An LSTM Network for highway trajectory prediction of a vehicle with given trajectories of the surrounding vehicles

Ankit Khandelwal
Department of Electronics and Infomation Technology
Technische Hochschule Ingolstadt
Ingolstadt, Germany
ankit.electrical01@gmail.com

Dr.-Ing. João Paulo C. L. da Costa
Department of Electronics and Infomation Technology
Technische Hochschule Ingolstadt
Ingolstadt, Germany
joaopaulo.dacosta@ene.unb.br

***Abstract*— In order to drive safely on public roads, for automotive vehicles having ADAS systems, main and the most difficult task is to predict the intention of a driver of the surrounding car. This can be achieved by the prediction of the surrounding vehicle in the traffic situation. For Autonomous vehicle the trajectory prediction is very important now a days. This paper will discuss about one of the technique using an LSTM network to predict trajectory of target vehicles on highway.The LSTM networks are very efficient to predict the driving behaviour in time instances. The data-set NGSIM US101, which is used here is not a small data with only trajectories of some cars. It is the data for trajectories of more then 6000 cars on a part of highway.**

I. Introduction

In order to build fully autonomous vehicle, it is necessary to guarantee high degree of safety even in uncertain and dynamically changing environments. In order to serve this goal, an autonomous vehicle should be able to anticipate what would happen to its environment in the future and respond to the change appropriately in advance. However, the behavior of the traffic participants (e.g. the vehicles surrounding the ego vehicle) is often hard to predict since it is affected by various latent factors such as driver's intention, traffic situations, road structure, and so on. The prediction based on vehicle's dynamics model is accurate only for very near future and it does not match with true trajectory well for long term prediction (more than one second). In addition, it is hard to know the accurate maneuver control (e.g. steering and acceleration) of the other vehicles so that the trajectory prediction based on vehicle's dynamics model might not be effective in practical scenarios [2].

Many approaches to motion prediction have been proposed in the literature, and survey can be found in [3]. As in many machine learning applications, existing techniques can be split between classification or regression methods. When applied to motion prediction, classification problems consist in determining a high-level behavior (or intention), for instance lane change left, lane change right or lane keeping for highway driving or turn left, turn right or go straight in an intersection. Many techniques have already been explored for behavior prediction, such as hidden Markov models [4], [5], Kalman filtering [6], Support Vector Machines [7], [8] or directly using a vehicle model [9]; more recently, artificial neural network approaches have also been proposed [10]–[12].

The main advantage of predicting behaviors is that the discrete outputs make it easier to train models and evaluate their performance. However, they only provide rough information on future vehicle states, which is not easy to use when planning a trajectory for the self-driving ego-vehicle.This approach requires multiple draining and is only as robust as the classifier accuracy. Regression problems, on the other hand, aim at directly obtaining a prediction for future positions of the considered vehicle, which can then be used for motion planning. Many regression algorithms could be used for this problem. for instance regression forests; more recently, artificial neural networks have attracted the most attention in the field of trajectory prediction for cars, cyclists or pedestrians. A potential downside of such approaches is that the output of many regression algorithms is a single "point" (e.g., a single predicted trajectory) without providing a measure of confidence[1].

In this article, we focus on trajectory prediction using long short-term memory (LSTM) neural networks, which are a particular implementation of recurrent neural networks. Because they are able to keep a memory of previous inputs, LSTMs are considered particularly efficient for time series prediction and have been widely used in the past few years for pedestrian trajectory prediction or to predict vehicle destinations at an intersection. Our main contribution is the design of an LSTM network to predict car trajectories on highways, which is notably critical for safe autonomous overtaking or lane changes, and for which very little literature exists. A particular challenge for this problem is that highway driving usually comprises a lot of constant velocity phases with rare punctual events such as lane changes, which are therefore hard to learn correctly. For this reason, many authors rely on purposely recorded or handpicked, trajectory sets which are not representative of actual, average driving. Therefore, the real-world performance of trained models can be significantly different[1]. A second contribution of this article is that we

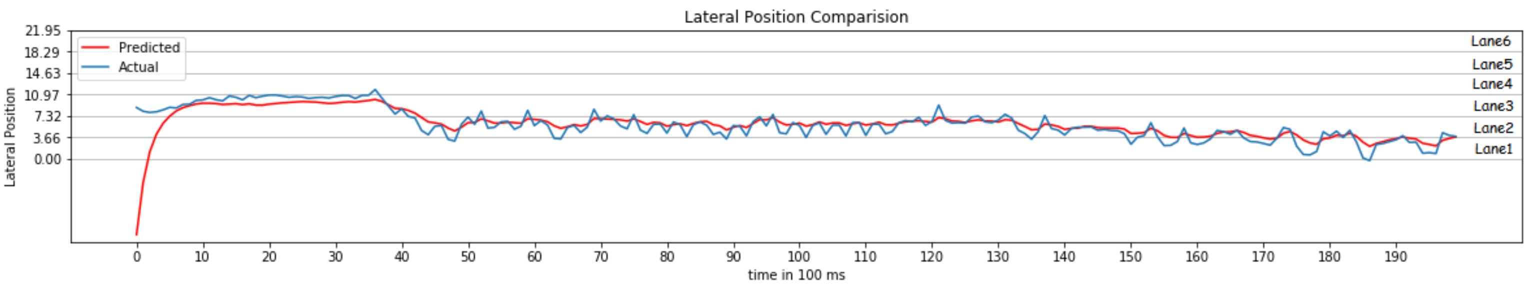


Fig. 1. Lateral Position(Vehicle Number 595).

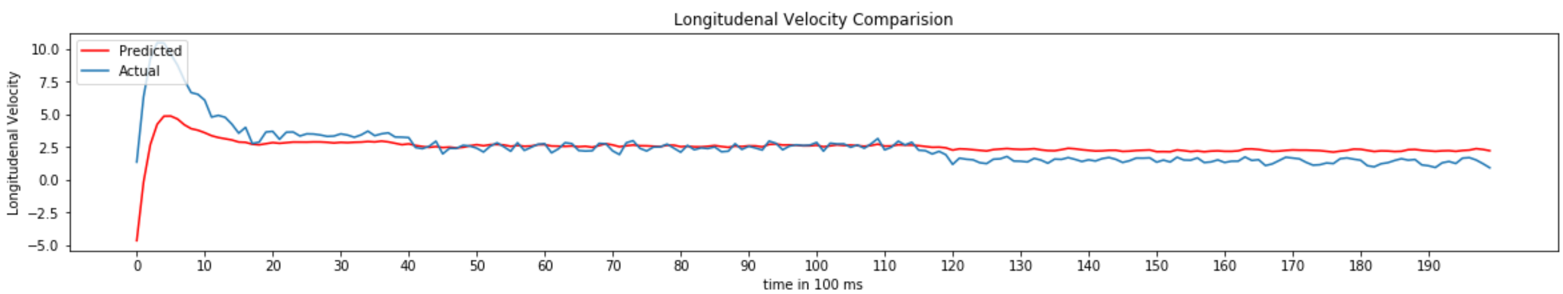


Fig. 2. Longitudinal Velocity(Vehicle Number 595).

train and validate our model using the entire NGSIM US101 dataset [18].

The rest of this article is structured as follows: in Section II, we define the trajectory prediction problem that we are aiming to solve. In Section III, we detail the pre-processing of the US101 data-set to extract the input features of the model, which is presented in same section. In section IV, We detail the approach used network selection and architecture used. In Section V, we present the training procedure and outputs of the trained model. Finally, Section VI concludes the study.

II. Problem Statement

We consider the problem of predicting future trajectories of vehicles driving on a highway, using previously observed data of vehicle in similar situation; these predictions can then be used to plan the motion of an autonomous vehicle[1].

Formally, we consider a set of observable features I and a set of target outputs O to be predicted. We assume that the features can all be acquired simultaneously at regular intervals,We let $T = \{0, ..., K\}$ and for $x \in I, k \in T$.

We denote by x_k the value of feature x observed k time steps earlier. Similarly, we denote by y_k the value of output $y \in O, k \in T$ time steps.

We use uppercase

$$X = (x_k), for x \in I, k \in T$$

and

$$Y = (y_k), for y \in O, k \in T$$

to respectively denote the tensors of the observed features and corresponding predicted outputs. We propose to use a machine learning approach, in which we train a regression function f such that the predicted outputs Ŷ = f(X) match the actual values as closely as possible.

In this article, our approach is to train a predictor for the trajectory of a single “target” vehicle; in order to only use data which can realistically be gathered, we limit the amount of available information to the vehicles immediately around the target vehicle, as described in Section III. As with many learning approaches, one difficulty is to design models that are able to generalize well from the training data. A second difficulty, more specific to the problem of highway trajectory prediction, is the imbalance between constant velocity driving phases, which are much more frequent than events such as lane changes.

