LSTM based trajectory prediction of exo vehicles

Ankit Khandelwal, Dr.-Ing. João Paulo C. L. da Costa
Department of Electronics and Infomation Technology
Technische Hochschule Ingolstadt
Ingolstadt, Germany
ankit.electrical01@gmail.com, joaopaulo.dacosta@ene.unb.br

***Abstract*— More than 90% of the traffic accidents are caused by humans. Therefore, the wide adoption of autonomous vehicles are expected to save hundreds of thousands of lives in the next decades. From the 3rd to the 5th level of the SAE J3016 standard for autonomous driving, the prediction of the trajectories of the surroundings vehicles, also known as exo vechiles, plays a crucial for the safe conduction of the autonomous ego vehicle. Generally prediction of the trajectories of exo vehicles is a highly non-linear problem including many possible candidate trajectories in the future for each exo vehicle. In this paper, we evaluate the performance of an Long Short-Term Memory (LSTM) Network in order to predict the trajectories of vehicles. We validate the approach using the data-set NGSIM US101 with highway measurements of more than 600 vehicles.**

I. Introduction

According to Fraunhofer Institute [13], 93,5% of the accidents caused by humans. Moreover, according to the Intel [21], between 2035 and 2045, 585,000 lives will be saved by self-driving vehicles. In the SAE J3016 standard [14] for autonomous driving, the monitoring of the driving environment us requred for the levels 3, 4 and 5, known as the Highly Automated Driving (HAD) or conditional automation, Fully Automated Driving (FAD) or High Automation, and Full Automation, respectively. By monitoring the driving environment, the autonomous vehicle should be able to predict the trajectories of its surrounding vehicles, also known as exo vehicles. The efficient and accurate trajectory prediction of exo vehicles is essential for the choice of a safe trajectory of the autonomous vehicle.

The trajectory prediction is a highly non-linear problem including many candidate trajectories for each exo vehicle. For the behaviour prediction, the following schemes have been considered in the literature, namely, hidden Markov models [4], [5], Kalman filtering [6], Support Vector Machines [7], [8] or directly using a vehicle model [9]; artificial neural network approaches [10]–[12].

In the previous study of trajectry prediction using LSTM networks[1] it is very evident that for time series trajectory prediction LSTM networks are one of the best networks. It gives highly accurate results and solves the complexity of trajectory prediction.

In this article, we will also focus on trajectory prediction using long short-term memory (LSTM) neural networks. Our main aim 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. We validate the LSTM based approach using data-set NGSIM US101 with highway measurements of more than 600 vehicles [20].

The rest of this article is structured as follows: Section II, defines the trajectory prediction problem Statement. Section III, details the approach used network selection and architecture used. Section IV, details the pre-processing of the US101 data-set to extract the input features of the model. Section V, shows the training procedure and outputs of the trained model. Section VI, shows the results of the study. Finally, Section VII concludes the study.

II. Problem Statement

Formally, Lets consider a set of observable features I and a set of target outputs O to be predicted. It is assumed that the features can all be acquired simultaneously at regular intervals,let $a \in I$ and $k = 0, 1, 2, ..., K$. a_k is the value of feature a observed k time steps earlier.

We use uppercase
$\hat{A}$= $[a_{ik}]$, for $a \in I$, $k = 0, 1, 2, ..., K$, $\hat{A} \in \mathbb{C}^{i \times k \times 1}$
$\hat{B}$ = $[b_{ik}]$, for $b \in O$, $k = 0, 1, 2, ..., K$, $\hat{B} \in \mathbb{C}^{j \times k \times 1}$
Here
i= Number of Input Features and j= Number of Output Feature
k= Number of time instances= (Total prediction time)x10
to respectively denote the tensors of the observed features and corresponding predicted outputs. Tensors are used because LSTM network works with tensors in time series prediction.

In section II-A, Feature selection strategy is described and in Section IV-E, details of output vectors are given.

A. Feature Selection

In this Section, 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 lane change intent. For a target vehicle, we consider 9 vehicles as shown in Figure 1.

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

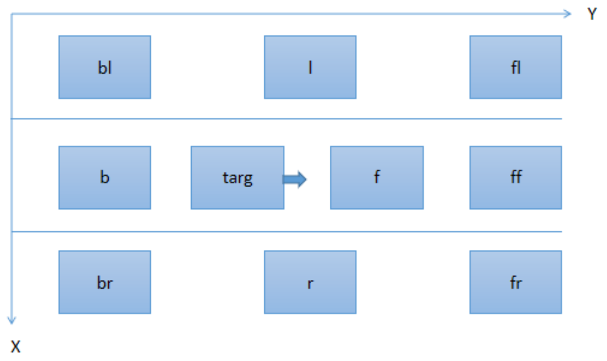


Fig. 1. Vehicle of interests around the Target Vehicle.

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.

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.

B. 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.

III. Network Selection and Architecture

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 the success of artificial neural network, in many applications, an artificial neural network is chosen for 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. 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.

In this article, we use the network presented in Figure 3 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:

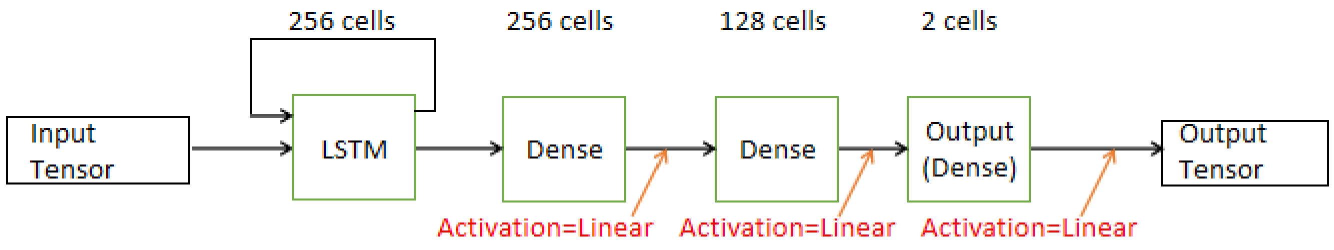
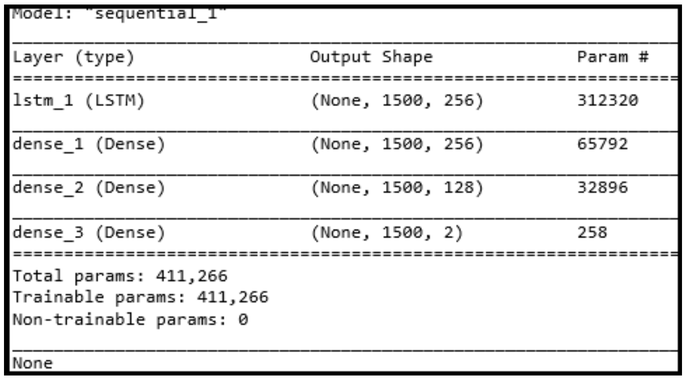


Fig. 2. Network Architecture.

