1.565	1.925
Encoder-decoder	0.851	1.12	1.293	1.544	1.871
Attention	0.752	1.03	1.257	1.51	1.82
Multi-car	0.71	0.99	1.25	1.49	1.8

Table 5.4: Result for model selection on NGSIM

Discussion

The result show the effectiveness of attention mechanism again. Besides, these method couldn't beat the multi-car model and this might because the interaction of surrounding car is so important for prediction.

doi:10.6342/NTU201803784

5.4 Prediction under different scenario

5.4.1 Experiment for maneuver class

Experiment design for private dataset

In the experiment, the 3 maneuver classes are compared.The training and the prediction will run for y direction. (tangential direction). The mixed one serves as a baseline to validate the classification based on maneuver.

Maneuver	input and output	metric
Change lane	5s	RMSE
Go straight	5s	RMSE
Turn	5s	RMSE
Mixed	5s	RMSE

Table 5.5: Experiment for maneuver classification

Experiment result for private dataset

For the result, the RMSE for prediction at 1s, 2s, 3s, 4s and 5s are shown. The smallest RMSE represent the best result. The unit for RMSE is meter per second.

Maneuver	1s	2s	3s	4s	5s
Turn	0.923	1.164	1.312	1.574	1.734
Go straight	0.82	1.021	1.228	1.454	1.543
Change lane	0.912	1.15	1.305	1.578	1.703
Mixed	0.881	1.1	1.293	1.544	1.674

Table 5.6: Result for maneuver on single car

Discussion

From the result, we found that go-straight maneuver has the best prediction, and it is better than the mixed model. One of the reason might be that this maneuver suffers from least influence of surrounded car. In order to validate this assumption, we used the multi-car model for experiment.

doi:10.6342/NTU201803784

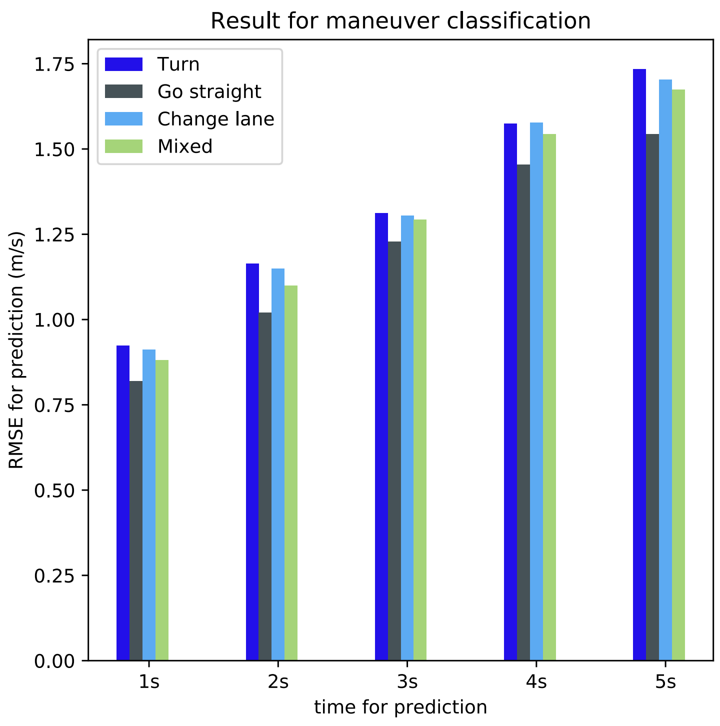


Figure 5.3: Result for maneuver classification

5.4.2 Experiment of maneuver for comparison between single car model and multi-car model

Experiment design

In the experiment, two models are compared for the maneuver class "change lane" and "go straight". Moreover, we use NGSIM as dataset.

Maneuver	model	input and output	metric
Change lane	multi-car	5s	RMSE
Change lane	single car	5s	RMSE
Go straight	multi-car	5s	RMSE
Go straight	single car	5s	RMSE

Table 5.7: Experiment for maneuver classification

doi:10.6342/NTU201803784

Experiment result

For the result, the RMSE for prediction at 1s, 2s, 3s, 4s and 5s are shown. The smallest RMSE represent the best result. The unit for RMSE is meter per second.

Maneuver	model	1s	2s	3s	4s	5s
Go straihgt	multi-car	0.713	0.992	1.228	1.454	1.543
Go straight	single car	0.755	1.045	1.253	1.602	1.723
Change lane	multi-car	0.718	1.012	1.273	1.483	1.602
Change lane	single car	0.77	1.151	1.355	1.652	1.792