III. Data-set Pre-processing and Feature Selection

A. Data-set

In this article, we use the Next Generation Simulation (NGSIM) dataset [18], collected in 2005 by the United States Federal Highway Administration, which is one of the largest publicly available source of naturalistic driving data and, as such, has been widely studied in the literature. More specifically, we consider the US101 dataset which contains 45 minutes of trajectories for vehicles on the US101 highway, between 7:50am and 8:35am during the transition from fluid traffic to saturation at rush hour. In total, the dataset contains trajectories for more than 6000 individual vehicles, recorded at 10 Hz. The NGSIM dataset provides vehicle trajectories in the form of (X; Y) coordinates of the front center of the vehicle in a global frame, and of local (x; y) coordinates of the same point on a road-aligned frame. In this article, we use the local coordinates (dataset columns 5 and 6), where x is the lateral position of the vehicle relative to the leftmost edge of the road, and y its longitudinal position. Moreover, the dataset contains each vehicle's lane identifier at every time step, as well as information on vehicle dimensions and type (motorcycle, car or truck). Finally, the data also contains the identifier of the preceding vehicle for every element in the set (when applicable).

B. Data preparation

One known limitation of the NGSIM set is that vehicle positioning data was obtained from video analysis, and the recorded trajectories contain a significant amount of noise. Velocities, which are obtained from numerical differentiation, suffer even more from this noise. For this reason, we used a first order Savitzky-Golay filter [1]– which performs well for signal differentiation – with window length 11 (corresponding to a time window of 1 s) to smooth the longitudinal and lateral positions and compute the corresponding velocities, as illustrated in Figure 3.

Figure 3 represents the comparison of of the actual values and the filtered output of Savitzky-Golay filter and python provides an API for this filtered. Already available api in python is used to apply this filter and the figure might be looking crowded/bulky because the output for all 1500 values. Unit of time is 100ms, It means each index on x axis has to be multiplied with 100ms for actual time. the output are shown for some randomly selected vehicle out of all 600 vehicle from data set. So, the vehicle chosen in figure 3 might have this velocity behavior but it can not be generalized for other vehicles. Figure 3 is intended to show just the actual and filtered value comparison and effectiveness of the filter used.

In this article, we hypothesize that the future behavior of a target vehicle can be reliably predicted by using local information on the vehicles immediately around it; a similar hypothesis was successfully tested in to detect lanechange intent. For a target vehicle, we consider 9 vehicles as shown in Figure 4.

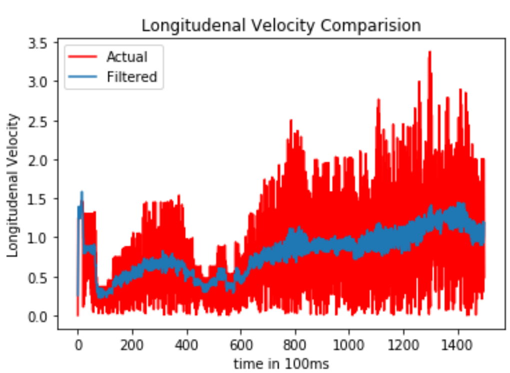


Fig. 3. Filtered output.

The blue arrow represents traffic direction. of interest, that we label according to their relative position with respect to

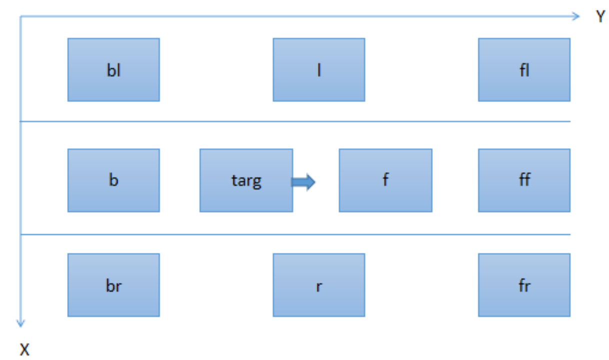


Fig. 4. Vehicle of interests around the Target Vehicle[1].

the target vehicle targ, as shown in Figure 5. By convention, we let r (respectively l) be the vehicle which is closest to the target vehicle in a different lane with $x > x_{targ}$ (respectively $x < x_{targ}$). We respectively denote by fl, f, fr and ff the vehicle preceding l, targ, r and f; similarly, vehicles bl, b and br are chosen so that their leader is respectively l, targ and r. During the data preprocessing phase, we compute the identifier of each vehicle of interest and perform join requests to append their information to the dataset. When such a vehicle does not exist, the corresponding data columns are set to zero. Note that the rationale behind the inclusion of information on ff is that only observing the state of the vehicle directly in front is not always sufficient to correctly determine future traffic evolution. For instance, in a jam, knowing that vehicle ff is accelerating can help infer that f, although currently stopped, will likely accelerate in the future instead of remaining stopped. The obvious limit to increasing the number of considered vehicles is the ability to realistically gather sufficient data using on-board sensors; for this reason, we restrict the available information to these 9 vehicles.

C. Data-set Preprocessing Steps

Because of the computational limitation data of only first 600 vehicles is filtered out at the start and it will be used for the network. Some steps are followed for pre-processing of data set, before creating feature vector. The steps are mentioned in Figure 5 and explained below:

- Step 0: First step is to figure out what are the important parameters required from initial data set to final data set.
- Step 1: Sort the dataset based on vehicle ids given.
- Step 2: Create separate csv file for each dataset with number of rows equal to minimum number of instances available for a vehicle out of all 600 vehicles. This step is done so that the feature vector has constant dimension. Also it will hep in velocity calculation in each direction
- Step 3: Calculate lateral x and longitudinal y direction velocities for all the vehicles and assign an indexer to each row because the time difference from the given global time is not possible to be calculated. Each index represents 100ms time.

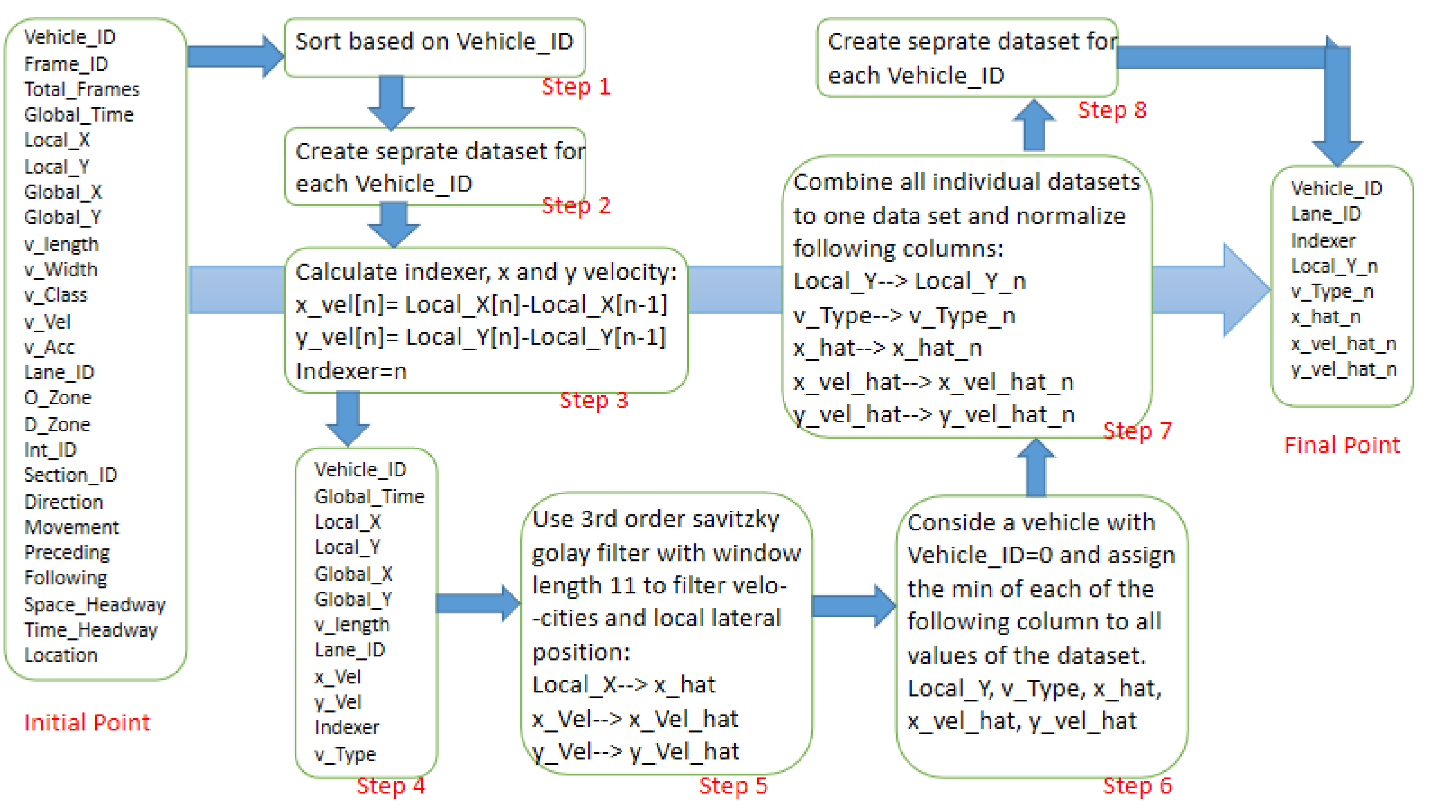


Fig. 5. Dataset Preprocessing.