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



```
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. 3. Model Definition.

IV. DATA-SET PRE-PROCESSING

In this Section, data pre-processing steps and feature selection procedure is described. In Section IV-A, Information about data set is given. In Scetion IV-B, data preparation using Savitzky-Golay filter is described. In Section IV-C, data set pre-processing steps are described. In Section IV-D, Target and surrounding vehicle selection procedure is described. Finally, in Section IV-E, Input and Output Tensor creation procedure is explained.

A. *Data-set*

In this article, we use the Next Generation Simulation (NGSIM) dataset [20], 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 4.

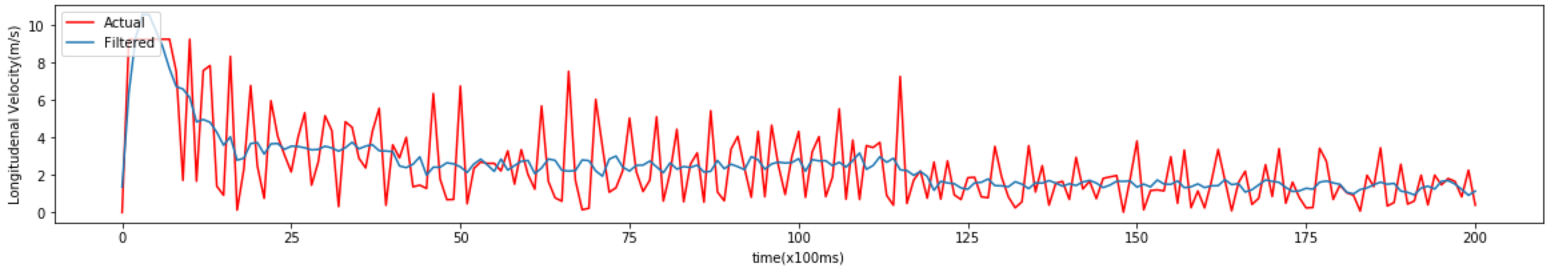


Fig. 4. Filtered output.

Figure 4 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 4 is intended to show just the actual and filtered value comparison and effectiveness of the filter used.

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, to keep only the data which is required to create feature vectors and delete all unused data. 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 period. To calculate actual time index has to be multiplied by 100ms.
- 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 IV-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 II-A.
- 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 3.