Table 5.8: Result for maneuver on single car and multi-car

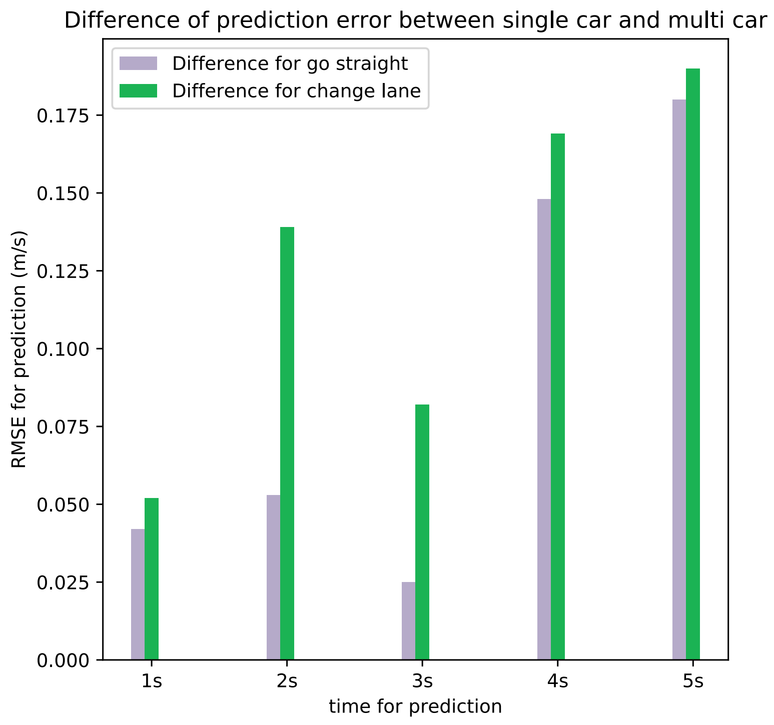


Figure 5.4: Difference of prediction error between single car and multi-car

Discussion

From the figure, we could see that the difference between single-car model and multi-car model is higher in change lane maneuver. The single car model could be enhanced

doi:10.6342/NTU201803784

by incorporating motion of ego vehicle in change lane maneuver. This serve as a future work.

5.4.3 Experiment for speed limit and average speed

Experiment design

In the experiment, totally 25 cases are compared, as each factor has 5 classes.The training and the prediction will run for y direction. (tangential direction)

The parameters and model are the same as the previous one. However, only the go-straight maneuver is used for prediction at this stage.

Class for average speed	class for speed limit	input and output	metric
V1, V2, V3, V4, V5	25,35, 40,45,65mph	5s	MSE

Table 5.9: Experiment for speed limit and average speed

Experiment result

In the following table, the index for row is speed limit. Some entries are denoted N/A as the data size is too small. For example, in highway, there are very few low speed trajectories. We choose the prediction at 1s to analyze, as we believe it is the most important prediction among 1-5s. The unit for MSE is meter per second.

limit	10-30mph	30-40mph	40-50mph	50-60mph	60mph up
25mph	0.923	0.91	NA	NA	NA
35mph	0.912	0.851	0.828	NA	NA
40mph	0.857	0.858	0.824	0.821	NA
45mph	0.845	0.838	0.809	0.813	0.79
65mph	NA	NA	0.816	0.805	0.807

Table 5.10: Result for speed limit and average speed

Discussion

From the result, we found that the worst prediction will be at smallest average speed and smallest speed limit. (Top-left corner) On the contrary, the best prediction will be

doi:10.6342/NTU201803784

at the bottom-right corner. The top-left corner is most likely to be in traffic jam. We speculate that it's the reason behind for this result.

doi:10.6342/NTU201803784

Chapter 6

Conclusion

To summary, our work improve the prediction in two aspect. At the frontend data preprocessing, we identify several key semantics like maneuver class, average speed, and road speed limit and use it to improve our prediction. Moreover, we utitlize the encoder-decoder model with attention to achieve a 11% improvement over kalman filter. This work could serve as a strong basis for future application like free space estimation.

doi:10.6342/NTU201803784

Bibliography

[1] Florent Altché and Arnaud de La Fortelle. An LSTM network for highway trajectory prediction. *CoRR*, abs/1801.07962, 2018.

[2] Samer Ammoun and Fawzi Nashashibi. Real time trajectory prediction for collision risk estimation between vehicles. In *Intelligent Computer Communication and Processing, 2009. ICCP 2009. IEEE 5th International Conference on*, pages 417–422. IEEE, 2009.

[3] ARTC. Automatic research and testing. `https://www.artc.org.tw/upfiles/ADUpload/knowledge/tw_knowledge_499017376.pdf`, 2017.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[5] Dongyao Chen, Kyong-Tak Cho, Sihui Han, Zhizhuo Jin, and Kang G. Shin. Invisible sensing of vehicle steering with smartphones. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, pages 1–13, New York, NY, USA, 2015. ACM.

[6] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[7] Mordechai (Muki) Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, October 2008.

[8] Rubén Izquierdo, Ignacio Parra, Jesús Muñoz-Bulnes, D Fernández-Llorca, and MA Sotelo. Vehicle trajectory and lane change prediction using ann and svm classifiers. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pages 1–6. IEEE, 2017.

[9] Xiao-Yun Lu and Alexander Skabardonis. Freeway traffic shockwave analysis: Exploring the ngsim trajectory data. 07 2018.

doi:10.6342/NTU201803784

[10] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

[11] J. Nilsson, J. Silvlin, M. Brannstrom, E. Coelingh, and J. Fredriksson. If, when, and how to perform lane change maneuvers on highways. *IEEE Intelligent Transportation Systems Magazine*, 8(4):68–78, winter 2016.

[12] SeongHyeon Park, Byeongdo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. *CoRR*, abs/1802.06338, 2018.

[13] Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *CoRR*, abs/1704.02971, 2017.

[14] Fan Wu, Kun Fu, Yang Wang, Zhibin Xiao, and Xingyu Fu. A spatial-temporal-semantic neural network algorithm for location prediction on moving objects. *Algorithms*, 10(2):37, 2017.

doi:10.6342/NTU201803784