- Step 4: Store the velocities and Index values in the dataset and drop the columns which are not required.
- Step 5: As the velocities are the first difference of the position values some noise will automatically gets included in the signal. To filter the velocities Savitzky-Golay filter is used, which is explained in more detail in Sub-Section III-B. Then the nun filetred values are replaced by the filtered values.
- Step 6: As the network will only take values between range {0,1}, the normalization of the values is required. To achieve this a new dataset is created manually with vehicle id 0. More details about this step is given in III-D.
- Step 7: Now the data of all the vehicle ids is concatenated to normalize the required columns. After normalization the non-normalized values are replaced by normalized values.
- Step 8: Sort the dataset again based on vehicle ids given as it was done in Step 1. Now we have the final datasets for all the vehicles which will be used to create input feature vector as shown in Figure 8.

Now the final point is achieved the vehicles can be selected based on Figure 4

D. Feature Selection

In this article, we aim at only using features which can be reasonably easily measured using on-board sensors such as GNSS and LiDAR, barring range or occlusion issues. For this reason, we consider a different set of features for the target vehicle (for which we want to compute the future trajectory) and for its surrounding vehicles as described above.

For the target vehicle, we define the following features:

- local lateral position x_{targ}, to account for different behaviors depending on the driving lane,
- local longitudinal position y_{targ}, to account for different behaviors when approaching the merging lane,
- lateral and longitudinal velocities $v_{x_{targ}}$ and $v_{y_{targ}}$,
- type (motorcycle, car or truck), encoded respectively as 1, 2 or 3.

For each vehicle $p \in \{bl; b; br; l; f; r; fl; fr; ff\}$, we define the following features:

- lateral velocity v_{x_p},
- longitudinal velocity relative to targ: $v_{y_p} = v_{y_{targ}} - v_{y_p}$,
- lateral distance from targ: $x_p = x_p - x_{targ}$,
- longitudinal distance from targ: $y_p = y_p - y_{targ}$,
- type (motorcycle, car or truck), encoded respectively as 1, 2 or 3.

These features are scaled to remain in an acceptable range with respect to the activation functions; in this article, The features are normalized between values {0,1} to make the learning fast. The approach for this is on the combined data set, calculate the minimum values and assume one vehicle with vehicle ID 0 having each feature's value one less then the minimum value of that respective feature. After this the normalization of data is done to fit all the values between {0,1}. By doing this network will not be confused between actual zero and the value which is considered as zero after normalization. At the end data is de-normalized to back to the original approximated value, before visualizing the results.

This choice of features was made to replicate the information a human driver is likely to base its decisions upon: the features from surrounding vehicles are all relative to the target vehicle, as we expect drivers to usually make decisions based on perceived distances and relative speeds rather than their values in an absolute frame. Features regarding the target vehicle's speed are given in a (road-relative) absolute frame as drivers are generally aware of speedometer information; similarly, we use road-relative positions since the driver is usually able to visually measure lateral distances from the side of the road, and knows its longitudinal position.

E. Output

our goal is to predict the future trajectory of the target vehicle. Since the region of interest spans roughly 1km longitudinally, the values of the longitudinal position can become quite large; for this reason, we prefer to predict future longitudinal velocities $\hat{v}_{y_{targ}}$ instead. Since the lateral position is bounded, we directly use $\hat{x}_{targ}$ for the output. In order to have different horizons of prediction, we choose a vector of outputs $\{\hat{x}_{k_{targ}}; \hat{v}_{y_{k_{targ}}}\}$ for k=1:K consisting in values taken k seconds in the time.

IV. Network Selection and Architecture

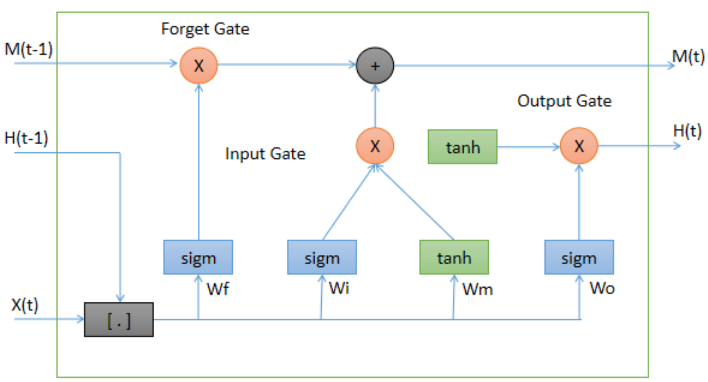


Fig. 6. LSTM Cell.

Contrary to many existing frameworks for intent or behavior prediction, which can be modeled as classification problems, our aim is to predict future (x; y) positions for the target vehicle, which intrinsically is a regression problem. Due to their success in many applications, we choose to use an artificial neural network for our learning architecture, in the form of a Long Short-Term Memory (LSTM) network. LSTMs are a particular implementation of recurrent neural networks (RNN), which are particularly well suited for time series; in this article, we used the Keras framework, which implements the extended LSTM, presented in Figure 6. Compared to simpler vanilla RNN implementations, LSTMs are generally considered more robust for long time series; future work will focus on comparing the performance of different RNN approaches on our particular dataset.

An interesting feature of LSTM cells is the presence of an internal state which serves as the cell's memory, denoted by mt in Figure 4. Based on a new input xt, its previous state c_{t-1} and previous output h_{t-1}, the cell performs different operations using so-called "gates":

- forget: uses the inputs to decide how much to "forget" from the cell's previous internal state c_{t-1};
- input: decides the amount of new information to be stored in memory based on x_t and h_{t-1};
- output: computes the new cell output from a mix of the previous states and output of the input gate.

This particular feature of LSTMs allows a network to learn long-term relations between features, which makes them very powerful for time series prediction. Due to their recurrent nature, even a single layer of LSTM nodes can be considered as a "deep" neural network. Although such layers may theoretically be stacked in a fashion similar to convolutional neural networks to learn higher-level features, previous studies and our own experiments (see Section V) seem to indicate that stacked layers of LSTM do not provide improvements over a single layer in our application.

In this article, we use the network presented in Figure 8 as our reference architecture, and we compare a few variations on this design in Section V. The reference architecture uses a first layer of 256 LSTM cells, followed by two dense (fully connected) and time-distributed layers of 256 and 128 neurons and a final dense output layer containing as many cells as the number of outputs. In this simple architecture, the role of the LSTM layer is to abstract a meaningful representation of the input time series; these higher-level "features" are then combined by the two dense layers in order to produce the output, The hyper-parameters chosen for the network layers are mentioned in Table I:

V. Training Procedure

In this section, the procedure to create feature vector and training the model is explained. A feature Vector $\hat{X}$ is prepared by using data mentioned in Section III. As the instances of each vehicle is not same in the data set. Lowest number of instances of the vehicle is chosen from given vehicles in the data-set. First the vehicle corresponding to $p \in \{bl; b; br; l; f; r; fl; fr; ff\}$ are determined. After that the feature matrix is created in such a way that the features are the columns and rows are the values of the features at that time instance. Then an output vector $\hat{Y}$ is created which contains lateral position and longitudinal velocity of the target vehicle.