Now the final point is achieved the vehicles can be selected based on Figure 1

D. Selection of Target and Surrounding Vehicle

In this section, the procedure to create feature vector is explained. A feature Vector $\hat{A}$ is prepared by using data mentioned in Section IV. 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 $\hat{B}$ is created which contains lateral position and longitudinal velocity of the target vehicle. It can be done in following steps:

- Step1: Select randomly a target vehicle.
- Step2: Calculate the Lateral and Longitudinal distances of all other vehicle with target vehicle, provided they target vehicle should have a different lane for Lateral distance and should have a same lane for Longitudinal distance.
- Step3: For minimum positive Lateral and Longitudinal distances mark those vehicles as r and f respectively.
- Step4: For maximum negative Lateral and Longitudinal distances mark those vehicles as l and b respectively.
- Step5: Calculate the Longitudinal distances of all other vehicle with f vehicle provided they are in same lane.
- Step6: For minimum positive Longitudinal distances mark that vehicles as ff.
- Step7: Calculate the Longitudinal distances of all other vehicle with l and r vehicle provided they are in same lane.
- Step8: For minimum positive Longitudinal distances with l mark that vehicles as fl and for minimum positive Longitudinal distances with r mark that vehicles as fr.

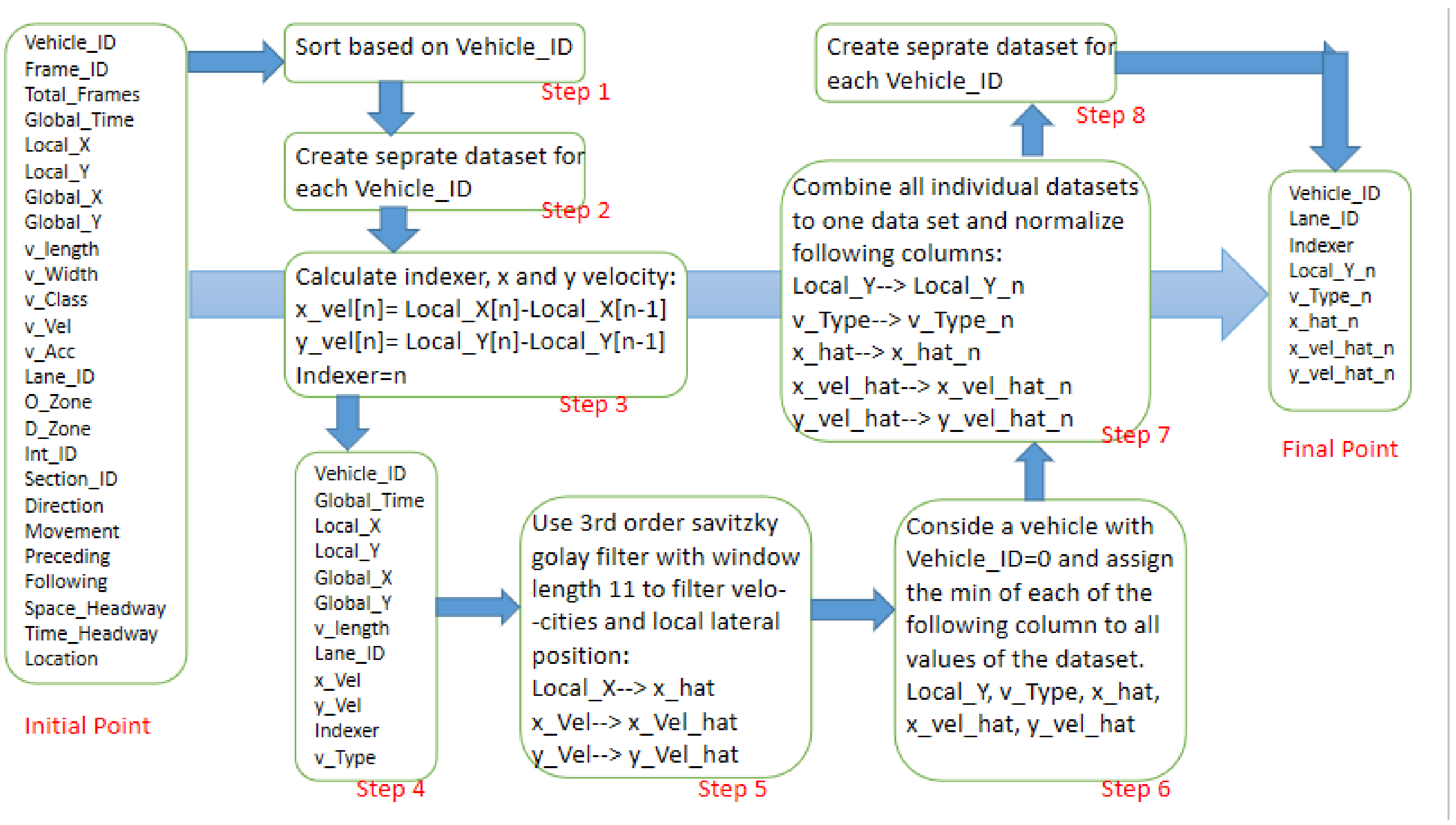


Fig. 5. Dataset Preprocessing.

- Step9: For maximum negative Longitudinal distances with l mark that vehicles as bl and for maximum negative Longitudinal distances with r mark that vehicles as br.

E. Input and Output Tensor Creation

Now all surrounding and target vehicle are selected. Next step is to create Feature vector which is done based on the steps mention in Section II-A. Feature matrix will look like this:

Input Feature Matrix:

$$\begin{aligned}\hat{A} = [&y_{targ_k}, v_{x_{targ_k}}, v_{type_{targ_k}},\\ &v_{x_{fl_k}}, \Delta v_{y_{fl_k}}, \Delta x_{fl_k}, \Delta y_{fl_k}, v_{type_{fl_k}},\\ &v_{x_{ff_k}}, \Delta v_{y_{ff_k}}, \Delta x_{ff_k}, \Delta y_{ff_k}, v_{type_{ff_k}},\\ &v_{x_{fr_k}}, \Delta v_{y_{fr_k}}, \Delta x_{fr_k}, \Delta y_{fr_k}, v_{type_{fr_k}},\\ &v_{x_{l_k}}, \Delta v_{y_{l_k}}, \Delta x_{l_k}, \Delta y_{l_k}, v_{type_{l_k}},\\ &v_{x_{f_k}}, \Delta v_{y_{f_k}}, \Delta x_{f_k}, \Delta y_{f_k}, v_{type_{f_k}},\\ &v_{x_{r_k}}, \Delta v_{y_{r_k}}, \Delta x_{r_k}, \Delta y_{r_k}, v_{type_r},\\ &v_{x_{bl_k}}, \Delta v_{y_{bl_k}}, \Delta x_{bl_k}, \Delta y_{bl_k}, v_{type_{bl_k}},\\ &v_{x_{b_k}}, \Delta v_{y_{b_k}}, \Delta x_{b_k}, \Delta y_{b_k}, v_{type_{b_k}},\\ &v_{x_{br_k}}, \Delta v_{y_{br_k}}, \Delta x_{br_k}, \Delta y_{br_k}, v_{type_{br_k}}]\end{aligned}$$

Output Matrix:

$\hat{B} = [x_{targ_k}, v_{y_{targ_k}}]$ $k = 0, 1, 2, ..., K$