During training the data is fed in such a way that for every training one vehicle is target vehicle and corresponding surrounding vehicles will be chosen every time. None of the training matrix has a same target vehicle. The data set of 600 vehicles is divided in 3 parts:

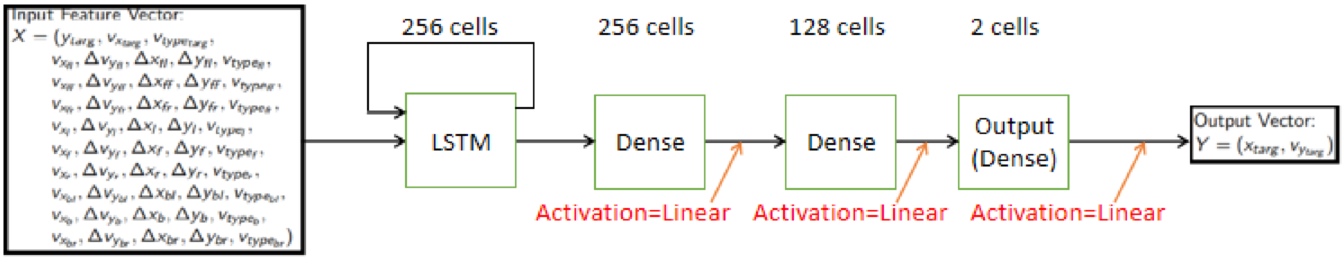


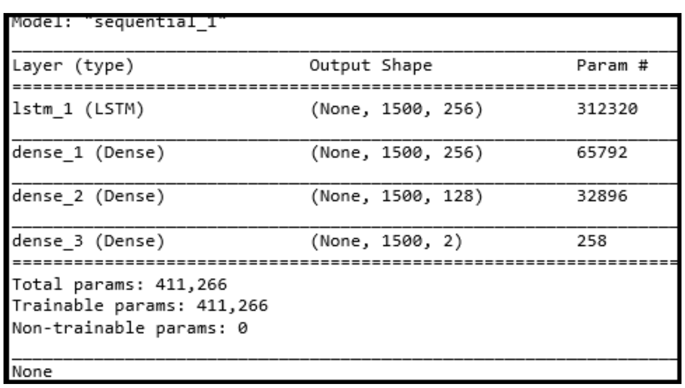
Fig. 7. Network Architecture.

TABLE I
HYPER PARAMETERS FOR MODEL LAYERS

Layer Name		lstm_1	dense_1	dense_2	dense_3
S.No.	Parameter	Values	Values	Values	Values
1	units	256	256	128	2
2	activation	-	linear	linear	linear
3	use_bias	-	True	True	True
4	kernel_initializer	-	'glorot_uniform'	'glorot_uniform'	'glorot_uniform'
5	bias_initializer	-	'zeros'	'zeros'	'zeros'
6	kernel_regularizer	-	None	None	None
7	bias_regularizer	-	None	None	None
8	activity_regularizer	-	None	None	None
9	kernel_constraint	-	None	None	None
10	bias_constraint	-	None	None	None

TABLE II
FINAL ERRORS(FOR VEHICLE 595)

	Prediction Horizon										
	1s	2s	3s	4s	5s	6s	7s	8s	9s	10s	20s
Quantity	Error										
$\epsilon_{x_{targ}}(m)$	5.11	7.09	7.77	8.18	8.77	8.99	9.23	9.40	9.57	9.66	10.10
$\epsilon_{v_{y_{targ}}}(m/s)$	-0.65	-0.47	-0.40	-0.36	-0.32	-0.30	-0.28	-0.27	-0.26	-0.25	-0.19
$\epsilon_{x_{targ_{modified}}}(m)$	-5.32e-16	2.22e-16	-2.22e-15	3.77e-15	-3.20e-15	1.14e-15	-2.23e-15	1.48e-15	-2.65e-15	-1.64e-15	-2.77e-17
$\epsilon_{v_{y_{targ_{modified}}}}(m/s)$	1.73e-14	9.76e-15	-5.18e-15	5.32e-15	1.49e-14	-2.81e-15	-2.41e-15	1.04e-14	1.33e-15	-2.79e-15	-5.66e-15