Next step is to convert Matrix into Tensor with third dimension as 1 so that the size of Tensors will be $\hat{A} \in \mathbb{C}^{48\times200\times1}$ and $\hat{B} \in \mathbb{C}^{2\times200\times1}$ respectively.

V. TRAINING PROCEDURE

In this Section Training procedure is explained. 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:

- 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 6. 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 II) are used:

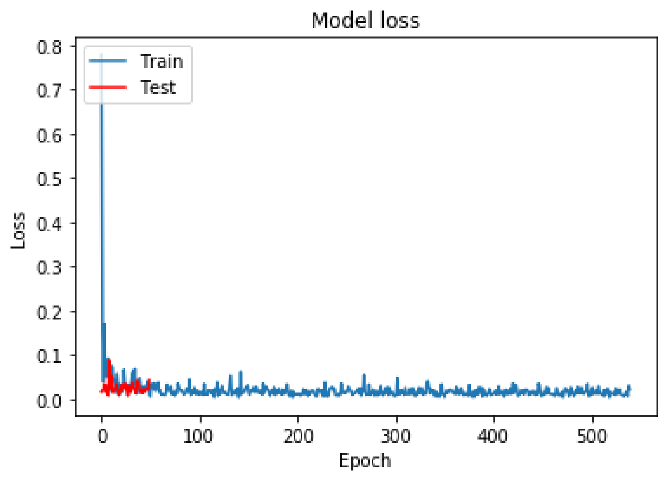


Fig. 6. Loss Function for Training(Train) and Cross Validation(Test) data-set.

TABLE II
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, absolute and modified errors are calculated from predictions using these formulas:

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

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 III. Table III shows the actual and modified errors with increasing prediction horizons.

As we can see in Figure 7 and 8 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 8 till t= 1 sec. It takes almost 1 second for the prediction to reach very near to actual values.

In Figure 7 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 8 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.

VII. CONCLUSION

From the above study it is clearly visible that the prediction using LSTM network is highly accurate once the static error is removed from the predicted values. Since out of 6000 vehicles given in US 101 NGSIM data[20], only 600 vehicles are used for this study, there are still chances of improvement in the results given by the network. But this will also increase the chances of over-fitting to the data during training. So, a trade-off between the data set required and accuracy must be done in such a way that the results achieved are most accurate and network uses minimum computational effort. But the results from this study with 6000 vehicles are accurate enough and shows results with improved accuracy as we increase the prediction horizon.

TABLE III
FINAL ERRORS(FOR VEHICLE 595)

	Prediction Horizon										
	1s	2s	3s	4s	5s	6s	7s	8s	9s	10s	20s
Quantity	Error										
$\epsilon_{x_{targ_{absolute}}}$ (m)	7.81	8.44	8.67	8.85	9.31	9.44	9.62	9.74	9.87	9.93	10.24
$\epsilon_{v_{y_{targ_{absolute}}}}$ (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)	5.08	3.21	2.32	1.84	1.64	1.47	1.43	1.38	1.36	1.31	1.04
$\epsilon_{v_{y_{targ_{modified}}}}$ (m/s)	1.23	1.80	1.69	1.47	1.34	1.23	1.14	1.05	0.98	0.91	0.72

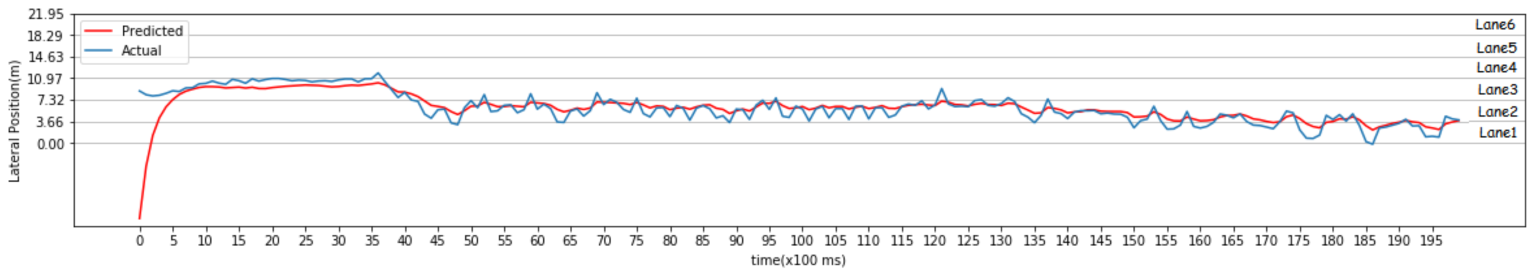


Fig. 7. Lateral Position(Vehicle Number 595).

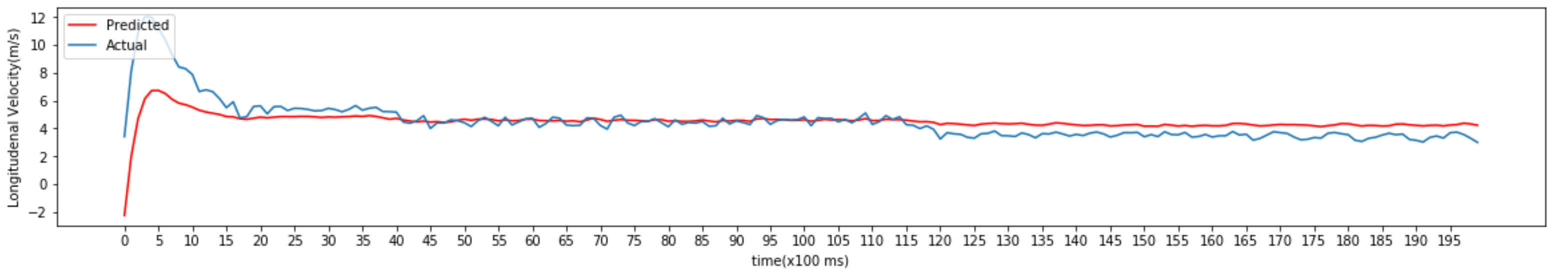


Fig. 8. Longitudinal Velocity(Vehicle Number 595).

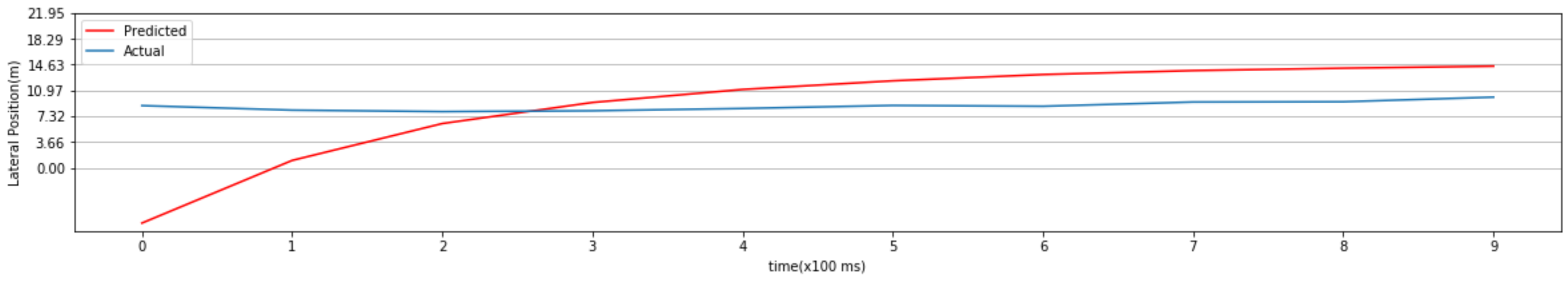


Fig. 9. Lateral Position for 1 Sec(Vehicle Number 595).

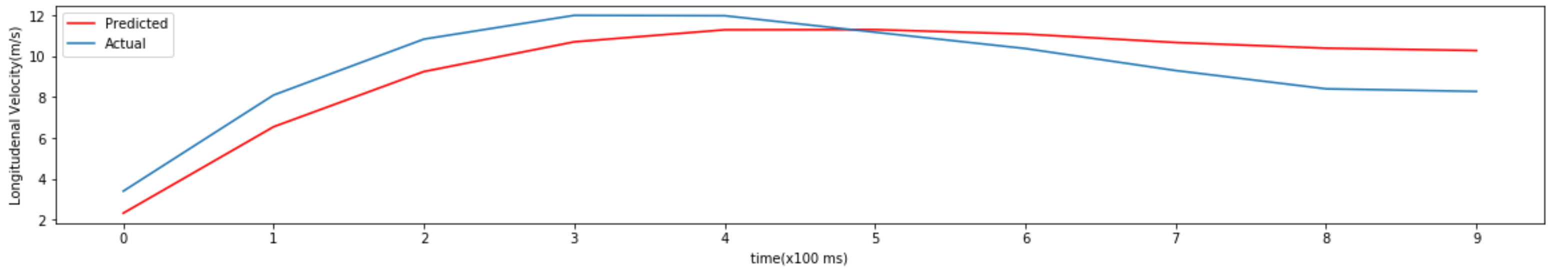


Fig. 10. Longitudinal Velocity for 1 Sec(Vehicle Number 595).

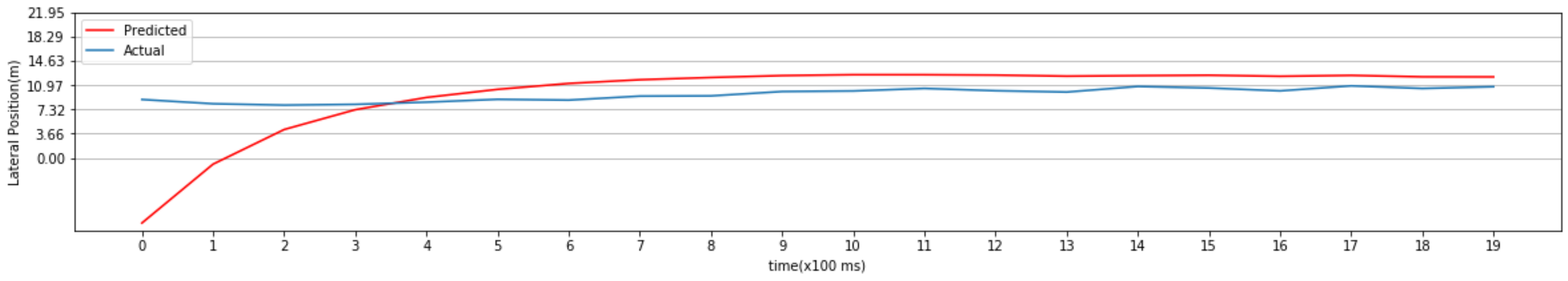