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 1500, 256)         312320
_________________________________________________________________
dense_1 (Dense)              (None, 1500, 256)         65792
_________________________________________________________________
dense_2 (Dense)              (None, 1500, 128)         32896
_________________________________________________________________
dense_3 (Dense)              (None, 1500, 2)           258
=================================================================
Total params: 411,266
Trainable params: 411,266
Non-trainable params: 0
_________________________________________________________________
None
```

Fig. 8. Model Definition.

- Training Data-set: Data of first 540 vehicles.
- Cross Validation Data-set: Data of next 50 Vehicles.
- Test Data-set: Data of last 10 vehicles.

The data of Cross Validation and test data-set should never be fed to network for training.

After this the loss function is plotted for training data-set and cross-Validation set is plotted. And the plot is given in Figure 9. By seeing this figure it can be said that the loss function goes to almost zero with time, which is a very good behaviour of model. Here since loss is selected as Mean Squared Error, this diagram can also be seen as the Mean Squared Error variation with number of training steps.

VI. RESULTS

In this section, the results are explained. The prediction is done for 20 seconds. But it is possible to predict for 150 seconds with this model.

To visualize the results in actual units the output of the neural network must be denormalized. To do this following calculations and values(from Table III) are used:

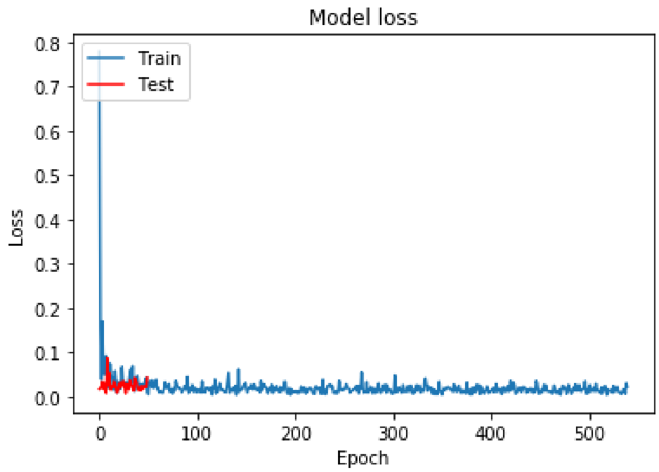


Fig. 9. Loss Function for Training(Train) and Cross Validation(Test) data-set.

TABLE III
VALUES USED FOR DENORMALIZATION(FOR VEHICLE 595)

S.No.	Quantity	Value
1	$x_{targ_{min}}$	-28.43 m
2	$x_{targ_{max}}$	30.33 m
3	$v_{y_{targ_{min}}}$	-13.38 m/s
4	$v_{y_{targ_{max}}}$	38.07 m/s

Actual values are calculated rom predictions using these formulas:

$$\epsilon_{x_{targ_{normalized}}} = \frac{1}{10T}\sum_{n=0}^{10T}\epsilon_{x_{targ}}$$
$$\epsilon_{v_{y_{targ_{normalized}}}} = \frac{1}{10T}\sum_{n=0}^{10T}\epsilon_{v_{y_{targ}}}$$