Fig. 11. Lateral Position for 2 Sec(Vehicle Number 595).

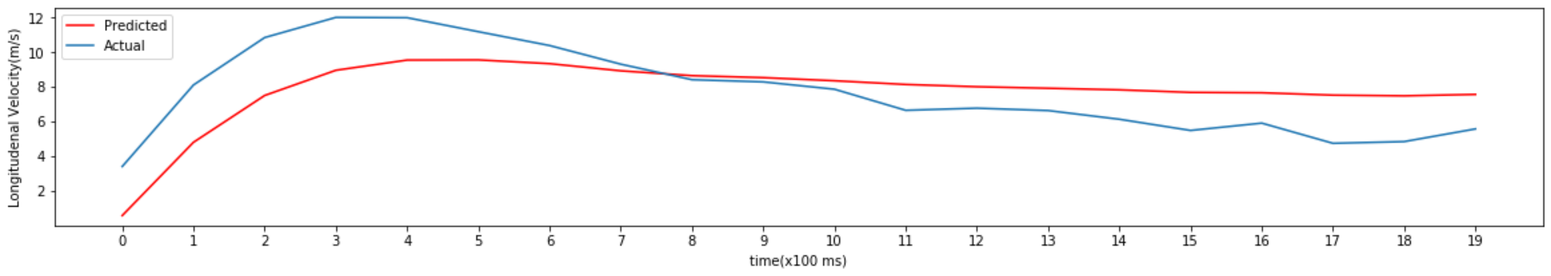


Fig. 12. Longitudinal Velocity for 2 Sec(Vehicle Number 595).

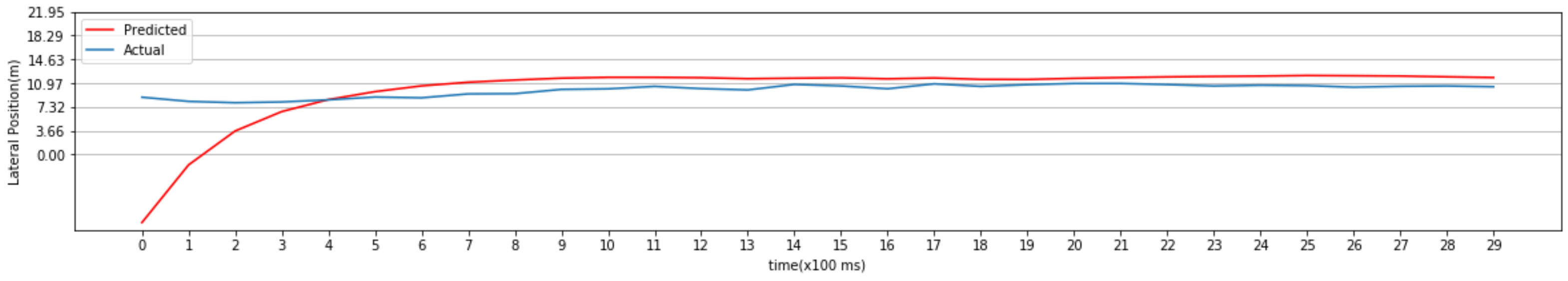


Fig. 13. Lateral Position for 3 Sec(Vehicle Number 595).

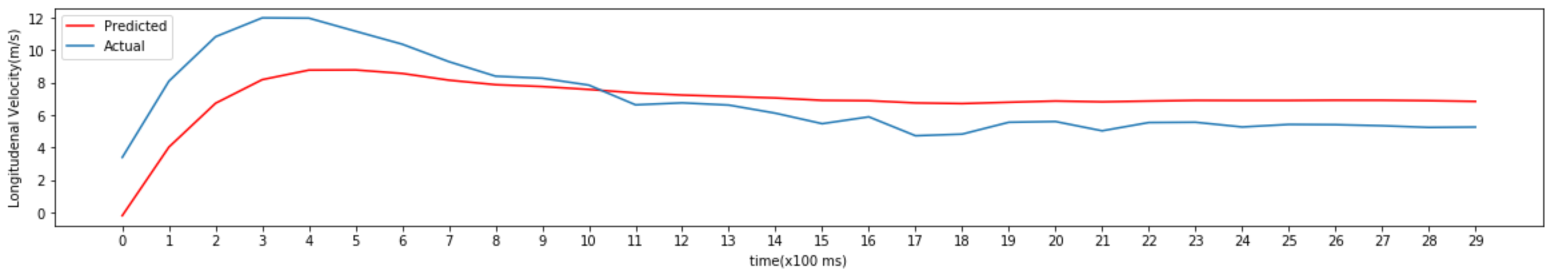


Fig. 14. Longitudinal Velocity for 3 Sec(Vehicle Number 595).

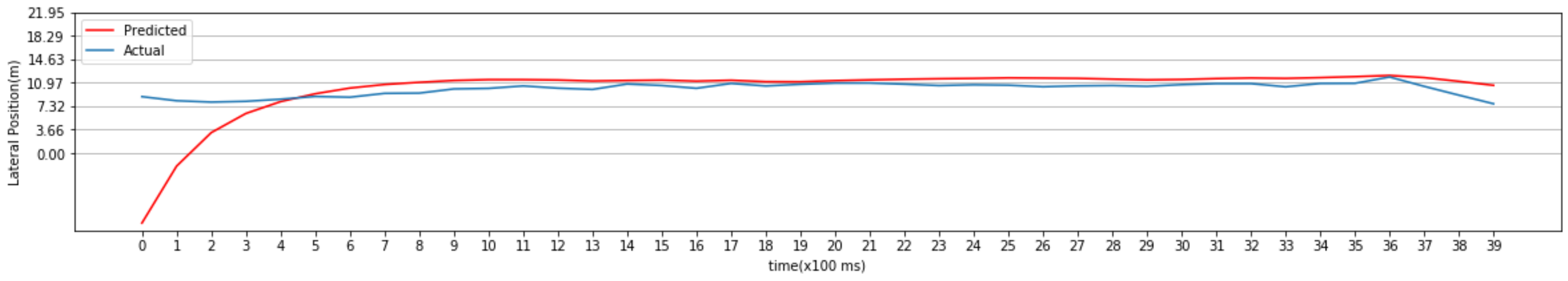


Fig. 15. Lateral Position for 4 Sec(Vehicle Number 595).

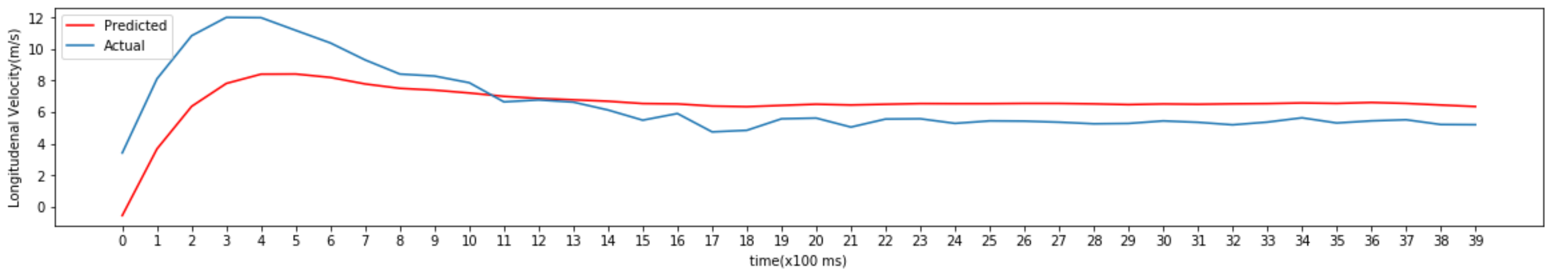


Fig. 16. Longitudinal Velocity for 4 Sec(Vehicle Number 595).

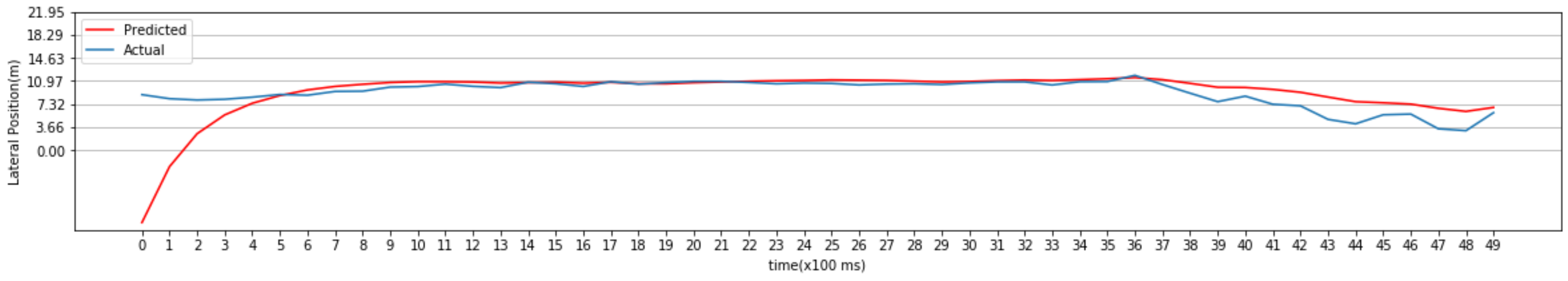


Fig. 17. Lateral Position for 5 Sec(Vehicle Number 595).

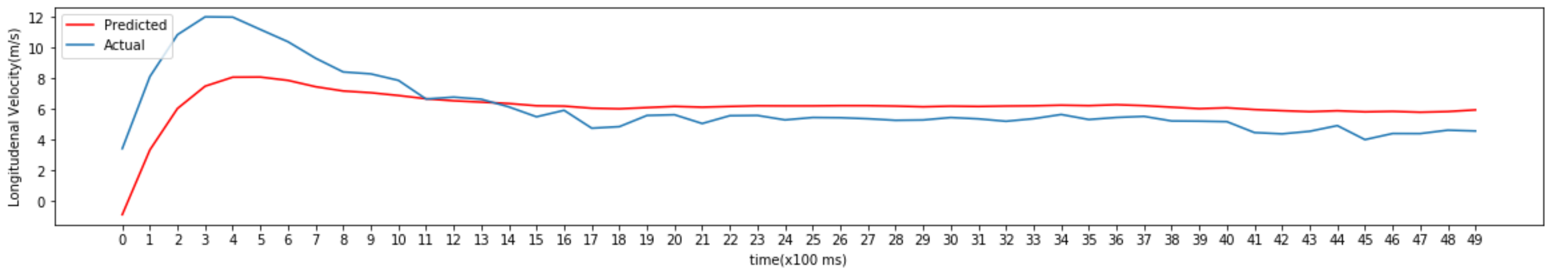


Fig. 18. Longitudinal Velocity for 5 Sec(Vehicle Number 595).

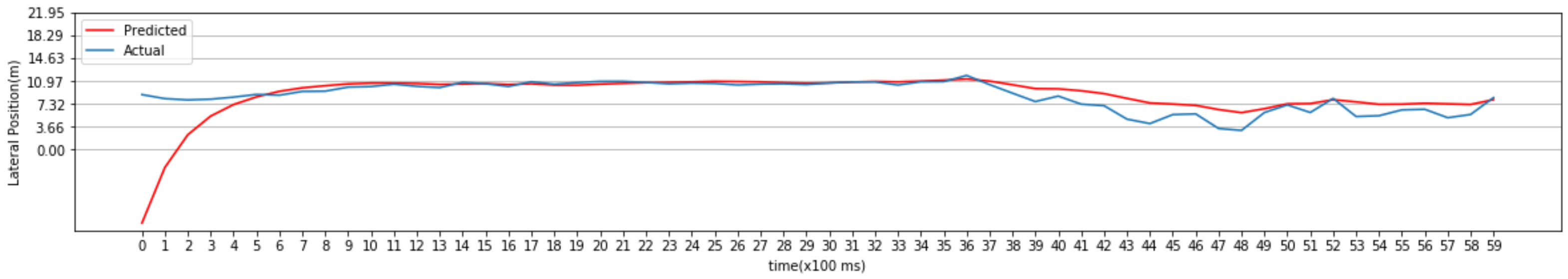


Fig. 19. Lateral Position for 6 Sec(Vehicle Number 595).

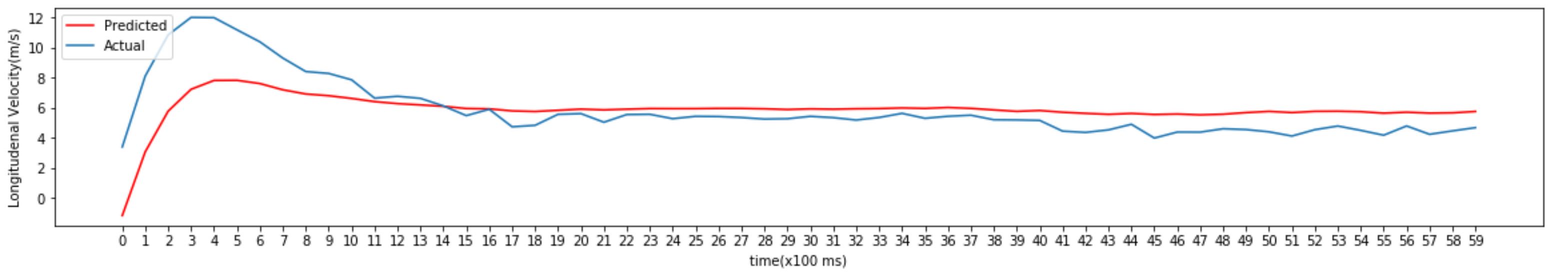


Fig. 20. Longitudinal Velocity for 6 Sec(Vehicle Number 595).

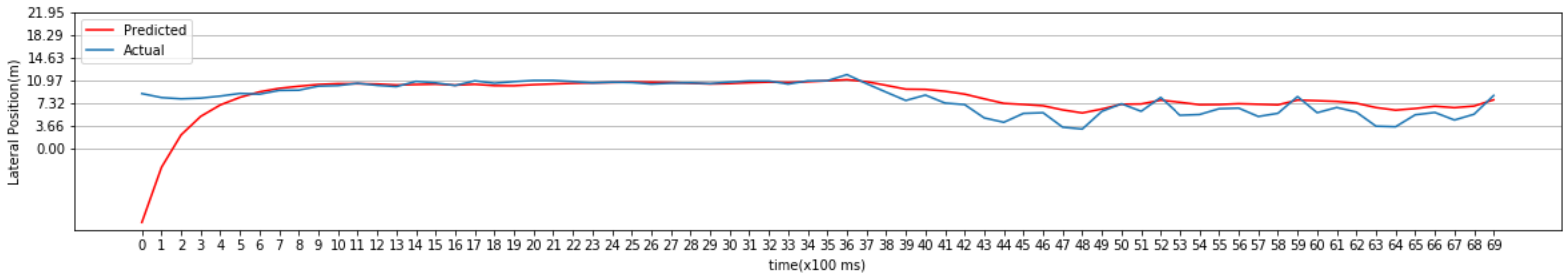


Fig. 21. Lateral Position for 7 Sec(Vehicle Number 595).

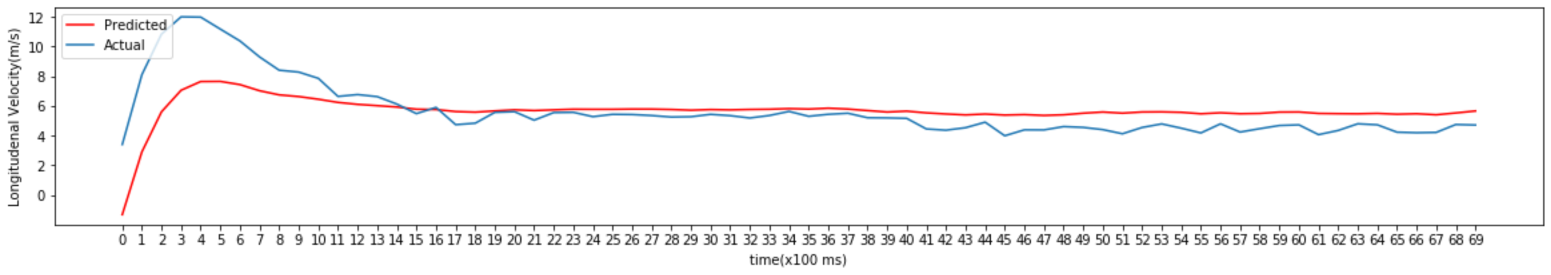


Fig. 22. Longitudinal Velocity for 7 Sec(Vehicle Number 595).

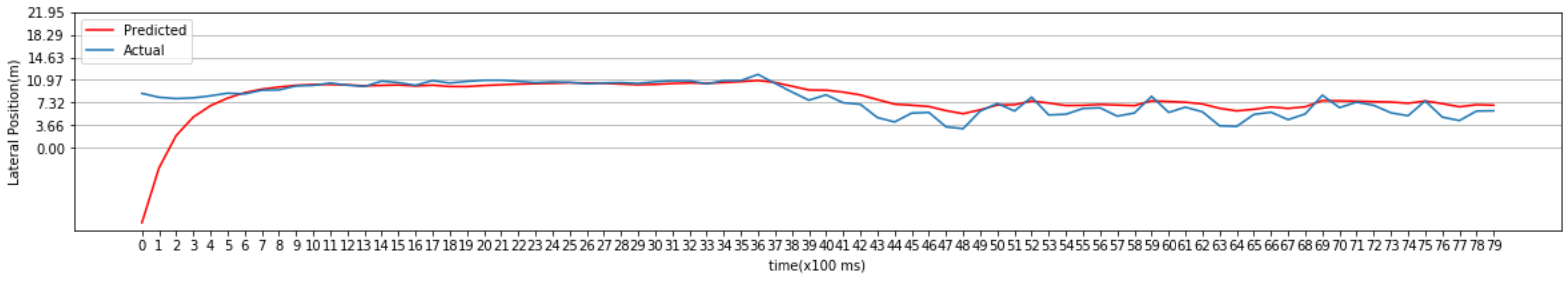


Fig. 23. Lateral Position for 7 Sec(Vehicle Number 595).

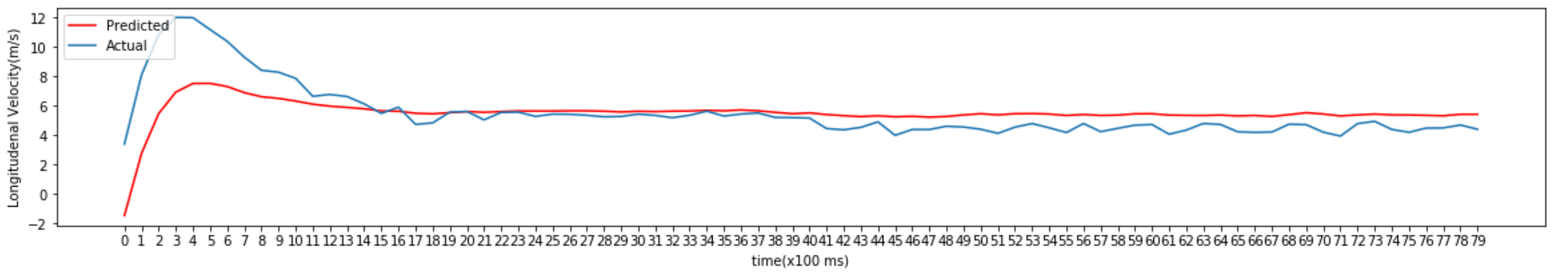


Fig. 24. Longitudinal Velocity for 7 Sec(Vehicle Number 595).

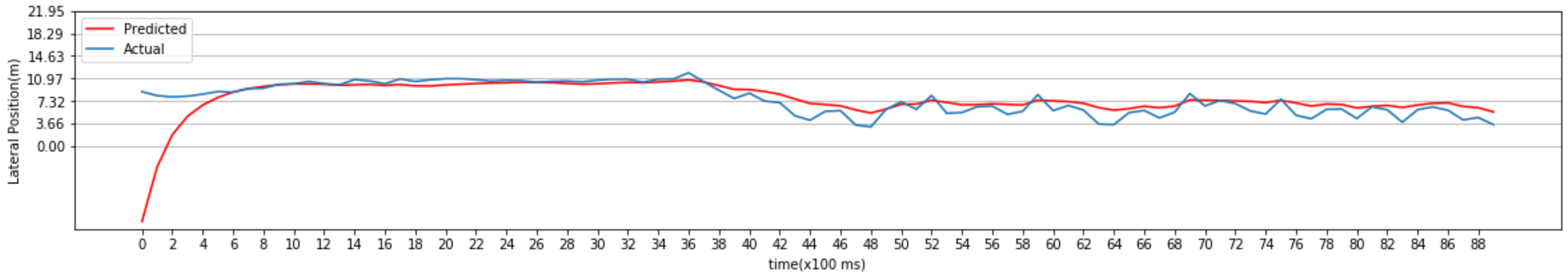