$$\epsilon_{x_{targ}} = x_{targ_{predicted}} - x_{targ_{given}}$$
$$\epsilon_{v_{y_{targ}}} = v_{y_{targ_{predicted}}} - v_{y_{targ_{given}}}$$
$$x_{targ_{actual}} = ((x_{targ_{normalized}} + \epsilon_{x_{targ_{normalized}}})* (x_{targ_{max}} - x_{targ_{min}}) + x_{targ_{min}})$$
$$v_{y_{targ_{actual}}} = ((v_{y_{targ_{normalized}}} + \epsilon_{v_{y_{targ_{normalized}}}})* (v_{y_{targ_{max}}} - v_{y_{targ_{min}}}) + v_{y_{targ_{min}}})$$
$$\epsilon_{x_{targ_{modified}}} = x_{targ_{actual}} - x_{targ_{given}}$$
$$\epsilon_{v_{y_{targ_{modified}}}} = v_{y_{targ_{actual}}} - v_{y_{targ_{given}}}$$

As visible in calculation by adding the average error value in the prediction the actual error remaining is very less, which is given in Table II. Table II shows the actual and modified errors with increasing prediction horizons.

As we can see in Figure 1 and 2 the accuracy level is very high.Due to initialization with minimum values it takes almost 1 second for the predictions to reach minimum amount of accuracy after that the difference between signals are very less. Since, the velocity value are calculated from position values. An Initial assumption of velocity value equal to 0, at first instance is taken, so the initial increase is visible in figure 2 till t= 1 sec. It takes almost 1 second for the prediction to reach very near to actual values.

In Figure 1 as we can see that the predicted lateral position is almost same as the actual values, Also to visualize the maneuvers y axis is divided based on lanes so that the lane changing maneuver can be visualized.

In figure 2 we can see that the actual velocity of the vehicle and the predicted velocity are almost same. Also due to filtering the little changes in velocity is removed but still the the network predicts the velocity around the actual value with almost zero error.

REFERENCES

[1] Florent Altché and Arnaud de La Fortelle (2017) *"An LSTM Network for Highway Trajectory Prediction"*, IEEE.

[2] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi*, Hanyang University, Seoul, Korea (2017) *"Probabilistic Vehicle Trajectory Prediction over Occupancy Grid Map via Recurrent Neural Network"*, IEEE.

[3] S. Lefèvre, D. Vasquez, and C. Laugier, *"A survey on motion prediction and risk assessment for intelligent vehicles,"* ROBOMECH Journal, vol. 1, no. 1, p. 1, dec 2014.

[4] C. Tay, K. Mekhnacha, and C. Laugier, *"Probabilistic Vehicle Motion Modeling and Risk Estimation,"* in Handbook of Intelligent Vehicles. Springer London, 2012, pp. 1479–1516.

[5] T. Streubel and K. H. Hoffmann, *"Prediction of driver intended path at intersections,"* IEEE Intelligent Vehicles Symposium, Proceedings, pp. 134–139, 2014.

[6] A. Carvalho, Y. Gao, S. Lefevre, and F. Borrelli, *"Stochastic predictive control of autonomous vehicles in uncertain environments,"* in 12th International Symposium on Advanced Vehicle Control, 2014.

[7] H. M. Mandalia and M. D. D. Salvucci, *"Using Support Vector Machines for Lane-Change Detection,"* Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 49, no. 22, pp. 1965–1969, sep 2005.

[8] 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). IEEE, jun 2013, pp. 797–802.

[9] A. Houenou, P. Bonnifait, V. Cherfaoui, and Wen Yao, *"Vehicle trajectory prediction based on motion model and maneuver recognition,"* in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, nov 2013, pp. 4363–4369.

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

[11] A. Khosroshahi, E. Ohn-Bar, and M. M. Trivedi, *"Surround vehicles trajectory analysis with recurrent neural networks,"* in 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). IEEE, nov 2016, pp. 2267–2272.

[12] D. J. Phillips, T. A. Wheeler, and M. J. Kochenderfer, *"Generalizable Intention Prediction of Human Drivers at Intersections,"* 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 1665–1670, 2017.

[13] Assaad MOAWAD (2018) *"Neural networks and back-propagation explained in a simple way"*, https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e.

[14] Matt Mazur (2015) *"A Step by Step Backpropagation Example"*, https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example.

[15] Michael Nguyen (2018) *"Illustrated Guide to LSTM's and GRU's: A step by step explanation"*, https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[16] Shi Yan (2016) *"Understanding LSTM and its diagrams"*, https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714.