Fig. 25. Lateral Position for 7 Sec(Vehicle Number 595).

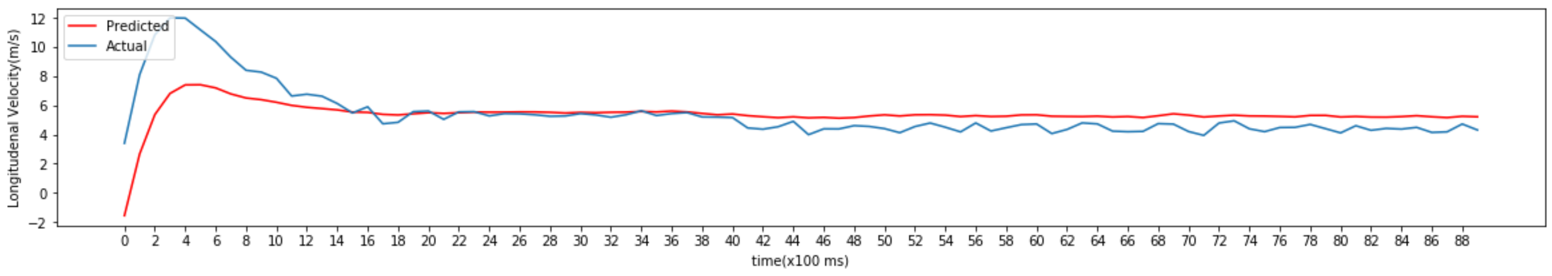


Fig. 26. Longitudinal Velocity for 7 Sec(Vehicle Number 595).

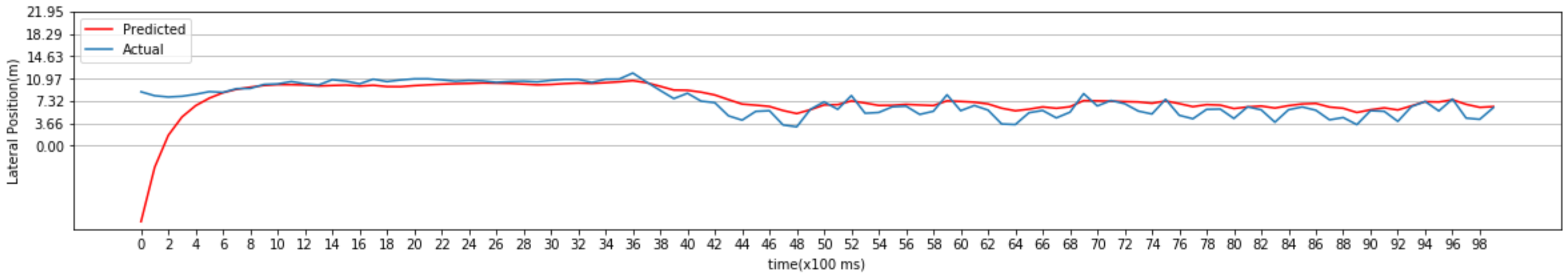


Fig. 27. Lateral Position for 7 Sec(Vehicle Number 595).

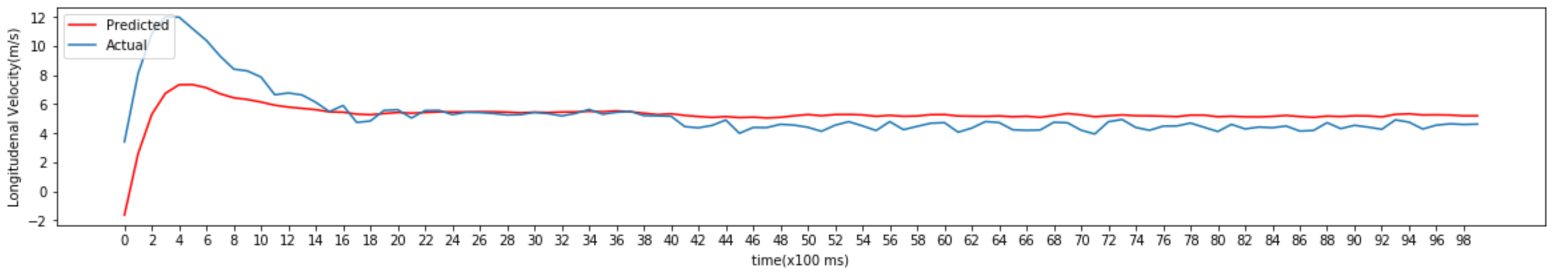


Fig. 28. Longitudinal Velocity for 7 Sec(Vehicle Number 595).

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] A. Cacilo, S. Schmidt, P. Wittlinger, F. Herrmann, W. Bauer, O. Sawade, H. O. Doderer, M. Hartwig and V. Scholz. *”Hochautomatisiertes Fahren auf Autobahnen - Industriepolitische Schlussfolgerungen.”* Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO), Nov. 2015.

[14] Grubmüller, Stephanie and Plihal, Jiri and Nedoma, Pavel. *”Automated Driving from the View of Technical Standards.”* Automated driving, 2017,

[15] 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.

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

[17] 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.

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

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

[20] U.S. Federal Highway Administration. (2005) *US Highway 101 dataset.* https://www.fhwa.dot.gov/publications/research/operations/07030/index.cfm

[21] Intel Report (2020) *”Intel’s report on Autonomous driving”* https://newsroom.intel.com/news/latest-intel-study-finds-people-expect-self-driving-cars-common-50-years