[17] Colah's Blog (2015) *"Understanding LSTM Networks"*, https://colah.github.io/posts/2015-08-Understanding-LSTMs.

[18] U.S. Federal Highway Administration. (2005) US Highway 101 dataset.

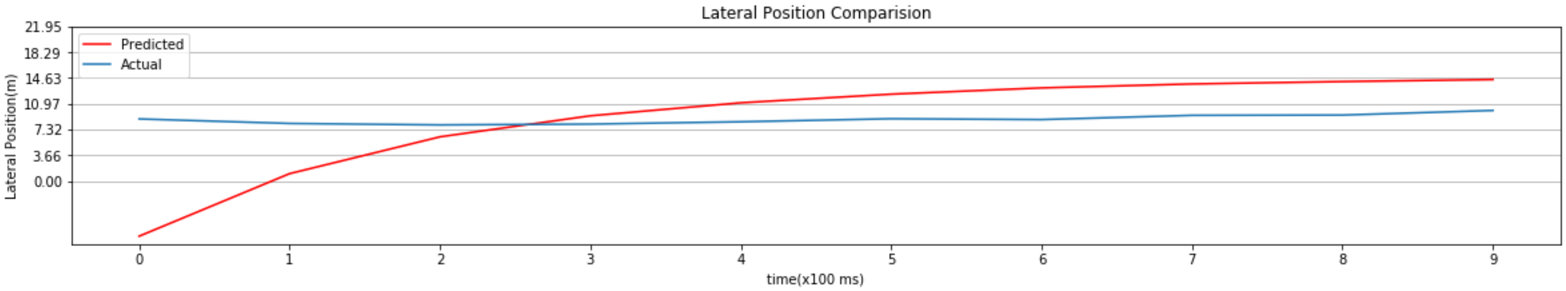


Fig. 10. Lateral Position for 1 Sec(Vehicle Number 595).

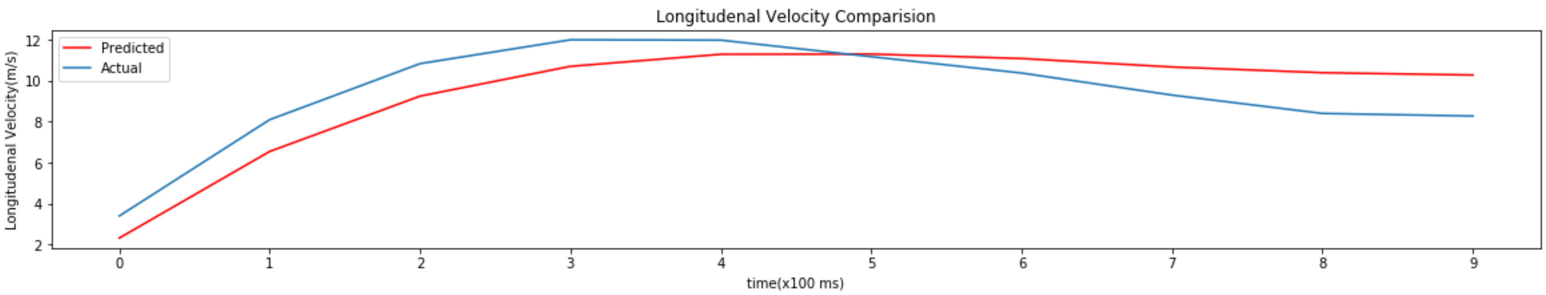


Fig. 11. Longitudinal Velocity for 1 Sec(Vehicle Number 595).

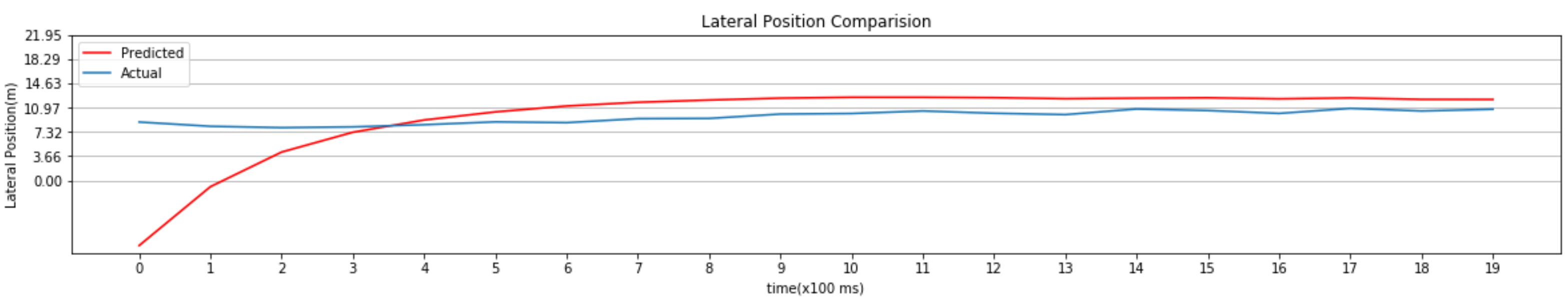


Fig. 12. Lateral Position for 2 Sec(Vehicle Number 595).

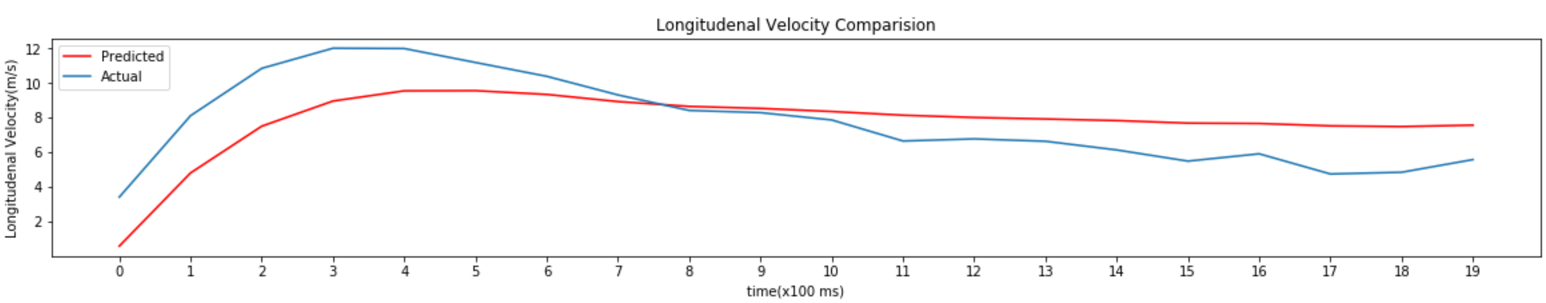


Fig. 13. Longitudinal Velocity for 2 Sec(Vehicle Number 595).

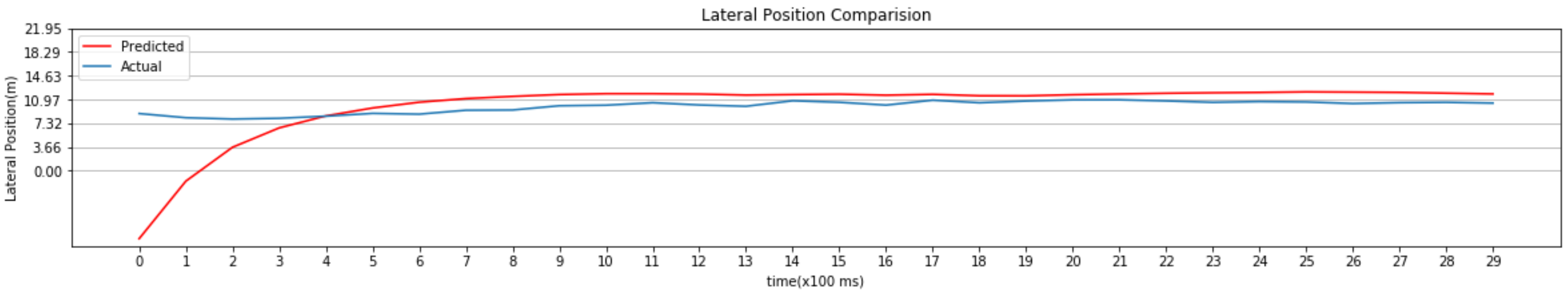


Fig. 14. Lateral Position for 3 Sec(Vehicle Number 595).

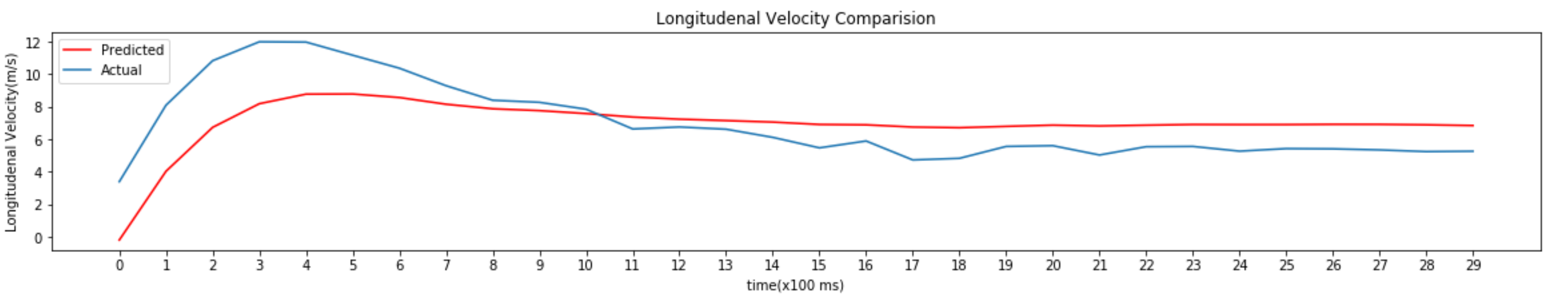


Fig. 15. Longitudinal Velocity for 3 Sec(Vehicle Number 595).

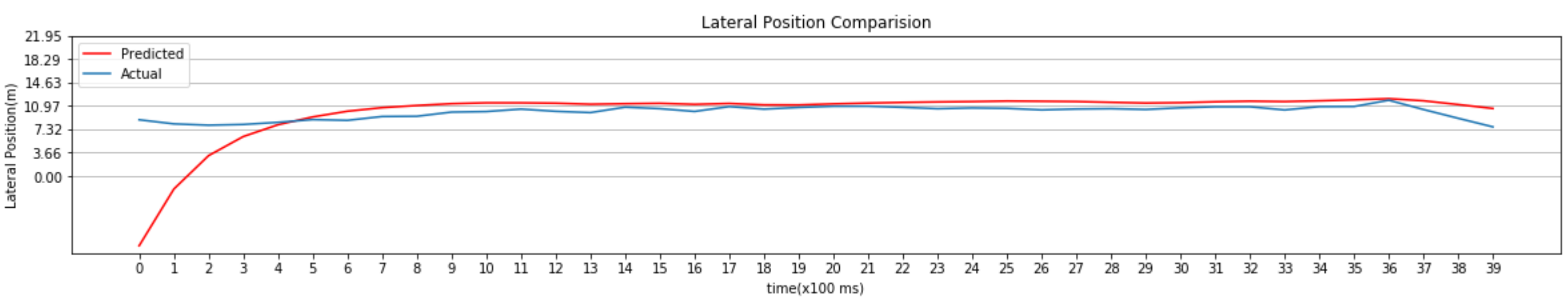


Fig. 16. Lateral Position for 4 Sec(Vehicle Number 595).

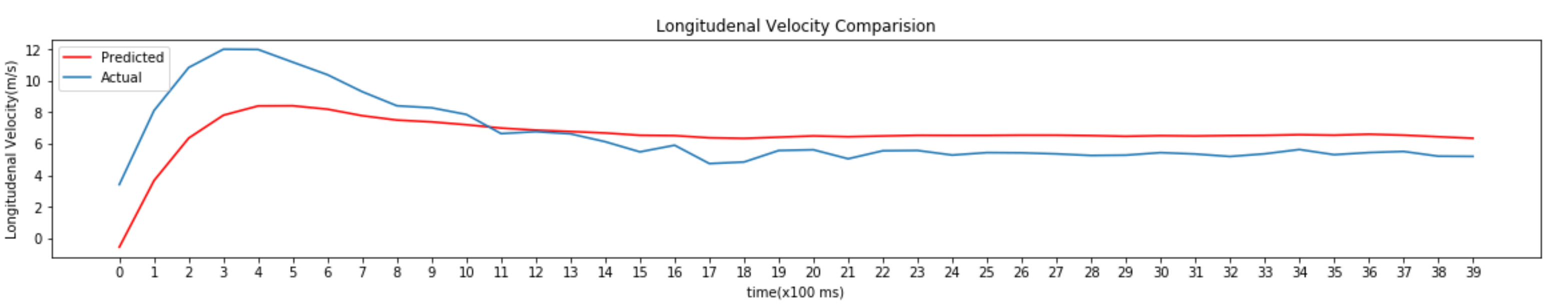


Fig. 17. Longitudinal Velocity for 4 Sec(Vehicle Number 595).

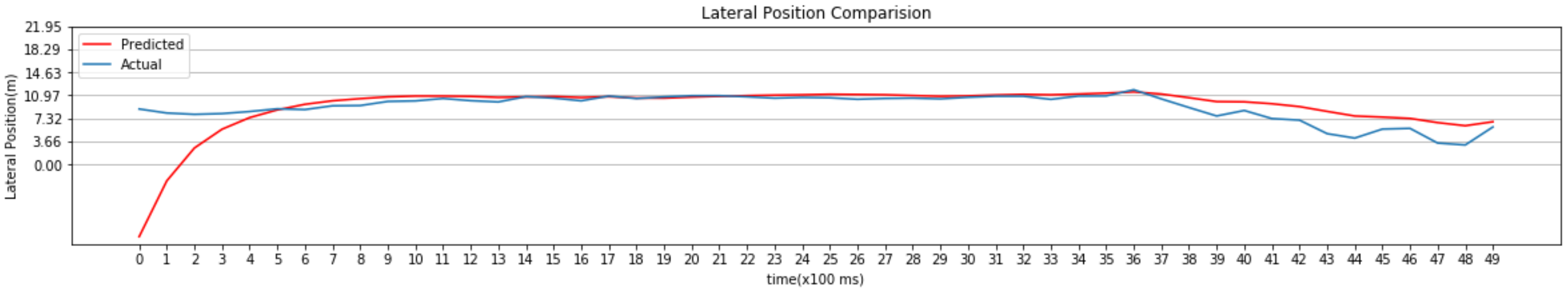


Fig. 18. Lateral Position for 5 Sec(Vehicle Number 595).

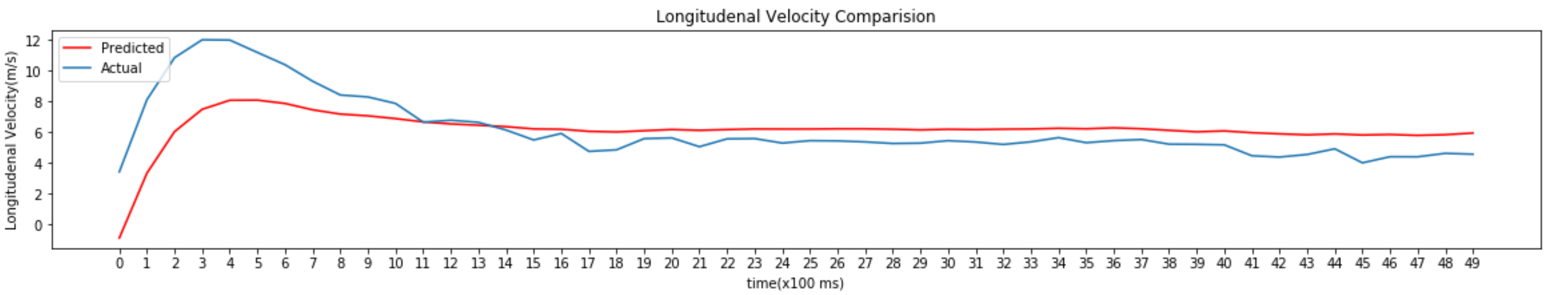


Fig. 19. Longitudinal Velocity for 5 Sec(Vehicle Number 595).

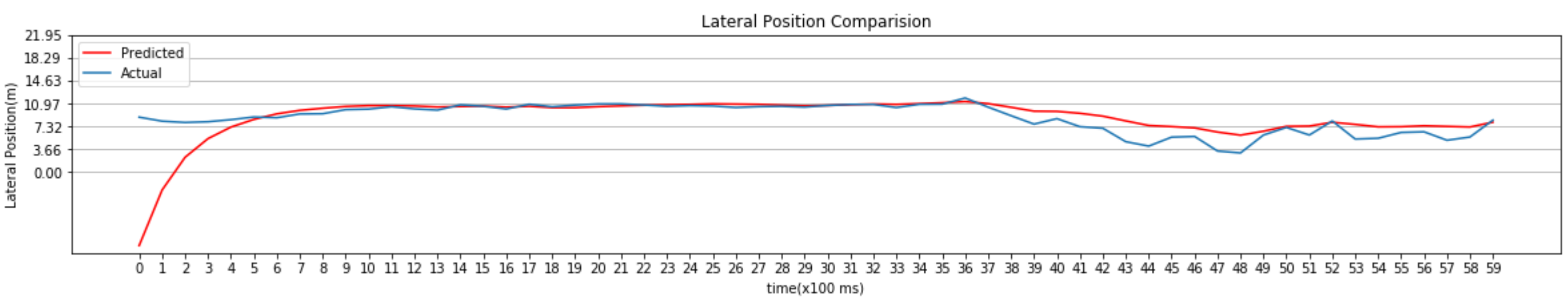


Fig. 20. Lateral Position for 6 Sec(Vehicle Number 595).

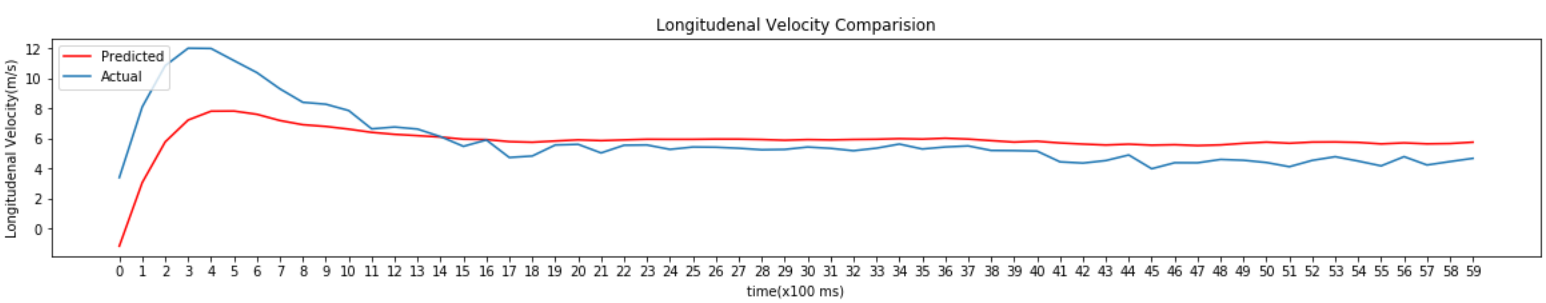


Fig. 21. Longitudinal Velocity for 6 Sec(Vehicle Number 595).

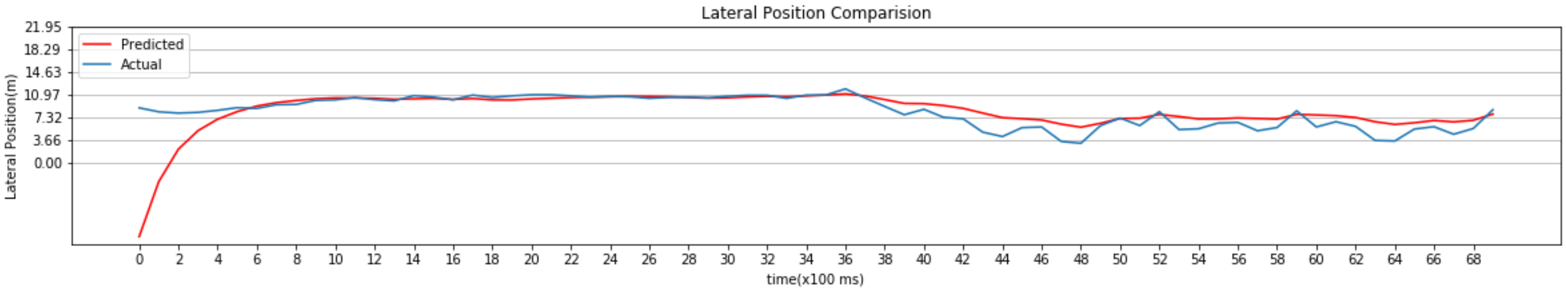


Fig. 22. Lateral Position for 7 Sec(Vehicle Number 595).

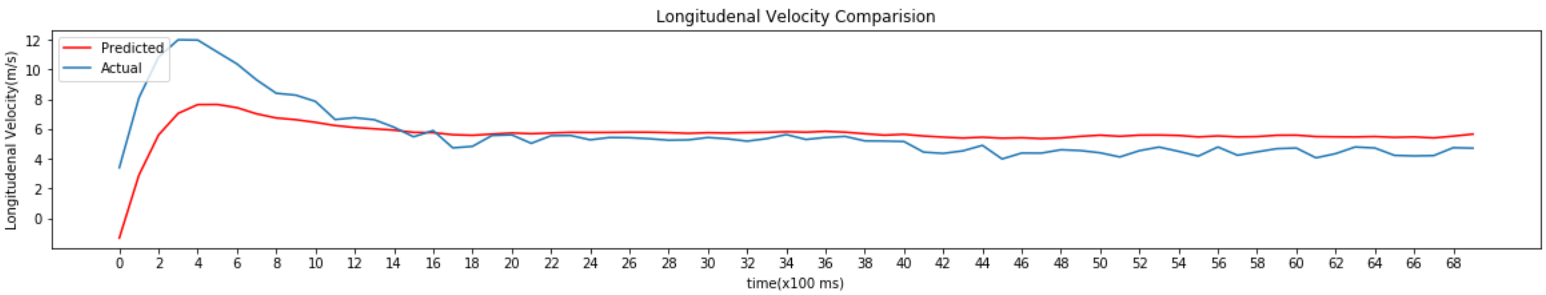


Fig. 23. Longitudinal Velocity for 7 Sec(Vehicle Number 595).

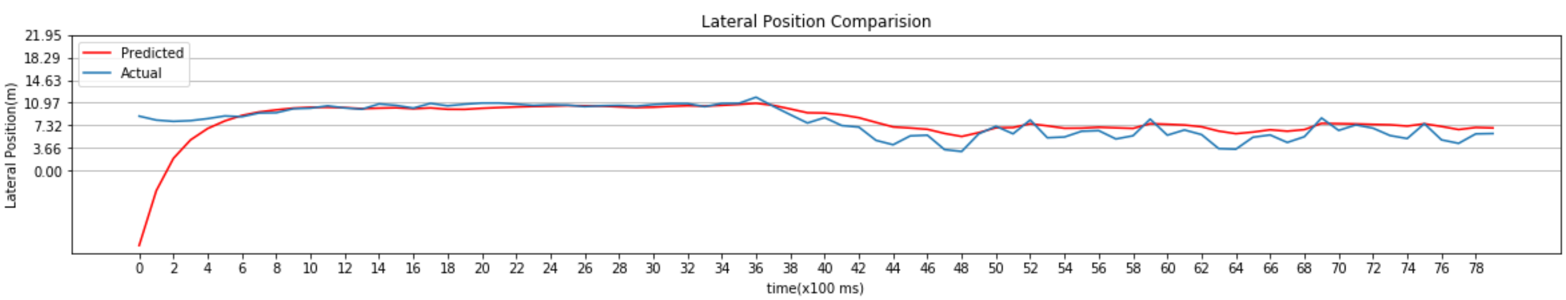


Fig. 24. Lateral Position for 7 Sec(Vehicle Number 595).

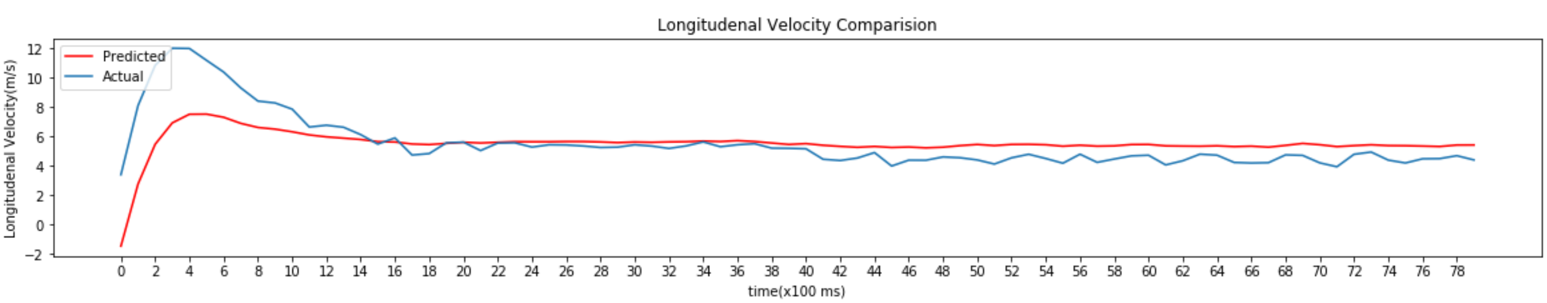


Fig. 25. Longitudinal Velocity for 7 Sec(Vehicle Number 595).

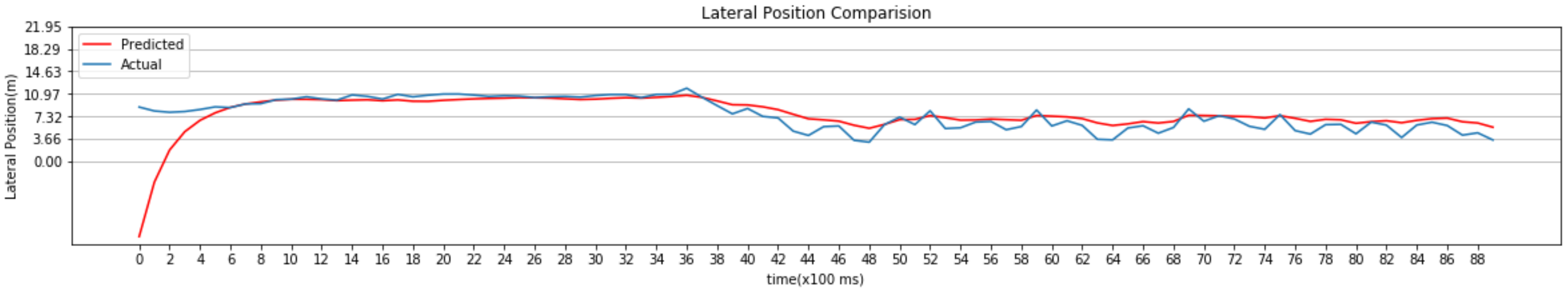


Fig. 26. Lateral Position for 7 Sec(Vehicle Number 595).

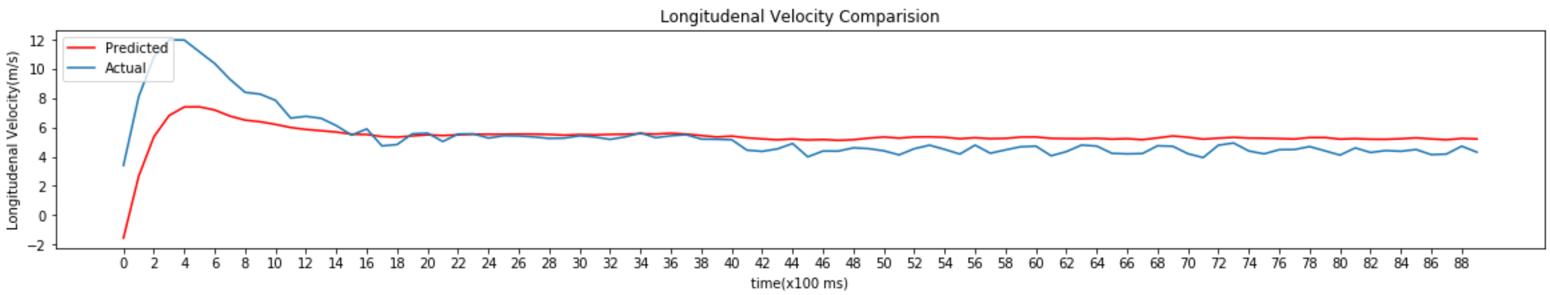


Fig. 27. Longitudinal Velocity for 7 Sec(Vehicle Number 595).

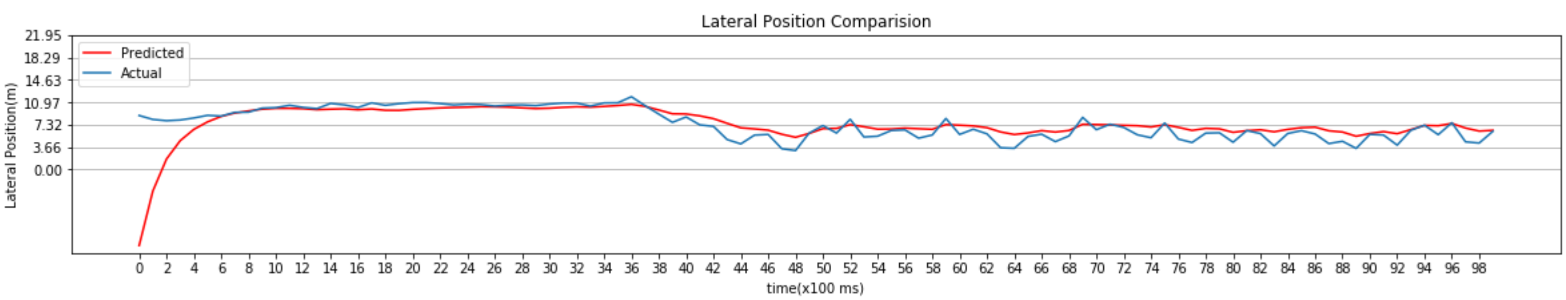


Fig. 28. Lateral Position for 7 Sec(Vehicle Number 595).

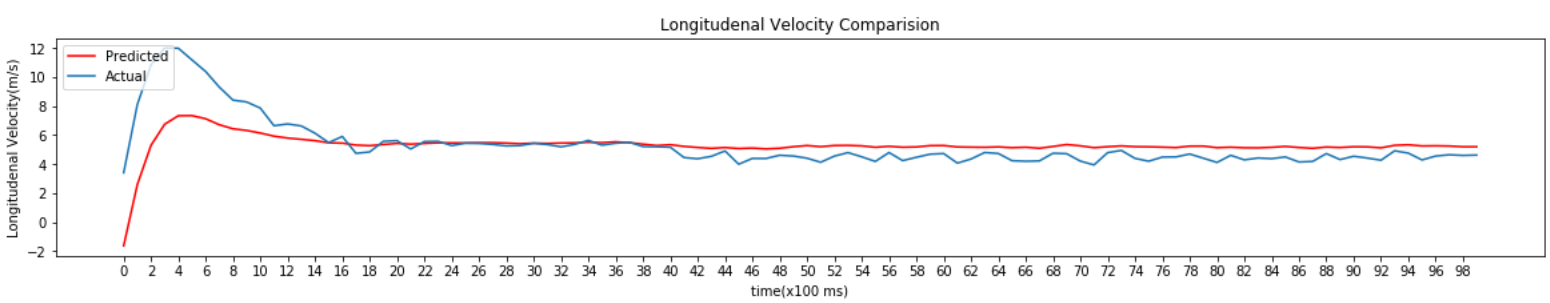


Fig. 29. Longitudinal Velocity for 7 Sec(Vehicle Number 595).